

---

# Vector Intrinsic Manual

*Release 1.1*

THEAD

Apr 15, 2020

THEAD

# CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Vector types . . . . .	1
1.2	Mask types . . . . .	2
1.3	Immediate operand . . . . .	3
1.4	Memory address operand . . . . .	3
1.5	Calling Convention . . . . .	3
1.6	Supported CPUs . . . . .	3
<b>2</b>	<b>Reference</b>	<b>5</b>
2.1	<b>Vector configuration</b> . . . . .	<b>5</b>
2.1.1	Change the granted vector length by vtype . . . . .	5
2.1.2	Change the granted vector length . . . . .	5
2.2	<b>Bit manipulation</b> . . . . .	<b>5</b>
2.2.1	Elementwise vector-immediate bitwise-and . . . . .	5
2.2.2	Elementwise vector-vector bitwise-and . . . . .	8
2.2.3	Elementwise vector-scalar bitwise-and . . . . .	11
2.2.4	Elementwise vector bitwise-not . . . . .	13
2.2.5	Elementwise vector-immediate bitwise-or . . . . .	16
2.2.6	Elementwise vector-vector bitwise-or . . . . .	19
2.2.7	Elementwise vector-scalar bitwise-or . . . . .	21
2.2.8	Elementwise vector-immediate logic shift left . . . . .	24
2.2.9	Elementwise vector-vector logic shift left . . . . .	27
2.2.10	Elementwise vector-scalar logic shift left . . . . .	30
2.2.11	Elementwise vector-immediate arithmetic shift right . . . . .	32
2.2.12	Elementwise vector-vector arithmetic shift right . . . . .	34
2.2.13	Elementwise vector-scalar arithmetic shift right . . . . .	36
2.2.14	Elementwise vector-immediate logic shift right . . . . .	37
2.2.15	Elementwise vector-vector logic shift right . . . . .	39
2.2.16	Elementwise vector-scalar logic shift right . . . . .	40
2.2.17	Elementwise vector-immediate bitwise-xor . . . . .	42
2.2.18	Elementwise vector-vector bitwise-xor . . . . .	45
2.2.19	Elementwise vector-scalar bitwise-xor . . . . .	47
2.3	<b>Conversions between floating-point vectors</b> . . . . .	<b>50</b>
2.3.1	Convert double-width floating-point to current-width . . . . .	50
2.3.2	Convert current-width floating-point to double-width . . . . .	51
2.4	<b>Conversions between integer and floating-point vector</b> . . . . .	<b>52</b>
2.4.1	Convert interger to floating-point . . . . .	52
2.4.2	Convert unsigned interger to floating-point . . . . .	53
2.4.3	Convert floating-point to interger . . . . .	54
2.4.4	Convert floating-point to unsigned interger . . . . .	55

2.4.5	Convert double-width interger to floating-point . . . . .	57
2.4.6	Convert double-width unsigned interger to floating-point . . . . .	58
2.4.7	Convert double-width floating-point to interger . . . . .	58
2.4.8	Convert double-width floating-point to unsigned interger . . . . .	59
2.4.9	Convert interger to double-width floating-point . . . . .	60
2.4.10	Convert unsigned interger to double-width floating-point . . . . .	61
2.4.11	Convert floating-point to double-width integer . . . . .	63
2.4.12	Convert floating-point to double-width unsigned integer . . . . .	63
2.5	<b>Floating-point arithmetic operations . . . . .</b>	64
2.5.1	Elementwise vector-scalar floating-point addition . . . . .	64
2.5.2	Elementwise vector-vector floating-point addition . . . . .	66
2.5.3	Classify every floating-point element as the saclar classify instruction do . . . . .	67
2.5.4	Elementwise vector-scalar floating-point division . . . . .	68
2.5.5	Elementwise vector-vector floating-point division . . . . .	69
2.5.6	Elementwise vector-vector floating-point dot-product . . . . .	71
2.5.7	Floating-point vector-scalar multiply and add(overwrite addend) . . . . .	72
2.5.8	Floating-point vector-vector multiply and add(overwrite addend) . . . . .	73
2.5.9	Floating-point vector-scalar multiply and add(overwrite multiplicand) . . . . .	74
2.5.10	Floating-point vector-vector multiply and add(overwrite multiplicand) . . . . .	75
2.5.11	Elementwise vector-scalar floating-point maximum . . . . .	77
2.5.12	Elementwise vector-vector floating-point maximum . . . . .	78
2.5.13	Elementwise vector-scalar floating-point minimum . . . . .	79
2.5.14	Elementwise vector-vector floating-point minimum . . . . .	81
2.5.15	Floating-point vector-scalar multiply and sub(overwrite subtrahend) . . . . .	82
2.5.16	Floating-point vector-vector multiply and sub(overwrite subtrahend) . . . . .	83
2.5.17	Floating-point vector-scalar multiply and sub(overwrite multiplicand) . . . . .	84
2.5.18	Floating-point vector-vector multiply and sub(overwrite multiplicand) . . . . .	85
2.5.19	Elementwise vector-scalar floating-point multiplication . . . . .	87
2.5.20	Elementwise vector-vector floating-point multiplication . . . . .	88
2.5.21	Floating-point vector-scalar negate multiply and add(overwrite addend) . . . . .	89
2.5.22	Floating-point vector-vector negate multiply and add(overwrite addend) . . . . .	90
2.5.23	Floating-point vector-scalar negate multiply and add(overwrite multiplicand) . . . . .	92
2.5.24	Floating-point vector-vector negate multiply and add(overwrite multiplicand) . . . . .	93
2.5.25	Floating-point vector-scalar negate multiply and sub(overwrite subtrahend) . . . . .	94
2.5.26	Floating-point vector-vector negate multiply and sub(overwrite subtrahend) . . . . .	95
2.5.27	Floating-point vector-scalar negate multiply and sub(overwrite multiplicand) . . . . .	97
2.5.28	Floating-point vector-vector negate multiply and sub(overwrite multiplicand) . . . . .	98
2.5.29	Elementwise vector-scalar floating-point reverse division . . . . .	99
2.5.30	Floating-point maximum of vector . . . . .	100
2.5.31	Floating-point minimum of vector . . . . .	102
2.5.32	Floating-point ordered sum of vector . . . . .	103
2.5.33	Floating-point sum of vector . . . . .	104
2.5.34	Elementwise vector-scalar floating-point reverse subtraction . . . . .	105
2.5.35	Elementwise vector-scalar floating-point sign copy . . . . .	107
2.5.36	Elementwise vector-vector floating-point sign copy . . . . .	108
2.5.37	Elementwise vector-scalar floating-point inverted sign copy . . . . .	109
2.5.38	Elementwise vector-vector floating-point inverted sign copy . . . . .	110
2.5.39	Elementwise vector-scalar floating-point XOR sign . . . . .	112
2.5.40	Elementwise vector-vector floating-point XOR sign . . . . .	113
2.5.41	Compute the square root . . . . .	114
2.5.42	Elementwise vector-scalar floating-point subtraction . . . . .	115
2.5.43	Elementwise vector-vector floating-point subtraction . . . . .	117
2.5.44	Floating-point vector-scalar widening additon . . . . .	118
2.5.45	Floating-point vector-vector widening additon . . . . .	119

2.5.46	Floating-point vector-scalar widening additon(second operand)	120
2.5.47	Floating-point vector-vector widening additon(second operand)	121
2.5.48	Floating-point vector-scalar widening multiply and add	122
2.5.49	Floating-point vector-vector widening multiply and add	123
2.5.50	Floating-point vector-scalar widening multiply and sub	124
2.5.51	Floating-point vector-vector widening multiply and sub	125
2.5.52	Floating-point vector-scalar widening multiplication	126
2.5.53	Floating-point vector-vector widening multiplication	127
2.5.54	Floating-point vector-scalar widening negate multiply and add	128
2.5.55	Floating-point vector-vector widening negate multiply and add	129
2.5.56	Floating-point vector-scalar widening negate multiply and sub	130
2.5.57	Floating-point vector-vector widening negate multiply and sub	132
2.5.58	Floating-point widening odered sum of vector	133
2.5.59	Floating-point widening sum of vector	134
2.5.60	Floating-point vector-scalar widening subtraction	135
2.5.61	Floating-point vector-vector widening subtraction	136
2.5.62	Floating-point vector-scalar widening subtraction(second operand)	137
2.5.63	Floating-point vector-vector widening subtraction(second operand)	138
2.6	<b>Floating-point relational operations</b>	139
2.6.1	Compare elementwise float vector-scalar for equality	139
2.6.2	Compare elementwise float vector-vector for equality	140
2.6.3	Compare elementwise float vector-scalar for greater-or-equal	141
2.6.4	Compare elementwise float vector-vector for greater-or-equal	143
2.6.5	Compare elementwise float vector-scalar for greater-than	144
2.6.6	Compare elementwise float vector-vector for greater-than	145
2.6.7	Compare elementwise float vector-scalar for lower-or-equal	147
2.6.8	Compare elementwise float vector-vector for lower-or-equal	148
2.6.9	Compare elementwise float vector-scalar for lower-than	150
2.6.10	Compare elementwise float vector-vector for lower-than	151
2.6.11	Compare elementwise float vector-scalar for inequality	152
2.6.12	Compare elementwise float vector-vector for inequality	154
2.6.13	Compute elementwise if vector-scalar are ordered floating-point values	155
2.6.14	Compute elementwise if vector-vector are ordered floating-point values	156
2.7	<b>Integer arithmetic operations</b>	158
2.7.1	Elementwise vector-immediate integer averge add	158
2.7.2	Elementwise vector-vector integer averge add	160
2.7.3	Elementwise vector-scalar integer averge add	161
2.7.4	Elementwise vector-immediate integer addition with carry	163
2.7.5	Elementwise vector-vector integer addition with carry	164
2.7.6	Elementwise vector-scalar integer addition with carry	165
2.7.7	Elementwise vector-immediate integer addition	167
2.7.8	Elementwise vector-vector integer addition	169
2.7.9	Elementwise vector-scalar integer addition	172
2.7.10	Elementwise vector-vector integer averge sub	175
2.7.11	Elementwise vector-scalar integer averge sub	176
2.7.12	Elementwise signed vector-vector division	178
2.7.13	Elementwise signed vector-scalar division	179
2.7.14	Elementwise unsigned vector-vector division	181
2.7.15	Elementwise unsigned vector-scalar division	183
2.7.16	Elementwise vector-vector integer dot-product	184
2.7.17	Elementwise vector-vector multiply-addition, overwrite addend	187
2.7.18	Elementwise vector-scalar multiply-addition, overwrite addend	189
2.7.19	Elementwise vector-immediate integer addition with carry in mask register format	192
2.7.20	Elementwise vector-vector integer addition with carry in mask register format	193

2.7.21	Elementwise vector-scalar integer addition with carry in mask register format . . . . .	195
2.7.22	Elementwise vector-vector multiply-addition, overwrite multiplicand . . . . .	197
2.7.23	Elementwise vector-scalar multiply-addition, overwrite multiplicand . . . . .	199
2.7.24	Elementwise signed vector-vector maximum . . . . .	202
2.7.25	Elementwise signed vector-scalar maximum . . . . .	203
2.7.26	Elementwise unsigned vector-vector maximum . . . . .	205
2.7.27	Elementwise unsigned vector-scalar maximum . . . . .	206
2.7.28	Elementwise signed vector-vector minumim . . . . .	208
2.7.29	Elementwise signed vector-scalar minumim . . . . .	209
2.7.30	Elementwise unsigned vector-vector minumim . . . . .	211
2.7.31	Elementwise unsigned vector-scalar minumim . . . . .	213
2.7.32	Elementwise vector-vector integer addition with borrow in mask register format . . . . .	214
2.7.33	Elementwise vector-scalar integer addition with borrow in mask register format . . . . .	216
2.7.34	Elementwise vector-vector integer multiplication . . . . .	217
2.7.35	Elementwise vector-scalar integer multiplication . . . . .	220
2.7.36	Elementwise signed vector-vector multiplication(higher bits) . . . . .	222
2.7.37	Elementwise signed vector-scalar multiplication(higher bits) . . . . .	223
2.7.38	Elementwise vector-vector signed-unsigned integer multiplication(higher bits) . . . . .	225
2.7.39	Elementwise vector-scalar signed-unsigned integer multiplication(higher bits) . . . . .	226
2.7.40	Elementwise unsigned vector-vector multiplication(higher bits) . . . . .	228
2.7.41	Elementwise unsigned vector-scalar multiplication(higher bits) . . . . .	230
2.7.42	Elementwise signed vector-immediate signed integer narrow clip . . . . .	231
2.7.43	Elementwise signed vector-vector signed integer narrow clip . . . . .	232
2.7.44	Elementwise signed vector-scalar signed integer narrow clip . . . . .	234
2.7.45	Elementwise unsigned vector-immediate unsigned integer narrow clip . . . . .	235
2.7.46	Elementwise unsigned vector-vector unsigned integer narrow clip . . . . .	236
2.7.47	Elementwise unsigned vector-scalar unsigned integer narrow clip . . . . .	237
2.7.48	Elementwise vector-vector multiply-subtarction, overwrite minuend . . . . .	239
2.7.49	Elementwise vector-scalar multiply-subtarction, overwrite minuend . . . . .	241
2.7.50	Elementwise vector-vector multiply-subtarction, overwrite multiplicand . . . . .	244
2.7.51	Elementwise vector-scalar multiply-subtarction, overwrite multiplicand . . . . .	246
2.7.52	Narrowing elementwise vector-immediate arithmetic shift right . . . . .	249
2.7.53	Narrowing elementwise vector-vector arithmetic shift right . . . . .	250
2.7.54	Narrowing elementwise vector-scalar arithmetic shift right . . . . .	251
2.7.55	Narrowing elementwise vector-immediate logic shift right . . . . .	252
2.7.56	Narrowing elementwise vector-vector logic shift right . . . . .	254
2.7.57	Narrowing elementwise vector-scalar logic shift right . . . . .	255
2.7.58	Integer vector bitwise-and reduction . . . . .	256
2.7.59	Integer vector signed maximum reduction . . . . .	259
2.7.60	Integer vector unsigned maximum reduction . . . . .	260
2.7.61	Integer vector signed minimum reduction . . . . .	262
2.7.62	Integer vector unsigned minimum reduction . . . . .	264
2.7.63	Integer vector bitwise-or reduction . . . . .	265
2.7.64	Integer vector sum reduction . . . . .	268
2.7.65	Integer vector bitwise-xor reduction . . . . .	271
2.7.66	Elementwise signed vector-vector division remainder . . . . .	274
2.7.67	Elementwise signed vector-scalar division remainder . . . . .	275
2.7.68	Elementwise unsigned vector-vector division remainder . . . . .	277
2.7.69	Elementwise unsigned vector-scalar division remainder . . . . .	278
2.7.70	Elementwise vector-immediate integer reverse subtraction . . . . .	280
2.7.71	Elementwise vector-scalar integer reverse subtraction . . . . .	283
2.7.72	Elementwise signed vector-immediate addition with saturation . . . . .	285
2.7.73	Elementwise signed vector-vector addition with saturation . . . . .	287
2.7.74	Elementwise signed vector-scalar addition with saturation . . . . .	288

2.7.75	Elementwise unsigned vector-immediate addition with saturation . . . . .	290
2.7.76	Elementwise unsigned vector-vector addition with saturation . . . . .	291
2.7.77	Elementwise unsigned vector-scalar addition with saturation . . . . .	293
2.7.78	Elementwise vector-vector integer addition with borrow . . . . .	294
2.7.79	Elementwise vector-scalar integer addition with borrow . . . . .	296
2.7.80	Elementwise vector-vector multiply with rounding and saturation . . . . .	297
2.7.81	Elementwise vector-scalar multiply with rounding and saturation . . . . .	299
2.7.82	Elementwise signed vector-immediate signed scaling shift . . . . .	300
2.7.83	Elementwise signed vector-vector signed scaling shift . . . . .	302
2.7.84	Elementwise signed vector-scalar signed scaling shift . . . . .	304
2.7.85	Elementwise unsigned vector-immediate unsigned scaling shift . . . . .	305
2.7.86	Elementwise unsigned vector-vector unsigned scaling shift . . . . .	307
2.7.87	Elementwise unsigned vector-scalar unsigned scaling shift . . . . .	308
2.7.88	Elementwise signed vector-vector subtraction with saturation . . . . .	310
2.7.89	Elementwise signed vector-scalar subtraction with saturation . . . . .	311
2.7.90	Elementwise unsigned vector-vector subtraction with saturation . . . . .	313
2.7.91	Elementwise unsigned vector-scalar subtraction with saturation . . . . .	315
2.7.92	Elementwise vector-vector integer subtraction . . . . .	316
2.7.93	Elementwise vector-scalar integer subtraction . . . . .	319
2.7.94	Widening elementwise signed vector-vector addition . . . . .	321
2.7.95	Widening elementwise signed vector-scalar addition . . . . .	322
2.7.96	Widening elementwise (Widening)signed vector-vector addition . . . . .	324
2.7.97	Widening elementwise (Widening)signed vector-scalar addition . . . . .	325
2.7.98	Widening elementwise unsigned vector-vector addition . . . . .	326
2.7.99	Widening elementwise unsigned vector-scalar addition . . . . .	327
2.7.100	Widening elementwise (Widening)unsigned vector-vector addition . . . . .	328
2.7.101	Widening elementwise (Widening)unsigned vector-scalar addition . . . . .	329
2.7.102	Widening elementwise signed vector-vector multiply-add, overwrite addend . . . . .	330
2.7.103	Widening elementwise signed vector-scalar multiply-add, overwrite addend . . . . .	332
2.7.104	Widening elementwise vector-vector signed-unsigned integer multiply-sub, overwrite addend . . . . .	333
2.7.105	Widening elementwise vector-scalar signed-unsigned integer multiply-sub, overwrite addend . . . . .	335
2.7.106	Widening elementwise unsigned vector-vector multiply-add, overwrite addend . . . . .	336
2.7.107	Widening elementwise unsigned vector-scalar multiply-add, overwrite addend . . . . .	338
2.7.108	Widening elementwise vector-scalar unsigned-signed integer multiply-sub, overwrite addend . . . . .	339
2.7.109	Widening elementwise signed vector-vector multiplition . . . . .	340
2.7.110	Widening elementwise signed vector-scalar multiplition . . . . .	342
2.7.111	Widening elementwise vector-vector signed-unsigned integer multiplication . . . . .	343
2.7.112	Widening elementwise vector-scalar signed-unsigned integer multiplication . . . . .	344
2.7.113	Widening elementwise unsigned vector-vector multiplication . . . . .	345
2.7.114	Widening elementwise unsigned vector-scalar multiplition . . . . .	346
2.7.115	Widening signed integer vector sum reduction . . . . .	348
2.7.116	Widening unsigned integer vector sum reduction . . . . .	349
2.7.117	Widening elementwise signed vector-vector multiply-add, overwrite addend, with round and saturation . . . . .	350
2.7.118	Widening elementwise signed vector-scalar multiply-add, overwrite addend, with round and saturation . . . . .	352
2.7.119	Widening elementwise vector-vector signed-unsigned integer multiply-sub, overwrite addend, with round and saturation . . . . .	353
2.7.120	Widening elementwise vector-scalar signed-unsigned integer multiply-sub, overwrite addend, with round and saturation . . . . .	355
2.7.121	Widening elementwise unsigned vector-vector multiply-add, overwrite addend, with round and saturation . . . . .	356
2.7.122	Widening elementwise unsigned vector-scalar multiply-add, overwrite addend, with round and saturation . . . . .	358



2.7.123	Widening elementwise vector-scalar unsigned-signed integer multiply-sub, overwrite addend, with round and saturation . . . . .	359
2.7.124	Widening elementwise signed vector-vector subtraction . . . . .	361
2.7.125	Widening elementwise signed vector-scalar subtraction . . . . .	362
2.7.126	Widening elementwise (Widening)signed vector-vector subtraction . . . . .	363
2.7.127	Widening elementwise (Widening)signed vector-scalar subtraction . . . . .	364
2.7.128	Widening elementwise unsigned vector-vector subtraction . . . . .	365
2.7.129	Widening elementwise unsigned vector-scalar subtraction . . . . .	366
2.7.130	Widening elementwise (Widening)unsigned vector-vector subtraction . . . . .	367
2.7.131	Widening elementwise (Widening)unsigned vector-scalar subtraction . . . . .	368
2.8	<b>Integer relational operations . . . . .</b>	370
2.8.1	Compare elementwise vector-immediate for equality . . . . .	370
2.8.2	Compare elementwise vector-vector for equality . . . . .	372
2.8.3	Compare elementwise vector-scalar for equality . . . . .	375
2.8.4	Compare elementwise signed integer one vector and one scalar for greater-than-or-equal . . . . .	378
2.8.5	Compare elementwise unsigned integer one vector and one scalar for greater-than-or-equal . . . . .	379
2.8.6	Compare elementwise signed integer one vector and one scalar immediate for greater-than . . . . .	381
2.8.7	Compare elementwise signed integer one vector and one scalar for greater-than . . . . .	382
2.8.8	Compare elementwise unsigned integer one vector and one scalar immediate for greater-than . . . . .	384
2.8.9	Compare elementwise unsigned integer one vector and one scalar for greater-than . . . . .	386
2.8.10	Compare elementwise signed integer one vector and one scalar immediate for lower-than-or-equal . . . . .	387
2.8.11	Compare elementwise signed vector-vector for lower-than-or-equal . . . . .	389
2.8.12	Compare elementwise signed vector-scalar for lower-than-or-equal . . . . .	390
2.8.13	Compare elementwise unsigned integer one vector and one scalar immediate for lower-than-or-equal . . . . .	392
2.8.14	Compare elementwise unsigned vector-vector for lower-than-or-equal . . . . .	393
2.8.15	Compare elementwise unsigned vector-scalar for lower-than-or-equal . . . . .	395
2.8.16	Compare elementwise signed vector-vector for lower-than . . . . .	396
2.8.17	Compare elementwise signed vector-scalar for lower-than . . . . .	398
2.8.18	Compare elementwise unsigned vector-vector for lower-than . . . . .	400
2.8.19	Compare elementwise unsigned vector-scalar for lower-than . . . . .	401
2.8.20	Compare elementwise vector-immediate for inequality . . . . .	403
2.8.21	Compare elementwise vector-vector for inequality . . . . .	405
2.8.22	Compare elementwise vector-scalar for inequality . . . . .	408
2.9	<b>Memory accesses . . . . .</b>	411
2.9.1	Load 8b signed in memory to vector . . . . .	411
2.9.2	Load 8b unsigned in memory to vector . . . . .	412
2.9.3	Load elements in memory to vector . . . . .	414
2.9.4	Load 16b signed in memory to vector . . . . .	417
2.9.5	Load 16b unsigned in memory to vector . . . . .	419
2.9.6	Load strided 8b signed in memory to vector . . . . .	421
2.9.7	Load strided 8b unsigned in memory to vector . . . . .	422
2.9.8	Load strided elements in memory to vector . . . . .	424
2.9.9	Load strided 16b signed in memory to vector . . . . .	427
2.9.10	Load strided 16b unsigned in memory to vector . . . . .	429
2.9.11	Load strided 32b signed in memory to vector . . . . .	431
2.9.12	Load strided 32b unsigned in memory to vector . . . . .	432
2.9.13	Load 32b signed in memory to vector . . . . .	434
2.9.14	Load 32b unsigned in memory to vector . . . . .	435
2.9.15	Load indexed 8b signed in memory to vector . . . . .	437
2.9.16	Load indexed 8b unsigned in memory to vector . . . . .	438
2.9.17	Load indexed element in memory to vector . . . . .	440
2.9.18	Load indexed 16b signed in memory to vector . . . . .	444



2.9.19	Load indexed 16b unsigned in memory to vector . . . . .	445
2.9.20	Load indexed 32b signed in memory to vector . . . . .	447
2.9.21	Load indexed 32b unsigned in memory to vector . . . . .	449
2.9.22	Store 8b in memory from vector . . . . .	450
2.9.23	Store elements in memory from vector . . . . .	453
2.9.24	Store 16b in memory from vector . . . . .	456
2.9.25	Store 8b in strided memory from vector . . . . .	458
2.9.26	Store elements in strided memory from vector . . . . .	461
2.9.27	Store 16b in strided memory from vector . . . . .	464
2.9.28	Store 32b in strided memory from vector . . . . .	467
2.9.29	Store 8b in unordered-indexed memory from vector . . . . .	469
2.9.30	Store element in unordered-indexed memory from vector . . . . .	472
2.9.31	Store 16b in unordered-indexed memory from vector . . . . .	476
2.9.32	Store 32b in unordered-indexed memory from vector . . . . .	478
2.9.33	Store 32b in memory from vector . . . . .	481
2.9.34	Store 8b in ordered-indexed memory from vector . . . . .	483
2.9.35	Store element in ordered-indexed memory from vector . . . . .	486
2.9.36	Store 16b in ordered-indexed memory from vector . . . . .	490
2.9.37	Store 32b in ordered-indexed memory from vector . . . . .	492
2.10	<b>Memory accesses(segment)</b> . . . . .	495
2.10.1	Load 2 contiguous 8b signed fields in memory to consecutively numbered vector registers . . . . .	495
2.10.2	Load 2 contiguous 8b unsigned fields in memory to consecutively numbered vector registers . . . . .	496
2.10.3	Load 2 contiguous element fields in memory to consecutively numbered vector registers . . . . .	498
2.10.4	Load 2 contiguous 16b signed fields in memory to consecutively numbered vector registers . . . . .	500
2.10.5	Load 2 contiguous 16b unsigned fields in memory to consecutively numbered vector registers . . . . .	502
2.10.6	Load 2 contiguous 32b signed fields in memory to consecutively numbered vector registers . . . . .	503
2.10.7	Load 2 contiguous 32b unsigned fields in memory to consecutively numbered vector registers . . . . .	504
2.10.8	Load 3 contiguous 8b signed fields in memory to consecutively numbered vector registers . . . . .	506
2.10.9	Load 3 contiguous 8b unsigned fields in memory to consecutively numbered vector registers . . . . .	507
2.10.10	Load 3 contiguous element fields in memory to consecutively numbered vector registers . . . . .	508
2.10.11	Load 3 contiguous 16b signed fields in memory to consecutively numbered vector registers . . . . .	510
2.10.12	Load 3 contiguous 16b unsigned fields in memory to consecutively numbered vector registers . . . . .	511
2.10.13	Load 3 contiguous 32b signed fields in memory to consecutively numbered vector registers . . . . .	512
2.10.14	Load 3 contiguous 32b unsigned fields in memory to consecutively numbered vector registers . . . . .	513
2.10.15	Load 4 contiguous 8b signed fields in memory to consecutively numbered vector registers . . . . .	514
2.10.16	Load 4 contiguous 8b unsigned fields in memory to consecutively numbered vector registers . . . . .	515
2.10.17	Load 4 contiguous element fields in memory to consecutively numbered vector registers . . . . .	516
2.10.18	Load 4 contiguous 16b signed fields in memory to consecutively numbered vector registers . . . . .	518
2.10.19	Load 4 contiguous 16b unsigned fields in memory to consecutively numbered vector registers . . . . .	519
2.10.20	Load 4 contiguous 32b signed fields in memory to consecutively numbered vector registers . . . . .	520
2.10.21	Load 4 contiguous 32b unsigned fields in memory to consecutively numbered vector registers . . . . .	521
2.10.22	Load 5 contiguous 8b signed fields in memory to consecutively numbered vector registers . . . . .	522
2.10.23	Load 5 contiguous 8b unsigned fields in memory to consecutively numbered vector registers . . . . .	523
2.10.24	Load 5 contiguous element fields in memory to consecutively numbered vector registers . . . . .	524
2.10.25	Load 5 contiguous 16b signed fields in memory to consecutively numbered vector registers . . . . .	525
2.10.26	Load 5 contiguous 16b unsigned fields in memory to consecutively numbered vector registers . . . . .	526
2.10.27	Load 5 contiguous 32b signed fields in memory to consecutively numbered vector registers . . . . .	527
2.10.28	Load 5 contiguous 32b unsigned fields in memory to consecutively numbered vector registers . . . . .	527
2.10.29	Load 6 contiguous 8b signed fields in memory to consecutively numbered vector registers . . . . .	528
2.10.30	Load 6 contiguous 8b unsigned fields in memory to consecutively numbered vector registers . . . . .	529
2.10.31	Load 6 contiguous element fields in memory to consecutively numbered vector registers . . . . .	530
2.10.32	Load 6 contiguous 16b signed fields in memory to consecutively numbered vector registers . . . . .	531
2.10.33	Load 6 contiguous 16b unsigned fields in memory to consecutively numbered vector registers . . . . .	532
2.10.34	Load 6 contiguous 32b signed fields in memory to consecutively numbered vector registers . . . . .	532

2.10.35	Load 6 contiguous 32b unsigned fields in memory to consecutively numbered vector registers	533
2.10.36	Load 7 contiguous 8b signed fields in memory to consecutively numbered vector registers .	534
2.10.37	Load 7 contiguous 8b unsigned fields in memory to consecutively numbered vector registers	535
2.10.38	Load 7 contiguous element fields in memory to consecutively numbered vector registers . .	535
2.10.39	Load 7 contiguous 16b signed fields in memory to consecutively numbered vector registers .	537
2.10.40	Load 7 contiguous 16b unsigned fields in memory to consecutively numbered vector registers	537
2.10.41	Load 7 contiguous 32b signed fields in memory to consecutively numbered vector registers .	538
2.10.42	Load 7 contiguous 32b unsigned fields in memory to consecutively numbered vector registers	539
2.10.43	Load 8 contiguous 8b signed fields in memory to consecutively numbered vector registers .	540
2.10.44	Load 8 contiguous 8b unsigned fields in memory to consecutively numbered vector registers	540
2.10.45	Load 8 contiguous element fields in memory to consecutively numbered vector registers . .	541
2.10.46	Load 8 contiguous 16b signed fields in memory to consecutively numbered vector registers .	542
2.10.47	Load 8 contiguous 16b unsigned fields in memory to consecutively numbered vector registers	543
2.10.48	Load 8 contiguous 32b signed fields in memory to consecutively numbered vector registers .	544
2.10.49	Load 8 contiguous 32b unsigned fields in memory to consecutively numbered vector registers	545
2.10.50	Load 2 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	545
2.10.51	Load 2 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	547
2.10.52	Load 2 contiguous element fields in memory(strided) to consecutively numbered vector registers . . . . .	548
2.10.53	Load 2 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	552
2.10.54	Load 2 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	553
2.10.55	Load 2 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	555
2.10.56	Load 2 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	556
2.10.57	Load 3 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	558
2.10.58	Load 3 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	559
2.10.59	Load 3 contiguous element fields in memory(strided) to consecutively numbered vector registers . . . . .	560
2.10.60	Load 3 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	562
2.10.61	Load 3 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	563
2.10.62	Load 3 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	564
2.10.63	Load 3 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	565
2.10.64	Load 4 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	567
2.10.65	Load 4 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	568
2.10.66	Load 4 contiguous element fields in memory(strided) to consecutively numbered vector registers . . . . .	569
2.10.67	Load 4 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	571
2.10.68	Load 4 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	572

2.10.69	Load 4 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	574
2.10.70	Load 4 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	575
2.10.71	Load 5 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	576
2.10.72	Load 5 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	577
2.10.73	Load 5 contiguous element fields in memory(strided) to consecutively numbered vector registers . . . . .	577
2.10.74	Load 5 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	579
2.10.75	Load 5 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	580
2.10.76	Load 5 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	580
2.10.77	Load 5 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	581
2.10.78	Load 6 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	582
2.10.79	Load 6 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	583
2.10.80	Load 6 contiguous element fields in memory(strided) to consecutively numbered vector registers . . . . .	584
2.10.81	Load 6 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	585
2.10.82	Load 6 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	586
2.10.83	Load 6 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	587
2.10.84	Load 6 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	587
2.10.85	Load 7 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	588
2.10.86	Load 7 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	589
2.10.87	Load 7 contiguous element fields in memory(strided) to consecutively numbered vector registers . . . . .	590
2.10.88	Load 7 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	591
2.10.89	Load 7 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	592
2.10.90	Load 7 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	593
2.10.91	Load 7 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	594
2.10.92	Load 8 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	595
2.10.93	Load 8 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	595
2.10.94	Load 8 contiguous element fields in memory(strided) to consecutively numbered vector registers . . . . .	596
2.10.95	Load 8 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	598

2.10.96	Load 8 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	598
2.10.97	Load 8 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers . . . . .	599
2.10.98	Load 8 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers . . . . .	600
2.10.99	Load 2 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	601
2.10.100	Load 2 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	603
2.10.101	Load 2 contiguous element fields in memory(indexed) to consecutively numbered vector registers . . . . .	605
2.10.102	Load 2 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	609
2.10.103	Load 2 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	611
2.10.104	Load 2 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	613
2.10.105	Load 2 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	615
2.10.106	Load 3 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	617
2.10.107	Load 3 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	618
2.10.108	Load 3 contiguous element fields in memory(indexed) to consecutively numbered vector registers . . . . .	620
2.10.109	Load 3 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	623
2.10.110	Load 3 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	624
2.10.111	Load 3 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	626
2.10.112	Load 3 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	627
2.10.113	Load 4 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	628
2.10.114	Load 4 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	630
2.10.115	Load 4 contiguous element fields in memory(indexed) to consecutively numbered vector registers . . . . .	631
2.10.116	Load 4 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	634
2.10.117	Load 4 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	636
2.10.118	Load 4 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	637
2.10.119	Load 4 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	639
2.10.120	Load 5 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	640
2.10.121	Load 5 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	641
2.10.122	Load 5 contiguous element fields in memory(indexed) to consecutively numbered vector registers . . . . .	642

2.10.123Load 5 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	644
2.10.124Load 5 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	645
2.10.125Load 5 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	646
2.10.126Load 5 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	647
2.10.127Load 6 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	648
2.10.128Load 6 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	648
2.10.129Load 6 contiguous element fields in memory(indexed) to consecutively numbered vector registers . . . . .	649
2.10.130Load 6 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	651
2.10.131Load 6 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	652
2.10.132Load 6 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	653
2.10.133Load 6 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	654
2.10.134Load 7 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	655
2.10.135Load 7 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	656
2.10.136Load 7 contiguous element fields in memory(indexed) to consecutively numbered vector registers . . . . .	657
2.10.137Load 7 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	659
2.10.138Load 7 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	660
2.10.139Load 7 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	661
2.10.140Load 7 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	662
2.10.141Load 8 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	663
2.10.142Load 8 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	663
2.10.143Load 8 contiguous element fields in memory(indexed) to consecutively numbered vector registers . . . . .	664
2.10.144Load 8 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	666
2.10.145Load 8 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	667
2.10.146Load 8 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers . . . . .	668
2.10.147Load 8 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers . . . . .	669
2.10.148Store 2 contiguous 8b fields in memory from consecutively numbered vector registers . . . .	670
2.10.149Store 2 contiguous element fields in memory from consecutively numbered vector registers .	672
2.10.150Store 2 contiguous 16b fields in memory from consecutively numbered vector registers . . .	675
2.10.151Store 2 contiguous 32b fields in memory from consecutively numbered vector registers . . .	677





2.10.200Store 8 contiguous 8b fields in memory(strided) from consecutively numbered vector registers	745
2.10.201Store 8 contiguous element fields in memory(strided) from consecutively numbered vector registers	746
2.10.202Store 8 contiguous 16b fields in memory(strided) from consecutively numbered vector registers	747
2.10.203Store 8 contiguous 32b fields in memory(strided) from consecutively numbered vector registers	748
2.10.204Store 2 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers	749
2.10.205Store 2 contiguous element fields in memory(indexed) from consecutively numbered vector registers	752
2.10.206Store 2 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers	756
2.10.207Store 2 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers	759
2.10.208Store 3 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers	762
2.10.209Store 3 contiguous element fields in memory(indexed) from consecutively numbered vector registers	764
2.10.210Store 3 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers	766
2.10.211Store 3 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers	768
2.10.212Store 4 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers	770
2.10.213Store 4 contiguous element fields in memory(indexed) from consecutively numbered vector registers	772
2.10.214Store 4 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers	775
2.10.215Store 4 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers	777
2.10.216Store 5 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers	779
2.10.217Store 5 contiguous element fields in memory(indexed) from consecutively numbered vector registers	780
2.10.218Store 5 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers	782
2.10.219Store 5 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers	783
2.10.220Store 6 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers	784
2.10.221Store 6 contiguous element fields in memory(indexed) from consecutively numbered vector registers	785
2.10.222Store 6 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers	787
2.10.223Store 6 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers	788
2.10.224Store 7 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers	789
2.10.225Store 7 contiguous element fields in memory(indexed) from consecutively numbered vector registers	790
2.10.226Store 7 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers	792
2.10.227Store 7 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers	793
2.10.228Store 8 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers	794
2.10.229Store 8 contiguous element fields in memory(indexed) from consecutively numbered vector registers	795
2.10.230Store 8 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers	797
2.10.231Store 8 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers	798



2.11	<b>Operations with masks</b>	799
2.11.1	Compute elementwise logical and between two masks	799
2.11.2	Compute elementwise logical andnot between two masks	800
2.11.3	Clear elementwise mask	801
2.11.4	Copy elementwise mask	801
2.11.5	Compute the index of the first enabled element	802
2.11.6	Compute elementwise logical negated and between two masks	803
2.11.7	Compute elementwise logical negated or between two masks	804
2.11.8	Invert elementwise mask	805
2.11.9	Compute elementwise logical or between two masks	805
2.11.10	Compute elementwise logical ornot between two masks	806
2.11.11	Population count of a mask vector	807
2.11.12	Enable elements before the first one enabled	808
2.11.13	Set elementwise mask	809
2.11.14	Enable elements until the first one enabled	810
2.11.15	Enable only the first element enabled	811
2.11.16	Compute elementwise logical xnor between two masks	813
2.11.17	Compute elementwise logical xor between two masks	813
2.12	<b>Vector elements manipulation</b>	814
2.12.1	Pack elements contiguously	814
2.12.2	Extract integer element	815
2.12.3	Merge two floating-point vectors using a mask vector	816
2.12.4	move the lowest element of a vector to the given floating-point	817
2.12.5	move a floating-point to the lowest element of the vector	818
2.12.6	Create a vector that all the elements the same as the given floating-point	818
2.12.7	Compute index vector	819
2.12.8	Compute a prefix sum of a mask	820
2.12.9	Elementwise vector-immediate integer merge	822
2.12.10	Elementwise vector-vector integer merge	823
2.12.11	Elementwise vector-scalar integer merge	825
2.12.12	Set first integer element of integer vector	826
2.12.13	Elementwise immediate integer move	827
2.12.14	Elementwise vector integer move	828
2.12.15	Elementwise scalar integer move	830
2.12.16	Get first integer element of integer vector	831
2.12.17	Gather vector-immediate (index)	832
2.12.18	Gather vector-vector (index)	835
2.12.19	Gather vector-scalar (index)	840
2.12.20	Slide down one element of a vector	843
2.12.21	Slide up one element of a vector	847
2.12.22	Slide down elements of vector-immediate (indexed)	851
2.12.23	Slide down elements of vector-scalar (indexed)	854
2.12.24	Slide up elements of vector-immediate (indexed)	858
2.12.25	Slide up elements of vector-scalar (indexed)	861
3	<b>Examples</b>	867
3.1	Vector-vector add	867
3.2	Vector-vector add with inner data type	867
4	<b>Appendix A: fcsr</b>	869
4.1	Vector Floating Rounding Mode Register frm	869
4.2	Vector Fixed-Point Rounding Mode Register vxrm	869
	<b>Index</b>	871

## INTRODUCTION

### 1.1 Vector types

An implementation of the RISC-V V-extension features 32 vector registers of length `VLEN` bits. Each vector register holds a number of elements. The wider element, in bits, that an implementation supports is called `ELEN`.

A vector, thus, can hold `VLEN/ELEN` elements of the widest element implemented. This also means that the same vector can hold twice that number of the element is half the size. This is, a vector of floats will always hold twice the number of elements that a vector of doubles can hold.

Vector registers in the V-extension can be grouped. Grouping can be 1 (no grouping actually), 2, 4 or 8. Grouping means larger vectors but in a smaller number (e.g. there are only 16 registers with grouping 2). Grouping is part of the state of the extension and it is called `LMUL` (length multiplier). A `LMUL` of 1 means no grouping.

The following types are available to operate the vectors under different `LMUL` configurations.

Vector of	LMUL=1	LMUL=2	LMUL=4	LMUL=8
double	<code>float64xm1_t</code>	<code>float64xm2_t</code>	<code>float64xm4_t</code>	<code>float64xm8_t</code>
float	<code>float32xm1_t</code>	<code>float32xm2_t</code>	<code>float32xm4_t</code>	<code>float32xm8_t</code>
float16	<code>float16xm1_t</code>	<code>float16xm2_t</code>	<code>float16xm4_t</code>	<code>float16xm8_t</code>
int64	<code>int64xm1_t</code>	<code>int64xm2_t</code>	<code>int64xm4_t</code>	<code>int64xm8_t</code>
int32	<code>int32xm1_t</code>	<code>int32xm2_t</code>	<code>int32xm4_t</code>	<code>int32xm8_t</code>
int16	<code>int16xm1_t</code>	<code>int16xm2_t</code>	<code>int16xm4_t</code>	<code>int16xm8_t</code>
int8	<code>int8xm1_t</code>	<code>int8xm2_t</code>	<code>int8xm4_t</code>	<code>int8xm8_t</code>
uint64	<code>uint64xm1_t</code>	<code>uint64xm2_t</code>	<code>uint64xm4_t</code>	<code>uint64xm8_t</code>
uint32	<code>uint32xm1_t</code>	<code>uint32xm2_t</code>	<code>uint32xm4_t</code>	<code>uint32xm8_t</code>
uint16	<code>uint16xm1_t</code>	<code>uint16xm2_t</code>	<code>uint16xm4_t</code>	<code>uint16xm8_t</code>
uint8	<code>uint8xm1_t</code>	<code>uint8xm2_t</code>	<code>uint8xm4_t</code>	<code>uint8xm8_t</code>

The syntax of vector types is <element type>x<lmul>\_t.

## 1.2 Mask types

Element bits	LMUL=1	LMUL=2	LMUL=4	LMUL=8
64	<code>e64xm1_t</code>	<code>e64xm2_t</code>	<code>e64xm4_t</code>	<code>e64xm8_t</code>
32	<code>e32xm1_t</code>	<code>e32xm2_t</code>	<code>e32xm4_t</code>	<code>e32xm8_t</code>
16	<code>e16xm1_t</code>	<code>e16xm2_t</code>	<code>e16xm4_t</code>	<code>e16xm8_t</code>
8	<code>e8xm1_t</code>	<code>e8xm2_t</code>	<code>e8xm4_t</code>	<code>e8xm8_t</code>

The syntax of mask types is <element bits>x<lmul>\_t. Mask types are unrelated to LMUL in that they always use a single vector register.

## 1.3 Immediate operand

In vector instruction, immediate is encoded in 5 bits as signed by default except comment of the specific instruction.

## 1.4 Memory address operand

If the elements accessed by a vector memory instruction are not naturally aligned to the memory element size, an address misaligned exception is raised on that element.

## 1.5 Calling Convention

The RISC-V calling convention passes arguments in registers when possible. Up to sixteen vector registers, v8–v23 are used for this purpose.

Table 1: vector ABI

Register	ABI Name	Description	Saver
v0-7	v0-7	Temporaries	Caller
v8-15	v8-15	Function arguments/return values	Caller
v16-23	v16-23	Function arguments	Caller
v24-31	v24-31	Saved register	Callee

Table 2: soft vector ABI

Register	ABI Name	Description	Saver
v0-31	v0-31	Temporaries	Caller

## 1.6 Supported CPUs

CPU	ABI	Arch	Spec
910v	vector	gcvxthead	0.7.1

THEAD

## REFERENCE

### 2.1 Vector configuration

#### 2.1.1 Change the granted vector length by *vtype*

**Instruction:** ['vsetvl']

**Prototypes:**

- unsigned int **vsetvl** (unsigned int *avl*, int *vtype*)

**Operation:**

```
>>> gvl = compute_vector_length(avl, vtype)
      result = gvl
```

#### 2.1.2 Change the granted vector length

**Instruction:** ['vsetvli']

**Prototypes:**

- unsigned int **vsetvli** (unsigned int *avl*, const int *sew*, const int *lmul*)

**Operation:**

```
>>> gvl = compute_vector_length(avl, sew, lmul)
      result = gvl
```

### 2.2 Bit manipulation

#### 2.2.1 Elementwise vector-immediate bitwise-and

**Instruction:** ['vand.vi']

**Prototypes:**

- *int16xm1\_t* **vandvi\_int16xm1** (*int16xm1\_t* *a*, const short *b*, unsigned int *gvl*)
- *int16xm2\_t* **vandvi\_int16xm2** (*int16xm2\_t* *a*, const short *b*, unsigned int *gvl*)
- *int16xm4\_t* **vandvi\_int16xm4** (*int16xm4\_t* *a*, const short *b*, unsigned int *gvl*)

- `int16xm8_t vandvi_int16xm8 (int16xm8_t a, const short b, unsigned int gvl)`
- `int32xm1_t vandvi_int32xm1 (int32xm1_t a, const int b, unsigned int gvl)`
- `int32xm2_t vandvi_int32xm2 (int32xm2_t a, const int b, unsigned int gvl)`
- `int32xm4_t vandvi_int32xm4 (int32xm4_t a, const int b, unsigned int gvl)`
- `int32xm8_t vandvi_int32xm8 (int32xm8_t a, const int b, unsigned int gvl)`
- `int64xm1_t vandvi_int64xm1 (int64xm1_t a, const long b, unsigned int gvl)`
- `int64xm2_t vandvi_int64xm2 (int64xm2_t a, const long b, unsigned int gvl)`
- `int64xm4_t vandvi_int64xm4 (int64xm4_t a, const long b, unsigned int gvl)`
- `int64xm8_t vandvi_int64xm8 (int64xm8_t a, const long b, unsigned int gvl)`
- `int8xm1_t vandvi_int8xm1 (int8xm1_t a, const signed char b, unsigned int gvl)`
- `int8xm2_t vandvi_int8xm2 (int8xm2_t a, const signed char b, unsigned int gvl)`
- `int8xm4_t vandvi_int8xm4 (int8xm4_t a, const signed char b, unsigned int gvl)`
- `int8xm8_t vandvi_int8xm8 (int8xm8_t a, const signed char b, unsigned int gvl)`
- `uint16xm1_t vandvi_uint16xm1 (uint16xm1_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm2_t vandvi_uint16xm2 (uint16xm2_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm4_t vandvi_uint16xm4 (uint16xm4_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm8_t vandvi_uint16xm8 (uint16xm8_t a, const unsigned short b, unsigned int gvl)`
- `uint32xm1_t vandvi_uint32xm1 (uint32xm1_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm2_t vandvi_uint32xm2 (uint32xm2_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm4_t vandvi_uint32xm4 (uint32xm4_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm8_t vandvi_uint32xm8 (uint32xm8_t a, const unsigned int b, unsigned int gvl)`
- `uint64xm1_t vandvi_uint64xm1 (uint64xm1_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm2_t vandvi_uint64xm2 (uint64xm2_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm4_t vandvi_uint64xm4 (uint64xm4_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm8_t vandvi_uint64xm8 (uint64xm8_t a, const unsigned long b, unsigned int gvl)`
- `uint8xm1_t vandvi_uint8xm1 (uint8xm1_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm2_t vandvi_uint8xm2 (uint8xm2_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm4_t vandvi_uint8xm4 (uint8xm4_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm8_t vandvi_uint8xm8 (uint8xm8_t a, const unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = bitwise_and (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vandvi_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, const short b, e16xm1_t mask, unsigned int gvl)`



- *int16xm2\_t vandvi\_mask\_int16xm2* (*int16xm2\_t merge*, *int16xm2\_t a*, const short *b*, *e16xm2\_t mask*, unsigned int gvl)
- *int16xm4\_t vandvi\_mask\_int16xm4* (*int16xm4\_t merge*, *int16xm4\_t a*, const short *b*, *e16xm4\_t mask*, unsigned int gvl)
- *int16xm8\_t vandvi\_mask\_int16xm8* (*int16xm8\_t merge*, *int16xm8\_t a*, const short *b*, *e16xm8\_t mask*, unsigned int gvl)
- *int32xm1\_t vandvi\_mask\_int32xm1* (*int32xm1\_t merge*, *int32xm1\_t a*, const int *b*, *e32xm1\_t mask*, unsigned int gvl)
- *int32xm2\_t vandvi\_mask\_int32xm2* (*int32xm2\_t merge*, *int32xm2\_t a*, const int *b*, *e32xm2\_t mask*, unsigned int gvl)
- *int32xm4\_t vandvi\_mask\_int32xm4* (*int32xm4\_t merge*, *int32xm4\_t a*, const int *b*, *e32xm4\_t mask*, unsigned int gvl)
- *int32xm8\_t vandvi\_mask\_int32xm8* (*int32xm8\_t merge*, *int32xm8\_t a*, const int *b*, *e32xm8\_t mask*, unsigned int gvl)
- *int64xm1\_t vandvi\_mask\_int64xm1* (*int64xm1\_t merge*, *int64xm1\_t a*, const long *b*, *e64xm1\_t mask*, unsigned int gvl)
- *int64xm2\_t vandvi\_mask\_int64xm2* (*int64xm2\_t merge*, *int64xm2\_t a*, const long *b*, *e64xm2\_t mask*, unsigned int gvl)
- *int64xm4\_t vandvi\_mask\_int64xm4* (*int64xm4\_t merge*, *int64xm4\_t a*, const long *b*, *e64xm4\_t mask*, unsigned int gvl)
- *int64xm8\_t vandvi\_mask\_int64xm8* (*int64xm8\_t merge*, *int64xm8\_t a*, const long *b*, *e64xm8\_t mask*, unsigned int gvl)
- *int8xm1\_t vandvi\_mask\_int8xm1* (*int8xm1\_t merge*, *int8xm1\_t a*, const signed char *b*, *e8xm1\_t mask*, unsigned int gvl)
- *int8xm2\_t vandvi\_mask\_int8xm2* (*int8xm2\_t merge*, *int8xm2\_t a*, const signed char *b*, *e8xm2\_t mask*, unsigned int gvl)
- *int8xm4\_t vandvi\_mask\_int8xm4* (*int8xm4\_t merge*, *int8xm4\_t a*, const signed char *b*, *e8xm4\_t mask*, unsigned int gvl)
- *int8xm8\_t vandvi\_mask\_int8xm8* (*int8xm8\_t merge*, *int8xm8\_t a*, const signed char *b*, *e8xm8\_t mask*, unsigned int gvl)
- *uint16xm1\_t vandvi\_mask\_uint16xm1* (*uint16xm1\_t merge*, *uint16xm1\_t a*, const unsigned short *b*, *e16xm1\_t mask*, unsigned int gvl)
- *uint16xm2\_t vandvi\_mask\_uint16xm2* (*uint16xm2\_t merge*, *uint16xm2\_t a*, const unsigned short *b*, *e16xm2\_t mask*, unsigned int gvl)
- *uint16xm4\_t vandvi\_mask\_uint16xm4* (*uint16xm4\_t merge*, *uint16xm4\_t a*, const unsigned short *b*, *e16xm4\_t mask*, unsigned int gvl)
- *uint16xm8\_t vandvi\_mask\_uint16xm8* (*uint16xm8\_t merge*, *uint16xm8\_t a*, const unsigned short *b*, *e16xm8\_t mask*, unsigned int gvl)
- *uint32xm1\_t vandvi\_mask\_uint32xm1* (*uint32xm1\_t merge*, *uint32xm1\_t a*, const unsigned int *b*, *e32xm1\_t mask*, unsigned int gvl)
- *uint32xm2\_t vandvi\_mask\_uint32xm2* (*uint32xm2\_t merge*, *uint32xm2\_t a*, const unsigned int *b*, *e32xm2\_t mask*, unsigned int gvl)
- *uint32xm4\_t vandvi\_mask\_uint32xm4* (*uint32xm4\_t merge*, *uint32xm4\_t a*, const unsigned int *b*, *e32xm4\_t mask*, unsigned int gvl)
- *uint32xm8\_t vandvi\_mask\_uint32xm8* (*uint32xm8\_t merge*, *uint32xm8\_t a*, const unsigned int *b*, *e32xm8\_t mask*, unsigned int gvl)

- `uint64xm1_t vandvi_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, const unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vandvi_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, const unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vandvi_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, const unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vandvi_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, const unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vandvi_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, const unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vandvi_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, const unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vandvi_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, const unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vandvi_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, const unsigned char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = bitwise_and (a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.2.2 Elementwise vector-vector bitwise-and

**Instruction:** ['vand.vv']

**Prototypes:**

- `int16xm1_t vandvv_int16xm1 (int16xm1_t a, int16xm1_t b, unsigned int gvl)`
- `int16xm2_t vandvv_int16xm2 (int16xm2_t a, int16xm2_t b, unsigned int gvl)`
- `int16xm4_t vandvv_int16xm4 (int16xm4_t a, int16xm4_t b, unsigned int gvl)`
- `int16xm8_t vandvv_int16xm8 (int16xm8_t a, int16xm8_t b, unsigned int gvl)`
- `int32xm1_t vandvv_int32xm1 (int32xm1_t a, int32xm1_t b, unsigned int gvl)`
- `int32xm2_t vandvv_int32xm2 (int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int32xm4_t vandvv_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vandvv_int32xm8 (int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vandvv_int64xm1 (int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vandvv_int64xm2 (int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vandvv_int64xm4 (int64xm4_t a, int64xm4_t b, unsigned int gvl)`
- `int64xm8_t vandvv_int64xm8 (int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vandvv_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vandvv_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`

- `int8xm4_t vandvv_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vandvv_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`
- `uint16xm1_t vandvv_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vandvv_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vandvv_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vandvv_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vandvv_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vandvv_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vandvv_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vandvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vandvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vandvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vandvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vandvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vandvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vandvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vandvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vandvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

#### Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = bitwise_and (a[element], b[element])
result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- `int16xm1_t vandvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vandvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vandvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vandvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vandvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vandvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vandvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vandvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`

- `int64xm1_t vandvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vandvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vandvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vandvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vandvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vandvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vandvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vandvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vandvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vandvv_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vandvv_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vandvv_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vandvv_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vandvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vandvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vandvv_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vandvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vandvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vandvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vandvv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vandvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vandvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vandvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`

- `uint8xm8_t vandvv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = bitwise_and (a[element], b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

### 2.2.3 Elementwise vector-scalar bitwise-and

Instruction: ['vand.vx']

Prototypes:

- `int16xm1_t vandvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int16xm2_t vandvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vandvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vandvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vandvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vandvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vandvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vandvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vandvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vandvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vandvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vandvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vandvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vandvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vandvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vandvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`
- `uint16xm1_t vandvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vandvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vandvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vandvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vandvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vandvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vandvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vandvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vandvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`

- `uint64xm2_t vandvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vandvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vandvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vandvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vandvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vandvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vandvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = bitwise_and (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vandvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vandvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vandvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vandvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vandvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vandvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vandvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vandvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vandvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vandvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vandvx_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vandvx_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vandvx_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vandvx_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vandvx_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, signed char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vandvx_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, signed char b, e8xm8_t mask, unsigned int gvl)`



- `uint16xm1_t vandvx_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vandvx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vandvx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vandvx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vandvx_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vandvx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vandvx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vandvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vandvx_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vandvx_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vandvx_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vandvx_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vandvx_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vandvx_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vandvx_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vandvx_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = bitwise_and (a[element], b)
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.2.4 Elementwise vector bitwise-not

**Instruction:** [`'vnot.v'`]

**Prototypes:**

- `int16xm1_t vnotv_int16xm1 (int16xm1_t a, unsigned int gvl)`



- `int16xm2_t vnotv_int16xm2 (int16xm2_t a, unsigned int gvl)`
- `int16xm4_t vnotv_int16xm4 (int16xm4_t a, unsigned int gvl)`
- `int16xm8_t vnotv_int16xm8 (int16xm8_t a, unsigned int gvl)`
- `int32xm1_t vnotv_int32xm1 (int32xm1_t a, unsigned int gvl)`
- `int32xm2_t vnotv_int32xm2 (int32xm2_t a, unsigned int gvl)`
- `int32xm4_t vnotv_int32xm4 (int32xm4_t a, unsigned int gvl)`
- `int32xm8_t vnotv_int32xm8 (int32xm8_t a, unsigned int gvl)`
- `int64xm1_t vnotv_int64xm1 (int64xm1_t a, unsigned int gvl)`
- `int64xm2_t vnotv_int64xm2 (int64xm2_t a, unsigned int gvl)`
- `int64xm4_t vnotv_int64xm4 (int64xm4_t a, unsigned int gvl)`
- `int64xm8_t vnotv_int64xm8 (int64xm8_t a, unsigned int gvl)`
- `int8xm1_t vnotv_int8xm1 (int8xm1_t a, unsigned int gvl)`
- `int8xm2_t vnotv_int8xm2 (int8xm2_t a, unsigned int gvl)`
- `int8xm4_t vnotv_int8xm4 (int8xm4_t a, unsigned int gvl)`
- `int8xm8_t vnotv_int8xm8 (int8xm8_t a, unsigned int gvl)`
- `uint16xm1_t vnotv_uint16xm1 (uint16xm1_t a, unsigned int gvl)`
- `uint16xm2_t vnotv_uint16xm2 (uint16xm2_t a, unsigned int gvl)`
- `uint16xm4_t vnotv_uint16xm4 (uint16xm4_t a, unsigned int gvl)`
- `uint16xm8_t vnotv_uint16xm8 (uint16xm8_t a, unsigned int gvl)`
- `uint32xm1_t vnotv_uint32xm1 (uint32xm1_t a, unsigned int gvl)`
- `uint32xm2_t vnotv_uint32xm2 (uint32xm2_t a, unsigned int gvl)`
- `uint32xm4_t vnotv_uint32xm4 (uint32xm4_t a, unsigned int gvl)`
- `uint32xm8_t vnotv_uint32xm8 (uint32xm8_t a, unsigned int gvl)`
- `uint64xm1_t vnotv_uint64xm1 (uint64xm1_t a, unsigned int gvl)`
- `uint64xm2_t vnotv_uint64xm2 (uint64xm2_t a, unsigned int gvl)`
- `uint64xm4_t vnotv_uint64xm4 (uint64xm4_t a, unsigned int gvl)`
- `uint64xm8_t vnotv_uint64xm8 (uint64xm8_t a, unsigned int gvl)`
- `uint8xm1_t vnotv_uint8xm1 (uint8xm1_t a, unsigned int gvl)`
- `uint8xm2_t vnotv_uint8xm2 (uint8xm2_t a, unsigned int gvl)`
- `uint8xm4_t vnotv_uint8xm4 (uint8xm4_t a, unsigned int gvl)`
- `uint8xm8_t vnotv_uint8xm8 (uint8xm8_t a, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = bitwise_not (a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vnotv_mask_int16xm1` (`int16xm1_t merge`, `int16xm1_t a`, `e16xm1_t mask`, unsigned int gvl)
- `int16xm2_t vnotv_mask_int16xm2` (`int16xm2_t merge`, `int16xm2_t a`, `e16xm2_t mask`, unsigned int gvl)
- `int16xm4_t vnotv_mask_int16xm4` (`int16xm4_t merge`, `int16xm4_t a`, `e16xm4_t mask`, unsigned int gvl)
- `int16xm8_t vnotv_mask_int16xm8` (`int16xm8_t merge`, `int16xm8_t a`, `e16xm8_t mask`, unsigned int gvl)
- `int32xm1_t vnotv_mask_int32xm1` (`int32xm1_t merge`, `int32xm1_t a`, `e32xm1_t mask`, unsigned int gvl)
- `int32xm2_t vnotv_mask_int32xm2` (`int32xm2_t merge`, `int32xm2_t a`, `e32xm2_t mask`, unsigned int gvl)
- `int32xm4_t vnotv_mask_int32xm4` (`int32xm4_t merge`, `int32xm4_t a`, `e32xm4_t mask`, unsigned int gvl)
- `int32xm8_t vnotv_mask_int32xm8` (`int32xm8_t merge`, `int32xm8_t a`, `e32xm8_t mask`, unsigned int gvl)
- `int64xm1_t vnotv_mask_int64xm1` (`int64xm1_t merge`, `int64xm1_t a`, `e64xm1_t mask`, unsigned int gvl)
- `int64xm2_t vnotv_mask_int64xm2` (`int64xm2_t merge`, `int64xm2_t a`, `e64xm2_t mask`, unsigned int gvl)
- `int64xm4_t vnotv_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, `e64xm4_t mask`, unsigned int gvl)
- `int64xm8_t vnotv_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, `e64xm8_t mask`, unsigned int gvl)
- `int8xm1_t vnotv_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, `e8xm1_t mask`, unsigned int gvl)
- `int8xm2_t vnotv_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, `e8xm2_t mask`, unsigned int gvl)
- `int8xm4_t vnotv_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, `e8xm4_t mask`, unsigned int gvl)
- `int8xm8_t vnotv_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, `e8xm8_t mask`, unsigned int gvl)
- `uint16xm1_t vnotv_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, `e16xm1_t mask`, unsigned int gvl)
- `uint16xm2_t vnotv_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, `e16xm2_t mask`, unsigned int gvl)
- `uint16xm4_t vnotv_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, `e16xm4_t mask`, unsigned int gvl)
- `uint16xm8_t vnotv_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, `e16xm8_t mask`, unsigned int gvl)
- `uint32xm1_t vnotv_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, `e32xm1_t mask`, unsigned int gvl)
- `uint32xm2_t vnotv_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, `e32xm2_t mask`, unsigned int gvl)
- `uint32xm4_t vnotv_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, `e32xm4_t mask`, unsigned int gvl)
- `uint32xm8_t vnotv_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, `e32xm8_t mask`, unsigned int gvl)
- `uint64xm1_t vnotv_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, `e64xm1_t mask`, unsigned int gvl)

- `uint64xm2_t vnotv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vnotv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vnotv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vnotv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vnotv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vnotv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vnotv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = bitwise_not (a[element])
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.2.5 Elementwise vector-immediate bitwise-or

Instruction: ['vor.vi']

Prototypes:

- `int16xm1_t vorvi_int16xm1 (int16xm1_t a, const short b, unsigned int gvl)`
- `int16xm2_t vorvi_int16xm2 (int16xm2_t a, const short b, unsigned int gvl)`
- `int16xm4_t vorvi_int16xm4 (int16xm4_t a, const short b, unsigned int gvl)`
- `int16xm8_t vorvi_int16xm8 (int16xm8_t a, const short b, unsigned int gvl)`
- `int32xm1_t vorvi_int32xm1 (int32xm1_t a, const int b, unsigned int gvl)`
- `int32xm2_t vorvi_int32xm2 (int32xm2_t a, const int b, unsigned int gvl)`
- `int32xm4_t vorvi_int32xm4 (int32xm4_t a, const int b, unsigned int gvl)`
- `int32xm8_t vorvi_int32xm8 (int32xm8_t a, const int b, unsigned int gvl)`
- `int64xm1_t vorvi_int64xm1 (int64xm1_t a, const long b, unsigned int gvl)`
- `int64xm2_t vorvi_int64xm2 (int64xm2_t a, const long b, unsigned int gvl)`
- `int64xm4_t vorvi_int64xm4 (int64xm4_t a, const long b, unsigned int gvl)`
- `int64xm8_t vorvi_int64xm8 (int64xm8_t a, const long b, unsigned int gvl)`
- `int8xm1_t vorvi_int8xm1 (int8xm1_t a, const signed char b, unsigned int gvl)`
- `int8xm2_t vorvi_int8xm2 (int8xm2_t a, const signed char b, unsigned int gvl)`
- `int8xm4_t vorvi_int8xm4 (int8xm4_t a, const signed char b, unsigned int gvl)`

- `int8xm8_t vorvi_int8xm8 (int8xm8_t a, const signed char b, unsigned int gvl)`
- `uint16xm1_t vorvi_uint16xm1 (uint16xm1_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm2_t vorvi_uint16xm2 (uint16xm2_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm4_t vorvi_uint16xm4 (uint16xm4_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm8_t vorvi_uint16xm8 (uint16xm8_t a, const unsigned short b, unsigned int gvl)`
- `uint32xm1_t vorvi_uint32xm1 (uint32xm1_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm2_t vorvi_uint32xm2 (uint32xm2_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm4_t vorvi_uint32xm4 (uint32xm4_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm8_t vorvi_uint32xm8 (uint32xm8_t a, const unsigned int b, unsigned int gvl)`
- `uint64xm1_t vorvi_uint64xm1 (uint64xm1_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm2_t vorvi_uint64xm2 (uint64xm2_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm4_t vorvi_uint64xm4 (uint64xm4_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm8_t vorvi_uint64xm8 (uint64xm8_t a, const unsigned long b, unsigned int gvl)`
- `uint8xm1_t vorvi_uint8xm1 (uint8xm1_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm2_t vorvi_uint8xm2 (uint8xm2_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm4_t vorvi_uint8xm4 (uint8xm4_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm8_t vorvi_uint8xm8 (uint8xm8_t a, const unsigned char b, unsigned int gvl)`

#### Operation:

```
>>> for element = 0 to gvl - 1
    result[element] = bitwise_or (a[element], b)
result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- `int16xm1_t vorvi_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, const short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vorvi_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, const short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vorvi_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, const short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vorvi_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, const short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vorvi_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, const int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vorvi_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, const int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vorvi_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, const int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vorvi_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, const int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vorvi_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, const long b, e64xm1_t mask, unsigned int gvl)`

- `int64xm2_t vorvi_mask_int64xm2` (`int64xm2_t merge`, `int64xm2_t a`, const long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `int64xm4_t vorvi_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, const long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `int64xm8_t vorvi_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, const long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `int8xm1_t vorvi_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, const signed char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8xm2_t vorvi_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, const signed char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `int8xm4_t vorvi_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, const signed char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int8xm8_t vorvi_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, const signed char `b`, `e8xm8_t mask`, unsigned int `gvl`)
- `uint16xm1_t vorvi_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, const unsigned short `b`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint16xm2_t vorvi_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, const unsigned short `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint16xm4_t vorvi_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, const unsigned short `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vorvi_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, const unsigned short `b`, `e16xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vorvi_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, const unsigned int `b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vorvi_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, const unsigned int `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vorvi_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, const unsigned int `b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint32xm8_t vorvi_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, const unsigned int `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint64xm1_t vorvi_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, const unsigned long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vorvi_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, const unsigned long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vorvi_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, const unsigned long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vorvi_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, const unsigned long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vorvi_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, const unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vorvi_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, const unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vorvi_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, const unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vorvi_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, const unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = bitwise_or (a[element], b)
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.2.6 Elementwise vector-vector bitwise-or****Instruction:** ['vor.vv']**Prototypes:**

- *int16xm1\_t vorvv\_int16xm1 (int16xm1\_t a, int16xm1\_t b, unsigned int gvl)*
- *int16xm2\_t vorvv\_int16xm2 (int16xm2\_t a, int16xm2\_t b, unsigned int gvl)*
- *int16xm4\_t vorvv\_int16xm4 (int16xm4\_t a, int16xm4\_t b, unsigned int gvl)*
- *int16xm8\_t vorvv\_int16xm8 (int16xm8\_t a, int16xm8\_t b, unsigned int gvl)*
- *int32xm1\_t vorvv\_int32xm1 (int32xm1\_t a, int32xm1\_t b, unsigned int gvl)*
- *int32xm2\_t vorvv\_int32xm2 (int32xm2\_t a, int32xm2\_t b, unsigned int gvl)*
- *int32xm4\_t vorvv\_int32xm4 (int32xm4\_t a, int32xm4\_t b, unsigned int gvl)*
- *int32xm8\_t vorvv\_int32xm8 (int32xm8\_t a, int32xm8\_t b, unsigned int gvl)*
- *int64xm1\_t vorvv\_int64xm1 (int64xm1\_t a, int64xm1\_t b, unsigned int gvl)*
- *int64xm2\_t vorvv\_int64xm2 (int64xm2\_t a, int64xm2\_t b, unsigned int gvl)*
- *int64xm4\_t vorvv\_int64xm4 (int64xm4\_t a, int64xm4\_t b, unsigned int gvl)*
- *int64xm8\_t vorvv\_int64xm8 (int64xm8\_t a, int64xm8\_t b, unsigned int gvl)*
- *int8xm1\_t vorvv\_int8xm1 (int8xm1\_t a, int8xm1\_t b, unsigned int gvl)*
- *int8xm2\_t vorvv\_int8xm2 (int8xm2\_t a, int8xm2\_t b, unsigned int gvl)*
- *int8xm4\_t vorvv\_int8xm4 (int8xm4\_t a, int8xm4\_t b, unsigned int gvl)*
- *int8xm8\_t vorvv\_int8xm8 (int8xm8\_t a, int8xm8\_t b, unsigned int gvl)*
- *uint16xm1\_t vorvv\_uint16xm1 (uint16xm1\_t a, uint16xm1\_t b, unsigned int gvl)*
- *uint16xm2\_t vorvv\_uint16xm2 (uint16xm2\_t a, uint16xm2\_t b, unsigned int gvl)*
- *uint16xm4\_t vorvv\_uint16xm4 (uint16xm4\_t a, uint16xm4\_t b, unsigned int gvl)*
- *uint16xm8\_t vorvv\_uint16xm8 (uint16xm8\_t a, uint16xm8\_t b, unsigned int gvl)*
- *uint32xm1\_t vorvv\_uint32xm1 (uint32xm1\_t a, uint32xm1\_t b, unsigned int gvl)*
- *uint32xm2\_t vorvv\_uint32xm2 (uint32xm2\_t a, uint32xm2\_t b, unsigned int gvl)*
- *uint32xm4\_t vorvv\_uint32xm4 (uint32xm4\_t a, uint32xm4\_t b, unsigned int gvl)*
- *uint32xm8\_t vorvv\_uint32xm8 (uint32xm8\_t a, uint32xm8\_t b, unsigned int gvl)*
- *uint64xm1\_t vorvv\_uint64xm1 (uint64xm1\_t a, uint64xm1\_t b, unsigned int gvl)*
- *uint64xm2\_t vorvv\_uint64xm2 (uint64xm2\_t a, uint64xm2\_t b, unsigned int gvl)*



- `uint64xm4_t vorvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vorvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vorvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vorvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vorvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vorvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = bitwise_or (a[element], b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vorvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vorvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vorvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vorvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vorvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vorvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vorvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vorvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vorvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vorvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vorvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vorvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vorvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vorvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vorvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vorvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`



- `uint16xm1_t vorvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vorvv_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vorvv_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vorvv_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vorvv_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vorvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vorvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vorvv_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vorvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vorvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vorvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vorvv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vorvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vorvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vorvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vorvv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = bitwise_or (a[element], b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.2.7 Elementwise vector-scalar bitwise-or

Instruction: [`'vor.vx'`]

Prototypes:

- `int16xm1_t vorvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`

- `int16xm2_t vorvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vorvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vorvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vorvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vorvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vorvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vorvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vorvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vorvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vorvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vorvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vorvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vorvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vorvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vorvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`
- `uint16xm1_t vorvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vorvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vorvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vorvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vorvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vorvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vorvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vorvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vorvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vorvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vorvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vorvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vorvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vorvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vorvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vorvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = bitwise_or (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vorvx_mask_int16xm1` (`int16xm1_t` merge, `int16xm1_t` a, short `b`, `e16xm1_t` mask, unsigned int gvl)
- `int16xm2_t vorvx_mask_int16xm2` (`int16xm2_t` merge, `int16xm2_t` a, short `b`, `e16xm2_t` mask, unsigned int gvl)
- `int16xm4_t vorvx_mask_int16xm4` (`int16xm4_t` merge, `int16xm4_t` a, short `b`, `e16xm4_t` mask, unsigned int gvl)
- `int16xm8_t vorvx_mask_int16xm8` (`int16xm8_t` merge, `int16xm8_t` a, short `b`, `e16xm8_t` mask, unsigned int gvl)
- `int32xm1_t vorvx_mask_int32xm1` (`int32xm1_t` merge, `int32xm1_t` a, int `b`, `e32xm1_t` mask, unsigned int gvl)
- `int32xm2_t vorvx_mask_int32xm2` (`int32xm2_t` merge, `int32xm2_t` a, int `b`, `e32xm2_t` mask, unsigned int gvl)
- `int32xm4_t vorvx_mask_int32xm4` (`int32xm4_t` merge, `int32xm4_t` a, int `b`, `e32xm4_t` mask, unsigned int gvl)
- `int32xm8_t vorvx_mask_int32xm8` (`int32xm8_t` merge, `int32xm8_t` a, int `b`, `e32xm8_t` mask, unsigned int gvl)
- `int64xm1_t vorvx_mask_int64xm1` (`int64xm1_t` merge, `int64xm1_t` a, long `b`, `e64xm1_t` mask, unsigned int gvl)
- `int64xm2_t vorvx_mask_int64xm2` (`int64xm2_t` merge, `int64xm2_t` a, long `b`, `e64xm2_t` mask, unsigned int gvl)
- `int64xm4_t vorvx_mask_int64xm4` (`int64xm4_t` merge, `int64xm4_t` a, long `b`, `e64xm4_t` mask, unsigned int gvl)
- `int64xm8_t vorvx_mask_int64xm8` (`int64xm8_t` merge, `int64xm8_t` a, long `b`, `e64xm8_t` mask, unsigned int gvl)
- `int8xm1_t vorvx_mask_int8xm1` (`int8xm1_t` merge, `int8xm1_t` a, signed char `b`, `e8xm1_t` mask, unsigned int gvl)
- `int8xm2_t vorvx_mask_int8xm2` (`int8xm2_t` merge, `int8xm2_t` a, signed char `b`, `e8xm2_t` mask, unsigned int gvl)
- `int8xm4_t vorvx_mask_int8xm4` (`int8xm4_t` merge, `int8xm4_t` a, signed char `b`, `e8xm4_t` mask, unsigned int gvl)
- `int8xm8_t vorvx_mask_int8xm8` (`int8xm8_t` merge, `int8xm8_t` a, signed char `b`, `e8xm8_t` mask, unsigned int gvl)
- `uint16xm1_t vorvx_mask_uint16xm1` (`uint16xm1_t` merge, `uint16xm1_t` a, unsigned short `b`, `e16xm1_t` mask, unsigned int gvl)
- `uint16xm2_t vorvx_mask_uint16xm2` (`uint16xm2_t` merge, `uint16xm2_t` a, unsigned short `b`, `e16xm2_t` mask, unsigned int gvl)
- `uint16xm4_t vorvx_mask_uint16xm4` (`uint16xm4_t` merge, `uint16xm4_t` a, unsigned short `b`, `e16xm4_t` mask, unsigned int gvl)
- `uint16xm8_t vorvx_mask_uint16xm8` (`uint16xm8_t` merge, `uint16xm8_t` a, unsigned short `b`, `e16xm8_t` mask, unsigned int gvl)
- `uint32xm1_t vorvx_mask_uint32xm1` (`uint32xm1_t` merge, `uint32xm1_t` a, unsigned int `b`, `e32xm1_t` mask, unsigned int gvl)
- `uint32xm2_t vorvx_mask_uint32xm2` (`uint32xm2_t` merge, `uint32xm2_t` a, unsigned int `b`, `e32xm2_t` mask, unsigned int gvl)
- `uint32xm4_t vorvx_mask_uint32xm4` (`uint32xm4_t` merge, `uint32xm4_t` a, unsigned int `b`, `e32xm4_t` mask, unsigned int gvl)

- `uint32xm8_t vorvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vorvx_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vorvx_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vorvx_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vorvx_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vorvx_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vorvx_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vorvx_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vorvx_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = bitwise_or (a[element], b)
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.2.8 Elementwise vector-immediate logic shift left

**Instruction:** [`'vsll.vi'`]**Prototypes:**

- `int16xm1_t vsllvi_int16xm1 (int16xm1_t a, const unsigned short b, unsigned int gvl)`
- `int16xm2_t vsllvi_int16xm2 (int16xm2_t a, const unsigned short b, unsigned int gvl)`
- `int16xm4_t vsllvi_int16xm4 (int16xm4_t a, const unsigned short b, unsigned int gvl)`
- `int16xm8_t vsllvi_int16xm8 (int16xm8_t a, const unsigned short b, unsigned int gvl)`
- `int32xm1_t vsllvi_int32xm1 (int32xm1_t a, const unsigned int b, unsigned int gvl)`
- `int32xm2_t vsllvi_int32xm2 (int32xm2_t a, const unsigned int b, unsigned int gvl)`
- `int32xm4_t vsllvi_int32xm4 (int32xm4_t a, const unsigned int b, unsigned int gvl)`
- `int32xm8_t vsllvi_int32xm8 (int32xm8_t a, const unsigned int b, unsigned int gvl)`
- `int64xm1_t vsllvi_int64xm1 (int64xm1_t a, const unsigned long b, unsigned int gvl)`
- `int64xm2_t vsllvi_int64xm2 (int64xm2_t a, const unsigned long b, unsigned int gvl)`
- `int64xm4_t vsllvi_int64xm4 (int64xm4_t a, const unsigned long b, unsigned int gvl)`
- `int64xm8_t vsllvi_int64xm8 (int64xm8_t a, const unsigned long b, unsigned int gvl)`

- `int8xm1_t vsllvi_int8xm1 (int8xm1_t a, const unsigned char b, unsigned int gvl)`
- `int8xm2_t vsllvi_int8xm2 (int8xm2_t a, const unsigned char b, unsigned int gvl)`
- `int8xm4_t vsllvi_int8xm4 (int8xm4_t a, const unsigned char b, unsigned int gvl)`
- `int8xm8_t vsllvi_int8xm8 (int8xm8_t a, const unsigned char b, unsigned int gvl)`
- `uint16xm1_t vsllvi_uint16xm1 (uint16xm1_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm2_t vsllvi_uint16xm2 (uint16xm2_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm4_t vsllvi_uint16xm4 (uint16xm4_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm8_t vsllvi_uint16xm8 (uint16xm8_t a, const unsigned short b, unsigned int gvl)`
- `uint32xm1_t vsllvi_uint32xm1 (uint32xm1_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm2_t vsllvi_uint32xm2 (uint32xm2_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm4_t vsllvi_uint32xm4 (uint32xm4_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm8_t vsllvi_uint32xm8 (uint32xm8_t a, const unsigned int b, unsigned int gvl)`
- `uint64xm1_t vsllvi_uint64xm1 (uint64xm1_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm2_t vsllvi_uint64xm2 (uint64xm2_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm4_t vsllvi_uint64xm4 (uint64xm4_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm8_t vsllvi_uint64xm8 (uint64xm8_t a, const unsigned long b, unsigned int gvl)`
- `uint8xm1_t vsllvi_uint8xm1 (uint8xm1_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm2_t vsllvi_uint8xm2 (uint8xm2_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm4_t vsllvi_uint8xm4 (uint8xm4_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm8_t vsllvi_uint8xm8 (uint8xm8_t a, const unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = sll (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vsllvi_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, const unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vsllvi_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, const unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vsllvi_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, const unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vsllvi_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, const unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vsllvi_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, const unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsllvi_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, const unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsllvi_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, const unsigned int b, e32xm4_t mask, unsigned int gvl)`

- *int32xm8\_t vsllvi\_mask\_int32xm8* (*int32xm8\_t merge*, *int32xm8\_t a*, const unsigned int *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *int64xm1\_t vsllvi\_mask\_int64xm1* (*int64xm1\_t merge*, *int64xm1\_t a*, const unsigned long *b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *int64xm2\_t vsllvi\_mask\_int64xm2* (*int64xm2\_t merge*, *int64xm2\_t a*, const unsigned long *b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *int64xm4\_t vsllvi\_mask\_int64xm4* (*int64xm4\_t merge*, *int64xm4\_t a*, const unsigned long *b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *int64xm8\_t vsllvi\_mask\_int64xm8* (*int64xm8\_t merge*, *int64xm8\_t a*, const unsigned long *b*, *e64xm8\_t mask*, unsigned int *gvl*)
- *int8xm1\_t vsllvi\_mask\_int8xm1* (*int8xm1\_t merge*, *int8xm1\_t a*, const unsigned char *b*, *e8xm1\_t mask*, unsigned int *gvl*)
- *int8xm2\_t vsllvi\_mask\_int8xm2* (*int8xm2\_t merge*, *int8xm2\_t a*, const unsigned char *b*, *e8xm2\_t mask*, unsigned int *gvl*)
- *int8xm4\_t vsllvi\_mask\_int8xm4* (*int8xm4\_t merge*, *int8xm4\_t a*, const unsigned char *b*, *e8xm4\_t mask*, unsigned int *gvl*)
- *int8xm8\_t vsllvi\_mask\_int8xm8* (*int8xm8\_t merge*, *int8xm8\_t a*, const unsigned char *b*, *e8xm8\_t mask*, unsigned int *gvl*)
- *uint16xm1\_t vsllvi\_mask\_uint16xm1* (*uint16xm1\_t merge*, *uint16xm1\_t a*, const unsigned short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *uint16xm2\_t vsllvi\_mask\_uint16xm2* (*uint16xm2\_t merge*, *uint16xm2\_t a*, const unsigned short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *uint16xm4\_t vsllvi\_mask\_uint16xm4* (*uint16xm4\_t merge*, *uint16xm4\_t a*, const unsigned short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *uint16xm8\_t vsllvi\_mask\_uint16xm8* (*uint16xm8\_t merge*, *uint16xm8\_t a*, const unsigned short *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *uint32xm1\_t vsllvi\_mask\_uint32xm1* (*uint32xm1\_t merge*, *uint32xm1\_t a*, const unsigned int *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *uint32xm2\_t vsllvi\_mask\_uint32xm2* (*uint32xm2\_t merge*, *uint32xm2\_t a*, const unsigned int *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *uint32xm4\_t vsllvi\_mask\_uint32xm4* (*uint32xm4\_t merge*, *uint32xm4\_t a*, const unsigned int *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *uint32xm8\_t vsllvi\_mask\_uint32xm8* (*uint32xm8\_t merge*, *uint32xm8\_t a*, const unsigned int *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *uint64xm1\_t vsllvi\_mask\_uint64xm1* (*uint64xm1\_t merge*, *uint64xm1\_t a*, const unsigned long *b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *uint64xm2\_t vsllvi\_mask\_uint64xm2* (*uint64xm2\_t merge*, *uint64xm2\_t a*, const unsigned long *b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *uint64xm4\_t vsllvi\_mask\_uint64xm4* (*uint64xm4\_t merge*, *uint64xm4\_t a*, const unsigned long *b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *uint64xm8\_t vsllvi\_mask\_uint64xm8* (*uint64xm8\_t merge*, *uint64xm8\_t a*, const unsigned long *b*, *e64xm8\_t mask*, unsigned int *gvl*)
- *uint8xm1\_t vsllvi\_mask\_uint8xm1* (*uint8xm1\_t merge*, *uint8xm1\_t a*, const unsigned char *b*, *e8xm1\_t mask*, unsigned int *gvl*)
- *uint8xm2\_t vsllvi\_mask\_uint8xm2* (*uint8xm2\_t merge*, *uint8xm2\_t a*, const unsigned char *b*, *e8xm2\_t mask*, unsigned int *gvl*)



- `uint8xm4_t vsllvi_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, const unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vsllvi_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, const unsigned char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = sll (a[element], b)
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.2.9 Elementwise vector-vector logic shift left

**Instruction:** [`vsll.vv`']

**Prototypes:**

- `int16xm1_t vsllvv_int16xm1_uint16xm1 (int16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `int16xm2_t vsllvv_int16xm2_uint16xm2 (int16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `int16xm4_t vsllvv_int16xm4_uint16xm4 (int16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `int16xm8_t vsllvv_int16xm8_uint16xm8 (int16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `int32xm1_t vsllvv_int32xm1_uint32xm1 (int32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `int32xm2_t vsllvv_int32xm2_uint32xm2 (int32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `int32xm4_t vsllvv_int32xm4_uint32xm4 (int32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `int32xm8_t vsllvv_int32xm8_uint32xm8 (int32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `int64xm1_t vsllvv_int64xm1_uint64xm1 (int64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `int64xm2_t vsllvv_int64xm2_uint64xm2 (int64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `int64xm4_t vsllvv_int64xm4_uint64xm4 (int64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `int64xm8_t vsllvv_int64xm8_uint64xm8 (int64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `int8xm1_t vsllvv_int8xm1_uint8xm1 (int8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `int8xm2_t vsllvv_int8xm2_uint8xm2 (int8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `int8xm4_t vsllvv_int8xm4_uint8xm4 (int8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `int8xm8_t vsllvv_int8xm8_uint8xm8 (int8xm8_t a, uint8xm8_t b, unsigned int gvl)`
- `uint16xm1_t vsllvv_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vsllvv_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vsllvv_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vsllvv_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vsllvv_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vsllvv_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vsllvv_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`



- `uint32xm8_t vsllvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vsllvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vsllvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vsllvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vsllvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vsllvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vsllvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vsllvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vsllvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = sll (a[element], b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vsllvv_mask_int16xm1_uint16xm1 (int16xm1_t merge, int16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vsllvv_mask_int16xm2_uint16xm2 (int16xm2_t merge, int16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vsllvv_mask_int16xm4_uint16xm4 (int16xm4_t merge, int16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vsllvv_mask_int16xm8_uint16xm8 (int16xm8_t merge, int16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vsllvv_mask_int32xm1_uint32xm1 (int32xm1_t merge, int32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsllvv_mask_int32xm2_uint32xm2 (int32xm2_t merge, int32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsllvv_mask_int32xm4_uint32xm4 (int32xm4_t merge, int32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vsllvv_mask_int32xm8_uint32xm8 (int32xm8_t merge, int32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vsllvv_mask_int64xm1_uint64xm1 (int64xm1_t merge, int64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vsllvv_mask_int64xm2_uint64xm2 (int64xm2_t merge, int64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`

- `int64xm4_t vsllvv_mask_int64xm4_uint64xm4` (`int64xm4_t` merge, `int64xm4_t` *a*, `uint64xm4_t` *b*, `e64xm4_t` *mask*, unsigned int gvl)
- `int64xm8_t vsllvv_mask_int64xm8_uint64xm8` (`int64xm8_t` merge, `int64xm8_t` *a*, `uint64xm8_t` *b*, `e64xm8_t` *mask*, unsigned int gvl)
- `int8xm1_t vsllvv_mask_int8xm1_uint8xm1` (`int8xm1_t` merge, `int8xm1_t` *a*, `uint8xm1_t` *b*, `e8xm1_t` *mask*, unsigned int gvl)
- `int8xm2_t vsllvv_mask_int8xm2_uint8xm2` (`int8xm2_t` merge, `int8xm2_t` *a*, `uint8xm2_t` *b*, `e8xm2_t` *mask*, unsigned int gvl)
- `int8xm4_t vsllvv_mask_int8xm4_uint8xm4` (`int8xm4_t` merge, `int8xm4_t` *a*, `uint8xm4_t` *b*, `e8xm4_t` *mask*, unsigned int gvl)
- `int8xm8_t vsllvv_mask_int8xm8_uint8xm8` (`int8xm8_t` merge, `int8xm8_t` *a*, `uint8xm8_t` *b*, `e8xm8_t` *mask*, unsigned int gvl)
- `uint16xm1_t vsllvv_mask_uint16xm1` (`uint16xm1_t` merge, `uint16xm1_t` *a*, `uint16xm1_t` *b*, `e16xm1_t` *mask*, unsigned int gvl)
- `uint16xm2_t vsllvv_mask_uint16xm2` (`uint16xm2_t` merge, `uint16xm2_t` *a*, `uint16xm2_t` *b*, `e16xm2_t` *mask*, unsigned int gvl)
- `uint16xm4_t vsllvv_mask_uint16xm4` (`uint16xm4_t` merge, `uint16xm4_t` *a*, `uint16xm4_t` *b*, `e16xm4_t` *mask*, unsigned int gvl)
- `uint16xm8_t vsllvv_mask_uint16xm8` (`uint16xm8_t` merge, `uint16xm8_t` *a*, `uint16xm8_t` *b*, `e16xm8_t` *mask*, unsigned int gvl)
- `uint32xm1_t vsllvv_mask_uint32xm1` (`uint32xm1_t` merge, `uint32xm1_t` *a*, `uint32xm1_t` *b*, `e32xm1_t` *mask*, unsigned int gvl)
- `uint32xm2_t vsllvv_mask_uint32xm2` (`uint32xm2_t` merge, `uint32xm2_t` *a*, `uint32xm2_t` *b*, `e32xm2_t` *mask*, unsigned int gvl)
- `uint32xm4_t vsllvv_mask_uint32xm4` (`uint32xm4_t` merge, `uint32xm4_t` *a*, `uint32xm4_t` *b*, `e32xm4_t` *mask*, unsigned int gvl)
- `uint32xm8_t vsllvv_mask_uint32xm8` (`uint32xm8_t` merge, `uint32xm8_t` *a*, `uint32xm8_t` *b*, `e32xm8_t` *mask*, unsigned int gvl)
- `uint64xm1_t vsllvv_mask_uint64xm1` (`uint64xm1_t` merge, `uint64xm1_t` *a*, `uint64xm1_t` *b*, `e64xm1_t` *mask*, unsigned int gvl)
- `uint64xm2_t vsllvv_mask_uint64xm2` (`uint64xm2_t` merge, `uint64xm2_t` *a*, `uint64xm2_t` *b*, `e64xm2_t` *mask*, unsigned int gvl)
- `uint64xm4_t vsllvv_mask_uint64xm4` (`uint64xm4_t` merge, `uint64xm4_t` *a*, `uint64xm4_t` *b*, `e64xm4_t` *mask*, unsigned int gvl)
- `uint64xm8_t vsllvv_mask_uint64xm8` (`uint64xm8_t` merge, `uint64xm8_t` *a*, `uint64xm8_t` *b*, `e64xm8_t` *mask*, unsigned int gvl)
- `uint8xm1_t vsllvv_mask_uint8xm1` (`uint8xm1_t` merge, `uint8xm1_t` *a*, `uint8xm1_t` *b*, `e8xm1_t` *mask*, unsigned int gvl)
- `uint8xm2_t vsllvv_mask_uint8xm2` (`uint8xm2_t` merge, `uint8xm2_t` *a*, `uint8xm2_t` *b*, `e8xm2_t` *mask*, unsigned int gvl)
- `uint8xm4_t vsllvv_mask_uint8xm4` (`uint8xm4_t` merge, `uint8xm4_t` *a*, `uint8xm4_t` *b*, `e8xm4_t` *mask*, unsigned int gvl)
- `uint8xm8_t vsllvv_mask_uint8xm8` (`uint8xm8_t` merge, `uint8xm8_t` *a*, `uint8xm8_t` *b*, `e8xm8_t` *mask*, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = sll (a[element], b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.2.10 Elementwise vector-scalar logic shift left

**Instruction:** [`'vsll.vx'`]

**Prototypes:**

- `int16xm1_t vsllvx_int16xm1 (int16xm1_t a, unsigned short b, unsigned int gvl)`
- `int16xm2_t vsllvx_int16xm2 (int16xm2_t a, unsigned short b, unsigned int gvl)`
- `int16xm4_t vsllvx_int16xm4 (int16xm4_t a, unsigned short b, unsigned int gvl)`
- `int16xm8_t vsllvx_int16xm8 (int16xm8_t a, unsigned short b, unsigned int gvl)`
- `int32xm1_t vsllvx_int32xm1 (int32xm1_t a, unsigned int b, unsigned int gvl)`
- `int32xm2_t vsllvx_int32xm2 (int32xm2_t a, unsigned int b, unsigned int gvl)`
- `int32xm4_t vsllvx_int32xm4 (int32xm4_t a, unsigned int b, unsigned int gvl)`
- `int32xm8_t vsllvx_int32xm8 (int32xm8_t a, unsigned int b, unsigned int gvl)`
- `int64xm1_t vsllvx_int64xm1 (int64xm1_t a, unsigned long b, unsigned int gvl)`
- `int64xm2_t vsllvx_int64xm2 (int64xm2_t a, unsigned long b, unsigned int gvl)`
- `int64xm4_t vsllvx_int64xm4 (int64xm4_t a, unsigned long b, unsigned int gvl)`
- `int64xm8_t vsllvx_int64xm8 (int64xm8_t a, unsigned long b, unsigned int gvl)`
- `int8xm1_t vsllvx_int8xm1 (int8xm1_t a, unsigned char b, unsigned int gvl)`
- `int8xm2_t vsllvx_int8xm2 (int8xm2_t a, unsigned char b, unsigned int gvl)`
- `int8xm4_t vsllvx_int8xm4 (int8xm4_t a, unsigned char b, unsigned int gvl)`
- `int8xm8_t vsllvx_int8xm8 (int8xm8_t a, unsigned char b, unsigned int gvl)`
- `uint16xm1_t vsllvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vsllvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vsllvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vsllvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vsllvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vsllvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vsllvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vsllvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vsllvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vsllvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vsllvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`

- `uint64xm8_t vsllvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vsllvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vsllvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vsllvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vsllvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = sll (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vsllvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vsllvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vsllvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vsllvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vsllvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsllvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsllvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vsllvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vsllvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vsllvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vsllvx_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vsllvx_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vsllvx_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vsllvx_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vsllvx_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vsllvx_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vsllvx_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`

- `uint16xm2_t vsllvx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vsllvx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vsllvx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vsllvx_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vsllvx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vsllvx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vsllvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vsllvx_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vsllvx_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vsllvx_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vsllvx_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vsllvx_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vsllvx_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vsllvx_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vsllvx_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = sll (a[element], b)
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.2.11 Elementwise vector-immediate arithmetic shift right

**Instruction:** [`vsra.vi`']**Prototypes:**

- `int16xm1_t vsravi_int16xm1 (int16xm1_t a, const unsigned short b, unsigned int gvl)`
- `int16xm2_t vsravi_int16xm2 (int16xm2_t a, const unsigned short b, unsigned int gvl)`
- `int16xm4_t vsravi_int16xm4 (int16xm4_t a, const unsigned short b, unsigned int gvl)`

- `int16xm8_t vsravi_int16xm8 (int16xm8_t a, const unsigned short b, unsigned int gvl)`
- `int32xm1_t vsravi_int32xm1 (int32xm1_t a, const unsigned int b, unsigned int gvl)`
- `int32xm2_t vsravi_int32xm2 (int32xm2_t a, const unsigned int b, unsigned int gvl)`
- `int32xm4_t vsravi_int32xm4 (int32xm4_t a, const unsigned int b, unsigned int gvl)`
- `int32xm8_t vsravi_int32xm8 (int32xm8_t a, const unsigned int b, unsigned int gvl)`
- `int64xm1_t vsravi_int64xm1 (int64xm1_t a, const unsigned long b, unsigned int gvl)`
- `int64xm2_t vsravi_int64xm2 (int64xm2_t a, const unsigned long b, unsigned int gvl)`
- `int64xm4_t vsravi_int64xm4 (int64xm4_t a, const unsigned long b, unsigned int gvl)`
- `int64xm8_t vsravi_int64xm8 (int64xm8_t a, const unsigned long b, unsigned int gvl)`
- `int8xm1_t vsravi_int8xm1 (int8xm1_t a, const unsigned char b, unsigned int gvl)`
- `int8xm2_t vsravi_int8xm2 (int8xm2_t a, const unsigned char b, unsigned int gvl)`
- `int8xm4_t vsravi_int8xm4 (int8xm4_t a, const unsigned char b, unsigned int gvl)`
- `int8xm8_t vsravi_int8xm8 (int8xm8_t a, const unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = sra (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vsravi_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, const unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vsravi_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, const unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vsravi_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, const unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vsravi_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, const unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vsravi_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, const unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsravi_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, const unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsravi_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, const unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vsravi_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, const unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vsravi_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, const unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vsravi_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, const unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vsravi_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, const unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vsravi_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, const unsigned long b, e64xm8_t mask, unsigned int gvl)`



- `int8xm1_t vsravi_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, const unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vsravi_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, const unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vsravi_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, const unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vsravi_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, const unsigned char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = sra (a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.2.12 Elementwise vector-vector arithmetic shift right

**Instruction:** ['vsra.vv']

**Prototypes:**

- `int16xm1_t vsravv_int16xm1_uint16xm1 (int16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `int16xm2_t vsravv_int16xm2_uint16xm2 (int16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `int16xm4_t vsravv_int16xm4_uint16xm4 (int16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `int16xm8_t vsravv_int16xm8_uint16xm8 (int16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `int32xm1_t vsravv_int32xm1_uint32xm1 (int32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `int32xm2_t vsravv_int32xm2_uint32xm2 (int32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `int32xm4_t vsravv_int32xm4_uint32xm4 (int32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `int32xm8_t vsravv_int32xm8_uint32xm8 (int32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `int64xm1_t vsravv_int64xm1_uint64xm1 (int64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `int64xm2_t vsravv_int64xm2_uint64xm2 (int64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `int64xm4_t vsravv_int64xm4_uint64xm4 (int64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `int64xm8_t vsravv_int64xm8_uint64xm8 (int64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `int8xm1_t vsravv_int8xm1_uint8xm1 (int8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `int8xm2_t vsravv_int8xm2_uint8xm2 (int8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `int8xm4_t vsravv_int8xm4_uint8xm4 (int8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `int8xm8_t vsravv_int8xm8_uint8xm8 (int8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = sra (a[element], b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vsravv_mask_int16xm1_uint16xm1 (int16xm1_t merge, int16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vsravv_mask_int16xm2_uint16xm2 (int16xm2_t merge, int16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vsravv_mask_int16xm4_uint16xm4 (int16xm4_t merge, int16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vsravv_mask_int16xm8_uint16xm8 (int16xm8_t merge, int16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vsravv_mask_int32xm1_uint32xm1 (int32xm1_t merge, int32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsravv_mask_int32xm2_uint32xm2 (int32xm2_t merge, int32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsravv_mask_int32xm4_uint32xm4 (int32xm4_t merge, int32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vsravv_mask_int32xm8_uint32xm8 (int32xm8_t merge, int32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vsravv_mask_int64xm1_uint64xm1 (int64xm1_t merge, int64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vsravv_mask_int64xm2_uint64xm2 (int64xm2_t merge, int64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vsravv_mask_int64xm4_uint64xm4 (int64xm4_t merge, int64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vsravv_mask_int64xm8_uint64xm8 (int64xm8_t merge, int64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vsravv_mask_int8xm1_uint8xm1 (int8xm1_t merge, int8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vsravv_mask_int8xm2_uint8xm2 (int8xm2_t merge, int8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vsravv_mask_int8xm4_uint8xm4 (int8xm4_t merge, int8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vsravv_mask_int8xm8_uint8xm8 (int8xm8_t merge, int8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = sra (a[element], b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.2.13 Elementwise vector-scalar arithmetic shift right

**Instruction:** ['vsra.vx']

**Prototypes:**

- *int16xm1\_t vsravx\_int16xm1* (*int16xm1\_t a*, unsigned short *b*, unsigned int *gvl*)
- *int16xm2\_t vsravx\_int16xm2* (*int16xm2\_t a*, unsigned short *b*, unsigned int *gvl*)
- *int16xm4\_t vsravx\_int16xm4* (*int16xm4\_t a*, unsigned short *b*, unsigned int *gvl*)
- *int16xm8\_t vsravx\_int16xm8* (*int16xm8\_t a*, unsigned short *b*, unsigned int *gvl*)
- *int32xm1\_t vsravx\_int32xm1* (*int32xm1\_t a*, unsigned int *b*, unsigned int *gvl*)
- *int32xm2\_t vsravx\_int32xm2* (*int32xm2\_t a*, unsigned int *b*, unsigned int *gvl*)
- *int32xm4\_t vsravx\_int32xm4* (*int32xm4\_t a*, unsigned int *b*, unsigned int *gvl*)
- *int32xm8\_t vsravx\_int32xm8* (*int32xm8\_t a*, unsigned int *b*, unsigned int *gvl*)
- *int64xm1\_t vsravx\_int64xm1* (*int64xm1\_t a*, unsigned long *b*, unsigned int *gvl*)
- *int64xm2\_t vsravx\_int64xm2* (*int64xm2\_t a*, unsigned long *b*, unsigned int *gvl*)
- *int64xm4\_t vsravx\_int64xm4* (*int64xm4\_t a*, unsigned long *b*, unsigned int *gvl*)
- *int64xm8\_t vsravx\_int64xm8* (*int64xm8\_t a*, unsigned long *b*, unsigned int *gvl*)
- *int8xm1\_t vsravx\_int8xm1* (*int8xm1\_t a*, unsigned char *b*, unsigned int *gvl*)
- *int8xm2\_t vsravx\_int8xm2* (*int8xm2\_t a*, unsigned char *b*, unsigned int *gvl*)
- *int8xm4\_t vsravx\_int8xm4* (*int8xm4\_t a*, unsigned char *b*, unsigned int *gvl*)
- *int8xm8\_t vsravx\_int8xm8* (*int8xm8\_t a*, unsigned char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = sra (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vsravx\_mask\_int16xm1* (*int16xm1\_t merge*, *int16xm1\_t a*, unsigned short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vsravx\_mask\_int16xm2* (*int16xm2\_t merge*, *int16xm2\_t a*, unsigned short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vsravx\_mask\_int16xm4* (*int16xm4\_t merge*, *int16xm4\_t a*, unsigned short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vsravx\_mask\_int16xm8* (*int16xm8\_t merge*, *int16xm8\_t a*, unsigned short *b*, *e16xm8\_t mask*, unsigned int *gvl*)

- `int32xm1_t vsravx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsravx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsravx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vsravx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vsravx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vsravx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vsravx_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vsravx_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vsravx_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vsravx_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vsravx_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vsravx_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = sra (a[element], b)
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.2.14 Elementwise vector-immediate logic shift right

Instruction: ['vsrl.vi']

Prototypes:

- `uint16xm1_t vsrlvi_uint16xm1 (uint16xm1_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm2_t vsrlvi_uint16xm2 (uint16xm2_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm4_t vsrlvi_uint16xm4 (uint16xm4_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm8_t vsrlvi_uint16xm8 (uint16xm8_t a, const unsigned short b, unsigned int gvl)`
- `uint32xm1_t vsrlvi_uint32xm1 (uint32xm1_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm2_t vsrlvi_uint32xm2 (uint32xm2_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm4_t vsrlvi_uint32xm4 (uint32xm4_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm8_t vsrlvi_uint32xm8 (uint32xm8_t a, const unsigned int b, unsigned int gvl)`

- `uint64xm1_t vsrlvi_uint64xm1 (uint64xm1_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm2_t vsrlvi_uint64xm2 (uint64xm2_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm4_t vsrlvi_uint64xm4 (uint64xm4_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm8_t vsrlvi_uint64xm8 (uint64xm8_t a, const unsigned long b, unsigned int gvl)`
- `uint8xm1_t vsrlvi_uint8xm1 (uint8xm1_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm2_t vsrlvi_uint8xm2 (uint8xm2_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm4_t vsrlvi_uint8xm4 (uint8xm4_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm8_t vsrlvi_uint8xm8 (uint8xm8_t a, const unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = srl (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vsrlvi_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, const unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vsrlvi_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, const unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vsrlvi_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, const unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vsrlvi_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, const unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vsrlvi_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, const unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vsrlvi_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, const unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vsrlvi_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, const unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vsrlvi_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, const unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vsrlvi_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, const unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vsrlvi_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, const unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vsrlvi_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, const unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vsrlvi_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, const unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vsrlvi_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, const unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vsrlvi_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, const unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vsrlvi_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, const unsigned char b, e8xm4_t mask, unsigned int gvl)`

- `uint8xm8_t vsrlvi_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, const unsigned char b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = srl (a[element], b)
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.2.15 Elementwise vector-vector logic shift right

Instruction: ['vsrl.vv']

Prototypes:

- `uint16xm1_t vsrlvv_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vsrlvv_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vsrlvv_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vsrlvv_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vsrlvv_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vsrlvv_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vsrlvv_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vsrlvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vsrlvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vsrlvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vsrlvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vsrlvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vsrlvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vsrlvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vsrlvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vsrlvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

Operation:

```
>>> for element = 0 to gvl - 1
    result[element] = srl (a[element], b[element])
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `uint16xm1_t vsrlvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vsrlvv_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`



- `uint16xm4_t vsrlvv_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, `uint16xm4_t b`, `e16xm4_t mask`, unsigned int gvl)
- `uint16xm8_t vsrlvv_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, `uint16xm8_t b`, `e16xm8_t mask`, unsigned int gvl)
- `uint32xm1_t vsrlvv_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, `uint32xm1_t b`, `e32xm1_t mask`, unsigned int gvl)
- `uint32xm2_t vsrlvv_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, `uint32xm2_t b`, `e32xm2_t mask`, unsigned int gvl)
- `uint32xm4_t vsrlvv_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, `uint32xm4_t b`, `e32xm4_t mask`, unsigned int gvl)
- `uint32xm8_t vsrlvv_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, `uint32xm8_t b`, `e32xm8_t mask`, unsigned int gvl)
- `uint64xm1_t vsrlvv_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, `uint64xm1_t b`, `e64xm1_t mask`, unsigned int gvl)
- `uint64xm2_t vsrlvv_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, `uint64xm2_t b`, `e64xm2_t mask`, unsigned int gvl)
- `uint64xm4_t vsrlvv_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, `uint64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)
- `uint64xm8_t vsrlvv_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, `uint64xm8_t b`, `e64xm8_t mask`, unsigned int gvl)
- `uint8xm1_t vsrlvv_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int gvl)
- `uint8xm2_t vsrlvv_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int gvl)
- `uint8xm4_t vsrlvv_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int gvl)
- `uint8xm8_t vsrlvv_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, `uint8xm8_t b`, `e8xm8_t mask`, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = srl (a[element], b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.2.16 Elementwise vector-scalar logic shift right

**Instruction:** [`'vsrl.vx'`]**Prototypes:**

- `uint16xm1_t vsrlvx_uint16xm1` (`uint16xm1_t a`, unsigned short `b`, unsigned int gvl)
- `uint16xm2_t vsrlvx_uint16xm2` (`uint16xm2_t a`, unsigned short `b`, unsigned int gvl)
- `uint16xm4_t vsrlvx_uint16xm4` (`uint16xm4_t a`, unsigned short `b`, unsigned int gvl)
- `uint16xm8_t vsrlvx_uint16xm8` (`uint16xm8_t a`, unsigned short `b`, unsigned int gvl)
- `uint32xm1_t vsrlvx_uint32xm1` (`uint32xm1_t a`, unsigned int `b`, unsigned int gvl)

- `uint32xm2_t vsrlvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vsrlvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vsrlvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vsrlvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vsrlvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vsrlvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vsrlvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vsrlvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vsrlvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vsrlvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vsrlvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = srl (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vsrlvx_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vsrlvx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vsrlvx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vsrlvx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vsrlvx_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vsrlvx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vsrlvx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vsrlvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vsrlvx_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vsrlvx_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vsrlvx_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vsrlvx_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vsrlvx_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`

- `uint8xm2_t vsrlvx_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vsrlvx_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vsrlvx_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = srl (a[element], b)
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.2.17 Elementwise vector-immediate bitwise-xor****Instruction:** ['v<sub>xor</sub>.vi']**Prototypes:**

- `int16xm1_t vxorvi_int16xm1 (int16xm1_t a, const short b, unsigned int gvl)`
- `int16xm2_t vxorvi_int16xm2 (int16xm2_t a, const short b, unsigned int gvl)`
- `int16xm4_t vxorvi_int16xm4 (int16xm4_t a, const short b, unsigned int gvl)`
- `int16xm8_t vxorvi_int16xm8 (int16xm8_t a, const short b, unsigned int gvl)`
- `int32xm1_t vxorvi_int32xm1 (int32xm1_t a, const int b, unsigned int gvl)`
- `int32xm2_t vxorvi_int32xm2 (int32xm2_t a, const int b, unsigned int gvl)`
- `int32xm4_t vxorvi_int32xm4 (int32xm4_t a, const int b, unsigned int gvl)`
- `int32xm8_t vxorvi_int32xm8 (int32xm8_t a, const int b, unsigned int gvl)`
- `int64xm1_t vxorvi_int64xm1 (int64xm1_t a, const long b, unsigned int gvl)`
- `int64xm2_t vxorvi_int64xm2 (int64xm2_t a, const long b, unsigned int gvl)`
- `int64xm4_t vxorvi_int64xm4 (int64xm4_t a, const long b, unsigned int gvl)`
- `int64xm8_t vxorvi_int64xm8 (int64xm8_t a, const long b, unsigned int gvl)`
- `int8xm1_t vxorvi_int8xm1 (int8xm1_t a, const signed char b, unsigned int gvl)`
- `int8xm2_t vxorvi_int8xm2 (int8xm2_t a, const signed char b, unsigned int gvl)`
- `int8xm4_t vxorvi_int8xm4 (int8xm4_t a, const signed char b, unsigned int gvl)`
- `int8xm8_t vxorvi_int8xm8 (int8xm8_t a, const signed char b, unsigned int gvl)`
- `uint16xm1_t vxorvi_uint16xm1 (uint16xm1_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm2_t vxorvi_uint16xm2 (uint16xm2_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm4_t vxorvi_uint16xm4 (uint16xm4_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm8_t vxorvi_uint16xm8 (uint16xm8_t a, const unsigned short b, unsigned int gvl)`
- `uint32xm1_t vxorvi_uint32xm1 (uint32xm1_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm2_t vxorvi_uint32xm2 (uint32xm2_t a, const unsigned int b, unsigned int gvl)`

- `uint32xm4_t vxorvi_uint32xm4 (uint32xm4_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm8_t vxorvi_uint32xm8 (uint32xm8_t a, const unsigned int b, unsigned int gvl)`
- `uint64xm1_t vxorvi_uint64xm1 (uint64xm1_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm2_t vxorvi_uint64xm2 (uint64xm2_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm4_t vxorvi_uint64xm4 (uint64xm4_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm8_t vxorvi_uint64xm8 (uint64xm8_t a, const unsigned long b, unsigned int gvl)`
- `uint8xm1_t vxorvi_uint8xm1 (uint8xm1_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm2_t vxorvi_uint8xm2 (uint8xm2_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm4_t vxorvi_uint8xm4 (uint8xm4_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm8_t vxorvi_uint8xm8 (uint8xm8_t a, const unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = bitwise_xor (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vxorvi_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, const short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vxorvi_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, const short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vxorvi_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, const short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vxorvi_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, const short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vxorvi_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, const int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vxorvi_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, const int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vxorvi_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, const int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vxorvi_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, const int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vxorvi_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, const long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vxorvi_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, const long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vxorvi_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, const long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vxorvi_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, const long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vxorvi_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, const signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vxorvi_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, const signed char b, e8xm2_t mask, unsigned int gvl)`

- `int8xm4_t vxorvi_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, const signed char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vxorvi_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, const signed char b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vxorvi_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, const unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vxorvi_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, const unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vxorvi_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, const unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vxorvi_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, const unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vxorvi_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, const unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vxorvi_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, const unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vxorvi_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, const unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vxorvi_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, const unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vxorvi_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, const unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vxorvi_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, const unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vxorvi_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, const unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vxorvi_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, const unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vxorvi_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, const unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vxorvi_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, const unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vxorvi_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, const unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vxorvi_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, const unsigned char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = bitwise_xor (a[element], b)
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.2.18 Elementwise vector-vector bitwise-xor

**Instruction:** ['vxor.vv']

**Prototypes:**

- *int16xm1\_t vxorvv\_int16xm1* (*int16xm1\_t a, int16xm1\_t b*, unsigned int gvl)
- *int16xm2\_t vxorvv\_int16xm2* (*int16xm2\_t a, int16xm2\_t b*, unsigned int gvl)
- *int16xm4\_t vxorvv\_int16xm4* (*int16xm4\_t a, int16xm4\_t b*, unsigned int gvl)
- *int16xm8\_t vxorvv\_int16xm8* (*int16xm8\_t a, int16xm8\_t b*, unsigned int gvl)
- *int32xm1\_t vxorvv\_int32xm1* (*int32xm1\_t a, int32xm1\_t b*, unsigned int gvl)
- *int32xm2\_t vxorvv\_int32xm2* (*int32xm2\_t a, int32xm2\_t b*, unsigned int gvl)
- *int32xm4\_t vxorvv\_int32xm4* (*int32xm4\_t a, int32xm4\_t b*, unsigned int gvl)
- *int32xm8\_t vxorvv\_int32xm8* (*int32xm8\_t a, int32xm8\_t b*, unsigned int gvl)
- *int64xm1\_t vxorvv\_int64xm1* (*int64xm1\_t a, int64xm1\_t b*, unsigned int gvl)
- *int64xm2\_t vxorvv\_int64xm2* (*int64xm2\_t a, int64xm2\_t b*, unsigned int gvl)
- *int64xm4\_t vxorvv\_int64xm4* (*int64xm4\_t a, int64xm4\_t b*, unsigned int gvl)
- *int64xm8\_t vxorvv\_int64xm8* (*int64xm8\_t a, int64xm8\_t b*, unsigned int gvl)
- *int8xm1\_t vxorvv\_int8xm1* (*int8xm1\_t a, int8xm1\_t b*, unsigned int gvl)
- *int8xm2\_t vxorvv\_int8xm2* (*int8xm2\_t a, int8xm2\_t b*, unsigned int gvl)
- *int8xm4\_t vxorvv\_int8xm4* (*int8xm4\_t a, int8xm4\_t b*, unsigned int gvl)
- *int8xm8\_t vxorvv\_int8xm8* (*int8xm8\_t a, int8xm8\_t b*, unsigned int gvl)
- *uint16xm1\_t vxorvv\_uint16xm1* (*uint16xm1\_t a, uint16xm1\_t b*, unsigned int gvl)
- *uint16xm2\_t vxorvv\_uint16xm2* (*uint16xm2\_t a, uint16xm2\_t b*, unsigned int gvl)
- *uint16xm4\_t vxorvv\_uint16xm4* (*uint16xm4\_t a, uint16xm4\_t b*, unsigned int gvl)
- *uint16xm8\_t vxorvv\_uint16xm8* (*uint16xm8\_t a, uint16xm8\_t b*, unsigned int gvl)
- *uint32xm1\_t vxorvv\_uint32xm1* (*uint32xm1\_t a, uint32xm1\_t b*, unsigned int gvl)
- *uint32xm2\_t vxorvv\_uint32xm2* (*uint32xm2\_t a, uint32xm2\_t b*, unsigned int gvl)
- *uint32xm4\_t vxorvv\_uint32xm4* (*uint32xm4\_t a, uint32xm4\_t b*, unsigned int gvl)
- *uint32xm8\_t vxorvv\_uint32xm8* (*uint32xm8\_t a, uint32xm8\_t b*, unsigned int gvl)
- *uint64xm1\_t vxorvv\_uint64xm1* (*uint64xm1\_t a, uint64xm1\_t b*, unsigned int gvl)
- *uint64xm2\_t vxorvv\_uint64xm2* (*uint64xm2\_t a, uint64xm2\_t b*, unsigned int gvl)
- *uint64xm4\_t vxorvv\_uint64xm4* (*uint64xm4\_t a, uint64xm4\_t b*, unsigned int gvl)
- *uint64xm8\_t vxorvv\_uint64xm8* (*uint64xm8\_t a, uint64xm8\_t b*, unsigned int gvl)
- *uint8xm1\_t vxorvv\_uint8xm1* (*uint8xm1\_t a, uint8xm1\_t b*, unsigned int gvl)
- *uint8xm2\_t vxorvv\_uint8xm2* (*uint8xm2\_t a, uint8xm2\_t b*, unsigned int gvl)
- *uint8xm4\_t vxorvv\_uint8xm4* (*uint8xm4\_t a, uint8xm4\_t b*, unsigned int gvl)
- *uint8xm8\_t vxorvv\_uint8xm8* (*uint8xm8\_t a, uint8xm8\_t b*, unsigned int gvl)

**Operation:**



```
>>> for element = 0 to gvl - 1
    result[element] = bitwise_xor (a[element], b[element])
result[gvl : VLMAX] = 0
```

### Masked prototypes:

- *int16xm1\_t vxorvv\_mask\_int16xm1* (*int16xm1\_t merge, int16xm1\_t a, int16xm1\_t b, e16xm1\_t mask*, unsigned int gvl)
- *int16xm2\_t vxorvv\_mask\_int16xm2* (*int16xm2\_t merge, int16xm2\_t a, int16xm2\_t b, e16xm2\_t mask*, unsigned int gvl)
- *int16xm4\_t vxorvv\_mask\_int16xm4* (*int16xm4\_t merge, int16xm4\_t a, int16xm4\_t b, e16xm4\_t mask*, unsigned int gvl)
- *int16xm8\_t vxorvv\_mask\_int16xm8* (*int16xm8\_t merge, int16xm8\_t a, int16xm8\_t b, e16xm8\_t mask*, unsigned int gvl)
- *int32xm1\_t vxorvv\_mask\_int32xm1* (*int32xm1\_t merge, int32xm1\_t a, int32xm1\_t b, e32xm1\_t mask*, unsigned int gvl)
- *int32xm2\_t vxorvv\_mask\_int32xm2* (*int32xm2\_t merge, int32xm2\_t a, int32xm2\_t b, e32xm2\_t mask*, unsigned int gvl)
- *int32xm4\_t vxorvv\_mask\_int32xm4* (*int32xm4\_t merge, int32xm4\_t a, int32xm4\_t b, e32xm4\_t mask*, unsigned int gvl)
- *int32xm8\_t vxorvv\_mask\_int32xm8* (*int32xm8\_t merge, int32xm8\_t a, int32xm8\_t b, e32xm8\_t mask*, unsigned int gvl)
- *int64xm1\_t vxorvv\_mask\_int64xm1* (*int64xm1\_t merge, int64xm1\_t a, int64xm1\_t b, e64xm1\_t mask*, unsigned int gvl)
- *int64xm2\_t vxorvv\_mask\_int64xm2* (*int64xm2\_t merge, int64xm2\_t a, int64xm2\_t b, e64xm2\_t mask*, unsigned int gvl)
- *int64xm4\_t vxorvv\_mask\_int64xm4* (*int64xm4\_t merge, int64xm4\_t a, int64xm4\_t b, e64xm4\_t mask*, unsigned int gvl)
- *int64xm8\_t vxorvv\_mask\_int64xm8* (*int64xm8\_t merge, int64xm8\_t a, int64xm8\_t b, e64xm8\_t mask*, unsigned int gvl)
- *int8xm1\_t vxorvv\_mask\_int8xm1* (*int8xm1\_t merge, int8xm1\_t a, int8xm1\_t b, e8xm1\_t mask*, unsigned int gvl)
- *int8xm2\_t vxorvv\_mask\_int8xm2* (*int8xm2\_t merge, int8xm2\_t a, int8xm2\_t b, e8xm2\_t mask*, unsigned int gvl)
- *int8xm4\_t vxorvv\_mask\_int8xm4* (*int8xm4\_t merge, int8xm4\_t a, int8xm4\_t b, e8xm4\_t mask*, unsigned int gvl)
- *int8xm8\_t vxorvv\_mask\_int8xm8* (*int8xm8\_t merge, int8xm8\_t a, int8xm8\_t b, e8xm8\_t mask*, unsigned int gvl)
- *uint16xm1\_t vxorvv\_mask\_uint16xm1* (*uint16xm1\_t merge, uint16xm1\_t a, uint16xm1\_t b, e16xm1\_t mask*, unsigned int gvl)
- *uint16xm2\_t vxorvv\_mask\_uint16xm2* (*uint16xm2\_t merge, uint16xm2\_t a, uint16xm2\_t b, e16xm2\_t mask*, unsigned int gvl)
- *uint16xm4\_t vxorvv\_mask\_uint16xm4* (*uint16xm4\_t merge, uint16xm4\_t a, uint16xm4\_t b, e16xm4\_t mask*, unsigned int gvl)
- *uint16xm8\_t vxorvv\_mask\_uint16xm8* (*uint16xm8\_t merge, uint16xm8\_t a, uint16xm8\_t b, e16xm8\_t mask*, unsigned int gvl)
- *uint32xm1\_t vxorvv\_mask\_uint32xm1* (*uint32xm1\_t merge, uint32xm1\_t a, uint32xm1\_t b, e32xm1\_t mask*, unsigned int gvl)

- `uint32xm2_t vxorvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vxorvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vxorvv_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vxorvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vxorvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vxorvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vxorvv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vxorvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vxorvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vxorvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vxorvv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = bitwise_xor (a[element], b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.2.19 Elementwise vector-scalar bitwise-xor

**Instruction:** ['vxor.vx']

#### Prototypes:

- `int16xm1_t vxorvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int16xm2_t vxorvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vxorvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vxorvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vxorvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vxorvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vxorvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vxorvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vxorvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`

- `int64xm2_t vxorvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vxorvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vxorvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vxorvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vxorvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vxorvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vxorvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`
- `uint16xm1_t vxorvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vxorvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vxorvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vxorvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vxorvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vxorvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vxorvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vxorvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vxorvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vxorvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vxorvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vxorvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vxorvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vxorvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vxorvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vxorvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = bitwise_xor (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vxorvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vxorvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vxorvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vxorvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vxorvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`

- `int32xm2_t vxorvx_mask_int32xm2` (`int32xm2_t merge`, `int32xm2_t a`, `int b`, `e32xm2_t mask`, unsigned int gvl)
- `int32xm4_t vxorvx_mask_int32xm4` (`int32xm4_t merge`, `int32xm4_t a`, `int b`, `e32xm4_t mask`, unsigned int gvl)
- `int32xm8_t vxorvx_mask_int32xm8` (`int32xm8_t merge`, `int32xm8_t a`, `int b`, `e32xm8_t mask`, unsigned int gvl)
- `int64xm1_t vxorvx_mask_int64xm1` (`int64xm1_t merge`, `int64xm1_t a`, `long b`, `e64xm1_t mask`, unsigned int gvl)
- `int64xm2_t vxorvx_mask_int64xm2` (`int64xm2_t merge`, `int64xm2_t a`, `long b`, `e64xm2_t mask`, unsigned int gvl)
- `int64xm4_t vxorvx_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, `long b`, `e64xm4_t mask`, unsigned int gvl)
- `int64xm8_t vxorvx_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, `long b`, `e64xm8_t mask`, unsigned int gvl)
- `int8xm1_t vxorvx_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, `signed char b`, `e8xm1_t mask`, unsigned int gvl)
- `int8xm2_t vxorvx_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, `signed char b`, `e8xm2_t mask`, unsigned int gvl)
- `int8xm4_t vxorvx_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, `signed char b`, `e8xm4_t mask`, unsigned int gvl)
- `int8xm8_t vxorvx_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, `signed char b`, `e8xm8_t mask`, unsigned int gvl)
- `uint16xm1_t vxorvx_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, `unsigned short b`, `e16xm1_t mask`, unsigned int gvl)
- `uint16xm2_t vxorvx_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, `unsigned short b`, `e16xm2_t mask`, unsigned int gvl)
- `uint16xm4_t vxorvx_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, `unsigned short b`, `e16xm4_t mask`, unsigned int gvl)
- `uint16xm8_t vxorvx_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, `unsigned short b`, `e16xm8_t mask`, unsigned int gvl)
- `uint32xm1_t vxorvx_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, `unsigned int b`, `e32xm1_t mask`, unsigned int gvl)
- `uint32xm2_t vxorvx_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, `unsigned int b`, `e32xm2_t mask`, unsigned int gvl)
- `uint32xm4_t vxorvx_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, `unsigned int b`, `e32xm4_t mask`, unsigned int gvl)
- `uint32xm8_t vxorvx_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, `unsigned int b`, `e32xm8_t mask`, unsigned int gvl)
- `uint64xm1_t vxorvx_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, `unsigned long b`, `e64xm1_t mask`, unsigned int gvl)
- `uint64xm2_t vxorvx_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, `unsigned long b`, `e64xm2_t mask`, unsigned int gvl)
- `uint64xm4_t vxorvx_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, `unsigned long b`, `e64xm4_t mask`, unsigned int gvl)
- `uint64xm8_t vxorvx_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, `unsigned long b`, `e64xm8_t mask`, unsigned int gvl)

- `uint8xm1_t vxorvx_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vxorvx_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vxorvx_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vxorvx_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = bitwise_xor (a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.3 Conversions between floating-point vectors

### 2.3.1 Convert double-width floating-point to current-width

Instruction: ['vfnvtt.f.f.v']

Prototypes:

- `float16xm1_t vfnvttffv_float16xm1_float32xm2` (`float32xm2_t a`, unsigned int `gvl`)
- `float16xm2_t vfnvttffv_float16xm2_float32xm4` (`float32xm4_t a`, unsigned int `gvl`)
- `float16xm4_t vfnvttffv_float16xm4_float32xm8` (`float32xm8_t a`, unsigned int `gvl`)
- `float32xm1_t vfnvttffv_float32xm1_float64xm2` (`float64xm2_t a`, unsigned int `gvl`)
- `float32xm2_t vfnvttffv_float32xm2_float64xm4` (`float64xm4_t a`, unsigned int `gvl`)
- `float32xm4_t vfnvttffv_float32xm4_float64xm8` (`float64xm8_t a`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = wide_fp_to_fp(a[element])
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `float16xm1_t vfnvttffv_mask_float16xm1_float32xm2` (`float16xm1_t merge`, `float32xm2_t a`, `e16xm1_t mask`, unsigned int `gvl`)
- `float16xm2_t vfnvttffv_mask_float16xm2_float32xm4` (`float16xm2_t merge`, `float32xm4_t a`, `e16xm2_t mask`, unsigned int `gvl`)
- `float16xm4_t vfnvttffv_mask_float16xm4_float32xm8` (`float16xm4_t merge`, `float32xm8_t a`, `e16xm4_t mask`, unsigned int `gvl`)
- `float32xm1_t vfnvttffv_mask_float32xm1_float64xm2` (`float32xm1_t merge`, `float64xm2_t a`, `e32xm1_t mask`, unsigned int `gvl`)
- `float32xm2_t vfnvttffv_mask_float32xm2_float64xm4` (`float32xm2_t merge`, `float64xm4_t a`, `e32xm2_t mask`, unsigned int `gvl`)

- *float32xm4\_t vfwcvtffv\_mask\_float32xm4\_float64xm8* (*float32xm4\_t merge, float64xm8\_t a, e32xm4\_t mask*, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = wide_fp_to_fp(a[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.3.2 Convert current-width floating-point to double-width

**Instruction:** ['vfwcvt.f.f.v']

**Prototypes:**

- *float32xm2\_t vfwcvtffv\_float32xm2\_float16xm1* (*float16xm1\_t a*, unsigned int gvl)
- *float32xm4\_t vfwcvtffv\_float32xm4\_float16xm2* (*float16xm2\_t a*, unsigned int gvl)
- *float32xm8\_t vfwcvtffv\_float32xm8\_float16xm4* (*float16xm4\_t a*, unsigned int gvl)
- *float64xm2\_t vfwcvtffv\_float64xm2\_float32xm1* (*float32xm1\_t a*, unsigned int gvl)
- *float64xm4\_t vfwcvtffv\_float64xm4\_float32xm2* (*float32xm2\_t a*, unsigned int gvl)
- *float64xm8\_t vfwcvtffv\_float64xm8\_float32xm4* (*float32xm4\_t a*, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = fp_to_wide_fp(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float32xm2\_t vfwcvtffv\_mask\_float32xm2\_float16xm1* (*float32xm2\_t merge, float16xm1\_t a, e16xm1\_t mask*, unsigned int gvl)
- *float32xm4\_t vfwcvtffv\_mask\_float32xm4\_float16xm2* (*float32xm4\_t merge, float16xm2\_t a, e16xm2\_t mask*, unsigned int gvl)
- *float32xm8\_t vfwcvtffv\_mask\_float32xm8\_float16xm4* (*float32xm8\_t merge, float16xm4\_t a, e16xm4\_t mask*, unsigned int gvl)
- *float64xm2\_t vfwcvtffv\_mask\_float64xm2\_float32xm1* (*float64xm2\_t merge, float32xm1\_t a, e32xm1\_t mask*, unsigned int gvl)
- *float64xm4\_t vfwcvtffv\_mask\_float64xm4\_float32xm2* (*float64xm4\_t merge, float32xm2\_t a, e32xm2\_t mask*, unsigned int gvl)
- *float64xm8\_t vfwcvtffv\_mask\_float64xm8\_float32xm4* (*float64xm8\_t merge, float32xm4\_t a, e32xm4\_t mask*, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = fp_to_wide_fp(a[element])
      else
```

(continues on next page)



(continued from previous page)

```

result[element] = merge[element]
result[gvl : VLMAX] = 0

```

## 2.4 Conversions between integer and floating-point vector

### 2.4.1 Convert integer to floating-point

**Instruction:** ['vfcvt.fx.v']

**Prototypes:**

- *float16xm1\_t* **vfcvtfxv\_float16xm1\_int16xm1** (*int16xm1\_t* a, unsigned int gvl)
- *float16xm2\_t* **vfcvtfxv\_float16xm2\_int16xm2** (*int16xm2\_t* a, unsigned int gvl)
- *float16xm4\_t* **vfcvtfxv\_float16xm4\_int16xm4** (*int16xm4\_t* a, unsigned int gvl)
- *float16xm8\_t* **vfcvtfxv\_float16xm8\_int16xm8** (*int16xm8\_t* a, unsigned int gvl)
- *float32xm1\_t* **vfcvtfxv\_float32xm1\_int32xm1** (*int32xm1\_t* a, unsigned int gvl)
- *float32xm2\_t* **vfcvtfxv\_float32xm2\_int32xm2** (*int32xm2\_t* a, unsigned int gvl)
- *float32xm4\_t* **vfcvtfxv\_float32xm4\_int32xm4** (*int32xm4\_t* a, unsigned int gvl)
- *float32xm8\_t* **vfcvtfxv\_float32xm8\_int32xm8** (*int32xm8\_t* a, unsigned int gvl)
- *float64xm1\_t* **vfcvtfxv\_float64xm1\_int64xm1** (*int64xm1\_t* a, unsigned int gvl)
- *float64xm2\_t* **vfcvtfxv\_float64xm2\_int64xm2** (*int64xm2\_t* a, unsigned int gvl)
- *float64xm4\_t* **vfcvtfxv\_float64xm4\_int64xm4** (*int64xm4\_t* a, unsigned int gvl)
- *float64xm8\_t* **vfcvtfxv\_float64xm8\_int64xm8** (*int64xm8\_t* a, unsigned int gvl)

**Operation:**

```

>>> for element = 0 to gvl - 1
    result[element] = int_to_fp(a[element])
result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- *float16xm1\_t* **vfcvtfxv\_mask\_float16xm1\_int16xm1** (*float16xm1\_t* merge, *int16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- *float16xm2\_t* **vfcvtfxv\_mask\_float16xm2\_int16xm2** (*float16xm2\_t* merge, *int16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- *float16xm4\_t* **vfcvtfxv\_mask\_float16xm4\_int16xm4** (*float16xm4\_t* merge, *int16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- *float16xm8\_t* **vfcvtfxv\_mask\_float16xm8\_int16xm8** (*float16xm8\_t* merge, *int16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- *float32xm1\_t* **vfcvtfxv\_mask\_float32xm1\_int32xm1** (*float32xm1\_t* merge, *int32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- *float32xm2\_t* **vfcvtfxv\_mask\_float32xm2\_int32xm2** (*float32xm2\_t* merge, *int32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- *float32xm4\_t* **vfcvtfxv\_mask\_float32xm4\_int32xm4** (*float32xm4\_t* merge, *int32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)

- `float32xm8_t vfcvtfxv_mask_float32xm8_int32xm8` (`float32xm8_t merge`, `int32xm8_t a`, `e32xm8_t mask`, unsigned int `gvl`)
- `float64xm1_t vfcvtfxv_mask_float64xm1_int64xm1` (`float64xm1_t merge`, `int64xm1_t a`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfcvtfxv_mask_float64xm2_int64xm2` (`float64xm2_t merge`, `int64xm2_t a`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfcvtfxv_mask_float64xm4_int64xm4` (`float64xm4_t merge`, `int64xm4_t a`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vfcvtfxv_mask_float64xm8_int64xm8` (`float64xm8_t merge`, `int64xm8_t a`, `e64xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = int_to_fp(a[element])
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.4.2 Convert unsigned interger to floating-point

Instruction: [`'vfcvt.f.xu.v'`]

Prototypes:

- `float16xm1_t vfcvtfxuv_float16xm1_uint16xm1` (`uint16xm1_t a`, unsigned int `gvl`)
- `float16xm2_t vfcvtfxuv_float16xm2_uint16xm2` (`uint16xm2_t a`, unsigned int `gvl`)
- `float16xm4_t vfcvtfxuv_float16xm4_uint16xm4` (`uint16xm4_t a`, unsigned int `gvl`)
- `float16xm8_t vfcvtfxuv_float16xm8_uint16xm8` (`uint16xm8_t a`, unsigned int `gvl`)
- `float32xm1_t vfcvtfxuv_float32xm1_uint32xm1` (`uint32xm1_t a`, unsigned int `gvl`)
- `float32xm2_t vfcvtfxuv_float32xm2_uint32xm2` (`uint32xm2_t a`, unsigned int `gvl`)
- `float32xm4_t vfcvtfxuv_float32xm4_uint32xm4` (`uint32xm4_t a`, unsigned int `gvl`)
- `float32xm8_t vfcvtfxuv_float32xm8_uint32xm8` (`uint32xm8_t a`, unsigned int `gvl`)
- `float64xm1_t vfcvtfxuv_float64xm1_uint64xm1` (`uint64xm1_t a`, unsigned int `gvl`)
- `float64xm2_t vfcvtfxuv_float64xm2_uint64xm2` (`uint64xm2_t a`, unsigned int `gvl`)
- `float64xm4_t vfcvtfxuv_float64xm4_uint64xm4` (`uint64xm4_t a`, unsigned int `gvl`)
- `float64xm8_t vfcvtfxuv_float64xm8_uint64xm8` (`uint64xm8_t a`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = uint_to_fp(a[element])
      result[gvl : VLMAX] = 0
```

Masked prototypes:

- `float16xm1_t vfcvtfxuv_mask_float16xm1_uint16xm1` (`float16xm1_t merge`, `uint16xm1_t a`, `e16xm1_t mask`, unsigned int `gvl`)

- *float16xm2\_t vfcvtfxuv\_mask\_float16xm2\_uint16xm2* (*float16xm2\_t* merge, *uint16xm2\_t* *a*, *e16xm2\_t* mask, unsigned int *gvl*)
- *float16xm4\_t vfcvtfxuv\_mask\_float16xm4\_uint16xm4* (*float16xm4\_t* merge, *uint16xm4\_t* *a*, *e16xm4\_t* mask, unsigned int *gvl*)
- *float16xm8\_t vfcvtfxuv\_mask\_float16xm8\_uint16xm8* (*float16xm8\_t* merge, *uint16xm8\_t* *a*, *e16xm8\_t* mask, unsigned int *gvl*)
- *float32xm1\_t vfcvtfxuv\_mask\_float32xm1\_uint32xm1* (*float32xm1\_t* merge, *uint32xm1\_t* *a*, *e32xm1\_t* mask, unsigned int *gvl*)
- *float32xm2\_t vfcvtfxuv\_mask\_float32xm2\_uint32xm2* (*float32xm2\_t* merge, *uint32xm2\_t* *a*, *e32xm2\_t* mask, unsigned int *gvl*)
- *float32xm4\_t vfcvtfxuv\_mask\_float32xm4\_uint32xm4* (*float32xm4\_t* merge, *uint32xm4\_t* *a*, *e32xm4\_t* mask, unsigned int *gvl*)
- *float32xm8\_t vfcvtfxuv\_mask\_float32xm8\_uint32xm8* (*float32xm8\_t* merge, *uint32xm8\_t* *a*, *e32xm8\_t* mask, unsigned int *gvl*)
- *float64xm1\_t vfcvtfxuv\_mask\_float64xm1\_uint64xm1* (*float64xm1\_t* merge, *uint64xm1\_t* *a*, *e64xm1\_t* mask, unsigned int *gvl*)
- *float64xm2\_t vfcvtfxuv\_mask\_float64xm2\_uint64xm2* (*float64xm2\_t* merge, *uint64xm2\_t* *a*, *e64xm2\_t* mask, unsigned int *gvl*)
- *float64xm4\_t vfcvtfxuv\_mask\_float64xm4\_uint64xm4* (*float64xm4\_t* merge, *uint64xm4\_t* *a*, *e64xm4\_t* mask, unsigned int *gvl*)
- *float64xm8\_t vfcvtfxuv\_mask\_float64xm8\_uint64xm8* (*float64xm8\_t* merge, *uint64xm8\_t* *a*, *e64xm8\_t* mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = uint_to_fp(a[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.4.3 Convert floating-point to integer

**Instruction:** ['vfcvt.x.f.v']**Prototypes:**

- *int16xm1\_t vfcvtxfv\_int16xm1\_float16xm1* (*float16xm1\_t* *a*, unsigned int *gvl*)
- *int16xm2\_t vfcvtxfv\_int16xm2\_float16xm2* (*float16xm2\_t* *a*, unsigned int *gvl*)
- *int16xm4\_t vfcvtxfv\_int16xm4\_float16xm4* (*float16xm4\_t* *a*, unsigned int *gvl*)
- *int16xm8\_t vfcvtxfv\_int16xm8\_float16xm8* (*float16xm8\_t* *a*, unsigned int *gvl*)
- *int32xm1\_t vfcvtxfv\_int32xm1\_float32xm1* (*float32xm1\_t* *a*, unsigned int *gvl*)
- *int32xm2\_t vfcvtxfv\_int32xm2\_float32xm2* (*float32xm2\_t* *a*, unsigned int *gvl*)
- *int32xm4\_t vfcvtxfv\_int32xm4\_float32xm4* (*float32xm4\_t* *a*, unsigned int *gvl*)
- *int32xm8\_t vfcvtxfv\_int32xm8\_float32xm8* (*float32xm8\_t* *a*, unsigned int *gvl*)
- *int64xm1\_t vfcvtxfv\_int64xm1\_float64xm1* (*float64xm1\_t* *a*, unsigned int *gvl*)

- *int64xm2\_t vfcvtxfv\_int64xm2\_float64xm2* (*float64xm2\_t a*, unsigned int *gvl*)
- *int64xm4\_t vfcvtxfv\_int64xm4\_float64xm4* (*float64xm4\_t a*, unsigned int *gvl*)
- *int64xm8\_t vfcvtxfv\_int64xm8\_float64xm8* (*float64xm8\_t a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = fp_to_int(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vfcvtxfv\_mask\_int16xm1\_float16xm1* (*int16xm1\_t merge*, *float16xm1\_t a*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vfcvtxfv\_mask\_int16xm2\_float16xm2* (*int16xm2\_t merge*, *float16xm2\_t a*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vfcvtxfv\_mask\_int16xm4\_float16xm4* (*int16xm4\_t merge*, *float16xm4\_t a*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vfcvtxfv\_mask\_int16xm8\_float16xm8* (*int16xm8\_t merge*, *float16xm8\_t a*, *e16xm8\_t mask*, unsigned int *gvl*)
- *int32xm1\_t vfcvtxfv\_mask\_int32xm1\_float32xm1* (*int32xm1\_t merge*, *float32xm1\_t a*, *e32xm1\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vfcvtxfv\_mask\_int32xm2\_float32xm2* (*int32xm2\_t merge*, *float32xm2\_t a*, *e32xm2\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vfcvtxfv\_mask\_int32xm4\_float32xm4* (*int32xm4\_t merge*, *float32xm4\_t a*, *e32xm4\_t mask*, unsigned int *gvl*)
- *int32xm8\_t vfcvtxfv\_mask\_int32xm8\_float32xm8* (*int32xm8\_t merge*, *float32xm8\_t a*, *e32xm8\_t mask*, unsigned int *gvl*)
- *int64xm1\_t vfcvtxfv\_mask\_int64xm1\_float64xm1* (*int64xm1\_t merge*, *float64xm1\_t a*, *e64xm1\_t mask*, unsigned int *gvl*)
- *int64xm2\_t vfcvtxfv\_mask\_int64xm2\_float64xm2* (*int64xm2\_t merge*, *float64xm2\_t a*, *e64xm2\_t mask*, unsigned int *gvl*)
- *int64xm4\_t vfcvtxfv\_mask\_int64xm4\_float64xm4* (*int64xm4\_t merge*, *float64xm4\_t a*, *e64xm4\_t mask*, unsigned int *gvl*)
- *int64xm8\_t vfcvtxfv\_mask\_int64xm8\_float64xm8* (*int64xm8\_t merge*, *float64xm8\_t a*, *e64xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = fp_to_int(a[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.4.4 Convert floating-point to unsigned interger

**Instruction:** ['vfcvt.xu.f.v']**Prototypes:**

- `uint16xm1_t vfcvtxufv_uint16xm1_float16xm1 (float16xm1_t a, unsigned int gvl)`
- `uint16xm2_t vfcvtxufv_uint16xm2_float16xm2 (float16xm2_t a, unsigned int gvl)`
- `uint16xm4_t vfcvtxufv_uint16xm4_float16xm4 (float16xm4_t a, unsigned int gvl)`
- `uint16xm8_t vfcvtxufv_uint16xm8_float16xm8 (float16xm8_t a, unsigned int gvl)`
- `uint32xm1_t vfcvtxufv_uint32xm1_float32xm1 (float32xm1_t a, unsigned int gvl)`
- `uint32xm2_t vfcvtxufv_uint32xm2_float32xm2 (float32xm2_t a, unsigned int gvl)`
- `uint32xm4_t vfcvtxufv_uint32xm4_float32xm4 (float32xm4_t a, unsigned int gvl)`
- `uint32xm8_t vfcvtxufv_uint32xm8_float32xm8 (float32xm8_t a, unsigned int gvl)`
- `uint64xm1_t vfcvtxufv_uint64xm1_float64xm1 (float64xm1_t a, unsigned int gvl)`
- `uint64xm2_t vfcvtxufv_uint64xm2_float64xm2 (float64xm2_t a, unsigned int gvl)`
- `uint64xm4_t vfcvtxufv_uint64xm4_float64xm4 (float64xm4_t a, unsigned int gvl)`
- `uint64xm8_t vfcvtxufv_uint64xm8_float64xm8 (float64xm8_t a, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[elemnt] = fp_to_uint(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vfcvtxufv_mask_uint16xm1_float16xm1 (uint16xm1_t merge, float16xm1_t a, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vfcvtxufv_mask_uint16xm2_float16xm2 (uint16xm2_t merge, float16xm2_t a, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vfcvtxufv_mask_uint16xm4_float16xm4 (uint16xm4_t merge, float16xm4_t a, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vfcvtxufv_mask_uint16xm8_float16xm8 (uint16xm8_t merge, float16xm8_t a, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vfcvtxufv_mask_uint32xm1_float32xm1 (uint32xm1_t merge, float32xm1_t a, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vfcvtxufv_mask_uint32xm2_float32xm2 (uint32xm2_t merge, float32xm2_t a, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vfcvtxufv_mask_uint32xm4_float32xm4 (uint32xm4_t merge, float32xm4_t a, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vfcvtxufv_mask_uint32xm8_float32xm8 (uint32xm8_t merge, float32xm8_t a, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vfcvtxufv_mask_uint64xm1_float64xm1 (uint64xm1_t merge, float64xm1_t a, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vfcvtxufv_mask_uint64xm2_float64xm2 (uint64xm2_t merge, float64xm2_t a, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vfcvtxufv_mask_uint64xm4_float64xm4 (uint64xm4_t merge, float64xm4_t a, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vfcvtxufv_mask_uint64xm8_float64xm8 (uint64xm8_t merge, float64xm8_t a, e64xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = fp_to_uint(a[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.4.5 Convert double-width integer to floating-point

**Instruction:** ['vfncvt.f.x.v']

**Prototypes:**

- *float16xm1\_t vfncvtfxv\_float16xm1\_int32xm2* (*int32xm2\_t a*, unsigned int *gvl*)
- *float16xm2\_t vfncvtfxv\_float16xm2\_int32xm4* (*int32xm4\_t a*, unsigned int *gvl*)
- *float16xm4\_t vfncvtfxv\_float16xm4\_int32xm8* (*int32xm8\_t a*, unsigned int *gvl*)
- *float32xm1\_t vfncvtfxv\_float32xm1\_int64xm2* (*int64xm2\_t a*, unsigned int *gvl*)
- *float32xm2\_t vfncvtfxv\_float32xm2\_int64xm4* (*int64xm4\_t a*, unsigned int *gvl*)
- *float32xm4\_t vfncvtfxv\_float32xm4\_int64xm8* (*int64xm8\_t a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = wide_int_to_fp(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfncvtfxv\_mask\_float16xm1\_int32xm2* (*float16xm1\_t merge*, *int32xm2\_t a*, *e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfncvtfxv\_mask\_float16xm2\_int32xm4* (*float16xm2\_t merge*, *int32xm4\_t a*, *e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfncvtfxv\_mask\_float16xm4\_int32xm8* (*float16xm4\_t merge*, *int32xm8\_t a*, *e16xm4\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfncvtfxv\_mask\_float32xm1\_int64xm2* (*float32xm1\_t merge*, *int64xm2\_t a*, *e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfncvtfxv\_mask\_float32xm2\_int64xm4* (*float32xm2\_t merge*, *int64xm4\_t a*, *e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfncvtfxv\_mask\_float32xm4\_int64xm8* (*float32xm4\_t merge*, *int64xm8\_t a*, *e32xm4\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = wide_int_to_fp(a[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```



## 2.4.6 Convert double-width unsigned interger to floating-point

**Instruction:** ['vfnvvt.f.xu.v']

**Prototypes:**

- *float16xm1\_t* **vfnvvtfxuv\_float16xm1\_uint32xm2** (*uint32xm2\_t* *a*, unsigned int *gvl*)
- *float16xm2\_t* **vfnvvtfxuv\_float16xm2\_uint32xm4** (*uint32xm4\_t* *a*, unsigned int *gvl*)
- *float16xm4\_t* **vfnvvtfxuv\_float16xm4\_uint32xm8** (*uint32xm8\_t* *a*, unsigned int *gvl*)
- *float32xm1\_t* **vfnvvtfxuv\_float32xm1\_uint64xm2** (*uint64xm2\_t* *a*, unsigned int *gvl*)
- *float32xm2\_t* **vfnvvtfxuv\_float32xm2\_uint64xm4** (*uint64xm4\_t* *a*, unsigned int *gvl*)
- *float32xm4\_t* **vfnvvtfxuv\_float32xm4\_uint64xm8** (*uint64xm8\_t* *a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = wide_uint_to_fp(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t* **vfnvvtfxuv\_mask\_float16xm1\_uint32xm2** (*float16xm1\_t* *merge*, *uint32xm2\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *float16xm2\_t* **vfnvvtfxuv\_mask\_float16xm2\_uint32xm4** (*float16xm2\_t* *merge*, *uint32xm4\_t* *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *float16xm4\_t* **vfnvvtfxuv\_mask\_float16xm4\_uint32xm8** (*float16xm4\_t* *merge*, *uint32xm8\_t* *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *float32xm1\_t* **vfnvvtfxuv\_mask\_float32xm1\_uint64xm2** (*float32xm1\_t* *merge*, *uint64xm2\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *float32xm2\_t* **vfnvvtfxuv\_mask\_float32xm2\_uint64xm4** (*float32xm2\_t* *merge*, *uint64xm4\_t* *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *float32xm4\_t* **vfnvvtfxuv\_mask\_float32xm4\_uint64xm8** (*float32xm4\_t* *merge*, *uint64xm8\_t* *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = wide_uint_to_fp(a[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.4.7 Convert double-width floating-point to interger

**Instruction:** ['vfnvvt.x.f.v']

**Prototypes:**

- *int16xm1\_t* **vfnvvtxfv\_int16xm1\_float32xm2** (*float32xm2\_t* *a*, unsigned int *gvl*)
- *int16xm2\_t* **vfnvvtxfv\_int16xm2\_float32xm4** (*float32xm4\_t* *a*, unsigned int *gvl*)
- *int16xm4\_t* **vfnvvtxfv\_int16xm4\_float32xm8** (*float32xm8\_t* *a*, unsigned int *gvl*)

- *int32xm1\_t vfncvtxfv\_int32xm1\_float64xm2* (*float64xm2\_t a*, unsigned int *gvl*)
- *int32xm2\_t vfncvtxfv\_int32xm2\_float64xm4* (*float64xm4\_t a*, unsigned int *gvl*)
- *int32xm4\_t vfncvtxfv\_int32xm4\_float64xm8* (*float64xm8\_t a*, unsigned int *gvl*)
- *int8xm1\_t vfncvtxfv\_int8xm1\_float16xm2* (*float16xm2\_t a*, unsigned int *gvl*)
- *int8xm2\_t vfncvtxfv\_int8xm2\_float16xm4* (*float16xm4\_t a*, unsigned int *gvl*)
- *int8xm4\_t vfncvtxfv\_int8xm4\_float16xm8* (*float16xm8\_t a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = wide_fp_to_int(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vfncvtxfv\_mask\_int16xm1\_float32xm2* (*int16xm1\_t merge*, *float32xm2\_t a*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vfncvtxfv\_mask\_int16xm2\_float32xm4* (*int16xm2\_t merge*, *float32xm4\_t a*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vfncvtxfv\_mask\_int16xm4\_float32xm8* (*int16xm4\_t merge*, *float32xm8\_t a*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int32xm1\_t vfncvtxfv\_mask\_int32xm1\_float64xm2* (*int32xm1\_t merge*, *float64xm2\_t a*, *e32xm1\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vfncvtxfv\_mask\_int32xm2\_float64xm4* (*int32xm2\_t merge*, *float64xm4\_t a*, *e32xm2\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vfncvtxfv\_mask\_int32xm4\_float64xm8* (*int32xm4\_t merge*, *float64xm8\_t a*, *e32xm4\_t mask*, unsigned int *gvl*)
- *int8xm1\_t vfncvtxfv\_mask\_int8xm1\_float16xm2* (*int8xm1\_t merge*, *float16xm2\_t a*, *e8xm1\_t mask*, unsigned int *gvl*)
- *int8xm2\_t vfncvtxfv\_mask\_int8xm2\_float16xm4* (*int8xm2\_t merge*, *float16xm4\_t a*, *e8xm2\_t mask*, unsigned int *gvl*)
- *int8xm4\_t vfncvtxfv\_mask\_int8xm4\_float16xm8* (*int8xm4\_t merge*, *float16xm8\_t a*, *e8xm4\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = wide_fp_to_int(a[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.4.8 Convert double-width floating-point to unsigned integer

**Instruction:** ['vfncvt.xu.f.v']**Prototypes:**

- *uint16xm1\_t vfncvtxufv\_uint16xm1\_float32xm2* (*float32xm2\_t a*, unsigned int *gvl*)
- *uint16xm2\_t vfncvtxufv\_uint16xm2\_float32xm4* (*float32xm4\_t a*, unsigned int *gvl*)

- *uint16xm4\_t* **vfncvtxufv\_uint16xm4\_float32xm8** (*float32xm8\_t* *a*, unsigned int *gvl*)
- *uint32xm1\_t* **vfncvtxufv\_uint32xm1\_float64xm2** (*float64xm2\_t* *a*, unsigned int *gvl*)
- *uint32xm2\_t* **vfncvtxufv\_uint32xm2\_float64xm4** (*float64xm4\_t* *a*, unsigned int *gvl*)
- *uint32xm4\_t* **vfncvtxufv\_uint32xm4\_float64xm8** (*float64xm8\_t* *a*, unsigned int *gvl*)
- *uint8xm1\_t* **vfncvtxufv\_uint8xm1\_float16xm2** (*float16xm2\_t* *a*, unsigned int *gvl*)
- *uint8xm2\_t* **vfncvtxufv\_uint8xm2\_float16xm4** (*float16xm4\_t* *a*, unsigned int *gvl*)
- *uint8xm4\_t* **vfncvtxufv\_uint8xm4\_float16xm8** (*float16xm8\_t* *a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = wide_fp_to_uint(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *uint16xm1\_t* **vfncvtxufv\_mask\_uint16xm1\_float32xm2** (*uint16xm1\_t* *merge*, *float32xm2\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *uint16xm2\_t* **vfncvtxufv\_mask\_uint16xm2\_float32xm4** (*uint16xm2\_t* *merge*, *float32xm4\_t* *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *uint16xm4\_t* **vfncvtxufv\_mask\_uint16xm4\_float32xm8** (*uint16xm4\_t* *merge*, *float32xm8\_t* *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *uint32xm1\_t* **vfncvtxufv\_mask\_uint32xm1\_float64xm2** (*uint32xm1\_t* *merge*, *float64xm2\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *uint32xm2\_t* **vfncvtxufv\_mask\_uint32xm2\_float64xm4** (*uint32xm2\_t* *merge*, *float64xm4\_t* *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *uint32xm4\_t* **vfncvtxufv\_mask\_uint32xm4\_float64xm8** (*uint32xm4\_t* *merge*, *float64xm8\_t* *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *uint8xm1\_t* **vfncvtxufv\_mask\_uint8xm1\_float16xm2** (*uint8xm1\_t* *merge*, *float16xm2\_t* *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *uint8xm2\_t* **vfncvtxufv\_mask\_uint8xm2\_float16xm4** (*uint8xm2\_t* *merge*, *float16xm4\_t* *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- *uint8xm4\_t* **vfncvtxufv\_mask\_uint8xm4\_float16xm8** (*uint8xm4\_t* *merge*, *float16xm8\_t* *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = wide_fp_to_uint(a[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.4.9 Convert interger to double-width floating-point

**Instruction:** [*'vfwcvt.f.x.v'*]**Prototypes:**

- *float16xm2\_t* **vfwcvtfxv\_float16xm2\_int8xm1** (*int8xm1\_t* *a*, unsigned int *gvl*)

- `float16xm4_t vfwcvtfxv_float16xm4_int8xm2 (int8xm2_t a, unsigned int gvl)`
- `float16xm8_t vfwcvtfxv_float16xm8_int8xm4 (int8xm4_t a, unsigned int gvl)`
- `float32xm2_t vfwcvtfxv_float32xm2_int16xm1 (int16xm1_t a, unsigned int gvl)`
- `float32xm4_t vfwcvtfxv_float32xm4_int16xm2 (int16xm2_t a, unsigned int gvl)`
- `float32xm8_t vfwcvtfxv_float32xm8_int16xm4 (int16xm4_t a, unsigned int gvl)`
- `float64xm2_t vfwcvtfxv_float64xm2_int32xm1 (int32xm1_t a, unsigned int gvl)`
- `float64xm4_t vfwcvtfxv_float64xm4_int32xm2 (int32xm2_t a, unsigned int gvl)`
- `float64xm8_t vfwcvtfxv_float64xm8_int32xm4 (int32xm4_t a, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = int_to_wide_fp(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm2_t vfwcvtfxv_mask_float16xm2_int8xm1 (float16xm2_t merge, int8xm1_t a, e8xm1_t mask, unsigned int gvl)`
- `float16xm4_t vfwcvtfxv_mask_float16xm4_int8xm2 (float16xm4_t merge, int8xm2_t a, e8xm2_t mask, unsigned int gvl)`
- `float16xm8_t vfwcvtfxv_mask_float16xm8_int8xm4 (float16xm8_t merge, int8xm4_t a, e8xm4_t mask, unsigned int gvl)`
- `float32xm2_t vfwcvtfxv_mask_float32xm2_int16xm1 (float32xm2_t merge, int16xm1_t a, e16xm1_t mask, unsigned int gvl)`
- `float32xm4_t vfwcvtfxv_mask_float32xm4_int16xm2 (float32xm4_t merge, int16xm2_t a, e16xm2_t mask, unsigned int gvl)`
- `float32xm8_t vfwcvtfxv_mask_float32xm8_int16xm4 (float32xm8_t merge, int16xm4_t a, e16xm4_t mask, unsigned int gvl)`
- `float64xm2_t vfwcvtfxv_mask_float64xm2_int32xm1 (float64xm2_t merge, int32xm1_t a, e32xm1_t mask, unsigned int gvl)`
- `float64xm4_t vfwcvtfxv_mask_float64xm4_int32xm2 (float64xm4_t merge, int32xm2_t a, e32xm2_t mask, unsigned int gvl)`
- `float64xm8_t vfwcvtfxv_mask_float64xm8_int32xm4 (float64xm8_t merge, int32xm4_t a, e32xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = int_to_wide_fp(a[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.4.10 Convert unsigned integer to double-width floating-point

**Instruction:** [`'vfwcvt.f.xu.v'`]**Prototypes:**

- *float16xm2\_t* **vfwcvtfxuv\_float16xm2\_uint8xm1** (*uint8xm1\_t* *a*, unsigned int *gvl*)
- *float16xm4\_t* **vfwcvtfxuv\_float16xm4\_uint8xm2** (*uint8xm2\_t* *a*, unsigned int *gvl*)
- *float16xm8\_t* **vfwcvtfxuv\_float16xm8\_uint8xm4** (*uint8xm4\_t* *a*, unsigned int *gvl*)
- *float32xm2\_t* **vfwcvtfxuv\_float32xm2\_uint16xm1** (*uint16xm1\_t* *a*, unsigned int *gvl*)
- *float32xm4\_t* **vfwcvtfxuv\_float32xm4\_uint16xm2** (*uint16xm2\_t* *a*, unsigned int *gvl*)
- *float32xm8\_t* **vfwcvtfxuv\_float32xm8\_uint16xm4** (*uint16xm4\_t* *a*, unsigned int *gvl*)
- *float64xm2\_t* **vfwcvtfxuv\_float64xm2\_uint32xm1** (*uint32xm1\_t* *a*, unsigned int *gvl*)
- *float64xm4\_t* **vfwcvtfxuv\_float64xm4\_uint32xm2** (*uint32xm2\_t* *a*, unsigned int *gvl*)
- *float64xm8\_t* **vfwcvtfxuv\_float64xm8\_uint32xm4** (*uint32xm4\_t* *a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = uint_to_wide_fp(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm2\_t* **vfwcvtfxuv\_mask\_float16xm2\_uint8xm1** (*float16xm2\_t* *merge*, *uint8xm1\_t* *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *float16xm4\_t* **vfwcvtfxuv\_mask\_float16xm4\_uint8xm2** (*float16xm4\_t* *merge*, *uint8xm2\_t* *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- *float16xm8\_t* **vfwcvtfxuv\_mask\_float16xm8\_uint8xm4** (*float16xm8\_t* *merge*, *uint8xm4\_t* *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- *float32xm2\_t* **vfwcvtfxuv\_mask\_float32xm2\_uint16xm1** (*float32xm2\_t* *merge*, *uint16xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *float32xm4\_t* **vfwcvtfxuv\_mask\_float32xm4\_uint16xm2** (*float32xm4\_t* *merge*, *uint16xm2\_t* *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *float32xm8\_t* **vfwcvtfxuv\_mask\_float32xm8\_uint16xm4** (*float32xm8\_t* *merge*, *uint16xm4\_t* *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *float64xm2\_t* **vfwcvtfxuv\_mask\_float64xm2\_uint32xm1** (*float64xm2\_t* *merge*, *uint32xm1\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *float64xm4\_t* **vfwcvtfxuv\_mask\_float64xm4\_uint32xm2** (*float64xm4\_t* *merge*, *uint32xm2\_t* *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *float64xm8\_t* **vfwcvtfxuv\_mask\_float64xm8\_uint32xm4** (*float64xm8\_t* *merge*, *uint32xm4\_t* *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = uint_to_wide_fp(a[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.4.11 Convert floating-point to double-width integer

**Instruction:** ['vfwcvt.x.f.v']

**Prototypes:**

- *int32xm2\_t vfwcvtxfv\_int32xm2\_float16xm1* (*float16xm1\_t a*, unsigned int *gvl*)
- *int32xm4\_t vfwcvtxfv\_int32xm4\_float16xm2* (*float16xm2\_t a*, unsigned int *gvl*)
- *int32xm8\_t vfwcvtxfv\_int32xm8\_float16xm4* (*float16xm4\_t a*, unsigned int *gvl*)
- *int64xm2\_t vfwcvtxfv\_int64xm2\_float32xm1* (*float32xm1\_t a*, unsigned int *gvl*)
- *int64xm4\_t vfwcvtxfv\_int64xm4\_float32xm2* (*float32xm2\_t a*, unsigned int *gvl*)
- *int64xm8\_t vfwcvtxfv\_int64xm8\_float32xm4* (*float32xm4\_t a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = fp_to_wide_int(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int32xm2\_t vfwcvtxfv\_mask\_int32xm2\_float16xm1* (*int32xm2\_t merge*, *float16xm1\_t a*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vfwcvtxfv\_mask\_int32xm4\_float16xm2* (*int32xm4\_t merge*, *float16xm2\_t a*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int32xm8\_t vfwcvtxfv\_mask\_int32xm8\_float16xm4* (*int32xm8\_t merge*, *float16xm4\_t a*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int64xm2\_t vfwcvtxfv\_mask\_int64xm2\_float32xm1* (*int64xm2\_t merge*, *float32xm1\_t a*, *e32xm1\_t mask*, unsigned int *gvl*)
- *int64xm4\_t vfwcvtxfv\_mask\_int64xm4\_float32xm2* (*int64xm4\_t merge*, *float32xm2\_t a*, *e32xm2\_t mask*, unsigned int *gvl*)
- *int64xm8\_t vfwcvtxfv\_mask\_int64xm8\_float32xm4* (*int64xm8\_t merge*, *float32xm4\_t a*, *e32xm4\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = fp_to_wide_int(a[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.4.12 Convert floating-point to double-width unsigned integer

**Instruction:** ['vfwcvt.xu.f.v']

**Prototypes:**

- *uint32xm2\_t vfwcvtxufv\_uint32xm2\_float16xm1* (*float16xm1\_t a*, unsigned int *gvl*)
- *uint32xm4\_t vfwcvtxufv\_uint32xm4\_float16xm2* (*float16xm2\_t a*, unsigned int *gvl*)
- *uint32xm8\_t vfwcvtxufv\_uint32xm8\_float16xm4* (*float16xm4\_t a*, unsigned int *gvl*)



- *uint64xm2\_t* **vwfvcvtxufv\_uint64xm2\_float32xm1** (*float32xm1\_t* *a*, unsigned int *gvl*)
- *uint64xm4\_t* **vwfvcvtxufv\_uint64xm4\_float32xm2** (*float32xm2\_t* *a*, unsigned int *gvl*)
- *uint64xm8\_t* **vwfvcvtxufv\_uint64xm8\_float32xm4** (*float32xm4\_t* *a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = fp_to_wide_uint(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *uint32xm2\_t* **vwfvcvtxufv\_mask\_uint32xm2\_float16xm1** (*uint32xm2\_t* *merge*, *float16xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *uint32xm4\_t* **vwfvcvtxufv\_mask\_uint32xm4\_float16xm2** (*uint32xm4\_t* *merge*, *float16xm2\_t* *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *uint32xm8\_t* **vwfvcvtxufv\_mask\_uint32xm8\_float16xm4** (*uint32xm8\_t* *merge*, *float16xm4\_t* *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *uint64xm2\_t* **vwfvcvtxufv\_mask\_uint64xm2\_float32xm1** (*uint64xm2\_t* *merge*, *float32xm1\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *uint64xm4\_t* **vwfvcvtxufv\_mask\_uint64xm4\_float32xm2** (*uint64xm4\_t* *merge*, *float32xm2\_t* *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *uint64xm8\_t* **vwfvcvtxufv\_mask\_uint64xm8\_float32xm4** (*uint64xm8\_t* *merge*, *float32xm4\_t* *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = fp_to_wide_uint(a[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5 Floating-point arithmetic operations

### 2.5.1 Elementwise vector-scalar floating-point addition

**Instruction:** [*'vfadd.vf'*]**Prototypes:**

- *float16xm1\_t* **vfaddvf\_float16xm1** (*float16xm1\_t* *a*, *float16\_t* *b*, unsigned int *gvl*)
- *float16xm2\_t* **vfaddvf\_float16xm2** (*float16xm2\_t* *a*, *float16\_t* *b*, unsigned int *gvl*)
- *float16xm4\_t* **vfaddvf\_float16xm4** (*float16xm4\_t* *a*, *float16\_t* *b*, unsigned int *gvl*)
- *float16xm8\_t* **vfaddvf\_float16xm8** (*float16xm8\_t* *a*, *float16\_t* *b*, unsigned int *gvl*)
- *float32xm1\_t* **vfaddvf\_float32xm1** (*float32xm1\_t* *a*, *float* *b*, unsigned int *gvl*)
- *float32xm2\_t* **vfaddvf\_float32xm2** (*float32xm2\_t* *a*, *float* *b*, unsigned int *gvl*)
- *float32xm4\_t* **vfaddvf\_float32xm4** (*float32xm4\_t* *a*, *float* *b*, unsigned int *gvl*)

- *float32xm8\_t* **vfaddvf\_float32xm8** (*float32xm8\_t* *a*, float *b*, unsigned int *gvl*)
- *float64xm1\_t* **vfaddvf\_float64xm1** (*float64xm1\_t* *a*, double *b*, unsigned int *gvl*)
- *float64xm2\_t* **vfaddvf\_float64xm2** (*float64xm2\_t* *a*, double *b*, unsigned int *gvl*)
- *float64xm4\_t* **vfaddvf\_float64xm4** (*float64xm4\_t* *a*, double *b*, unsigned int *gvl*)
- *float64xm8\_t* **vfaddvf\_float64xm8** (*float64xm8\_t* *a*, double *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t* **vfaddvf\_mask\_float16xm1** (*float16xm1\_t* *merge*, *float16xm1\_t* *a*, float16\_t *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *float16xm2\_t* **vfaddvf\_mask\_float16xm2** (*float16xm2\_t* *merge*, *float16xm2\_t* *a*, float16\_t *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *float16xm4\_t* **vfaddvf\_mask\_float16xm4** (*float16xm4\_t* *merge*, *float16xm4\_t* *a*, float16\_t *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *float16xm8\_t* **vfaddvf\_mask\_float16xm8** (*float16xm8\_t* *merge*, *float16xm8\_t* *a*, float16\_t *b*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- *float32xm1\_t* **vfaddvf\_mask\_float32xm1** (*float32xm1\_t* *merge*, *float32xm1\_t* *a*, float *b*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *float32xm2\_t* **vfaddvf\_mask\_float32xm2** (*float32xm2\_t* *merge*, *float32xm2\_t* *a*, float *b*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *float32xm4\_t* **vfaddvf\_mask\_float32xm4** (*float32xm4\_t* *merge*, *float32xm4\_t* *a*, float *b*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *float32xm8\_t* **vfaddvf\_mask\_float32xm8** (*float32xm8\_t* *merge*, *float32xm8\_t* *a*, float *b*, *e32xm8\_t* *mask*, unsigned int *gvl*)
- *float64xm1\_t* **vfaddvf\_mask\_float64xm1** (*float64xm1\_t* *merge*, *float64xm1\_t* *a*, double *b*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *float64xm2\_t* **vfaddvf\_mask\_float64xm2** (*float64xm2\_t* *merge*, *float64xm2\_t* *a*, double *b*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *float64xm4\_t* **vfaddvf\_mask\_float64xm4** (*float64xm4\_t* *merge*, *float64xm4\_t* *a*, double *b*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *float64xm8\_t* **vfaddvf\_mask\_float64xm8** (*float64xm8\_t* *merge*, *float64xm8\_t* *a*, double *b*, *e64xm8\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = a[element] + b
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.2 Elementwise vector-vector floating-point addition

**Instruction:** [`'vfadd.vv'`]

**Prototypes:**

- `float16xm1_t vfaddvv_float16xm1 (float16xm1_t a, float16xm1_t b, unsigned int gvl)`
- `float16xm2_t vfaddvv_float16xm2 (float16xm2_t a, float16xm2_t b, unsigned int gvl)`
- `float16xm4_t vfaddvv_float16xm4 (float16xm4_t a, float16xm4_t b, unsigned int gvl)`
- `float16xm8_t vfaddvv_float16xm8 (float16xm8_t a, float16xm8_t b, unsigned int gvl)`
- `float32xm1_t vfaddvv_float32xm1 (float32xm1_t a, float32xm1_t b, unsigned int gvl)`
- `float32xm2_t vfaddvv_float32xm2 (float32xm2_t a, float32xm2_t b, unsigned int gvl)`
- `float32xm4_t vfaddvv_float32xm4 (float32xm4_t a, float32xm4_t b, unsigned int gvl)`
- `float32xm8_t vfaddvv_float32xm8 (float32xm8_t a, float32xm8_t b, unsigned int gvl)`
- `float64xm1_t vfaddvv_float64xm1 (float64xm1_t a, float64xm1_t b, unsigned int gvl)`
- `float64xm2_t vfaddvv_float64xm2 (float64xm2_t a, float64xm2_t b, unsigned int gvl)`
- `float64xm4_t vfaddvv_float64xm4 (float64xm4_t a, float64xm4_t b, unsigned int gvl)`
- `float64xm8_t vfaddvv_float64xm8 (float64xm8_t a, float64xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vfaddvv_mask_float16xm1 (float16xm1_t merge, float16xm1_t a, float16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `float16xm2_t vfaddvv_mask_float16xm2 (float16xm2_t merge, float16xm2_t a, float16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `float16xm4_t vfaddvv_mask_float16xm4 (float16xm4_t merge, float16xm4_t a, float16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `float16xm8_t vfaddvv_mask_float16xm8 (float16xm8_t merge, float16xm8_t a, float16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `float32xm1_t vfaddvv_mask_float32xm1 (float32xm1_t merge, float32xm1_t a, float32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `float32xm2_t vfaddvv_mask_float32xm2 (float32xm2_t merge, float32xm2_t a, float32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `float32xm4_t vfaddvv_mask_float32xm4 (float32xm4_t merge, float32xm4_t a, float32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `float32xm8_t vfaddvv_mask_float32xm8 (float32xm8_t merge, float32xm8_t a, float32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `float64xm1_t vfaddvv_mask_float64xm1 (float64xm1_t merge, float64xm1_t a, float64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `float64xm2_t vfaddvv_mask_float64xm2 (float64xm2_t merge, float64xm2_t a, float64xm2_t b, e64xm2_t mask, unsigned int gvl)`

- `float64xm4_t vfaddvv_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `float64xm4_t b`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vfaddvv_mask_float64xm8` (`float64xm8_t merge`, `float64xm8_t a`, `float64xm8_t b`, `e64xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = a[element] + b[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

### 2.5.3 Classify every floating-point element as the saclar classify instruction do

**Instruction:** [`'vfclass.v'`]

**Prototypes:**

- `uint16xm1_t vfclassv_uint16xm1_float16xm1` (`float16xm1_t a`, unsigned int `gvl`)
- `uint16xm2_t vfclassv_uint16xm2_float16xm2` (`float16xm2_t a`, unsigned int `gvl`)
- `uint16xm4_t vfclassv_uint16xm4_float16xm4` (`float16xm4_t a`, unsigned int `gvl`)
- `uint16xm8_t vfclassv_uint16xm8_float16xm8` (`float16xm8_t a`, unsigned int `gvl`)
- `uint32xm1_t vfclassv_uint32xm1_float32xm1` (`float32xm1_t a`, unsigned int `gvl`)
- `uint32xm2_t vfclassv_uint32xm2_float32xm2` (`float32xm2_t a`, unsigned int `gvl`)
- `uint32xm4_t vfclassv_uint32xm4_float32xm4` (`float32xm4_t a`, unsigned int `gvl`)
- `uint32xm8_t vfclassv_uint32xm8_float32xm8` (`float32xm8_t a`, unsigned int `gvl`)
- `uint64xm1_t vfclassv_uint64xm1_float64xm1` (`float64xm1_t a`, unsigned int `gvl`)
- `uint64xm2_t vfclassv_uint64xm2_float64xm2` (`float64xm2_t a`, unsigned int `gvl`)
- `uint64xm4_t vfclassv_uint64xm4_float64xm4` (`float64xm4_t a`, unsigned int `gvl`)
- `uint64xm8_t vfclassv_uint64xm8_float64xm8` (`float64xm8_t a`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = fclassify(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vfclassv_mask_uint16xm1_float16xm1` (`uint16xm1_t merge`, `float16xm1_t a`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint16xm2_t vfclassv_mask_uint16xm2_float16xm2` (`uint16xm2_t merge`, `float16xm2_t a`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint16xm4_t vfclassv_mask_uint16xm4_float16xm4` (`uint16xm4_t merge`, `float16xm4_t a`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vfclassv_mask_uint16xm8_float16xm8` (`uint16xm8_t merge`, `float16xm8_t a`, `e16xm8_t mask`, unsigned int `gvl`)

- `uint32xm1_t vfclassv_mask_uint32xm1_float32xm1 (uint32xm1_t merge, float32xm1_t a, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vfclassv_mask_uint32xm2_float32xm2 (uint32xm2_t merge, float32xm2_t a, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vfclassv_mask_uint32xm4_float32xm4 (uint32xm4_t merge, float32xm4_t a, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vfclassv_mask_uint32xm8_float32xm8 (uint32xm8_t merge, float32xm8_t a, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vfclassv_mask_uint64xm1_float64xm1 (uint64xm1_t merge, float64xm1_t a, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vfclassv_mask_uint64xm2_float64xm2 (uint64xm2_t merge, float64xm2_t a, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vfclassv_mask_uint64xm4_float64xm4 (uint64xm4_t merge, float64xm4_t a, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vfclassv_mask_uint64xm8_float64xm8 (uint64xm8_t merge, float64xm8_t a, e64xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = fclassify(a[element])
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.5.4 Elementwise vector-scalar floating-point division

**Instruction:** [`'vfdiv.vf'`]**Prototypes:**

- `float16xm1_t vfdivvf_float16xm1 (float16xm1_t a, float16_t b, unsigned int gvl)`
- `float16xm2_t vfdivvf_float16xm2 (float16xm2_t a, float16_t b, unsigned int gvl)`
- `float16xm4_t vfdivvf_float16xm4 (float16xm4_t a, float16_t b, unsigned int gvl)`
- `float16xm8_t vfdivvf_float16xm8 (float16xm8_t a, float16_t b, unsigned int gvl)`
- `float32xm1_t vfdivvf_float32xm1 (float32xm1_t a, float b, unsigned int gvl)`
- `float32xm2_t vfdivvf_float32xm2 (float32xm2_t a, float b, unsigned int gvl)`
- `float32xm4_t vfdivvf_float32xm4 (float32xm4_t a, float b, unsigned int gvl)`
- `float32xm8_t vfdivvf_float32xm8 (float32xm8_t a, float b, unsigned int gvl)`
- `float64xm1_t vfdivvf_float64xm1 (float64xm1_t a, double b, unsigned int gvl)`
- `float64xm2_t vfdivvf_float64xm2 (float64xm2_t a, double b, unsigned int gvl)`
- `float64xm4_t vfdivvf_float64xm4 (float64xm4_t a, double b, unsigned int gvl)`
- `float64xm8_t vfdivvf_float64xm8 (float64xm8_t a, double b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] / b
      result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- *float16xm1\_t vfdiuvf\_mask\_float16xm1* (*float16xm1\_t merge, float16xm1\_t a, float16\_t b, e16xm1\_t mask, unsigned int gvl*)
- *float16xm2\_t vfdiuvf\_mask\_float16xm2* (*float16xm2\_t merge, float16xm2\_t a, float16\_t b, e16xm2\_t mask, unsigned int gvl*)
- *float16xm4\_t vfdiuvf\_mask\_float16xm4* (*float16xm4\_t merge, float16xm4\_t a, float16\_t b, e16xm4\_t mask, unsigned int gvl*)
- *float16xm8\_t vfdiuvf\_mask\_float16xm8* (*float16xm8\_t merge, float16xm8\_t a, float16\_t b, e16xm8\_t mask, unsigned int gvl*)
- *float32xm1\_t vfdiuvf\_mask\_float32xm1* (*float32xm1\_t merge, float32xm1\_t a, float b, e32xm1\_t mask, unsigned int gvl*)
- *float32xm2\_t vfdiuvf\_mask\_float32xm2* (*float32xm2\_t merge, float32xm2\_t a, float b, e32xm2\_t mask, unsigned int gvl*)
- *float32xm4\_t vfdiuvf\_mask\_float32xm4* (*float32xm4\_t merge, float32xm4\_t a, float b, e32xm4\_t mask, unsigned int gvl*)
- *float32xm8\_t vfdiuvf\_mask\_float32xm8* (*float32xm8\_t merge, float32xm8\_t a, float b, e32xm8\_t mask, unsigned int gvl*)
- *float64xm1\_t vfdiuvf\_mask\_float64xm1* (*float64xm1\_t merge, float64xm1\_t a, double b, e64xm1\_t mask, unsigned int gvl*)
- *float64xm2\_t vfdiuvf\_mask\_float64xm2* (*float64xm2\_t merge, float64xm2\_t a, double b, e64xm2\_t mask, unsigned int gvl*)
- *float64xm4\_t vfdiuvf\_mask\_float64xm4* (*float64xm4\_t merge, float64xm4\_t a, double b, e64xm4\_t mask, unsigned int gvl*)
- *float64xm8\_t vfdiuvf\_mask\_float64xm8* (*float64xm8\_t merge, float64xm8\_t a, double b, e64xm8\_t mask, unsigned int gvl*)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = a[element] / b
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.5.5 Elementwise vector-vector floating-point division

**Instruction:** [*'vfdiuvv'*]

#### Prototypes:

- *float16xm1\_t vfdiuvv\_float16xm1* (*float16xm1\_t a, float16xm1\_t b, unsigned int gvl*)
- *float16xm2\_t vfdiuvv\_float16xm2* (*float16xm2\_t a, float16xm2\_t b, unsigned int gvl*)
- *float16xm4\_t vfdiuvv\_float16xm4* (*float16xm4\_t a, float16xm4\_t b, unsigned int gvl*)
- *float16xm8\_t vfdiuvv\_float16xm8* (*float16xm8\_t a, float16xm8\_t b, unsigned int gvl*)



- *float32xm1\_t vfdiuvv\_float32xm1* (*float32xm1\_t a*, *float32xm1\_t b*, unsigned int *gvl*)
- *float32xm2\_t vfdiuvv\_float32xm2* (*float32xm2\_t a*, *float32xm2\_t b*, unsigned int *gvl*)
- *float32xm4\_t vfdiuvv\_float32xm4* (*float32xm4\_t a*, *float32xm4\_t b*, unsigned int *gvl*)
- *float32xm8\_t vfdiuvv\_float32xm8* (*float32xm8\_t a*, *float32xm8\_t b*, unsigned int *gvl*)
- *float64xm1\_t vfdiuvv\_float64xm1* (*float64xm1\_t a*, *float64xm1\_t b*, unsigned int *gvl*)
- *float64xm2\_t vfdiuvv\_float64xm2* (*float64xm2\_t a*, *float64xm2\_t b*, unsigned int *gvl*)
- *float64xm4\_t vfdiuvv\_float64xm4* (*float64xm4\_t a*, *float64xm4\_t b*, unsigned int *gvl*)
- *float64xm8\_t vfdiuvv\_float64xm8* (*float64xm8\_t a*, *float64xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] / b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfdiuvv\_mask\_float16xm1* (*float16xm1\_t merge*, *float16xm1\_t a*, *float16xm1\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfdiuvv\_mask\_float16xm2* (*float16xm2\_t merge*, *float16xm2\_t a*, *float16xm2\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfdiuvv\_mask\_float16xm4* (*float16xm4\_t merge*, *float16xm4\_t a*, *float16xm4\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *float16xm8\_t vfdiuvv\_mask\_float16xm8* (*float16xm8\_t merge*, *float16xm8\_t a*, *float16xm8\_t b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfdiuvv\_mask\_float32xm1* (*float32xm1\_t merge*, *float32xm1\_t a*, *float32xm1\_t b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfdiuvv\_mask\_float32xm2* (*float32xm2\_t merge*, *float32xm2\_t a*, *float32xm2\_t b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfdiuvv\_mask\_float32xm4* (*float32xm4\_t merge*, *float32xm4\_t a*, *float32xm4\_t b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *float32xm8\_t vfdiuvv\_mask\_float32xm8* (*float32xm8\_t merge*, *float32xm8\_t a*, *float32xm8\_t b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *float64xm1\_t vfdiuvv\_mask\_float64xm1* (*float64xm1\_t merge*, *float64xm1\_t a*, *float64xm1\_t b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vfdiuvv\_mask\_float64xm2* (*float64xm2\_t merge*, *float64xm2\_t a*, *float64xm2\_t b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *float64xm4\_t vfdiuvv\_mask\_float64xm4* (*float64xm4\_t merge*, *float64xm4\_t a*, *float64xm4\_t b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *float64xm8\_t vfdiuvv\_mask\_float64xm8* (*float64xm8\_t merge*, *float64xm8\_t a*, *float64xm8\_t b*, *e64xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = a[element] / b[element]
      else
```

(continues on next page)

(continued from previous page)

```

    result[element] = merge[element]
    result[gvl : VLMAX] = 0

```

## 2.5.6 Elementwise vector-vector floating-point dot-product

**Instruction:** ['vfdot.vv']

**Prototypes:**

- *float16xm1\_t vfdotvv\_float16xm1* (*float16xm1\_t a, float16xm1\_t b*, unsigned int gvl)
- *float16xm2\_t vfdotvv\_float16xm2* (*float16xm2\_t a, float16xm2\_t b*, unsigned int gvl)
- *float16xm4\_t vfdotvv\_float16xm4* (*float16xm4\_t a, float16xm4\_t b*, unsigned int gvl)
- *float16xm8\_t vfdotvv\_float16xm8* (*float16xm8\_t a, float16xm8\_t b*, unsigned int gvl)
- *float32xm1\_t vfdotvv\_float32xm1* (*float32xm1\_t a, float32xm1\_t b*, unsigned int gvl)
- *float32xm2\_t vfdotvv\_float32xm2* (*float32xm2\_t a, float32xm2\_t b*, unsigned int gvl)
- *float32xm4\_t vfdotvv\_float32xm4* (*float32xm4\_t a, float32xm4\_t b*, unsigned int gvl)
- *float32xm8\_t vfdotvv\_float32xm8* (*float32xm8\_t a, float32xm8\_t b*, unsigned int gvl)
- *float64xm1\_t vfdotvv\_float64xm1* (*float64xm1\_t a, float64xm1\_t b*, unsigned int gvl)
- *float64xm2\_t vfdotvv\_float64xm2* (*float64xm2\_t a, float64xm2\_t b*, unsigned int gvl)
- *float64xm4\_t vfdotvv\_float64xm4* (*float64xm4\_t a, float64xm4\_t b*, unsigned int gvl)
- *float64xm8\_t vfdotvv\_float64xm8* (*float64xm8\_t a, float64xm8\_t b*, unsigned int gvl)

**Operation:**

```

>>> for element = 0 to gvl - 1
    result[element] = dot-product(a[element], b[element])
    result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- *float16xm1\_t vfdotvv\_mask\_float16xm1* (*float16xm1\_t merge, float16xm1\_t a, float16xm1\_t b, e16xm1\_t mask*, unsigned int gvl)
- *float16xm2\_t vfdotvv\_mask\_float16xm2* (*float16xm2\_t merge, float16xm2\_t a, float16xm2\_t b, e16xm2\_t mask*, unsigned int gvl)
- *float16xm4\_t vfdotvv\_mask\_float16xm4* (*float16xm4\_t merge, float16xm4\_t a, float16xm4\_t b, e16xm4\_t mask*, unsigned int gvl)
- *float16xm8\_t vfdotvv\_mask\_float16xm8* (*float16xm8\_t merge, float16xm8\_t a, float16xm8\_t b, e16xm8\_t mask*, unsigned int gvl)
- *float32xm1\_t vfdotvv\_mask\_float32xm1* (*float32xm1\_t merge, float32xm1\_t a, float32xm1\_t b, e32xm1\_t mask*, unsigned int gvl)
- *float32xm2\_t vfdotvv\_mask\_float32xm2* (*float32xm2\_t merge, float32xm2\_t a, float32xm2\_t b, e32xm2\_t mask*, unsigned int gvl)
- *float32xm4\_t vfdotvv\_mask\_float32xm4* (*float32xm4\_t merge, float32xm4\_t a, float32xm4\_t b, e32xm4\_t mask*, unsigned int gvl)
- *float32xm8\_t vfdotvv\_mask\_float32xm8* (*float32xm8\_t merge, float32xm8\_t a, float32xm8\_t b, e32xm8\_t mask*, unsigned int gvl)

- `float64xm1_t vfdotvv_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `float64xm1_t b`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfdotvv_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `float64xm2_t b`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfdotvv_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `float64xm4_t b`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vfdotvv_mask_float64xm8` (`float64xm8_t merge`, `float64xm8_t a`, `float64xm8_t b`, `e64xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = dot-product(a[element], b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.7 Floating-point vector-scalar multiply and add(overwrite addend)

Instruction: ['vfmacc.vf']

Prototypes:

- `float16xm1_t vfmaccvf_float16xm1` (`float16xm1_t a`, `float16_t b`, `float16xm1_t c`, unsigned int `gvl`)
- `float16xm2_t vfmaccvf_float16xm2` (`float16xm2_t a`, `float16_t b`, `float16xm2_t c`, unsigned int `gvl`)
- `float16xm4_t vfmaccvf_float16xm4` (`float16xm4_t a`, `float16_t b`, `float16xm4_t c`, unsigned int `gvl`)
- `float16xm8_t vfmaccvf_float16xm8` (`float16xm8_t a`, `float16_t b`, `float16xm8_t c`, unsigned int `gvl`)
- `float32xm1_t vfmaccvf_float32xm1` (`float32xm1_t a`, `float b`, `float32xm1_t c`, unsigned int `gvl`)
- `float32xm2_t vfmaccvf_float32xm2` (`float32xm2_t a`, `float b`, `float32xm2_t c`, unsigned int `gvl`)
- `float32xm4_t vfmaccvf_float32xm4` (`float32xm4_t a`, `float b`, `float32xm4_t c`, unsigned int `gvl`)
- `float32xm8_t vfmaccvf_float32xm8` (`float32xm8_t a`, `float b`, `float32xm8_t c`, unsigned int `gvl`)
- `float64xm1_t vfmaccvf_float64xm1` (`float64xm1_t a`, `double b`, `float64xm1_t c`, unsigned int `gvl`)
- `float64xm2_t vfmaccvf_float64xm2` (`float64xm2_t a`, `double b`, `float64xm2_t c`, unsigned int `gvl`)
- `float64xm4_t vfmaccvf_float64xm4` (`float64xm4_t a`, `double b`, `float64xm4_t c`, unsigned int `gvl`)
- `float64xm8_t vfmaccvf_float64xm8` (`float64xm8_t a`, `double b`, `float64xm8_t c`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = b[element] * c[element] + a[element]
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `float16xm1_t vfmaccvf_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `float16_t b`, `float16xm1_t c`, `e16xm1_t mask`, unsigned int `gvl`)
- `float16xm2_t vfmaccvf_mask_float16xm2` (`float16xm2_t merge`, `float16xm2_t a`, `float16_t b`, `float16xm2_t c`, `e16xm2_t mask`, unsigned int `gvl`)

- `float16xm4_t vfmaccvf_mask_float16xm4` (`float16xm4_t merge`, `float16xm4_t a`, `float16_t b`, `float16xm4_t c`, `e16xm4_t mask`, unsigned int gvl)
- `float32xm1_t vfmaccvf_mask_float32xm1` (`float32xm1_t merge`, `float32xm1_t a`, `float b`, `float32xm1_t c`, `e32xm1_t mask`, unsigned int gvl)
- `float32xm2_t vfmaccvf_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float b`, `float32xm2_t c`, `e32xm2_t mask`, unsigned int gvl)
- `float32xm4_t vfmaccvf_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float b`, `float32xm4_t c`, `e32xm4_t mask`, unsigned int gvl)
- `float64xm1_t vfmaccvf_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `double b`, `float64xm1_t c`, `e64xm1_t mask`, unsigned int gvl)
- `float64xm2_t vfmaccvf_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `double b`, `float64xm2_t c`, `e64xm2_t mask`, unsigned int gvl)
- `float64xm4_t vfmaccvf_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `double b`, `float64xm4_t c`, `e64xm4_t mask`, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = b[element] * c[element] + a[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.8 Floating-point vector-vector multiply and add(overwrite addend)

Instruction: ['vfmacc.vv']

Prototypes:

- `float16xm1_t vfmaccvv_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, `float16xm1_t c`, unsigned int gvl)
- `float16xm2_t vfmaccvv_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, `float16xm2_t c`, unsigned int gvl)
- `float16xm4_t vfmaccvv_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, `float16xm4_t c`, unsigned int gvl)
- `float16xm8_t vfmaccvv_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, `float16xm8_t c`, unsigned int gvl)
- `float32xm1_t vfmaccvv_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, `float32xm1_t c`, unsigned int gvl)
- `float32xm2_t vfmaccvv_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, `float32xm2_t c`, unsigned int gvl)
- `float32xm4_t vfmaccvv_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, `float32xm4_t c`, unsigned int gvl)
- `float32xm8_t vfmaccvv_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, `float32xm8_t c`, unsigned int gvl)
- `float64xm1_t vfmaccvv_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, `float64xm1_t c`, unsigned int gvl)
- `float64xm2_t vfmaccvv_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, `float64xm2_t c`, unsigned int gvl)

- *float64xm4\_t vfmaccvv\_float64xm4* (*float64xm4\_t a, float64xm4\_t b, float64xm4\_t c*, unsigned int *gvl*)
- *float64xm8\_t vfmaccvv\_float64xm8* (*float64xm8\_t a, float64xm8\_t b, float64xm8\_t c*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = b[element] * c[element] + a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfmaccvv\_mask\_float16xm1* (*float16xm1\_t merge, float16xm1\_t a, float16xm1\_t b, float16xm1\_t c, e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfmaccvv\_mask\_float16xm2* (*float16xm2\_t merge, float16xm2\_t a, float16xm2\_t b, float16xm2\_t c, e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfmaccvv\_mask\_float16xm4* (*float16xm4\_t merge, float16xm4\_t a, float16xm4\_t b, float16xm4\_t c, e16xm4\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfmaccvv\_mask\_float32xm1* (*float32xm1\_t merge, float32xm1\_t a, float32xm1\_t b, float32xm1\_t c, e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfmaccvv\_mask\_float32xm2* (*float32xm2\_t merge, float32xm2\_t a, float32xm2\_t b, float32xm2\_t c, e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfmaccvv\_mask\_float32xm4* (*float32xm4\_t merge, float32xm4\_t a, float32xm4\_t b, float32xm4\_t c, e32xm4\_t mask*, unsigned int *gvl*)
- *float64xm1\_t vfmaccvv\_mask\_float64xm1* (*float64xm1\_t merge, float64xm1\_t a, float64xm1\_t b, float64xm1\_t c, e64xm1\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vfmaccvv\_mask\_float64xm2* (*float64xm2\_t merge, float64xm2\_t a, float64xm2\_t b, float64xm2\_t c, e64xm2\_t mask*, unsigned int *gvl*)
- *float64xm4\_t vfmaccvv\_mask\_float64xm4* (*float64xm4\_t merge, float64xm4\_t a, float64xm4\_t b, float64xm4\_t c, e64xm4\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = b[element] * c[element] + a[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.9 Floating-point vector-scalar multiply and add(overwrite multiplicand)

**Instruction:** ['vfmadd.vf']**Prototypes:**

- *float16xm1\_t vfmaddvf\_float16xm1* (*float16xm1\_t a, float16\_t b, float16xm1\_t c*, unsigned int *gvl*)
- *float16xm2\_t vfmaddvf\_float16xm2* (*float16xm2\_t a, float16\_t b, float16xm2\_t c*, unsigned int *gvl*)
- *float16xm4\_t vfmaddvf\_float16xm4* (*float16xm4\_t a, float16\_t b, float16xm4\_t c*, unsigned int *gvl*)
- *float16xm8\_t vfmaddvf\_float16xm8* (*float16xm8\_t a, float16\_t b, float16xm8\_t c*, unsigned int *gvl*)
- *float32xm1\_t vfmaddvf\_float32xm1* (*float32xm1\_t a, float b, float32xm1\_t c*, unsigned int *gvl*)

- *float32xm2\_t vfmaddvf\_float32xm2* (*float32xm2\_t a*, *float b*, *float32xm2\_t c*, unsigned int *gvl*)
- *float32xm4\_t vfmaddvf\_float32xm4* (*float32xm4\_t a*, *float b*, *float32xm4\_t c*, unsigned int *gvl*)
- *float32xm8\_t vfmaddvf\_float32xm8* (*float32xm8\_t a*, *float b*, *float32xm8\_t c*, unsigned int *gvl*)
- *float64xm1\_t vfmaddvf\_float64xm1* (*float64xm1\_t a*, *double b*, *float64xm1\_t c*, unsigned int *gvl*)
- *float64xm2\_t vfmaddvf\_float64xm2* (*float64xm2\_t a*, *double b*, *float64xm2\_t c*, unsigned int *gvl*)
- *float64xm4\_t vfmaddvf\_float64xm4* (*float64xm4\_t a*, *double b*, *float64xm4\_t c*, unsigned int *gvl*)
- *float64xm8\_t vfmaddvf\_float64xm8* (*float64xm8\_t a*, *double b*, *float64xm8\_t c*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] * b[element] + c[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfmaddvf\_mask\_float16xm1* (*float16xm1\_t merge*, *float16xm1\_t a*, *float16\_t b*, *float16xm1\_t c*, *e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfmaddvf\_mask\_float16xm2* (*float16xm2\_t merge*, *float16xm2\_t a*, *float16\_t b*, *float16xm2\_t c*, *e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfmaddvf\_mask\_float16xm4* (*float16xm4\_t merge*, *float16xm4\_t a*, *float16\_t b*, *float16xm4\_t c*, *e16xm4\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfmaddvf\_mask\_float32xm1* (*float32xm1\_t merge*, *float32xm1\_t a*, *float b*, *float32xm1\_t c*, *e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfmaddvf\_mask\_float32xm2* (*float32xm2\_t merge*, *float32xm2\_t a*, *float b*, *float32xm2\_t c*, *e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfmaddvf\_mask\_float32xm4* (*float32xm4\_t merge*, *float32xm4\_t a*, *float b*, *float32xm4\_t c*, *e32xm4\_t mask*, unsigned int *gvl*)
- *float64xm1\_t vfmaddvf\_mask\_float64xm1* (*float64xm1\_t merge*, *float64xm1\_t a*, *double b*, *float64xm1\_t c*, *e64xm1\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vfmaddvf\_mask\_float64xm2* (*float64xm2\_t merge*, *float64xm2\_t a*, *double b*, *float64xm2\_t c*, *e64xm2\_t mask*, unsigned int *gvl*)
- *float64xm4\_t vfmaddvf\_mask\_float64xm4* (*float64xm4\_t merge*, *float64xm4\_t a*, *double b*, *float64xm4\_t c*, *e64xm4\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = a[element] * b[element] + c[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.10 Floating-point vector-vector multiply and add(overwrite multiplicand)****Instruction:** ['vfmadd.vv']**Prototypes:**



- `float16xm1_t vfmaddvv_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, `float16xm1_t c`, unsigned int `gvl`)
- `float16xm2_t vfmaddvv_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, `float16xm2_t c`, unsigned int `gvl`)
- `float16xm4_t vfmaddvv_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, `float16xm4_t c`, unsigned int `gvl`)
- `float16xm8_t vfmaddvv_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, `float16xm8_t c`, unsigned int `gvl`)
- `float32xm1_t vfmaddvv_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, `float32xm1_t c`, unsigned int `gvl`)
- `float32xm2_t vfmaddvv_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, `float32xm2_t c`, unsigned int `gvl`)
- `float32xm4_t vfmaddvv_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, `float32xm4_t c`, unsigned int `gvl`)
- `float32xm8_t vfmaddvv_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, `float32xm8_t c`, unsigned int `gvl`)
- `float64xm1_t vfmaddvv_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, `float64xm1_t c`, unsigned int `gvl`)
- `float64xm2_t vfmaddvv_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, `float64xm2_t c`, unsigned int `gvl`)
- `float64xm4_t vfmaddvv_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, `float64xm4_t c`, unsigned int `gvl`)
- `float64xm8_t vfmaddvv_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, `float64xm8_t c`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] * b[element] + c[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vfmaddvv_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `float16xm1_t b`, `float16xm1_t c`, `e16xm1_t mask`, unsigned int `gvl`)
- `float16xm2_t vfmaddvv_mask_float16xm2` (`float16xm2_t merge`, `float16xm2_t a`, `float16xm2_t b`, `float16xm2_t c`, `e16xm2_t mask`, unsigned int `gvl`)
- `float16xm4_t vfmaddvv_mask_float16xm4` (`float16xm4_t merge`, `float16xm4_t a`, `float16xm4_t b`, `float16xm4_t c`, `e16xm4_t mask`, unsigned int `gvl`)
- `float32xm1_t vfmaddvv_mask_float32xm1` (`float32xm1_t merge`, `float32xm1_t a`, `float32xm1_t b`, `float32xm1_t c`, `e32xm1_t mask`, unsigned int `gvl`)
- `float32xm2_t vfmaddvv_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float32xm2_t b`, `float32xm2_t c`, `e32xm2_t mask`, unsigned int `gvl`)
- `float32xm4_t vfmaddvv_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float32xm4_t b`, `float32xm4_t c`, `e32xm4_t mask`, unsigned int `gvl`)
- `float64xm1_t vfmaddvv_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `float64xm1_t b`, `float64xm1_t c`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfmaddvv_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `float64xm2_t b`, `float64xm2_t c`, `e64xm2_t mask`, unsigned int `gvl`)

- *float64xm4\_t* **vfmaddvv\_mask\_float64xm4** (*float64xm4\_t* merge, *float64xm4\_t* a, *float64xm4\_t* b, *float64xm4\_t* c, *e64xm4\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = a[element] * b[element] + c[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

### 2.5.11 Elementwise vector-scalar floating-point maximum

Instruction: ['vfmmax.vf']

Prototypes:

- *float16xm1\_t* **vfmmaxvf\_float16xm1** (*float16xm1\_t* a, *float16\_t* b, unsigned int gvl)
- *float16xm2\_t* **vfmmaxvf\_float16xm2** (*float16xm2\_t* a, *float16\_t* b, unsigned int gvl)
- *float16xm4\_t* **vfmmaxvf\_float16xm4** (*float16xm4\_t* a, *float16\_t* b, unsigned int gvl)
- *float16xm8\_t* **vfmmaxvf\_float16xm8** (*float16xm8\_t* a, *float16\_t* b, unsigned int gvl)
- *float32xm1\_t* **vfmmaxvf\_float32xm1** (*float32xm1\_t* a, *float b*, unsigned int gvl)
- *float32xm2\_t* **vfmmaxvf\_float32xm2** (*float32xm2\_t* a, *float b*, unsigned int gvl)
- *float32xm4\_t* **vfmmaxvf\_float32xm4** (*float32xm4\_t* a, *float b*, unsigned int gvl)
- *float32xm8\_t* **vfmmaxvf\_float32xm8** (*float32xm8\_t* a, *float b*, unsigned int gvl)
- *float64xm1\_t* **vfmmaxvf\_float64xm1** (*float64xm1\_t* a, *double b*, unsigned int gvl)
- *float64xm2\_t* **vfmmaxvf\_float64xm2** (*float64xm2\_t* a, *double b*, unsigned int gvl)
- *float64xm4\_t* **vfmmaxvf\_float64xm4** (*float64xm4\_t* a, *double b*, unsigned int gvl)
- *float64xm8\_t* **vfmmaxvf\_float64xm8** (*float64xm8\_t* a, *double b*, unsigned int gvl)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = max(a[element], b)
result[gvl : VLMAX] = 0
```

Masked prototypes:

- *float16xm1\_t* **vfmmaxvf\_mask\_float16xm1** (*float16xm1\_t* merge, *float16xm1\_t* a, *float16\_t* b, *e16xm1\_t* mask, unsigned int gvl)
- *float16xm2\_t* **vfmmaxvf\_mask\_float16xm2** (*float16xm2\_t* merge, *float16xm2\_t* a, *float16\_t* b, *e16xm2\_t* mask, unsigned int gvl)
- *float16xm4\_t* **vfmmaxvf\_mask\_float16xm4** (*float16xm4\_t* merge, *float16xm4\_t* a, *float16\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *float16xm8\_t* **vfmmaxvf\_mask\_float16xm8** (*float16xm8\_t* merge, *float16xm8\_t* a, *float16\_t* b, *e16xm8\_t* mask, unsigned int gvl)
- *float32xm1\_t* **vfmmaxvf\_mask\_float32xm1** (*float32xm1\_t* merge, *float32xm1\_t* a, *float b*, *e32xm1\_t* mask, unsigned int gvl)

- `float32xm2_t vfmavf_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float b`, `e32xm2_t mask`, unsigned int `gvl`)
- `float32xm4_t vfmavf_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float b`, `e32xm4_t mask`, unsigned int `gvl`)
- `float32xm8_t vfmavf_mask_float32xm8` (`float32xm8_t merge`, `float32xm8_t a`, `float b`, `e32xm8_t mask`, unsigned int `gvl`)
- `float64xm1_t vfmavf_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `double b`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfmavf_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `double b`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfmavf_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `double b`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vfmavf_mask_float64xm8` (`float64xm8_t merge`, `float64xm8_t a`, `double b`, `e64xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = max(a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.12 Elementwise vector-vector floating-point maximum

Instruction: ['vfmavv']

Prototypes:

- `float16xm1_t vfmavv_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, unsigned int `gvl`)
- `float16xm2_t vfmavv_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, unsigned int `gvl`)
- `float16xm4_t vfmavv_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, unsigned int `gvl`)
- `float16xm8_t vfmavv_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, unsigned int `gvl`)
- `float32xm1_t vfmavv_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, unsigned int `gvl`)
- `float32xm2_t vfmavv_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, unsigned int `gvl`)
- `float32xm4_t vfmavv_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, unsigned int `gvl`)
- `float32xm8_t vfmavv_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, unsigned int `gvl`)
- `float64xm1_t vfmavv_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, unsigned int `gvl`)
- `float64xm2_t vfmavv_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, unsigned int `gvl`)
- `float64xm4_t vfmavv_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, unsigned int `gvl`)
- `float64xm8_t vfmavv_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = max(a[element], b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vfmavv_mask_float16xm1` (`float16xm1_t` merge, `float16xm1_t` a, `float16xm1_t` b, `e16xm1_t` mask, unsigned int gvl)
- `float16xm2_t vfmavv_mask_float16xm2` (`float16xm2_t` merge, `float16xm2_t` a, `float16xm2_t` b, `e16xm2_t` mask, unsigned int gvl)
- `float16xm4_t vfmavv_mask_float16xm4` (`float16xm4_t` merge, `float16xm4_t` a, `float16xm4_t` b, `e16xm4_t` mask, unsigned int gvl)
- `float16xm8_t vfmavv_mask_float16xm8` (`float16xm8_t` merge, `float16xm8_t` a, `float16xm8_t` b, `e16xm8_t` mask, unsigned int gvl)
- `float32xm1_t vfmavv_mask_float32xm1` (`float32xm1_t` merge, `float32xm1_t` a, `float32xm1_t` b, `e32xm1_t` mask, unsigned int gvl)
- `float32xm2_t vfmavv_mask_float32xm2` (`float32xm2_t` merge, `float32xm2_t` a, `float32xm2_t` b, `e32xm2_t` mask, unsigned int gvl)
- `float32xm4_t vfmavv_mask_float32xm4` (`float32xm4_t` merge, `float32xm4_t` a, `float32xm4_t` b, `e32xm4_t` mask, unsigned int gvl)
- `float32xm8_t vfmavv_mask_float32xm8` (`float32xm8_t` merge, `float32xm8_t` a, `float32xm8_t` b, `e32xm8_t` mask, unsigned int gvl)
- `float64xm1_t vfmavv_mask_float64xm1` (`float64xm1_t` merge, `float64xm1_t` a, `float64xm1_t` b, `e64xm1_t` mask, unsigned int gvl)
- `float64xm2_t vfmavv_mask_float64xm2` (`float64xm2_t` merge, `float64xm2_t` a, `float64xm2_t` b, `e64xm2_t` mask, unsigned int gvl)
- `float64xm4_t vfmavv_mask_float64xm4` (`float64xm4_t` merge, `float64xm4_t` a, `float64xm4_t` b, `e64xm4_t` mask, unsigned int gvl)
- `float64xm8_t vfmavv_mask_float64xm8` (`float64xm8_t` merge, `float64xm8_t` a, `float64xm8_t` b, `e64xm8_t` mask, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = max(a[element], b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.13 Elementwise vector-scalar floating-point minimum****Instruction:** [`'vfmavf'`]**Prototypes:**

- `float16xm1_t vfmavf_float16xm1` (`float16xm1_t` a, `float16_t` b, unsigned int gvl)
- `float16xm2_t vfmavf_float16xm2` (`float16xm2_t` a, `float16_t` b, unsigned int gvl)
- `float16xm4_t vfmavf_float16xm4` (`float16xm4_t` a, `float16_t` b, unsigned int gvl)
- `float16xm8_t vfmavf_float16xm8` (`float16xm8_t` a, `float16_t` b, unsigned int gvl)
- `float32xm1_t vfmavf_float32xm1` (`float32xm1_t` a, `float` b, unsigned int gvl)
- `float32xm2_t vfmavf_float32xm2` (`float32xm2_t` a, `float` b, unsigned int gvl)
- `float32xm4_t vfmavf_float32xm4` (`float32xm4_t` a, `float` b, unsigned int gvl)

- *float32xm8\_t* **vfminvf\_float32xm8** (*float32xm8\_t* *a*, float *b*, unsigned int *gvl*)
- *float64xm1\_t* **vfminvf\_float64xm1** (*float64xm1\_t* *a*, double *b*, unsigned int *gvl*)
- *float64xm2\_t* **vfminvf\_float64xm2** (*float64xm2\_t* *a*, double *b*, unsigned int *gvl*)
- *float64xm4\_t* **vfminvf\_float64xm4** (*float64xm4\_t* *a*, double *b*, unsigned int *gvl*)
- *float64xm8\_t* **vfminvf\_float64xm8** (*float64xm8\_t* *a*, double *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = min(a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t* **vfminvf\_mask\_float16xm1** (*float16xm1\_t* *merge*, *float16xm1\_t* *a*, float16\_t *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *float16xm2\_t* **vfminvf\_mask\_float16xm2** (*float16xm2\_t* *merge*, *float16xm2\_t* *a*, float16\_t *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *float16xm4\_t* **vfminvf\_mask\_float16xm4** (*float16xm4\_t* *merge*, *float16xm4\_t* *a*, float16\_t *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *float16xm8\_t* **vfminvf\_mask\_float16xm8** (*float16xm8\_t* *merge*, *float16xm8\_t* *a*, float16\_t *b*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- *float32xm1\_t* **vfminvf\_mask\_float32xm1** (*float32xm1\_t* *merge*, *float32xm1\_t* *a*, float *b*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *float32xm2\_t* **vfminvf\_mask\_float32xm2** (*float32xm2\_t* *merge*, *float32xm2\_t* *a*, float *b*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *float32xm4\_t* **vfminvf\_mask\_float32xm4** (*float32xm4\_t* *merge*, *float32xm4\_t* *a*, float *b*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *float32xm8\_t* **vfminvf\_mask\_float32xm8** (*float32xm8\_t* *merge*, *float32xm8\_t* *a*, float *b*, *e32xm8\_t* *mask*, unsigned int *gvl*)
- *float64xm1\_t* **vfminvf\_mask\_float64xm1** (*float64xm1\_t* *merge*, *float64xm1\_t* *a*, double *b*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *float64xm2\_t* **vfminvf\_mask\_float64xm2** (*float64xm2\_t* *merge*, *float64xm2\_t* *a*, double *b*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *float64xm4\_t* **vfminvf\_mask\_float64xm4** (*float64xm4\_t* *merge*, *float64xm4\_t* *a*, double *b*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *float64xm8\_t* **vfminvf\_mask\_float64xm8** (*float64xm8\_t* *merge*, *float64xm8\_t* *a*, double *b*, *e64xm8\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = min(a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.14 Elementwise vector-vector floating-point minimum

**Instruction:** ['vfmmin.vv']

**Prototypes:**

- *float16xm1\_t vfmminvv\_float16xm1* (*float16xm1\_t a, float16xm1\_t b*, unsigned int *gvl*)
- *float16xm2\_t vfmminvv\_float16xm2* (*float16xm2\_t a, float16xm2\_t b*, unsigned int *gvl*)
- *float16xm4\_t vfmminvv\_float16xm4* (*float16xm4\_t a, float16xm4\_t b*, unsigned int *gvl*)
- *float16xm8\_t vfmminvv\_float16xm8* (*float16xm8\_t a, float16xm8\_t b*, unsigned int *gvl*)
- *float32xm1\_t vfmminvv\_float32xm1* (*float32xm1\_t a, float32xm1\_t b*, unsigned int *gvl*)
- *float32xm2\_t vfmminvv\_float32xm2* (*float32xm2\_t a, float32xm2\_t b*, unsigned int *gvl*)
- *float32xm4\_t vfmminvv\_float32xm4* (*float32xm4\_t a, float32xm4\_t b*, unsigned int *gvl*)
- *float32xm8\_t vfmminvv\_float32xm8* (*float32xm8\_t a, float32xm8\_t b*, unsigned int *gvl*)
- *float64xm1\_t vfmminvv\_float64xm1* (*float64xm1\_t a, float64xm1\_t b*, unsigned int *gvl*)
- *float64xm2\_t vfmminvv\_float64xm2* (*float64xm2\_t a, float64xm2\_t b*, unsigned int *gvl*)
- *float64xm4\_t vfmminvv\_float64xm4* (*float64xm4\_t a, float64xm4\_t b*, unsigned int *gvl*)
- *float64xm8\_t vfmminvv\_float64xm8* (*float64xm8\_t a, float64xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = min(a[element], b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfmminvv\_mask\_float16xm1* (*float16xm1\_t merge, float16xm1\_t a, float16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfmminvv\_mask\_float16xm2* (*float16xm2\_t merge, float16xm2\_t a, float16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfmminvv\_mask\_float16xm4* (*float16xm4\_t merge, float16xm4\_t a, float16xm4\_t b, e16xm4\_t mask*, unsigned int *gvl*)
- *float16xm8\_t vfmminvv\_mask\_float16xm8* (*float16xm8\_t merge, float16xm8\_t a, float16xm8\_t b, e16xm8\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfmminvv\_mask\_float32xm1* (*float32xm1\_t merge, float32xm1\_t a, float32xm1\_t b, e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfmminvv\_mask\_float32xm2* (*float32xm2\_t merge, float32xm2\_t a, float32xm2\_t b, e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfmminvv\_mask\_float32xm4* (*float32xm4\_t merge, float32xm4\_t a, float32xm4\_t b, e32xm4\_t mask*, unsigned int *gvl*)
- *float32xm8\_t vfmminvv\_mask\_float32xm8* (*float32xm8\_t merge, float32xm8\_t a, float32xm8\_t b, e32xm8\_t mask*, unsigned int *gvl*)
- *float64xm1\_t vfmminvv\_mask\_float64xm1* (*float64xm1\_t merge, float64xm1\_t a, float64xm1\_t b, e64xm1\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vfmminvv\_mask\_float64xm2* (*float64xm2\_t merge, float64xm2\_t a, float64xm2\_t b, e64xm2\_t mask*, unsigned int *gvl*)



- *float64xm4\_t* **vfminvv\_mask\_float64xm4** (*float64xm4\_t* merge, *float64xm4\_t* a, *float64xm4\_t* b, *e64xm4\_t* mask, unsigned int gvl)
- *float64xm8\_t* **vfminvv\_mask\_float64xm8** (*float64xm8\_t* merge, *float64xm8\_t* a, *float64xm8\_t* b, *e64xm8\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = min(a[element], b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.15 Floating-point vector-scalar multiply and sub(overwrite subtrahend)

Instruction: ['vfmsac.vf']

Prototypes:

- *float16xm1\_t* **vfmsacvf\_float16xm1** (*float16xm1\_t* a, *float16\_t* b, *float16xm1\_t* c, unsigned int gvl)
- *float16xm2\_t* **vfmsacvf\_float16xm2** (*float16xm2\_t* a, *float16\_t* b, *float16xm2\_t* c, unsigned int gvl)
- *float16xm4\_t* **vfmsacvf\_float16xm4** (*float16xm4\_t* a, *float16\_t* b, *float16xm4\_t* c, unsigned int gvl)
- *float16xm8\_t* **vfmsacvf\_float16xm8** (*float16xm8\_t* a, *float16\_t* b, *float16xm8\_t* c, unsigned int gvl)
- *float32xm1\_t* **vfmsacvf\_float32xm1** (*float32xm1\_t* a, *float* b, *float32xm1\_t* c, unsigned int gvl)
- *float32xm2\_t* **vfmsacvf\_float32xm2** (*float32xm2\_t* a, *float* b, *float32xm2\_t* c, unsigned int gvl)
- *float32xm4\_t* **vfmsacvf\_float32xm4** (*float32xm4\_t* a, *float* b, *float32xm4\_t* c, unsigned int gvl)
- *float32xm8\_t* **vfmsacvf\_float32xm8** (*float32xm8\_t* a, *float* b, *float32xm8\_t* c, unsigned int gvl)
- *float64xm1\_t* **vfmsacvf\_float64xm1** (*float64xm1\_t* a, *double* b, *float64xm1\_t* c, unsigned int gvl)
- *float64xm2\_t* **vfmsacvf\_float64xm2** (*float64xm2\_t* a, *double* b, *float64xm2\_t* c, unsigned int gvl)
- *float64xm4\_t* **vfmsacvf\_float64xm4** (*float64xm4\_t* a, *double* b, *float64xm4\_t* c, unsigned int gvl)
- *float64xm8\_t* **vfmsacvf\_float64xm8** (*float64xm8\_t* a, *double* b, *float64xm8\_t* c, unsigned int gvl)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = b[element] * c[element] - a[element]
result[gvl : VLMAX] = 0
```

Masked prototypes:

- *float16xm1\_t* **vfmsacvf\_mask\_float16xm1** (*float16xm1\_t* merge, *float16xm1\_t* a, *float16\_t* b, *float16xm1\_t* c, *e16xm1\_t* mask, unsigned int gvl)
- *float16xm2\_t* **vfmsacvf\_mask\_float16xm2** (*float16xm2\_t* merge, *float16xm2\_t* a, *float16\_t* b, *float16xm2\_t* c, *e16xm2\_t* mask, unsigned int gvl)
- *float16xm4\_t* **vfmsacvf\_mask\_float16xm4** (*float16xm4\_t* merge, *float16xm4\_t* a, *float16\_t* b, *float16xm4\_t* c, *e16xm4\_t* mask, unsigned int gvl)
- *float32xm1\_t* **vfmsacvf\_mask\_float32xm1** (*float32xm1\_t* merge, *float32xm1\_t* a, *float* b, *float32xm1\_t* c, *e32xm1\_t* mask, unsigned int gvl)

- `float32xm2_t vfmsacvf_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float b`, `float32xm2_t c`, `e32xm2_t mask`, unsigned int `gvl`)
- `float32xm4_t vfmsacvf_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float b`, `float32xm4_t c`, `e32xm4_t mask`, unsigned int `gvl`)
- `float64xm1_t vfmsacvf_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `double b`, `float64xm1_t c`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfmsacvf_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `double b`, `float64xm2_t c`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfmsacvf_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `double b`, `float64xm4_t c`, `e64xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = b[element] * c[element] - a[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.16 Floating-point vector-vector multiply and sub(overwrite subtrahend)****Instruction:** [`'vfmsac.vv'`]**Prototypes:**

- `float16xm1_t vfmsacvv_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, `float16xm1_t c`, unsigned int `gvl`)
- `float16xm2_t vfmsacvv_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, `float16xm2_t c`, unsigned int `gvl`)
- `float16xm4_t vfmsacvv_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, `float16xm4_t c`, unsigned int `gvl`)
- `float16xm8_t vfmsacvv_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, `float16xm8_t c`, unsigned int `gvl`)
- `float32xm1_t vfmsacvv_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, `float32xm1_t c`, unsigned int `gvl`)
- `float32xm2_t vfmsacvv_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, `float32xm2_t c`, unsigned int `gvl`)
- `float32xm4_t vfmsacvv_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, `float32xm4_t c`, unsigned int `gvl`)
- `float32xm8_t vfmsacvv_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, `float32xm8_t c`, unsigned int `gvl`)
- `float64xm1_t vfmsacvv_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, `float64xm1_t c`, unsigned int `gvl`)
- `float64xm2_t vfmsacvv_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, `float64xm2_t c`, unsigned int `gvl`)
- `float64xm4_t vfmsacvv_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, `float64xm4_t c`, unsigned int `gvl`)
- `float64xm8_t vfmsacvv_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, `float64xm8_t c`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = b[element] * c[element] - a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfmsacvv\_mask\_float16xm1* (*float16xm1\_t merge, float16xm1\_t a, float16xm1\_t b, float16xm1\_t c, e16xm1\_t mask, unsigned int gvl*)
- *float16xm2\_t vfmsacvv\_mask\_float16xm2* (*float16xm2\_t merge, float16xm2\_t a, float16xm2\_t b, float16xm2\_t c, e16xm2\_t mask, unsigned int gvl*)
- *float16xm4\_t vfmsacvv\_mask\_float16xm4* (*float16xm4\_t merge, float16xm4\_t a, float16xm4\_t b, float16xm4\_t c, e16xm4\_t mask, unsigned int gvl*)
- *float32xm1\_t vfmsacvv\_mask\_float32xm1* (*float32xm1\_t merge, float32xm1\_t a, float32xm1\_t b, float32xm1\_t c, e32xm1\_t mask, unsigned int gvl*)
- *float32xm2\_t vfmsacvv\_mask\_float32xm2* (*float32xm2\_t merge, float32xm2\_t a, float32xm2\_t b, float32xm2\_t c, e32xm2\_t mask, unsigned int gvl*)
- *float32xm4\_t vfmsacvv\_mask\_float32xm4* (*float32xm4\_t merge, float32xm4\_t a, float32xm4\_t b, float32xm4\_t c, e32xm4\_t mask, unsigned int gvl*)
- *float64xm1\_t vfmsacvv\_mask\_float64xm1* (*float64xm1\_t merge, float64xm1\_t a, float64xm1\_t b, float64xm1\_t c, e64xm1\_t mask, unsigned int gvl*)
- *float64xm2\_t vfmsacvv\_mask\_float64xm2* (*float64xm2\_t merge, float64xm2\_t a, float64xm2\_t b, float64xm2\_t c, e64xm2\_t mask, unsigned int gvl*)
- *float64xm4\_t vfmsacvv\_mask\_float64xm4* (*float64xm4\_t merge, float64xm4\_t a, float64xm4\_t b, float64xm4\_t c, e64xm4\_t mask, unsigned int gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = b[element] * c[element] - a[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.17 Floating-point vector-scalar multiply and sub(overwrite multiplicand)****Instruction:** [*'vfmsub.vf'*]**Prototypes:**

- *float16xm1\_t vfmsubvf\_float16xm1* (*float16xm1\_t a, float16\_t b, float16xm1\_t c, unsigned int gvl*)
- *float16xm2\_t vfmsubvf\_float16xm2* (*float16xm2\_t a, float16\_t b, float16xm2\_t c, unsigned int gvl*)
- *float16xm4\_t vfmsubvf\_float16xm4* (*float16xm4\_t a, float16\_t b, float16xm4\_t c, unsigned int gvl*)
- *float16xm8\_t vfmsubvf\_float16xm8* (*float16xm8\_t a, float16\_t b, float16xm8\_t c, unsigned int gvl*)
- *float32xm1\_t vfmsubvf\_float32xm1* (*float32xm1\_t a, float b, float32xm1\_t c, unsigned int gvl*)
- *float32xm2\_t vfmsubvf\_float32xm2* (*float32xm2\_t a, float b, float32xm2\_t c, unsigned int gvl*)
- *float32xm4\_t vfmsubvf\_float32xm4* (*float32xm4\_t a, float b, float32xm4\_t c, unsigned int gvl*)
- *float32xm8\_t vfmsubvf\_float32xm8* (*float32xm8\_t a, float b, float32xm8\_t c, unsigned int gvl*)

- `float64xm1_t vfmsubvf_float64xm1` (`float64xm1_t a`, double `b`, `float64xm1_t c`, unsigned int `gvl`)
- `float64xm2_t vfmsubvf_float64xm2` (`float64xm2_t a`, double `b`, `float64xm2_t c`, unsigned int `gvl`)
- `float64xm4_t vfmsubvf_float64xm4` (`float64xm4_t a`, double `b`, `float64xm4_t c`, unsigned int `gvl`)
- `float64xm8_t vfmsubvf_float64xm8` (`float64xm8_t a`, double `b`, `float64xm8_t c`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = a[element] * b[element] - c[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vfmsubvf_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `float16_t b`, `float16xm1_t c`, `e16xm1_t mask`, unsigned int `gvl`)
- `float16xm2_t vfmsubvf_mask_float16xm2` (`float16xm2_t merge`, `float16xm2_t a`, `float16_t b`, `float16xm2_t c`, `e16xm2_t mask`, unsigned int `gvl`)
- `float16xm4_t vfmsubvf_mask_float16xm4` (`float16xm4_t merge`, `float16xm4_t a`, `float16_t b`, `float16xm4_t c`, `e16xm4_t mask`, unsigned int `gvl`)
- `float32xm1_t vfmsubvf_mask_float32xm1` (`float32xm1_t merge`, `float32xm1_t a`, `float b`, `float32xm1_t c`, `e32xm1_t mask`, unsigned int `gvl`)
- `float32xm2_t vfmsubvf_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float b`, `float32xm2_t c`, `e32xm2_t mask`, unsigned int `gvl`)
- `float32xm4_t vfmsubvf_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float b`, `float32xm4_t c`, `e32xm4_t mask`, unsigned int `gvl`)
- `float64xm1_t vfmsubvf_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, double `b`, `float64xm1_t c`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfmsubvf_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, double `b`, `float64xm2_t c`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfmsubvf_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, double `b`, `float64xm4_t c`, `e64xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = a[element] * b[element] - c[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.18 Floating-point vector-vector multiply and sub(overwrite multiplicand)****Instruction:** [`'vfmsub.vv'`]**Prototypes:**

- `float16xm1_t vfmsubvv_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, `float16xm1_t c`, unsigned int `gvl`)
- `float16xm2_t vfmsubvv_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, `float16xm2_t c`, unsigned int `gvl`)

- `float16xm4_t vfmsubvv_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, `float16xm4_t c`, unsigned int gvl)
- `float16xm8_t vfmsubvv_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, `float16xm8_t c`, unsigned int gvl)
- `float32xm1_t vfmsubvv_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, `float32xm1_t c`, unsigned int gvl)
- `float32xm2_t vfmsubvv_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, `float32xm2_t c`, unsigned int gvl)
- `float32xm4_t vfmsubvv_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, `float32xm4_t c`, unsigned int gvl)
- `float32xm8_t vfmsubvv_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, `float32xm8_t c`, unsigned int gvl)
- `float64xm1_t vfmsubvv_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, `float64xm1_t c`, unsigned int gvl)
- `float64xm2_t vfmsubvv_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, `float64xm2_t c`, unsigned int gvl)
- `float64xm4_t vfmsubvv_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, `float64xm4_t c`, unsigned int gvl)
- `float64xm8_t vfmsubvv_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, `float64xm8_t c`, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] * b[element] - c[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vfmsubvv_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `float16xm1_t b`, `float16xm1_t c`, `e16xm1_t mask`, unsigned int gvl)
- `float16xm2_t vfmsubvv_mask_float16xm2` (`float16xm2_t merge`, `float16xm2_t a`, `float16xm2_t b`, `float16xm2_t c`, `e16xm2_t mask`, unsigned int gvl)
- `float16xm4_t vfmsubvv_mask_float16xm4` (`float16xm4_t merge`, `float16xm4_t a`, `float16xm4_t b`, `float16xm4_t c`, `e16xm4_t mask`, unsigned int gvl)
- `float32xm1_t vfmsubvv_mask_float32xm1` (`float32xm1_t merge`, `float32xm1_t a`, `float32xm1_t b`, `float32xm1_t c`, `e32xm1_t mask`, unsigned int gvl)
- `float32xm2_t vfmsubvv_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float32xm2_t b`, `float32xm2_t c`, `e32xm2_t mask`, unsigned int gvl)
- `float32xm4_t vfmsubvv_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float32xm4_t b`, `float32xm4_t c`, `e32xm4_t mask`, unsigned int gvl)
- `float64xm1_t vfmsubvv_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `float64xm1_t b`, `float64xm1_t c`, `e64xm1_t mask`, unsigned int gvl)
- `float64xm2_t vfmsubvv_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `float64xm2_t b`, `float64xm2_t c`, `e64xm2_t mask`, unsigned int gvl)
- `float64xm4_t vfmsubvv_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `float64xm4_t b`, `float64xm4_t c`, `e64xm4_t mask`, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = a[element] * b[element] - c[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.19 Elementwise vector-scalar floating-point multiplication

**Instruction:** ['vfmul.vf']

**Prototypes:**

- *float16xm1\_t vfmulvf\_float16xm1* (*float16xm1\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float16xm2\_t vfmulvf\_float16xm2* (*float16xm2\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float16xm4\_t vfmulvf\_float16xm4* (*float16xm4\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float16xm8\_t vfmulvf\_float16xm8* (*float16xm8\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float32xm1\_t vfmulvf\_float32xm1* (*float32xm1\_t a*, *float b*, unsigned int *gvl*)
- *float32xm2\_t vfmulvf\_float32xm2* (*float32xm2\_t a*, *float b*, unsigned int *gvl*)
- *float32xm4\_t vfmulvf\_float32xm4* (*float32xm4\_t a*, *float b*, unsigned int *gvl*)
- *float32xm8\_t vfmulvf\_float32xm8* (*float32xm8\_t a*, *float b*, unsigned int *gvl*)
- *float64xm1\_t vfmulvf\_float64xm1* (*float64xm1\_t a*, *double b*, unsigned int *gvl*)
- *float64xm2\_t vfmulvf\_float64xm2* (*float64xm2\_t a*, *double b*, unsigned int *gvl*)
- *float64xm4\_t vfmulvf\_float64xm4* (*float64xm4\_t a*, *double b*, unsigned int *gvl*)
- *float64xm8\_t vfmulvf\_float64xm8* (*float64xm8\_t a*, *double b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = a[element] * b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfmulvf\_mask\_float16xm1* (*float16xm1\_t merge*, *float16xm1\_t a*, *float16\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfmulvf\_mask\_float16xm2* (*float16xm2\_t merge*, *float16xm2\_t a*, *float16\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfmulvf\_mask\_float16xm4* (*float16xm4\_t merge*, *float16xm4\_t a*, *float16\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *float16xm8\_t vfmulvf\_mask\_float16xm8* (*float16xm8\_t merge*, *float16xm8\_t a*, *float16\_t b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfmulvf\_mask\_float32xm1* (*float32xm1\_t merge*, *float32xm1\_t a*, *float b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfmulvf\_mask\_float32xm2* (*float32xm2\_t merge*, *float32xm2\_t a*, *float b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfmulvf\_mask\_float32xm4* (*float32xm4\_t merge*, *float32xm4\_t a*, *float b*, *e32xm4\_t mask*, unsigned int *gvl*)



- `float32xm8_t vfmulvf_mask_float32xm8` (`float32xm8_t merge`, `float32xm8_t a`, `float b`, `e32xm8_t mask`, unsigned int `gvl`)
- `float64xm1_t vfmulvf_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `double b`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfmulvf_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `double b`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfmulvf_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `double b`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vfmulvf_mask_float64xm8` (`float64xm8_t merge`, `float64xm8_t a`, `double b`, `e64xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = a[element] * b
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.20 Elementwise vector-vector floating-point multiplication

Instruction: ['vfmul.vv']

Prototypes:

- `float16xm1_t vfmulvv_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, unsigned int `gvl`)
- `float16xm2_t vfmulvv_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, unsigned int `gvl`)
- `float16xm4_t vfmulvv_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, unsigned int `gvl`)
- `float16xm8_t vfmulvv_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, unsigned int `gvl`)
- `float32xm1_t vfmulvv_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, unsigned int `gvl`)
- `float32xm2_t vfmulvv_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, unsigned int `gvl`)
- `float32xm4_t vfmulvv_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, unsigned int `gvl`)
- `float32xm8_t vfmulvv_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, unsigned int `gvl`)
- `float64xm1_t vfmulvv_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, unsigned int `gvl`)
- `float64xm2_t vfmulvv_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, unsigned int `gvl`)
- `float64xm4_t vfmulvv_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, unsigned int `gvl`)
- `float64xm8_t vfmulvv_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] * b[element]
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `float16xm1_t vfmulvv_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `float16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)

- `float16xm2_t vfmulvv_mask_float16xm2` (`float16xm2_t merge`, `float16xm2_t a`, `float16xm2_t b`, `e16xm2_t mask`, unsigned int gvl)
- `float16xm4_t vfmulvv_mask_float16xm4` (`float16xm4_t merge`, `float16xm4_t a`, `float16xm4_t b`, `e16xm4_t mask`, unsigned int gvl)
- `float16xm8_t vfmulvv_mask_float16xm8` (`float16xm8_t merge`, `float16xm8_t a`, `float16xm8_t b`, `e16xm8_t mask`, unsigned int gvl)
- `float32xm1_t vfmulvv_mask_float32xm1` (`float32xm1_t merge`, `float32xm1_t a`, `float32xm1_t b`, `e32xm1_t mask`, unsigned int gvl)
- `float32xm2_t vfmulvv_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float32xm2_t b`, `e32xm2_t mask`, unsigned int gvl)
- `float32xm4_t vfmulvv_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float32xm4_t b`, `e32xm4_t mask`, unsigned int gvl)
- `float32xm8_t vfmulvv_mask_float32xm8` (`float32xm8_t merge`, `float32xm8_t a`, `float32xm8_t b`, `e32xm8_t mask`, unsigned int gvl)
- `float64xm1_t vfmulvv_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `float64xm1_t b`, `e64xm1_t mask`, unsigned int gvl)
- `float64xm2_t vfmulvv_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `float64xm2_t b`, `e64xm2_t mask`, unsigned int gvl)
- `float64xm4_t vfmulvv_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `float64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)
- `float64xm8_t vfmulvv_mask_float64xm8` (`float64xm8_t merge`, `float64xm8_t a`, `float64xm8_t b`, `e64xm8_t mask`, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = a[element] * b[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.21 Floating-point vector-scalar negate multiply and add(overwrite addend)

Instruction: [`vfnmacc.vf`]

Prototypes:

- `float16xm1_t vfnmaccvf_float16xm1` (`float16xm1_t a`, `float16_t b`, `float16xm1_t c`, unsigned int gvl)
- `float16xm2_t vfnmaccvf_float16xm2` (`float16xm2_t a`, `float16_t b`, `float16xm2_t c`, unsigned int gvl)
- `float16xm4_t vfnmaccvf_float16xm4` (`float16xm4_t a`, `float16_t b`, `float16xm4_t c`, unsigned int gvl)
- `float16xm8_t vfnmaccvf_float16xm8` (`float16xm8_t a`, `float16_t b`, `float16xm8_t c`, unsigned int gvl)
- `float32xm1_t vfnmaccvf_float32xm1` (`float32xm1_t a`, `float b`, `float32xm1_t c`, unsigned int gvl)
- `float32xm2_t vfnmaccvf_float32xm2` (`float32xm2_t a`, `float b`, `float32xm2_t c`, unsigned int gvl)
- `float32xm4_t vfnmaccvf_float32xm4` (`float32xm4_t a`, `float b`, `float32xm4_t c`, unsigned int gvl)

- *float32xm8\_t vfnmaccvf\_float32xm8* (*float32xm8\_t a*, *float b*, *float32xm8\_t c*, unsigned int *gvl*)
- *float64xm1\_t vfnmaccvf\_float64xm1* (*float64xm1\_t a*, double *b*, *float64xm1\_t c*, unsigned int *gvl*)
- *float64xm2\_t vfnmaccvf\_float64xm2* (*float64xm2\_t a*, double *b*, *float64xm2\_t c*, unsigned int *gvl*)
- *float64xm4\_t vfnmaccvf\_float64xm4* (*float64xm4\_t a*, double *b*, *float64xm4\_t c*, unsigned int *gvl*)
- *float64xm8\_t vfnmaccvf\_float64xm8* (*float64xm8\_t a*, double *b*, *float64xm8\_t c*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = -(b[element] * c[element]) - a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfnmaccvf\_mask\_float16xm1* (*float16xm1\_t merge*, *float16xm1\_t a*, *float16\_t b*, *float16xm1\_t c*, *e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfnmaccvf\_mask\_float16xm2* (*float16xm2\_t merge*, *float16xm2\_t a*, *float16\_t b*, *float16xm2\_t c*, *e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfnmaccvf\_mask\_float16xm4* (*float16xm4\_t merge*, *float16xm4\_t a*, *float16\_t b*, *float16xm4\_t c*, *e16xm4\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfnmaccvf\_mask\_float32xm1* (*float32xm1\_t merge*, *float32xm1\_t a*, *float b*, *float32xm1\_t c*, *e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfnmaccvf\_mask\_float32xm2* (*float32xm2\_t merge*, *float32xm2\_t a*, *float b*, *float32xm2\_t c*, *e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfnmaccvf\_mask\_float32xm4* (*float32xm4\_t merge*, *float32xm4\_t a*, *float b*, *float32xm4\_t c*, *e32xm4\_t mask*, unsigned int *gvl*)
- *float64xm1\_t vfnmaccvf\_mask\_float64xm1* (*float64xm1\_t merge*, *float64xm1\_t a*, double *b*, *float64xm1\_t c*, *e64xm1\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vfnmaccvf\_mask\_float64xm2* (*float64xm2\_t merge*, *float64xm2\_t a*, double *b*, *float64xm2\_t c*, *e64xm2\_t mask*, unsigned int *gvl*)
- *float64xm4\_t vfnmaccvf\_mask\_float64xm4* (*float64xm4\_t merge*, *float64xm4\_t a*, double *b*, *float64xm4\_t c*, *e64xm4\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = -(b[element] * c[element]) - a[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.22 Floating-point vector-vector negate multiply and add(overwrite addend)

**Instruction:** [*'vfnmacc.vv'*]**Prototypes:**

- *float16xm1\_t vfnmaccvv\_float16xm1* (*float16xm1\_t a*, *float16xm1\_t b*, *float16xm1\_t c*, unsigned int *gvl*)
- *float16xm2\_t vfnmaccvv\_float16xm2* (*float16xm2\_t a*, *float16xm2\_t b*, *float16xm2\_t c*, unsigned int *gvl*)

- `float16xm4_t vfnmaccvv_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, `float16xm4_t c`, unsigned int `gvl`)
- `float16xm8_t vfnmaccvv_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, `float16xm8_t c`, unsigned int `gvl`)
- `float32xm1_t vfnmaccvv_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, `float32xm1_t c`, unsigned int `gvl`)
- `float32xm2_t vfnmaccvv_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, `float32xm2_t c`, unsigned int `gvl`)
- `float32xm4_t vfnmaccvv_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, `float32xm4_t c`, unsigned int `gvl`)
- `float32xm8_t vfnmaccvv_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, `float32xm8_t c`, unsigned int `gvl`)
- `float64xm1_t vfnmaccvv_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, `float64xm1_t c`, unsigned int `gvl`)
- `float64xm2_t vfnmaccvv_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, `float64xm2_t c`, unsigned int `gvl`)
- `float64xm4_t vfnmaccvv_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, `float64xm4_t c`, unsigned int `gvl`)
- `float64xm8_t vfnmaccvv_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, `float64xm8_t c`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = -(b[element] * c[element]) - a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vfnmaccvv_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `float16xm1_t b`, `float16xm1_t c`, `e16xm1_t mask`, unsigned int `gvl`)
- `float16xm2_t vfnmaccvv_mask_float16xm2` (`float16xm2_t merge`, `float16xm2_t a`, `float16xm2_t b`, `float16xm2_t c`, `e16xm2_t mask`, unsigned int `gvl`)
- `float16xm4_t vfnmaccvv_mask_float16xm4` (`float16xm4_t merge`, `float16xm4_t a`, `float16xm4_t b`, `float16xm4_t c`, `e16xm4_t mask`, unsigned int `gvl`)
- `float32xm1_t vfnmaccvv_mask_float32xm1` (`float32xm1_t merge`, `float32xm1_t a`, `float32xm1_t b`, `float32xm1_t c`, `e32xm1_t mask`, unsigned int `gvl`)
- `float32xm2_t vfnmaccvv_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float32xm2_t b`, `float32xm2_t c`, `e32xm2_t mask`, unsigned int `gvl`)
- `float32xm4_t vfnmaccvv_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float32xm4_t b`, `float32xm4_t c`, `e32xm4_t mask`, unsigned int `gvl`)
- `float64xm1_t vfnmaccvv_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `float64xm1_t b`, `float64xm1_t c`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfnmaccvv_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `float64xm2_t b`, `float64xm2_t c`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfnmaccvv_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `float64xm4_t b`, `float64xm4_t c`, `e64xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = -(b[element] * c[element]) - a[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.23 Floating-point vector-scalar negate multiply and add(overwrite multiplicand)

**Instruction:** [`vfnmadd.vf`']

**Prototypes:**

- `float16xm1_t vfnmaddvf_float16xm1` (`float16xm1_t a`, `float16_t b`, `float16xm1_t c`, unsigned int `gvl`)
- `float16xm2_t vfnmaddvf_float16xm2` (`float16xm2_t a`, `float16_t b`, `float16xm2_t c`, unsigned int `gvl`)
- `float16xm4_t vfnmaddvf_float16xm4` (`float16xm4_t a`, `float16_t b`, `float16xm4_t c`, unsigned int `gvl`)
- `float16xm8_t vfnmaddvf_float16xm8` (`float16xm8_t a`, `float16_t b`, `float16xm8_t c`, unsigned int `gvl`)
- `float32xm1_t vfnmaddvf_float32xm1` (`float32xm1_t a`, `float b`, `float32xm1_t c`, unsigned int `gvl`)
- `float32xm2_t vfnmaddvf_float32xm2` (`float32xm2_t a`, `float b`, `float32xm2_t c`, unsigned int `gvl`)
- `float32xm4_t vfnmaddvf_float32xm4` (`float32xm4_t a`, `float b`, `float32xm4_t c`, unsigned int `gvl`)
- `float32xm8_t vfnmaddvf_float32xm8` (`float32xm8_t a`, `float b`, `float32xm8_t c`, unsigned int `gvl`)
- `float64xm1_t vfnmaddvf_float64xm1` (`float64xm1_t a`, `double b`, `float64xm1_t c`, unsigned int `gvl`)
- `float64xm2_t vfnmaddvf_float64xm2` (`float64xm2_t a`, `double b`, `float64xm2_t c`, unsigned int `gvl`)
- `float64xm4_t vfnmaddvf_float64xm4` (`float64xm4_t a`, `double b`, `float64xm4_t c`, unsigned int `gvl`)
- `float64xm8_t vfnmaddvf_float64xm8` (`float64xm8_t a`, `double b`, `float64xm8_t c`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = -(a[element] * b[element]) - c[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vfnmaddvf_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `float16_t b`, `float16xm1_t c`, `e16xm1_t mask`, unsigned int `gvl`)
- `float16xm2_t vfnmaddvf_mask_float16xm2` (`float16xm2_t merge`, `float16xm2_t a`, `float16_t b`, `float16xm2_t c`, `e16xm2_t mask`, unsigned int `gvl`)
- `float16xm4_t vfnmaddvf_mask_float16xm4` (`float16xm4_t merge`, `float16xm4_t a`, `float16_t b`, `float16xm4_t c`, `e16xm4_t mask`, unsigned int `gvl`)
- `float32xm1_t vfnmaddvf_mask_float32xm1` (`float32xm1_t merge`, `float32xm1_t a`, `float b`, `float32xm1_t c`, `e32xm1_t mask`, unsigned int `gvl`)
- `float32xm2_t vfnmaddvf_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float b`, `float32xm2_t c`, `e32xm2_t mask`, unsigned int `gvl`)

- `float32xm4_t vfnmaddvf_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float b`, `float32xm4_t c`, `e32xm4_t mask`, unsigned int gvl)
- `float64xm1_t vfnmaddvf_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `double b`, `float64xm1_t c`, `e64xm1_t mask`, unsigned int gvl)
- `float64xm2_t vfnmaddvf_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `double b`, `float64xm2_t c`, `e64xm2_t mask`, unsigned int gvl)
- `float64xm4_t vfnmaddvf_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `double b`, `float64xm4_t c`, `e64xm4_t mask`, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = -(a[element] * b[element]) - c[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.24 Floating-point vector-vector negate multiply and add(overwrite multiplicand)

Instruction: ['vfnmadd.vv']

Prototypes:

- `float16xm1_t vfnmaddvv_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, `float16xm1_t c`, unsigned int gvl)
- `float16xm2_t vfnmaddvv_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, `float16xm2_t c`, unsigned int gvl)
- `float16xm4_t vfnmaddvv_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, `float16xm4_t c`, unsigned int gvl)
- `float16xm8_t vfnmaddvv_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, `float16xm8_t c`, unsigned int gvl)
- `float32xm1_t vfnmaddvv_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, `float32xm1_t c`, unsigned int gvl)
- `float32xm2_t vfnmaddvv_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, `float32xm2_t c`, unsigned int gvl)
- `float32xm4_t vfnmaddvv_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, `float32xm4_t c`, unsigned int gvl)
- `float32xm8_t vfnmaddvv_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, `float32xm8_t c`, unsigned int gvl)
- `float64xm1_t vfnmaddvv_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, `float64xm1_t c`, unsigned int gvl)
- `float64xm2_t vfnmaddvv_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, `float64xm2_t c`, unsigned int gvl)
- `float64xm4_t vfnmaddvv_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, `float64xm4_t c`, unsigned int gvl)
- `float64xm8_t vfnmaddvv_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, `float64xm8_t c`, unsigned int gvl)

Operation:



```
>>> for element = 0 to gvl - 1
      result[element] = -(a[element] * b[element]) - c[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfnmaddvv\_mask\_float16xm1* (*float16xm1\_t merge, float16xm1\_t a, float16xm1\_t b, float16xm1\_t c, e16xm1\_t mask*, unsigned int gvl)
- *float16xm2\_t vfnmaddvv\_mask\_float16xm2* (*float16xm2\_t merge, float16xm2\_t a, float16xm2\_t b, float16xm2\_t c, e16xm2\_t mask*, unsigned int gvl)
- *float16xm4\_t vfnmaddvv\_mask\_float16xm4* (*float16xm4\_t merge, float16xm4\_t a, float16xm4\_t b, float16xm4\_t c, e16xm4\_t mask*, unsigned int gvl)
- *float32xm1\_t vfnmaddvv\_mask\_float32xm1* (*float32xm1\_t merge, float32xm1\_t a, float32xm1\_t b, float32xm1\_t c, e32xm1\_t mask*, unsigned int gvl)
- *float32xm2\_t vfnmaddvv\_mask\_float32xm2* (*float32xm2\_t merge, float32xm2\_t a, float32xm2\_t b, float32xm2\_t c, e32xm2\_t mask*, unsigned int gvl)
- *float32xm4\_t vfnmaddvv\_mask\_float32xm4* (*float32xm4\_t merge, float32xm4\_t a, float32xm4\_t b, float32xm4\_t c, e32xm4\_t mask*, unsigned int gvl)
- *float64xm1\_t vfnmaddvv\_mask\_float64xm1* (*float64xm1\_t merge, float64xm1\_t a, float64xm1\_t b, float64xm1\_t c, e64xm1\_t mask*, unsigned int gvl)
- *float64xm2\_t vfnmaddvv\_mask\_float64xm2* (*float64xm2\_t merge, float64xm2\_t a, float64xm2\_t b, float64xm2\_t c, e64xm2\_t mask*, unsigned int gvl)
- *float64xm4\_t vfnmaddvv\_mask\_float64xm4* (*float64xm4\_t merge, float64xm4\_t a, float64xm4\_t b, float64xm4\_t c, e64xm4\_t mask*, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = -(a[element] * b[element]) - c[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.25 Floating-point vector-scalar negate multiply and sub(overwrite subtrahend)****Instruction:** [*vfnmsac.vf*]**Prototypes:**

- *float16xm1\_t vfnmsacvf\_float16xm1* (*float16xm1\_t a, float16\_t b, float16xm1\_t c*, unsigned int gvl)
- *float16xm2\_t vfnmsacvf\_float16xm2* (*float16xm2\_t a, float16\_t b, float16xm2\_t c*, unsigned int gvl)
- *float16xm4\_t vfnmsacvf\_float16xm4* (*float16xm4\_t a, float16\_t b, float16xm4\_t c*, unsigned int gvl)
- *float16xm8\_t vfnmsacvf\_float16xm8* (*float16xm8\_t a, float16\_t b, float16xm8\_t c*, unsigned int gvl)
- *float32xm1\_t vfnmsacvf\_float32xm1* (*float32xm1\_t a, float b, float32xm1\_t c*, unsigned int gvl)
- *float32xm2\_t vfnmsacvf\_float32xm2* (*float32xm2\_t a, float b, float32xm2\_t c*, unsigned int gvl)

- *float32xm4\_t vfnmsacvf\_float32xm4* (*float32xm4\_t a*, *float b*, *float32xm4\_t c*, unsigned int *gvl*)
- *float32xm8\_t vfnmsacvf\_float32xm8* (*float32xm8\_t a*, *float b*, *float32xm8\_t c*, unsigned int *gvl*)
- *float64xm1\_t vfnmsacvf\_float64xm1* (*float64xm1\_t a*, *double b*, *float64xm1\_t c*, unsigned int *gvl*)
- *float64xm2\_t vfnmsacvf\_float64xm2* (*float64xm2\_t a*, *double b*, *float64xm2\_t c*, unsigned int *gvl*)
- *float64xm4\_t vfnmsacvf\_float64xm4* (*float64xm4\_t a*, *double b*, *float64xm4\_t c*, unsigned int *gvl*)
- *float64xm8\_t vfnmsacvf\_float64xm8* (*float64xm8\_t a*, *double b*, *float64xm8\_t c*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = -(b[element] * c[element]) + a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfnmsacvf\_mask\_float16xm1* (*float16xm1\_t merge*, *float16xm1\_t a*, *float16\_t b*, *float16xm1\_t c*, *e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfnmsacvf\_mask\_float16xm2* (*float16xm2\_t merge*, *float16xm2\_t a*, *float16\_t b*, *float16xm2\_t c*, *e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfnmsacvf\_mask\_float16xm4* (*float16xm4\_t merge*, *float16xm4\_t a*, *float16\_t b*, *float16xm4\_t c*, *e16xm4\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfnmsacvf\_mask\_float32xm1* (*float32xm1\_t merge*, *float32xm1\_t a*, *float b*, *float32xm1\_t c*, *e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfnmsacvf\_mask\_float32xm2* (*float32xm2\_t merge*, *float32xm2\_t a*, *float b*, *float32xm2\_t c*, *e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfnmsacvf\_mask\_float32xm4* (*float32xm4\_t merge*, *float32xm4\_t a*, *float b*, *float32xm4\_t c*, *e32xm4\_t mask*, unsigned int *gvl*)
- *float64xm1\_t vfnmsacvf\_mask\_float64xm1* (*float64xm1\_t merge*, *float64xm1\_t a*, *double b*, *float64xm1\_t c*, *e64xm1\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vfnmsacvf\_mask\_float64xm2* (*float64xm2\_t merge*, *float64xm2\_t a*, *double b*, *float64xm2\_t c*, *e64xm2\_t mask*, unsigned int *gvl*)
- *float64xm4\_t vfnmsacvf\_mask\_float64xm4* (*float64xm4\_t merge*, *float64xm4\_t a*, *double b*, *float64xm4\_t c*, *e64xm4\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = -(b[element] * c[element]) + a[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.26 Floating-point vector-vector negate multiply and sub(overwrite subtrahend)****Instruction:** [*'vfnmsac.vv'*]**Prototypes:**

- *float16xm1\_t vfnmsacvv\_float16xm1* (*float16xm1\_t a*, *float16xm1\_t b*, *float16xm1\_t c*, unsigned int *gvl*)

- `float16xm2_t vfnmsacvv_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, `float16xm2_t c`, unsigned int `gvl`)
- `float16xm4_t vfnmsacvv_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, `float16xm4_t c`, unsigned int `gvl`)
- `float16xm8_t vfnmsacvv_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, `float16xm8_t c`, unsigned int `gvl`)
- `float32xm1_t vfnmsacvv_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, `float32xm1_t c`, unsigned int `gvl`)
- `float32xm2_t vfnmsacvv_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, `float32xm2_t c`, unsigned int `gvl`)
- `float32xm4_t vfnmsacvv_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, `float32xm4_t c`, unsigned int `gvl`)
- `float32xm8_t vfnmsacvv_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, `float32xm8_t c`, unsigned int `gvl`)
- `float64xm1_t vfnmsacvv_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, `float64xm1_t c`, unsigned int `gvl`)
- `float64xm2_t vfnmsacvv_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, `float64xm2_t c`, unsigned int `gvl`)
- `float64xm4_t vfnmsacvv_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, `float64xm4_t c`, unsigned int `gvl`)
- `float64xm8_t vfnmsacvv_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, `float64xm8_t c`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = -(b[element] * c[element]) + a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vfnmsacvv_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `float16xm1_t b`, `float16xm1_t c`, `e16xm1_t mask`, unsigned int `gvl`)
- `float16xm2_t vfnmsacvv_mask_float16xm2` (`float16xm2_t merge`, `float16xm2_t a`, `float16xm2_t b`, `float16xm2_t c`, `e16xm2_t mask`, unsigned int `gvl`)
- `float16xm4_t vfnmsacvv_mask_float16xm4` (`float16xm4_t merge`, `float16xm4_t a`, `float16xm4_t b`, `float16xm4_t c`, `e16xm4_t mask`, unsigned int `gvl`)
- `float32xm1_t vfnmsacvv_mask_float32xm1` (`float32xm1_t merge`, `float32xm1_t a`, `float32xm1_t b`, `float32xm1_t c`, `e32xm1_t mask`, unsigned int `gvl`)
- `float32xm2_t vfnmsacvv_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float32xm2_t b`, `float32xm2_t c`, `e32xm2_t mask`, unsigned int `gvl`)
- `float32xm4_t vfnmsacvv_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float32xm4_t b`, `float32xm4_t c`, `e32xm4_t mask`, unsigned int `gvl`)
- `float64xm1_t vfnmsacvv_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `float64xm1_t b`, `float64xm1_t c`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfnmsacvv_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `float64xm2_t b`, `float64xm2_t c`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfnmsacvv_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `float64xm4_t b`, `float64xm4_t c`, `e64xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = -(b[element] * c[element]) + a[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.27 Floating-point vector-scalar negate multiply and sub(overwrite multiplicand)****Instruction:** [`'vfnmsub.vf'`]**Prototypes:**

- `float16xm1_t vfnmsubvf_float16xm1` (`float16xm1_t a`, `float16_t b`, `float16xm1_t c`, unsigned int `gvl`)
- `float16xm2_t vfnmsubvf_float16xm2` (`float16xm2_t a`, `float16_t b`, `float16xm2_t c`, unsigned int `gvl`)
- `float16xm4_t vfnmsubvf_float16xm4` (`float16xm4_t a`, `float16_t b`, `float16xm4_t c`, unsigned int `gvl`)
- `float16xm8_t vfnmsubvf_float16xm8` (`float16xm8_t a`, `float16_t b`, `float16xm8_t c`, unsigned int `gvl`)
- `float32xm1_t vfnmsubvf_float32xm1` (`float32xm1_t a`, `float b`, `float32xm1_t c`, unsigned int `gvl`)
- `float32xm2_t vfnmsubvf_float32xm2` (`float32xm2_t a`, `float b`, `float32xm2_t c`, unsigned int `gvl`)
- `float32xm4_t vfnmsubvf_float32xm4` (`float32xm4_t a`, `float b`, `float32xm4_t c`, unsigned int `gvl`)
- `float32xm8_t vfnmsubvf_float32xm8` (`float32xm8_t a`, `float b`, `float32xm8_t c`, unsigned int `gvl`)
- `float64xm1_t vfnmsubvf_float64xm1` (`float64xm1_t a`, `double b`, `float64xm1_t c`, unsigned int `gvl`)
- `float64xm2_t vfnmsubvf_float64xm2` (`float64xm2_t a`, `double b`, `float64xm2_t c`, unsigned int `gvl`)
- `float64xm4_t vfnmsubvf_float64xm4` (`float64xm4_t a`, `double b`, `float64xm4_t c`, unsigned int `gvl`)
- `float64xm8_t vfnmsubvf_float64xm8` (`float64xm8_t a`, `double b`, `float64xm8_t c`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = -(a[element] * b[element]) + c[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vfnmsubvf_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `float16_t b`, `float16xm1_t c`, `e16xm1_t mask`, unsigned int `gvl`)
- `float16xm2_t vfnmsubvf_mask_float16xm2` (`float16xm2_t merge`, `float16xm2_t a`, `float16_t b`, `float16xm2_t c`, `e16xm2_t mask`, unsigned int `gvl`)
- `float16xm4_t vfnmsubvf_mask_float16xm4` (`float16xm4_t merge`, `float16xm4_t a`, `float16_t b`, `float16xm4_t c`, `e16xm4_t mask`, unsigned int `gvl`)
- `float32xm1_t vfnmsubvf_mask_float32xm1` (`float32xm1_t merge`, `float32xm1_t a`, `float b`, `float32xm1_t c`, `e32xm1_t mask`, unsigned int `gvl`)
- `float32xm2_t vfnmsubvf_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float b`, `float32xm2_t c`, `e32xm2_t mask`, unsigned int `gvl`)

- `float32xm4_t vfnmsubvf_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float b`, `float32xm4_t c`, `e32xm4_t mask`, unsigned int gvl)
- `float64xm1_t vfnmsubvf_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `double b`, `float64xm1_t c`, `e64xm1_t mask`, unsigned int gvl)
- `float64xm2_t vfnmsubvf_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `double b`, `float64xm2_t c`, `e64xm2_t mask`, unsigned int gvl)
- `float64xm4_t vfnmsubvf_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `double b`, `float64xm4_t c`, `e64xm4_t mask`, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = -(a[element] * b[element]) + c[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.28 Floating-point vector-vector negate multiply and sub(overwrite multiplicand)

Instruction: ['vfnmsub.vv']

Prototypes:

- `float16xm1_t vfnmsubvv_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, `float16xm1_t c`, unsigned int gvl)
- `float16xm2_t vfnmsubvv_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, `float16xm2_t c`, unsigned int gvl)
- `float16xm4_t vfnmsubvv_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, `float16xm4_t c`, unsigned int gvl)
- `float16xm8_t vfnmsubvv_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, `float16xm8_t c`, unsigned int gvl)
- `float32xm1_t vfnmsubvv_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, `float32xm1_t c`, unsigned int gvl)
- `float32xm2_t vfnmsubvv_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, `float32xm2_t c`, unsigned int gvl)
- `float32xm4_t vfnmsubvv_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, `float32xm4_t c`, unsigned int gvl)
- `float32xm8_t vfnmsubvv_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, `float32xm8_t c`, unsigned int gvl)
- `float64xm1_t vfnmsubvv_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, `float64xm1_t c`, unsigned int gvl)
- `float64xm2_t vfnmsubvv_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, `float64xm2_t c`, unsigned int gvl)
- `float64xm4_t vfnmsubvv_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, `float64xm4_t c`, unsigned int gvl)
- `float64xm8_t vfnmsubvv_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, `float64xm8_t c`, unsigned int gvl)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = -(a[element] * b[element]) + c[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfnmsubvv\_mask\_float16xm1* (*float16xm1\_t merge, float16xm1\_t a, float16xm1\_t b, float16xm1\_t c, e16xm1\_t mask*, unsigned int gvl)
- *float16xm2\_t vfnmsubvv\_mask\_float16xm2* (*float16xm2\_t merge, float16xm2\_t a, float16xm2\_t b, float16xm2\_t c, e16xm2\_t mask*, unsigned int gvl)
- *float16xm4\_t vfnmsubvv\_mask\_float16xm4* (*float16xm4\_t merge, float16xm4\_t a, float16xm4\_t b, float16xm4\_t c, e16xm4\_t mask*, unsigned int gvl)
- *float32xm1\_t vfnmsubvv\_mask\_float32xm1* (*float32xm1\_t merge, float32xm1\_t a, float32xm1\_t b, float32xm1\_t c, e32xm1\_t mask*, unsigned int gvl)
- *float32xm2\_t vfnmsubvv\_mask\_float32xm2* (*float32xm2\_t merge, float32xm2\_t a, float32xm2\_t b, float32xm2\_t c, e32xm2\_t mask*, unsigned int gvl)
- *float32xm4\_t vfnmsubvv\_mask\_float32xm4* (*float32xm4\_t merge, float32xm4\_t a, float32xm4\_t b, float32xm4\_t c, e32xm4\_t mask*, unsigned int gvl)
- *float64xm1\_t vfnmsubvv\_mask\_float64xm1* (*float64xm1\_t merge, float64xm1\_t a, float64xm1\_t b, float64xm1\_t c, e64xm1\_t mask*, unsigned int gvl)
- *float64xm2\_t vfnmsubvv\_mask\_float64xm2* (*float64xm2\_t merge, float64xm2\_t a, float64xm2\_t b, float64xm2\_t c, e64xm2\_t mask*, unsigned int gvl)
- *float64xm4\_t vfnmsubvv\_mask\_float64xm4* (*float64xm4\_t merge, float64xm4\_t a, float64xm4\_t b, float64xm4\_t c, e64xm4\_t mask*, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = -(a[element] * b[element]) + c[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.29 Elementwise vector-scalar floating-point reverse division

**Instruction:** ['vfrdiv.vf']**Prototypes:**

- *float16xm1\_t vfrdivvf\_float16xm1* (*float16xm1\_t a, float16\_t b*, unsigned int gvl)
- *float16xm2\_t vfrdivvf\_float16xm2* (*float16xm2\_t a, float16\_t b*, unsigned int gvl)
- *float16xm4\_t vfrdivvf\_float16xm4* (*float16xm4\_t a, float16\_t b*, unsigned int gvl)
- *float16xm8\_t vfrdivvf\_float16xm8* (*float16xm8\_t a, float16\_t b*, unsigned int gvl)
- *float32xm1\_t vfrdivvf\_float32xm1* (*float32xm1\_t a, float b*, unsigned int gvl)
- *float32xm2\_t vfrdivvf\_float32xm2* (*float32xm2\_t a, float b*, unsigned int gvl)
- *float32xm4\_t vfrdivvf\_float32xm4* (*float32xm4\_t a, float b*, unsigned int gvl)
- *float32xm8\_t vfrdivvf\_float32xm8* (*float32xm8\_t a, float b*, unsigned int gvl)



- `float64xm1_t vfrdivvf_float64xm1` (`float64xm1_t a`, double `b`, unsigned int `gvl`)
- `float64xm2_t vfrdivvf_float64xm2` (`float64xm2_t a`, double `b`, unsigned int `gvl`)
- `float64xm4_t vfrdivvf_float64xm4` (`float64xm4_t a`, double `b`, unsigned int `gvl`)
- `float64xm8_t vfrdivvf_float64xm8` (`float64xm8_t a`, double `b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = b / a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vfrdivvf_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `float16_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `float16xm2_t vfrdivvf_mask_float16xm2` (`float16xm2_t merge`, `float16xm2_t a`, `float16_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `float16xm4_t vfrdivvf_mask_float16xm4` (`float16xm4_t merge`, `float16xm4_t a`, `float16_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `float16xm8_t vfrdivvf_mask_float16xm8` (`float16xm8_t merge`, `float16xm8_t a`, `float16_t b`, `e16xm8_t mask`, unsigned int `gvl`)
- `float32xm1_t vfrdivvf_mask_float32xm1` (`float32xm1_t merge`, `float32xm1_t a`, `float b`, `e32xm1_t mask`, unsigned int `gvl`)
- `float32xm2_t vfrdivvf_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float b`, `e32xm2_t mask`, unsigned int `gvl`)
- `float32xm4_t vfrdivvf_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float b`, `e32xm4_t mask`, unsigned int `gvl`)
- `float32xm8_t vfrdivvf_mask_float32xm8` (`float32xm8_t merge`, `float32xm8_t a`, `float b`, `e32xm8_t mask`, unsigned int `gvl`)
- `float64xm1_t vfrdivvf_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, double `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfrdivvf_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, double `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfrdivvf_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, double `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vfrdivvf_mask_float64xm8` (`float64xm8_t merge`, `float64xm8_t a`, double `b`, `e64xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = b / a[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.30 Floating-point maximum of vector

**Instruction:** [`'vfredmax.vs'`]

**Prototypes:**

- *float16xm1\_t vfredmaxvs\_float16xm1* (*float16xm1\_t a, float16xm1\_t b*, unsigned int *gvl*)
- *float16xm2\_t vfredmaxvs\_float16xm2* (*float16xm2\_t a, float16xm2\_t b*, unsigned int *gvl*)
- *float16xm4\_t vfredmaxvs\_float16xm4* (*float16xm4\_t a, float16xm4\_t b*, unsigned int *gvl*)
- *float16xm8\_t vfredmaxvs\_float16xm8* (*float16xm8\_t a, float16xm8\_t b*, unsigned int *gvl*)
- *float32xm1\_t vfredmaxvs\_float32xm1* (*float32xm1\_t a, float32xm1\_t b*, unsigned int *gvl*)
- *float32xm2\_t vfredmaxvs\_float32xm2* (*float32xm2\_t a, float32xm2\_t b*, unsigned int *gvl*)
- *float32xm4\_t vfredmaxvs\_float32xm4* (*float32xm4\_t a, float32xm4\_t b*, unsigned int *gvl*)
- *float32xm8\_t vfredmaxvs\_float32xm8* (*float32xm8\_t a, float32xm8\_t b*, unsigned int *gvl*)
- *float64xm1\_t vfredmaxvs\_float64xm1* (*float64xm1\_t a, float64xm1\_t b*, unsigned int *gvl*)
- *float64xm2\_t vfredmaxvs\_float64xm2* (*float64xm2\_t a, float64xm2\_t b*, unsigned int *gvl*)
- *float64xm4\_t vfredmaxvs\_float64xm4* (*float64xm4\_t a, float64xm4\_t b*, unsigned int *gvl*)
- *float64xm8\_t vfredmaxvs\_float64xm8* (*float64xm8\_t a, float64xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> result[0] = b[0]
    for element = 0 to gvl - 1
        result[0] = max(result[0], a[element])
```

**Masked prototypes:**

- *float16xm1\_t vfredmaxvs\_mask\_float16xm1* (*float16xm1\_t a, float16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfredmaxvs\_mask\_float16xm2* (*float16xm2\_t a, float16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfredmaxvs\_mask\_float16xm4* (*float16xm4\_t a, float16xm4\_t b, e16xm4\_t mask*, unsigned int *gvl*)
- *float16xm8\_t vfredmaxvs\_mask\_float16xm8* (*float16xm8\_t a, float16xm8\_t b, e16xm8\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfredmaxvs\_mask\_float32xm1* (*float32xm1\_t a, float32xm1\_t b, e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfredmaxvs\_mask\_float32xm2* (*float32xm2\_t a, float32xm2\_t b, e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfredmaxvs\_mask\_float32xm4* (*float32xm4\_t a, float32xm4\_t b, e32xm4\_t mask*, unsigned int *gvl*)
- *float32xm8\_t vfredmaxvs\_mask\_float32xm8* (*float32xm8\_t a, float32xm8\_t b, e32xm8\_t mask*, unsigned int *gvl*)
- *float64xm1\_t vfredmaxvs\_mask\_float64xm1* (*float64xm1\_t a, float64xm1\_t b, e64xm1\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vfredmaxvs\_mask\_float64xm2* (*float64xm2\_t a, float64xm2\_t b, e64xm2\_t mask*, unsigned int *gvl*)
- *float64xm4\_t vfredmaxvs\_mask\_float64xm4* (*float64xm4\_t a, float64xm4\_t b, e64xm4\_t mask*, unsigned int *gvl*)
- *float64xm8\_t vfredmaxvs\_mask\_float64xm8* (*float64xm8\_t a, float64xm8\_t b, e64xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> result[0] = b[0]
    for element = 0 to gvl - 1
        if mask[element]
            result[0] = max(result[0], a[element])
```

**2.5.31 Floating-point minimum of vector****Instruction:** [`'vfredmin.vs'`]**Prototypes:**

- `float16xm1_t vfredminvs_float16xm1` (`float16xm1_t a, float16xm1_t b, unsigned int gvl`)
- `float16xm2_t vfredminvs_float16xm2` (`float16xm2_t a, float16xm2_t b, unsigned int gvl`)
- `float16xm4_t vfredminvs_float16xm4` (`float16xm4_t a, float16xm4_t b, unsigned int gvl`)
- `float16xm8_t vfredminvs_float16xm8` (`float16xm8_t a, float16xm8_t b, unsigned int gvl`)
- `float32xm1_t vfredminvs_float32xm1` (`float32xm1_t a, float32xm1_t b, unsigned int gvl`)
- `float32xm2_t vfredminvs_float32xm2` (`float32xm2_t a, float32xm2_t b, unsigned int gvl`)
- `float32xm4_t vfredminvs_float32xm4` (`float32xm4_t a, float32xm4_t b, unsigned int gvl`)
- `float32xm8_t vfredminvs_float32xm8` (`float32xm8_t a, float32xm8_t b, unsigned int gvl`)
- `float64xm1_t vfredminvs_float64xm1` (`float64xm1_t a, float64xm1_t b, unsigned int gvl`)
- `float64xm2_t vfredminvs_float64xm2` (`float64xm2_t a, float64xm2_t b, unsigned int gvl`)
- `float64xm4_t vfredminvs_float64xm4` (`float64xm4_t a, float64xm4_t b, unsigned int gvl`)
- `float64xm8_t vfredminvs_float64xm8` (`float64xm8_t a, float64xm8_t b, unsigned int gvl`)

**Operation:**

```
>>> result[0] = b[0]
    for element = 0 to gvl - 1
        result[0] = min(result[0], a[element])
```

**Masked prototypes:**

- `float16xm1_t vfredminvs_mask_float16xm1` (`float16xm1_t a, float16xm1_t b, e16xm1_t mask, unsigned int gvl`)
- `float16xm2_t vfredminvs_mask_float16xm2` (`float16xm2_t a, float16xm2_t b, e16xm2_t mask, unsigned int gvl`)
- `float16xm4_t vfredminvs_mask_float16xm4` (`float16xm4_t a, float16xm4_t b, e16xm4_t mask, unsigned int gvl`)
- `float16xm8_t vfredminvs_mask_float16xm8` (`float16xm8_t a, float16xm8_t b, e16xm8_t mask, unsigned int gvl`)
- `float32xm1_t vfredminvs_mask_float32xm1` (`float32xm1_t a, float32xm1_t b, e32xm1_t mask, unsigned int gvl`)
- `float32xm2_t vfredminvs_mask_float32xm2` (`float32xm2_t a, float32xm2_t b, e32xm2_t mask, unsigned int gvl`)
- `float32xm4_t vfredminvs_mask_float32xm4` (`float32xm4_t a, float32xm4_t b, e32xm4_t mask, unsigned int gvl`)

- `float32xm8_t vfredminvs_mask_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, `e32xm8_t mask`, unsigned int `gvl`)
- `float64xm1_t vfredminvs_mask_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfredminvs_mask_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfredminvs_mask_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vfredminvs_mask_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, `e64xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> result[0] = b[0]
    for element = 0 to gvl - 1
        if mask[element]
            result[0] = min(result[0], a[element])
```

## 2.5.32 Floating-point odered sum of vector

Instruction: ['vfredosum.vs']

Prototypes:

- `float16xm1_t vfredosumvs_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, unsigned int `gvl`)
- `float16xm2_t vfredosumvs_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, unsigned int `gvl`)
- `float16xm4_t vfredosumvs_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, unsigned int `gvl`)
- `float16xm8_t vfredosumvs_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, unsigned int `gvl`)
- `float32xm1_t vfredosumvs_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, unsigned int `gvl`)
- `float32xm2_t vfredosumvs_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, unsigned int `gvl`)
- `float32xm4_t vfredosumvs_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, unsigned int `gvl`)
- `float32xm8_t vfredosumvs_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, unsigned int `gvl`)
- `float64xm1_t vfredosumvs_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, unsigned int `gvl`)
- `float64xm2_t vfredosumvs_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, unsigned int `gvl`)
- `float64xm4_t vfredosumvs_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, unsigned int `gvl`)
- `float64xm8_t vfredosumvs_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, unsigned int `gvl`)

Operation:

```
>>> result[0] = b[0]
    for element = 0 to gvl - 1
        result[0] = result[0] + a[element]
```

Masked prototypes:

- `float16xm1_t vfredosumvs_mask_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `float16xm2_t vfredosumvs_mask_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, `e16xm2_t mask`, unsigned int `gvl`)

- *float16xm4\_t* **vfredosumvs\_mask\_float16xm4** (*float16xm4\_t* a, *float16xm4\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *float16xm8\_t* **vfredosumvs\_mask\_float16xm8** (*float16xm8\_t* a, *float16xm8\_t* b, *e16xm8\_t* mask, unsigned int gvl)
- *float32xm1\_t* **vfredosumvs\_mask\_float32xm1** (*float32xm1\_t* a, *float32xm1\_t* b, *e32xm1\_t* mask, unsigned int gvl)
- *float32xm2\_t* **vfredosumvs\_mask\_float32xm2** (*float32xm2\_t* a, *float32xm2\_t* b, *e32xm2\_t* mask, unsigned int gvl)
- *float32xm4\_t* **vfredosumvs\_mask\_float32xm4** (*float32xm4\_t* a, *float32xm4\_t* b, *e32xm4\_t* mask, unsigned int gvl)
- *float32xm8\_t* **vfredosumvs\_mask\_float32xm8** (*float32xm8\_t* a, *float32xm8\_t* b, *e32xm8\_t* mask, unsigned int gvl)
- *float64xm1\_t* **vfredosumvs\_mask\_float64xm1** (*float64xm1\_t* a, *float64xm1\_t* b, *e64xm1\_t* mask, unsigned int gvl)
- *float64xm2\_t* **vfredosumvs\_mask\_float64xm2** (*float64xm2\_t* a, *float64xm2\_t* b, *e64xm2\_t* mask, unsigned int gvl)
- *float64xm4\_t* **vfredosumvs\_mask\_float64xm4** (*float64xm4\_t* a, *float64xm4\_t* b, *e64xm4\_t* mask, unsigned int gvl)
- *float64xm8\_t* **vfredosumvs\_mask\_float64xm8** (*float64xm8\_t* a, *float64xm8\_t* b, *e64xm8\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> result[0] = b[0]
    for element = 0 to gvl - 1
        if mask[element]
            result[0] = result[0] + a[element]
```

## 2.5.33 Floating-point sum of vector

**Instruction:** ['vfredsum.vs']

**Prototypes:**

- *float16xm1\_t* **vfredsumvs\_float16xm1** (*float16xm1\_t* a, *float16xm1\_t* b, unsigned int gvl)
- *float16xm2\_t* **vfredsumvs\_float16xm2** (*float16xm2\_t* a, *float16xm2\_t* b, unsigned int gvl)
- *float16xm4\_t* **vfredsumvs\_float16xm4** (*float16xm4\_t* a, *float16xm4\_t* b, unsigned int gvl)
- *float16xm8\_t* **vfredsumvs\_float16xm8** (*float16xm8\_t* a, *float16xm8\_t* b, unsigned int gvl)
- *float32xm1\_t* **vfredsumvs\_float32xm1** (*float32xm1\_t* a, *float32xm1\_t* b, unsigned int gvl)
- *float32xm2\_t* **vfredsumvs\_float32xm2** (*float32xm2\_t* a, *float32xm2\_t* b, unsigned int gvl)
- *float32xm4\_t* **vfredsumvs\_float32xm4** (*float32xm4\_t* a, *float32xm4\_t* b, unsigned int gvl)
- *float32xm8\_t* **vfredsumvs\_float32xm8** (*float32xm8\_t* a, *float32xm8\_t* b, unsigned int gvl)
- *float64xm1\_t* **vfredsumvs\_float64xm1** (*float64xm1\_t* a, *float64xm1\_t* b, unsigned int gvl)
- *float64xm2\_t* **vfredsumvs\_float64xm2** (*float64xm2\_t* a, *float64xm2\_t* b, unsigned int gvl)
- *float64xm4\_t* **vfredsumvs\_float64xm4** (*float64xm4\_t* a, *float64xm4\_t* b, unsigned int gvl)
- *float64xm8\_t* **vfredsumvs\_float64xm8** (*float64xm8\_t* a, *float64xm8\_t* b, unsigned int gvl)

**Operation:**

```
>>> result[0] = b[0]
    for element = 0 to gvl - 1
        result[0] = result[0] + a[element]
```

**Masked prototypes:**

- *float16xm1\_t* **vfiredsumvs\_mask\_float16xm1** (*float16xm1\_t* a, *float16xm1\_t* b, *e16xm1\_t* mask, unsigned int gvl)
- *float16xm2\_t* **vfiredsumvs\_mask\_float16xm2** (*float16xm2\_t* a, *float16xm2\_t* b, *e16xm2\_t* mask, unsigned int gvl)
- *float16xm4\_t* **vfiredsumvs\_mask\_float16xm4** (*float16xm4\_t* a, *float16xm4\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *float16xm8\_t* **vfiredsumvs\_mask\_float16xm8** (*float16xm8\_t* a, *float16xm8\_t* b, *e16xm8\_t* mask, unsigned int gvl)
- *float32xm1\_t* **vfiredsumvs\_mask\_float32xm1** (*float32xm1\_t* a, *float32xm1\_t* b, *e32xm1\_t* mask, unsigned int gvl)
- *float32xm2\_t* **vfiredsumvs\_mask\_float32xm2** (*float32xm2\_t* a, *float32xm2\_t* b, *e32xm2\_t* mask, unsigned int gvl)
- *float32xm4\_t* **vfiredsumvs\_mask\_float32xm4** (*float32xm4\_t* a, *float32xm4\_t* b, *e32xm4\_t* mask, unsigned int gvl)
- *float32xm8\_t* **vfiredsumvs\_mask\_float32xm8** (*float32xm8\_t* a, *float32xm8\_t* b, *e32xm8\_t* mask, unsigned int gvl)
- *float64xm1\_t* **vfiredsumvs\_mask\_float64xm1** (*float64xm1\_t* a, *float64xm1\_t* b, *e64xm1\_t* mask, unsigned int gvl)
- *float64xm2\_t* **vfiredsumvs\_mask\_float64xm2** (*float64xm2\_t* a, *float64xm2\_t* b, *e64xm2\_t* mask, unsigned int gvl)
- *float64xm4\_t* **vfiredsumvs\_mask\_float64xm4** (*float64xm4\_t* a, *float64xm4\_t* b, *e64xm4\_t* mask, unsigned int gvl)
- *float64xm8\_t* **vfiredsumvs\_mask\_float64xm8** (*float64xm8\_t* a, *float64xm8\_t* b, *e64xm8\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> result[0] = b[0]
    for element = 0 to gvl - 1
        if mask[element]
            result[0] = result[0] + a[element]
```

## 2.5.34 Elementwise vector-scalar floating-point reverse subtraction

**Instruction:** ['vfrsub.vf']**Prototypes:**

- *float16xm1\_t* **vfrsubvf\_float16xm1** (*float16xm1\_t* a, *float16\_t* b, unsigned int gvl)
- *float16xm2\_t* **vfrsubvf\_float16xm2** (*float16xm2\_t* a, *float16\_t* b, unsigned int gvl)
- *float16xm4\_t* **vfrsubvf\_float16xm4** (*float16xm4\_t* a, *float16\_t* b, unsigned int gvl)
- *float16xm8\_t* **vfrsubvf\_float16xm8** (*float16xm8\_t* a, *float16\_t* b, unsigned int gvl)



- *float32xm1\_t* **vfrsubvf\_float32xm1** (*float32xm1\_t* *a*, float *b*, unsigned int *gvl*)
- *float32xm2\_t* **vfrsubvf\_float32xm2** (*float32xm2\_t* *a*, float *b*, unsigned int *gvl*)
- *float32xm4\_t* **vfrsubvf\_float32xm4** (*float32xm4\_t* *a*, float *b*, unsigned int *gvl*)
- *float32xm8\_t* **vfrsubvf\_float32xm8** (*float32xm8\_t* *a*, float *b*, unsigned int *gvl*)
- *float64xm1\_t* **vfrsubvf\_float64xm1** (*float64xm1\_t* *a*, double *b*, unsigned int *gvl*)
- *float64xm2\_t* **vfrsubvf\_float64xm2** (*float64xm2\_t* *a*, double *b*, unsigned int *gvl*)
- *float64xm4\_t* **vfrsubvf\_float64xm4** (*float64xm4\_t* *a*, double *b*, unsigned int *gvl*)
- *float64xm8\_t* **vfrsubvf\_float64xm8** (*float64xm8\_t* *a*, double *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = b - a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t* **vfrsubvf\_mask\_float16xm1** (*float16xm1\_t* *merge*, *float16xm1\_t* *a*, float16\_t *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *float16xm2\_t* **vfrsubvf\_mask\_float16xm2** (*float16xm2\_t* *merge*, *float16xm2\_t* *a*, float16\_t *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *float16xm4\_t* **vfrsubvf\_mask\_float16xm4** (*float16xm4\_t* *merge*, *float16xm4\_t* *a*, float16\_t *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *float16xm8\_t* **vfrsubvf\_mask\_float16xm8** (*float16xm8\_t* *merge*, *float16xm8\_t* *a*, float16\_t *b*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- *float32xm1\_t* **vfrsubvf\_mask\_float32xm1** (*float32xm1\_t* *merge*, *float32xm1\_t* *a*, float *b*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *float32xm2\_t* **vfrsubvf\_mask\_float32xm2** (*float32xm2\_t* *merge*, *float32xm2\_t* *a*, float *b*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *float32xm4\_t* **vfrsubvf\_mask\_float32xm4** (*float32xm4\_t* *merge*, *float32xm4\_t* *a*, float *b*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *float32xm8\_t* **vfrsubvf\_mask\_float32xm8** (*float32xm8\_t* *merge*, *float32xm8\_t* *a*, float *b*, *e32xm8\_t* *mask*, unsigned int *gvl*)
- *float64xm1\_t* **vfrsubvf\_mask\_float64xm1** (*float64xm1\_t* *merge*, *float64xm1\_t* *a*, double *b*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *float64xm2\_t* **vfrsubvf\_mask\_float64xm2** (*float64xm2\_t* *merge*, *float64xm2\_t* *a*, double *b*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *float64xm4\_t* **vfrsubvf\_mask\_float64xm4** (*float64xm4\_t* *merge*, *float64xm4\_t* *a*, double *b*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *float64xm8\_t* **vfrsubvf\_mask\_float64xm8** (*float64xm8\_t* *merge*, *float64xm8\_t* *a*, double *b*, *e64xm8\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = b - a[element]
      else
```

(continues on next page)

(continued from previous page)

```
result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.35 Elementwise vector-scalar floating-point sign copy

**Instruction:** ['vfsgnj.vf']

**Prototypes:**

- *float16xm1\_t vfsgnjvf\_float16xm1* (*float16xm1\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float16xm2\_t vfsgnjvf\_float16xm2* (*float16xm2\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float16xm4\_t vfsgnjvf\_float16xm4* (*float16xm4\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float16xm8\_t vfsgnjvf\_float16xm8* (*float16xm8\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float32xm1\_t vfsgnjvf\_float32xm1* (*float32xm1\_t a*, *float b*, unsigned int *gvl*)
- *float32xm2\_t vfsgnjvf\_float32xm2* (*float32xm2\_t a*, *float b*, unsigned int *gvl*)
- *float32xm4\_t vfsgnjvf\_float32xm4* (*float32xm4\_t a*, *float b*, unsigned int *gvl*)
- *float32xm8\_t vfsgnjvf\_float32xm8* (*float32xm8\_t a*, *float b*, unsigned int *gvl*)
- *float64xm1\_t vfsgnjvf\_float64xm1* (*float64xm1\_t a*, *double b*, unsigned int *gvl*)
- *float64xm2\_t vfsgnjvf\_float64xm2* (*float64xm2\_t a*, *double b*, unsigned int *gvl*)
- *float64xm4\_t vfsgnjvf\_float64xm4* (*float64xm4\_t a*, *double b*, unsigned int *gvl*)
- *float64xm8\_t vfsgnjvf\_float64xm8* (*float64xm8\_t a*, *double b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = fsgnj(a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfsgnjvf\_mask\_float16xm1* (*float16xm1\_t merge*, *float16xm1\_t a*, *float16\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfsgnjvf\_mask\_float16xm2* (*float16xm2\_t merge*, *float16xm2\_t a*, *float16\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfsgnjvf\_mask\_float16xm4* (*float16xm4\_t merge*, *float16xm4\_t a*, *float16\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *float16xm8\_t vfsgnjvf\_mask\_float16xm8* (*float16xm8\_t merge*, *float16xm8\_t a*, *float16\_t b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfsgnjvf\_mask\_float32xm1* (*float32xm1\_t merge*, *float32xm1\_t a*, *float b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfsgnjvf\_mask\_float32xm2* (*float32xm2\_t merge*, *float32xm2\_t a*, *float b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfsgnjvf\_mask\_float32xm4* (*float32xm4\_t merge*, *float32xm4\_t a*, *float b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *float32xm8\_t vfsgnjvf\_mask\_float32xm8* (*float32xm8\_t merge*, *float32xm8\_t a*, *float b*, *e32xm8\_t mask*, unsigned int *gvl*)

- `float64xm1_t vfsgnjvf_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `double b`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfsgnjvf_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `double b`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfsgnjvf_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `double b`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vfsgnjvf_mask_float64xm8` (`float64xm8_t merge`, `float64xm8_t a`, `double b`, `e64xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = fsgnj(a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.36 Elementwise vector-vector floating-point sign copy

Instruction: ['vfsgnj.vv']

Prototypes:

- `float16xm1_t vfsgnjvv_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, unsigned int `gvl`)
- `float16xm2_t vfsgnjvv_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, unsigned int `gvl`)
- `float16xm4_t vfsgnjvv_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, unsigned int `gvl`)
- `float16xm8_t vfsgnjvv_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, unsigned int `gvl`)
- `float32xm1_t vfsgnjvv_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, unsigned int `gvl`)
- `float32xm2_t vfsgnjvv_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, unsigned int `gvl`)
- `float32xm4_t vfsgnjvv_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, unsigned int `gvl`)
- `float32xm8_t vfsgnjvv_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, unsigned int `gvl`)
- `float64xm1_t vfsgnjvv_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, unsigned int `gvl`)
- `float64xm2_t vfsgnjvv_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, unsigned int `gvl`)
- `float64xm4_t vfsgnjvv_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, unsigned int `gvl`)
- `float64xm8_t vfsgnjvv_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = fsgnj(a[element], b[element])
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `float16xm1_t vfsgnjvv_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `float16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `float16xm2_t vfsgnjvv_mask_float16xm2` (`float16xm2_t merge`, `float16xm2_t a`, `float16xm2_t b`, `e16xm2_t mask`, unsigned int `gvl`)

- `float16xm4_t vfsgnjvv_mask_float16xm4` (`float16xm4_t merge`, `float16xm4_t a`, `float16xm4_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `float16xm8_t vfsgnjvv_mask_float16xm8` (`float16xm8_t merge`, `float16xm8_t a`, `float16xm8_t b`, `e16xm8_t mask`, unsigned int `gvl`)
- `float32xm1_t vfsgnjvv_mask_float32xm1` (`float32xm1_t merge`, `float32xm1_t a`, `float32xm1_t b`, `e32xm1_t mask`, unsigned int `gvl`)
- `float32xm2_t vfsgnjvv_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `float32xm2_t b`, `e32xm2_t mask`, unsigned int `gvl`)
- `float32xm4_t vfsgnjvv_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `float32xm4_t b`, `e32xm4_t mask`, unsigned int `gvl`)
- `float32xm8_t vfsgnjvv_mask_float32xm8` (`float32xm8_t merge`, `float32xm8_t a`, `float32xm8_t b`, `e32xm8_t mask`, unsigned int `gvl`)
- `float64xm1_t vfsgnjvv_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `float64xm1_t b`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfsgnjvv_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `float64xm2_t b`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfsgnjvv_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `float64xm4_t b`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vfsgnjvv_mask_float64xm8` (`float64xm8_t merge`, `float64xm8_t a`, `float64xm8_t b`, `e64xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = fsignj(a[element], b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.37 Elementwise vector-scalar floating-point inverted sign copy

Instruction: [`'vfsgnjn.vf'`]

Prototypes:

- `float16xm1_t vfsgnjnvf_float16xm1` (`float16xm1_t a`, `float16_t b`, unsigned int `gvl`)
- `float16xm2_t vfsgnjnvf_float16xm2` (`float16xm2_t a`, `float16_t b`, unsigned int `gvl`)
- `float16xm4_t vfsgnjnvf_float16xm4` (`float16xm4_t a`, `float16_t b`, unsigned int `gvl`)
- `float16xm8_t vfsgnjnvf_float16xm8` (`float16xm8_t a`, `float16_t b`, unsigned int `gvl`)
- `float32xm1_t vfsgnjnvf_float32xm1` (`float32xm1_t a`, `float b`, unsigned int `gvl`)
- `float32xm2_t vfsgnjnvf_float32xm2` (`float32xm2_t a`, `float b`, unsigned int `gvl`)
- `float32xm4_t vfsgnjnvf_float32xm4` (`float32xm4_t a`, `float b`, unsigned int `gvl`)
- `float32xm8_t vfsgnjnvf_float32xm8` (`float32xm8_t a`, `float b`, unsigned int `gvl`)
- `float64xm1_t vfsgnjnvf_float64xm1` (`float64xm1_t a`, `double b`, unsigned int `gvl`)
- `float64xm2_t vfsgnjnvf_float64xm2` (`float64xm2_t a`, `double b`, unsigned int `gvl`)
- `float64xm4_t vfsgnjnvf_float64xm4` (`float64xm4_t a`, `double b`, unsigned int `gvl`)

- *float64xm8\_t vfsgnjnvf\_float64xm8* (*float64xm8\_t a*, double *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = fsignjn(a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfsgnjnvf\_mask\_float16xm1* (*float16xm1\_t merge*, *float16xm1\_t a*, float16\_t *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfsgnjnvf\_mask\_float16xm2* (*float16xm2\_t merge*, *float16xm2\_t a*, float16\_t *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfsgnjnvf\_mask\_float16xm4* (*float16xm4\_t merge*, *float16xm4\_t a*, float16\_t *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *float16xm8\_t vfsgnjnvf\_mask\_float16xm8* (*float16xm8\_t merge*, *float16xm8\_t a*, float16\_t *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfsgnjnvf\_mask\_float32xm1* (*float32xm1\_t merge*, *float32xm1\_t a*, float *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfsgnjnvf\_mask\_float32xm2* (*float32xm2\_t merge*, *float32xm2\_t a*, float *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfsgnjnvf\_mask\_float32xm4* (*float32xm4\_t merge*, *float32xm4\_t a*, float *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *float32xm8\_t vfsgnjnvf\_mask\_float32xm8* (*float32xm8\_t merge*, *float32xm8\_t a*, float *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *float64xm1\_t vfsgnjnvf\_mask\_float64xm1* (*float64xm1\_t merge*, *float64xm1\_t a*, double *b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vfsgnjnvf\_mask\_float64xm2* (*float64xm2\_t merge*, *float64xm2\_t a*, double *b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *float64xm4\_t vfsgnjnvf\_mask\_float64xm4* (*float64xm4\_t merge*, *float64xm4\_t a*, double *b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *float64xm8\_t vfsgnjnvf\_mask\_float64xm8* (*float64xm8\_t merge*, *float64xm8\_t a*, double *b*, *e64xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = fsignjn(a[element], b)
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.38 Elementwise vector-vector floating-point inverted sign copy****Instruction:** ['vfsgnjn.vv']**Prototypes:**

- *float16xm1\_t vfsgnjnvv\_float16xm1* (*float16xm1\_t a*, *float16xm1\_t b*, unsigned int *gvl*)
- *float16xm2\_t vfsgnjnvv\_float16xm2* (*float16xm2\_t a*, *float16xm2\_t b*, unsigned int *gvl*)

- *float16xm4\_t vfsgnjnvv\_float16xm4* (*float16xm4\_t a, float16xm4\_t b*, unsigned int *gvl*)
- *float16xm8\_t vfsgnjnvv\_float16xm8* (*float16xm8\_t a, float16xm8\_t b*, unsigned int *gvl*)
- *float32xm1\_t vfsgnjnvv\_float32xm1* (*float32xm1\_t a, float32xm1\_t b*, unsigned int *gvl*)
- *float32xm2\_t vfsgnjnvv\_float32xm2* (*float32xm2\_t a, float32xm2\_t b*, unsigned int *gvl*)
- *float32xm4\_t vfsgnjnvv\_float32xm4* (*float32xm4\_t a, float32xm4\_t b*, unsigned int *gvl*)
- *float32xm8\_t vfsgnjnvv\_float32xm8* (*float32xm8\_t a, float32xm8\_t b*, unsigned int *gvl*)
- *float64xm1\_t vfsgnjnvv\_float64xm1* (*float64xm1\_t a, float64xm1\_t b*, unsigned int *gvl*)
- *float64xm2\_t vfsgnjnvv\_float64xm2* (*float64xm2\_t a, float64xm2\_t b*, unsigned int *gvl*)
- *float64xm4\_t vfsgnjnvv\_float64xm4* (*float64xm4\_t a, float64xm4\_t b*, unsigned int *gvl*)
- *float64xm8\_t vfsgnjnvv\_float64xm8* (*float64xm8\_t a, float64xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = fsignjn(a[element], b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfsgnjnvv\_mask\_float16xm1* (*float16xm1\_t merge, float16xm1\_t a, float16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfsgnjnvv\_mask\_float16xm2* (*float16xm2\_t merge, float16xm2\_t a, float16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfsgnjnvv\_mask\_float16xm4* (*float16xm4\_t merge, float16xm4\_t a, float16xm4\_t b, e16xm4\_t mask*, unsigned int *gvl*)
- *float16xm8\_t vfsgnjnvv\_mask\_float16xm8* (*float16xm8\_t merge, float16xm8\_t a, float16xm8\_t b, e16xm8\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfsgnjnvv\_mask\_float32xm1* (*float32xm1\_t merge, float32xm1\_t a, float32xm1\_t b, e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfsgnjnvv\_mask\_float32xm2* (*float32xm2\_t merge, float32xm2\_t a, float32xm2\_t b, e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfsgnjnvv\_mask\_float32xm4* (*float32xm4\_t merge, float32xm4\_t a, float32xm4\_t b, e32xm4\_t mask*, unsigned int *gvl*)
- *float32xm8\_t vfsgnjnvv\_mask\_float32xm8* (*float32xm8\_t merge, float32xm8\_t a, float32xm8\_t b, e32xm8\_t mask*, unsigned int *gvl*)
- *float64xm1\_t vfsgnjnvv\_mask\_float64xm1* (*float64xm1\_t merge, float64xm1\_t a, float64xm1\_t b, e64xm1\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vfsgnjnvv\_mask\_float64xm2* (*float64xm2\_t merge, float64xm2\_t a, float64xm2\_t b, e64xm2\_t mask*, unsigned int *gvl*)
- *float64xm4\_t vfsgnjnvv\_mask\_float64xm4* (*float64xm4\_t merge, float64xm4\_t a, float64xm4\_t b, e64xm4\_t mask*, unsigned int *gvl*)
- *float64xm8\_t vfsgnjnvv\_mask\_float64xm8* (*float64xm8\_t merge, float64xm8\_t a, float64xm8\_t b, e64xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**



```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = fsignjn(a[element], b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.39 Elementwise vector-scalar floating-point XOR sign

**Instruction:** ['vfsgnjx.vf']

**Prototypes:**

- *float16xm1\_t vfsgnjxvf\_float16xm1* (*float16xm1\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float16xm2\_t vfsgnjxvf\_float16xm2* (*float16xm2\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float16xm4\_t vfsgnjxvf\_float16xm4* (*float16xm4\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float16xm8\_t vfsgnjxvf\_float16xm8* (*float16xm8\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float32xm1\_t vfsgnjxvf\_float32xm1* (*float32xm1\_t a*, *float b*, unsigned int *gvl*)
- *float32xm2\_t vfsgnjxvf\_float32xm2* (*float32xm2\_t a*, *float b*, unsigned int *gvl*)
- *float32xm4\_t vfsgnjxvf\_float32xm4* (*float32xm4\_t a*, *float b*, unsigned int *gvl*)
- *float32xm8\_t vfsgnjxvf\_float32xm8* (*float32xm8\_t a*, *float b*, unsigned int *gvl*)
- *float64xm1\_t vfsgnjxvf\_float64xm1* (*float64xm1\_t a*, *double b*, unsigned int *gvl*)
- *float64xm2\_t vfsgnjxvf\_float64xm2* (*float64xm2\_t a*, *double b*, unsigned int *gvl*)
- *float64xm4\_t vfsgnjxvf\_float64xm4* (*float64xm4\_t a*, *double b*, unsigned int *gvl*)
- *float64xm8\_t vfsgnjxvf\_float64xm8* (*float64xm8\_t a*, *double b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = fsignx(a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vfsgnjxvf\_mask\_float16xm1* (*float16xm1\_t merge*, *float16xm1\_t a*, *float16\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vfsgnjxvf\_mask\_float16xm2* (*float16xm2\_t merge*, *float16xm2\_t a*, *float16\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfsgnjxvf\_mask\_float16xm4* (*float16xm4\_t merge*, *float16xm4\_t a*, *float16\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *float16xm8\_t vfsgnjxvf\_mask\_float16xm8* (*float16xm8\_t merge*, *float16xm8\_t a*, *float16\_t b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfsgnjxvf\_mask\_float32xm1* (*float32xm1\_t merge*, *float32xm1\_t a*, *float b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfsgnjxvf\_mask\_float32xm2* (*float32xm2\_t merge*, *float32xm2\_t a*, *float b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfsgnjxvf\_mask\_float32xm4* (*float32xm4\_t merge*, *float32xm4\_t a*, *float b*, *e32xm4\_t mask*, unsigned int *gvl*)

- `float32xm8_t vfsgnjxvf_mask_float32xm8` (`float32xm8_t merge`, `float32xm8_t a`, `float b`, `e32xm8_t mask`, unsigned int `gvl`)
- `float64xm1_t vfsgnjxvf_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `double b`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfsgnjxvf_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `double b`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfsgnjxvf_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `double b`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vfsgnjxvf_mask_float64xm8` (`float64xm8_t merge`, `float64xm8_t a`, `double b`, `e64xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = fsignx(a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.40 Elementwise vector-vector floating-point XOR sign

Instruction: ['vfsgnjx.vv']

Prototypes:

- `float16xm1_t vfsgnjxvv_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, unsigned int `gvl`)
- `float16xm2_t vfsgnjxvv_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, unsigned int `gvl`)
- `float16xm4_t vfsgnjxvv_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, unsigned int `gvl`)
- `float16xm8_t vfsgnjxvv_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, unsigned int `gvl`)
- `float32xm1_t vfsgnjxvv_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, unsigned int `gvl`)
- `float32xm2_t vfsgnjxvv_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, unsigned int `gvl`)
- `float32xm4_t vfsgnjxvv_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, unsigned int `gvl`)
- `float32xm8_t vfsgnjxvv_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, unsigned int `gvl`)
- `float64xm1_t vfsgnjxvv_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, unsigned int `gvl`)
- `float64xm2_t vfsgnjxvv_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, unsigned int `gvl`)
- `float64xm4_t vfsgnjxvv_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, unsigned int `gvl`)
- `float64xm8_t vfsgnjxvv_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = fsignx(a[element], b[element])
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `float16xm1_t vfsgnjxvv_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `float16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)

- *float16xm2\_t vfsgnjxvv\_mask\_float16xm2* (*float16xm2\_t merge, float16xm2\_t a, float16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vfsgnjxvv\_mask\_float16xm4* (*float16xm4\_t merge, float16xm4\_t a, float16xm4\_t b, e16xm4\_t mask*, unsigned int *gvl*)
- *float16xm8\_t vfsgnjxvv\_mask\_float16xm8* (*float16xm8\_t merge, float16xm8\_t a, float16xm8\_t b, e16xm8\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vfsgnjxvv\_mask\_float32xm1* (*float32xm1\_t merge, float32xm1\_t a, float32xm1\_t b, e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vfsgnjxvv\_mask\_float32xm2* (*float32xm2\_t merge, float32xm2\_t a, float32xm2\_t b, e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfsgnjxvv\_mask\_float32xm4* (*float32xm4\_t merge, float32xm4\_t a, float32xm4\_t b, e32xm4\_t mask*, unsigned int *gvl*)
- *float32xm8\_t vfsgnjxvv\_mask\_float32xm8* (*float32xm8\_t merge, float32xm8\_t a, float32xm8\_t b, e32xm8\_t mask*, unsigned int *gvl*)
- *float64xm1\_t vfsgnjxvv\_mask\_float64xm1* (*float64xm1\_t merge, float64xm1\_t a, float64xm1\_t b, e64xm1\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vfsgnjxvv\_mask\_float64xm2* (*float64xm2\_t merge, float64xm2\_t a, float64xm2\_t b, e64xm2\_t mask*, unsigned int *gvl*)
- *float64xm4\_t vfsgnjxvv\_mask\_float64xm4* (*float64xm4\_t merge, float64xm4\_t a, float64xm4\_t b, e64xm4\_t mask*, unsigned int *gvl*)
- *float64xm8\_t vfsgnjxvv\_mask\_float64xm8* (*float64xm8\_t merge, float64xm8\_t a, float64xm8\_t b, e64xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = fsignx(a[element], b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.41 Compute the square root****Instruction:** ['vfsqrt.v']**Prototypes:**

- *float16xm1\_t vfsqrtv\_float16xm1* (*float16xm1\_t a*, unsigned int *gvl*)
- *float16xm2\_t vfsqrtv\_float16xm2* (*float16xm2\_t a*, unsigned int *gvl*)
- *float16xm4\_t vfsqrtv\_float16xm4* (*float16xm4\_t a*, unsigned int *gvl*)
- *float16xm8\_t vfsqrtv\_float16xm8* (*float16xm8\_t a*, unsigned int *gvl*)
- *float32xm1\_t vfsqrtv\_float32xm1* (*float32xm1\_t a*, unsigned int *gvl*)
- *float32xm2\_t vfsqrtv\_float32xm2* (*float32xm2\_t a*, unsigned int *gvl*)
- *float32xm4\_t vfsqrtv\_float32xm4* (*float32xm4\_t a*, unsigned int *gvl*)
- *float32xm8\_t vfsqrtv\_float32xm8* (*float32xm8\_t a*, unsigned int *gvl*)
- *float64xm1\_t vfsqrtv\_float64xm1* (*float64xm1\_t a*, unsigned int *gvl*)

- `float64xm2_t vfsqrtv_float64xm2` (`float64xm2_t a`, unsigned int `gvl`)
- `float64xm4_t vfsqrtv_float64xm4` (`float64xm4_t a`, unsigned int `gvl`)
- `float64xm8_t vfsqrtv_float64xm8` (`float64xm8_t a`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = sqrt(a[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vfsqrtv_mask_float16xm1` (`float16xm1_t merge`, `float16xm1_t a`, `e16xm1_t mask`, unsigned int `gvl`)
- `float16xm2_t vfsqrtv_mask_float16xm2` (`float16xm2_t merge`, `float16xm2_t a`, `e16xm2_t mask`, unsigned int `gvl`)
- `float16xm4_t vfsqrtv_mask_float16xm4` (`float16xm4_t merge`, `float16xm4_t a`, `e16xm4_t mask`, unsigned int `gvl`)
- `float16xm8_t vfsqrtv_mask_float16xm8` (`float16xm8_t merge`, `float16xm8_t a`, `e16xm8_t mask`, unsigned int `gvl`)
- `float32xm1_t vfsqrtv_mask_float32xm1` (`float32xm1_t merge`, `float32xm1_t a`, `e32xm1_t mask`, unsigned int `gvl`)
- `float32xm2_t vfsqrtv_mask_float32xm2` (`float32xm2_t merge`, `float32xm2_t a`, `e32xm2_t mask`, unsigned int `gvl`)
- `float32xm4_t vfsqrtv_mask_float32xm4` (`float32xm4_t merge`, `float32xm4_t a`, `e32xm4_t mask`, unsigned int `gvl`)
- `float32xm8_t vfsqrtv_mask_float32xm8` (`float32xm8_t merge`, `float32xm8_t a`, `e32xm8_t mask`, unsigned int `gvl`)
- `float64xm1_t vfsqrtv_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfsqrtv_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfsqrtv_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vfsqrtv_mask_float64xm8` (`float64xm8_t merge`, `float64xm8_t a`, `e64xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = sqrt(a[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.42 Elementwise vector-scalar floating-point subtraction

**Instruction:** [`'vsub.vf'`]**Prototypes:**

- *float16xm1\_t vsubvbf\_float16xm1* (*float16xm1\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float16xm2\_t vsubvbf\_float16xm2* (*float16xm2\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float16xm4\_t vsubvbf\_float16xm4* (*float16xm4\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float16xm8\_t vsubvbf\_float16xm8* (*float16xm8\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float32xm1\_t vsubvbf\_float32xm1* (*float32xm1\_t a*, *float b*, unsigned int *gvl*)
- *float32xm2\_t vsubvbf\_float32xm2* (*float32xm2\_t a*, *float b*, unsigned int *gvl*)
- *float32xm4\_t vsubvbf\_float32xm4* (*float32xm4\_t a*, *float b*, unsigned int *gvl*)
- *float32xm8\_t vsubvbf\_float32xm8* (*float32xm8\_t a*, *float b*, unsigned int *gvl*)
- *float64xm1\_t vsubvbf\_float64xm1* (*float64xm1\_t a*, *double b*, unsigned int *gvl*)
- *float64xm2\_t vsubvbf\_float64xm2* (*float64xm2\_t a*, *double b*, unsigned int *gvl*)
- *float64xm4\_t vsubvbf\_float64xm4* (*float64xm4\_t a*, *double b*, unsigned int *gvl*)
- *float64xm8\_t vsubvbf\_float64xm8* (*float64xm8\_t a*, *double b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] - b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vsubvbf\_mask\_float16xm1* (*float16xm1\_t merge*, *float16xm1\_t a*, *float16\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vsubvbf\_mask\_float16xm2* (*float16xm2\_t merge*, *float16xm2\_t a*, *float16\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vsubvbf\_mask\_float16xm4* (*float16xm4\_t merge*, *float16xm4\_t a*, *float16\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *float16xm8\_t vsubvbf\_mask\_float16xm8* (*float16xm8\_t merge*, *float16xm8\_t a*, *float16\_t b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vsubvbf\_mask\_float32xm1* (*float32xm1\_t merge*, *float32xm1\_t a*, *float b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vsubvbf\_mask\_float32xm2* (*float32xm2\_t merge*, *float32xm2\_t a*, *float b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vsubvbf\_mask\_float32xm4* (*float32xm4\_t merge*, *float32xm4\_t a*, *float b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *float32xm8\_t vsubvbf\_mask\_float32xm8* (*float32xm8\_t merge*, *float32xm8\_t a*, *float b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *float64xm1\_t vsubvbf\_mask\_float64xm1* (*float64xm1\_t merge*, *float64xm1\_t a*, *double b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vsubvbf\_mask\_float64xm2* (*float64xm2\_t merge*, *float64xm2\_t a*, *double b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *float64xm4\_t vsubvbf\_mask\_float64xm4* (*float64xm4\_t merge*, *float64xm4\_t a*, *double b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *float64xm8\_t vsubvbf\_mask\_float64xm8* (*float64xm8\_t merge*, *float64xm8\_t a*, *double b*, *e64xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = a[element] - b
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.43 Elementwise vector-vector floating-point subtraction

**Instruction:** ['vsub.vv']

**Prototypes:**

- *float16xm1\_t vsubvv\_float16xm1* (*float16xm1\_t a, float16xm1\_t b*, unsigned int *gvl*)
- *float16xm2\_t vsubvv\_float16xm2* (*float16xm2\_t a, float16xm2\_t b*, unsigned int *gvl*)
- *float16xm4\_t vsubvv\_float16xm4* (*float16xm4\_t a, float16xm4\_t b*, unsigned int *gvl*)
- *float16xm8\_t vsubvv\_float16xm8* (*float16xm8\_t a, float16xm8\_t b*, unsigned int *gvl*)
- *float32xm1\_t vsubvv\_float32xm1* (*float32xm1\_t a, float32xm1\_t b*, unsigned int *gvl*)
- *float32xm2\_t vsubvv\_float32xm2* (*float32xm2\_t a, float32xm2\_t b*, unsigned int *gvl*)
- *float32xm4\_t vsubvv\_float32xm4* (*float32xm4\_t a, float32xm4\_t b*, unsigned int *gvl*)
- *float32xm8\_t vsubvv\_float32xm8* (*float32xm8\_t a, float32xm8\_t b*, unsigned int *gvl*)
- *float64xm1\_t vsubvv\_float64xm1* (*float64xm1\_t a, float64xm1\_t b*, unsigned int *gvl*)
- *float64xm2\_t vsubvv\_float64xm2* (*float64xm2\_t a, float64xm2\_t b*, unsigned int *gvl*)
- *float64xm4\_t vsubvv\_float64xm4* (*float64xm4\_t a, float64xm4\_t b*, unsigned int *gvl*)
- *float64xm8\_t vsubvv\_float64xm8* (*float64xm8\_t a, float64xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = a[element] - b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vsubvv\_mask\_float16xm1* (*float16xm1\_t merge, float16xm1\_t a, float16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vsubvv\_mask\_float16xm2* (*float16xm2\_t merge, float16xm2\_t a, float16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vsubvv\_mask\_float16xm4* (*float16xm4\_t merge, float16xm4\_t a, float16xm4\_t b, e16xm4\_t mask*, unsigned int *gvl*)
- *float16xm8\_t vsubvv\_mask\_float16xm8* (*float16xm8\_t merge, float16xm8\_t a, float16xm8\_t b, e16xm8\_t mask*, unsigned int *gvl*)
- *float32xm1\_t vsubvv\_mask\_float32xm1* (*float32xm1\_t merge, float32xm1\_t a, float32xm1\_t b, e32xm1\_t mask*, unsigned int *gvl*)
- *float32xm2\_t vsubvv\_mask\_float32xm2* (*float32xm2\_t merge, float32xm2\_t a, float32xm2\_t b, e32xm2\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vsubvv\_mask\_float32xm4* (*float32xm4\_t merge, float32xm4\_t a, float32xm4\_t b, e32xm4\_t mask*, unsigned int *gvl*)



- `float32xm8_t vsubvv_mask_float32xm8` (`float32xm8_t merge`, `float32xm8_t a`, `float32xm8_t b`, `e32xm8_t mask`, unsigned int `gvl`)
- `float64xm1_t vsubvv_mask_float64xm1` (`float64xm1_t merge`, `float64xm1_t a`, `float64xm1_t b`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vsubvv_mask_float64xm2` (`float64xm2_t merge`, `float64xm2_t a`, `float64xm2_t b`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vsubvv_mask_float64xm4` (`float64xm4_t merge`, `float64xm4_t a`, `float64xm4_t b`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vsubvv_mask_float64xm8` (`float64xm8_t merge`, `float64xm8_t a`, `float64xm8_t b`, `e64xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = a[element] - b[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.44 Floating-point vector-scalar widening additon

Instruction: ['vfwadd.vf']

Prototypes:

- `float32xm2_t vfwaddvf_float32xm2_float16xm1` (`float16xm1_t a`, `float16_t b`, unsigned int `gvl`)
- `float32xm4_t vfwaddvf_float32xm4_float16xm2` (`float16xm2_t a`, `float16_t b`, unsigned int `gvl`)
- `float32xm8_t vfwaddvf_float32xm8_float16xm4` (`float16xm4_t a`, `float16_t b`, unsigned int `gvl`)
- `float64xm2_t vfwaddvf_float64xm2_float32xm1` (`float32xm1_t a`, `float b`, unsigned int `gvl`)
- `float64xm4_t vfwaddvf_float64xm4_float32xm2` (`float32xm2_t a`, `float b`, unsigned int `gvl`)
- `float64xm8_t vfwaddvf_float64xm8_float32xm4` (`float32xm4_t a`, `float b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = wide_fp(a[element]) + wide_fp(b[element])
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `float32xm2_t vfwaddvf_mask_float32xm2_float16xm1` (`float32xm2_t merge`, `float16xm1_t a`, `float16_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `float32xm4_t vfwaddvf_mask_float32xm4_float16xm2` (`float32xm4_t merge`, `float16xm2_t a`, `float16_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `float32xm8_t vfwaddvf_mask_float32xm8_float16xm4` (`float32xm8_t merge`, `float16xm4_t a`, `float16_t b`, `e16xm4_t mask`, unsigned int `gvl`)

- *float64xm2\_t* **vfwaddvf\_mask\_float64xm2\_float32xm1** (*float64xm2\_t* merge, *float32xm1\_t* a, float b, *e32xm1\_t* mask, unsigned int gvl)
- *float64xm4\_t* **vfwaddvf\_mask\_float64xm4\_float32xm2** (*float64xm4\_t* merge, *float32xm2\_t* a, float b, *e32xm2\_t* mask, unsigned int gvl)
- *float64xm8\_t* **vfwaddvf\_mask\_float64xm8\_float32xm4** (*float64xm8\_t* merge, *float32xm4\_t* a, float b, *e32xm4\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = wide_fp(a[element]) + wide_fp(b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.45 Floating-point vector-vector widening additon****Instruction:** ['vfwadd.vv']**Prototypes:**

- *float32xm2\_t* **vfwaddvv\_float32xm2\_float16xm1** (*float16xm1\_t* a, *float16xm1\_t* b, unsigned int gvl)
- *float32xm4\_t* **vfwaddvv\_float32xm4\_float16xm2** (*float16xm2\_t* a, *float16xm2\_t* b, unsigned int gvl)
- *float32xm8\_t* **vfwaddvv\_float32xm8\_float16xm4** (*float16xm4\_t* a, *float16xm4\_t* b, unsigned int gvl)
- *float64xm2\_t* **vfwaddvv\_float64xm2\_float32xm1** (*float32xm1\_t* a, *float32xm1\_t* b, unsigned int gvl)
- *float64xm4\_t* **vfwaddvv\_float64xm4\_float32xm2** (*float32xm2\_t* a, *float32xm2\_t* b, unsigned int gvl)
- *float64xm8\_t* **vfwaddvv\_float64xm8\_float32xm4** (*float32xm4\_t* a, *float32xm4\_t* b, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = wide_fp(a[element]) + wide_fp(b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float32xm2\_t* **vfwaddvv\_mask\_float32xm2\_float16xm1** (*float32xm2\_t* merge, *float16xm1\_t* a, *float16xm1\_t* b, *e16xm1\_t* mask, unsigned int gvl)
- *float32xm4\_t* **vfwaddvv\_mask\_float32xm4\_float16xm2** (*float32xm4\_t* merge, *float16xm2\_t* a, *float16xm2\_t* b, *e16xm2\_t* mask, unsigned int gvl)

- `float32xm8_t vfwaddvv_mask_float32xm8_float16xm4` (`float32xm8_t` merge, `float16xm4_t` *a*, `float16xm4_t` *b*, `e16xm4_t` mask, unsigned int *gvl*)
- `float64xm2_t vfwaddvv_mask_float64xm2_float32xm1` (`float64xm2_t` merge, `float32xm1_t` *a*, `float32xm1_t` *b*, `e32xm1_t` mask, unsigned int *gvl*)
- `float64xm4_t vfwaddvv_mask_float64xm4_float32xm2` (`float64xm4_t` merge, `float32xm2_t` *a*, `float32xm2_t` *b*, `e32xm2_t` mask, unsigned int *gvl*)
- `float64xm8_t vfwaddvv_mask_float64xm8_float32xm4` (`float64xm8_t` merge, `float32xm4_t` *a*, `float32xm4_t` *b*, `e32xm4_t` mask, unsigned int *gvl*)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = wide_fp(a[element]) + wide_fp(b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.46 Floating-point vector-scalar widening additon(second operand)

Instruction: ['vfwadd.wf']

Prototypes:

- `float32xm2_t vfwaddwf_float32xm2` (`float32xm2_t` *a*, `float16_t` *b*, unsigned int *gvl*)
- `float32xm4_t vfwaddwf_float32xm4` (`float32xm4_t` *a*, `float16_t` *b*, unsigned int *gvl*)
- `float32xm8_t vfwaddwf_float32xm8` (`float32xm8_t` *a*, `float16_t` *b*, unsigned int *gvl*)
- `float64xm2_t vfwaddwf_float64xm2` (`float64xm2_t` *a*, `float` *b*, unsigned int *gvl*)
- `float64xm4_t vfwaddwf_float64xm4` (`float64xm4_t` *a*, `float` *b*, unsigned int *gvl*)
- `float64xm8_t vfwaddwf_float64xm8` (`float64xm8_t` *a*, `float` *b*, unsigned int *gvl*)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + wide_fp(b[element])
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `float32xm2_t vfwaddwf_mask_float32xm2` (`float32xm2_t` merge, `float32xm2_t` *a*, `float16_t` *b*, `e16xm1_t` mask, unsigned int *gvl*)
- `float32xm4_t vfwaddwf_mask_float32xm4` (`float32xm4_t` merge, `float32xm4_t` *a*, `float16_t` *b*, `e16xm2_t` mask, unsigned int *gvl*)
- `float32xm8_t vfwaddwf_mask_float32xm8` (`float32xm8_t` merge, `float32xm8_t` *a*, `float16_t` *b*, `e16xm4_t` mask, unsigned int *gvl*)
- `float64xm2_t vfwaddwf_mask_float64xm2` (`float64xm2_t` merge, `float64xm2_t` *a*, `float` *b*, `e32xm1_t` mask, unsigned int *gvl*)

- *float64xm4\_t* **vfwaddwf\_mask\_float64xm4** (*float64xm4\_t* merge, *float64xm4\_t* a, float b, *e32xm2\_t* mask, unsigned int gvl)
- *float64xm8\_t* **vfwaddwf\_mask\_float64xm8** (*float64xm8\_t* merge, *float64xm8\_t* a, float b, *e32xm4\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = a[element] + wide_fp(b[element])
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.5.47 Floating-point vector-vector widening additon(second operand)

Instruction: ['vfwadd.wv']

Prototypes:

- *float32xm2\_t* **vfwaddwv\_float32xm2\_float16xm1** (*float32xm2\_t* a, *float16xm1\_t* b, unsigned int gvl)
- *float32xm4\_t* **vfwaddwv\_float32xm4\_float16xm2** (*float32xm4\_t* a, *float16xm2\_t* b, unsigned int gvl)
- *float32xm8\_t* **vfwaddwv\_float32xm8\_float16xm4** (*float32xm8\_t* a, *float16xm4\_t* b, unsigned int gvl)
- *float64xm2\_t* **vfwaddwv\_float64xm2\_float32xm1** (*float64xm2\_t* a, *float32xm1\_t* b, unsigned int gvl)
- *float64xm4\_t* **vfwaddwv\_float64xm4\_float32xm2** (*float64xm4\_t* a, *float32xm2\_t* b, unsigned int gvl)
- *float64xm8\_t* **vfwaddwv\_float64xm8\_float32xm4** (*float64xm8\_t* a, *float32xm4\_t* b, unsigned int gvl)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + wide_fp(b[element])
      result[gvl : VLMAX] = 0
```

Masked prototypes:

- *float32xm2\_t* **vfwaddwv\_mask\_float32xm2\_float16xm1** (*float32xm2\_t* merge, *float32xm2\_t* a, *float16xm1\_t* b, *e16xm1\_t* mask, unsigned int gvl)
- *float32xm4\_t* **vfwaddwv\_mask\_float32xm4\_float16xm2** (*float32xm4\_t* merge, *float32xm4\_t* a, *float16xm2\_t* b, *e16xm2\_t* mask, unsigned int gvl)
- *float32xm8\_t* **vfwaddwv\_mask\_float32xm8\_float16xm4** (*float32xm8\_t* merge, *float32xm8\_t* a, *float16xm4\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *float64xm2\_t* **vfwaddwv\_mask\_float64xm2\_float32xm1** (*float64xm2\_t* merge, *float64xm2\_t* a, *float32xm1\_t* b, *e32xm1\_t* mask, unsigned int gvl)

- *float64xm4\_t* **vfwaddwv\_mask\_float64xm4\_float32xm2** (*float64xm4\_t* merge, *float64xm4\_t* a, *float32xm2\_t* b, *e32xm2\_t* mask, unsigned int gvl)
- *float64xm8\_t* **vfwaddwv\_mask\_float64xm8\_float32xm4** (*float64xm8\_t* merge, *float64xm8\_t* a, *float32xm4\_t* b, *e32xm4\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = a[element] + wide_fp(b[element])
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.5.48 Floating-point vector-scalar widening multiply and add

Instruction: [*vfwmaccvf.vf*]

Prototypes:

- *float32xm2\_t* **vfwmaccvf\_float32xm2\_float16xm1** (*float32xm2\_t* a, *float16\_t* b, *float16xm1\_t* c, unsigned int gvl)
- *float32xm4\_t* **vfwmaccvf\_float32xm4\_float16xm2** (*float32xm4\_t* a, *float16\_t* b, *float16xm2\_t* c, unsigned int gvl)
- *float32xm8\_t* **vfwmaccvf\_float32xm8\_float16xm4** (*float32xm8\_t* a, *float16\_t* b, *float16xm4\_t* c, unsigned int gvl)
- *float64xm2\_t* **vfwmaccvf\_float64xm2\_float32xm1** (*float64xm2\_t* a, float b, *float32xm1\_t* c, unsigned int gvl)
- *float64xm4\_t* **vfwmaccvf\_float64xm4\_float32xm2** (*float64xm4\_t* a, float b, *float32xm2\_t* c, unsigned int gvl)
- *float64xm8\_t* **vfwmaccvf\_float64xm8\_float32xm4** (*float64xm8\_t* a, float b, *float32xm4\_t* c, unsigned int gvl)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = wide_fp(b[element]) * wide_fp(c[element]) + a[element]
    result[gvl : VLMAX] = 0
```

Masked prototypes:

- *float32xm2\_t* **vfwmaccvf\_mask\_float32xm2\_float16xm1** (*float32xm2\_t* merge, *float32xm2\_t* a, *float16\_t* b, *float16xm1\_t* c, *e16xm1\_t* mask, unsigned int gvl)
- *float32xm4\_t* **vfwmaccvf\_mask\_float32xm4\_float16xm2** (*float32xm4\_t* merge, *float32xm4\_t* a, *float16\_t* b, *float16xm2\_t* c, *e16xm2\_t* mask, unsigned int gvl)
- *float32xm8\_t* **vfwmaccvf\_mask\_float32xm8\_float16xm4** (*float32xm8\_t* merge, *float32xm8\_t* a, *float16\_t* b, *float16xm4\_t* c, *e16xm4\_t* mask, unsigned int gvl)

- `float64xm2_t vfwmaccvf_mask_float64xm2_float32xm1` (`float64xm2_t merge, float64xm2_t a, float b, float32xm1_t c, e32xm1_t mask, unsigned int gvl`)
- `float64xm4_t vfwmaccvf_mask_float64xm4_float32xm2` (`float64xm4_t merge, float64xm4_t a, float b, float32xm2_t c, e32xm2_t mask, unsigned int gvl`)
- `float64xm8_t vfwmaccvf_mask_float64xm8_float32xm4` (`float64xm8_t merge, float64xm8_t a, float b, float32xm4_t c, e32xm4_t mask, unsigned int gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = wide_fp(b[element]) * wide_fp(c[element]) + a[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.49 Floating-point vector-vector widening multiply and add****Instruction:** [`'vfwmacv.vv'`]**Prototypes:**

- `float32xm2_t vfwmacvv_float32xm2_float16xm1` (`float32xm2_t a, float16xm1_t b, float16xm1_t c, unsigned int gvl`)
- `float32xm4_t vfwmacvv_float32xm4_float16xm2` (`float32xm4_t a, float16xm2_t b, float16xm2_t c, unsigned int gvl`)
- `float32xm8_t vfwmacvv_float32xm8_float16xm4` (`float32xm8_t a, float16xm4_t b, float16xm4_t c, unsigned int gvl`)
- `float64xm2_t vfwmacvv_float64xm2_float32xm1` (`float64xm2_t a, float32xm1_t b, float32xm1_t c, unsigned int gvl`)
- `float64xm4_t vfwmacvv_float64xm4_float32xm2` (`float64xm4_t a, float32xm2_t b, float32xm2_t c, unsigned int gvl`)
- `float64xm8_t vfwmacvv_float64xm8_float32xm4` (`float64xm8_t a, float32xm4_t b, float32xm4_t c, unsigned int gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = wide_fp(b[element]) * wide_fp(c[element]) + a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float32xm2_t vfwmacvv_mask_float32xm2_float16xm1` (`float32xm2_t merge, float32xm2_t a, float16xm1_t b, float16xm1_t c, e16xm1_t mask, unsigned int gvl`)
- `float32xm4_t vfwmacvv_mask_float32xm4_float16xm2` (`float32xm4_t merge, float32xm4_t a, float16xm2_t b, float16xm2_t c, e16xm2_t mask, unsigned int gvl`)



- *float32xm8\_t* **vfwmaccvv\_mask\_float32xm8\_float16xm4** (*float32xm8\_t* merge, *float32xm8\_t* a, *float16xm4\_t* b, *float16xm4\_t* c, *e16xm4\_t* mask, unsigned int gvl)
- *float64xm2\_t* **vfwmaccvv\_mask\_float64xm2\_float32xm1** (*float64xm2\_t* merge, *float64xm2\_t* a, *float32xm1\_t* b, *float32xm1\_t* c, *e32xm1\_t* mask, unsigned int gvl)
- *float64xm4\_t* **vfwmaccvv\_mask\_float64xm4\_float32xm2** (*float64xm4\_t* merge, *float64xm4\_t* a, *float32xm2\_t* b, *float32xm2\_t* c, *e32xm2\_t* mask, unsigned int gvl)
- *float64xm8\_t* **vfwmaccvv\_mask\_float64xm8\_float32xm4** (*float64xm8\_t* merge, *float64xm8\_t* a, *float32xm4\_t* b, *float32xm4\_t* c, *e32xm4\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = wide_fp(b[element]) * wide_fp(c[element]) + a[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.50 Floating-point vector-scalar widening multiply and sub

Instruction: ['vfwmsac.vf']

Prototypes:

- *float32xm2\_t* **vfwmsacvf\_float32xm2\_float16xm1** (*float32xm2\_t* a, *float16\_t* b, *float16xm1\_t* c, unsigned int gvl)
- *float32xm4\_t* **vfwmsacvf\_float32xm4\_float16xm2** (*float32xm4\_t* a, *float16\_t* b, *float16xm2\_t* c, unsigned int gvl)
- *float32xm8\_t* **vfwmsacvf\_float32xm8\_float16xm4** (*float32xm8\_t* a, *float16\_t* b, *float16xm4\_t* c, unsigned int gvl)
- *float64xm2\_t* **vfwmsacvf\_float64xm2\_float32xm1** (*float64xm2\_t* a, *float b*, *float32xm1\_t* c, unsigned int gvl)
- *float64xm4\_t* **vfwmsacvf\_float64xm4\_float32xm2** (*float64xm4\_t* a, *float b*, *float32xm2\_t* c, unsigned int gvl)
- *float64xm8\_t* **vfwmsacvf\_float64xm8\_float32xm4** (*float64xm8\_t* a, *float b*, *float32xm4\_t* c, unsigned int gvl)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = wide_fp(b[element]) * wide_fp(c[element]) - a[element]
result[gvl : VLMAX] = 0
```

Masked prototypes:

- *float32xm2\_t* **vfwmsacvf\_mask\_float32xm2\_float16xm1** (*float32xm2\_t* merge, *float32xm2\_t* a, *float16\_t* b, *float16xm1\_t* c, *e16xm1\_t* mask, unsigned int gvl)

- `float32xm4_t vfwmsacvf_mask_float32xm4_float16xm2` (`float32xm4_t` merge, `float32xm4_t` *a*, `float16_t` *b*, `float16xm2_t` *c*, `e16xm2_t` mask, unsigned int *gvl*)
- `float32xm8_t vfwmsacvf_mask_float32xm8_float16xm4` (`float32xm8_t` merge, `float32xm8_t` *a*, `float16_t` *b*, `float16xm4_t` *c*, `e16xm4_t` mask, unsigned int *gvl*)
- `float64xm2_t vfwmsacvf_mask_float64xm2_float32xm1` (`float64xm2_t` merge, `float64xm2_t` *a*, `float` *b*, `float32xm1_t` *c*, `e32xm1_t` mask, unsigned int *gvl*)
- `float64xm4_t vfwmsacvf_mask_float64xm4_float32xm2` (`float64xm4_t` merge, `float64xm4_t` *a*, `float` *b*, `float32xm2_t` *c*, `e32xm2_t` mask, unsigned int *gvl*)
- `float64xm8_t vfwmsacvf_mask_float64xm8_float32xm4` (`float64xm8_t` merge, `float64xm8_t` *a*, `float` *b*, `float32xm4_t` *c*, `e32xm4_t` mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = wide_fp(b[element]) * wide_fp(c[element]) - a[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.51 Floating-point vector-vector widening multiply and sub

**Instruction:** [`'vfwmsac.vv'`]**Prototypes:**

- `float32xm2_t vfwmsacvv_float32xm2_float16xm1` (`float32xm2_t` *a*, `float16xm1_t` *b*, `float16xm1_t` *c*, unsigned int *gvl*)
- `float32xm4_t vfwmsacvv_float32xm4_float16xm2` (`float32xm4_t` *a*, `float16xm2_t` *b*, `float16xm2_t` *c*, unsigned int *gvl*)
- `float32xm8_t vfwmsacvv_float32xm8_float16xm4` (`float32xm8_t` *a*, `float16xm4_t` *b*, `float16xm4_t` *c*, unsigned int *gvl*)
- `float64xm2_t vfwmsacvv_float64xm2_float32xm1` (`float64xm2_t` *a*, `float32xm1_t` *b*, `float32xm1_t` *c*, unsigned int *gvl*)
- `float64xm4_t vfwmsacvv_float64xm4_float32xm2` (`float64xm4_t` *a*, `float32xm2_t` *b*, `float32xm2_t` *c*, unsigned int *gvl*)
- `float64xm8_t vfwmsacvv_float64xm8_float32xm4` (`float64xm8_t` *a*, `float32xm4_t` *b*, `float32xm4_t` *c*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = wide_fp(b[element]) * wide_fp(c[element]) - a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float32xm2_t vfwmsacvv_mask_float32xm2_float16xm1` (`float32xm2_t merge`, `float32xm2_t a`, `float16xm1_t b`, `float16xm1_t c`, `e16xm1_t mask`, unsigned int `gvl`)
- `float32xm4_t vfwmsacvv_mask_float32xm4_float16xm2` (`float32xm4_t merge`, `float32xm4_t a`, `float16xm2_t b`, `float16xm2_t c`, `e16xm2_t mask`, unsigned int `gvl`)
- `float32xm8_t vfwmsacvv_mask_float32xm8_float16xm4` (`float32xm8_t merge`, `float32xm8_t a`, `float16xm4_t b`, `float16xm4_t c`, `e16xm4_t mask`, unsigned int `gvl`)
- `float64xm2_t vfwmsacvv_mask_float64xm2_float32xm1` (`float64xm2_t merge`, `float64xm2_t a`, `float32xm1_t b`, `float32xm1_t c`, `e32xm1_t mask`, unsigned int `gvl`)
- `float64xm4_t vfwmsacvv_mask_float64xm4_float32xm2` (`float64xm4_t merge`, `float64xm4_t a`, `float32xm2_t b`, `float32xm2_t c`, `e32xm2_t mask`, unsigned int `gvl`)
- `float64xm8_t vfwmsacvv_mask_float64xm8_float32xm4` (`float64xm8_t merge`, `float64xm8_t a`, `float32xm4_t b`, `float32xm4_t c`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = wide_fp(b[element]) * wide_fp(c[element]) - a[element]
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.5.52 Floating-point vector-scalar widening multiplication

**Instruction:** [`'vfwmul.vf'`]**Prototypes:**

- `float32xm2_t vfwmulvf_float32xm2_float16xm1` (`float16xm1_t a`, `float16_t b`, unsigned int `gvl`)
- `float32xm4_t vfwmulvf_float32xm4_float16xm2` (`float16xm2_t a`, `float16_t b`, unsigned int `gvl`)
- `float32xm8_t vfwmulvf_float32xm8_float16xm4` (`float16xm4_t a`, `float16_t b`, unsigned int `gvl`)
- `float64xm2_t vfwmulvf_float64xm2_float32xm1` (`float32xm1_t a`, `float b`, unsigned int `gvl`)
- `float64xm4_t vfwmulvf_float64xm4_float32xm2` (`float32xm2_t a`, `float b`, unsigned int `gvl`)
- `float64xm8_t vfwmulvf_float64xm8_float32xm4` (`float32xm4_t a`, `float b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = wide_fp(a[element]) * wide_fp(b[element])
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float32xm2_t vfwmulvf_mask_float32xm2_float16xm1` (`float32xm2_t merge`, `float16xm1_t a`, `float16_t b`, `e16xm1_t mask`, unsigned int `gvl`)

- `float32xm4_t vfwmulvf_mask_float32xm4_float16xm2` (`float32xm4_t` merge, `float16xm2_t` `a`, `float16_t` `b`, `e16xm2_t` mask, unsigned int `gvl`)
- `float32xm8_t vfwmulvf_mask_float32xm8_float16xm4` (`float32xm8_t` merge, `float16xm4_t` `a`, `float16_t` `b`, `e16xm4_t` mask, unsigned int `gvl`)
- `float64xm2_t vfwmulvf_mask_float64xm2_float32xm1` (`float64xm2_t` merge, `float32xm1_t` `a`, `float` `b`, `e32xm1_t` mask, unsigned int `gvl`)
- `float64xm4_t vfwmulvf_mask_float64xm4_float32xm2` (`float64xm4_t` merge, `float32xm2_t` `a`, `float` `b`, `e32xm2_t` mask, unsigned int `gvl`)
- `float64xm8_t vfwmulvf_mask_float64xm8_float32xm4` (`float64xm8_t` merge, `float32xm4_t` `a`, `float` `b`, `e32xm4_t` mask, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = wide_fp(a[element]) * wide_fp(b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.53 Floating-point vector-vector widening multiplication

**Instruction:** [`'vfwmul.vv'`]**Prototypes:**

- `float32xm2_t vfwmulvv_float32xm2_float16xm1` (`float16xm1_t` `a`, `float16xm1_t` `b`, unsigned int `gvl`)
- `float32xm4_t vfwmulvv_float32xm4_float16xm2` (`float16xm2_t` `a`, `float16xm2_t` `b`, unsigned int `gvl`)
- `float32xm8_t vfwmulvv_float32xm8_float16xm4` (`float16xm4_t` `a`, `float16xm4_t` `b`, unsigned int `gvl`)
- `float64xm2_t vfwmulvv_float64xm2_float32xm1` (`float32xm1_t` `a`, `float32xm1_t` `b`, unsigned int `gvl`)
- `float64xm4_t vfwmulvv_float64xm4_float32xm2` (`float32xm2_t` `a`, `float32xm2_t` `b`, unsigned int `gvl`)
- `float64xm8_t vfwmulvv_float64xm8_float32xm4` (`float32xm4_t` `a`, `float32xm4_t` `b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = wide_fp(a[element]) * wide_fp(b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float32xm2_t vfwmulvv_mask_float32xm2_float16xm1` (`float32xm2_t` merge, `float16xm1_t` *a*, `float16xm1_t` *b*, `e16xm1_t` mask, unsigned int *gvl*)
- `float32xm4_t vfwmulvv_mask_float32xm4_float16xm2` (`float32xm4_t` merge, `float16xm2_t` *a*, `float16xm2_t` *b*, `e16xm2_t` mask, unsigned int *gvl*)
- `float32xm8_t vfwmulvv_mask_float32xm8_float16xm4` (`float32xm8_t` merge, `float16xm4_t` *a*, `float16xm4_t` *b*, `e16xm4_t` mask, unsigned int *gvl*)
- `float64xm2_t vfwmulvv_mask_float64xm2_float32xm1` (`float64xm2_t` merge, `float32xm1_t` *a*, `float32xm1_t` *b*, `e32xm1_t` mask, unsigned int *gvl*)
- `float64xm4_t vfwmulvv_mask_float64xm4_float32xm2` (`float64xm4_t` merge, `float32xm2_t` *a*, `float32xm2_t` *b*, `e32xm2_t` mask, unsigned int *gvl*)
- `float64xm8_t vfwmulvv_mask_float64xm8_float32xm4` (`float64xm8_t` merge, `float32xm4_t` *a*, `float32xm4_t` *b*, `e32xm4_t` mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = wide_fp(a[element]) * wide_fp(b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.54 Floating-point vector-scalar widening negate multiply and add

**Instruction:** [`'vfwnmacc.vf'`]**Prototypes:**

- `float32xm2_t vfwnmaccvf_float32xm2_float16xm1` (`float32xm2_t` *a*, `float16_t` *b*, `float16xm1_t` *c*, unsigned int *gvl*)
- `float32xm4_t vfwnmaccvf_float32xm4_float16xm2` (`float32xm4_t` *a*, `float16_t` *b*, `float16xm2_t` *c*, unsigned int *gvl*)
- `float32xm8_t vfwnmaccvf_float32xm8_float16xm4` (`float32xm8_t` *a*, `float16_t` *b*, `float16xm4_t` *c*, unsigned int *gvl*)
- `float64xm2_t vfwnmaccvf_float64xm2_float32xm1` (`float64xm2_t` *a*, `float` *b*, `float32xm1_t` *c*, unsigned int *gvl*)
- `float64xm4_t vfwnmaccvf_float64xm4_float32xm2` (`float64xm4_t` *a*, `float` *b*, `float32xm2_t` *c*, unsigned int *gvl*)
- `float64xm8_t vfwnmaccvf_float64xm8_float32xm4` (`float64xm8_t` *a*, `float` *b*, `float32xm4_t` *c*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = -(wide_fp(b[element]) * wide_fp(c[element])) - a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float32xm2_t vfwnmaccvf_mask_float32xm2_float16xm1` (`float32xm2_t` *merge*,  
`float32xm2_t` *a*, `float16_t` *b*,  
`float16xm1_t` *c*, `e16xm1_t` *mask*,  
`unsigned int gvl`)
- `float32xm4_t vfwnmaccvf_mask_float32xm4_float16xm2` (`float32xm4_t` *merge*,  
`float32xm4_t` *a*, `float16_t` *b*,  
`float16xm2_t` *c*, `e16xm2_t` *mask*,  
`unsigned int gvl`)
- `float32xm8_t vfwnmaccvf_mask_float32xm8_float16xm4` (`float32xm8_t` *merge*,  
`float32xm8_t` *a*, `float16_t` *b*,  
`float16xm4_t` *c*, `e16xm4_t` *mask*,  
`unsigned int gvl`)
- `float64xm2_t vfwnmaccvf_mask_float64xm2_float32xm1` (`float64xm2_t` *merge*,  
`float64xm2_t` *a*, `float` *b*,  
`float32xm1_t` *c*, `e32xm1_t` *mask*,  
`unsigned int gvl`)
- `float64xm4_t vfwnmaccvf_mask_float64xm4_float32xm2` (`float64xm4_t` *merge*,  
`float64xm4_t` *a*, `float` *b*,  
`float32xm2_t` *c*, `e32xm2_t` *mask*,  
`unsigned int gvl`)
- `float64xm8_t vfwnmaccvf_mask_float64xm8_float32xm4` (`float64xm8_t` *merge*,  
`float64xm8_t` *a*, `float` *b*,  
`float32xm4_t` *c*, `e32xm4_t` *mask*,  
`unsigned int gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = -(wide_fp(b[element]) * wide_fp(c[element])) - a[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.5.55 Floating-point vector-vector widening negate multiply and add****Instruction:** ['vfwnmacc.vv']**Prototypes:**

- `float32xm2_t vfwnmaccvv_float32xm2_float16xm1` (`float32xm2_t` *a*, `float16xm1_t` *b*,  
`float16xm1_t` *c*, `unsigned int gvl`)
- `float32xm4_t vfwnmaccvv_float32xm4_float16xm2` (`float32xm4_t` *a*, `float16xm2_t` *b*,  
`float16xm2_t` *c*, `unsigned int gvl`)
- `float32xm8_t vfwnmaccvv_float32xm8_float16xm4` (`float32xm8_t` *a*, `float16xm4_t` *b*,  
`float16xm4_t` *c*, `unsigned int gvl`)
- `float64xm2_t vfwnmaccvv_float64xm2_float32xm1` (`float64xm2_t` *a*, `float32xm1_t` *b*,  
`float32xm1_t` *c*, `unsigned int gvl`)
- `float64xm4_t vfwnmaccvv_float64xm4_float32xm2` (`float64xm4_t` *a*, `float32xm2_t` *b*,  
`float32xm2_t` *c*, `unsigned int gvl`)



- *float64xm8\_t* **vfwnmaccvv\_float64xm8\_float32xm4** (*float64xm8\_t* *a*, *float32xm4\_t* *b*, *float32xm4\_t* *c*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = -(wide_fp(b[element]) * wide_fp(c[element])) - a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float32xm2\_t* **vfwnmaccvv\_mask\_float32xm2\_float16xm1** (*float32xm2\_t* *merge*, *float32xm2\_t* *a*, *float16xm1\_t* *b*, *float16xm1\_t* *c*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *float32xm4\_t* **vfwnmaccvv\_mask\_float32xm4\_float16xm2** (*float32xm4\_t* *merge*, *float32xm4\_t* *a*, *float16xm2\_t* *b*, *float16xm2\_t* *c*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *float32xm8\_t* **vfwnmaccvv\_mask\_float32xm8\_float16xm4** (*float32xm8\_t* *merge*, *float32xm8\_t* *a*, *float16xm4\_t* *b*, *float16xm4\_t* *c*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *float64xm2\_t* **vfwnmaccvv\_mask\_float64xm2\_float32xm1** (*float64xm2\_t* *merge*, *float64xm2\_t* *a*, *float32xm1\_t* *b*, *float32xm1\_t* *c*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *float64xm4\_t* **vfwnmaccvv\_mask\_float64xm4\_float32xm2** (*float64xm4\_t* *merge*, *float64xm4\_t* *a*, *float32xm2\_t* *b*, *float32xm2\_t* *c*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *float64xm8\_t* **vfwnmaccvv\_mask\_float64xm8\_float32xm4** (*float64xm8\_t* *merge*, *float64xm8\_t* *a*, *float32xm4\_t* *b*, *float32xm4\_t* *c*, *e32xm4\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = -(wide_fp(b[element]) * wide_fp(c[element])) - a[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.56 Floating-point vector-scalar widening negate multiply and sub

**Instruction:** [*'vfwnmsac.vf'*]**Prototypes:**

- *float32xm2\_t* **vfwnmsacvf\_float32xm2\_float16xm1** (*float32xm2\_t* *a*, *float16\_t* *b*, *float16xm1\_t* *c*, unsigned int *gvl*)
- *float32xm4\_t* **vfwnmsacvf\_float32xm4\_float16xm2** (*float32xm4\_t* *a*, *float16\_t* *b*, *float16xm2\_t* *c*, unsigned int *gvl*)

- `float32xm8_t vfwnmsacvf_float32xm8_float16xm4` (`float32xm8_t a`, `float16_t b`, `float16xm4_t c`, unsigned int `gvl`)
- `float64xm2_t vfwnmsacvf_float64xm2_float32xm1` (`float64xm2_t a`, `float b`, `float32xm1_t c`, unsigned int `gvl`)
- `float64xm4_t vfwnmsacvf_float64xm4_float32xm2` (`float64xm4_t a`, `float b`, `float32xm2_t c`, unsigned int `gvl`)
- `float64xm8_t vfwnmsacvf_float64xm8_float32xm4` (`float64xm8_t a`, `float b`, `float32xm4_t c`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = -(wide_fp(b[element]) * wide_fp(c[element])) + a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float32xm2_t vfwnmsacvf_mask_float32xm2_float16xm1` (`float32xm2_t merge`, `float32xm2_t a`, `float16_t b`, `float16xm1_t c`, `e16xm1_t mask`, unsigned int `gvl`)
- `float32xm4_t vfwnmsacvf_mask_float32xm4_float16xm2` (`float32xm4_t merge`, `float32xm4_t a`, `float16_t b`, `float16xm2_t c`, `e16xm2_t mask`, unsigned int `gvl`)
- `float32xm8_t vfwnmsacvf_mask_float32xm8_float16xm4` (`float32xm8_t merge`, `float32xm8_t a`, `float16_t b`, `float16xm4_t c`, `e16xm4_t mask`, unsigned int `gvl`)
- `float64xm2_t vfwnmsacvf_mask_float64xm2_float32xm1` (`float64xm2_t merge`, `float64xm2_t a`, `float b`, `float32xm1_t c`, `e32xm1_t mask`, unsigned int `gvl`)
- `float64xm4_t vfwnmsacvf_mask_float64xm4_float32xm2` (`float64xm4_t merge`, `float64xm4_t a`, `float b`, `float32xm2_t c`, `e32xm2_t mask`, unsigned int `gvl`)
- `float64xm8_t vfwnmsacvf_mask_float64xm8_float32xm4` (`float64xm8_t merge`, `float64xm8_t a`, `float b`, `float32xm4_t c`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = -(wide_fp(b[element]) * wide_fp(c[element])) + a[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.57 Floating-point vector-vector widening negate multiply and sub

**Instruction:** ['vfnmsacvv']

**Prototypes:**

- `float32xm2_t vfnmsacvv_float32xm2_float16xm1` (`float32xm2_t a`, `float16xm1_t b`, `float16xm1_t c`, unsigned int `gvl`)
- `float32xm4_t vfnmsacvv_float32xm4_float16xm2` (`float32xm4_t a`, `float16xm2_t b`, `float16xm2_t c`, unsigned int `gvl`)
- `float32xm8_t vfnmsacvv_float32xm8_float16xm4` (`float32xm8_t a`, `float16xm4_t b`, `float16xm4_t c`, unsigned int `gvl`)
- `float64xm2_t vfnmsacvv_float64xm2_float32xm1` (`float64xm2_t a`, `float32xm1_t b`, `float32xm1_t c`, unsigned int `gvl`)
- `float64xm4_t vfnmsacvv_float64xm4_float32xm2` (`float64xm4_t a`, `float32xm2_t b`, `float32xm2_t c`, unsigned int `gvl`)
- `float64xm8_t vfnmsacvv_float64xm8_float32xm4` (`float64xm8_t a`, `float32xm4_t b`, `float32xm4_t c`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = -(wide_fp(b[element]) * wide_fp(c[element])) + a[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float32xm2_t vfnmsacvv_mask_float32xm2_float16xm1` (`float32xm2_t merge`, `float32xm2_t a`, `float16xm1_t b`, `float16xm1_t c`, `e16xm1_t mask`, unsigned int `gvl`)
- `float32xm4_t vfnmsacvv_mask_float32xm4_float16xm2` (`float32xm4_t merge`, `float32xm4_t a`, `float16xm2_t b`, `float16xm2_t c`, `e16xm2_t mask`, unsigned int `gvl`)
- `float32xm8_t vfnmsacvv_mask_float32xm8_float16xm4` (`float32xm8_t merge`, `float32xm8_t a`, `float16xm4_t b`, `float16xm4_t c`, `e16xm4_t mask`, unsigned int `gvl`)
- `float64xm2_t vfnmsacvv_mask_float64xm2_float32xm1` (`float64xm2_t merge`, `float64xm2_t a`, `float32xm1_t b`, `float32xm1_t c`, `e32xm1_t mask`, unsigned int `gvl`)
- `float64xm4_t vfnmsacvv_mask_float64xm4_float32xm2` (`float64xm4_t merge`, `float64xm4_t a`, `float32xm2_t b`, `float32xm2_t c`, `e32xm2_t mask`, unsigned int `gvl`)
- `float64xm8_t vfnmsacvv_mask_float64xm8_float32xm4` (`float64xm8_t merge`, `float64xm8_t a`, `float32xm4_t b`, `float32xm4_t c`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = -(wide_fp(b[element]) * wide_fp(c[element])) + a[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.58 Floating-point widening ordered sum of vector

**Instruction:** ['vfwredosum.vs']

**Prototypes:**

- *float32xm2\_t* **vfwredosumvs\_float32xm2\_float16xm1** (*float16xm1\_t* a, *float32xm2\_t* b, unsigned int gvl)
- *float32xm4\_t* **vfwredosumvs\_float32xm4\_float16xm2** (*float16xm2\_t* a, *float32xm4\_t* b, unsigned int gvl)
- *float32xm8\_t* **vfwredosumvs\_float32xm8\_float16xm4** (*float16xm4\_t* a, *float32xm8\_t* b, unsigned int gvl)
- *float64xm2\_t* **vfwredosumvs\_float64xm2\_float32xm1** (*float32xm1\_t* a, *float64xm2\_t* b, unsigned int gvl)
- *float64xm4\_t* **vfwredosumvs\_float64xm4\_float32xm2** (*float32xm2\_t* a, *float64xm4\_t* b, unsigned int gvl)
- *float64xm8\_t* **vfwredosumvs\_float64xm8\_float32xm4** (*float32xm4\_t* a, *float64xm8\_t* b, unsigned int gvl)

**Operation:**

```
>>> result[0] = b[0]
for element = 0 to gvl - 1
    result[0] = result[0] + wide_fp(a[element])
```

**Masked prototypes:**

- *float32xm2\_t* **vfwredosumvs\_mask\_float32xm2\_float16xm1** (*float16xm1\_t* a, *float32xm2\_t* b, *e16xm1\_t* mask, unsigned int gvl)
- *float32xm4\_t* **vfwredosumvs\_mask\_float32xm4\_float16xm2** (*float16xm2\_t* a, *float32xm4\_t* b, *e16xm2\_t* mask, unsigned int gvl)
- *float32xm8\_t* **vfwredosumvs\_mask\_float32xm8\_float16xm4** (*float16xm4\_t* a, *float32xm8\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *float64xm2\_t* **vfwredosumvs\_mask\_float64xm2\_float32xm1** (*float32xm1\_t* a, *float64xm2\_t* b, *e32xm1\_t* mask, unsigned int gvl)
- *float64xm4\_t* **vfwredosumvs\_mask\_float64xm4\_float32xm2** (*float32xm2\_t* a, *float64xm4\_t* b, *e32xm2\_t* mask, unsigned int gvl)
- *float64xm8\_t* **vfwredosumvs\_mask\_float64xm8\_float32xm4** (*float32xm4\_t* a, *float64xm8\_t* b, *e32xm4\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> result[0] = b[0]
    for element = 0 to gvl - 1
        if mask[element]
            result[0] = result[0] + wide_fp(a[element])
```

**2.5.59 Floating-point widening sum of vector****Instruction:** ['vfwredsum.vs']**Prototypes:**

- *float32xm2\_t* **vfwredsumvs\_float32xm2\_float16xm1** (*float16xm1\_t* a, *float32xm2\_t* b, unsigned int gvl)
- *float32xm4\_t* **vfwredsumvs\_float32xm4\_float16xm2** (*float16xm2\_t* a, *float32xm4\_t* b, unsigned int gvl)
- *float32xm8\_t* **vfwredsumvs\_float32xm8\_float16xm4** (*float16xm4\_t* a, *float32xm8\_t* b, unsigned int gvl)
- *float64xm2\_t* **vfwredsumvs\_float64xm2\_float32xm1** (*float32xm1\_t* a, *float64xm2\_t* b, unsigned int gvl)
- *float64xm4\_t* **vfwredsumvs\_float64xm4\_float32xm2** (*float32xm2\_t* a, *float64xm4\_t* b, unsigned int gvl)
- *float64xm8\_t* **vfwredsumvs\_float64xm8\_float32xm4** (*float32xm4\_t* a, *float64xm8\_t* b, unsigned int gvl)

**Operation:**

```
>>> result[0] = b[0]
    for element = 0 to gvl - 1
        result[0] = result[0] + wide_fp(a[element])
```

**Masked prototypes:**

- *float32xm2\_t* **vfwredsumvs\_mask\_float32xm2\_float16xm1** (*float16xm1\_t* a, *float32xm2\_t* b, *e16xm1\_t* mask, unsigned int gvl)
- *float32xm4\_t* **vfwredsumvs\_mask\_float32xm4\_float16xm2** (*float16xm2\_t* a, *float32xm4\_t* b, *e16xm2\_t* mask, unsigned int gvl)
- *float32xm8\_t* **vfwredsumvs\_mask\_float32xm8\_float16xm4** (*float16xm4\_t* a, *float32xm8\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *float64xm2\_t* **vfwredsumvs\_mask\_float64xm2\_float32xm1** (*float32xm1\_t* a, *float64xm2\_t* b, *e32xm1\_t* mask, unsigned int gvl)
- *float64xm4\_t* **vfwredsumvs\_mask\_float64xm4\_float32xm2** (*float32xm2\_t* a, *float64xm4\_t* b, *e32xm2\_t* mask, unsigned int gvl)
- *float64xm8\_t* **vfwredsumvs\_mask\_float64xm8\_float32xm4** (*float32xm4\_t* a, *float64xm8\_t* b, *e32xm4\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> result[0] = b[0]
    for element = 0 to gvl - 1
        if mask[element]
            result[0] = result[0] + wide_fp(a[element])
```

**2.5.60 Floating-point vector-scalar widening subtraction****Instruction:** [`'vfwsbvf'`]**Prototypes:**

- `float32xm2_t vfwsbvf_float32xm2_float16xm1` (`float16xm1_t a`, `float16_t b`, unsigned int `gvl`)
- `float32xm4_t vfwsbvf_float32xm4_float16xm2` (`float16xm2_t a`, `float16_t b`, unsigned int `gvl`)
- `float32xm8_t vfwsbvf_float32xm8_float16xm4` (`float16xm4_t a`, `float16_t b`, unsigned int `gvl`)
- `float64xm2_t vfwsbvf_float64xm2_float32xm1` (`float32xm1_t a`, `float b`, unsigned int `gvl`)
- `float64xm4_t vfwsbvf_float64xm4_float32xm2` (`float32xm2_t a`, `float b`, unsigned int `gvl`)
- `float64xm8_t vfwsbvf_float64xm8_float32xm4` (`float32xm4_t a`, `float b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = wide_fp(a[element]) - wide_fp(b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float32xm2_t vfwsbvf_mask_float32xm2_float16xm1` (`float32xm2_t merge`, `float16xm1_t a`, `float16_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `float32xm4_t vfwsbvf_mask_float32xm4_float16xm2` (`float32xm4_t merge`, `float16xm2_t a`, `float16_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `float32xm8_t vfwsbvf_mask_float32xm8_float16xm4` (`float32xm8_t merge`, `float16xm4_t a`, `float16_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `float64xm2_t vfwsbvf_mask_float64xm2_float32xm1` (`float64xm2_t merge`, `float32xm1_t a`, `float b`, `e32xm1_t mask`, unsigned int `gvl`)
- `float64xm4_t vfwsbvf_mask_float64xm4_float32xm2` (`float64xm4_t merge`, `float32xm2_t a`, `float b`, `e32xm2_t mask`, unsigned int `gvl`)
- `float64xm8_t vfwsbvf_mask_float64xm8_float32xm4` (`float64xm8_t merge`, `float32xm4_t a`, `float b`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = wide_fp(a[element]) - wide_fp(b[element])
```

(continues on next page)



(continued from previous page)

```

else
    result[element] = merge[element]
result[gvl : VLMAX] = 0

```

## 2.5.61 Floating-point vector-vector widening subtraction

**Instruction:** ['vfwsbvv.vv']

**Prototypes:**

- *float32xm2\_t vfwsbvv\_float32xm2\_float16xm1* (*float16xm1\_t a, float16xm1\_t b*, unsigned int gvl)
- *float32xm4\_t vfwsbvv\_float32xm4\_float16xm2* (*float16xm2\_t a, float16xm2\_t b*, unsigned int gvl)
- *float32xm8\_t vfwsbvv\_float32xm8\_float16xm4* (*float16xm4\_t a, float16xm4\_t b*, unsigned int gvl)
- *float64xm2\_t vfwsbvv\_float64xm2\_float32xm1* (*float32xm1\_t a, float32xm1\_t b*, unsigned int gvl)
- *float64xm4\_t vfwsbvv\_float64xm4\_float32xm2* (*float32xm2\_t a, float32xm2\_t b*, unsigned int gvl)
- *float64xm8\_t vfwsbvv\_float64xm8\_float32xm4* (*float32xm4\_t a, float32xm4\_t b*, unsigned int gvl)

**Operation:**

```

>>> for element = 0 to gvl - 1
    result[element] = wide_fp(a[element]) - wide_fp(b[element])
result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- *float32xm2\_t vfwsbvv\_mask\_float32xm2\_float16xm1* (*float32xm2\_t merge, float16xm1\_t a, float16xm1\_t b, e16xm1\_t mask*, unsigned int gvl)
- *float32xm4\_t vfwsbvv\_mask\_float32xm4\_float16xm2* (*float32xm4\_t merge, float16xm2\_t a, float16xm2\_t b, e16xm2\_t mask*, unsigned int gvl)
- *float32xm8\_t vfwsbvv\_mask\_float32xm8\_float16xm4* (*float32xm8\_t merge, float16xm4\_t a, float16xm4\_t b, e16xm4\_t mask*, unsigned int gvl)
- *float64xm2\_t vfwsbvv\_mask\_float64xm2\_float32xm1* (*float64xm2\_t merge, float32xm1\_t a, float32xm1\_t b, e32xm1\_t mask*, unsigned int gvl)
- *float64xm4\_t vfwsbvv\_mask\_float64xm4\_float32xm2* (*float64xm4\_t merge, float32xm2\_t a, float32xm2\_t b, e32xm2\_t mask*, unsigned int gvl)
- *float64xm8\_t vfwsbvv\_mask\_float64xm8\_float32xm4* (*float64xm8\_t merge, float32xm4\_t a, float32xm4\_t b, e32xm4\_t mask*, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = wide_fp(a[element]) - wide_fp(b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.62 Floating-point vector-scalar widening subtraction(second operand)

**Instruction:** ['vfwsbf.wf']

**Prototypes:**

- *float32xm2\_t vfwsbf\_float32xm2* (*float32xm2\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float32xm4\_t vfwsbf\_float32xm4* (*float32xm4\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float32xm8\_t vfwsbf\_float32xm8* (*float32xm8\_t a*, *float16\_t b*, unsigned int *gvl*)
- *float64xm2\_t vfwsbf\_float64xm2* (*float64xm2\_t a*, *float b*, unsigned int *gvl*)
- *float64xm4\_t vfwsbf\_float64xm4* (*float64xm4\_t a*, *float b*, unsigned int *gvl*)
- *float64xm8\_t vfwsbf\_float64xm8* (*float64xm8\_t a*, *float b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = a[element] - wide_fp(b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float32xm2\_t vfwsbf\_mask\_float32xm2* (*float32xm2\_t merge*, *float32xm2\_t a*, *float16\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *float32xm4\_t vfwsbf\_mask\_float32xm4* (*float32xm4\_t merge*, *float32xm4\_t a*, *float16\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *float32xm8\_t vfwsbf\_mask\_float32xm8* (*float32xm8\_t merge*, *float32xm8\_t a*, *float16\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vfwsbf\_mask\_float64xm2* (*float64xm2\_t merge*, *float64xm2\_t a*, *float b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *float64xm4\_t vfwsbf\_mask\_float64xm4* (*float64xm4\_t merge*, *float64xm4\_t a*, *float b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *float64xm8\_t vfwsbf\_mask\_float64xm8* (*float64xm8\_t merge*, *float64xm8\_t a*, *float b*, *e32xm4\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element]
        result[element] = a[element] - wide_fp(b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.5.63 Floating-point vector-vector widening subtraction(second operand)

**Instruction:** ['vfwsbvwv']

**Prototypes:**

- `float32xm2_t vfwsbvwv_float32xm2_float16xm1` (`float32xm2_t a`, `float16xm1_t b`, unsigned int `gvl`)
- `float32xm4_t vfwsbvwv_float32xm4_float16xm2` (`float32xm4_t a`, `float16xm2_t b`, unsigned int `gvl`)
- `float32xm8_t vfwsbvwv_float32xm8_float16xm4` (`float32xm8_t a`, `float16xm4_t b`, unsigned int `gvl`)
- `float64xm2_t vfwsbvwv_float64xm2_float32xm1` (`float64xm2_t a`, `float32xm1_t b`, unsigned int `gvl`)
- `float64xm4_t vfwsbvwv_float64xm4_float32xm2` (`float64xm4_t a`, `float32xm2_t b`, unsigned int `gvl`)
- `float64xm8_t vfwsbvwv_float64xm8_float32xm4` (`float64xm8_t a`, `float32xm4_t b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] - wide_fp(b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float32xm2_t vfwsbvwv_mask_float32xm2_float16xm1` (`float32xm2_t merge`, `float32xm2_t a`, `float16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `float32xm4_t vfwsbvwv_mask_float32xm4_float16xm2` (`float32xm4_t merge`, `float32xm4_t a`, `float16xm2_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `float32xm8_t vfwsbvwv_mask_float32xm8_float16xm4` (`float32xm8_t merge`, `float32xm8_t a`, `float16xm4_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `float64xm2_t vfwsbvwv_mask_float64xm2_float32xm1` (`float64xm2_t merge`, `float64xm2_t a`, `float32xm1_t b`, `e32xm1_t mask`, unsigned int `gvl`)
- `float64xm4_t vfwsbvwv_mask_float64xm4_float32xm2` (`float64xm4_t merge`, `float64xm4_t a`, `float32xm2_t b`, `e32xm2_t mask`, unsigned int `gvl`)
- `float64xm8_t vfwsbvwv_mask_float64xm8_float32xm4` (`float64xm8_t merge`, `float64xm8_t a`, `float32xm4_t b`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element]
        result[element] = a[element] - wide_fp(b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.6 Floating-point relational operations

### 2.6.1 Compare elementwise float vector-scalar for equality

**Instruction:** [`'vmfeqv.f'`]

**Prototypes:**

- `e16xm1_t vmfeqv.f_e16xm1_float16xm1` (`float16xm1_t a`, `float16_t b`, unsigned int `gvl`)
- `e16xm2_t vmfeqv.f_e16xm2_float16xm2` (`float16xm2_t a`, `float16_t b`, unsigned int `gvl`)
- `e16xm4_t vmfeqv.f_e16xm4_float16xm4` (`float16xm4_t a`, `float16_t b`, unsigned int `gvl`)
- `e16xm8_t vmfeqv.f_e16xm8_float16xm8` (`float16xm8_t a`, `float16_t b`, unsigned int `gvl`)
- `e32xm1_t vmfeqv.f_e32xm1_float32xm1` (`float32xm1_t a`, `float b`, unsigned int `gvl`)
- `e32xm2_t vmfeqv.f_e32xm2_float32xm2` (`float32xm2_t a`, `float b`, unsigned int `gvl`)
- `e32xm4_t vmfeqv.f_e32xm4_float32xm4` (`float32xm4_t a`, `float b`, unsigned int `gvl`)
- `e32xm8_t vmfeqv.f_e32xm8_float32xm8` (`float32xm8_t a`, `float b`, unsigned int `gvl`)
- `e64xm1_t vmfeqv.f_e64xm1_float64xm1` (`float64xm1_t a`, `double b`, unsigned int `gvl`)
- `e64xm2_t vmfeqv.f_e64xm2_float64xm2` (`float64xm2_t a`, `double b`, unsigned int `gvl`)
- `e64xm4_t vmfeqv.f_e64xm4_float64xm4` (`float64xm4_t a`, `double b`, unsigned int `gvl`)
- `e64xm8_t vmfeqv.f_e64xm8_float64xm8` (`float64xm8_t a`, `double b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] == b) ? 1 : 0
reuslt[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `e16xm1_t vmfeqv.f_mask_e16xm1_float16xm1` (`e16xm1_t merge`, `float16xm1_t a`, `float16_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `e16xm2_t vmfeqv.f_mask_e16xm2_float16xm2` (`e16xm2_t merge`, `float16xm2_t a`, `float16_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `e16xm4_t vmfeqv.f_mask_e16xm4_float16xm4` (`e16xm4_t merge`, `float16xm4_t a`, `float16_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `e16xm8_t vmfeqv.f_mask_e16xm8_float16xm8` (`e16xm8_t merge`, `float16xm8_t a`, `float16_t b`, `e16xm8_t mask`, unsigned int `gvl`)
- `e32xm1_t vmfeqv.f_mask_e32xm1_float32xm1` (`e32xm1_t merge`, `float32xm1_t a`, `float b`, `e32xm1_t mask`, unsigned int `gvl`)
- `e32xm2_t vmfeqv.f_mask_e32xm2_float32xm2` (`e32xm2_t merge`, `float32xm2_t a`, `float b`, `e32xm2_t mask`, unsigned int `gvl`)
- `e32xm4_t vmfeqv.f_mask_e32xm4_float32xm4` (`e32xm4_t merge`, `float32xm4_t a`, `float b`, `e32xm4_t mask`, unsigned int `gvl`)
- `e32xm8_t vmfeqv.f_mask_e32xm8_float32xm8` (`e32xm8_t merge`, `float32xm8_t a`, `float b`, `e32xm8_t mask`, unsigned int `gvl`)
- `e64xm1_t vmfeqv.f_mask_e64xm1_float64xm1` (`e64xm1_t merge`, `float64xm1_t a`, `double b`, `e64xm1_t mask`, unsigned int `gvl`)

- `e64xm2_t vmfeqvfv_mask_e64xm2_float64xm2` (`e64xm2_t merge`, `float64xm2_t a`, double `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `e64xm4_t vmfeqvfv_mask_e64xm4_float64xm4` (`e64xm4_t merge`, `float64xm4_t a`, double `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `e64xm8_t vmfeqvfv_mask_e64xm8_float64xm8` (`e64xm8_t merge`, `float64xm8_t a`, double `b`, `e64xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[element] == b) ? 1 : 0
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.6.2 Compare elementwise float vector-vector for equality

**Instruction:** ['vmfeqvfv']**Prototypes:**

- `e16xm1_t vmfeqvfv_e16xm1_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, unsigned int `gvl`)
- `e16xm2_t vmfeqvfv_e16xm2_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, unsigned int `gvl`)
- `e16xm4_t vmfeqvfv_e16xm4_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, unsigned int `gvl`)
- `e16xm8_t vmfeqvfv_e16xm8_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, unsigned int `gvl`)
- `e32xm1_t vmfeqvfv_e32xm1_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, unsigned int `gvl`)
- `e32xm2_t vmfeqvfv_e32xm2_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, unsigned int `gvl`)
- `e32xm4_t vmfeqvfv_e32xm4_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, unsigned int `gvl`)
- `e32xm8_t vmfeqvfv_e32xm8_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, unsigned int `gvl`)
- `e64xm1_t vmfeqvfv_e64xm1_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, unsigned int `gvl`)
- `e64xm2_t vmfeqvfv_e64xm2_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, unsigned int `gvl`)
- `e64xm4_t vmfeqvfv_e64xm4_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, unsigned int `gvl`)
- `e64xm8_t vmfeqvfv_e64xm8_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] == b[element]) ? 1 : 0
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `e16xm1_t vmfeqvfv_mask_e16xm1_float16xm1` (`e16xm1_t merge`, `float16xm1_t a`, `float16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `e16xm2_t vmfeqvfv_mask_e16xm2_float16xm2` (`e16xm2_t merge`, `float16xm2_t a`, `float16xm2_t b`, `e16xm2_t mask`, unsigned int `gvl`)

- `e16xm4_t vmfeqv_mask_e16xm4_float16xm4` (`e16xm4_t merge, float16xm4_t a, float16xm4_t b, e16xm4_t mask, unsigned int gvl`)
- `e16xm8_t vmfeqv_mask_e16xm8_float16xm8` (`e16xm8_t merge, float16xm8_t a, float16xm8_t b, e16xm8_t mask, unsigned int gvl`)
- `e32xm1_t vmfeqv_mask_e32xm1_float32xm1` (`e32xm1_t merge, float32xm1_t a, float32xm1_t b, e32xm1_t mask, unsigned int gvl`)
- `e32xm2_t vmfeqv_mask_e32xm2_float32xm2` (`e32xm2_t merge, float32xm2_t a, float32xm2_t b, e32xm2_t mask, unsigned int gvl`)
- `e32xm4_t vmfeqv_mask_e32xm4_float32xm4` (`e32xm4_t merge, float32xm4_t a, float32xm4_t b, e32xm4_t mask, unsigned int gvl`)
- `e32xm8_t vmfeqv_mask_e32xm8_float32xm8` (`e32xm8_t merge, float32xm8_t a, float32xm8_t b, e32xm8_t mask, unsigned int gvl`)
- `e64xm1_t vmfeqv_mask_e64xm1_float64xm1` (`e64xm1_t merge, float64xm1_t a, float64xm1_t b, e64xm1_t mask, unsigned int gvl`)
- `e64xm2_t vmfeqv_mask_e64xm2_float64xm2` (`e64xm2_t merge, float64xm2_t a, float64xm2_t b, e64xm2_t mask, unsigned int gvl`)
- `e64xm4_t vmfeqv_mask_e64xm4_float64xm4` (`e64xm4_t merge, float64xm4_t a, float64xm4_t b, e64xm4_t mask, unsigned int gvl`)
- `e64xm8_t vmfeqv_mask_e64xm8_float64xm8` (`e64xm8_t merge, float64xm8_t a, float64xm8_t b, e64xm8_t mask, unsigned int gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = (a[element] == b[element]) ? 1 : 0
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.6.3 Compare elementwise float vector-scalar for greater-or-equal

Instruction: [`vmfge.vf`]

Prototypes:

- `e16xm1_t vmfgevf_e16xm1_float16xm1` (`float16xm1_t a, float16_t b, unsigned int gvl`)
- `e16xm2_t vmfgevf_e16xm2_float16xm2` (`float16xm2_t a, float16_t b, unsigned int gvl`)
- `e16xm4_t vmfgevf_e16xm4_float16xm4` (`float16xm4_t a, float16_t b, unsigned int gvl`)
- `e16xm8_t vmfgevf_e16xm8_float16xm8` (`float16xm8_t a, float16_t b, unsigned int gvl`)



- *e32xm1\_t* `vmfgevf_e32xm1_float32xm1` (*float32xm1\_t* *a*, float *b*, unsigned int *gvl*)
- *e32xm2\_t* `vmfgevf_e32xm2_float32xm2` (*float32xm2\_t* *a*, float *b*, unsigned int *gvl*)
- *e32xm4\_t* `vmfgevf_e32xm4_float32xm4` (*float32xm4\_t* *a*, float *b*, unsigned int *gvl*)
- *e32xm8\_t* `vmfgevf_e32xm8_float32xm8` (*float32xm8\_t* *a*, float *b*, unsigned int *gvl*)
- *e64xm1\_t* `vmfgevf_e64xm1_float64xm1` (*float64xm1\_t* *a*, double *b*, unsigned int *gvl*)
- *e64xm2\_t* `vmfgevf_e64xm2_float64xm2` (*float64xm2\_t* *a*, double *b*, unsigned int *gvl*)
- *e64xm4\_t* `vmfgevf_e64xm4_float64xm4` (*float64xm4\_t* *a*, double *b*, unsigned int *gvl*)
- *e64xm8\_t* `vmfgevf_e64xm8_float64xm8` (*float64xm8\_t* *a*, double *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] >= b) ? 1 : 0
reuslt[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t* `vmfgevf_mask_e16xm1_float16xm1` (*e16xm1\_t* *merge*, *float16xm1\_t* *a*, float16\_t *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *e16xm2\_t* `vmfgevf_mask_e16xm2_float16xm2` (*e16xm2\_t* *merge*, *float16xm2\_t* *a*, float16\_t *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *e16xm4\_t* `vmfgevf_mask_e16xm4_float16xm4` (*e16xm4\_t* *merge*, *float16xm4\_t* *a*, float16\_t *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *e16xm8\_t* `vmfgevf_mask_e16xm8_float16xm8` (*e16xm8\_t* *merge*, *float16xm8\_t* *a*, float16\_t *b*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- *e32xm1\_t* `vmfgevf_mask_e32xm1_float32xm1` (*e32xm1\_t* *merge*, *float32xm1\_t* *a*, float *b*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *e32xm2\_t* `vmfgevf_mask_e32xm2_float32xm2` (*e32xm2\_t* *merge*, *float32xm2\_t* *a*, float *b*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *e32xm4\_t* `vmfgevf_mask_e32xm4_float32xm4` (*e32xm4\_t* *merge*, *float32xm4\_t* *a*, float *b*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *e32xm8\_t* `vmfgevf_mask_e32xm8_float32xm8` (*e32xm8\_t* *merge*, *float32xm8\_t* *a*, float *b*, *e32xm8\_t* *mask*, unsigned int *gvl*)
- *e64xm1\_t* `vmfgevf_mask_e64xm1_float64xm1` (*e64xm1\_t* *merge*, *float64xm1\_t* *a*, double *b*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *e64xm2\_t* `vmfgevf_mask_e64xm2_float64xm2` (*e64xm2\_t* *merge*, *float64xm2\_t* *a*, double *b*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *e64xm4\_t* `vmfgevf_mask_e64xm4_float64xm4` (*e64xm4\_t* *merge*, *float64xm4\_t* *a*, double *b*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *e64xm8\_t* `vmfgevf_mask_e64xm8_float64xm8` (*e64xm8\_t* *merge*, *float64xm8\_t* *a*, double *b*, *e64xm8\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[element] >= b) ? 1 : 0
      else
```

(continues on next page)

(continued from previous page)

```
result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.6.4 Compare elementwise float vector-vector for greater-or-equal

**Instruction:** ['vmfge.vv']

**Prototypes:**

- *e16xm1\_t vmfgevv\_e16xm1\_float16xm1* (*float16xm1\_t a, float16xm1\_t b*, unsigned int *gvl*)
- *e16xm2\_t vmfgevv\_e16xm2\_float16xm2* (*float16xm2\_t a, float16xm2\_t b*, unsigned int *gvl*)
- *e16xm4\_t vmfgevv\_e16xm4\_float16xm4* (*float16xm4\_t a, float16xm4\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmfgevv\_e16xm8\_float16xm8* (*float16xm8\_t a, float16xm8\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmfgevv\_e32xm1\_float32xm1* (*float32xm1\_t a, float32xm1\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmfgevv\_e32xm2\_float32xm2* (*float32xm2\_t a, float32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmfgevv\_e32xm4\_float32xm4* (*float32xm4\_t a, float32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmfgevv\_e32xm8\_float32xm8* (*float32xm8\_t a, float32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmfgevv\_e64xm1\_float64xm1* (*float64xm1\_t a, float64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmfgevv\_e64xm2\_float64xm2* (*float64xm2\_t a, float64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmfgevv\_e64xm4\_float64xm4* (*float64xm4\_t a, float64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmfgevv\_e64xm8\_float64xm8* (*float64xm8\_t a, float64xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = (a[element] >= b[element]) ? 1 : 0
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmfgevv\_mask\_e16xm1\_float16xm1* (*e16xm1\_t merge, float16xm1\_t a, float16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmfgevv\_mask\_e16xm2\_float16xm2* (*e16xm2\_t merge, float16xm2\_t a, float16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmfgevv\_mask\_e16xm4\_float16xm4* (*e16xm4\_t merge, float16xm4\_t a, float16xm4\_t b, e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmfgevv\_mask\_e16xm8\_float16xm8* (*e16xm8\_t merge, float16xm8\_t a, float16xm8\_t b, e16xm8\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmfgevv\_mask\_e32xm1\_float32xm1* (*e32xm1\_t merge, float32xm1\_t a, float32xm1\_t b, e32xm1\_t mask*, unsigned int *gvl*)

- `e32xm2_t vmfgevv_mask_e32xm2_float32xm2` (`e32xm2_t merge, float32xm2_t a, float32xm2_t b, e32xm2_t mask, unsigned int gvl`)
- `e32xm4_t vmfgevv_mask_e32xm4_float32xm4` (`e32xm4_t merge, float32xm4_t a, float32xm4_t b, e32xm4_t mask, unsigned int gvl`)
- `e32xm8_t vmfgevv_mask_e32xm8_float32xm8` (`e32xm8_t merge, float32xm8_t a, float32xm8_t b, e32xm8_t mask, unsigned int gvl`)
- `e64xm1_t vmfgevv_mask_e64xm1_float64xm1` (`e64xm1_t merge, float64xm1_t a, float64xm1_t b, e64xm1_t mask, unsigned int gvl`)
- `e64xm2_t vmfgevv_mask_e64xm2_float64xm2` (`e64xm2_t merge, float64xm2_t a, float64xm2_t b, e64xm2_t mask, unsigned int gvl`)
- `e64xm4_t vmfgevv_mask_e64xm4_float64xm4` (`e64xm4_t merge, float64xm4_t a, float64xm4_t b, e64xm4_t mask, unsigned int gvl`)
- `e64xm8_t vmfgevv_mask_e64xm8_float64xm8` (`e64xm8_t merge, float64xm8_t a, float64xm8_t b, e64xm8_t mask, unsigned int gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = (a[element] >= b[element]) ? 1 : 0
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.6.5 Compare elementwise float vector-scalar for greater-than

Instruction: ['vmfgt.vf']

Prototypes:

- `e16xm1_t vmfgtvf_e16xm1_float16xm1` (`float16xm1_t a, float16_t b, unsigned int gvl`)
- `e16xm2_t vmfgtvf_e16xm2_float16xm2` (`float16xm2_t a, float16_t b, unsigned int gvl`)
- `e16xm4_t vmfgtvf_e16xm4_float16xm4` (`float16xm4_t a, float16_t b, unsigned int gvl`)
- `e16xm8_t vmfgtvf_e16xm8_float16xm8` (`float16xm8_t a, float16_t b, unsigned int gvl`)
- `e32xm1_t vmfgtvf_e32xm1_float32xm1` (`float32xm1_t a, float b, unsigned int gvl`)
- `e32xm2_t vmfgtvf_e32xm2_float32xm2` (`float32xm2_t a, float b, unsigned int gvl`)
- `e32xm4_t vmfgtvf_e32xm4_float32xm4` (`float32xm4_t a, float b, unsigned int gvl`)
- `e32xm8_t vmfgtvf_e32xm8_float32xm8` (`float32xm8_t a, float b, unsigned int gvl`)
- `e64xm1_t vmfgtvf_e64xm1_float64xm1` (`float64xm1_t a, double b, unsigned int gvl`)
- `e64xm2_t vmfgtvf_e64xm2_float64xm2` (`float64xm2_t a, double b, unsigned int gvl`)
- `e64xm4_t vmfgtvf_e64xm4_float64xm4` (`float64xm4_t a, double b, unsigned int gvl`)

- *e64xm8\_t vmfgtvmf\_e64xm8\_float64xm8* (*float64xm8\_t a*, double *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = (a[element] > b) ? 1 : 0
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmfgtvmf\_mask\_e16xm1\_float16xm1* (*e16xm1\_t merge*, *float16xm1\_t a*, *float16\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmfgtvmf\_mask\_e16xm2\_float16xm2* (*e16xm2\_t merge*, *float16xm2\_t a*, *float16\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmfgtvmf\_mask\_e16xm4\_float16xm4* (*e16xm4\_t merge*, *float16xm4\_t a*, *float16\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmfgtvmf\_mask\_e16xm8\_float16xm8* (*e16xm8\_t merge*, *float16xm8\_t a*, *float16\_t b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmfgtvmf\_mask\_e32xm1\_float32xm1* (*e32xm1\_t merge*, *float32xm1\_t a*, *float b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm2\_t vmfgtvmf\_mask\_e32xm2\_float32xm2* (*e32xm2\_t merge*, *float32xm2\_t a*, *float b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *e32xm4\_t vmfgtvmf\_mask\_e32xm4\_float32xm4* (*e32xm4\_t merge*, *float32xm4\_t a*, *float b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *e32xm8\_t vmfgtvmf\_mask\_e32xm8\_float32xm8* (*e32xm8\_t merge*, *float32xm8\_t a*, *float b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *e64xm1\_t vmfgtvmf\_mask\_e64xm1\_float64xm1* (*e64xm1\_t merge*, *float64xm1\_t a*, double *b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *e64xm2\_t vmfgtvmf\_mask\_e64xm2\_float64xm2* (*e64xm2\_t merge*, *float64xm2\_t a*, double *b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *e64xm4\_t vmfgtvmf\_mask\_e64xm4\_float64xm4* (*e64xm4\_t merge*, *float64xm4\_t a*, double *b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *e64xm8\_t vmfgtvmf\_mask\_e64xm8\_float64xm8* (*e64xm8\_t merge*, *float64xm8\_t a*, double *b*, *e64xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = (a[element] > b) ? 1 : 0
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.6.6 Compare elementwise float vector-vector for greater-than

**Instruction:** ['vmfgt.vv']**Prototypes:**

- *e16xm1\_t vmfgtvmf\_e16xm1\_float16xm1* (*float16xm1\_t a*, *float16xm1\_t b*, unsigned int *gvl*)
- *e16xm2\_t vmfgtvmf\_e16xm2\_float16xm2* (*float16xm2\_t a*, *float16xm2\_t b*, unsigned int *gvl*)

- *e16xm4\_t* **vmfgtvv\_e16xm4\_float16xm4** (*float16xm4\_t* *a*, *float16xm4\_t* *b*, unsigned int *gvl*)
- *e16xm8\_t* **vmfgtvv\_e16xm8\_float16xm8** (*float16xm8\_t* *a*, *float16xm8\_t* *b*, unsigned int *gvl*)
- *e32xm1\_t* **vmfgtvv\_e32xm1\_float32xm1** (*float32xm1\_t* *a*, *float32xm1\_t* *b*, unsigned int *gvl*)
- *e32xm2\_t* **vmfgtvv\_e32xm2\_float32xm2** (*float32xm2\_t* *a*, *float32xm2\_t* *b*, unsigned int *gvl*)
- *e32xm4\_t* **vmfgtvv\_e32xm4\_float32xm4** (*float32xm4\_t* *a*, *float32xm4\_t* *b*, unsigned int *gvl*)
- *e32xm8\_t* **vmfgtvv\_e32xm8\_float32xm8** (*float32xm8\_t* *a*, *float32xm8\_t* *b*, unsigned int *gvl*)
- *e64xm1\_t* **vmfgtvv\_e64xm1\_float64xm1** (*float64xm1\_t* *a*, *float64xm1\_t* *b*, unsigned int *gvl*)
- *e64xm2\_t* **vmfgtvv\_e64xm2\_float64xm2** (*float64xm2\_t* *a*, *float64xm2\_t* *b*, unsigned int *gvl*)
- *e64xm4\_t* **vmfgtvv\_e64xm4\_float64xm4** (*float64xm4\_t* *a*, *float64xm4\_t* *b*, unsigned int *gvl*)
- *e64xm8\_t* **vmfgtvv\_e64xm8\_float64xm8** (*float64xm8\_t* *a*, *float64xm8\_t* *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] > b[element]) ? 1 : 0
reuslt[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t* **vmfgtvv\_mask\_e16xm1\_float16xm1** (*e16xm1\_t* *merge*, *float16xm1\_t* *a*, *float16xm1\_t* *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *e16xm2\_t* **vmfgtvv\_mask\_e16xm2\_float16xm2** (*e16xm2\_t* *merge*, *float16xm2\_t* *a*, *float16xm2\_t* *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *e16xm4\_t* **vmfgtvv\_mask\_e16xm4\_float16xm4** (*e16xm4\_t* *merge*, *float16xm4\_t* *a*, *float16xm4\_t* *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *e16xm8\_t* **vmfgtvv\_mask\_e16xm8\_float16xm8** (*e16xm8\_t* *merge*, *float16xm8\_t* *a*, *float16xm8\_t* *b*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- *e32xm1\_t* **vmfgtvv\_mask\_e32xm1\_float32xm1** (*e32xm1\_t* *merge*, *float32xm1\_t* *a*, *float32xm1\_t* *b*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *e32xm2\_t* **vmfgtvv\_mask\_e32xm2\_float32xm2** (*e32xm2\_t* *merge*, *float32xm2\_t* *a*, *float32xm2\_t* *b*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *e32xm4\_t* **vmfgtvv\_mask\_e32xm4\_float32xm4** (*e32xm4\_t* *merge*, *float32xm4\_t* *a*, *float32xm4\_t* *b*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *e32xm8\_t* **vmfgtvv\_mask\_e32xm8\_float32xm8** (*e32xm8\_t* *merge*, *float32xm8\_t* *a*, *float32xm8\_t* *b*, *e32xm8\_t* *mask*, unsigned int *gvl*)
- *e64xm1\_t* **vmfgtvv\_mask\_e64xm1\_float64xm1** (*e64xm1\_t* *merge*, *float64xm1\_t* *a*, *float64xm1\_t* *b*, *e64xm1\_t* *mask*, unsigned int *gvl*)

- *e64xm2\_t* `vmfgtvv_mask_e64xm2_float64xm2` (*e64xm2\_t* merge, *float64xm2\_t* *a*, *float64xm2\_t* *b*, *e64xm2\_t* mask, unsigned int *gvl*)
- *e64xm4\_t* `vmfgtvv_mask_e64xm4_float64xm4` (*e64xm4\_t* merge, *float64xm4\_t* *a*, *float64xm4\_t* *b*, *e64xm4\_t* mask, unsigned int *gvl*)
- *e64xm8\_t* `vmfgtvv_mask_e64xm8_float64xm8` (*e64xm8\_t* merge, *float64xm8\_t* *a*, *float64xm8\_t* *b*, *e64xm8\_t* mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = (a[element] > b[element]) ? 1 : 0
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.6.7 Compare elementwise float vector-scalar for lower-or-equal****Instruction:** ['vmfle.vf']**Prototypes:**

- *e16xm1\_t* `vmflevf_e16xm1_float16xm1` (*float16xm1\_t* *a*, *float16\_t* *b*, unsigned int *gvl*)
- *e16xm2\_t* `vmflevf_e16xm2_float16xm2` (*float16xm2\_t* *a*, *float16\_t* *b*, unsigned int *gvl*)
- *e16xm4\_t* `vmflevf_e16xm4_float16xm4` (*float16xm4\_t* *a*, *float16\_t* *b*, unsigned int *gvl*)
- *e16xm8\_t* `vmflevf_e16xm8_float16xm8` (*float16xm8\_t* *a*, *float16\_t* *b*, unsigned int *gvl*)
- *e32xm1\_t* `vmflevf_e32xm1_float32xm1` (*float32xm1\_t* *a*, *float* *b*, unsigned int *gvl*)
- *e32xm2\_t* `vmflevf_e32xm2_float32xm2` (*float32xm2\_t* *a*, *float* *b*, unsigned int *gvl*)
- *e32xm4\_t* `vmflevf_e32xm4_float32xm4` (*float32xm4\_t* *a*, *float* *b*, unsigned int *gvl*)
- *e32xm8\_t* `vmflevf_e32xm8_float32xm8` (*float32xm8\_t* *a*, *float* *b*, unsigned int *gvl*)
- *e64xm1\_t* `vmflevf_e64xm1_float64xm1` (*float64xm1\_t* *a*, *double* *b*, unsigned int *gvl*)
- *e64xm2\_t* `vmflevf_e64xm2_float64xm2` (*float64xm2\_t* *a*, *double* *b*, unsigned int *gvl*)
- *e64xm4\_t* `vmflevf_e64xm4_float64xm4` (*float64xm4\_t* *a*, *double* *b*, unsigned int *gvl*)
- *e64xm8\_t* `vmflevf_e64xm8_float64xm8` (*float64xm8\_t* *a*, *double* *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = (a[element] <= b) ? 1 : 0
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t* `vmflevf_mask_e16xm1_float16xm1` (*e16xm1\_t* merge, *float16xm1\_t* *a*, *float16\_t* *b*, *e16xm1\_t* mask, unsigned int *gvl*)
- *e16xm2\_t* `vmflevf_mask_e16xm2_float16xm2` (*e16xm2\_t* merge, *float16xm2\_t* *a*, *float16\_t* *b*, *e16xm2\_t* mask, unsigned int *gvl*)



- `e16xm4_t vmflevf_mask_e16xm4_float16xm4` (`e16xm4_t merge`, `float16xm4_t a`, `float16_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `e16xm8_t vmflevf_mask_e16xm8_float16xm8` (`e16xm8_t merge`, `float16xm8_t a`, `float16_t b`, `e16xm8_t mask`, unsigned int `gvl`)
- `e32xm1_t vmflevf_mask_e32xm1_float32xm1` (`e32xm1_t merge`, `float32xm1_t a`, `float b`, `e32xm1_t mask`, unsigned int `gvl`)
- `e32xm2_t vmflevf_mask_e32xm2_float32xm2` (`e32xm2_t merge`, `float32xm2_t a`, `float b`, `e32xm2_t mask`, unsigned int `gvl`)
- `e32xm4_t vmflevf_mask_e32xm4_float32xm4` (`e32xm4_t merge`, `float32xm4_t a`, `float b`, `e32xm4_t mask`, unsigned int `gvl`)
- `e32xm8_t vmflevf_mask_e32xm8_float32xm8` (`e32xm8_t merge`, `float32xm8_t a`, `float b`, `e32xm8_t mask`, unsigned int `gvl`)
- `e64xm1_t vmflevf_mask_e64xm1_float64xm1` (`e64xm1_t merge`, `float64xm1_t a`, `double b`, `e64xm1_t mask`, unsigned int `gvl`)
- `e64xm2_t vmflevf_mask_e64xm2_float64xm2` (`e64xm2_t merge`, `float64xm2_t a`, `double b`, `e64xm2_t mask`, unsigned int `gvl`)
- `e64xm4_t vmflevf_mask_e64xm4_float64xm4` (`e64xm4_t merge`, `float64xm4_t a`, `double b`, `e64xm4_t mask`, unsigned int `gvl`)
- `e64xm8_t vmflevf_mask_e64xm8_float64xm8` (`e64xm8_t merge`, `float64xm8_t a`, `double b`, `e64xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[element] =< b) ? 1 : 0
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.6.8 Compare elementwise float vector-vector for lower-or-equal

Instruction: [`'vmfle.vv'`]

Prototypes:

- `e16xm1_t vmflevv_e16xm1_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, unsigned int `gvl`)
- `e16xm2_t vmflevv_e16xm2_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, unsigned int `gvl`)
- `e16xm4_t vmflevv_e16xm4_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, unsigned int `gvl`)
- `e16xm8_t vmflevv_e16xm8_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, unsigned int `gvl`)
- `e32xm1_t vmflevv_e32xm1_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, unsigned int `gvl`)
- `e32xm2_t vmflevv_e32xm2_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, unsigned int `gvl`)
- `e32xm4_t vmflevv_e32xm4_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, unsigned int `gvl`)
- `e32xm8_t vmflevv_e32xm8_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, unsigned int `gvl`)
- `e64xm1_t vmflevv_e64xm1_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, unsigned int `gvl`)
- `e64xm2_t vmflevv_e64xm2_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, unsigned int `gvl`)
- `e64xm4_t vmflevv_e64xm4_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, unsigned int `gvl`)

- *e64xm8\_t vmflevv\_e64xm8\_float64xm8* (*float64xm8\_t a, float64xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] =< b[element]) ? 1 : 0
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmflevv\_mask\_e16xm1\_float16xm1* (*e16xm1\_t merge, float16xm1\_t a, float16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmflevv\_mask\_e16xm2\_float16xm2* (*e16xm2\_t merge, float16xm2\_t a, float16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmflevv\_mask\_e16xm4\_float16xm4* (*e16xm4\_t merge, float16xm4\_t a, float16xm4\_t b, e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmflevv\_mask\_e16xm8\_float16xm8* (*e16xm8\_t merge, float16xm8\_t a, float16xm8\_t b, e16xm8\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmflevv\_mask\_e32xm1\_float32xm1* (*e32xm1\_t merge, float32xm1\_t a, float32xm1\_t b, e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm2\_t vmflevv\_mask\_e32xm2\_float32xm2* (*e32xm2\_t merge, float32xm2\_t a, float32xm2\_t b, e32xm2\_t mask*, unsigned int *gvl*)
- *e32xm4\_t vmflevv\_mask\_e32xm4\_float32xm4* (*e32xm4\_t merge, float32xm4\_t a, float32xm4\_t b, e32xm4\_t mask*, unsigned int *gvl*)
- *e32xm8\_t vmflevv\_mask\_e32xm8\_float32xm8* (*e32xm8\_t merge, float32xm8\_t a, float32xm8\_t b, e32xm8\_t mask*, unsigned int *gvl*)
- *e64xm1\_t vmflevv\_mask\_e64xm1\_float64xm1* (*e64xm1\_t merge, float64xm1\_t a, float64xm1\_t b, e64xm1\_t mask*, unsigned int *gvl*)
- *e64xm2\_t vmflevv\_mask\_e64xm2\_float64xm2* (*e64xm2\_t merge, float64xm2\_t a, float64xm2\_t b, e64xm2\_t mask*, unsigned int *gvl*)
- *e64xm4\_t vmflevv\_mask\_e64xm4\_float64xm4* (*e64xm4\_t merge, float64xm4\_t a, float64xm4\_t b, e64xm4\_t mask*, unsigned int *gvl*)
- *e64xm8\_t vmflevv\_mask\_e64xm8\_float64xm8* (*e64xm8\_t merge, float64xm8\_t a, float64xm8\_t b, e64xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[element] =< b[element]) ? 1 : 0
```

(continues on next page)

(continued from previous page)

```

else
    result[element] = merge[element]
result[gvl : VLMAX] = 0

```

## 2.6.9 Compare elementwise float vector-scalar for lower-than

**Instruction:** [`vmflt.vf`]

**Prototypes:**

- *e16xm1\_t vmflt.vf\_e16xm1\_float16xm1* (*float16xm1\_t a*, *float16\_t b*, unsigned int *gvl*)
- *e16xm2\_t vmflt.vf\_e16xm2\_float16xm2* (*float16xm2\_t a*, *float16\_t b*, unsigned int *gvl*)
- *e16xm4\_t vmflt.vf\_e16xm4\_float16xm4* (*float16xm4\_t a*, *float16\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmflt.vf\_e16xm8\_float16xm8* (*float16xm8\_t a*, *float16\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmflt.vf\_e32xm1\_float32xm1* (*float32xm1\_t a*, *float b*, unsigned int *gvl*)
- *e32xm2\_t vmflt.vf\_e32xm2\_float32xm2* (*float32xm2\_t a*, *float b*, unsigned int *gvl*)
- *e32xm4\_t vmflt.vf\_e32xm4\_float32xm4* (*float32xm4\_t a*, *float b*, unsigned int *gvl*)
- *e32xm8\_t vmflt.vf\_e32xm8\_float32xm8* (*float32xm8\_t a*, *float b*, unsigned int *gvl*)
- *e64xm1\_t vmflt.vf\_e64xm1\_float64xm1* (*float64xm1\_t a*, *double b*, unsigned int *gvl*)
- *e64xm2\_t vmflt.vf\_e64xm2\_float64xm2* (*float64xm2\_t a*, *double b*, unsigned int *gvl*)
- *e64xm4\_t vmflt.vf\_e64xm4\_float64xm4* (*float64xm4\_t a*, *double b*, unsigned int *gvl*)
- *e64xm8\_t vmflt.vf\_e64xm8\_float64xm8* (*float64xm8\_t a*, *double b*, unsigned int *gvl*)

**Operation:**

```

>>> for element = 0 to gvl - 1
    result[element] = (a[element] < b) ? 1 : 0
result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- *e16xm1\_t vmflt.vf\_mask\_e16xm1\_float16xm1* (*e16xm1\_t merge*, *float16xm1\_t a*, *float16\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmflt.vf\_mask\_e16xm2\_float16xm2* (*e16xm2\_t merge*, *float16xm2\_t a*, *float16\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmflt.vf\_mask\_e16xm4\_float16xm4* (*e16xm4\_t merge*, *float16xm4\_t a*, *float16\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmflt.vf\_mask\_e16xm8\_float16xm8* (*e16xm8\_t merge*, *float16xm8\_t a*, *float16\_t b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmflt.vf\_mask\_e32xm1\_float32xm1* (*e32xm1\_t merge*, *float32xm1\_t a*, *float b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm2\_t vmflt.vf\_mask\_e32xm2\_float32xm2* (*e32xm2\_t merge*, *float32xm2\_t a*, *float b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *e32xm4\_t vmflt.vf\_mask\_e32xm4\_float32xm4* (*e32xm4\_t merge*, *float32xm4\_t a*, *float b*, *e32xm4\_t mask*, unsigned int *gvl*)

- `e32xm8_t vmfltvmf_mask_e32xm8_float32xm8` (`e32xm8_t merge`, `float32xm8_t a`, `float b`, `e32xm8_t mask`, unsigned int `gvl`)
- `e64xm1_t vmfltvmf_mask_e64xm1_float64xm1` (`e64xm1_t merge`, `float64xm1_t a`, `double b`, `e64xm1_t mask`, unsigned int `gvl`)
- `e64xm2_t vmfltvmf_mask_e64xm2_float64xm2` (`e64xm2_t merge`, `float64xm2_t a`, `double b`, `e64xm2_t mask`, unsigned int `gvl`)
- `e64xm4_t vmfltvmf_mask_e64xm4_float64xm4` (`e64xm4_t merge`, `float64xm4_t a`, `double b`, `e64xm4_t mask`, unsigned int `gvl`)
- `e64xm8_t vmfltvmf_mask_e64xm8_float64xm8` (`e64xm8_t merge`, `float64xm8_t a`, `double b`, `e64xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[element] < b) ? 1 : 0
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.6.10 Compare elementwise float vector-vector for lower-than****Instruction:** [`'vmflt.vv'`]**Prototypes:**

- `e16xm1_t vmflttvv_e16xm1_float16xm1` (`float16xm1_t a`, `float16xm1_t b`, unsigned int `gvl`)
- `e16xm2_t vmflttvv_e16xm2_float16xm2` (`float16xm2_t a`, `float16xm2_t b`, unsigned int `gvl`)
- `e16xm4_t vmflttvv_e16xm4_float16xm4` (`float16xm4_t a`, `float16xm4_t b`, unsigned int `gvl`)
- `e16xm8_t vmflttvv_e16xm8_float16xm8` (`float16xm8_t a`, `float16xm8_t b`, unsigned int `gvl`)
- `e32xm1_t vmflttvv_e32xm1_float32xm1` (`float32xm1_t a`, `float32xm1_t b`, unsigned int `gvl`)
- `e32xm2_t vmflttvv_e32xm2_float32xm2` (`float32xm2_t a`, `float32xm2_t b`, unsigned int `gvl`)
- `e32xm4_t vmflttvv_e32xm4_float32xm4` (`float32xm4_t a`, `float32xm4_t b`, unsigned int `gvl`)
- `e32xm8_t vmflttvv_e32xm8_float32xm8` (`float32xm8_t a`, `float32xm8_t b`, unsigned int `gvl`)
- `e64xm1_t vmflttvv_e64xm1_float64xm1` (`float64xm1_t a`, `float64xm1_t b`, unsigned int `gvl`)
- `e64xm2_t vmflttvv_e64xm2_float64xm2` (`float64xm2_t a`, `float64xm2_t b`, unsigned int `gvl`)
- `e64xm4_t vmflttvv_e64xm4_float64xm4` (`float64xm4_t a`, `float64xm4_t b`, unsigned int `gvl`)
- `e64xm8_t vmflttvv_e64xm8_float64xm8` (`float64xm8_t a`, `float64xm8_t b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] < b[element]) ? 1 : 0
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `e16xm1_t vmflttvv_mask_e16xm1_float16xm1` (`e16xm1_t merge`, `float16xm1_t a`, `float16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)

- `e16xm2_t vmfltvv_mask_e16xm2_float16xm2` (`e16xm2_t` merge, `float16xm2_t` *a*, `float16xm2_t` *b*, `e16xm2_t` mask, unsigned int *gvl*)
- `e16xm4_t vmfltvv_mask_e16xm4_float16xm4` (`e16xm4_t` merge, `float16xm4_t` *a*, `float16xm4_t` *b*, `e16xm4_t` mask, unsigned int *gvl*)
- `e16xm8_t vmfltvv_mask_e16xm8_float16xm8` (`e16xm8_t` merge, `float16xm8_t` *a*, `float16xm8_t` *b*, `e16xm8_t` mask, unsigned int *gvl*)
- `e32xm1_t vmfltvv_mask_e32xm1_float32xm1` (`e32xm1_t` merge, `float32xm1_t` *a*, `float32xm1_t` *b*, `e32xm1_t` mask, unsigned int *gvl*)
- `e32xm2_t vmfltvv_mask_e32xm2_float32xm2` (`e32xm2_t` merge, `float32xm2_t` *a*, `float32xm2_t` *b*, `e32xm2_t` mask, unsigned int *gvl*)
- `e32xm4_t vmfltvv_mask_e32xm4_float32xm4` (`e32xm4_t` merge, `float32xm4_t` *a*, `float32xm4_t` *b*, `e32xm4_t` mask, unsigned int *gvl*)
- `e32xm8_t vmfltvv_mask_e32xm8_float32xm8` (`e32xm8_t` merge, `float32xm8_t` *a*, `float32xm8_t` *b*, `e32xm8_t` mask, unsigned int *gvl*)
- `e64xm1_t vmfltvv_mask_e64xm1_float64xm1` (`e64xm1_t` merge, `float64xm1_t` *a*, `float64xm1_t` *b*, `e64xm1_t` mask, unsigned int *gvl*)
- `e64xm2_t vmfltvv_mask_e64xm2_float64xm2` (`e64xm2_t` merge, `float64xm2_t` *a*, `float64xm2_t` *b*, `e64xm2_t` mask, unsigned int *gvl*)
- `e64xm4_t vmfltvv_mask_e64xm4_float64xm4` (`e64xm4_t` merge, `float64xm4_t` *a*, `float64xm4_t` *b*, `e64xm4_t` mask, unsigned int *gvl*)
- `e64xm8_t vmfltvv_mask_e64xm8_float64xm8` (`e64xm8_t` merge, `float64xm8_t` *a*, `float64xm8_t` *b*, `e64xm8_t` mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = (a[element] < b[element]) ? 1 : 0
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.6.11 Compare elementwise float vector-scalar for inequality****Instruction:** [`'vmfnevf'`]**Prototypes:**

- `e16xm1_t vmfnevf_e16xm1_float16xm1` (`float16xm1_t` *a*, `float16_t` *b*, unsigned int *gvl*)
- `e16xm2_t vmfnevf_e16xm2_float16xm2` (`float16xm2_t` *a*, `float16_t` *b*, unsigned int *gvl*)

- *e16xm4\_t* `vmfnevf_e16xm4_float16xm4` (*float16xm4\_t* *a*, *float16\_t* *b*, unsigned int *gvl*)
- *e16xm8\_t* `vmfnevf_e16xm8_float16xm8` (*float16xm8\_t* *a*, *float16\_t* *b*, unsigned int *gvl*)
- *e32xm1\_t* `vmfnevf_e32xm1_float32xm1` (*float32xm1\_t* *a*, *float b*, unsigned int *gvl*)
- *e32xm2\_t* `vmfnevf_e32xm2_float32xm2` (*float32xm2\_t* *a*, *float b*, unsigned int *gvl*)
- *e32xm4\_t* `vmfnevf_e32xm4_float32xm4` (*float32xm4\_t* *a*, *float b*, unsigned int *gvl*)
- *e32xm8\_t* `vmfnevf_e32xm8_float32xm8` (*float32xm8\_t* *a*, *float b*, unsigned int *gvl*)
- *e64xm1\_t* `vmfnevf_e64xm1_float64xm1` (*float64xm1\_t* *a*, *double b*, unsigned int *gvl*)
- *e64xm2\_t* `vmfnevf_e64xm2_float64xm2` (*float64xm2\_t* *a*, *double b*, unsigned int *gvl*)
- *e64xm4\_t* `vmfnevf_e64xm4_float64xm4` (*float64xm4\_t* *a*, *double b*, unsigned int *gvl*)
- *e64xm8\_t* `vmfnevf_e64xm8_float64xm8` (*float64xm8\_t* *a*, *double b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] != b) ? 1 : 0
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t* `vmfnevf_mask_e16xm1_float16xm1` (*e16xm1\_t* *merge*, *float16xm1\_t* *a*, *float16\_t* *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *e16xm2\_t* `vmfnevf_mask_e16xm2_float16xm2` (*e16xm2\_t* *merge*, *float16xm2\_t* *a*, *float16\_t* *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *e16xm4\_t* `vmfnevf_mask_e16xm4_float16xm4` (*e16xm4\_t* *merge*, *float16xm4\_t* *a*, *float16\_t* *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *e16xm8\_t* `vmfnevf_mask_e16xm8_float16xm8` (*e16xm8\_t* *merge*, *float16xm8\_t* *a*, *float16\_t* *b*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- *e32xm1\_t* `vmfnevf_mask_e32xm1_float32xm1` (*e32xm1\_t* *merge*, *float32xm1\_t* *a*, *float b*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *e32xm2\_t* `vmfnevf_mask_e32xm2_float32xm2` (*e32xm2\_t* *merge*, *float32xm2\_t* *a*, *float b*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *e32xm4\_t* `vmfnevf_mask_e32xm4_float32xm4` (*e32xm4\_t* *merge*, *float32xm4\_t* *a*, *float b*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *e32xm8\_t* `vmfnevf_mask_e32xm8_float32xm8` (*e32xm8\_t* *merge*, *float32xm8\_t* *a*, *float b*, *e32xm8\_t* *mask*, unsigned int *gvl*)
- *e64xm1\_t* `vmfnevf_mask_e64xm1_float64xm1` (*e64xm1\_t* *merge*, *float64xm1\_t* *a*, *double b*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *e64xm2\_t* `vmfnevf_mask_e64xm2_float64xm2` (*e64xm2\_t* *merge*, *float64xm2\_t* *a*, *double b*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *e64xm4\_t* `vmfnevf_mask_e64xm4_float64xm4` (*e64xm4\_t* *merge*, *float64xm4\_t* *a*, *double b*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *e64xm8\_t* `vmfnevf_mask_e64xm8_float64xm8` (*e64xm8\_t* *merge*, *float64xm8\_t* *a*, *double b*, *e64xm8\_t* *mask*, unsigned int *gvl*)

**Masked operation:**



```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = (a[element] != b) ? 1 : 0
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.6.12 Compare elementwise float vector-vector for inequality

**Instruction:** ['vmfne.vv']

**Prototypes:**

- *e16xm1\_t vmfnevv\_e16xm1\_float16xm1* (*float16xm1\_t a, float16xm1\_t b*, unsigned int *gvl*)
- *e16xm2\_t vmfnevv\_e16xm2\_float16xm2* (*float16xm2\_t a, float16xm2\_t b*, unsigned int *gvl*)
- *e16xm4\_t vmfnevv\_e16xm4\_float16xm4* (*float16xm4\_t a, float16xm4\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmfnevv\_e16xm8\_float16xm8* (*float16xm8\_t a, float16xm8\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmfnevv\_e32xm1\_float32xm1* (*float32xm1\_t a, float32xm1\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmfnevv\_e32xm2\_float32xm2* (*float32xm2\_t a, float32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmfnevv\_e32xm4\_float32xm4* (*float32xm4\_t a, float32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmfnevv\_e32xm8\_float32xm8* (*float32xm8\_t a, float32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmfnevv\_e64xm1\_float64xm1* (*float64xm1\_t a, float64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmfnevv\_e64xm2\_float64xm2* (*float64xm2\_t a, float64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmfnevv\_e64xm4\_float64xm4* (*float64xm4\_t a, float64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmfnevv\_e64xm8\_float64xm8* (*float64xm8\_t a, float64xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = (a[element] != b[element]) ? 1 : 0
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmfnevv\_mask\_e16xm1\_float16xm1* (*e16xm1\_t merge, float16xm1\_t a, float16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmfnevv\_mask\_e16xm2\_float16xm2* (*e16xm2\_t merge, float16xm2\_t a, float16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmfnevv\_mask\_e16xm4\_float16xm4* (*e16xm4\_t merge, float16xm4\_t a, float16xm4\_t b, e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmfnevv\_mask\_e16xm8\_float16xm8* (*e16xm8\_t merge, float16xm8\_t a, float16xm8\_t b, e16xm8\_t mask*, unsigned int *gvl*)

- `e32xm1_t vmfnevv_mask_e32xm1_float32xm1` (`e32xm1_t merge, float32xm1_t a, float32xm1_t b, e32xm1_t mask, unsigned int gvl`)
- `e32xm2_t vmfnevv_mask_e32xm2_float32xm2` (`e32xm2_t merge, float32xm2_t a, float32xm2_t b, e32xm2_t mask, unsigned int gvl`)
- `e32xm4_t vmfnevv_mask_e32xm4_float32xm4` (`e32xm4_t merge, float32xm4_t a, float32xm4_t b, e32xm4_t mask, unsigned int gvl`)
- `e32xm8_t vmfnevv_mask_e32xm8_float32xm8` (`e32xm8_t merge, float32xm8_t a, float32xm8_t b, e32xm8_t mask, unsigned int gvl`)
- `e64xm1_t vmfnevv_mask_e64xm1_float64xm1` (`e64xm1_t merge, float64xm1_t a, float64xm1_t b, e64xm1_t mask, unsigned int gvl`)
- `e64xm2_t vmfnevv_mask_e64xm2_float64xm2` (`e64xm2_t merge, float64xm2_t a, float64xm2_t b, e64xm2_t mask, unsigned int gvl`)
- `e64xm4_t vmfnevv_mask_e64xm4_float64xm4` (`e64xm4_t merge, float64xm4_t a, float64xm4_t b, e64xm4_t mask, unsigned int gvl`)
- `e64xm8_t vmfnevv_mask_e64xm8_float64xm8` (`e64xm8_t merge, float64xm8_t a, float64xm8_t b, e64xm8_t mask, unsigned int gvl`)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = (a[element] != b[element]) ? 1 : 0
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

### 2.6.13 Compute elementwise if vector-scalar are ordered floating-point values

**Instruction:** ['vmfordvf']

**Prototypes:**

- `e16xm1_t vmfordvf_e16xm1_float16xm1` (`float16xm1_t a, float16_t b, unsigned int gvl`)
- `e16xm2_t vmfordvf_e16xm2_float16xm2` (`float16xm2_t a, float16_t b, unsigned int gvl`)
- `e16xm4_t vmfordvf_e16xm4_float16xm4` (`float16xm4_t a, float16_t b, unsigned int gvl`)
- `e16xm8_t vmfordvf_e16xm8_float16xm8` (`float16xm8_t a, float16_t b, unsigned int gvl`)
- `e32xm1_t vmfordvf_e32xm1_float32xm1` (`float32xm1_t a, float b, unsigned int gvl`)
- `e32xm2_t vmfordvf_e32xm2_float32xm2` (`float32xm2_t a, float b, unsigned int gvl`)
- `e32xm4_t vmfordvf_e32xm4_float32xm4` (`float32xm4_t a, float b, unsigned int gvl`)
- `e32xm8_t vmfordvf_e32xm8_float32xm8` (`float32xm8_t a, float b, unsigned int gvl`)
- `e64xm1_t vmfordvf_e64xm1_float64xm1` (`float64xm1_t a, double b, unsigned int gvl`)

- *e64xm2\_t* **vmfordvf\_e64xm2\_float64xm2** (*float64xm2\_t* *a*, double *b*, unsigned int *gvl*)
- *e64xm4\_t* **vmfordvf\_e64xm4\_float64xm4** (*float64xm4\_t* *a*, double *b*, unsigned int *gvl*)
- *e64xm8\_t* **vmfordvf\_e64xm8\_float64xm8** (*float64xm8\_t* *a*, double *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] vmford.vf b) ? 1 : 0
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t* **vmfordvf\_mask\_e16xm1\_float16xm1** (*e16xm1\_t* *merge*, *float16xm1\_t* *a*, float16\_t *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *e16xm2\_t* **vmfordvf\_mask\_e16xm2\_float16xm2** (*e16xm2\_t* *merge*, *float16xm2\_t* *a*, float16\_t *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *e16xm4\_t* **vmfordvf\_mask\_e16xm4\_float16xm4** (*e16xm4\_t* *merge*, *float16xm4\_t* *a*, float16\_t *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *e16xm8\_t* **vmfordvf\_mask\_e16xm8\_float16xm8** (*e16xm8\_t* *merge*, *float16xm8\_t* *a*, float16\_t *b*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- *e32xm1\_t* **vmfordvf\_mask\_e32xm1\_float32xm1** (*e32xm1\_t* *merge*, *float32xm1\_t* *a*, float *b*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *e32xm2\_t* **vmfordvf\_mask\_e32xm2\_float32xm2** (*e32xm2\_t* *merge*, *float32xm2\_t* *a*, float *b*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *e32xm4\_t* **vmfordvf\_mask\_e32xm4\_float32xm4** (*e32xm4\_t* *merge*, *float32xm4\_t* *a*, float *b*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *e32xm8\_t* **vmfordvf\_mask\_e32xm8\_float32xm8** (*e32xm8\_t* *merge*, *float32xm8\_t* *a*, float *b*, *e32xm8\_t* *mask*, unsigned int *gvl*)
- *e64xm1\_t* **vmfordvf\_mask\_e64xm1\_float64xm1** (*e64xm1\_t* *merge*, *float64xm1\_t* *a*, double *b*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *e64xm2\_t* **vmfordvf\_mask\_e64xm2\_float64xm2** (*e64xm2\_t* *merge*, *float64xm2\_t* *a*, double *b*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *e64xm4\_t* **vmfordvf\_mask\_e64xm4\_float64xm4** (*e64xm4\_t* *merge*, *float64xm4\_t* *a*, double *b*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *e64xm8\_t* **vmfordvf\_mask\_e64xm8\_float64xm8** (*e64xm8\_t* *merge*, *float64xm8\_t* *a*, double *b*, *e64xm8\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[element] vmford.vf b) ? 1 : 0
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.6.14 Compute elementwise if vector-vector are ordered floating-point values****Instruction:** ['vmford.vv']**Prototypes:**

- *e16xm1\_t* **vmfordvv\_e16xm1\_float16xm1** (*float16xm1\_t* a, *float16xm1\_t* b, unsigned int gvl)
- *e16xm2\_t* **vmfordvv\_e16xm2\_float16xm2** (*float16xm2\_t* a, *float16xm2\_t* b, unsigned int gvl)
- *e16xm4\_t* **vmfordvv\_e16xm4\_float16xm4** (*float16xm4\_t* a, *float16xm4\_t* b, unsigned int gvl)
- *e16xm8\_t* **vmfordvv\_e16xm8\_float16xm8** (*float16xm8\_t* a, *float16xm8\_t* b, unsigned int gvl)
- *e32xm1\_t* **vmfordvv\_e32xm1\_float32xm1** (*float32xm1\_t* a, *float32xm1\_t* b, unsigned int gvl)
- *e32xm2\_t* **vmfordvv\_e32xm2\_float32xm2** (*float32xm2\_t* a, *float32xm2\_t* b, unsigned int gvl)
- *e32xm4\_t* **vmfordvv\_e32xm4\_float32xm4** (*float32xm4\_t* a, *float32xm4\_t* b, unsigned int gvl)
- *e32xm8\_t* **vmfordvv\_e32xm8\_float32xm8** (*float32xm8\_t* a, *float32xm8\_t* b, unsigned int gvl)
- *e64xm1\_t* **vmfordvv\_e64xm1\_float64xm1** (*float64xm1\_t* a, *float64xm1\_t* b, unsigned int gvl)
- *e64xm2\_t* **vmfordvv\_e64xm2\_float64xm2** (*float64xm2\_t* a, *float64xm2\_t* b, unsigned int gvl)
- *e64xm4\_t* **vmfordvv\_e64xm4\_float64xm4** (*float64xm4\_t* a, *float64xm4\_t* b, unsigned int gvl)
- *e64xm8\_t* **vmfordvv\_e64xm8\_float64xm8** (*float64xm8\_t* a, *float64xm8\_t* b, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = !is_nan(a[element]) & !is_nan(b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t* **vmfordvv\_mask\_e16xm1\_float16xm1** (*e16xm1\_t* merge, *float16xm1\_t* a, *float16xm1\_t* b, *e16xm1\_t* mask, unsigned int gvl)
- *e16xm2\_t* **vmfordvv\_mask\_e16xm2\_float16xm2** (*e16xm2\_t* merge, *float16xm2\_t* a, *float16xm2\_t* b, *e16xm2\_t* mask, unsigned int gvl)
- *e16xm4\_t* **vmfordvv\_mask\_e16xm4\_float16xm4** (*e16xm4\_t* merge, *float16xm4\_t* a, *float16xm4\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *e16xm8\_t* **vmfordvv\_mask\_e16xm8\_float16xm8** (*e16xm8\_t* merge, *float16xm8\_t* a, *float16xm8\_t* b, *e16xm8\_t* mask, unsigned int gvl)
- *e32xm1\_t* **vmfordvv\_mask\_e32xm1\_float32xm1** (*e32xm1\_t* merge, *float32xm1\_t* a, *float32xm1\_t* b, *e32xm1\_t* mask, unsigned int gvl)
- *e32xm2\_t* **vmfordvv\_mask\_e32xm2\_float32xm2** (*e32xm2\_t* merge, *float32xm2\_t* a, *float32xm2\_t* b, *e32xm2\_t* mask, unsigned int gvl)
- *e32xm4\_t* **vmfordvv\_mask\_e32xm4\_float32xm4** (*e32xm4\_t* merge, *float32xm4\_t* a, *float32xm4\_t* b, *e32xm4\_t* mask, unsigned int gvl)
- *e32xm8\_t* **vmfordvv\_mask\_e32xm8\_float32xm8** (*e32xm8\_t* merge, *float32xm8\_t* a, *float32xm8\_t* b, *e32xm8\_t* mask, unsigned int gvl)
- *e64xm1\_t* **vmfordvv\_mask\_e64xm1\_float64xm1** (*e64xm1\_t* merge, *float64xm1\_t* a, *float64xm1\_t* b, *e64xm1\_t* mask, unsigned int gvl)

- `e64xm2_t vmfordvv_mask_e64xm2_float64xm2 (e64xm2_t merge, float64xm2_t a, float64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `e64xm4_t vmfordvv_mask_e64xm4_float64xm4 (e64xm4_t merge, float64xm4_t a, float64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `e64xm8_t vmfordvv_mask_e64xm8_float64xm8 (e64xm8_t merge, float64xm8_t a, float64xm8_t b, e64xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = !is_nan(a[element]) & !is_nan(b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7 Integer arithmetic operations

### 2.7.1 Elementwise vector-immediate integer average add

**Instruction:** ['vaadd.vi']**Prototypes:**

- `int16xm1_t vaaddvi_int16xm1 (int16xm1_t a, const short b, unsigned int gvl)`
- `int16xm2_t vaaddvi_int16xm2 (int16xm2_t a, const short b, unsigned int gvl)`
- `int16xm4_t vaaddvi_int16xm4 (int16xm4_t a, const short b, unsigned int gvl)`
- `int16xm8_t vaaddvi_int16xm8 (int16xm8_t a, const short b, unsigned int gvl)`
- `int32xm1_t vaaddvi_int32xm1 (int32xm1_t a, const int b, unsigned int gvl)`
- `int32xm2_t vaaddvi_int32xm2 (int32xm2_t a, const int b, unsigned int gvl)`
- `int32xm4_t vaaddvi_int32xm4 (int32xm4_t a, const int b, unsigned int gvl)`
- `int32xm8_t vaaddvi_int32xm8 (int32xm8_t a, const int b, unsigned int gvl)`
- `int64xm1_t vaaddvi_int64xm1 (int64xm1_t a, const long b, unsigned int gvl)`
- `int64xm2_t vaaddvi_int64xm2 (int64xm2_t a, const long b, unsigned int gvl)`
- `int64xm4_t vaaddvi_int64xm4 (int64xm4_t a, const long b, unsigned int gvl)`
- `int64xm8_t vaaddvi_int64xm8 (int64xm8_t a, const long b, unsigned int gvl)`
- `int8xm1_t vaaddvi_int8xm1 (int8xm1_t a, const signed char b, unsigned int gvl)`
- `int8xm2_t vaaddvi_int8xm2 (int8xm2_t a, const signed char b, unsigned int gvl)`
- `int8xm4_t vaaddvi_int8xm4 (int8xm4_t a, const signed char b, unsigned int gvl)`
- `int8xm8_t vaaddvi_int8xm8 (int8xm8_t a, const signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] + b) / 2
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vaaddvi\_mask\_int16xm1* (*int16xm1\_t merge, int16xm1\_t a, const short b, e16xm1\_t mask*, unsigned int gvl)
- *int16xm2\_t vaaddvi\_mask\_int16xm2* (*int16xm2\_t merge, int16xm2\_t a, const short b, e16xm2\_t mask*, unsigned int gvl)
- *int16xm4\_t vaaddvi\_mask\_int16xm4* (*int16xm4\_t merge, int16xm4\_t a, const short b, e16xm4\_t mask*, unsigned int gvl)
- *int16xm8\_t vaaddvi\_mask\_int16xm8* (*int16xm8\_t merge, int16xm8\_t a, const short b, e16xm8\_t mask*, unsigned int gvl)
- *int32xm1\_t vaaddvi\_mask\_int32xm1* (*int32xm1\_t merge, int32xm1\_t a, const int b, e32xm1\_t mask*, unsigned int gvl)
- *int32xm2\_t vaaddvi\_mask\_int32xm2* (*int32xm2\_t merge, int32xm2\_t a, const int b, e32xm2\_t mask*, unsigned int gvl)
- *int32xm4\_t vaaddvi\_mask\_int32xm4* (*int32xm4\_t merge, int32xm4\_t a, const int b, e32xm4\_t mask*, unsigned int gvl)
- *int32xm8\_t vaaddvi\_mask\_int32xm8* (*int32xm8\_t merge, int32xm8\_t a, const int b, e32xm8\_t mask*, unsigned int gvl)
- *int64xm1\_t vaaddvi\_mask\_int64xm1* (*int64xm1\_t merge, int64xm1\_t a, const long b, e64xm1\_t mask*, unsigned int gvl)
- *int64xm2\_t vaaddvi\_mask\_int64xm2* (*int64xm2\_t merge, int64xm2\_t a, const long b, e64xm2\_t mask*, unsigned int gvl)
- *int64xm4\_t vaaddvi\_mask\_int64xm4* (*int64xm4\_t merge, int64xm4\_t a, const long b, e64xm4\_t mask*, unsigned int gvl)
- *int64xm8\_t vaaddvi\_mask\_int64xm8* (*int64xm8\_t merge, int64xm8\_t a, const long b, e64xm8\_t mask*, unsigned int gvl)
- *int8xm1\_t vaaddvi\_mask\_int8xm1* (*int8xm1\_t merge, int8xm1\_t a, const signed char b, e8xm1\_t mask*, unsigned int gvl)
- *int8xm2\_t vaaddvi\_mask\_int8xm2* (*int8xm2\_t merge, int8xm2\_t a, const signed char b, e8xm2\_t mask*, unsigned int gvl)
- *int8xm4\_t vaaddvi\_mask\_int8xm4* (*int8xm4\_t merge, int8xm4\_t a, const signed char b, e8xm4\_t mask*, unsigned int gvl)
- *int8xm8\_t vaaddvi\_mask\_int8xm8* (*int8xm8\_t merge, int8xm8\_t a, const signed char b, e8xm8\_t mask*, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[element] + b) / 2
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```



## 2.7.2 Elementwise vector-vector integer average add

**Instruction:** ['vaadd.vv']

**Prototypes:**

- `int16xm1_t vaaddvv_int16xm1 (int16xm1_t a, int16xm1_t b, unsigned int gvl)`
- `int16xm2_t vaaddvv_int16xm2 (int16xm2_t a, int16xm2_t b, unsigned int gvl)`
- `int16xm4_t vaaddvv_int16xm4 (int16xm4_t a, int16xm4_t b, unsigned int gvl)`
- `int16xm8_t vaaddvv_int16xm8 (int16xm8_t a, int16xm8_t b, unsigned int gvl)`
- `int32xm1_t vaaddvv_int32xm1 (int32xm1_t a, int32xm1_t b, unsigned int gvl)`
- `int32xm2_t vaaddvv_int32xm2 (int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int32xm4_t vaaddvv_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vaaddvv_int32xm8 (int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vaaddvv_int64xm1 (int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vaaddvv_int64xm2 (int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vaaddvv_int64xm4 (int64xm4_t a, int64xm4_t b, unsigned int gvl)`
- `int64xm8_t vaaddvv_int64xm8 (int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vaaddvv_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vaaddvv_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vaaddvv_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vaaddvv_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] + b[element]) / 2
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vaaddvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vaaddvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vaaddvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vaaddvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vaaddvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vaaddvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vaaddvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vaaddvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`

- `int64xm1_t vaaddvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vaaddvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vaaddvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vaaddvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vaaddvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vaaddvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vaaddvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vaaddvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[element] + b[element]) / 2
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

### 2.7.3 Elementwise vector-scalar integer average add

**Instruction:** ['vaadd.vx']

#### Prototypes:

- `int16xm1_t vaaddvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int16xm2_t vaaddvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vaaddvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vaaddvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vaaddvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vaaddvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vaaddvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vaaddvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vaaddvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vaaddvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vaaddvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vaaddvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vaaddvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vaaddvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`

- `int8xm4_t vaaddvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vaaddvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] + b)/2
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vaaddvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vaaddvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vaaddvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vaaddvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vaaddvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vaaddvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vaaddvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vaaddvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vaaddvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vaaddvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vaaddvx_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vaaddvx_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vaaddvx_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vaaddvx_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vaaddvx_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, signed char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vaaddvx_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, signed char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[element] + b)/2
      else
```

(continues on next page)

(continued from previous page)

```
result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.4 Elementwise vector-immediate integer addition with carry

**Instruction:** [`vadc.vim`']

**Masked prototypes:**

- `int16xm1_t vadcvim_mask_int16xm1 (int16xm1_t a, const int b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vadcvim_mask_int16xm2 (int16xm2_t a, const int b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vadcvim_mask_int16xm4 (int16xm4_t a, const int b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vadcvim_mask_int16xm8 (int16xm8_t a, const int b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vadcvim_mask_int32xm1 (int32xm1_t a, const int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vadcvim_mask_int32xm2 (int32xm2_t a, const int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vadcvim_mask_int32xm4 (int32xm4_t a, const int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vadcvim_mask_int32xm8 (int32xm8_t a, const int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vadcvim_mask_int64xm1 (int64xm1_t a, const int b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vadcvim_mask_int64xm2 (int64xm2_t a, const int b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vadcvim_mask_int64xm4 (int64xm4_t a, const int b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vadcvim_mask_int64xm8 (int64xm8_t a, const int b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vadcvim_mask_int8xm1 (int8xm1_t a, const int b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vadcvim_mask_int8xm2 (int8xm2_t a, const int b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vadcvim_mask_int8xm4 (int8xm4_t a, const int b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vadcvim_mask_int8xm8 (int8xm8_t a, const int b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vadcvim_mask_uint16xm1 (uint16xm1_t a, const int b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vadcvim_mask_uint16xm2 (uint16xm2_t a, const int b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vadcvim_mask_uint16xm4 (uint16xm4_t a, const int b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vadcvim_mask_uint16xm8 (uint16xm8_t a, const int b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vadcvim_mask_uint32xm1 (uint32xm1_t a, const int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vadcvim_mask_uint32xm2 (uint32xm2_t a, const int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vadcvim_mask_uint32xm4 (uint32xm4_t a, const int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vadcvim_mask_uint32xm8 (uint32xm8_t a, const int b, e32xm8_t mask, unsigned int gvl)`

- `uint64xm1_t vadcvm_mask_uint64xm1 (uint64xm1_t a, const int b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vadcvm_mask_uint64xm2 (uint64xm2_t a, const int b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vadcvm_mask_uint64xm4 (uint64xm4_t a, const int b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vadcvm_mask_uint64xm8 (uint64xm8_t a, const int b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vadcvm_mask_uint8xm1 (uint8xm1_t a, const int b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vadcvm_mask_uint8xm2 (uint8xm2_t a, const int b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vadcvm_mask_uint8xm4 (uint8xm4_t a, const int b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vadcvm_mask_uint8xm8 (uint8xm8_t a, const int b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + b + mask[elemet]
result[gvl : VLMAX] = 0
```

## 2.7.5 Elementwise vector-vector integer addition with carry

Instruction: ['vadc.vvm']

Masked prototypes:

- `int16xm1_t vadcvm_mask_int16xm1 (int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vadcvm_mask_int16xm2 (int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vadcvm_mask_int16xm4 (int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vadcvm_mask_int16xm8 (int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vadcvm_mask_int32xm1 (int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vadcvm_mask_int32xm2 (int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vadcvm_mask_int32xm4 (int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vadcvm_mask_int32xm8 (int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vadcvm_mask_int64xm1 (int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vadcvm_mask_int64xm2 (int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vadcvm_mask_int64xm4 (int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`

- `int64xm8_t vadcvvm_mask_int64xm8 (int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vadcvvm_mask_int8xm1 (int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vadcvvm_mask_int8xm2 (int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vadcvvm_mask_int8xm4 (int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vadcvvm_mask_int8xm8 (int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vadcvvm_mask_uint16xm1 (uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vadcvvm_mask_uint16xm2 (uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vadcvvm_mask_uint16xm4 (uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vadcvvm_mask_uint16xm8 (uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vadcvvm_mask_uint32xm1 (uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vadcvvm_mask_uint32xm2 (uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vadcvvm_mask_uint32xm4 (uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vadcvvm_mask_uint32xm8 (uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vadcvvm_mask_uint64xm1 (uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vadcvvm_mask_uint64xm2 (uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vadcvvm_mask_uint64xm4 (uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vadcvvm_mask_uint64xm8 (uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vadcvvm_mask_uint8xm1 (uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vadcvvm_mask_uint8xm2 (uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vadcvvm_mask_uint8xm4 (uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vadcvvm_mask_uint8xm8 (uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + b[element] + mask[elemet]
result[gvl : VLMAX] = 0
```

## 2.7.6 Elementwise vector-scalar integer addition with carry

**Instruction:** [`vadc.vxm`']

**Masked prototypes:**



- `int16xm1_t vadcvxm_mask_int16xm1 (int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vadcvxm_mask_int16xm2 (int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vadcvxm_mask_int16xm4 (int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vadcvxm_mask_int16xm8 (int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vadcvxm_mask_int32xm1 (int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vadcvxm_mask_int32xm2 (int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vadcvxm_mask_int32xm4 (int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vadcvxm_mask_int32xm8 (int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vadcvxm_mask_int64xm1 (int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vadcvxm_mask_int64xm2 (int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vadcvxm_mask_int64xm4 (int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vadcvxm_mask_int64xm8 (int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vadcvxm_mask_int8xm1 (int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vadcvxm_mask_int8xm2 (int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vadcvxm_mask_int8xm4 (int8xm4_t a, signed char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vadcvxm_mask_int8xm8 (int8xm8_t a, signed char b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vadcvxm_mask_uint16xm1 (uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vadcvxm_mask_uint16xm2 (uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vadcvxm_mask_uint16xm4 (uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vadcvxm_mask_uint16xm8 (uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vadcvxm_mask_uint32xm1 (uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vadcvxm_mask_uint32xm2 (uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vadcvxm_mask_uint32xm4 (uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vadcvxm_mask_uint32xm8 (uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vadcvxm_mask_uint64xm1 (uint64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vadcvxm_mask_uint64xm2 (uint64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vadcvxm_mask_uint64xm4 (uint64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vadcvxm_mask_uint64xm8 (uint64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vadcvxm_mask_uint8xm1 (uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`

- `uint8xm2_t vadcvm_mask_uint8xm2 (uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vadcvm_mask_uint8xm4 (uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vadcvm_mask_uint8xm8 (uint8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + b + mask[elemet]
result[gvl : VLMAX] = 0
```

## 2.7.7 Elementwise vector-immediate integer addition

**Instruction:** [`vadd.vi`']

#### Prototypes:

- `int16xm1_t vaddvi_int16xm1 (int16xm1_t a, const short b, unsigned int gvl)`
- `int16xm2_t vaddvi_int16xm2 (int16xm2_t a, const short b, unsigned int gvl)`
- `int16xm4_t vaddvi_int16xm4 (int16xm4_t a, const short b, unsigned int gvl)`
- `int16xm8_t vaddvi_int16xm8 (int16xm8_t a, const short b, unsigned int gvl)`
- `int32xm1_t vaddvi_int32xm1 (int32xm1_t a, const int b, unsigned int gvl)`
- `int32xm2_t vaddvi_int32xm2 (int32xm2_t a, const int b, unsigned int gvl)`
- `int32xm4_t vaddvi_int32xm4 (int32xm4_t a, const int b, unsigned int gvl)`
- `int32xm8_t vaddvi_int32xm8 (int32xm8_t a, const int b, unsigned int gvl)`
- `int64xm1_t vaddvi_int64xm1 (int64xm1_t a, const long b, unsigned int gvl)`
- `int64xm2_t vaddvi_int64xm2 (int64xm2_t a, const long b, unsigned int gvl)`
- `int64xm4_t vaddvi_int64xm4 (int64xm4_t a, const long b, unsigned int gvl)`
- `int64xm8_t vaddvi_int64xm8 (int64xm8_t a, const long b, unsigned int gvl)`
- `int8xm1_t vaddvi_int8xm1 (int8xm1_t a, const signed char b, unsigned int gvl)`
- `int8xm2_t vaddvi_int8xm2 (int8xm2_t a, const signed char b, unsigned int gvl)`
- `int8xm4_t vaddvi_int8xm4 (int8xm4_t a, const signed char b, unsigned int gvl)`
- `int8xm8_t vaddvi_int8xm8 (int8xm8_t a, const signed char b, unsigned int gvl)`
- `uint16xm1_t vaddvi_uint16xm1 (uint16xm1_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm2_t vaddvi_uint16xm2 (uint16xm2_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm4_t vaddvi_uint16xm4 (uint16xm4_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm8_t vaddvi_uint16xm8 (uint16xm8_t a, const unsigned short b, unsigned int gvl)`
- `uint32xm1_t vaddvi_uint32xm1 (uint32xm1_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm2_t vaddvi_uint32xm2 (uint32xm2_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm4_t vaddvi_uint32xm4 (uint32xm4_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm8_t vaddvi_uint32xm8 (uint32xm8_t a, const unsigned int b, unsigned int gvl)`

- `uint64xm1_t vaddvi_uint64xm1 (uint64xm1_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm2_t vaddvi_uint64xm2 (uint64xm2_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm4_t vaddvi_uint64xm4 (uint64xm4_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm8_t vaddvi_uint64xm8 (uint64xm8_t a, const unsigned long b, unsigned int gvl)`
- `uint8xm1_t vaddvi_uint8xm1 (uint8xm1_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm2_t vaddvi_uint8xm2 (uint8xm2_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm4_t vaddvi_uint8xm4 (uint8xm4_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm8_t vaddvi_uint8xm8 (uint8xm8_t a, const unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vaddvi_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, const short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vaddvi_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, const short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vaddvi_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, const short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vaddvi_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, const short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vaddvi_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, const int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vaddvi_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, const int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vaddvi_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, const int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vaddvi_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, const int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vaddvi_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, const long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vaddvi_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, const long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vaddvi_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, const long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vaddvi_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, const long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vaddvi_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, const signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vaddvi_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, const signed char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vaddvi_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, const signed char b, e8xm4_t mask, unsigned int gvl)`

- `int8xm8_t vaddvi_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, const signed char b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vaddvi_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, const unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vaddvi_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, const unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vaddvi_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, const unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vaddvi_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, const unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vaddvi_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, const unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vaddvi_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, const unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vaddvi_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, const unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vaddvi_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, const unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vaddvi_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, const unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vaddvi_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, const unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vaddvi_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, const unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vaddvi_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, const unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vaddvi_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, const unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vaddvi_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, const unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vaddvi_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, const unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vaddvi_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, const unsigned char b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = a[element] + b
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.8 Elementwise vector-vector integer addition

Instruction: ['vadd.vv']

Prototypes:

- `int16xm1_t vaddvv_int16xm1 (int16xm1_t a, int16xm1_t b, unsigned int gvl)`
- `int16xm2_t vaddvv_int16xm2 (int16xm2_t a, int16xm2_t b, unsigned int gvl)`
- `int16xm4_t vaddvv_int16xm4 (int16xm4_t a, int16xm4_t b, unsigned int gvl)`
- `int16xm8_t vaddvv_int16xm8 (int16xm8_t a, int16xm8_t b, unsigned int gvl)`
- `int32xm1_t vaddvv_int32xm1 (int32xm1_t a, int32xm1_t b, unsigned int gvl)`
- `int32xm2_t vaddvv_int32xm2 (int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int32xm4_t vaddvv_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vaddvv_int32xm8 (int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vaddvv_int64xm1 (int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vaddvv_int64xm2 (int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vaddvv_int64xm4 (int64xm4_t a, int64xm4_t b, unsigned int gvl)`
- `int64xm8_t vaddvv_int64xm8 (int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vaddvv_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vaddvv_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vaddvv_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vaddvv_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`
- `uint16xm1_t vaddvv_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vaddvv_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vaddvv_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vaddvv_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vaddvv_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vaddvv_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vaddvv_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vaddvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vaddvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vaddvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vaddvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vaddvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vaddvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vaddvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vaddvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vaddvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + b[element]
result[gvl : VLMAX] = 0
```



**Masked prototypes:**

- `int16xm1_t vaddvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vaddvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vaddvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vaddvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vaddvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vaddvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vaddvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vaddvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vaddvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vaddvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vaddvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vaddvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vaddvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vaddvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vaddvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vaddvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vaddvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vaddvv_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vaddvv_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vaddvv_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vaddvv_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vaddvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`



- `uint32xm4_t vaddvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vaddvv_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vaddvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vaddvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vaddvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vaddvv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vaddvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vaddvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vaddvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vaddvv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] + b[element]
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.7.9 Elementwise vector-scalar integer addition

Instruction: `['vadd.vx']`

Prototypes:

- `int16xm1_t vaddvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int16xm2_t vaddvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vaddvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vaddvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vaddvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vaddvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vaddvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vaddvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vaddvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vaddvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vaddvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`

- `int64xm8_t vaddvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vaddvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vaddvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vaddvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vaddvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`
- `uint16xm1_t vaddvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vaddvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vaddvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vaddvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vaddvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vaddvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vaddvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vaddvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vaddvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vaddvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vaddvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vaddvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vaddvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vaddvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vaddvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vaddvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vaddvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vaddvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vaddvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vaddvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vaddvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vaddvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vaddvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`

- *int32xm8\_t vaddvx\_mask\_int32xm8* (*int32xm8\_t merge*, *int32xm8\_t a*, *int b*, *e32xm8\_t mask*, unsigned int gvl)
- *int64xm1\_t vaddvx\_mask\_int64xm1* (*int64xm1\_t merge*, *int64xm1\_t a*, *long b*, *e64xm1\_t mask*, unsigned int gvl)
- *int64xm2\_t vaddvx\_mask\_int64xm2* (*int64xm2\_t merge*, *int64xm2\_t a*, *long b*, *e64xm2\_t mask*, unsigned int gvl)
- *int64xm4\_t vaddvx\_mask\_int64xm4* (*int64xm4\_t merge*, *int64xm4\_t a*, *long b*, *e64xm4\_t mask*, unsigned int gvl)
- *int64xm8\_t vaddvx\_mask\_int64xm8* (*int64xm8\_t merge*, *int64xm8\_t a*, *long b*, *e64xm8\_t mask*, unsigned int gvl)
- *int8xm1\_t vaddvx\_mask\_int8xm1* (*int8xm1\_t merge*, *int8xm1\_t a*, *signed char b*, *e8xm1\_t mask*, unsigned int gvl)
- *int8xm2\_t vaddvx\_mask\_int8xm2* (*int8xm2\_t merge*, *int8xm2\_t a*, *signed char b*, *e8xm2\_t mask*, unsigned int gvl)
- *int8xm4\_t vaddvx\_mask\_int8xm4* (*int8xm4\_t merge*, *int8xm4\_t a*, *signed char b*, *e8xm4\_t mask*, unsigned int gvl)
- *int8xm8\_t vaddvx\_mask\_int8xm8* (*int8xm8\_t merge*, *int8xm8\_t a*, *signed char b*, *e8xm8\_t mask*, unsigned int gvl)
- *uint16xm1\_t vaddvx\_mask\_uint16xm1* (*uint16xm1\_t merge*, *uint16xm1\_t a*, *unsigned short b*, *e16xm1\_t mask*, unsigned int gvl)
- *uint16xm2\_t vaddvx\_mask\_uint16xm2* (*uint16xm2\_t merge*, *uint16xm2\_t a*, *unsigned short b*, *e16xm2\_t mask*, unsigned int gvl)
- *uint16xm4\_t vaddvx\_mask\_uint16xm4* (*uint16xm4\_t merge*, *uint16xm4\_t a*, *unsigned short b*, *e16xm4\_t mask*, unsigned int gvl)
- *uint16xm8\_t vaddvx\_mask\_uint16xm8* (*uint16xm8\_t merge*, *uint16xm8\_t a*, *unsigned short b*, *e16xm8\_t mask*, unsigned int gvl)
- *uint32xm1\_t vaddvx\_mask\_uint32xm1* (*uint32xm1\_t merge*, *uint32xm1\_t a*, *unsigned int b*, *e32xm1\_t mask*, unsigned int gvl)
- *uint32xm2\_t vaddvx\_mask\_uint32xm2* (*uint32xm2\_t merge*, *uint32xm2\_t a*, *unsigned int b*, *e32xm2\_t mask*, unsigned int gvl)
- *uint32xm4\_t vaddvx\_mask\_uint32xm4* (*uint32xm4\_t merge*, *uint32xm4\_t a*, *unsigned int b*, *e32xm4\_t mask*, unsigned int gvl)
- *uint32xm8\_t vaddvx\_mask\_uint32xm8* (*uint32xm8\_t merge*, *uint32xm8\_t a*, *unsigned int b*, *e32xm8\_t mask*, unsigned int gvl)
- *uint64xm1\_t vaddvx\_mask\_uint64xm1* (*uint64xm1\_t merge*, *uint64xm1\_t a*, *unsigned long b*, *e64xm1\_t mask*, unsigned int gvl)
- *uint64xm2\_t vaddvx\_mask\_uint64xm2* (*uint64xm2\_t merge*, *uint64xm2\_t a*, *unsigned long b*, *e64xm2\_t mask*, unsigned int gvl)
- *uint64xm4\_t vaddvx\_mask\_uint64xm4* (*uint64xm4\_t merge*, *uint64xm4\_t a*, *unsigned long b*, *e64xm4\_t mask*, unsigned int gvl)
- *uint64xm8\_t vaddvx\_mask\_uint64xm8* (*uint64xm8\_t merge*, *uint64xm8\_t a*, *unsigned long b*, *e64xm8\_t mask*, unsigned int gvl)
- *uint8xm1\_t vaddvx\_mask\_uint8xm1* (*uint8xm1\_t merge*, *uint8xm1\_t a*, *unsigned char b*, *e8xm1\_t mask*, unsigned int gvl)
- *uint8xm2\_t vaddvx\_mask\_uint8xm2* (*uint8xm2\_t merge*, *uint8xm2\_t a*, *unsigned char b*, *e8xm2\_t mask*, unsigned int gvl)

- `uint8xm4_t vaddvx_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vaddvx_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] + b
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.10 Elementwise vector-vector integer average sub

**Instruction:** [`vasub.vv`']

**Prototypes:**

- `int16xm1_t vasubvv_int16xm1 (int16xm1_t a, int16xm1_t b, unsigned int gvl)`
- `int16xm2_t vasubvv_int16xm2 (int16xm2_t a, int16xm2_t b, unsigned int gvl)`
- `int16xm4_t vasubvv_int16xm4 (int16xm4_t a, int16xm4_t b, unsigned int gvl)`
- `int16xm8_t vasubvv_int16xm8 (int16xm8_t a, int16xm8_t b, unsigned int gvl)`
- `int32xm1_t vasubvv_int32xm1 (int32xm1_t a, int32xm1_t b, unsigned int gvl)`
- `int32xm2_t vasubvv_int32xm2 (int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int32xm4_t vasubvv_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vasubvv_int32xm8 (int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vasubvv_int64xm1 (int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vasubvv_int64xm2 (int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vasubvv_int64xm4 (int64xm4_t a, int64xm4_t b, unsigned int gvl)`
- `int64xm8_t vasubvv_int64xm8 (int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vasubvv_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vasubvv_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vasubvv_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vasubvv_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] - b[element]) / 2
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vasubvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`

- `int16xm2_t vasubvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vasubvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vasubvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vasubvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vasubvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vasubvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vasubvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vasubvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vasubvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vasubvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vasubvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vasubvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vasubvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vasubvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vasubvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[element] - b[element]) / 2
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.7.11 Elementwise vector-scalar integer average sub****Instruction:** [`'vasub.vx'`]**Prototypes:**

- `int16xm1_t vasubvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int16xm2_t vasubvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vasubvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`

- `int16xm8_t vasubvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vasubvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vasubvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vasubvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vasubvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vasubvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vasubvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vasubvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vasubvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vasubvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vasubvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vasubvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vasubvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] - b) / 2
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vasubvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vasubvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vasubvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vasubvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vasubvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vasubvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vasubvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vasubvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vasubvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vasubvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vasubvx_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vasubvx_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`



- `int8xm1_t vasubvx_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, signed char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8xm2_t vasubvx_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, signed char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `int8xm4_t vasubvx_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, signed char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int8xm8_t vasubvx_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, signed char `b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[element] - b) / 2
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.12 Elementwise signed vector-vector division

Instruction: ['vdiv.vv']

Prototypes:

- `int16xm1_t vdivvv_int16xm1` (`int16xm1_t a`, `int16xm1_t b`, unsigned int `gvl`)
- `int16xm2_t vdivvv_int16xm2` (`int16xm2_t a`, `int16xm2_t b`, unsigned int `gvl`)
- `int16xm4_t vdivvv_int16xm4` (`int16xm4_t a`, `int16xm4_t b`, unsigned int `gvl`)
- `int16xm8_t vdivvv_int16xm8` (`int16xm8_t a`, `int16xm8_t b`, unsigned int `gvl`)
- `int32xm1_t vdivvv_int32xm1` (`int32xm1_t a`, `int32xm1_t b`, unsigned int `gvl`)
- `int32xm2_t vdivvv_int32xm2` (`int32xm2_t a`, `int32xm2_t b`, unsigned int `gvl`)
- `int32xm4_t vdivvv_int32xm4` (`int32xm4_t a`, `int32xm4_t b`, unsigned int `gvl`)
- `int32xm8_t vdivvv_int32xm8` (`int32xm8_t a`, `int32xm8_t b`, unsigned int `gvl`)
- `int64xm1_t vdivvv_int64xm1` (`int64xm1_t a`, `int64xm1_t b`, unsigned int `gvl`)
- `int64xm2_t vdivvv_int64xm2` (`int64xm2_t a`, `int64xm2_t b`, unsigned int `gvl`)
- `int64xm4_t vdivvv_int64xm4` (`int64xm4_t a`, `int64xm4_t b`, unsigned int `gvl`)
- `int64xm8_t vdivvv_int64xm8` (`int64xm8_t a`, `int64xm8_t b`, unsigned int `gvl`)
- `int8xm1_t vdivvv_int8xm1` (`int8xm1_t a`, `int8xm1_t b`, unsigned int `gvl`)
- `int8xm2_t vdivvv_int8xm2` (`int8xm2_t a`, `int8xm2_t b`, unsigned int `gvl`)
- `int8xm4_t vdivvv_int8xm4` (`int8xm4_t a`, `int8xm4_t b`, unsigned int `gvl`)
- `int8xm8_t vdivvv_int8xm8` (`int8xm8_t a`, `int8xm8_t b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] / b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vdivvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vdivvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vdivvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vdivvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vdivvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vdivvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vdivvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vdivvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vdivvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vdivvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vdivvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vdivvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vdivvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vdivvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vdivvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vdivvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] / b[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.13 Elementwise signed vector-scalar division****Instruction:** [`'vdiv.vx'`]**Prototypes:**

- `int16xm1_t vdivvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int16xm2_t vdivvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vdivvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vdivvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vdivvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vdivvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vdivvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vdivvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vdivvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vdivvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vdivvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vdivvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vdivvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vdivvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vdivvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vdivvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = / (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vdivvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vdivvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vdivvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vdivvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vdivvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vdivvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vdivvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vdivvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vdivvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vdivvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`

- `int64xm4_t vdivvx_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `int64xm8_t vdivvx_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `int8xm1_t vdivvx_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, signed char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8xm2_t vdivvx_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, signed char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `int8xm4_t vdivvx_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, signed char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int8xm8_t vdivvx_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, signed char `b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = / (a[element], b)
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.7.14 Elementwise unsigned vector-vector division****Instruction:** [`'vdivu.vv'`]**Prototypes:**

- `uint16xm1_t vdivuvv_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, unsigned int `gvl`)
- `uint16xm2_t vdivuvv_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, unsigned int `gvl`)
- `uint16xm4_t vdivuvv_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, unsigned int `gvl`)
- `uint16xm8_t vdivuvv_uint16xm8` (`uint16xm8_t a`, `uint16xm8_t b`, unsigned int `gvl`)
- `uint32xm1_t vdivuvv_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, unsigned int `gvl`)
- `uint32xm2_t vdivuvv_uint32xm2` (`uint32xm2_t a`, `uint32xm2_t b`, unsigned int `gvl`)
- `uint32xm4_t vdivuvv_uint32xm4` (`uint32xm4_t a`, `uint32xm4_t b`, unsigned int `gvl`)
- `uint32xm8_t vdivuvv_uint32xm8` (`uint32xm8_t a`, `uint32xm8_t b`, unsigned int `gvl`)
- `uint64xm1_t vdivuvv_uint64xm1` (`uint64xm1_t a`, `uint64xm1_t b`, unsigned int `gvl`)
- `uint64xm2_t vdivuvv_uint64xm2` (`uint64xm2_t a`, `uint64xm2_t b`, unsigned int `gvl`)
- `uint64xm4_t vdivuvv_uint64xm4` (`uint64xm4_t a`, `uint64xm4_t b`, unsigned int `gvl`)
- `uint64xm8_t vdivuvv_uint64xm8` (`uint64xm8_t a`, `uint64xm8_t b`, unsigned int `gvl`)
- `uint8xm1_t vdivuvv_uint8xm1` (`uint8xm1_t a`, `uint8xm1_t b`, unsigned int `gvl`)
- `uint8xm2_t vdivuvv_uint8xm2` (`uint8xm2_t a`, `uint8xm2_t b`, unsigned int `gvl`)
- `uint8xm4_t vdivuvv_uint8xm4` (`uint8xm4_t a`, `uint8xm4_t b`, unsigned int `gvl`)
- `uint8xm8_t vdivuvv_uint8xm8` (`uint8xm8_t a`, `uint8xm8_t b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] / b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vdivuvv_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, `uint16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint16xm2_t vdivuvv_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, `uint16xm2_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint16xm4_t vdivuvv_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, `uint16xm4_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vdivuvv_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, `uint16xm8_t b`, `e16xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vdivuvv_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, `uint32xm1_t b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vdivuvv_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, `uint32xm2_t b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vdivuvv_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, `uint32xm4_t b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint32xm8_t vdivuvv_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, `uint32xm8_t b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint64xm1_t vdivuvv_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, `uint64xm1_t b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vdivuvv_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, `uint64xm2_t b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vdivuvv_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, `uint64xm4_t b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vdivuvv_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, `uint64xm8_t b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vdivuvv_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vdivuvv_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vdivuvv_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vdivuvv_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, `uint8xm8_t b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] / b[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.15 Elementwise unsigned vector-scalar division

**Instruction:** ['vdivu.vx']

**Prototypes:**

- `uint16xm1_t vdivuvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vdivuvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vdivuvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vdivuvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vdivuvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vdivuvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vdivuvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vdivuvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vdivuvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vdivuvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vdivuvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vdivuvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vdivuvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vdivuvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vdivuvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vdivuvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = / (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vdivuvx_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vdivuvx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vdivuvx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vdivuvx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vdivuvx_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vdivuvx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vdivuvx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vdivuvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`



- `uint64xm1_t vdivuvx_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, unsigned long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vdivuvx_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, unsigned long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vdivuvx_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, unsigned long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vdivuvx_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, unsigned long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vdivuvx_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vdivuvx_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vdivuvx_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vdivuvx_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = / (a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.16 Elementwise vector-vector integer dot-product****Instruction:** ['vdot.vv']**Prototypes:**

- `int16xm1_t vdotvv_int16xm1` (`int16xm1_t a`, `int16xm1_t b`, unsigned int `gvl`)
- `int16xm2_t vdotvv_int16xm2` (`int16xm2_t a`, `int16xm2_t b`, unsigned int `gvl`)
- `int16xm4_t vdotvv_int16xm4` (`int16xm4_t a`, `int16xm4_t b`, unsigned int `gvl`)
- `int16xm8_t vdotvv_int16xm8` (`int16xm8_t a`, `int16xm8_t b`, unsigned int `gvl`)
- `int32xm1_t vdotvv_int32xm1` (`int32xm1_t a`, `int32xm1_t b`, unsigned int `gvl`)
- `int32xm2_t vdotvv_int32xm2` (`int32xm2_t a`, `int32xm2_t b`, unsigned int `gvl`)
- `int32xm4_t vdotvv_int32xm4` (`int32xm4_t a`, `int32xm4_t b`, unsigned int `gvl`)
- `int32xm8_t vdotvv_int32xm8` (`int32xm8_t a`, `int32xm8_t b`, unsigned int `gvl`)
- `int64xm1_t vdotvv_int64xm1` (`int64xm1_t a`, `int64xm1_t b`, unsigned int `gvl`)
- `int64xm2_t vdotvv_int64xm2` (`int64xm2_t a`, `int64xm2_t b`, unsigned int `gvl`)
- `int64xm4_t vdotvv_int64xm4` (`int64xm4_t a`, `int64xm4_t b`, unsigned int `gvl`)
- `int64xm8_t vdotvv_int64xm8` (`int64xm8_t a`, `int64xm8_t b`, unsigned int `gvl`)
- `int8xm1_t vdotvv_int8xm1` (`int8xm1_t a`, `int8xm1_t b`, unsigned int `gvl`)
- `int8xm2_t vdotvv_int8xm2` (`int8xm2_t a`, `int8xm2_t b`, unsigned int `gvl`)

- `int8xm4_t vdotvv_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vdotvv_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`
- `uint16xm1_t vdotvv_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vdotvv_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vdotvv_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vdotvv_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vdotvv_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vdotvv_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vdotvv_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vdotvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vdotvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vdotvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vdotvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vdotvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vdotvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vdotvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vdotvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vdotvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

#### Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = dot-product (a[element], b[element])
result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- `int16xm1_t vdotvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vdotvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vdotvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vdotvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vdotvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vdotvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vdotvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vdotvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`

- `int64xm1_t vdotvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vdotvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vdotvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vdotvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vdotvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vdotvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vdotvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vdotvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vdotvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vdotvv_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vdotvv_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vdotvv_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vdotvv_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vdotvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vdotvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vdotvv_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vdotvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vdotvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vdotvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vdotvv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vdotvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vdotvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vdotvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`

- `uint8xm8_t vdotvv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = dot-product(a[element], b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.17 Elementwise vector-vector multiply-addition, overwrite addend

Instruction: ['vmacc.vv']

Prototypes:

- `int16xm1_t vmaccvv_int16xm1 (int16xm1_t a, int16xm1_t b, int16xm1_t result, unsigned int gvl)`
- `int16xm2_t vmaccvv_int16xm2 (int16xm2_t a, int16xm2_t b, int16xm2_t result, unsigned int gvl)`
- `int16xm4_t vmaccvv_int16xm4 (int16xm4_t a, int16xm4_t b, int16xm4_t result, unsigned int gvl)`
- `int16xm8_t vmaccvv_int16xm8 (int16xm8_t a, int16xm8_t b, int16xm8_t result, unsigned int gvl)`
- `int32xm1_t vmaccvv_int32xm1 (int32xm1_t a, int32xm1_t b, int32xm1_t result, unsigned int gvl)`
- `int32xm2_t vmaccvv_int32xm2 (int32xm2_t a, int32xm2_t b, int32xm2_t result, unsigned int gvl)`
- `int32xm4_t vmaccvv_int32xm4 (int32xm4_t a, int32xm4_t b, int32xm4_t result, unsigned int gvl)`
- `int32xm8_t vmaccvv_int32xm8 (int32xm8_t a, int32xm8_t b, int32xm8_t result, unsigned int gvl)`
- `int64xm1_t vmaccvv_int64xm1 (int64xm1_t a, int64xm1_t b, int64xm1_t result, unsigned int gvl)`
- `int64xm2_t vmaccvv_int64xm2 (int64xm2_t a, int64xm2_t b, int64xm2_t result, unsigned int gvl)`
- `int64xm4_t vmaccvv_int64xm4 (int64xm4_t a, int64xm4_t b, int64xm4_t result, unsigned int gvl)`
- `int64xm8_t vmaccvv_int64xm8 (int64xm8_t a, int64xm8_t b, int64xm8_t result, unsigned int gvl)`
- `int8xm1_t vmaccvv_int8xm1 (int8xm1_t a, int8xm1_t b, int8xm1_t result, unsigned int gvl)`
- `int8xm2_t vmaccvv_int8xm2 (int8xm2_t a, int8xm2_t b, int8xm2_t result, unsigned int gvl)`
- `int8xm4_t vmaccvv_int8xm4 (int8xm4_t a, int8xm4_t b, int8xm4_t result, unsigned int gvl)`
- `int8xm8_t vmaccvv_int8xm8 (int8xm8_t a, int8xm8_t b, int8xm8_t result, unsigned int gvl)`
- `uint16xm1_t vmaccvv_uint16xm1 (uint16xm1_t a, uint16xm1_t b, uint16xm1_t result, unsigned int gvl)`
- `uint16xm2_t vmaccvv_uint16xm2 (uint16xm2_t a, uint16xm2_t b, uint16xm2_t result, unsigned int gvl)`
- `uint16xm4_t vmaccvv_uint16xm4 (uint16xm4_t a, uint16xm4_t b, uint16xm4_t result, unsigned int gvl)`
- `uint16xm8_t vmaccvv_uint16xm8 (uint16xm8_t a, uint16xm8_t b, uint16xm8_t result, unsigned int gvl)`
- `uint32xm1_t vmaccvv_uint32xm1 (uint32xm1_t a, uint32xm1_t b, uint32xm1_t result, unsigned int gvl)`
- `uint32xm2_t vmaccvv_uint32xm2 (uint32xm2_t a, uint32xm2_t b, uint32xm2_t result, unsigned int gvl)`

- `uint32xm4_t vmaccvv_uint32xm4 (uint32xm4_t a, uint32xm4_t b, uint32xm4_t result, unsigned int gvl)`
- `uint32xm8_t vmaccvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, uint32xm8_t result, unsigned int gvl)`
- `uint64xm1_t vmaccvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, uint64xm1_t result, unsigned int gvl)`
- `uint64xm2_t vmaccvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, uint64xm2_t result, unsigned int gvl)`
- `uint64xm4_t vmaccvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, uint64xm4_t result, unsigned int gvl)`
- `uint64xm8_t vmaccvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, uint64xm8_t result, unsigned int gvl)`
- `uint8xm1_t vmaccvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, uint8xm1_t result, unsigned int gvl)`
- `uint8xm2_t vmaccvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, uint8xm2_t result, unsigned int gvl)`
- `uint8xm4_t vmaccvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, uint8xm4_t result, unsigned int gvl)`
- `uint8xm8_t vmaccvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, uint8xm8_t result, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = +(a[element] * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vmaccvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, int16xm1_t result, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vmaccvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, int16xm2_t result, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vmaccvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, int16xm4_t result, e16xm4_t mask, unsigned int gvl)`
- `int32xm1_t vmaccvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, int32xm1_t result, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vmaccvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, int32xm2_t result, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vmaccvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, int32xm4_t result, e32xm4_t mask, unsigned int gvl)`
- `int64xm1_t vmaccvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, int64xm1_t result, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vmaccvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, int64xm2_t result, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vmaccvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, int64xm4_t result, e64xm4_t mask, unsigned int gvl)`
- `int8xm1_t vmaccvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, int8xm1_t result, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vmaccvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, int8xm2_t result, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vmaccvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, int8xm4_t result, e8xm4_t mask, unsigned int gvl)`

- `uint16xm1_t vmaccvv_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, `uint16xm1_t b`, `uint16xm1_t result`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint16xm2_t vmaccvv_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, `uint16xm2_t b`, `uint16xm2_t result`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint16xm4_t vmaccvv_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, `uint16xm4_t b`, `uint16xm4_t result`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint32xm1_t vmaccvv_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, `uint32xm1_t b`, `uint32xm1_t result`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vmaccvv_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, `uint32xm2_t b`, `uint32xm2_t result`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vmaccvv_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, `uint32xm4_t b`, `uint32xm4_t result`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint64xm1_t vmaccvv_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, `uint64xm1_t b`, `uint64xm1_t result`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vmaccvv_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, `uint64xm2_t b`, `uint64xm2_t result`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vmaccvv_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, `uint64xm4_t b`, `uint64xm4_t result`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint8xm1_t vmaccvv_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `uint8xm1_t result`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vmaccvv_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `uint8xm2_t result`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vmaccvv_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `uint8xm4_t result`, `e8xm4_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = +(a[element] * b[element]) + result[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.18 Elementwise vector-scalar multiply-addition, overwrite addend

Instruction: ['vmacc.vx']

Prototypes:

- `int16xm1_t vmaccvx_int16xm1` (short `a`, `int16xm1_t b`, `int16xm1_t result`, unsigned int `gvl`)
- `int16xm2_t vmaccvx_int16xm2` (short `a`, `int16xm2_t b`, `int16xm2_t result`, unsigned int `gvl`)
- `int16xm4_t vmaccvx_int16xm4` (short `a`, `int16xm4_t b`, `int16xm4_t result`, unsigned int `gvl`)
- `int16xm8_t vmaccvx_int16xm8` (short `a`, `int16xm8_t b`, `int16xm8_t result`, unsigned int `gvl`)
- `int32xm1_t vmaccvx_int32xm1` (int `a`, `int32xm1_t b`, `int32xm1_t result`, unsigned int `gvl`)
- `int32xm2_t vmaccvx_int32xm2` (int `a`, `int32xm2_t b`, `int32xm2_t result`, unsigned int `gvl`)
- `int32xm4_t vmaccvx_int32xm4` (int `a`, `int32xm4_t b`, `int32xm4_t result`, unsigned int `gvl`)
- `int32xm8_t vmaccvx_int32xm8` (int `a`, `int32xm8_t b`, `int32xm8_t result`, unsigned int `gvl`)



- `int64xm1_t vmaccvx_int64xm1` (long *a*, `int64xm1_t b`, `int64xm1_t result`, unsigned int *gvl*)
- `int64xm2_t vmaccvx_int64xm2` (long *a*, `int64xm2_t b`, `int64xm2_t result`, unsigned int *gvl*)
- `int64xm4_t vmaccvx_int64xm4` (long *a*, `int64xm4_t b`, `int64xm4_t result`, unsigned int *gvl*)
- `int64xm8_t vmaccvx_int64xm8` (long *a*, `int64xm8_t b`, `int64xm8_t result`, unsigned int *gvl*)
- `int8xm1_t vmaccvx_int8xm1` (signed char *a*, `int8xm1_t b`, `int8xm1_t result`, unsigned int *gvl*)
- `int8xm2_t vmaccvx_int8xm2` (signed char *a*, `int8xm2_t b`, `int8xm2_t result`, unsigned int *gvl*)
- `int8xm4_t vmaccvx_int8xm4` (signed char *a*, `int8xm4_t b`, `int8xm4_t result`, unsigned int *gvl*)
- `int8xm8_t vmaccvx_int8xm8` (signed char *a*, `int8xm8_t b`, `int8xm8_t result`, unsigned int *gvl*)
- `uint16xm1_t vmaccvx_uint16xm1` (unsigned short *a*, `uint16xm1_t b`, `uint16xm1_t result`, unsigned int *gvl*)
- `uint16xm2_t vmaccvx_uint16xm2` (unsigned short *a*, `uint16xm2_t b`, `uint16xm2_t result`, unsigned int *gvl*)
- `uint16xm4_t vmaccvx_uint16xm4` (unsigned short *a*, `uint16xm4_t b`, `uint16xm4_t result`, unsigned int *gvl*)
- `uint16xm8_t vmaccvx_uint16xm8` (unsigned short *a*, `uint16xm8_t b`, `uint16xm8_t result`, unsigned int *gvl*)
- `uint32xm1_t vmaccvx_uint32xm1` (unsigned int *a*, `uint32xm1_t b`, `uint32xm1_t result`, unsigned int *gvl*)
- `uint32xm2_t vmaccvx_uint32xm2` (unsigned int *a*, `uint32xm2_t b`, `uint32xm2_t result`, unsigned int *gvl*)
- `uint32xm4_t vmaccvx_uint32xm4` (unsigned int *a*, `uint32xm4_t b`, `uint32xm4_t result`, unsigned int *gvl*)
- `uint32xm8_t vmaccvx_uint32xm8` (unsigned int *a*, `uint32xm8_t b`, `uint32xm8_t result`, unsigned int *gvl*)
- `uint64xm1_t vmaccvx_uint64xm1` (unsigned long *a*, `uint64xm1_t b`, `uint64xm1_t result`, unsigned int *gvl*)
- `uint64xm2_t vmaccvx_uint64xm2` (unsigned long *a*, `uint64xm2_t b`, `uint64xm2_t result`, unsigned int *gvl*)
- `uint64xm4_t vmaccvx_uint64xm4` (unsigned long *a*, `uint64xm4_t b`, `uint64xm4_t result`, unsigned int *gvl*)
- `uint64xm8_t vmaccvx_uint64xm8` (unsigned long *a*, `uint64xm8_t b`, `uint64xm8_t result`, unsigned int *gvl*)
- `uint8xm1_t vmaccvx_uint8xm1` (unsigned char *a*, `uint8xm1_t b`, `uint8xm1_t result`, unsigned int *gvl*)
- `uint8xm2_t vmaccvx_uint8xm2` (unsigned char *a*, `uint8xm2_t b`, `uint8xm2_t result`, unsigned int *gvl*)
- `uint8xm4_t vmaccvx_uint8xm4` (unsigned char *a*, `uint8xm4_t b`, `uint8xm4_t result`, unsigned int *gvl*)
- `uint8xm8_t vmaccvx_uint8xm8` (unsigned char *a*, `uint8xm8_t b`, `uint8xm8_t result`, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = +(a * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vmaccvx\_mask\_int16xm1* (*int16xm1\_t* merge, short *a*, *int16xm1\_t* *b*, *int16xm1\_t* *result*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *int16xm2\_t vmaccvx\_mask\_int16xm2* (*int16xm2\_t* merge, short *a*, *int16xm2\_t* *b*, *int16xm2\_t* *result*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *int16xm4\_t vmaccvx\_mask\_int16xm4* (*int16xm4\_t* merge, short *a*, *int16xm4\_t* *b*, *int16xm4\_t* *result*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *int32xm1\_t vmaccvx\_mask\_int32xm1* (*int32xm1\_t* merge, int *a*, *int32xm1\_t* *b*, *int32xm1\_t* *result*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *int32xm2\_t vmaccvx\_mask\_int32xm2* (*int32xm2\_t* merge, int *a*, *int32xm2\_t* *b*, *int32xm2\_t* *result*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *int32xm4\_t vmaccvx\_mask\_int32xm4* (*int32xm4\_t* merge, int *a*, *int32xm4\_t* *b*, *int32xm4\_t* *result*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *int64xm1\_t vmaccvx\_mask\_int64xm1* (*int64xm1\_t* merge, long *a*, *int64xm1\_t* *b*, *int64xm1\_t* *result*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *int64xm2\_t vmaccvx\_mask\_int64xm2* (*int64xm2\_t* merge, long *a*, *int64xm2\_t* *b*, *int64xm2\_t* *result*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *int64xm4\_t vmaccvx\_mask\_int64xm4* (*int64xm4\_t* merge, long *a*, *int64xm4\_t* *b*, *int64xm4\_t* *result*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *int8xm1\_t vmaccvx\_mask\_int8xm1* (*int8xm1\_t* merge, signed char *a*, *int8xm1\_t* *b*, *int8xm1\_t* *result*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *int8xm2\_t vmaccvx\_mask\_int8xm2* (*int8xm2\_t* merge, signed char *a*, *int8xm2\_t* *b*, *int8xm2\_t* *result*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- *int8xm4\_t vmaccvx\_mask\_int8xm4* (*int8xm4\_t* merge, signed char *a*, *int8xm4\_t* *b*, *int8xm4\_t* *result*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- *uint16xm1\_t vmaccvx\_mask\_uint16xm1* (*uint16xm1\_t* merge, unsigned short *a*, *uint16xm1\_t* *b*, *uint16xm1\_t* *result*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *uint16xm2\_t vmaccvx\_mask\_uint16xm2* (*uint16xm2\_t* merge, unsigned short *a*, *uint16xm2\_t* *b*, *uint16xm2\_t* *result*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *uint16xm4\_t vmaccvx\_mask\_uint16xm4* (*uint16xm4\_t* merge, unsigned short *a*, *uint16xm4\_t* *b*, *uint16xm4\_t* *result*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *uint32xm1\_t vmaccvx\_mask\_uint32xm1* (*uint32xm1\_t* merge, unsigned int *a*, *uint32xm1\_t* *b*, *uint32xm1\_t* *result*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *uint32xm2\_t vmaccvx\_mask\_uint32xm2* (*uint32xm2\_t* merge, unsigned int *a*, *uint32xm2\_t* *b*, *uint32xm2\_t* *result*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *uint32xm4\_t vmaccvx\_mask\_uint32xm4* (*uint32xm4\_t* merge, unsigned int *a*, *uint32xm4\_t* *b*, *uint32xm4\_t* *result*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *uint64xm1\_t vmaccvx\_mask\_uint64xm1* (*uint64xm1\_t* merge, unsigned long *a*, *uint64xm1\_t* *b*, *uint64xm1\_t* *result*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *uint64xm2\_t vmaccvx\_mask\_uint64xm2* (*uint64xm2\_t* merge, unsigned long *a*, *uint64xm2\_t* *b*, *uint64xm2\_t* *result*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *uint64xm4\_t vmaccvx\_mask\_uint64xm4* (*uint64xm4\_t* merge, unsigned long *a*, *uint64xm4\_t* *b*, *uint64xm4\_t* *result*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *uint8xm1\_t vmaccvx\_mask\_uint8xm1* (*uint8xm1\_t* merge, unsigned char *a*, *uint8xm1\_t* *b*, *uint8xm1\_t* *result*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *uint8xm2\_t vmaccvx\_mask\_uint8xm2* (*uint8xm2\_t* merge, unsigned char *a*, *uint8xm2\_t* *b*, *uint8xm2\_t* *result*, *e8xm2\_t* *mask*, unsigned int *gvl*)

- `uint8xm4_t vmaccvx_mask_uint8xm4(uint8xm4_t merge, unsigned char a, uint8xm4_t b, uint8xm4_t result, e8xm4_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = +(a * b[element]) + result[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.19 Elementwise vector-immediate integer addition with carry in mask register format

Instruction: ['vmadc.vim']

Masked prototypes:

- `e16xm1_t vmadcvim_mask_e16xm1_int16xm1(int16xm1_t a, const int b, e16xm1_t mask, unsigned int gvl)`
- `e16xm1_t vmadcvim_mask_e16xm1_uint16xm1(uint16xm1_t a, const int b, e16xm1_t mask, unsigned int gvl)`
- `e16xm2_t vmadcvim_mask_e16xm2_int16xm2(int16xm2_t a, const int b, e16xm2_t mask, unsigned int gvl)`
- `e16xm2_t vmadcvim_mask_e16xm2_uint16xm2(uint16xm2_t a, const int b, e16xm2_t mask, unsigned int gvl)`
- `e16xm4_t vmadcvim_mask_e16xm4_int16xm4(int16xm4_t a, const int b, e16xm4_t mask, unsigned int gvl)`
- `e16xm4_t vmadcvim_mask_e16xm4_uint16xm4(uint16xm4_t a, const int b, e16xm4_t mask, unsigned int gvl)`
- `e16xm8_t vmadcvim_mask_e16xm8_int16xm8(int16xm8_t a, const int b, e16xm8_t mask, unsigned int gvl)`
- `e16xm8_t vmadcvim_mask_e16xm8_uint16xm8(uint16xm8_t a, const int b, e16xm8_t mask, unsigned int gvl)`
- `e32xm1_t vmadcvim_mask_e32xm1_int32xm1(int32xm1_t a, const int b, e32xm1_t mask, unsigned int gvl)`
- `e32xm1_t vmadcvim_mask_e32xm1_uint32xm1(uint32xm1_t a, const int b, e32xm1_t mask, unsigned int gvl)`
- `e32xm2_t vmadcvim_mask_e32xm2_int32xm2(int32xm2_t a, const int b, e32xm2_t mask, unsigned int gvl)`
- `e32xm2_t vmadcvim_mask_e32xm2_uint32xm2(uint32xm2_t a, const int b, e32xm2_t mask, unsigned int gvl)`
- `e32xm4_t vmadcvim_mask_e32xm4_int32xm4(int32xm4_t a, const int b, e32xm4_t mask, unsigned int gvl)`
- `e32xm4_t vmadcvim_mask_e32xm4_uint32xm4(uint32xm4_t a, const int b, e32xm4_t mask, unsigned int gvl)`
- `e32xm8_t vmadcvim_mask_e32xm8_int32xm8(int32xm8_t a, const int b, e32xm8_t mask, unsigned int gvl)`

- `e32xm8_t vmadcvm_mask_e32xm8_uint32xm8 (uint32xm8_t a, const int b, e32xm8_t mask, unsigned int gvl)`
- `e64xm1_t vmadcvm_mask_e64xm1_int64xm1 (int64xm1_t a, const int b, e64xm1_t mask, unsigned int gvl)`
- `e64xm1_t vmadcvm_mask_e64xm1_uint64xm1 (uint64xm1_t a, const int b, e64xm1_t mask, unsigned int gvl)`
- `e64xm2_t vmadcvm_mask_e64xm2_int64xm2 (int64xm2_t a, const int b, e64xm2_t mask, unsigned int gvl)`
- `e64xm2_t vmadcvm_mask_e64xm2_uint64xm2 (uint64xm2_t a, const int b, e64xm2_t mask, unsigned int gvl)`
- `e64xm4_t vmadcvm_mask_e64xm4_int64xm4 (int64xm4_t a, const int b, e64xm4_t mask, unsigned int gvl)`
- `e64xm4_t vmadcvm_mask_e64xm4_uint64xm4 (uint64xm4_t a, const int b, e64xm4_t mask, unsigned int gvl)`
- `e64xm8_t vmadcvm_mask_e64xm8_int64xm8 (int64xm8_t a, const int b, e64xm8_t mask, unsigned int gvl)`
- `e64xm8_t vmadcvm_mask_e64xm8_uint64xm8 (uint64xm8_t a, const int b, e64xm8_t mask, unsigned int gvl)`
- `e8xm1_t vmadcvm_mask_e8xm1_int8xm1 (int8xm1_t a, const int b, e8xm1_t mask, unsigned int gvl)`
- `e8xm1_t vmadcvm_mask_e8xm1_uint8xm1 (uint8xm1_t a, const int b, e8xm1_t mask, unsigned int gvl)`
- `e8xm2_t vmadcvm_mask_e8xm2_int8xm2 (int8xm2_t a, const int b, e8xm2_t mask, unsigned int gvl)`
- `e8xm2_t vmadcvm_mask_e8xm2_uint8xm2 (uint8xm2_t a, const int b, e8xm2_t mask, unsigned int gvl)`
- `e8xm4_t vmadcvm_mask_e8xm4_int8xm4 (int8xm4_t a, const int b, e8xm4_t mask, unsigned int gvl)`
- `e8xm4_t vmadcvm_mask_e8xm4_uint8xm4 (uint8xm4_t a, const int b, e8xm4_t mask, unsigned int gvl)`
- `e8xm8_t vmadcvm_mask_e8xm8_int8xm8 (int8xm8_t a, const int b, e8xm8_t mask, unsigned int gvl)`
- `e8xm8_t vmadcvm_mask_e8xm8_uint8xm8 (uint8xm8_t a, const int b, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
    result[element] = carry_out(a[element] + b + mask[elemet])
result[gvl : VLMAX] = 0
```

## 2.7.20 Elementwise vector-vector integer addition with carry in mask register format

**Instruction:** [`'vmadc.vvm'`]

**Masked prototypes:**

- `e16xm1_t vmadcvm_mask_e16xm1_int16xm1 (int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`

- `e16xm1_t vmadcvvm_mask_e16xm1_uint16xm1 (uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `e16xm2_t vmadcvvm_mask_e16xm2_int16xm2 (int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `e16xm2_t vmadcvvm_mask_e16xm2_uint16xm2 (uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `e16xm4_t vmadcvvm_mask_e16xm4_int16xm4 (int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `e16xm4_t vmadcvvm_mask_e16xm4_uint16xm4 (uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `e16xm8_t vmadcvvm_mask_e16xm8_int16xm8 (int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `e16xm8_t vmadcvvm_mask_e16xm8_uint16xm8 (uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `e32xm1_t vmadcvvm_mask_e32xm1_int32xm1 (int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `e32xm1_t vmadcvvm_mask_e32xm1_uint32xm1 (uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `e32xm2_t vmadcvvm_mask_e32xm2_int32xm2 (int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `e32xm2_t vmadcvvm_mask_e32xm2_uint32xm2 (uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `e32xm4_t vmadcvvm_mask_e32xm4_int32xm4 (int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `e32xm4_t vmadcvvm_mask_e32xm4_uint32xm4 (uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `e32xm8_t vmadcvvm_mask_e32xm8_int32xm8 (int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `e32xm8_t vmadcvvm_mask_e32xm8_uint32xm8 (uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `e64xm1_t vmadcvvm_mask_e64xm1_int64xm1 (int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `e64xm1_t vmadcvvm_mask_e64xm1_uint64xm1 (uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `e64xm2_t vmadcvvm_mask_e64xm2_int64xm2 (int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `e64xm2_t vmadcvvm_mask_e64xm2_uint64xm2 (uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `e64xm4_t vmadcvvm_mask_e64xm4_int64xm4 (int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `e64xm4_t vmadcvvm_mask_e64xm4_uint64xm4 (uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `e64xm8_t vmadcvvm_mask_e64xm8_int64xm8 (int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `e64xm8_t vmadcvvm_mask_e64xm8_uint64xm8 (uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`

- `e8xm1_t vmadcvvm_mask_e8xm1_int8xm1` (`int8xm1_t a`, `int8xm1_t b`, `e8xm1_t mask`, unsigned int `gvl`)
- `e8xm1_t vmadcvvm_mask_e8xm1_uint8xm1` (`uint8xm1_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int `gvl`)
- `e8xm2_t vmadcvvm_mask_e8xm2_int8xm2` (`int8xm2_t a`, `int8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `e8xm2_t vmadcvvm_mask_e8xm2_uint8xm2` (`uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `e8xm4_t vmadcvvm_mask_e8xm4_int8xm4` (`int8xm4_t a`, `int8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `e8xm4_t vmadcvvm_mask_e8xm4_uint8xm4` (`uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `e8xm8_t vmadcvvm_mask_e8xm8_int8xm8` (`int8xm8_t a`, `int8xm8_t b`, `e8xm8_t mask`, unsigned int `gvl`)
- `e8xm8_t vmadcvvm_mask_e8xm8_uint8xm8` (`uint8xm8_t a`, `uint8xm8_t b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
    result[element] = carry_out(a[element] + b[element] + mask[elemet])
result[gvl : VLMAX] = 0
```

## 2.7.21 Elementwise vector-scalar integer addition with carry in mask register format

Instruction: [`'vmadc.vxm'`]

Masked prototypes:

- `e16xm1_t vmadcvxm_mask_e16xm1_int16xm1` (`int16xm1_t a`, short `b`, `e16xm1_t mask`, unsigned int `gvl`)
- `e16xm1_t vmadcvxm_mask_e16xm1_uint16xm1` (`uint16xm1_t a`, unsigned short `b`, `e16xm1_t mask`, unsigned int `gvl`)
- `e16xm2_t vmadcvxm_mask_e16xm2_int16xm2` (`int16xm2_t a`, short `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `e16xm2_t vmadcvxm_mask_e16xm2_uint16xm2` (`uint16xm2_t a`, unsigned short `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `e16xm4_t vmadcvxm_mask_e16xm4_int16xm4` (`int16xm4_t a`, short `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `e16xm4_t vmadcvxm_mask_e16xm4_uint16xm4` (`uint16xm4_t a`, unsigned short `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `e16xm8_t vmadcvxm_mask_e16xm8_int16xm8` (`int16xm8_t a`, short `b`, `e16xm8_t mask`, unsigned int `gvl`)
- `e16xm8_t vmadcvxm_mask_e16xm8_uint16xm8` (`uint16xm8_t a`, unsigned short `b`, `e16xm8_t mask`, unsigned int `gvl`)
- `e32xm1_t vmadcvxm_mask_e32xm1_int32xm1` (`int32xm1_t a`, int `b`, `e32xm1_t mask`, unsigned int `gvl`)
- `e32xm1_t vmadcvxm_mask_e32xm1_uint32xm1` (`uint32xm1_t a`, unsigned int `b`, `e32xm1_t mask`, unsigned int `gvl`)



- `e32xm2_t vmadcvxm_mask_e32xm2_int32xm2` (`int32xm2_t a`, `int b`, `e32xm2_t mask`, unsigned int gvl)
- `e32xm2_t vmadcvxm_mask_e32xm2_uint32xm2` (`uint32xm2_t a`, unsigned int `b`, `e32xm2_t mask`, unsigned int gvl)
- `e32xm4_t vmadcvxm_mask_e32xm4_int32xm4` (`int32xm4_t a`, `int b`, `e32xm4_t mask`, unsigned int gvl)
- `e32xm4_t vmadcvxm_mask_e32xm4_uint32xm4` (`uint32xm4_t a`, unsigned int `b`, `e32xm4_t mask`, unsigned int gvl)
- `e32xm8_t vmadcvxm_mask_e32xm8_int32xm8` (`int32xm8_t a`, `int b`, `e32xm8_t mask`, unsigned int gvl)
- `e32xm8_t vmadcvxm_mask_e32xm8_uint32xm8` (`uint32xm8_t a`, unsigned int `b`, `e32xm8_t mask`, unsigned int gvl)
- `e64xm1_t vmadcvxm_mask_e64xm1_int64xm1` (`int64xm1_t a`, long `b`, `e64xm1_t mask`, unsigned int gvl)
- `e64xm1_t vmadcvxm_mask_e64xm1_uint64xm1` (`uint64xm1_t a`, unsigned long `b`, `e64xm1_t mask`, unsigned int gvl)
- `e64xm2_t vmadcvxm_mask_e64xm2_int64xm2` (`int64xm2_t a`, long `b`, `e64xm2_t mask`, unsigned int gvl)
- `e64xm2_t vmadcvxm_mask_e64xm2_uint64xm2` (`uint64xm2_t a`, unsigned long `b`, `e64xm2_t mask`, unsigned int gvl)
- `e64xm4_t vmadcvxm_mask_e64xm4_int64xm4` (`int64xm4_t a`, long `b`, `e64xm4_t mask`, unsigned int gvl)
- `e64xm4_t vmadcvxm_mask_e64xm4_uint64xm4` (`uint64xm4_t a`, unsigned long `b`, `e64xm4_t mask`, unsigned int gvl)
- `e64xm8_t vmadcvxm_mask_e64xm8_int64xm8` (`int64xm8_t a`, long `b`, `e64xm8_t mask`, unsigned int gvl)
- `e64xm8_t vmadcvxm_mask_e64xm8_uint64xm8` (`uint64xm8_t a`, unsigned long `b`, `e64xm8_t mask`, unsigned int gvl)
- `e8xm1_t vmadcvxm_mask_e8xm1_int8xm1` (`int8xm1_t a`, signed char `b`, `e8xm1_t mask`, unsigned int gvl)
- `e8xm1_t vmadcvxm_mask_e8xm1_uint8xm1` (`uint8xm1_t a`, unsigned char `b`, `e8xm1_t mask`, unsigned int gvl)
- `e8xm2_t vmadcvxm_mask_e8xm2_int8xm2` (`int8xm2_t a`, signed char `b`, `e8xm2_t mask`, unsigned int gvl)
- `e8xm2_t vmadcvxm_mask_e8xm2_uint8xm2` (`uint8xm2_t a`, unsigned char `b`, `e8xm2_t mask`, unsigned int gvl)
- `e8xm4_t vmadcvxm_mask_e8xm4_int8xm4` (`int8xm4_t a`, signed char `b`, `e8xm4_t mask`, unsigned int gvl)
- `e8xm4_t vmadcvxm_mask_e8xm4_uint8xm4` (`uint8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int gvl)
- `e8xm8_t vmadcvxm_mask_e8xm8_int8xm8` (`int8xm8_t a`, signed char `b`, `e8xm8_t mask`, unsigned int gvl)
- `e8xm8_t vmadcvxm_mask_e8xm8_uint8xm8` (`uint8xm8_t a`, unsigned char `b`, `e8xm8_t mask`, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      result[element] = carry_out(a[element] + b + mask[elemet])
      result[gvl : VLMAX] = 0
```

## 2.7.22 Elementwise vector-vector multiply-addition, overwrite multiplicand

**Instruction:** [`vmadd.vv`]

**Prototypes:**

- `int16xm1_t vmaddvv_int16xm1 (int16xm1_t a, int16xm1_t b, int16xm1_t result, unsigned int gvl)`
- `int16xm2_t vmaddvv_int16xm2 (int16xm2_t a, int16xm2_t b, int16xm2_t result, unsigned int gvl)`
- `int16xm4_t vmaddvv_int16xm4 (int16xm4_t a, int16xm4_t b, int16xm4_t result, unsigned int gvl)`
- `int16xm8_t vmaddvv_int16xm8 (int16xm8_t a, int16xm8_t b, int16xm8_t result, unsigned int gvl)`
- `int32xm1_t vmaddvv_int32xm1 (int32xm1_t a, int32xm1_t b, int32xm1_t result, unsigned int gvl)`
- `int32xm2_t vmaddvv_int32xm2 (int32xm2_t a, int32xm2_t b, int32xm2_t result, unsigned int gvl)`
- `int32xm4_t vmaddvv_int32xm4 (int32xm4_t a, int32xm4_t b, int32xm4_t result, unsigned int gvl)`
- `int32xm8_t vmaddvv_int32xm8 (int32xm8_t a, int32xm8_t b, int32xm8_t result, unsigned int gvl)`
- `int64xm1_t vmaddvv_int64xm1 (int64xm1_t a, int64xm1_t b, int64xm1_t result, unsigned int gvl)`
- `int64xm2_t vmaddvv_int64xm2 (int64xm2_t a, int64xm2_t b, int64xm2_t result, unsigned int gvl)`
- `int64xm4_t vmaddvv_int64xm4 (int64xm4_t a, int64xm4_t b, int64xm4_t result, unsigned int gvl)`
- `int64xm8_t vmaddvv_int64xm8 (int64xm8_t a, int64xm8_t b, int64xm8_t result, unsigned int gvl)`
- `int8xm1_t vmaddvv_int8xm1 (int8xm1_t a, int8xm1_t b, int8xm1_t result, unsigned int gvl)`
- `int8xm2_t vmaddvv_int8xm2 (int8xm2_t a, int8xm2_t b, int8xm2_t result, unsigned int gvl)`
- `int8xm4_t vmaddvv_int8xm4 (int8xm4_t a, int8xm4_t b, int8xm4_t result, unsigned int gvl)`
- `int8xm8_t vmaddvv_int8xm8 (int8xm8_t a, int8xm8_t b, int8xm8_t result, unsigned int gvl)`
- `uint16xm1_t vmaddvv_uint16xm1 (uint16xm1_t a, uint16xm1_t b, uint16xm1_t result, unsigned int gvl)`
- `uint16xm2_t vmaddvv_uint16xm2 (uint16xm2_t a, uint16xm2_t b, uint16xm2_t result, unsigned int gvl)`
- `uint16xm4_t vmaddvv_uint16xm4 (uint16xm4_t a, uint16xm4_t b, uint16xm4_t result, unsigned int gvl)`
- `uint16xm8_t vmaddvv_uint16xm8 (uint16xm8_t a, uint16xm8_t b, uint16xm8_t result, unsigned int gvl)`
- `uint32xm1_t vmaddvv_uint32xm1 (uint32xm1_t a, uint32xm1_t b, uint32xm1_t result, unsigned int gvl)`
- `uint32xm2_t vmaddvv_uint32xm2 (uint32xm2_t a, uint32xm2_t b, uint32xm2_t result, unsigned int gvl)`
- `uint32xm4_t vmaddvv_uint32xm4 (uint32xm4_t a, uint32xm4_t b, uint32xm4_t result, unsigned int gvl)`
- `uint32xm8_t vmaddvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, uint32xm8_t result, unsigned int gvl)`

- `uint64xm1_t vmaddvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, uint64xm1_t result, unsigned int gvl)`
- `uint64xm2_t vmaddvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, uint64xm2_t result, unsigned int gvl)`
- `uint64xm4_t vmaddvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, uint64xm4_t result, unsigned int gvl)`
- `uint64xm8_t vmaddvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, uint64xm8_t result, unsigned int gvl)`
- `uint8xm1_t vmaddvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, uint8xm1_t result, unsigned int gvl)`
- `uint8xm2_t vmaddvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, uint8xm2_t result, unsigned int gvl)`
- `uint8xm4_t vmaddvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, uint8xm4_t result, unsigned int gvl)`
- `uint8xm8_t vmaddvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, uint8xm8_t result, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = +(a[element] * result[element]) + b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vmaddvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, int16xm1_t result, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vmaddvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, int16xm2_t result, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vmaddvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, int16xm4_t result, e16xm4_t mask, unsigned int gvl)`
- `int32xm1_t vmaddvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, int32xm1_t result, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vmaddvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, int32xm2_t result, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vmaddvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, int32xm4_t result, e32xm4_t mask, unsigned int gvl)`
- `int64xm1_t vmaddvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, int64xm1_t result, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vmaddvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, int64xm2_t result, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vmaddvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, int64xm4_t result, e64xm4_t mask, unsigned int gvl)`
- `int8xm1_t vmaddvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, int8xm1_t result, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vmaddvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, int8xm2_t result, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vmaddvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, int8xm4_t result, e8xm4_t mask, unsigned int gvl)`
- `uint16xm1_t vmaddvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, uint16xm1_t result, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vmaddvv_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, uint16xm2_t result, e16xm2_t mask, unsigned int gvl)`

- `uint16xm4_t vmaddvv_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, uint16xm4_t result, e16xm4_t mask, unsigned int gvl)`
- `uint32xm1_t vmaddvv_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, uint32xm1_t result, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vmaddvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, uint32xm2_t result, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vmaddvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, uint32xm4_t result, e32xm4_t mask, unsigned int gvl)`
- `uint64xm1_t vmaddvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, uint64xm1_t result, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vmaddvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, uint64xm2_t result, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vmaddvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, uint64xm4_t result, e64xm4_t mask, unsigned int gvl)`
- `uint8xm1_t vmaddvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, uint8xm1_t result, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vmaddvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, uint8xm2_t result, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vmaddvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, uint8xm4_t result, e8xm4_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = +(a[element] * result[element]) + b[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

### 2.7.23 Elementwise vector-scalar multiply-addition, overwrite multiplicand

Instruction: [`vmadd.vx`]

Prototypes:

- `int16xm1_t vmaddvx_int16xm1 (short a, int16xm1_t b, int16xm1_t result, unsigned int gvl)`
- `int16xm2_t vmaddvx_int16xm2 (short a, int16xm2_t b, int16xm2_t result, unsigned int gvl)`
- `int16xm4_t vmaddvx_int16xm4 (short a, int16xm4_t b, int16xm4_t result, unsigned int gvl)`
- `int16xm8_t vmaddvx_int16xm8 (short a, int16xm8_t b, int16xm8_t result, unsigned int gvl)`
- `int32xm1_t vmaddvx_int32xm1 (int a, int32xm1_t b, int32xm1_t result, unsigned int gvl)`
- `int32xm2_t vmaddvx_int32xm2 (int a, int32xm2_t b, int32xm2_t result, unsigned int gvl)`
- `int32xm4_t vmaddvx_int32xm4 (int a, int32xm4_t b, int32xm4_t result, unsigned int gvl)`
- `int32xm8_t vmaddvx_int32xm8 (int a, int32xm8_t b, int32xm8_t result, unsigned int gvl)`
- `int64xm1_t vmaddvx_int64xm1 (long a, int64xm1_t b, int64xm1_t result, unsigned int gvl)`
- `int64xm2_t vmaddvx_int64xm2 (long a, int64xm2_t b, int64xm2_t result, unsigned int gvl)`
- `int64xm4_t vmaddvx_int64xm4 (long a, int64xm4_t b, int64xm4_t result, unsigned int gvl)`

- *int64xm8\_t vmaddvx\_int64xm8* (long *a*, *int64xm8\_t b*, *int64xm8\_t result*, unsigned int *gvl*)
- *int8xm1\_t vmaddvx\_int8xm1* (signed char *a*, *int8xm1\_t b*, *int8xm1\_t result*, unsigned int *gvl*)
- *int8xm2\_t vmaddvx\_int8xm2* (signed char *a*, *int8xm2\_t b*, *int8xm2\_t result*, unsigned int *gvl*)
- *int8xm4\_t vmaddvx\_int8xm4* (signed char *a*, *int8xm4\_t b*, *int8xm4\_t result*, unsigned int *gvl*)
- *int8xm8\_t vmaddvx\_int8xm8* (signed char *a*, *int8xm8\_t b*, *int8xm8\_t result*, unsigned int *gvl*)
- *uint16xm1\_t vmaddvx\_uint16xm1* (unsigned short *a*, *uint16xm1\_t b*, *uint16xm1\_t result*, unsigned int *gvl*)
- *uint16xm2\_t vmaddvx\_uint16xm2* (unsigned short *a*, *uint16xm2\_t b*, *uint16xm2\_t result*, unsigned int *gvl*)
- *uint16xm4\_t vmaddvx\_uint16xm4* (unsigned short *a*, *uint16xm4\_t b*, *uint16xm4\_t result*, unsigned int *gvl*)
- *uint16xm8\_t vmaddvx\_uint16xm8* (unsigned short *a*, *uint16xm8\_t b*, *uint16xm8\_t result*, unsigned int *gvl*)
- *uint32xm1\_t vmaddvx\_uint32xm1* (unsigned int *a*, *uint32xm1\_t b*, *uint32xm1\_t result*, unsigned int *gvl*)
- *uint32xm2\_t vmaddvx\_uint32xm2* (unsigned int *a*, *uint32xm2\_t b*, *uint32xm2\_t result*, unsigned int *gvl*)
- *uint32xm4\_t vmaddvx\_uint32xm4* (unsigned int *a*, *uint32xm4\_t b*, *uint32xm4\_t result*, unsigned int *gvl*)
- *uint32xm8\_t vmaddvx\_uint32xm8* (unsigned int *a*, *uint32xm8\_t b*, *uint32xm8\_t result*, unsigned int *gvl*)
- *uint64xm1\_t vmaddvx\_uint64xm1* (unsigned long *a*, *uint64xm1\_t b*, *uint64xm1\_t result*, unsigned int *gvl*)
- *uint64xm2\_t vmaddvx\_uint64xm2* (unsigned long *a*, *uint64xm2\_t b*, *uint64xm2\_t result*, unsigned int *gvl*)
- *uint64xm4\_t vmaddvx\_uint64xm4* (unsigned long *a*, *uint64xm4\_t b*, *uint64xm4\_t result*, unsigned int *gvl*)
- *uint64xm8\_t vmaddvx\_uint64xm8* (unsigned long *a*, *uint64xm8\_t b*, *uint64xm8\_t result*, unsigned int *gvl*)
- *uint8xm1\_t vmaddvx\_uint8xm1* (unsigned char *a*, *uint8xm1\_t b*, *uint8xm1\_t result*, unsigned int *gvl*)
- *uint8xm2\_t vmaddvx\_uint8xm2* (unsigned char *a*, *uint8xm2\_t b*, *uint8xm2\_t result*, unsigned int *gvl*)
- *uint8xm4\_t vmaddvx\_uint8xm4* (unsigned char *a*, *uint8xm4\_t b*, *uint8xm4\_t result*, unsigned int *gvl*)
- *uint8xm8\_t vmaddvx\_uint8xm8* (unsigned char *a*, *uint8xm8\_t b*, *uint8xm8\_t result*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = +(a * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vmaddvx\_mask\_int16xm1* (*int16xm1\_t merge*, short *a*, *int16xm1\_t b*, *int16xm1\_t result*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vmaddvx\_mask\_int16xm2* (*int16xm2\_t merge*, short *a*, *int16xm2\_t b*, *int16xm2\_t result*, *e16xm2\_t mask*, unsigned int *gvl*)



- `int16xm4_t vmaddvx_mask_int16xm4` (`int16xm4_t` merge, short `a`, `int16xm4_t` `b`, `int16xm4_t` result, `e16xm4_t` mask, unsigned int gvl)
- `int32xm1_t vmaddvx_mask_int32xm1` (`int32xm1_t` merge, int `a`, `int32xm1_t` `b`, `int32xm1_t` result, `e32xm1_t` mask, unsigned int gvl)
- `int32xm2_t vmaddvx_mask_int32xm2` (`int32xm2_t` merge, int `a`, `int32xm2_t` `b`, `int32xm2_t` result, `e32xm2_t` mask, unsigned int gvl)
- `int32xm4_t vmaddvx_mask_int32xm4` (`int32xm4_t` merge, int `a`, `int32xm4_t` `b`, `int32xm4_t` result, `e32xm4_t` mask, unsigned int gvl)
- `int64xm1_t vmaddvx_mask_int64xm1` (`int64xm1_t` merge, long `a`, `int64xm1_t` `b`, `int64xm1_t` result, `e64xm1_t` mask, unsigned int gvl)
- `int64xm2_t vmaddvx_mask_int64xm2` (`int64xm2_t` merge, long `a`, `int64xm2_t` `b`, `int64xm2_t` result, `e64xm2_t` mask, unsigned int gvl)
- `int64xm4_t vmaddvx_mask_int64xm4` (`int64xm4_t` merge, long `a`, `int64xm4_t` `b`, `int64xm4_t` result, `e64xm4_t` mask, unsigned int gvl)
- `int8xm1_t vmaddvx_mask_int8xm1` (`int8xm1_t` merge, signed char `a`, `int8xm1_t` `b`, `int8xm1_t` result, `e8xm1_t` mask, unsigned int gvl)
- `int8xm2_t vmaddvx_mask_int8xm2` (`int8xm2_t` merge, signed char `a`, `int8xm2_t` `b`, `int8xm2_t` result, `e8xm2_t` mask, unsigned int gvl)
- `int8xm4_t vmaddvx_mask_int8xm4` (`int8xm4_t` merge, signed char `a`, `int8xm4_t` `b`, `int8xm4_t` result, `e8xm4_t` mask, unsigned int gvl)
- `uint16xm1_t vmaddvx_mask_uint16xm1` (`uint16xm1_t` merge, unsigned short `a`, `uint16xm1_t` `b`, `uint16xm1_t` result, `e16xm1_t` mask, unsigned int gvl)
- `uint16xm2_t vmaddvx_mask_uint16xm2` (`uint16xm2_t` merge, unsigned short `a`, `uint16xm2_t` `b`, `uint16xm2_t` result, `e16xm2_t` mask, unsigned int gvl)
- `uint16xm4_t vmaddvx_mask_uint16xm4` (`uint16xm4_t` merge, unsigned short `a`, `uint16xm4_t` `b`, `uint16xm4_t` result, `e16xm4_t` mask, unsigned int gvl)
- `uint32xm1_t vmaddvx_mask_uint32xm1` (`uint32xm1_t` merge, unsigned int `a`, `uint32xm1_t` `b`, `uint32xm1_t` result, `e32xm1_t` mask, unsigned int gvl)
- `uint32xm2_t vmaddvx_mask_uint32xm2` (`uint32xm2_t` merge, unsigned int `a`, `uint32xm2_t` `b`, `uint32xm2_t` result, `e32xm2_t` mask, unsigned int gvl)
- `uint32xm4_t vmaddvx_mask_uint32xm4` (`uint32xm4_t` merge, unsigned int `a`, `uint32xm4_t` `b`, `uint32xm4_t` result, `e32xm4_t` mask, unsigned int gvl)
- `uint64xm1_t vmaddvx_mask_uint64xm1` (`uint64xm1_t` merge, unsigned long `a`, `uint64xm1_t` `b`, `uint64xm1_t` result, `e64xm1_t` mask, unsigned int gvl)
- `uint64xm2_t vmaddvx_mask_uint64xm2` (`uint64xm2_t` merge, unsigned long `a`, `uint64xm2_t` `b`, `uint64xm2_t` result, `e64xm2_t` mask, unsigned int gvl)
- `uint64xm4_t vmaddvx_mask_uint64xm4` (`uint64xm4_t` merge, unsigned long `a`, `uint64xm4_t` `b`, `uint64xm4_t` result, `e64xm4_t` mask, unsigned int gvl)
- `uint8xm1_t vmaddvx_mask_uint8xm1` (`uint8xm1_t` merge, unsigned char `a`, `uint8xm1_t` `b`, `uint8xm1_t` result, `e8xm1_t` mask, unsigned int gvl)
- `uint8xm2_t vmaddvx_mask_uint8xm2` (`uint8xm2_t` merge, unsigned char `a`, `uint8xm2_t` `b`, `uint8xm2_t` result, `e8xm2_t` mask, unsigned int gvl)
- `uint8xm4_t vmaddvx_mask_uint8xm4` (`uint8xm4_t` merge, unsigned char `a`, `uint8xm4_t` `b`, `uint8xm4_t` result, `e8xm4_t` mask, unsigned int gvl)

Masked operation:



```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = +(a * b[element]) + result[element]
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.7.24 Elementwise signed vector-vector maximum

**Instruction:** ['vmax.vv']

**Prototypes:**

- *int16xm1\_t vmaxvv\_int16xm1* (*int16xm1\_t a*, *int16xm1\_t b*, unsigned int *gvl*)
- *int16xm2\_t vmaxvv\_int16xm2* (*int16xm2\_t a*, *int16xm2\_t b*, unsigned int *gvl*)
- *int16xm4\_t vmaxvv\_int16xm4* (*int16xm4\_t a*, *int16xm4\_t b*, unsigned int *gvl*)
- *int16xm8\_t vmaxvv\_int16xm8* (*int16xm8\_t a*, *int16xm8\_t b*, unsigned int *gvl*)
- *int32xm1\_t vmaxvv\_int32xm1* (*int32xm1\_t a*, *int32xm1\_t b*, unsigned int *gvl*)
- *int32xm2\_t vmaxvv\_int32xm2* (*int32xm2\_t a*, *int32xm2\_t b*, unsigned int *gvl*)
- *int32xm4\_t vmaxvv\_int32xm4* (*int32xm4\_t a*, *int32xm4\_t b*, unsigned int *gvl*)
- *int32xm8\_t vmaxvv\_int32xm8* (*int32xm8\_t a*, *int32xm8\_t b*, unsigned int *gvl*)
- *int64xm1\_t vmaxvv\_int64xm1* (*int64xm1\_t a*, *int64xm1\_t b*, unsigned int *gvl*)
- *int64xm2\_t vmaxvv\_int64xm2* (*int64xm2\_t a*, *int64xm2\_t b*, unsigned int *gvl*)
- *int64xm4\_t vmaxvv\_int64xm4* (*int64xm4\_t a*, *int64xm4\_t b*, unsigned int *gvl*)
- *int64xm8\_t vmaxvv\_int64xm8* (*int64xm8\_t a*, *int64xm8\_t b*, unsigned int *gvl*)
- *int8xm1\_t vmaxvv\_int8xm1* (*int8xm1\_t a*, *int8xm1\_t b*, unsigned int *gvl*)
- *int8xm2\_t vmaxvv\_int8xm2* (*int8xm2\_t a*, *int8xm2\_t b*, unsigned int *gvl*)
- *int8xm4\_t vmaxvv\_int8xm4* (*int8xm4\_t a*, *int8xm4\_t b*, unsigned int *gvl*)
- *int8xm8\_t vmaxvv\_int8xm8* (*int8xm8\_t a*, *int8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = max (a[element], b[element])
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vmaxvv\_mask\_int16xm1* (*int16xm1\_t merge*, *int16xm1\_t a*, *int16xm1\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vmaxvv\_mask\_int16xm2* (*int16xm2\_t merge*, *int16xm2\_t a*, *int16xm2\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vmaxvv\_mask\_int16xm4* (*int16xm4\_t merge*, *int16xm4\_t a*, *int16xm4\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vmaxvv\_mask\_int16xm8* (*int16xm8\_t merge*, *int16xm8\_t a*, *int16xm8\_t b*, *e16xm8\_t mask*, unsigned int *gvl*)

- `int32xm1_t vmaxvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vmaxvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vmaxvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vmaxvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vmaxvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vmaxvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vmaxvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vmaxvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vmaxvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vmaxvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vmaxvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vmaxvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = max (a[element], b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.25 Elementwise signed vector-scalar maximum

Instruction: ['vmax.vx']

Prototypes:

- `int16xm1_t vmaxvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int16xm2_t vmaxvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vmaxvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vmaxvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vmaxvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vmaxvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vmaxvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vmaxvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`

- `int64xm1_t vmaxvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vmaxvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vmaxvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vmaxvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vmaxvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vmaxvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vmaxvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vmaxvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = max (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vmaxvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vmaxvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vmaxvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vmaxvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vmaxvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vmaxvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vmaxvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vmaxvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vmaxvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vmaxvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vmaxvx_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vmaxvx_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vmaxvx_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vmaxvx_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vmaxvx_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, signed char b, e8xm4_t mask, unsigned int gvl)`

- `int8xm8_t vmaxvx_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, signed char `b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = max (a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.26 Elementwise unsigned vector-vector maximum

Instruction: ['vmaxu.vv']

Prototypes:

- `uint16xm1_t vmaxuvv_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, unsigned int `gvl`)
- `uint16xm2_t vmaxuvv_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, unsigned int `gvl`)
- `uint16xm4_t vmaxuvv_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, unsigned int `gvl`)
- `uint16xm8_t vmaxuvv_uint16xm8` (`uint16xm8_t a`, `uint16xm8_t b`, unsigned int `gvl`)
- `uint32xm1_t vmaxuvv_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, unsigned int `gvl`)
- `uint32xm2_t vmaxuvv_uint32xm2` (`uint32xm2_t a`, `uint32xm2_t b`, unsigned int `gvl`)
- `uint32xm4_t vmaxuvv_uint32xm4` (`uint32xm4_t a`, `uint32xm4_t b`, unsigned int `gvl`)
- `uint32xm8_t vmaxuvv_uint32xm8` (`uint32xm8_t a`, `uint32xm8_t b`, unsigned int `gvl`)
- `uint64xm1_t vmaxuvv_uint64xm1` (`uint64xm1_t a`, `uint64xm1_t b`, unsigned int `gvl`)
- `uint64xm2_t vmaxuvv_uint64xm2` (`uint64xm2_t a`, `uint64xm2_t b`, unsigned int `gvl`)
- `uint64xm4_t vmaxuvv_uint64xm4` (`uint64xm4_t a`, `uint64xm4_t b`, unsigned int `gvl`)
- `uint64xm8_t vmaxuvv_uint64xm8` (`uint64xm8_t a`, `uint64xm8_t b`, unsigned int `gvl`)
- `uint8xm1_t vmaxuvv_uint8xm1` (`uint8xm1_t a`, `uint8xm1_t b`, unsigned int `gvl`)
- `uint8xm2_t vmaxuvv_uint8xm2` (`uint8xm2_t a`, `uint8xm2_t b`, unsigned int `gvl`)
- `uint8xm4_t vmaxuvv_uint8xm4` (`uint8xm4_t a`, `uint8xm4_t b`, unsigned int `gvl`)
- `uint8xm8_t vmaxuvv_uint8xm8` (`uint8xm8_t a`, `uint8xm8_t b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = max (a[element], b[element])
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `uint16xm1_t vmaxuvv_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, `uint16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint16xm2_t vmaxuvv_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, `uint16xm2_t b`, `e16xm2_t mask`, unsigned int `gvl`)

- `uint16xm4_t vmaxuvv_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vmaxuvv_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vmaxuvv_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vmaxuvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vmaxuvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vmaxuvv_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vmaxuvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vmaxuvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vmaxuvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vmaxuvv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vmaxuvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vmaxuvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vmaxuvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vmaxuvv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = max (a[element], b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.27 Elementwise unsigned vector-scalar maximum****Instruction:** [`'vmaxu.vx'`]**Prototypes:**

- `uint16xm1_t vmaxuvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vmaxuvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vmaxuvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vmaxuvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vmaxuvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`

- `uint32xm2_t vmaxuvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vmaxuvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vmaxuvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vmaxuvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vmaxuvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vmaxuvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vmaxuvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vmaxuvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vmaxuvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vmaxuvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vmaxuvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = max (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vmaxuvx_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vmaxuvx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vmaxuvx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vmaxuvx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vmaxuvx_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vmaxuvx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vmaxuvx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vmaxuvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vmaxuvx_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vmaxuvx_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vmaxuvx_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vmaxuvx_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vmaxuvx_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`



- `uint8xm2_t vmaxuvx_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vmaxuvx_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vmaxuvx_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = max (a[element], b)
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.7.28 Elementwise signed vector-vector minumim****Instruction:** ['vmin.vv']**Prototypes:**

- `int16xm1_t vminvv_int16xm1 (int16xm1_t a, int16xm1_t b, unsigned int gvl)`
- `int16xm2_t vminvv_int16xm2 (int16xm2_t a, int16xm2_t b, unsigned int gvl)`
- `int16xm4_t vminvv_int16xm4 (int16xm4_t a, int16xm4_t b, unsigned int gvl)`
- `int16xm8_t vminvv_int16xm8 (int16xm8_t a, int16xm8_t b, unsigned int gvl)`
- `int32xm1_t vminvv_int32xm1 (int32xm1_t a, int32xm1_t b, unsigned int gvl)`
- `int32xm2_t vminvv_int32xm2 (int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int32xm4_t vminvv_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vminvv_int32xm8 (int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vminvv_int64xm1 (int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vminvv_int64xm2 (int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vminvv_int64xm4 (int64xm4_t a, int64xm4_t b, unsigned int gvl)`
- `int64xm8_t vminvv_int64xm8 (int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vminvv_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vminvv_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vminvv_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vminvv_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = min (a[element], b[element])
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vminvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vminvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vminvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vminvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vminvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vminvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vminvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vminvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vminvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vminvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vminvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vminvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vminvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vminvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vminvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vminvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = min (a[element], b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.29 Elementwise signed vector-scalar minumim

**Instruction:** [`'vmin.vx'`]

**Prototypes:**

- `int16xm1_t vminvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`

- `int16xm2_t vminvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vminvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vminvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vminvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vminvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vminvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vminvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vminvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vminvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vminvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vminvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vminvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vminvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vminvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vminvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = min (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vminvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vminvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vminvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vminvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vminvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vminvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vminvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vminvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vminvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vminvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`

- `int64xm4_t vminvx_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `int64xm8_t vminvx_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `int8xm1_t vminvx_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, signed char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8xm2_t vminvx_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, signed char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `int8xm4_t vminvx_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, signed char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int8xm8_t vminvx_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, signed char `b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = min (a[element], b)
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.7.30 Elementwise unsigned vector-vector minumim****Instruction:** [`'vminu.vv'`]**Prototypes:**

- `uint16xm1_t vminuvv_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, unsigned int `gvl`)
- `uint16xm2_t vminuvv_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, unsigned int `gvl`)
- `uint16xm4_t vminuvv_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, unsigned int `gvl`)
- `uint16xm8_t vminuvv_uint16xm8` (`uint16xm8_t a`, `uint16xm8_t b`, unsigned int `gvl`)
- `uint32xm1_t vminuvv_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, unsigned int `gvl`)
- `uint32xm2_t vminuvv_uint32xm2` (`uint32xm2_t a`, `uint32xm2_t b`, unsigned int `gvl`)
- `uint32xm4_t vminuvv_uint32xm4` (`uint32xm4_t a`, `uint32xm4_t b`, unsigned int `gvl`)
- `uint32xm8_t vminuvv_uint32xm8` (`uint32xm8_t a`, `uint32xm8_t b`, unsigned int `gvl`)
- `uint64xm1_t vminuvv_uint64xm1` (`uint64xm1_t a`, `uint64xm1_t b`, unsigned int `gvl`)
- `uint64xm2_t vminuvv_uint64xm2` (`uint64xm2_t a`, `uint64xm2_t b`, unsigned int `gvl`)
- `uint64xm4_t vminuvv_uint64xm4` (`uint64xm4_t a`, `uint64xm4_t b`, unsigned int `gvl`)
- `uint64xm8_t vminuvv_uint64xm8` (`uint64xm8_t a`, `uint64xm8_t b`, unsigned int `gvl`)
- `uint8xm1_t vminuvv_uint8xm1` (`uint8xm1_t a`, `uint8xm1_t b`, unsigned int `gvl`)
- `uint8xm2_t vminuvv_uint8xm2` (`uint8xm2_t a`, `uint8xm2_t b`, unsigned int `gvl`)
- `uint8xm4_t vminuvv_uint8xm4` (`uint8xm4_t a`, `uint8xm4_t b`, unsigned int `gvl`)
- `uint8xm8_t vminuvv_uint8xm8` (`uint8xm8_t a`, `uint8xm8_t b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = min (a[element], b[element])
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vminuvv_mask_uint16xm1` (`uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask`, unsigned int gvl)
- `uint16xm2_t vminuvv_mask_uint16xm2` (`uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask`, unsigned int gvl)
- `uint16xm4_t vminuvv_mask_uint16xm4` (`uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask`, unsigned int gvl)
- `uint16xm8_t vminuvv_mask_uint16xm8` (`uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask`, unsigned int gvl)
- `uint32xm1_t vminuvv_mask_uint32xm1` (`uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask`, unsigned int gvl)
- `uint32xm2_t vminuvv_mask_uint32xm2` (`uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask`, unsigned int gvl)
- `uint32xm4_t vminuvv_mask_uint32xm4` (`uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask`, unsigned int gvl)
- `uint32xm8_t vminuvv_mask_uint32xm8` (`uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask`, unsigned int gvl)
- `uint64xm1_t vminuvv_mask_uint64xm1` (`uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask`, unsigned int gvl)
- `uint64xm2_t vminuvv_mask_uint64xm2` (`uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask`, unsigned int gvl)
- `uint64xm4_t vminuvv_mask_uint64xm4` (`uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask`, unsigned int gvl)
- `uint64xm8_t vminuvv_mask_uint64xm8` (`uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask`, unsigned int gvl)
- `uint8xm1_t vminuvv_mask_uint8xm1` (`uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask`, unsigned int gvl)
- `uint8xm2_t vminuvv_mask_uint8xm2` (`uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask`, unsigned int gvl)
- `uint8xm4_t vminuvv_mask_uint8xm4` (`uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask`, unsigned int gvl)
- `uint8xm8_t vminuvv_mask_uint8xm8` (`uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask`, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = min (a[element], b[element])
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.7.31 Elementwise unsigned vector-scalar minumim

**Instruction:** ['vminu.vx']

**Prototypes:**

- `uint16xm1_t vminuvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vminuvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vminuvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vminuvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vminuvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vminuvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vminuvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vminuvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vminuvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vminuvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vminuvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vminuvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vminuvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vminuvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vminuvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vminuvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = min (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vminuvx_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vminuvx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vminuvx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vminuvx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vminuvx_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vminuvx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vminuvx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vminuvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`



- `uint64xm1_t vminuvx_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, unsigned long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vminuvx_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, unsigned long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vminuvx_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, unsigned long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vminuvx_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, unsigned long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vminuvx_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vminuvx_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vminuvx_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vminuvx_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = min (a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.32 Elementwise vector-vector integer addition with borrow in mask register format****Instruction:** [`'vmsbcvmm'`]**Masked prototypes:**

- `e16xm1_t vmsbcvmm_mask_e16xm1_int16xm1` (`int16xm1_t a`, `int16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `e16xm1_t vmsbcvmm_mask_e16xm1_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `e16xm2_t vmsbcvmm_mask_e16xm2_int16xm2` (`int16xm2_t a`, `int16xm2_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `e16xm2_t vmsbcvmm_mask_e16xm2_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `e16xm4_t vmsbcvmm_mask_e16xm4_int16xm4` (`int16xm4_t a`, `int16xm4_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `e16xm4_t vmsbcvmm_mask_e16xm4_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `e16xm8_t vmsbcvmm_mask_e16xm8_int16xm8` (`int16xm8_t a`, `int16xm8_t b`, `e16xm8_t mask`, unsigned int `gvl`)
- `e16xm8_t vmsbcvmm_mask_e16xm8_uint16xm8` (`uint16xm8_t a`, `uint16xm8_t b`, `e16xm8_t mask`, unsigned int `gvl`)

- `e32xm1_t vmsbcvmm_mask_e32xm1_int32xm1` (`int32xm1_t a`, `int32xm1_t b`, `e32xm1_t mask`, unsigned int gvl)
- `e32xm1_t vmsbcvmm_mask_e32xm1_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, `e32xm1_t mask`, unsigned int gvl)
- `e32xm2_t vmsbcvmm_mask_e32xm2_int32xm2` (`int32xm2_t a`, `int32xm2_t b`, `e32xm2_t mask`, unsigned int gvl)
- `e32xm2_t vmsbcvmm_mask_e32xm2_uint32xm2` (`uint32xm2_t a`, `uint32xm2_t b`, `e32xm2_t mask`, unsigned int gvl)
- `e32xm4_t vmsbcvmm_mask_e32xm4_int32xm4` (`int32xm4_t a`, `int32xm4_t b`, `e32xm4_t mask`, unsigned int gvl)
- `e32xm4_t vmsbcvmm_mask_e32xm4_uint32xm4` (`uint32xm4_t a`, `uint32xm4_t b`, `e32xm4_t mask`, unsigned int gvl)
- `e32xm8_t vmsbcvmm_mask_e32xm8_int32xm8` (`int32xm8_t a`, `int32xm8_t b`, `e32xm8_t mask`, unsigned int gvl)
- `e32xm8_t vmsbcvmm_mask_e32xm8_uint32xm8` (`uint32xm8_t a`, `uint32xm8_t b`, `e32xm8_t mask`, unsigned int gvl)
- `e64xm1_t vmsbcvmm_mask_e64xm1_int64xm1` (`int64xm1_t a`, `int64xm1_t b`, `e64xm1_t mask`, unsigned int gvl)
- `e64xm1_t vmsbcvmm_mask_e64xm1_uint64xm1` (`uint64xm1_t a`, `uint64xm1_t b`, `e64xm1_t mask`, unsigned int gvl)
- `e64xm2_t vmsbcvmm_mask_e64xm2_int64xm2` (`int64xm2_t a`, `int64xm2_t b`, `e64xm2_t mask`, unsigned int gvl)
- `e64xm2_t vmsbcvmm_mask_e64xm2_uint64xm2` (`uint64xm2_t a`, `uint64xm2_t b`, `e64xm2_t mask`, unsigned int gvl)
- `e64xm4_t vmsbcvmm_mask_e64xm4_int64xm4` (`int64xm4_t a`, `int64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)
- `e64xm4_t vmsbcvmm_mask_e64xm4_uint64xm4` (`uint64xm4_t a`, `uint64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)
- `e64xm8_t vmsbcvmm_mask_e64xm8_int64xm8` (`int64xm8_t a`, `int64xm8_t b`, `e64xm8_t mask`, unsigned int gvl)
- `e64xm8_t vmsbcvmm_mask_e64xm8_uint64xm8` (`uint64xm8_t a`, `uint64xm8_t b`, `e64xm8_t mask`, unsigned int gvl)
- `e8xm1_t vmsbcvmm_mask_e8xm1_int8xm1` (`int8xm1_t a`, `int8xm1_t b`, `e8xm1_t mask`, unsigned int gvl)
- `e8xm1_t vmsbcvmm_mask_e8xm1_uint8xm1` (`uint8xm1_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int gvl)
- `e8xm2_t vmsbcvmm_mask_e8xm2_int8xm2` (`int8xm2_t a`, `int8xm2_t b`, `e8xm2_t mask`, unsigned int gvl)
- `e8xm2_t vmsbcvmm_mask_e8xm2_uint8xm2` (`uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int gvl)
- `e8xm4_t vmsbcvmm_mask_e8xm4_int8xm4` (`int8xm4_t a`, `int8xm4_t b`, `e8xm4_t mask`, unsigned int gvl)
- `e8xm4_t vmsbcvmm_mask_e8xm4_uint8xm4` (`uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int gvl)
- `e8xm8_t vmsbcvmm_mask_e8xm8_int8xm8` (`int8xm8_t a`, `int8xm8_t b`, `e8xm8_t mask`, unsigned int gvl)

- `e8xm8_t vmsbcvmm_mask_e8xm8_uint8xm8 (uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    result[element] = borrow_out(a[element] - b[element] - mask[element])
result[gvl : VLMAX] = 0
```

### 2.7.33 Elementwise vector-scalar integer addition with borrow in mask register format

Instruction: ['vmsbc.vxm']

Masked prototypes:

- `e16xm1_t vmsbcvmm_mask_e16xm1_int16xm1 (int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `e16xm1_t vmsbcvmm_mask_e16xm1_uint16xm1 (uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `e16xm2_t vmsbcvmm_mask_e16xm2_int16xm2 (int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `e16xm2_t vmsbcvmm_mask_e16xm2_uint16xm2 (uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `e16xm4_t vmsbcvmm_mask_e16xm4_int16xm4 (int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `e16xm4_t vmsbcvmm_mask_e16xm4_uint16xm4 (uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `e16xm8_t vmsbcvmm_mask_e16xm8_int16xm8 (int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `e16xm8_t vmsbcvmm_mask_e16xm8_uint16xm8 (uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `e32xm1_t vmsbcvmm_mask_e32xm1_int32xm1 (int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `e32xm1_t vmsbcvmm_mask_e32xm1_uint32xm1 (uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `e32xm2_t vmsbcvmm_mask_e32xm2_int32xm2 (int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `e32xm2_t vmsbcvmm_mask_e32xm2_uint32xm2 (uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `e32xm4_t vmsbcvmm_mask_e32xm4_int32xm4 (int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `e32xm4_t vmsbcvmm_mask_e32xm4_uint32xm4 (uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `e32xm8_t vmsbcvmm_mask_e32xm8_int32xm8 (int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `e32xm8_t vmsbcvmm_mask_e32xm8_uint32xm8 (uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `e64xm1_t vmsbcvmm_mask_e64xm1_int64xm1 (int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`

- `e64xm1_t vmsbcvxm_mask_e64xm1_uint64xm1` (`uint64xm1_t a`, unsigned long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `e64xm2_t vmsbcvxm_mask_e64xm2_int64xm2` (`int64xm2_t a`, long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `e64xm2_t vmsbcvxm_mask_e64xm2_uint64xm2` (`uint64xm2_t a`, unsigned long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `e64xm4_t vmsbcvxm_mask_e64xm4_int64xm4` (`int64xm4_t a`, long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `e64xm4_t vmsbcvxm_mask_e64xm4_uint64xm4` (`uint64xm4_t a`, unsigned long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `e64xm8_t vmsbcvxm_mask_e64xm8_int64xm8` (`int64xm8_t a`, long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `e64xm8_t vmsbcvxm_mask_e64xm8_uint64xm8` (`uint64xm8_t a`, unsigned long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `e8xm1_t vmsbcvxm_mask_e8xm1_int8xm1` (`int8xm1_t a`, signed char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `e8xm1_t vmsbcvxm_mask_e8xm1_uint8xm1` (`uint8xm1_t a`, unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `e8xm2_t vmsbcvxm_mask_e8xm2_int8xm2` (`int8xm2_t a`, signed char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `e8xm2_t vmsbcvxm_mask_e8xm2_uint8xm2` (`uint8xm2_t a`, unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `e8xm4_t vmsbcvxm_mask_e8xm4_int8xm4` (`int8xm4_t a`, signed char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `e8xm4_t vmsbcvxm_mask_e8xm4_uint8xm4` (`uint8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `e8xm8_t vmsbcvxm_mask_e8xm8_int8xm8` (`int8xm8_t a`, signed char `b`, `e8xm8_t mask`, unsigned int `gvl`)
- `e8xm8_t vmsbcvxm_mask_e8xm8_uint8xm8` (`uint8xm8_t a`, unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = carry_out(a[element] + b + mask[elemet])
      result[gvl : VLMAX] = 0
```

## 2.7.34 Elementwise vector-vector integer multiplication

**Instruction:** [`'vmul.vv'`]**Prototypes:**

- `int16xm1_t vmulvv_int16xm1` (`int16xm1_t a`, `int16xm1_t b`, unsigned int `gvl`)
- `int16xm2_t vmulvv_int16xm2` (`int16xm2_t a`, `int16xm2_t b`, unsigned int `gvl`)
- `int16xm4_t vmulvv_int16xm4` (`int16xm4_t a`, `int16xm4_t b`, unsigned int `gvl`)
- `int16xm8_t vmulvv_int16xm8` (`int16xm8_t a`, `int16xm8_t b`, unsigned int `gvl`)
- `int32xm1_t vmulvv_int32xm1` (`int32xm1_t a`, `int32xm1_t b`, unsigned int `gvl`)

- `int32xm2_t vmulvv_int32xm2 (int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int32xm4_t vmulvv_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vmulvv_int32xm8 (int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vmulvv_int64xm1 (int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vmulvv_int64xm2 (int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vmulvv_int64xm4 (int64xm4_t a, int64xm4_t b, unsigned int gvl)`
- `int64xm8_t vmulvv_int64xm8 (int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vmulvv_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vmulvv_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vmulvv_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vmulvv_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`
- `uint16xm1_t vmulvv_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vmulvv_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vmulvv_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vmulvv_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vmulvv_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vmulvv_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vmulvv_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vmulvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vmulvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vmulvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vmulvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vmulvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vmulvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vmulvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vmulvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vmulvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] * b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vmulvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vmulvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vmulvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`

- `int16xm8_t vmulvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vmulvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vmulvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vmulvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vmulvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vmulvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vmulvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vmulvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vmulvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vmulvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vmulvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vmulvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vmulvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vmulvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vmulvv_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vmulvv_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vmulvv_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vmulvv_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vmulvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vmulvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vmulvv_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vmulvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vmulvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`



- `uint64xm4_t vmulvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vmulvv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vmulvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vmulvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vmulvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vmulvv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] * b[element]
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.7.35 Elementwise vector-scalar integer multiplication****Instruction:** [`'vmul.vx'`]**Prototypes:**

- `int16xm1_t vmulvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int16xm2_t vmulvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vmulvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vmulvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vmulvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vmulvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vmulvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vmulvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vmulvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vmulvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vmulvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vmulvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vmulvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vmulvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vmulvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vmulvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] x b
result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- *int16xm1\_t vmulvx\_mask\_int16xm1* (*int16xm1\_t* merge, *int16xm1\_t* a, short b, *e16xm1\_t* mask, unsigned int gvl)
- *int16xm2\_t vmulvx\_mask\_int16xm2* (*int16xm2\_t* merge, *int16xm2\_t* a, short b, *e16xm2\_t* mask, unsigned int gvl)
- *int16xm4\_t vmulvx\_mask\_int16xm4* (*int16xm4\_t* merge, *int16xm4\_t* a, short b, *e16xm4\_t* mask, unsigned int gvl)
- *int16xm8\_t vmulvx\_mask\_int16xm8* (*int16xm8\_t* merge, *int16xm8\_t* a, short b, *e16xm8\_t* mask, unsigned int gvl)
- *int32xm1\_t vmulvx\_mask\_int32xm1* (*int32xm1\_t* merge, *int32xm1\_t* a, int b, *e32xm1\_t* mask, unsigned int gvl)
- *int32xm2\_t vmulvx\_mask\_int32xm2* (*int32xm2\_t* merge, *int32xm2\_t* a, int b, *e32xm2\_t* mask, unsigned int gvl)
- *int32xm4\_t vmulvx\_mask\_int32xm4* (*int32xm4\_t* merge, *int32xm4\_t* a, int b, *e32xm4\_t* mask, unsigned int gvl)
- *int32xm8\_t vmulvx\_mask\_int32xm8* (*int32xm8\_t* merge, *int32xm8\_t* a, int b, *e32xm8\_t* mask, unsigned int gvl)
- *int64xm1\_t vmulvx\_mask\_int64xm1* (*int64xm1\_t* merge, *int64xm1\_t* a, long b, *e64xm1\_t* mask, unsigned int gvl)
- *int64xm2\_t vmulvx\_mask\_int64xm2* (*int64xm2\_t* merge, *int64xm2\_t* a, long b, *e64xm2\_t* mask, unsigned int gvl)
- *int64xm4\_t vmulvx\_mask\_int64xm4* (*int64xm4\_t* merge, *int64xm4\_t* a, long b, *e64xm4\_t* mask, unsigned int gvl)
- *int64xm8\_t vmulvx\_mask\_int64xm8* (*int64xm8\_t* merge, *int64xm8\_t* a, long b, *e64xm8\_t* mask, unsigned int gvl)
- *int8xm1\_t vmulvx\_mask\_int8xm1* (*int8xm1\_t* merge, *int8xm1\_t* a, signed char b, *e8xm1\_t* mask, unsigned int gvl)
- *int8xm2\_t vmulvx\_mask\_int8xm2* (*int8xm2\_t* merge, *int8xm2\_t* a, signed char b, *e8xm2\_t* mask, unsigned int gvl)
- *int8xm4\_t vmulvx\_mask\_int8xm4* (*int8xm4\_t* merge, *int8xm4\_t* a, signed char b, *e8xm4\_t* mask, unsigned int gvl)
- *int8xm8\_t vmulvx\_mask\_int8xm8* (*int8xm8\_t* merge, *int8xm8\_t* a, signed char b, *e8xm8\_t* mask, unsigned int gvl)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] x b
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.36 Elementwise signed vector-vector multiplication(higher bits)

**Instruction:** ['vmulh.vv']

**Prototypes:**

- `int16xm1_t vmulhvv_int16xm1 (int16xm1_t a, int16xm1_t b, unsigned int gvl)`
- `int16xm2_t vmulhvv_int16xm2 (int16xm2_t a, int16xm2_t b, unsigned int gvl)`
- `int16xm4_t vmulhvv_int16xm4 (int16xm4_t a, int16xm4_t b, unsigned int gvl)`
- `int16xm8_t vmulhvv_int16xm8 (int16xm8_t a, int16xm8_t b, unsigned int gvl)`
- `int32xm1_t vmulhvv_int32xm1 (int32xm1_t a, int32xm1_t b, unsigned int gvl)`
- `int32xm2_t vmulhvv_int32xm2 (int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int32xm4_t vmulhvv_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vmulhvv_int32xm8 (int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vmulhvv_int64xm1 (int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vmulhvv_int64xm2 (int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vmulhvv_int64xm4 (int64xm4_t a, int64xm4_t b, unsigned int gvl)`
- `int64xm8_t vmulhvv_int64xm8 (int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vmulhvv_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vmulhvv_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vmulhvv_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vmulhvv_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = mulh (a[element], b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vmulhvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vmulhvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vmulhvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vmulhvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vmulhvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vmulhvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vmulhvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vmulhvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`

- `int64xm1_t vmulhvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vmulhvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vmulhvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vmulhvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vmulhvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vmulhvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vmulhvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vmulhvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = mulh (a[element], b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

### 2.7.37 Elementwise signed vector-scalar multiplication(higher bits)

**Instruction:** ['vmulh.vx']

#### Prototypes:

- `int16xm1_t vmulhvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int16xm2_t vmulhvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vmulhvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vmulhvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vmulhvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vmulhvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vmulhvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vmulhvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vmulhvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vmulhvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vmulhvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vmulhvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vmulhvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vmulhvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`

- `int8xm4_t vmulhvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vmulhvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = mulh (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vmulhvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vmulhvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vmulhvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vmulhvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vmulhvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vmulhvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vmulhvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vmulhvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vmulhvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vmulhvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vmulhvx_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vmulhvx_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vmulhvx_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vmulhvx_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vmulhvx_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, signed char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vmulhvx_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, signed char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = mulh (a[element], b)
      else
```

(continues on next page)

(continued from previous page)

```

    result[element] = merge[element]
    result[gvl : VLMAX] = 0

```

## 2.7.38 Elementwise vector-vector signed-unsigned integer multiplication(higher bits)

**Instruction:** ['vmulhsu.vv']

**Prototypes:**

- *int16xm1\_t vmulhsuvv\_int16xm1\_uint16xm1* (*int16xm1\_t a, uint16xm1\_t b*, unsigned int *gvl*)
- *int16xm2\_t vmulhsuvv\_int16xm2\_uint16xm2* (*int16xm2\_t a, uint16xm2\_t b*, unsigned int *gvl*)
- *int16xm4\_t vmulhsuvv\_int16xm4\_uint16xm4* (*int16xm4\_t a, uint16xm4\_t b*, unsigned int *gvl*)
- *int16xm8\_t vmulhsuvv\_int16xm8\_uint16xm8* (*int16xm8\_t a, uint16xm8\_t b*, unsigned int *gvl*)
- *int32xm1\_t vmulhsuvv\_int32xm1\_uint32xm1* (*int32xm1\_t a, uint32xm1\_t b*, unsigned int *gvl*)
- *int32xm2\_t vmulhsuvv\_int32xm2\_uint32xm2* (*int32xm2\_t a, uint32xm2\_t b*, unsigned int *gvl*)
- *int32xm4\_t vmulhsuvv\_int32xm4\_uint32xm4* (*int32xm4\_t a, uint32xm4\_t b*, unsigned int *gvl*)
- *int32xm8\_t vmulhsuvv\_int32xm8\_uint32xm8* (*int32xm8\_t a, uint32xm8\_t b*, unsigned int *gvl*)
- *int64xm1\_t vmulhsuvv\_int64xm1\_uint64xm1* (*int64xm1\_t a, uint64xm1\_t b*, unsigned int *gvl*)
- *int64xm2\_t vmulhsuvv\_int64xm2\_uint64xm2* (*int64xm2\_t a, uint64xm2\_t b*, unsigned int *gvl*)
- *int64xm4\_t vmulhsuvv\_int64xm4\_uint64xm4* (*int64xm4\_t a, uint64xm4\_t b*, unsigned int *gvl*)
- *int64xm8\_t vmulhsuvv\_int64xm8\_uint64xm8* (*int64xm8\_t a, uint64xm8\_t b*, unsigned int *gvl*)
- *int8xm1\_t vmulhsuvv\_int8xm1\_uint8xm1* (*int8xm1\_t a, uint8xm1\_t b*, unsigned int *gvl*)
- *int8xm2\_t vmulhsuvv\_int8xm2\_uint8xm2* (*int8xm2\_t a, uint8xm2\_t b*, unsigned int *gvl*)
- *int8xm4\_t vmulhsuvv\_int8xm4\_uint8xm4* (*int8xm4\_t a, uint8xm4\_t b*, unsigned int *gvl*)
- *int8xm8\_t vmulhsuvv\_int8xm8\_uint8xm8* (*int8xm8\_t a, uint8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```

>>> for element = 0 to gvl - 1
    result[element] = mulhsu (a[element], b[element])
    result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- *int16xm1\_t vmulhsuvv\_mask\_int16xm1\_uint16xm1* (*int16xm1\_t merge, int16xm1\_t a, uint16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vmulhsuvv\_mask\_int16xm2\_uint16xm2* (*int16xm2\_t merge, int16xm2\_t a, uint16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vmulhsuvv\_mask\_int16xm4\_uint16xm4* (*int16xm4\_t merge, int16xm4\_t a, uint16xm4\_t b, e16xm4\_t mask*, unsigned int *gvl*)



- `int16xm8_t vmulhsuvv_mask_int16xm8_uint16xm8 (int16xm8_t merge, int16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vmulhsuvv_mask_int32xm1_uint32xm1 (int32xm1_t merge, int32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vmulhsuvv_mask_int32xm2_uint32xm2 (int32xm2_t merge, int32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vmulhsuvv_mask_int32xm4_uint32xm4 (int32xm4_t merge, int32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vmulhsuvv_mask_int32xm8_uint32xm8 (int32xm8_t merge, int32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vmulhsuvv_mask_int64xm1_uint64xm1 (int64xm1_t merge, int64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vmulhsuvv_mask_int64xm2_uint64xm2 (int64xm2_t merge, int64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vmulhsuvv_mask_int64xm4_uint64xm4 (int64xm4_t merge, int64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vmulhsuvv_mask_int64xm8_uint64xm8 (int64xm8_t merge, int64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vmulhsuvv_mask_int8xm1_uint8xm1 (int8xm1_t merge, int8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vmulhsuvv_mask_int8xm2_uint8xm2 (int8xm2_t merge, int8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vmulhsuvv_mask_int8xm4_uint8xm4 (int8xm4_t merge, int8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vmulhsuvv_mask_int8xm8_uint8xm8 (int8xm8_t merge, int8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = mulhsu (a[element], b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.39 Elementwise vector-scalar signed-unsigned integer multiplication(higher bits)

**Instruction:** [`'vmulhsu.vx'`]

**Prototypes:**

- `int16xm1_t vmulhsuvx_int16xm1 (int16xm1_t a, unsigned short b, unsigned int gvl)`
- `int16xm2_t vmulhsuvx_int16xm2 (int16xm2_t a, unsigned short b, unsigned int gvl)`
- `int16xm4_t vmulhsuvx_int16xm4 (int16xm4_t a, unsigned short b, unsigned int gvl)`
- `int16xm8_t vmulhsuvx_int16xm8 (int16xm8_t a, unsigned short b, unsigned int gvl)`
- `int32xm1_t vmulhsuvx_int32xm1 (int32xm1_t a, unsigned int b, unsigned int gvl)`
- `int32xm2_t vmulhsuvx_int32xm2 (int32xm2_t a, unsigned int b, unsigned int gvl)`
- `int32xm4_t vmulhsuvx_int32xm4 (int32xm4_t a, unsigned int b, unsigned int gvl)`
- `int32xm8_t vmulhsuvx_int32xm8 (int32xm8_t a, unsigned int b, unsigned int gvl)`
- `int64xm1_t vmulhsuvx_int64xm1 (int64xm1_t a, unsigned long b, unsigned int gvl)`
- `int64xm2_t vmulhsuvx_int64xm2 (int64xm2_t a, unsigned long b, unsigned int gvl)`
- `int64xm4_t vmulhsuvx_int64xm4 (int64xm4_t a, unsigned long b, unsigned int gvl)`
- `int64xm8_t vmulhsuvx_int64xm8 (int64xm8_t a, unsigned long b, unsigned int gvl)`
- `int8xm1_t vmulhsuvx_int8xm1 (int8xm1_t a, unsigned char b, unsigned int gvl)`
- `int8xm2_t vmulhsuvx_int8xm2 (int8xm2_t a, unsigned char b, unsigned int gvl)`
- `int8xm4_t vmulhsuvx_int8xm4 (int8xm4_t a, unsigned char b, unsigned int gvl)`
- `int8xm8_t vmulhsuvx_int8xm8 (int8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = mulhsu (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vmulhsuvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vmulhsuvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vmulhsuvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vmulhsuvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vmulhsuvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vmulhsuvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vmulhsuvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vmulhsuvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vmulhsuvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`

- `int64xm2_t vmulhsuvx_mask_int64xm2` (`int64xm2_t merge`, `int64xm2_t a`, unsigned long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `int64xm4_t vmulhsuvx_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, unsigned long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `int64xm8_t vmulhsuvx_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, unsigned long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `int8xm1_t vmulhsuvx_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8xm2_t vmulhsuvx_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `int8xm4_t vmulhsuvx_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int8xm8_t vmulhsuvx_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = mulhsu (a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.40 Elementwise unsigned vector-vector multiplication(higher bits)

Instruction: ['vmulhu.vv']

Prototypes:

- `uint16xm1_t vmulhuvv_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, unsigned int `gvl`)
- `uint16xm2_t vmulhuvv_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, unsigned int `gvl`)
- `uint16xm4_t vmulhuvv_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, unsigned int `gvl`)
- `uint16xm8_t vmulhuvv_uint16xm8` (`uint16xm8_t a`, `uint16xm8_t b`, unsigned int `gvl`)
- `uint32xm1_t vmulhuvv_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, unsigned int `gvl`)
- `uint32xm2_t vmulhuvv_uint32xm2` (`uint32xm2_t a`, `uint32xm2_t b`, unsigned int `gvl`)
- `uint32xm4_t vmulhuvv_uint32xm4` (`uint32xm4_t a`, `uint32xm4_t b`, unsigned int `gvl`)
- `uint32xm8_t vmulhuvv_uint32xm8` (`uint32xm8_t a`, `uint32xm8_t b`, unsigned int `gvl`)
- `uint64xm1_t vmulhuvv_uint64xm1` (`uint64xm1_t a`, `uint64xm1_t b`, unsigned int `gvl`)
- `uint64xm2_t vmulhuvv_uint64xm2` (`uint64xm2_t a`, `uint64xm2_t b`, unsigned int `gvl`)
- `uint64xm4_t vmulhuvv_uint64xm4` (`uint64xm4_t a`, `uint64xm4_t b`, unsigned int `gvl`)
- `uint64xm8_t vmulhuvv_uint64xm8` (`uint64xm8_t a`, `uint64xm8_t b`, unsigned int `gvl`)
- `uint8xm1_t vmulhuvv_uint8xm1` (`uint8xm1_t a`, `uint8xm1_t b`, unsigned int `gvl`)
- `uint8xm2_t vmulhuvv_uint8xm2` (`uint8xm2_t a`, `uint8xm2_t b`, unsigned int `gvl`)
- `uint8xm4_t vmulhuvv_uint8xm4` (`uint8xm4_t a`, `uint8xm4_t b`, unsigned int `gvl`)

- `uint8xm8_t vmulhuvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = mulh (a[element], b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vmulhuvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vmulhuvv_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vmulhuvv_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vmulhuvv_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vmulhuvv_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vmulhuvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vmulhuvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vmulhuvv_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vmulhuvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vmulhuvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vmulhuvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vmulhuvv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vmulhuvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vmulhuvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vmulhuvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vmulhuvv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = mulh (a[element], b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.41 Elementwise unsigned vector-scalar multiplication(higher bits)

**Instruction:** ['vmulhu.vx']

**Prototypes:**

- *uint16xm1\_t* **vmulhuvx\_uint16xm1** (*uint16xm1\_t* *a*, unsigned short *b*, unsigned int *gvl*)
- *uint16xm2\_t* **vmulhuvx\_uint16xm2** (*uint16xm2\_t* *a*, unsigned short *b*, unsigned int *gvl*)
- *uint16xm4\_t* **vmulhuvx\_uint16xm4** (*uint16xm4\_t* *a*, unsigned short *b*, unsigned int *gvl*)
- *uint16xm8\_t* **vmulhuvx\_uint16xm8** (*uint16xm8\_t* *a*, unsigned short *b*, unsigned int *gvl*)
- *uint32xm1\_t* **vmulhuvx\_uint32xm1** (*uint32xm1\_t* *a*, unsigned int *b*, unsigned int *gvl*)
- *uint32xm2\_t* **vmulhuvx\_uint32xm2** (*uint32xm2\_t* *a*, unsigned int *b*, unsigned int *gvl*)
- *uint32xm4\_t* **vmulhuvx\_uint32xm4** (*uint32xm4\_t* *a*, unsigned int *b*, unsigned int *gvl*)
- *uint32xm8\_t* **vmulhuvx\_uint32xm8** (*uint32xm8\_t* *a*, unsigned int *b*, unsigned int *gvl*)
- *uint64xm1\_t* **vmulhuvx\_uint64xm1** (*uint64xm1\_t* *a*, unsigned long *b*, unsigned int *gvl*)
- *uint64xm2\_t* **vmulhuvx\_uint64xm2** (*uint64xm2\_t* *a*, unsigned long *b*, unsigned int *gvl*)
- *uint64xm4\_t* **vmulhuvx\_uint64xm4** (*uint64xm4\_t* *a*, unsigned long *b*, unsigned int *gvl*)
- *uint64xm8\_t* **vmulhuvx\_uint64xm8** (*uint64xm8\_t* *a*, unsigned long *b*, unsigned int *gvl*)
- *uint8xm1\_t* **vmulhuvx\_uint8xm1** (*uint8xm1\_t* *a*, unsigned char *b*, unsigned int *gvl*)
- *uint8xm2\_t* **vmulhuvx\_uint8xm2** (*uint8xm2\_t* *a*, unsigned char *b*, unsigned int *gvl*)
- *uint8xm4\_t* **vmulhuvx\_uint8xm4** (*uint8xm4\_t* *a*, unsigned char *b*, unsigned int *gvl*)
- *uint8xm8\_t* **vmulhuvx\_uint8xm8** (*uint8xm8\_t* *a*, unsigned char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = mulh (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *uint16xm1\_t* **vmulhuvx\_mask\_uint16xm1** (*uint16xm1\_t* *merge*, *uint16xm1\_t* *a*, unsigned short *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *uint16xm2\_t* **vmulhuvx\_mask\_uint16xm2** (*uint16xm2\_t* *merge*, *uint16xm2\_t* *a*, unsigned short *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *uint16xm4\_t* **vmulhuvx\_mask\_uint16xm4** (*uint16xm4\_t* *merge*, *uint16xm4\_t* *a*, unsigned short *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *uint16xm8\_t* **vmulhuvx\_mask\_uint16xm8** (*uint16xm8\_t* *merge*, *uint16xm8\_t* *a*, unsigned short *b*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- *uint32xm1\_t* **vmulhuvx\_mask\_uint32xm1** (*uint32xm1\_t* *merge*, *uint32xm1\_t* *a*, unsigned int *b*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *uint32xm2\_t* **vmulhuvx\_mask\_uint32xm2** (*uint32xm2\_t* *merge*, *uint32xm2\_t* *a*, unsigned int *b*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *uint32xm4\_t* **vmulhuvx\_mask\_uint32xm4** (*uint32xm4\_t* *merge*, *uint32xm4\_t* *a*, unsigned int *b*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *uint32xm8\_t* **vmulhuvx\_mask\_uint32xm8** (*uint32xm8\_t* *merge*, *uint32xm8\_t* *a*, unsigned int *b*, *e32xm8\_t* *mask*, unsigned int *gvl*)

- `uint64xm1_t vmulhuvx_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vmulhuvx_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vmulhuvx_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vmulhuvx_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vmulhuvx_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vmulhuvx_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vmulhuvx_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vmulhuvx_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = mulh (a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.42 Elementwise signed vector-immediate signed integer narrow clip****Instruction:** [`vnclip.vi`']**Prototypes:**

- `int16xm1_t vnclipvi_int16xm1_int32xm2 (int32xm2_t a, const unsigned short b, unsigned int gvl)`
- `int16xm2_t vnclipvi_int16xm2_int32xm4 (int32xm4_t a, const unsigned short b, unsigned int gvl)`
- `int16xm4_t vnclipvi_int16xm4_int32xm8 (int32xm8_t a, const unsigned short b, unsigned int gvl)`
- `int32xm1_t vnclipvi_int32xm1_int64xm2 (int64xm2_t a, const unsigned int b, unsigned int gvl)`
- `int32xm2_t vnclipvi_int32xm2_int64xm4 (int64xm4_t a, const unsigned int b, unsigned int gvl)`
- `int32xm4_t vnclipvi_int32xm4_int64xm8 (int64xm8_t a, const unsigned int b, unsigned int gvl)`
- `int8xm1_t vnclipvi_int8xm1_int16xm2 (int16xm2_t a, const unsigned char b, unsigned int gvl)`
- `int8xm2_t vnclipvi_int8xm2_int16xm4 (int16xm4_t a, const unsigned char b, unsigned int gvl)`
- `int8xm4_t vnclipvi_int8xm4_int16xm8 (int16xm8_t a, const unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = clip(roundoff(a[element], b))
result[gvl : VLMAX] = 0
```



**Masked prototypes:**

- *int16xm1\_t vnclipvi\_mask\_int16xm1\_int32xm2* (*int16xm1\_t merge*, *int32xm2\_t a*, const unsigned short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vnclipvi\_mask\_int16xm2\_int32xm4* (*int16xm2\_t merge*, *int32xm4\_t a*, const unsigned short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vnclipvi\_mask\_int16xm4\_int32xm8* (*int16xm4\_t merge*, *int32xm8\_t a*, const unsigned short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int32xm1\_t vnclipvi\_mask\_int32xm1\_int64xm2* (*int32xm1\_t merge*, *int64xm2\_t a*, const unsigned int *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vnclipvi\_mask\_int32xm2\_int64xm4* (*int32xm2\_t merge*, *int64xm4\_t a*, const unsigned int *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vnclipvi\_mask\_int32xm4\_int64xm8* (*int32xm4\_t merge*, *int64xm8\_t a*, const unsigned int *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *int8xm1\_t vnclipvi\_mask\_int8xm1\_int16xm2* (*int8xm1\_t merge*, *int16xm2\_t a*, const unsigned char *b*, *e8xm1\_t mask*, unsigned int *gvl*)
- *int8xm2\_t vnclipvi\_mask\_int8xm2\_int16xm4* (*int8xm2\_t merge*, *int16xm4\_t a*, const unsigned char *b*, *e8xm2\_t mask*, unsigned int *gvl*)
- *int8xm4\_t vnclipvi\_mask\_int8xm4\_int16xm8* (*int8xm4\_t merge*, *int16xm8\_t a*, const unsigned char *b*, *e8xm4\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = clip(roundoff(a[element], b))
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.43 Elementwise signed vector-vector signed integer narrow clip****Instruction:** ['vnclip.vv']**Prototypes:**

- *int16xm1\_t vnclipvv\_int16xm1\_int32xm2\_uint16xm1* (*int32xm2\_t a*, *uint16xm1\_t b*, unsigned int *gvl*)
- *int16xm2\_t vnclipvv\_int16xm2\_int32xm4\_uint16xm2* (*int32xm4\_t a*, *uint16xm2\_t b*, unsigned int *gvl*)
- *int16xm4\_t vnclipvv\_int16xm4\_int32xm8\_uint16xm4* (*int32xm8\_t a*, *uint16xm4\_t b*, unsigned int *gvl*)
- *int32xm1\_t vnclipvv\_int32xm1\_int64xm2\_uint32xm1* (*int64xm2\_t a*, *uint32xm1\_t b*, unsigned int *gvl*)
- *int32xm2\_t vnclipvv\_int32xm2\_int64xm4\_uint32xm2* (*int64xm4\_t a*, *uint32xm2\_t b*, unsigned int *gvl*)

- `int32xm4_t vnclipvv_int32xm4_int64xm8_uint32xm4 (int64xm8_t a, uint32xm4_t b, unsigned int gvl)`
- `int8xm1_t vnclipvv_int8xm1_int16xm2_uint8xm1 (int16xm2_t a, uint8xm1_t b, unsigned int gvl)`
- `int8xm2_t vnclipvv_int8xm2_int16xm4_uint8xm2 (int16xm4_t a, uint8xm2_t b, unsigned int gvl)`
- `int8xm4_t vnclipvv_int8xm4_int16xm8_uint8xm4 (int16xm8_t a, uint8xm4_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = clip(roundoff(a[element], b[element]))
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vnclipvv_mask_int16xm1_int32xm2_uint16xm1 (int16xm1_t merge, int32xm2_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vnclipvv_mask_int16xm2_int32xm4_uint16xm2 (int16xm2_t merge, int32xm4_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vnclipvv_mask_int16xm4_int32xm8_uint16xm4 (int16xm4_t merge, int32xm8_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int32xm1_t vnclipvv_mask_int32xm1_int64xm2_uint32xm1 (int32xm1_t merge, int64xm2_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vnclipvv_mask_int32xm2_int64xm4_uint32xm2 (int32xm2_t merge, int64xm4_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vnclipvv_mask_int32xm4_int64xm8_uint32xm4 (int32xm4_t merge, int64xm8_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int8xm1_t vnclipvv_mask_int8xm1_int16xm2_uint8xm1 (int8xm1_t merge, int16xm2_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vnclipvv_mask_int8xm2_int16xm4_uint8xm2 (int8xm2_t merge, int16xm4_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vnclipvv_mask_int8xm4_int16xm8_uint8xm4 (int8xm4_t merge, int16xm8_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = clip(roundoff(a[element], b[element]))
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.7.44 Elementwise signed vector-scalar signed integer narrow clip

**Instruction:** ['vnclip.vx']

**Prototypes:**

- *int16xm1\_t vnclipvx\_int16xm1\_int32xm2* (*int32xm2\_t a*, unsigned short *b*, unsigned int *gvl*)
- *int16xm2\_t vnclipvx\_int16xm2\_int32xm4* (*int32xm4\_t a*, unsigned short *b*, unsigned int *gvl*)
- *int16xm4\_t vnclipvx\_int16xm4\_int32xm8* (*int32xm8\_t a*, unsigned short *b*, unsigned int *gvl*)
- *int32xm1\_t vnclipvx\_int32xm1\_int64xm2* (*int64xm2\_t a*, unsigned int *b*, unsigned int *gvl*)
- *int32xm2\_t vnclipvx\_int32xm2\_int64xm4* (*int64xm4\_t a*, unsigned int *b*, unsigned int *gvl*)
- *int32xm4\_t vnclipvx\_int32xm4\_int64xm8* (*int64xm8\_t a*, unsigned int *b*, unsigned int *gvl*)
- *int8xm1\_t vnclipvx\_int8xm1\_int16xm2* (*int16xm2\_t a*, unsigned char *b*, unsigned int *gvl*)
- *int8xm2\_t vnclipvx\_int8xm2\_int16xm4* (*int16xm4\_t a*, unsigned char *b*, unsigned int *gvl*)
- *int8xm4\_t vnclipvx\_int8xm4\_int16xm8* (*int16xm8\_t a*, unsigned char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = clip(roundoff(a[element], b))
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vnclipvx\_mask\_int16xm1\_int32xm2* (*int16xm1\_t merge*, *int32xm2\_t a*, unsigned short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vnclipvx\_mask\_int16xm2\_int32xm4* (*int16xm2\_t merge*, *int32xm4\_t a*, unsigned short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vnclipvx\_mask\_int16xm4\_int32xm8* (*int16xm4\_t merge*, *int32xm8\_t a*, unsigned short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int32xm1\_t vnclipvx\_mask\_int32xm1\_int64xm2* (*int32xm1\_t merge*, *int64xm2\_t a*, unsigned int *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vnclipvx\_mask\_int32xm2\_int64xm4* (*int32xm2\_t merge*, *int64xm4\_t a*, unsigned int *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vnclipvx\_mask\_int32xm4\_int64xm8* (*int32xm4\_t merge*, *int64xm8\_t a*, unsigned int *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *int8xm1\_t vnclipvx\_mask\_int8xm1\_int16xm2* (*int8xm1\_t merge*, *int16xm2\_t a*, unsigned char *b*, *e8xm1\_t mask*, unsigned int *gvl*)
- *int8xm2\_t vnclipvx\_mask\_int8xm2\_int16xm4* (*int8xm2\_t merge*, *int16xm4\_t a*, unsigned char *b*, *e8xm2\_t mask*, unsigned int *gvl*)
- *int8xm4\_t vnclipvx\_mask\_int8xm4\_int16xm8* (*int8xm4\_t merge*, *int16xm8\_t a*, unsigned char *b*, *e8xm4\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = clip(roundoff(a[element], b))
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.45 Elementwise unsigned vector-immediate unsigned integer narrow clip****Instruction:** [`vnclipuvi`]**Prototypes:**

- `uint16xm1_t vnclipuvi_uint16xm1_uint32xm2 (uint32xm2_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm2_t vnclipuvi_uint16xm2_uint32xm4 (uint32xm4_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm4_t vnclipuvi_uint16xm4_uint32xm8 (uint32xm8_t a, const unsigned short b, unsigned int gvl)`
- `uint32xm1_t vnclipuvi_uint32xm1_uint64xm2 (uint64xm2_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm2_t vnclipuvi_uint32xm2_uint64xm4 (uint64xm4_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm4_t vnclipuvi_uint32xm4_uint64xm8 (uint64xm8_t a, const unsigned int b, unsigned int gvl)`
- `uint8xm1_t vnclipuvi_uint8xm1_uint16xm2 (uint16xm2_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm2_t vnclipuvi_uint8xm2_uint16xm4 (uint16xm4_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm4_t vnclipuvi_uint8xm4_uint16xm8 (uint16xm8_t a, const unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = clip(roundoff(a[element], b))
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vnclipuvi_mask_uint16xm1_uint32xm2 (uint16xm1_t merge, uint32xm2_t a, const unsigned short b, uint16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vnclipuvi_mask_uint16xm2_uint32xm4 (uint16xm2_t merge, uint32xm4_t a, const unsigned short b, uint16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vnclipuvi_mask_uint16xm4_uint32xm8 (uint16xm4_t merge, uint32xm8_t a, const unsigned short b, uint16xm4_t mask, unsigned int gvl)`

- `uint32xm1_t vnclipuvi_mask_uint32xm1_uint64xm2` (`uint32xm1_t merge`, `uint64xm2_t a`, `const unsigned int b`, `e32xm1_t mask`, `unsigned int gvl`)
- `uint32xm2_t vnclipuvi_mask_uint32xm2_uint64xm4` (`uint32xm2_t merge`, `uint64xm4_t a`, `const unsigned int b`, `e32xm2_t mask`, `unsigned int gvl`)
- `uint32xm4_t vnclipuvi_mask_uint32xm4_uint64xm8` (`uint32xm4_t merge`, `uint64xm8_t a`, `const unsigned int b`, `e32xm4_t mask`, `unsigned int gvl`)
- `uint8xm1_t vnclipuvi_mask_uint8xm1_uint16xm2` (`uint8xm1_t merge`, `uint16xm2_t a`, `const unsigned char b`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8xm2_t vnclipuvi_mask_uint8xm2_uint16xm4` (`uint8xm2_t merge`, `uint16xm4_t a`, `const unsigned char b`, `e8xm2_t mask`, `unsigned int gvl`)
- `uint8xm4_t vnclipuvi_mask_uint8xm4_uint16xm8` (`uint8xm4_t merge`, `uint16xm8_t a`, `const unsigned char b`, `e8xm4_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = clip(roundoff(a[element], b))
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.46 Elementwise unsigned vector-vector unsigned integer narrow clip****Instruction:** [`vnclipu.vv`']**Prototypes:**

- `uint16xm1_t vnclipuvv_uint16xm1_uint32xm2` (`uint32xm2_t a`, `uint16xm1_t b`, `unsigned int gvl`)
- `uint16xm2_t vnclipuvv_uint16xm2_uint32xm4` (`uint32xm4_t a`, `uint16xm2_t b`, `unsigned int gvl`)
- `uint16xm4_t vnclipuvv_uint16xm4_uint32xm8` (`uint32xm8_t a`, `uint16xm4_t b`, `unsigned int gvl`)
- `uint32xm1_t vnclipuvv_uint32xm1_uint64xm2` (`uint64xm2_t a`, `uint32xm1_t b`, `unsigned int gvl`)
- `uint32xm2_t vnclipuvv_uint32xm2_uint64xm4` (`uint64xm4_t a`, `uint32xm2_t b`, `unsigned int gvl`)
- `uint32xm4_t vnclipuvv_uint32xm4_uint64xm8` (`uint64xm8_t a`, `uint32xm4_t b`, `unsigned int gvl`)
- `uint8xm1_t vnclipuvv_uint8xm1_uint16xm2` (`uint16xm2_t a`, `uint8xm1_t b`, `unsigned int gvl`)
- `uint8xm2_t vnclipuvv_uint8xm2_uint16xm4` (`uint16xm4_t a`, `uint8xm2_t b`, `unsigned int gvl`)
- `uint8xm4_t vnclipuvv_uint8xm4_uint16xm8` (`uint16xm8_t a`, `uint8xm4_t b`, `unsigned int gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = clip(roundoff(a[element], b[element]))
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vnclipuvv_mask_uint16xm1_uint32xm2` (`uint16xm1_t merge`, `uint32xm2_t a`, `uint16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint16xm2_t vnclipuvv_mask_uint16xm2_uint32xm4` (`uint16xm2_t merge`, `uint32xm4_t a`, `uint16xm2_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint16xm4_t vnclipuvv_mask_uint16xm4_uint32xm8` (`uint16xm4_t merge`, `uint32xm8_t a`, `uint16xm4_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint32xm1_t vnclipuvv_mask_uint32xm1_uint64xm2` (`uint32xm1_t merge`, `uint64xm2_t a`, `uint32xm1_t b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vnclipuvv_mask_uint32xm2_uint64xm4` (`uint32xm2_t merge`, `uint64xm4_t a`, `uint32xm2_t b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vnclipuvv_mask_uint32xm4_uint64xm8` (`uint32xm4_t merge`, `uint64xm8_t a`, `uint32xm4_t b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint8xm1_t vnclipuvv_mask_uint8xm1_uint16xm2` (`uint8xm1_t merge`, `uint16xm2_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vnclipuvv_mask_uint8xm2_uint16xm4` (`uint8xm2_t merge`, `uint16xm4_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vnclipuvv_mask_uint8xm4_uint16xm8` (`uint8xm4_t merge`, `uint16xm8_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = clip(roundoff(a[element], b[element]))
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.47 Elementwise unsigned vector-scalar unsigned integer narrow clip****Instruction:** [`'vnclipu.vx'`]**Prototypes:**

- `uint16xm1_t vnclipuvx_uint16xm1_uint32xm2` (`uint32xm2_t a`, unsigned short `b`, unsigned int `gvl`)



- `uint16xm2_t vnclipuvx_uint16xm2_uint32xm4 (uint32xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vnclipuvx_uint16xm4_uint32xm8 (uint32xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vnclipuvx_uint32xm1_uint64xm2 (uint64xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vnclipuvx_uint32xm2_uint64xm4 (uint64xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vnclipuvx_uint32xm4_uint64xm8 (uint64xm8_t a, unsigned int b, unsigned int gvl)`
- `uint8xm1_t vnclipuvx_uint8xm1_uint16xm2 (uint16xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vnclipuvx_uint8xm2_uint16xm4 (uint16xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vnclipuvx_uint8xm4_uint16xm8 (uint16xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = clip(roundoff(a[element], b))
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vnclipuvx_mask_uint16xm1_uint32xm2 (uint16xm1_t merge, uint32xm2_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vnclipuvx_mask_uint16xm2_uint32xm4 (uint16xm2_t merge, uint32xm4_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vnclipuvx_mask_uint16xm4_uint32xm8 (uint16xm4_t merge, uint32xm8_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint32xm1_t vnclipuvx_mask_uint32xm1_uint64xm2 (uint32xm1_t merge, uint64xm2_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vnclipuvx_mask_uint32xm2_uint64xm4 (uint32xm2_t merge, uint64xm4_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vnclipuvx_mask_uint32xm4_uint64xm8 (uint32xm4_t merge, uint64xm8_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint8xm1_t vnclipuvx_mask_uint8xm1_uint16xm2 (uint8xm1_t merge, uint16xm2_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vnclipuvx_mask_uint8xm2_uint16xm4 (uint8xm2_t merge, uint16xm4_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vnclipuvx_mask_uint8xm4_uint16xm8 (uint8xm4_t merge, uint16xm8_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = clip(roundoff(a[element], b))
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.7.48 Elementwise vector-vector multiply-subtraction, overwrite minuend

**Instruction:** ['vnmsacv.vv']

**Prototypes:**

- *int16xm1\_t vnmsacvv\_int16xm1* (*int16xm1\_t a, int16xm1\_t b, int16xm1\_t result*, unsigned int gvl)
- *int16xm2\_t vnmsacvv\_int16xm2* (*int16xm2\_t a, int16xm2\_t b, int16xm2\_t result*, unsigned int gvl)
- *int16xm4\_t vnmsacvv\_int16xm4* (*int16xm4\_t a, int16xm4\_t b, int16xm4\_t result*, unsigned int gvl)
- *int16xm8\_t vnmsacvv\_int16xm8* (*int16xm8\_t a, int16xm8\_t b, int16xm8\_t result*, unsigned int gvl)
- *int32xm1\_t vnmsacvv\_int32xm1* (*int32xm1\_t a, int32xm1\_t b, int32xm1\_t result*, unsigned int gvl)
- *int32xm2\_t vnmsacvv\_int32xm2* (*int32xm2\_t a, int32xm2\_t b, int32xm2\_t result*, unsigned int gvl)
- *int32xm4\_t vnmsacvv\_int32xm4* (*int32xm4\_t a, int32xm4\_t b, int32xm4\_t result*, unsigned int gvl)
- *int32xm8\_t vnmsacvv\_int32xm8* (*int32xm8\_t a, int32xm8\_t b, int32xm8\_t result*, unsigned int gvl)
- *int64xm1\_t vnmsacvv\_int64xm1* (*int64xm1\_t a, int64xm1\_t b, int64xm1\_t result*, unsigned int gvl)
- *int64xm2\_t vnmsacvv\_int64xm2* (*int64xm2\_t a, int64xm2\_t b, int64xm2\_t result*, unsigned int gvl)
- *int64xm4\_t vnmsacvv\_int64xm4* (*int64xm4\_t a, int64xm4\_t b, int64xm4\_t result*, unsigned int gvl)
- *int64xm8\_t vnmsacvv\_int64xm8* (*int64xm8\_t a, int64xm8\_t b, int64xm8\_t result*, unsigned int gvl)
- *int8xm1\_t vnmsacvv\_int8xm1* (*int8xm1\_t a, int8xm1\_t b, int8xm1\_t result*, unsigned int gvl)
- *int8xm2\_t vnmsacvv\_int8xm2* (*int8xm2\_t a, int8xm2\_t b, int8xm2\_t result*, unsigned int gvl)
- *int8xm4\_t vnmsacvv\_int8xm4* (*int8xm4\_t a, int8xm4\_t b, int8xm4\_t result*, unsigned int gvl)
- *int8xm8\_t vnmsacvv\_int8xm8* (*int8xm8\_t a, int8xm8\_t b, int8xm8\_t result*, unsigned int gvl)
- *uint16xm1\_t vnmsacvv\_uint16xm1* (*uint16xm1\_t a, uint16xm1\_t b, uint16xm1\_t result*, unsigned int gvl)
- *uint16xm2\_t vnmsacvv\_uint16xm2* (*uint16xm2\_t a, uint16xm2\_t b, uint16xm2\_t result*, unsigned int gvl)
- *uint16xm4\_t vnmsacvv\_uint16xm4* (*uint16xm4\_t a, uint16xm4\_t b, uint16xm4\_t result*, unsigned int gvl)
- *uint16xm8\_t vnmsacvv\_uint16xm8* (*uint16xm8\_t a, uint16xm8\_t b, uint16xm8\_t result*, unsigned int gvl)
- *uint32xm1\_t vnmsacvv\_uint32xm1* (*uint32xm1\_t a, uint32xm1\_t b, uint32xm1\_t result*, unsigned int gvl)
- *uint32xm2\_t vnmsacvv\_uint32xm2* (*uint32xm2\_t a, uint32xm2\_t b, uint32xm2\_t result*, unsigned int gvl)
- *uint32xm4\_t vnmsacvv\_uint32xm4* (*uint32xm4\_t a, uint32xm4\_t b, uint32xm4\_t result*, unsigned int gvl)

- `uint32xm8_t vnmsacvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, uint32xm8_t result, unsigned int gvl)`
- `uint64xm1_t vnmsacvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, uint64xm1_t result, unsigned int gvl)`
- `uint64xm2_t vnmsacvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, uint64xm2_t result, unsigned int gvl)`
- `uint64xm4_t vnmsacvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, uint64xm4_t result, unsigned int gvl)`
- `uint64xm8_t vnmsacvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, uint64xm8_t result, unsigned int gvl)`
- `uint8xm1_t vnmsacvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, uint8xm1_t result, unsigned int gvl)`
- `uint8xm2_t vnmsacvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, uint8xm2_t result, unsigned int gvl)`
- `uint8xm4_t vnmsacvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, uint8xm4_t result, unsigned int gvl)`
- `uint8xm8_t vnmsacvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, uint8xm8_t result, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = -(a[element] * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vnmsacvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, int16xm1_t result, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vnmsacvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, int16xm2_t result, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vnmsacvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, int16xm4_t result, e16xm4_t mask, unsigned int gvl)`
- `int32xm1_t vnmsacvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, int32xm1_t result, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vnmsacvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, int32xm2_t result, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vnmsacvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, int32xm4_t result, e32xm4_t mask, unsigned int gvl)`
- `int64xm1_t vnmsacvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, int64xm1_t result, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vnmsacvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, int64xm2_t result, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vnmsacvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, int64xm4_t result, e64xm4_t mask, unsigned int gvl)`
- `int8xm1_t vnmsacvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, int8xm1_t result, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vnmsacvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, int8xm2_t result, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vnmsacvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, int8xm4_t result, e8xm4_t mask, unsigned int gvl)`
- `uint16xm1_t vnmsacvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, uint16xm1_t result, e16xm1_t mask, unsigned int gvl)`

- `uint16xm2_t vnmsacvv_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, `uint16xm2_t b`, `uint16xm2_t result`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint16xm4_t vnmsacvv_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, `uint16xm4_t b`, `uint16xm4_t result`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint32xm1_t vnmsacvv_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, `uint32xm1_t b`, `uint32xm1_t result`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vnmsacvv_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, `uint32xm2_t b`, `uint32xm2_t result`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vnmsacvv_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, `uint32xm4_t b`, `uint32xm4_t result`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint64xm1_t vnmsacvv_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, `uint64xm1_t b`, `uint64xm1_t result`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vnmsacvv_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, `uint64xm2_t b`, `uint64xm2_t result`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vnmsacvv_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, `uint64xm4_t b`, `uint64xm4_t result`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint8xm1_t vnmsacvv_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `uint8xm1_t result`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vnmsacvv_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `uint8xm2_t result`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vnmsacvv_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `uint8xm4_t result`, `e8xm4_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = -(a[element] * b[element]) + result[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.49 Elementwise vector-scalar multiply-subtraction, overwrite minuend

Instruction: ['vnmsac.vx']

Prototypes:

- `int16xm1_t vnmsacvx_int16xm1` (short `a`, `int16xm1_t b`, `int16xm1_t result`, unsigned int `gvl`)
- `int16xm2_t vnmsacvx_int16xm2` (short `a`, `int16xm2_t b`, `int16xm2_t result`, unsigned int `gvl`)
- `int16xm4_t vnmsacvx_int16xm4` (short `a`, `int16xm4_t b`, `int16xm4_t result`, unsigned int `gvl`)
- `int16xm8_t vnmsacvx_int16xm8` (short `a`, `int16xm8_t b`, `int16xm8_t result`, unsigned int `gvl`)
- `int32xm1_t vnmsacvx_int32xm1` (int `a`, `int32xm1_t b`, `int32xm1_t result`, unsigned int `gvl`)
- `int32xm2_t vnmsacvx_int32xm2` (int `a`, `int32xm2_t b`, `int32xm2_t result`, unsigned int `gvl`)
- `int32xm4_t vnmsacvx_int32xm4` (int `a`, `int32xm4_t b`, `int32xm4_t result`, unsigned int `gvl`)
- `int32xm8_t vnmsacvx_int32xm8` (int `a`, `int32xm8_t b`, `int32xm8_t result`, unsigned int `gvl`)
- `int64xm1_t vnmsacvx_int64xm1` (long `a`, `int64xm1_t b`, `int64xm1_t result`, unsigned int `gvl`)

- `int64xm2_t vnmsacvx_int64xm2` (long *a*, `int64xm2_t b`, `int64xm2_t result`, unsigned int *gvl*)
- `int64xm4_t vnmsacvx_int64xm4` (long *a*, `int64xm4_t b`, `int64xm4_t result`, unsigned int *gvl*)
- `int64xm8_t vnmsacvx_int64xm8` (long *a*, `int64xm8_t b`, `int64xm8_t result`, unsigned int *gvl*)
- `int8xm1_t vnmsacvx_int8xm1` (signed char *a*, `int8xm1_t b`, `int8xm1_t result`, unsigned int *gvl*)
- `int8xm2_t vnmsacvx_int8xm2` (signed char *a*, `int8xm2_t b`, `int8xm2_t result`, unsigned int *gvl*)
- `int8xm4_t vnmsacvx_int8xm4` (signed char *a*, `int8xm4_t b`, `int8xm4_t result`, unsigned int *gvl*)
- `int8xm8_t vnmsacvx_int8xm8` (signed char *a*, `int8xm8_t b`, `int8xm8_t result`, unsigned int *gvl*)
- `uint16xm1_t vnmsacvx_uint16xm1` (unsigned short *a*, `uint16xm1_t b`, `uint16xm1_t result`, unsigned int *gvl*)
- `uint16xm2_t vnmsacvx_uint16xm2` (unsigned short *a*, `uint16xm2_t b`, `uint16xm2_t result`, unsigned int *gvl*)
- `uint16xm4_t vnmsacvx_uint16xm4` (unsigned short *a*, `uint16xm4_t b`, `uint16xm4_t result`, unsigned int *gvl*)
- `uint16xm8_t vnmsacvx_uint16xm8` (unsigned short *a*, `uint16xm8_t b`, `uint16xm8_t result`, unsigned int *gvl*)
- `uint32xm1_t vnmsacvx_uint32xm1` (unsigned int *a*, `uint32xm1_t b`, `uint32xm1_t result`, unsigned int *gvl*)
- `uint32xm2_t vnmsacvx_uint32xm2` (unsigned int *a*, `uint32xm2_t b`, `uint32xm2_t result`, unsigned int *gvl*)
- `uint32xm4_t vnmsacvx_uint32xm4` (unsigned int *a*, `uint32xm4_t b`, `uint32xm4_t result`, unsigned int *gvl*)
- `uint32xm8_t vnmsacvx_uint32xm8` (unsigned int *a*, `uint32xm8_t b`, `uint32xm8_t result`, unsigned int *gvl*)
- `uint64xm1_t vnmsacvx_uint64xm1` (unsigned long *a*, `uint64xm1_t b`, `uint64xm1_t result`, unsigned int *gvl*)
- `uint64xm2_t vnmsacvx_uint64xm2` (unsigned long *a*, `uint64xm2_t b`, `uint64xm2_t result`, unsigned int *gvl*)
- `uint64xm4_t vnmsacvx_uint64xm4` (unsigned long *a*, `uint64xm4_t b`, `uint64xm4_t result`, unsigned int *gvl*)
- `uint64xm8_t vnmsacvx_uint64xm8` (unsigned long *a*, `uint64xm8_t b`, `uint64xm8_t result`, unsigned int *gvl*)
- `uint8xm1_t vnmsacvx_uint8xm1` (unsigned char *a*, `uint8xm1_t b`, `uint8xm1_t result`, unsigned int *gvl*)
- `uint8xm2_t vnmsacvx_uint8xm2` (unsigned char *a*, `uint8xm2_t b`, `uint8xm2_t result`, unsigned int *gvl*)
- `uint8xm4_t vnmsacvx_uint8xm4` (unsigned char *a*, `uint8xm4_t b`, `uint8xm4_t result`, unsigned int *gvl*)
- `uint8xm8_t vnmsacvx_uint8xm8` (unsigned char *a*, `uint8xm8_t b`, `uint8xm8_t result`, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = +(a * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vnmsacvx_mask_int16xm1` (`int16xm1_t merge`, short *a*, `int16xm1_t b`, `int16xm1_t result`, `e16xm1_t mask`, unsigned int *gvl*)



- *int16xm2\_t vnmsacvx\_mask\_int16xm2* (*int16xm2\_t* merge, short *a*, *int16xm2\_t* *b*, *int16xm2\_t* *result*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *int16xm4\_t vnmsacvx\_mask\_int16xm4* (*int16xm4\_t* merge, short *a*, *int16xm4\_t* *b*, *int16xm4\_t* *result*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *int32xm1\_t vnmsacvx\_mask\_int32xm1* (*int32xm1\_t* merge, int *a*, *int32xm1\_t* *b*, *int32xm1\_t* *result*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *int32xm2\_t vnmsacvx\_mask\_int32xm2* (*int32xm2\_t* merge, int *a*, *int32xm2\_t* *b*, *int32xm2\_t* *result*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *int32xm4\_t vnmsacvx\_mask\_int32xm4* (*int32xm4\_t* merge, int *a*, *int32xm4\_t* *b*, *int32xm4\_t* *result*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *int64xm1\_t vnmsacvx\_mask\_int64xm1* (*int64xm1\_t* merge, long *a*, *int64xm1\_t* *b*, *int64xm1\_t* *result*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *int64xm2\_t vnmsacvx\_mask\_int64xm2* (*int64xm2\_t* merge, long *a*, *int64xm2\_t* *b*, *int64xm2\_t* *result*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *int64xm4\_t vnmsacvx\_mask\_int64xm4* (*int64xm4\_t* merge, long *a*, *int64xm4\_t* *b*, *int64xm4\_t* *result*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *int8xm1\_t vnmsacvx\_mask\_int8xm1* (*int8xm1\_t* merge, signed char *a*, *int8xm1\_t* *b*, *int8xm1\_t* *result*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *int8xm2\_t vnmsacvx\_mask\_int8xm2* (*int8xm2\_t* merge, signed char *a*, *int8xm2\_t* *b*, *int8xm2\_t* *result*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- *int8xm4\_t vnmsacvx\_mask\_int8xm4* (*int8xm4\_t* merge, signed char *a*, *int8xm4\_t* *b*, *int8xm4\_t* *result*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- *uint16xm1\_t vnmsacvx\_mask\_uint16xm1* (*uint16xm1\_t* merge, unsigned short *a*, *uint16xm1\_t* *b*, *uint16xm1\_t* *result*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *uint16xm2\_t vnmsacvx\_mask\_uint16xm2* (*uint16xm2\_t* merge, unsigned short *a*, *uint16xm2\_t* *b*, *uint16xm2\_t* *result*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *uint16xm4\_t vnmsacvx\_mask\_uint16xm4* (*uint16xm4\_t* merge, unsigned short *a*, *uint16xm4\_t* *b*, *uint16xm4\_t* *result*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *uint32xm1\_t vnmsacvx\_mask\_uint32xm1* (*uint32xm1\_t* merge, unsigned int *a*, *uint32xm1\_t* *b*, *uint32xm1\_t* *result*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *uint32xm2\_t vnmsacvx\_mask\_uint32xm2* (*uint32xm2\_t* merge, unsigned int *a*, *uint32xm2\_t* *b*, *uint32xm2\_t* *result*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *uint32xm4\_t vnmsacvx\_mask\_uint32xm4* (*uint32xm4\_t* merge, unsigned int *a*, *uint32xm4\_t* *b*, *uint32xm4\_t* *result*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *uint64xm1\_t vnmsacvx\_mask\_uint64xm1* (*uint64xm1\_t* merge, unsigned long *a*, *uint64xm1\_t* *b*, *uint64xm1\_t* *result*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *uint64xm2\_t vnmsacvx\_mask\_uint64xm2* (*uint64xm2\_t* merge, unsigned long *a*, *uint64xm2\_t* *b*, *uint64xm2\_t* *result*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *uint64xm4\_t vnmsacvx\_mask\_uint64xm4* (*uint64xm4\_t* merge, unsigned long *a*, *uint64xm4\_t* *b*, *uint64xm4\_t* *result*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *uint8xm1\_t vnmsacvx\_mask\_uint8xm1* (*uint8xm1\_t* merge, unsigned char *a*, *uint8xm1\_t* *b*, *uint8xm1\_t* *result*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *uint8xm2\_t vnmsacvx\_mask\_uint8xm2* (*uint8xm2\_t* merge, unsigned char *a*, *uint8xm2\_t* *b*, *uint8xm2\_t* *result*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- *uint8xm4\_t vnmsacvx\_mask\_uint8xm4* (*uint8xm4\_t* merge, unsigned char *a*, *uint8xm4\_t* *b*, *uint8xm4\_t* *result*, *e8xm4\_t* *mask*, unsigned int *gvl*)



**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = +(a * b[element]) + result[element]
    else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.7.50 Elementwise vector-vector multiply-subtraction, overwrite multiplicand****Instruction:** ['vnmsub.vv']**Prototypes:**

- *int16xm1\_t vnmsubvv\_int16xm1* (*int16xm1\_t a, int16xm1\_t b, int16xm1\_t result*, unsigned int gvl)
- *int16xm2\_t vnmsubvv\_int16xm2* (*int16xm2\_t a, int16xm2\_t b, int16xm2\_t result*, unsigned int gvl)
- *int16xm4\_t vnmsubvv\_int16xm4* (*int16xm4\_t a, int16xm4\_t b, int16xm4\_t result*, unsigned int gvl)
- *int16xm8\_t vnmsubvv\_int16xm8* (*int16xm8\_t a, int16xm8\_t b, int16xm8\_t result*, unsigned int gvl)
- *int32xm1\_t vnmsubvv\_int32xm1* (*int32xm1\_t a, int32xm1\_t b, int32xm1\_t result*, unsigned int gvl)
- *int32xm2\_t vnmsubvv\_int32xm2* (*int32xm2\_t a, int32xm2\_t b, int32xm2\_t result*, unsigned int gvl)
- *int32xm4\_t vnmsubvv\_int32xm4* (*int32xm4\_t a, int32xm4\_t b, int32xm4\_t result*, unsigned int gvl)
- *int32xm8\_t vnmsubvv\_int32xm8* (*int32xm8\_t a, int32xm8\_t b, int32xm8\_t result*, unsigned int gvl)
- *int64xm1\_t vnmsubvv\_int64xm1* (*int64xm1\_t a, int64xm1\_t b, int64xm1\_t result*, unsigned int gvl)
- *int64xm2\_t vnmsubvv\_int64xm2* (*int64xm2\_t a, int64xm2\_t b, int64xm2\_t result*, unsigned int gvl)
- *int64xm4\_t vnmsubvv\_int64xm4* (*int64xm4\_t a, int64xm4\_t b, int64xm4\_t result*, unsigned int gvl)
- *int64xm8\_t vnmsubvv\_int64xm8* (*int64xm8\_t a, int64xm8\_t b, int64xm8\_t result*, unsigned int gvl)
- *int8xm1\_t vnmsubvv\_int8xm1* (*int8xm1\_t a, int8xm1\_t b, int8xm1\_t result*, unsigned int gvl)
- *int8xm2\_t vnmsubvv\_int8xm2* (*int8xm2\_t a, int8xm2\_t b, int8xm2\_t result*, unsigned int gvl)
- *int8xm4\_t vnmsubvv\_int8xm4* (*int8xm4\_t a, int8xm4\_t b, int8xm4\_t result*, unsigned int gvl)
- *int8xm8\_t vnmsubvv\_int8xm8* (*int8xm8\_t a, int8xm8\_t b, int8xm8\_t result*, unsigned int gvl)
- *uint16xm1\_t vnmsubvv\_uint16xm1* (*uint16xm1\_t a, uint16xm1\_t b, uint16xm1\_t result*, unsigned int gvl)
- *uint16xm2\_t vnmsubvv\_uint16xm2* (*uint16xm2\_t a, uint16xm2\_t b, uint16xm2\_t result*, unsigned int gvl)
- *uint16xm4\_t vnmsubvv\_uint16xm4* (*uint16xm4\_t a, uint16xm4\_t b, uint16xm4\_t result*, unsigned int gvl)
- *uint16xm8\_t vnmsubvv\_uint16xm8* (*uint16xm8\_t a, uint16xm8\_t b, uint16xm8\_t result*, unsigned int gvl)
- *uint32xm1\_t vnmsubvv\_uint32xm1* (*uint32xm1\_t a, uint32xm1\_t b, uint32xm1\_t result*, unsigned int gvl)
- *uint32xm2\_t vnmsubvv\_uint32xm2* (*uint32xm2\_t a, uint32xm2\_t b, uint32xm2\_t result*, unsigned int gvl)
- *uint32xm4\_t vnmsubvv\_uint32xm4* (*uint32xm4\_t a, uint32xm4\_t b, uint32xm4\_t result*, unsigned int gvl)

- `uint32xm8_t vnmsubvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, uint32xm8_t result, unsigned int gvl)`
- `uint64xm1_t vnmsubvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, uint64xm1_t result, unsigned int gvl)`
- `uint64xm2_t vnmsubvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, uint64xm2_t result, unsigned int gvl)`
- `uint64xm4_t vnmsubvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, uint64xm4_t result, unsigned int gvl)`
- `uint64xm8_t vnmsubvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, uint64xm8_t result, unsigned int gvl)`
- `uint8xm1_t vnmsubvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, uint8xm1_t result, unsigned int gvl)`
- `uint8xm2_t vnmsubvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, uint8xm2_t result, unsigned int gvl)`
- `uint8xm4_t vnmsubvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, uint8xm4_t result, unsigned int gvl)`
- `uint8xm8_t vnmsubvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, uint8xm8_t result, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = -(a[element] * result[element]) + b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vnmsubvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, int16xm1_t result, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vnmsubvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, int16xm2_t result, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vnmsubvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, int16xm4_t result, e16xm4_t mask, unsigned int gvl)`
- `int32xm1_t vnmsubvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, int32xm1_t result, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vnmsubvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, int32xm2_t result, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vnmsubvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, int32xm4_t result, e32xm4_t mask, unsigned int gvl)`
- `int64xm1_t vnmsubvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, int64xm1_t result, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vnmsubvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, int64xm2_t result, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vnmsubvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, int64xm4_t result, e64xm4_t mask, unsigned int gvl)`
- `int8xm1_t vnmsubvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, int8xm1_t result, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vnmsubvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, int8xm2_t result, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vnmsubvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, int8xm4_t result, e8xm4_t mask, unsigned int gvl)`
- `uint16xm1_t vnmsubvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, uint16xm1_t result, e16xm1_t mask, unsigned int gvl)`

- `uint16xm2_t vnmsubvv_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, `uint16xm2_t b`, `uint16xm2_t result`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint16xm4_t vnmsubvv_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, `uint16xm4_t b`, `uint16xm4_t result`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint32xm1_t vnmsubvv_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, `uint32xm1_t b`, `uint32xm1_t result`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vnmsubvv_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, `uint32xm2_t b`, `uint32xm2_t result`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vnmsubvv_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, `uint32xm4_t b`, `uint32xm4_t result`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint64xm1_t vnmsubvv_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, `uint64xm1_t b`, `uint64xm1_t result`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vnmsubvv_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, `uint64xm2_t b`, `uint64xm2_t result`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vnmsubvv_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, `uint64xm4_t b`, `uint64xm4_t result`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint8xm1_t vnmsubvv_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `uint8xm1_t result`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vnmsubvv_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `uint8xm2_t result`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vnmsubvv_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `uint8xm4_t result`, `e8xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = -(a[element] * result[element]) + b[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.51 Elementwise vector-scalar multiply-subtraction, overwrite multiplicand****Instruction:** [`'vnmsub.vx'`]**Prototypes:**

- `int16xm1_t vnmsubvx_int16xm1` (short `a`, `int16xm1_t b`, `int16xm1_t result`, unsigned int `gvl`)
- `int16xm2_t vnmsubvx_int16xm2` (short `a`, `int16xm2_t b`, `int16xm2_t result`, unsigned int `gvl`)
- `int16xm4_t vnmsubvx_int16xm4` (short `a`, `int16xm4_t b`, `int16xm4_t result`, unsigned int `gvl`)
- `int16xm8_t vnmsubvx_int16xm8` (short `a`, `int16xm8_t b`, `int16xm8_t result`, unsigned int `gvl`)
- `int32xm1_t vnmsubvx_int32xm1` (int `a`, `int32xm1_t b`, `int32xm1_t result`, unsigned int `gvl`)
- `int32xm2_t vnmsubvx_int32xm2` (int `a`, `int32xm2_t b`, `int32xm2_t result`, unsigned int `gvl`)
- `int32xm4_t vnmsubvx_int32xm4` (int `a`, `int32xm4_t b`, `int32xm4_t result`, unsigned int `gvl`)
- `int32xm8_t vnmsubvx_int32xm8` (int `a`, `int32xm8_t b`, `int32xm8_t result`, unsigned int `gvl`)
- `int64xm1_t vnmsubvx_int64xm1` (long `a`, `int64xm1_t b`, `int64xm1_t result`, unsigned int `gvl`)

- `int64xm2_t vnmsubvx_int64xm2` (long *a*, `int64xm2_t b`, `int64xm2_t result`, unsigned int *gvl*)
- `int64xm4_t vnmsubvx_int64xm4` (long *a*, `int64xm4_t b`, `int64xm4_t result`, unsigned int *gvl*)
- `int64xm8_t vnmsubvx_int64xm8` (long *a*, `int64xm8_t b`, `int64xm8_t result`, unsigned int *gvl*)
- `int8xm1_t vnmsubvx_int8xm1` (signed char *a*, `int8xm1_t b`, `int8xm1_t result`, unsigned int *gvl*)
- `int8xm2_t vnmsubvx_int8xm2` (signed char *a*, `int8xm2_t b`, `int8xm2_t result`, unsigned int *gvl*)
- `int8xm4_t vnmsubvx_int8xm4` (signed char *a*, `int8xm4_t b`, `int8xm4_t result`, unsigned int *gvl*)
- `int8xm8_t vnmsubvx_int8xm8` (signed char *a*, `int8xm8_t b`, `int8xm8_t result`, unsigned int *gvl*)
- `uint16xm1_t vnmsubvx_uint16xm1` (unsigned short *a*, `uint16xm1_t b`, `uint16xm1_t result`, unsigned int *gvl*)
- `uint16xm2_t vnmsubvx_uint16xm2` (unsigned short *a*, `uint16xm2_t b`, `uint16xm2_t result`, unsigned int *gvl*)
- `uint16xm4_t vnmsubvx_uint16xm4` (unsigned short *a*, `uint16xm4_t b`, `uint16xm4_t result`, unsigned int *gvl*)
- `uint16xm8_t vnmsubvx_uint16xm8` (unsigned short *a*, `uint16xm8_t b`, `uint16xm8_t result`, unsigned int *gvl*)
- `uint32xm1_t vnmsubvx_uint32xm1` (unsigned int *a*, `uint32xm1_t b`, `uint32xm1_t result`, unsigned int *gvl*)
- `uint32xm2_t vnmsubvx_uint32xm2` (unsigned int *a*, `uint32xm2_t b`, `uint32xm2_t result`, unsigned int *gvl*)
- `uint32xm4_t vnmsubvx_uint32xm4` (unsigned int *a*, `uint32xm4_t b`, `uint32xm4_t result`, unsigned int *gvl*)
- `uint32xm8_t vnmsubvx_uint32xm8` (unsigned int *a*, `uint32xm8_t b`, `uint32xm8_t result`, unsigned int *gvl*)
- `uint64xm1_t vnmsubvx_uint64xm1` (unsigned long *a*, `uint64xm1_t b`, `uint64xm1_t result`, unsigned int *gvl*)
- `uint64xm2_t vnmsubvx_uint64xm2` (unsigned long *a*, `uint64xm2_t b`, `uint64xm2_t result`, unsigned int *gvl*)
- `uint64xm4_t vnmsubvx_uint64xm4` (unsigned long *a*, `uint64xm4_t b`, `uint64xm4_t result`, unsigned int *gvl*)
- `uint64xm8_t vnmsubvx_uint64xm8` (unsigned long *a*, `uint64xm8_t b`, `uint64xm8_t result`, unsigned int *gvl*)
- `uint8xm1_t vnmsubvx_uint8xm1` (unsigned char *a*, `uint8xm1_t b`, `uint8xm1_t result`, unsigned int *gvl*)
- `uint8xm2_t vnmsubvx_uint8xm2` (unsigned char *a*, `uint8xm2_t b`, `uint8xm2_t result`, unsigned int *gvl*)
- `uint8xm4_t vnmsubvx_uint8xm4` (unsigned char *a*, `uint8xm4_t b`, `uint8xm4_t result`, unsigned int *gvl*)
- `uint8xm8_t vnmsubvx_uint8xm8` (unsigned char *a*, `uint8xm8_t b`, `uint8xm8_t result`, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = +(a * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vnmsubvx_mask_int16xm1` (`int16xm1_t merge`, short *a*, `int16xm1_t b`, `int16xm1_t result`, `e16xm1_t mask`, unsigned int *gvl*)

- *int16xm2\_t vnmsubvx\_mask\_int16xm2* (*int16xm2\_t* merge, short *a*, *int16xm2\_t* *b*, *int16xm2\_t* *result*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *int16xm4\_t vnmsubvx\_mask\_int16xm4* (*int16xm4\_t* merge, short *a*, *int16xm4\_t* *b*, *int16xm4\_t* *result*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *int32xm1\_t vnmsubvx\_mask\_int32xm1* (*int32xm1\_t* merge, int *a*, *int32xm1\_t* *b*, *int32xm1\_t* *result*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *int32xm2\_t vnmsubvx\_mask\_int32xm2* (*int32xm2\_t* merge, int *a*, *int32xm2\_t* *b*, *int32xm2\_t* *result*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *int32xm4\_t vnmsubvx\_mask\_int32xm4* (*int32xm4\_t* merge, int *a*, *int32xm4\_t* *b*, *int32xm4\_t* *result*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *int64xm1\_t vnmsubvx\_mask\_int64xm1* (*int64xm1\_t* merge, long *a*, *int64xm1\_t* *b*, *int64xm1\_t* *result*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *int64xm2\_t vnmsubvx\_mask\_int64xm2* (*int64xm2\_t* merge, long *a*, *int64xm2\_t* *b*, *int64xm2\_t* *result*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *int64xm4\_t vnmsubvx\_mask\_int64xm4* (*int64xm4\_t* merge, long *a*, *int64xm4\_t* *b*, *int64xm4\_t* *result*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *int8xm1\_t vnmsubvx\_mask\_int8xm1* (*int8xm1\_t* merge, signed char *a*, *int8xm1\_t* *b*, *int8xm1\_t* *result*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *int8xm2\_t vnmsubvx\_mask\_int8xm2* (*int8xm2\_t* merge, signed char *a*, *int8xm2\_t* *b*, *int8xm2\_t* *result*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- *int8xm4\_t vnmsubvx\_mask\_int8xm4* (*int8xm4\_t* merge, signed char *a*, *int8xm4\_t* *b*, *int8xm4\_t* *result*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- *uint16xm1\_t vnmsubvx\_mask\_uint16xm1* (*uint16xm1\_t* merge, unsigned short *a*, *uint16xm1\_t* *b*, *uint16xm1\_t* *result*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *uint16xm2\_t vnmsubvx\_mask\_uint16xm2* (*uint16xm2\_t* merge, unsigned short *a*, *uint16xm2\_t* *b*, *uint16xm2\_t* *result*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *uint16xm4\_t vnmsubvx\_mask\_uint16xm4* (*uint16xm4\_t* merge, unsigned short *a*, *uint16xm4\_t* *b*, *uint16xm4\_t* *result*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *uint32xm1\_t vnmsubvx\_mask\_uint32xm1* (*uint32xm1\_t* merge, unsigned int *a*, *uint32xm1\_t* *b*, *uint32xm1\_t* *result*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *uint32xm2\_t vnmsubvx\_mask\_uint32xm2* (*uint32xm2\_t* merge, unsigned int *a*, *uint32xm2\_t* *b*, *uint32xm2\_t* *result*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *uint32xm4\_t vnmsubvx\_mask\_uint32xm4* (*uint32xm4\_t* merge, unsigned int *a*, *uint32xm4\_t* *b*, *uint32xm4\_t* *result*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *uint64xm1\_t vnmsubvx\_mask\_uint64xm1* (*uint64xm1\_t* merge, unsigned long *a*, *uint64xm1\_t* *b*, *uint64xm1\_t* *result*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *uint64xm2\_t vnmsubvx\_mask\_uint64xm2* (*uint64xm2\_t* merge, unsigned long *a*, *uint64xm2\_t* *b*, *uint64xm2\_t* *result*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *uint64xm4\_t vnmsubvx\_mask\_uint64xm4* (*uint64xm4\_t* merge, unsigned long *a*, *uint64xm4\_t* *b*, *uint64xm4\_t* *result*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *uint8xm1\_t vnmsubvx\_mask\_uint8xm1* (*uint8xm1\_t* merge, unsigned char *a*, *uint8xm1\_t* *b*, *uint8xm1\_t* *result*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *uint8xm2\_t vnmsubvx\_mask\_uint8xm2* (*uint8xm2\_t* merge, unsigned char *a*, *uint8xm2\_t* *b*, *uint8xm2\_t* *result*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- *uint8xm4\_t vnmsubvx\_mask\_uint8xm4* (*uint8xm4\_t* merge, unsigned char *a*, *uint8xm4\_t* *b*, *uint8xm4\_t* *result*, *e8xm4\_t* *mask*, unsigned int *gvl*)



**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = +(a * b[element]) + result[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.52 Narrowing elementwise vector-immediate arithmetic shift right****Instruction:** ['vnsra.vi']**Prototypes:**

- *int16xm1\_t vnsravi\_int16xm1\_int32xm2* (*int32xm2\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *int16xm2\_t vnsravi\_int16xm2\_int32xm4* (*int32xm4\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *int16xm4\_t vnsravi\_int16xm4\_int32xm8* (*int32xm8\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *int32xm1\_t vnsravi\_int32xm1\_int64xm2* (*int64xm2\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *int32xm2\_t vnsravi\_int32xm2\_int64xm4* (*int64xm4\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *int32xm4\_t vnsravi\_int32xm4\_int64xm8* (*int64xm8\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *int8xm1\_t vnsravi\_int8xm1\_int16xm2* (*int16xm2\_t a*, const unsigned char *b*, unsigned int *gvl*)
- *int8xm2\_t vnsravi\_int8xm2\_int16xm4* (*int16xm4\_t a*, const unsigned char *b*, unsigned int *gvl*)
- *int8xm4\_t vnsravi\_int8xm4\_int16xm8* (*int16xm8\_t a*, const unsigned char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = narrow_int (sra (a[element], b))
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vnsravi\_mask\_int16xm1\_int32xm2* (*int16xm1\_t merge*, *int32xm2\_t a*, const unsigned short *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vnsravi\_mask\_int16xm2\_int32xm4* (*int16xm2\_t merge*, *int32xm4\_t a*, const unsigned short *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vnsravi\_mask\_int16xm4\_int32xm8* (*int16xm4\_t merge*, *int32xm8\_t a*, const unsigned short *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *int32xm1\_t vnsravi\_mask\_int32xm1\_int64xm2* (*int32xm1\_t merge*, *int64xm2\_t a*, const unsigned int *b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vnsravi\_mask\_int32xm2\_int64xm4* (*int32xm2\_t merge*, *int64xm4\_t a*, const unsigned int *b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vnsravi\_mask\_int32xm4\_int64xm8* (*int32xm4\_t merge*, *int64xm8\_t a*, const unsigned int *b*, *e64xm8\_t mask*, unsigned int *gvl*)
- *int8xm1\_t vnsravi\_mask\_int8xm1\_int16xm2* (*int8xm1\_t merge*, *int16xm2\_t a*, const unsigned char *b*, *e16xm2\_t mask*, unsigned int *gvl*)



- `int8xm2_t vnsravi_mask_int8xm2_int16xm4` (`int8xm2_t merge`, `int16xm4_t a`, const unsigned char `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `int8xm4_t vnsravi_mask_int8xm4_int16xm8` (`int8xm4_t merge`, `int16xm8_t a`, const unsigned char `b`, `e16xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = narrow_int (sra (a[element], b))
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.53 Narrowing elementwise vector-vector arithmetic shift right

Instruction: ['vnsra.vv']

Prototypes:

- `int16xm1_t vnsravv_int16xm1_int32xm2_uint16xm1` (`int32xm2_t a`, `uint16xm1_t b`, unsigned int `gvl`)
- `int16xm2_t vnsravv_int16xm2_int32xm4_uint16xm2` (`int32xm4_t a`, `uint16xm2_t b`, unsigned int `gvl`)
- `int16xm4_t vnsravv_int16xm4_int32xm8_uint16xm4` (`int32xm8_t a`, `uint16xm4_t b`, unsigned int `gvl`)
- `int32xm1_t vnsravv_int32xm1_int64xm2_uint32xm1` (`int64xm2_t a`, `uint32xm1_t b`, unsigned int `gvl`)
- `int32xm2_t vnsravv_int32xm2_int64xm4_uint32xm2` (`int64xm4_t a`, `uint32xm2_t b`, unsigned int `gvl`)
- `int32xm4_t vnsravv_int32xm4_int64xm8_uint32xm4` (`int64xm8_t a`, `uint32xm4_t b`, unsigned int `gvl`)
- `int8xm1_t vnsravv_int8xm1_int16xm2_uint8xm1` (`int16xm2_t a`, `uint8xm1_t b`, unsigned int `gvl`)
- `int8xm2_t vnsravv_int8xm2_int16xm4_uint8xm2` (`int16xm4_t a`, `uint8xm2_t b`, unsigned int `gvl`)
- `int8xm4_t vnsravv_int8xm4_int16xm8_uint8xm4` (`int16xm8_t a`, `uint8xm4_t b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = narrow_int (sra (a[element], b[element]))
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `int16xm1_t vnsravv_mask_int16xm1_int32xm2_uint16xm1` (`int16xm1_t merge`, `int32xm2_t a`, `uint16xm1_t b`, `e32xm2_t mask`, unsigned int `gvl`)

- `int16xm2_t vnsravv_mask_int16xm2_int32xm4_uint16xm2` (`int16xm2_t` *merge*,  
`int32xm4_t` *a*, `uint16xm2_t` *b*,  
`e32xm4_t` *mask*, unsigned  
int *gvl*)
- `int16xm4_t vnsravv_mask_int16xm4_int32xm8_uint16xm4` (`int16xm4_t` *merge*,  
`int32xm8_t` *a*, `uint16xm4_t` *b*,  
`e32xm8_t` *mask*, unsigned  
int *gvl*)
- `int32xm1_t vnsravv_mask_int32xm1_int64xm2_uint32xm1` (`int32xm1_t` *merge*,  
`int64xm2_t` *a*, `uint32xm1_t` *b*,  
`e64xm2_t` *mask*, unsigned  
int *gvl*)
- `int32xm2_t vnsravv_mask_int32xm2_int64xm4_uint32xm2` (`int32xm2_t` *merge*,  
`int64xm4_t` *a*, `uint32xm2_t` *b*,  
`e64xm4_t` *mask*, unsigned  
int *gvl*)
- `int32xm4_t vnsravv_mask_int32xm4_int64xm8_uint32xm4` (`int32xm4_t` *merge*,  
`int64xm8_t` *a*, `uint32xm4_t` *b*,  
`e64xm8_t` *mask*, unsigned  
int *gvl*)
- `int8xm1_t vnsravv_mask_int8xm1_int16xm2_uint8xm1` (`int8xm1_t` *merge*, `int16xm2_t` *a*,  
`uint8xm1_t` *b*, `e16xm2_t` *mask*, un-  
signed int *gvl*)
- `int8xm2_t vnsravv_mask_int8xm2_int16xm4_uint8xm2` (`int8xm2_t` *merge*, `int16xm4_t` *a*,  
`uint8xm2_t` *b*, `e16xm4_t` *mask*, un-  
signed int *gvl*)
- `int8xm4_t vnsravv_mask_int8xm4_int16xm8_uint8xm4` (`int8xm4_t` *merge*, `int16xm8_t` *a*,  
`uint8xm4_t` *b*, `e16xm8_t` *mask*, un-  
signed int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = narrow_int (sra (a[element], b[element]))
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.54 Narrowing elementwise vector-scalar arithmetic shift right****Instruction:** ['vnsra.vx']**Prototypes:**

- `int16xm1_t vnsravx_int16xm1_int32xm2` (`int32xm2_t` *a*, unsigned short *b*, unsigned int *gvl*)
- `int16xm2_t vnsravx_int16xm2_int32xm4` (`int32xm4_t` *a*, unsigned short *b*, unsigned int *gvl*)
- `int16xm4_t vnsravx_int16xm4_int32xm8` (`int32xm8_t` *a*, unsigned short *b*, unsigned int *gvl*)
- `int32xm1_t vnsravx_int32xm1_int64xm2` (`int64xm2_t` *a*, unsigned int *b*, unsigned int *gvl*)
- `int32xm2_t vnsravx_int32xm2_int64xm4` (`int64xm4_t` *a*, unsigned int *b*, unsigned int *gvl*)

- *int32xm4\_t vnsravx\_int32xm4\_int64xm8* (*int64xm8\_t a*, unsigned int *b*, unsigned int *gvl*)
- *int8xm1\_t vnsravx\_int8xm1\_int16xm2* (*int16xm2\_t a*, unsigned char *b*, unsigned int *gvl*)
- *int8xm2\_t vnsravx\_int8xm2\_int16xm4* (*int16xm4\_t a*, unsigned char *b*, unsigned int *gvl*)
- *int8xm4\_t vnsravx\_int8xm4\_int16xm8* (*int16xm8\_t a*, unsigned char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = narrow_int (sra (a[element], b))
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vnsravx\_mask\_int16xm1\_int32xm2* (*int16xm1\_t merge*, *int32xm2\_t a*, unsigned short *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vnsravx\_mask\_int16xm2\_int32xm4* (*int16xm2\_t merge*, *int32xm4\_t a*, unsigned short *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vnsravx\_mask\_int16xm4\_int32xm8* (*int16xm4\_t merge*, *int32xm8\_t a*, unsigned short *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *int32xm1\_t vnsravx\_mask\_int32xm1\_int64xm2* (*int32xm1\_t merge*, *int64xm2\_t a*, unsigned int *b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vnsravx\_mask\_int32xm2\_int64xm4* (*int32xm2\_t merge*, *int64xm4\_t a*, unsigned int *b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vnsravx\_mask\_int32xm4\_int64xm8* (*int32xm4\_t merge*, *int64xm8\_t a*, unsigned int *b*, *e64xm8\_t mask*, unsigned int *gvl*)
- *int8xm1\_t vnsravx\_mask\_int8xm1\_int16xm2* (*int8xm1\_t merge*, *int16xm2\_t a*, unsigned char *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int8xm2\_t vnsravx\_mask\_int8xm2\_int16xm4* (*int8xm2\_t merge*, *int16xm4\_t a*, unsigned char *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int8xm4\_t vnsravx\_mask\_int8xm4\_int16xm8* (*int8xm4\_t merge*, *int16xm8\_t a*, unsigned char *b*, *e16xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = narrow_int (sra (a[element], b))
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.55 Narrowing elementwise vector-immediate logic shift right

**Instruction:** ['vnsrl.vi']**Prototypes:**

- *uint16xm1\_t vnsrlvi\_uint16xm1\_uint32xm2* (*uint32xm2\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *uint16xm2\_t vnsrlvi\_uint16xm2\_uint32xm4* (*uint32xm4\_t a*, const unsigned short *b*, unsigned int *gvl*)

- `uint16xm4_t vnsrlvi_uint16xm4_uint32xm8` (`uint32xm8_t a`, const unsigned short `b`, unsigned int `gvl`)
- `uint32xm1_t vnsrlvi_uint32xm1_uint64xm2` (`uint64xm2_t a`, const unsigned int `b`, unsigned int `gvl`)
- `uint32xm2_t vnsrlvi_uint32xm2_uint64xm4` (`uint64xm4_t a`, const unsigned int `b`, unsigned int `gvl`)
- `uint32xm4_t vnsrlvi_uint32xm4_uint64xm8` (`uint64xm8_t a`, const unsigned int `b`, unsigned int `gvl`)
- `uint8xm1_t vnsrlvi_uint8xm1_uint16xm2` (`uint16xm2_t a`, const unsigned char `b`, unsigned int `gvl`)
- `uint8xm2_t vnsrlvi_uint8xm2_uint16xm4` (`uint16xm4_t a`, const unsigned char `b`, unsigned int `gvl`)
- `uint8xm4_t vnsrlvi_uint8xm4_uint16xm8` (`uint16xm8_t a`, const unsigned char `b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = narrow_int (srl (a[element], b))
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vnsrlvi_mask_uint16xm1_uint32xm2` (`uint16xm1_t merge`, `uint32xm2_t a`, const unsigned short `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint16xm2_t vnsrlvi_mask_uint16xm2_uint32xm4` (`uint16xm2_t merge`, `uint32xm4_t a`, const unsigned short `b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint16xm4_t vnsrlvi_mask_uint16xm4_uint32xm8` (`uint16xm4_t merge`, `uint32xm8_t a`, const unsigned short `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vnsrlvi_mask_uint32xm1_uint64xm2` (`uint32xm1_t merge`, `uint64xm2_t a`, const unsigned int `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint32xm2_t vnsrlvi_mask_uint32xm2_uint64xm4` (`uint32xm2_t merge`, `uint64xm4_t a`, const unsigned int `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint32xm4_t vnsrlvi_mask_uint32xm4_uint64xm8` (`uint32xm4_t merge`, `uint64xm8_t a`, const unsigned int `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vnsrlvi_mask_uint8xm1_uint16xm2` (`uint8xm1_t merge`, `uint16xm2_t a`, const unsigned char `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint8xm2_t vnsrlvi_mask_uint8xm2_uint16xm4` (`uint8xm2_t merge`, `uint16xm4_t a`, const unsigned char `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint8xm4_t vnsrlvi_mask_uint8xm4_uint16xm8` (`uint8xm4_t merge`, `uint16xm8_t a`, const unsigned char `b`, `e16xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = narrow_int (srl (a[element], b))
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.56 Narrowing elementwise vector-vector logic shift right

**Instruction:** ['vnsrl.vv']

**Prototypes:**

- *uint16xm1\_t vnsrlvv\_uint16xm1\_uint32xm2* (*uint32xm2\_t a, uint16xm1\_t b*, unsigned int *gvl*)
- *uint16xm2\_t vnsrlvv\_uint16xm2\_uint32xm4* (*uint32xm4\_t a, uint16xm2\_t b*, unsigned int *gvl*)
- *uint16xm4\_t vnsrlvv\_uint16xm4\_uint32xm8* (*uint32xm8\_t a, uint16xm4\_t b*, unsigned int *gvl*)
- *uint32xm1\_t vnsrlvv\_uint32xm1\_uint64xm2* (*uint64xm2\_t a, uint32xm1\_t b*, unsigned int *gvl*)
- *uint32xm2\_t vnsrlvv\_uint32xm2\_uint64xm4* (*uint64xm4\_t a, uint32xm2\_t b*, unsigned int *gvl*)
- *uint32xm4\_t vnsrlvv\_uint32xm4\_uint64xm8* (*uint64xm8\_t a, uint32xm4\_t b*, unsigned int *gvl*)
- *uint8xm1\_t vnsrlvv\_uint8xm1\_uint16xm2* (*uint16xm2\_t a, uint8xm1\_t b*, unsigned int *gvl*)
- *uint8xm2\_t vnsrlvv\_uint8xm2\_uint16xm4* (*uint16xm4\_t a, uint8xm2\_t b*, unsigned int *gvl*)
- *uint8xm4\_t vnsrlvv\_uint8xm4\_uint16xm8* (*uint16xm8\_t a, uint8xm4\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = narrow_int (srl (a[element], b[element]))
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *uint16xm1\_t vnsrlvv\_mask\_uint16xm1\_uint32xm2* (*uint16xm1\_t merge, uint32xm2\_t a, uint16xm1\_t b, e32xm2\_t mask*, unsigned int *gvl*)
- *uint16xm2\_t vnsrlvv\_mask\_uint16xm2\_uint32xm4* (*uint16xm2\_t merge, uint32xm4\_t a, uint16xm2\_t b, e32xm4\_t mask*, unsigned int *gvl*)
- *uint16xm4\_t vnsrlvv\_mask\_uint16xm4\_uint32xm8* (*uint16xm4\_t merge, uint32xm8\_t a, uint16xm4\_t b, e32xm8\_t mask*, unsigned int *gvl*)
- *uint32xm1\_t vnsrlvv\_mask\_uint32xm1\_uint64xm2* (*uint32xm1\_t merge, uint64xm2\_t a, uint32xm1\_t b, e64xm2\_t mask*, unsigned int *gvl*)
- *uint32xm2\_t vnsrlvv\_mask\_uint32xm2\_uint64xm4* (*uint32xm2\_t merge, uint64xm4\_t a, uint32xm2\_t b, e64xm4\_t mask*, unsigned int *gvl*)
- *uint32xm4\_t vnsrlvv\_mask\_uint32xm4\_uint64xm8* (*uint32xm4\_t merge, uint64xm8\_t a, uint32xm4\_t b, e64xm8\_t mask*, unsigned int *gvl*)

- `uint8xm1_t vnsrlvv_mask_uint8xm1_uint16xm2` (`uint8xm1_t merge`, `uint16xm2_t a`, `uint8xm1_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint8xm2_t vnsrlvv_mask_uint8xm2_uint16xm4` (`uint8xm2_t merge`, `uint16xm4_t a`, `uint8xm2_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint8xm4_t vnsrlvv_mask_uint8xm4_uint16xm8` (`uint8xm4_t merge`, `uint16xm8_t a`, `uint8xm4_t b`, `e16xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = narrow_int (srl (a[element], b[element]))
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.57 Narrowing elementwise vector-scalar logic shift right****Instruction:** [`'vnsrl.vx'`]**Prototypes:**

- `uint16xm1_t vnsrlvx_uint16xm1_uint32xm2` (`uint32xm2_t a`, unsigned short `b`, unsigned int `gvl`)
- `uint16xm2_t vnsrlvx_uint16xm2_uint32xm4` (`uint32xm4_t a`, unsigned short `b`, unsigned int `gvl`)
- `uint16xm4_t vnsrlvx_uint16xm4_uint32xm8` (`uint32xm8_t a`, unsigned short `b`, unsigned int `gvl`)
- `uint32xm1_t vnsrlvx_uint32xm1_uint64xm2` (`uint64xm2_t a`, unsigned int `b`, unsigned int `gvl`)
- `uint32xm2_t vnsrlvx_uint32xm2_uint64xm4` (`uint64xm4_t a`, unsigned int `b`, unsigned int `gvl`)
- `uint32xm4_t vnsrlvx_uint32xm4_uint64xm8` (`uint64xm8_t a`, unsigned int `b`, unsigned int `gvl`)
- `uint8xm1_t vnsrlvx_uint8xm1_uint16xm2` (`uint16xm2_t a`, unsigned char `b`, unsigned int `gvl`)
- `uint8xm2_t vnsrlvx_uint8xm2_uint16xm4` (`uint16xm4_t a`, unsigned char `b`, unsigned int `gvl`)
- `uint8xm4_t vnsrlvx_uint8xm4_uint16xm8` (`uint16xm8_t a`, unsigned char `b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = narrow_int (srl (a[element], b))
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vnsrlvx_mask_uint16xm1_uint32xm2` (`uint16xm1_t merge`, `uint32xm2_t a`, unsigned short `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint16xm2_t vnsrlvx_mask_uint16xm2_uint32xm4` (`uint16xm2_t merge`, `uint32xm4_t a`, unsigned short `b`, `e32xm4_t mask`, unsigned int `gvl`)



- `uint16xm4_t vnsrlvx_mask_uint16xm4_uint32xm8` (`uint16xm4_t merge`, `uint32xm8_t a`, unsigned short `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vnsrlvx_mask_uint32xm1_uint64xm2` (`uint32xm1_t merge`, `uint64xm2_t a`, unsigned int `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint32xm2_t vnsrlvx_mask_uint32xm2_uint64xm4` (`uint32xm2_t merge`, `uint64xm4_t a`, unsigned int `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint32xm4_t vnsrlvx_mask_uint32xm4_uint64xm8` (`uint32xm4_t merge`, `uint64xm8_t a`, unsigned int `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vnsrlvx_mask_uint8xm1_uint16xm2` (`uint8xm1_t merge`, `uint16xm2_t a`, unsigned char `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint8xm2_t vnsrlvx_mask_uint8xm2_uint16xm4` (`uint8xm2_t merge`, `uint16xm4_t a`, unsigned char `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint8xm4_t vnsrlvx_mask_uint8xm4_uint16xm8` (`uint8xm4_t merge`, `uint16xm8_t a`, unsigned char `b`, `e16xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = narrow_int (srl (a[element], b))
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.58 Integer vector bitwise-and reduction

Instruction: ['vredand.vs']

Prototypes:

- `int16xm1_t vredandvs_int16xm1` (`int16xm1_t a`, `int16xm1_t b`, unsigned int `gvl`)
- `int16xm2_t vredandvs_int16xm2` (`int16xm2_t a`, `int16xm2_t b`, unsigned int `gvl`)
- `int16xm4_t vredandvs_int16xm4` (`int16xm4_t a`, `int16xm4_t b`, unsigned int `gvl`)
- `int16xm8_t vredandvs_int16xm8` (`int16xm8_t a`, `int16xm8_t b`, unsigned int `gvl`)
- `int32xm1_t vredandvs_int32xm1` (`int32xm1_t a`, `int32xm1_t b`, unsigned int `gvl`)
- `int32xm2_t vredandvs_int32xm2` (`int32xm2_t a`, `int32xm2_t b`, unsigned int `gvl`)
- `int32xm4_t vredandvs_int32xm4` (`int32xm4_t a`, `int32xm4_t b`, unsigned int `gvl`)
- `int32xm8_t vredandvs_int32xm8` (`int32xm8_t a`, `int32xm8_t b`, unsigned int `gvl`)
- `int64xm1_t vredandvs_int64xm1` (`int64xm1_t a`, `int64xm1_t b`, unsigned int `gvl`)
- `int64xm2_t vredandvs_int64xm2` (`int64xm2_t a`, `int64xm2_t b`, unsigned int `gvl`)
- `int64xm4_t vredandvs_int64xm4` (`int64xm4_t a`, `int64xm4_t b`, unsigned int `gvl`)
- `int64xm8_t vredandvs_int64xm8` (`int64xm8_t a`, `int64xm8_t b`, unsigned int `gvl`)
- `int8xm1_t vredandvs_int8xm1` (`int8xm1_t a`, `int8xm1_t b`, unsigned int `gvl`)

- `int8xm2_t vredandvs_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vredandvs_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vredandvs_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`
- `uint16xm1_t vredandvs_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vredandvs_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vredandvs_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vredandvs_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vredandvs_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vredandvs_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vredandvs_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vredandvs_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vredandvs_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vredandvs_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vredandvs_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vredandvs_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vredandvs_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vredandvs_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vredandvs_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vredandvs_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        current_red = bitwise_and (current_red, a[element])
    result[0] = current_red
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vredandvs_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vredandvs_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vredandvs_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vredandvs_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vredandvs_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vredandvs_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vredandvs_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`

- `int32xm8_t vredandvs_mask_int32xm8` (`int32xm8_t merge`, `int32xm8_t a`, `int32xm8_t b`, `e32xm8_t mask`, unsigned int gvl)
- `int64xm1_t vredandvs_mask_int64xm1` (`int64xm1_t merge`, `int64xm1_t a`, `int64xm1_t b`, `e64xm1_t mask`, unsigned int gvl)
- `int64xm2_t vredandvs_mask_int64xm2` (`int64xm2_t merge`, `int64xm2_t a`, `int64xm2_t b`, `e64xm2_t mask`, unsigned int gvl)
- `int64xm4_t vredandvs_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, `int64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)
- `int64xm8_t vredandvs_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, `int64xm8_t b`, `e64xm8_t mask`, unsigned int gvl)
- `int8xm1_t vredandvs_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, `int8xm1_t b`, `e8xm1_t mask`, unsigned int gvl)
- `int8xm2_t vredandvs_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, `int8xm2_t b`, `e8xm2_t mask`, unsigned int gvl)
- `int8xm4_t vredandvs_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, `int8xm4_t b`, `e8xm4_t mask`, unsigned int gvl)
- `int8xm8_t vredandvs_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, `int8xm8_t b`, `e8xm8_t mask`, unsigned int gvl)
- `uint16xm1_t vredandvs_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, `uint16xm1_t b`, `e16xm1_t mask`, unsigned int gvl)
- `uint16xm2_t vredandvs_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, `uint16xm2_t b`, `e16xm2_t mask`, unsigned int gvl)
- `uint16xm4_t vredandvs_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, `uint16xm4_t b`, `e16xm4_t mask`, unsigned int gvl)
- `uint16xm8_t vredandvs_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, `uint16xm8_t b`, `e16xm8_t mask`, unsigned int gvl)
- `uint32xm1_t vredandvs_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, `uint32xm1_t b`, `e32xm1_t mask`, unsigned int gvl)
- `uint32xm2_t vredandvs_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, `uint32xm2_t b`, `e32xm2_t mask`, unsigned int gvl)
- `uint32xm4_t vredandvs_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, `uint32xm4_t b`, `e32xm4_t mask`, unsigned int gvl)
- `uint32xm8_t vredandvs_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, `uint32xm8_t b`, `e32xm8_t mask`, unsigned int gvl)
- `uint64xm1_t vredandvs_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, `uint64xm1_t b`, `e64xm1_t mask`, unsigned int gvl)
- `uint64xm2_t vredandvs_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, `uint64xm2_t b`, `e64xm2_t mask`, unsigned int gvl)
- `uint64xm4_t vredandvs_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, `uint64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)
- `uint64xm8_t vredandvs_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, `uint64xm8_t b`, `e64xm8_t mask`, unsigned int gvl)
- `uint8xm1_t vredandvs_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int gvl)
- `uint8xm2_t vredandvs_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int gvl)

- `uint8xm4_t vredandvs_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vredandvs_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        if mask[element] then
            current_red = bitwise_and (current_red, a[element])
        else
            result[element] = merge[element]
    result[0] = current_red
    result[1 : VLMAX] = 0
```

## 2.7.59 Integer vector signed maximum reduction

**Instruction:** [`'vredmax.vs'`]

**Prototypes:**

- `int16xm1_t vredmaxvs_int16xm1 (int16xm1_t a, int16xm1_t b, unsigned int gvl)`
- `int16xm2_t vredmaxvs_int16xm2 (int16xm2_t a, int16xm2_t b, unsigned int gvl)`
- `int16xm4_t vredmaxvs_int16xm4 (int16xm4_t a, int16xm4_t b, unsigned int gvl)`
- `int16xm8_t vredmaxvs_int16xm8 (int16xm8_t a, int16xm8_t b, unsigned int gvl)`
- `int32xm1_t vredmaxvs_int32xm1 (int32xm1_t a, int32xm1_t b, unsigned int gvl)`
- `int32xm2_t vredmaxvs_int32xm2 (int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int32xm4_t vredmaxvs_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vredmaxvs_int32xm8 (int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vredmaxvs_int64xm1 (int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vredmaxvs_int64xm2 (int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vredmaxvs_int64xm4 (int64xm4_t a, int64xm4_t b, unsigned int gvl)`
- `int64xm8_t vredmaxvs_int64xm8 (int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vredmaxvs_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vredmaxvs_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vredmaxvs_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vredmaxvs_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        current_red = max (current_red, a[element])
    result[0] = current_red
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vredmaxvs_mask_int16xm1(int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vredmaxvs_mask_int16xm2(int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vredmaxvs_mask_int16xm4(int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vredmaxvs_mask_int16xm8(int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vredmaxvs_mask_int32xm1(int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vredmaxvs_mask_int32xm2(int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vredmaxvs_mask_int32xm4(int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vredmaxvs_mask_int32xm8(int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vredmaxvs_mask_int64xm1(int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vredmaxvs_mask_int64xm2(int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vredmaxvs_mask_int64xm4(int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vredmaxvs_mask_int64xm8(int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vredmaxvs_mask_int8xm1(int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vredmaxvs_mask_int8xm2(int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vredmaxvs_mask_int8xm4(int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vredmaxvs_mask_int8xm8(int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        if mask[element] then
            current_red = max (current_red, a[element])
        else
            result[element] = merge[element]
    result[0] = current_red
    result[1 : VLMAX] = 0
```

**2.7.60 Integer vector unsigned maximum reduction****Instruction:** ['vredmaxu.vs']

**Prototypes:**

- `uint16xm1_t vredmaxuvs_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vredmaxuvs_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vredmaxuvs_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vredmaxuvs_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vredmaxuvs_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vredmaxuvs_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vredmaxuvs_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vredmaxuvs_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vredmaxuvs_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vredmaxuvs_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vredmaxuvs_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vredmaxuvs_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vredmaxuvs_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vredmaxuvs_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vredmaxuvs_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vredmaxuvs_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        current_red = max (current_red, a[element])
    result[0] = current_red
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vredmaxuvs_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vredmaxuvs_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vredmaxuvs_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vredmaxuvs_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vredmaxuvs_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vredmaxuvs_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vredmaxuvs_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vredmaxuvs_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`



- `uint64xm1_t vredmaxuvs_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, `uint64xm1_t b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vredmaxuvs_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, `uint64xm2_t b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vredmaxuvs_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, `uint64xm4_t b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vredmaxuvs_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, `uint64xm8_t b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vredmaxuvs_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vredmaxuvs_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vredmaxuvs_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vredmaxuvs_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, `uint8xm8_t b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        if mask[element] then
            current_red = max (current_red, a[element])
        else
            result[element] = merge[element]
    result[0] = current_red
    result[1 : VLMAX] = 0
```

## 2.7.61 Integer vector signed minimum reduction

**Instruction:** [`'vredmin.vs'`]**Prototypes:**

- `int16xm1_t vredminvs_int16xm1` (`int16xm1_t a`, `int16xm1_t b`, unsigned int `gvl`)
- `int16xm2_t vredminvs_int16xm2` (`int16xm2_t a`, `int16xm2_t b`, unsigned int `gvl`)
- `int16xm4_t vredminvs_int16xm4` (`int16xm4_t a`, `int16xm4_t b`, unsigned int `gvl`)
- `int16xm8_t vredminvs_int16xm8` (`int16xm8_t a`, `int16xm8_t b`, unsigned int `gvl`)
- `int32xm1_t vredminvs_int32xm1` (`int32xm1_t a`, `int32xm1_t b`, unsigned int `gvl`)
- `int32xm2_t vredminvs_int32xm2` (`int32xm2_t a`, `int32xm2_t b`, unsigned int `gvl`)
- `int32xm4_t vredminvs_int32xm4` (`int32xm4_t a`, `int32xm4_t b`, unsigned int `gvl`)
- `int32xm8_t vredminvs_int32xm8` (`int32xm8_t a`, `int32xm8_t b`, unsigned int `gvl`)
- `int64xm1_t vredminvs_int64xm1` (`int64xm1_t a`, `int64xm1_t b`, unsigned int `gvl`)
- `int64xm2_t vredminvs_int64xm2` (`int64xm2_t a`, `int64xm2_t b`, unsigned int `gvl`)
- `int64xm4_t vredminvs_int64xm4` (`int64xm4_t a`, `int64xm4_t b`, unsigned int `gvl`)
- `int64xm8_t vredminvs_int64xm8` (`int64xm8_t a`, `int64xm8_t b`, unsigned int `gvl`)

- `int8xm1_t vredminvs_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vredminvs_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vredminvs_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vredminvs_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        current_red = min (current_red, a[element])
    result[0] = current_red
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vredminvs_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vredminvs_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vredminvs_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vredminvs_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vredminvs_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vredminvs_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vredminvs_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vredminvs_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vredminvs_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vredminvs_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vredminvs_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vredminvs_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vredminvs_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vredminvs_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vredminvs_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vredminvs_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```

>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        if mask[element] then
            current_red = min (current_red, a[element])
        else
            result[element] = merge[element]
    result[0] = current_red
    result[1 : VLMAX] = 0

```

## 2.7.62 Integer vector unsigned minimum reduction

**Instruction:** ['vredminu.vs']

**Prototypes:**

- *uint16xm1\_t* **vredminuvs\_uint16xm1** (*uint16xm1\_t* a, *uint16xm1\_t* b, unsigned int gvl)
- *uint16xm2\_t* **vredminuvs\_uint16xm2** (*uint16xm2\_t* a, *uint16xm2\_t* b, unsigned int gvl)
- *uint16xm4\_t* **vredminuvs\_uint16xm4** (*uint16xm4\_t* a, *uint16xm4\_t* b, unsigned int gvl)
- *uint16xm8\_t* **vredminuvs\_uint16xm8** (*uint16xm8\_t* a, *uint16xm8\_t* b, unsigned int gvl)
- *uint32xm1\_t* **vredminuvs\_uint32xm1** (*uint32xm1\_t* a, *uint32xm1\_t* b, unsigned int gvl)
- *uint32xm2\_t* **vredminuvs\_uint32xm2** (*uint32xm2\_t* a, *uint32xm2\_t* b, unsigned int gvl)
- *uint32xm4\_t* **vredminuvs\_uint32xm4** (*uint32xm4\_t* a, *uint32xm4\_t* b, unsigned int gvl)
- *uint32xm8\_t* **vredminuvs\_uint32xm8** (*uint32xm8\_t* a, *uint32xm8\_t* b, unsigned int gvl)
- *uint64xm1\_t* **vredminuvs\_uint64xm1** (*uint64xm1\_t* a, *uint64xm1\_t* b, unsigned int gvl)
- *uint64xm2\_t* **vredminuvs\_uint64xm2** (*uint64xm2\_t* a, *uint64xm2\_t* b, unsigned int gvl)
- *uint64xm4\_t* **vredminuvs\_uint64xm4** (*uint64xm4\_t* a, *uint64xm4\_t* b, unsigned int gvl)
- *uint64xm8\_t* **vredminuvs\_uint64xm8** (*uint64xm8\_t* a, *uint64xm8\_t* b, unsigned int gvl)
- *uint8xm1\_t* **vredminuvs\_uint8xm1** (*uint8xm1\_t* a, *uint8xm1\_t* b, unsigned int gvl)
- *uint8xm2\_t* **vredminuvs\_uint8xm2** (*uint8xm2\_t* a, *uint8xm2\_t* b, unsigned int gvl)
- *uint8xm4\_t* **vredminuvs\_uint8xm4** (*uint8xm4\_t* a, *uint8xm4\_t* b, unsigned int gvl)
- *uint8xm8\_t* **vredminuvs\_uint8xm8** (*uint8xm8\_t* a, *uint8xm8\_t* b, unsigned int gvl)

**Operation:**

```

>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        current_red = min (current_red, a[element])
    result[0] = current_red
    result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- *uint16xm1\_t* **vredminuvs\_mask\_uint16xm1** (*uint16xm1\_t* merge, *uint16xm1\_t* a, *uint16xm1\_t* b, *e16xm1\_t* mask, unsigned int gvl)
- *uint16xm2\_t* **vredminuvs\_mask\_uint16xm2** (*uint16xm2\_t* merge, *uint16xm2\_t* a, *uint16xm2\_t* b, *e16xm2\_t* mask, unsigned int gvl)

- `uint16xm4_t vredminuvs_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, `uint16xm4_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vredminuvs_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, `uint16xm8_t b`, `e16xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vredminuvs_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, `uint32xm1_t b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vredminuvs_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, `uint32xm2_t b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vredminuvs_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, `uint32xm4_t b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint32xm8_t vredminuvs_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, `uint32xm8_t b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint64xm1_t vredminuvs_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, `uint64xm1_t b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vredminuvs_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, `uint64xm2_t b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vredminuvs_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, `uint64xm4_t b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vredminuvs_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, `uint64xm8_t b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vredminuvs_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vredminuvs_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vredminuvs_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vredminuvs_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, `uint8xm8_t b`, `e8xm8_t mask`, unsigned int `gvl`)

#### Masked operation:

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        if mask[element] then
            current_red = min (current_red, a[element])
        else
            result[element] = merge[element]
    result[0] = current_red
    result[1 : VLMAX] = 0
```

### 2.7.63 Integer vector bitwise-or reduction

**Instruction:** [`'vredor.vs'`]

**Prototypes:**

- `int16xm1_t vredorvs_int16xm1` (`int16xm1_t a`, `int16xm1_t b`, unsigned int `gvl`)
- `int16xm2_t vredorvs_int16xm2` (`int16xm2_t a`, `int16xm2_t b`, unsigned int `gvl`)
- `int16xm4_t vredorvs_int16xm4` (`int16xm4_t a`, `int16xm4_t b`, unsigned int `gvl`)

- `int16xm8_t vredorvs_int16xm8 (int16xm8_t a, int16xm8_t b, unsigned int gvl)`
- `int32xm1_t vredorvs_int32xm1 (int32xm1_t a, int32xm1_t b, unsigned int gvl)`
- `int32xm2_t vredorvs_int32xm2 (int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int32xm4_t vredorvs_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vredorvs_int32xm8 (int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vredorvs_int64xm1 (int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vredorvs_int64xm2 (int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vredorvs_int64xm4 (int64xm4_t a, int64xm4_t b, unsigned int gvl)`
- `int64xm8_t vredorvs_int64xm8 (int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vredorvs_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vredorvs_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vredorvs_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vredorvs_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`
- `uint16xm1_t vredorvs_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vredorvs_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vredorvs_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vredorvs_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vredorvs_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vredorvs_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vredorvs_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vredorvs_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vredorvs_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vredorvs_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vredorvs_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vredorvs_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vredorvs_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vredorvs_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vredorvs_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vredorvs_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        current_red = bitwise_or (current_red, a[element])
    result[0] = current_red
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vredorvs\_mask\_int16xm1* (*int16xm1\_t merge, int16xm1\_t a, int16xm1\_t b, e16xm1\_t mask*, unsigned int gvl)
- *int16xm2\_t vredorvs\_mask\_int16xm2* (*int16xm2\_t merge, int16xm2\_t a, int16xm2\_t b, e16xm2\_t mask*, unsigned int gvl)
- *int16xm4\_t vredorvs\_mask\_int16xm4* (*int16xm4\_t merge, int16xm4\_t a, int16xm4\_t b, e16xm4\_t mask*, unsigned int gvl)
- *int16xm8\_t vredorvs\_mask\_int16xm8* (*int16xm8\_t merge, int16xm8\_t a, int16xm8\_t b, e16xm8\_t mask*, unsigned int gvl)
- *int32xm1\_t vredorvs\_mask\_int32xm1* (*int32xm1\_t merge, int32xm1\_t a, int32xm1\_t b, e32xm1\_t mask*, unsigned int gvl)
- *int32xm2\_t vredorvs\_mask\_int32xm2* (*int32xm2\_t merge, int32xm2\_t a, int32xm2\_t b, e32xm2\_t mask*, unsigned int gvl)
- *int32xm4\_t vredorvs\_mask\_int32xm4* (*int32xm4\_t merge, int32xm4\_t a, int32xm4\_t b, e32xm4\_t mask*, unsigned int gvl)
- *int32xm8\_t vredorvs\_mask\_int32xm8* (*int32xm8\_t merge, int32xm8\_t a, int32xm8\_t b, e32xm8\_t mask*, unsigned int gvl)
- *int64xm1\_t vredorvs\_mask\_int64xm1* (*int64xm1\_t merge, int64xm1\_t a, int64xm1\_t b, e64xm1\_t mask*, unsigned int gvl)
- *int64xm2\_t vredorvs\_mask\_int64xm2* (*int64xm2\_t merge, int64xm2\_t a, int64xm2\_t b, e64xm2\_t mask*, unsigned int gvl)
- *int64xm4\_t vredorvs\_mask\_int64xm4* (*int64xm4\_t merge, int64xm4\_t a, int64xm4\_t b, e64xm4\_t mask*, unsigned int gvl)
- *int64xm8\_t vredorvs\_mask\_int64xm8* (*int64xm8\_t merge, int64xm8\_t a, int64xm8\_t b, e64xm8\_t mask*, unsigned int gvl)
- *int8xm1\_t vredorvs\_mask\_int8xm1* (*int8xm1\_t merge, int8xm1\_t a, int8xm1\_t b, e8xm1\_t mask*, unsigned int gvl)
- *int8xm2\_t vredorvs\_mask\_int8xm2* (*int8xm2\_t merge, int8xm2\_t a, int8xm2\_t b, e8xm2\_t mask*, unsigned int gvl)
- *int8xm4\_t vredorvs\_mask\_int8xm4* (*int8xm4\_t merge, int8xm4\_t a, int8xm4\_t b, e8xm4\_t mask*, unsigned int gvl)
- *int8xm8\_t vredorvs\_mask\_int8xm8* (*int8xm8\_t merge, int8xm8\_t a, int8xm8\_t b, e8xm8\_t mask*, unsigned int gvl)
- *uint16xm1\_t vredorvs\_mask\_uint16xm1* (*uint16xm1\_t merge, uint16xm1\_t a, uint16xm1\_t b, e16xm1\_t mask*, unsigned int gvl)
- *uint16xm2\_t vredorvs\_mask\_uint16xm2* (*uint16xm2\_t merge, uint16xm2\_t a, uint16xm2\_t b, e16xm2\_t mask*, unsigned int gvl)
- *uint16xm4\_t vredorvs\_mask\_uint16xm4* (*uint16xm4\_t merge, uint16xm4\_t a, uint16xm4\_t b, e16xm4\_t mask*, unsigned int gvl)
- *uint16xm8\_t vredorvs\_mask\_uint16xm8* (*uint16xm8\_t merge, uint16xm8\_t a, uint16xm8\_t b, e16xm8\_t mask*, unsigned int gvl)
- *uint32xm1\_t vredorvs\_mask\_uint32xm1* (*uint32xm1\_t merge, uint32xm1\_t a, uint32xm1\_t b, e32xm1\_t mask*, unsigned int gvl)
- *uint32xm2\_t vredorvs\_mask\_uint32xm2* (*uint32xm2\_t merge, uint32xm2\_t a, uint32xm2\_t b, e32xm2\_t mask*, unsigned int gvl)
- *uint32xm4\_t vredorvs\_mask\_uint32xm4* (*uint32xm4\_t merge, uint32xm4\_t a, uint32xm4\_t b, e32xm4\_t mask*, unsigned int gvl)



- `uint32xm8_t vredorvs_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vredorvs_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vredorvs_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vredorvs_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vredorvs_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vredorvs_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vredorvs_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vredorvs_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vredorvs_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        if mask[element] then
            current_red = bitwise_or (current_red, a[element])
        else
            result[element] = merge[element]
    result[0] = current_red
    result[1 : VLMAX] = 0
```

**2.7.64 Integer vector sum reduction****Instruction:** ['vredsum.vs']**Prototypes:**

- `int16xm1_t vredsumvs_int16xm1 (int16xm1_t a, int16xm1_t b, unsigned int gvl)`
- `int16xm2_t vredsumvs_int16xm2 (int16xm2_t a, int16xm2_t b, unsigned int gvl)`
- `int16xm4_t vredsumvs_int16xm4 (int16xm4_t a, int16xm4_t b, unsigned int gvl)`
- `int16xm8_t vredsumvs_int16xm8 (int16xm8_t a, int16xm8_t b, unsigned int gvl)`
- `int32xm1_t vredsumvs_int32xm1 (int32xm1_t a, int32xm1_t b, unsigned int gvl)`
- `int32xm2_t vredsumvs_int32xm2 (int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int32xm4_t vredsumvs_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vredsumvs_int32xm8 (int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vredsumvs_int64xm1 (int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vredsumvs_int64xm2 (int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vredsumvs_int64xm4 (int64xm4_t a, int64xm4_t b, unsigned int gvl)`

- `int64xm8_t vredsumvs_int64xm8 (int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vredsumvs_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vredsumvs_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vredsumvs_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vredsumvs_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`
- `uint16xm1_t vredsumvs_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vredsumvs_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vredsumvs_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vredsumvs_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vredsumvs_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vredsumvs_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vredsumvs_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vredsumvs_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vredsumvs_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vredsumvs_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vredsumvs_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vredsumvs_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vredsumvs_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vredsumvs_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vredsumvs_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vredsumvs_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        current_red = sum (current_red, a[element])
    result[0] = current_red
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vredsumvs_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vredsumvs_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vredsumvs_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vredsumvs_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vredsumvs_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`

- `int32xm2_t vredsumvs_mask_int32xm2(int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vredsumvs_mask_int32xm4(int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vredsumvs_mask_int32xm8(int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vredsumvs_mask_int64xm1(int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vredsumvs_mask_int64xm2(int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vredsumvs_mask_int64xm4(int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vredsumvs_mask_int64xm8(int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vredsumvs_mask_int8xm1(int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vredsumvs_mask_int8xm2(int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vredsumvs_mask_int8xm4(int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vredsumvs_mask_int8xm8(int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vredsumvs_mask_uint16xm1(uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vredsumvs_mask_uint16xm2(uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vredsumvs_mask_uint16xm4(uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vredsumvs_mask_uint16xm8(uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vredsumvs_mask_uint32xm1(uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vredsumvs_mask_uint32xm2(uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vredsumvs_mask_uint32xm4(uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vredsumvs_mask_uint32xm8(uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vredsumvs_mask_uint64xm1(uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vredsumvs_mask_uint64xm2(uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vredsumvs_mask_uint64xm4(uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vredsumvs_mask_uint64xm8(uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`

- `uint8xm1_t vredsumvs_mask_uint8xm1(uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vredsumvs_mask_uint8xm2(uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vredsumvs_mask_uint8xm4(uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vredsumvs_mask_uint8xm8(uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        if mask[element] then
            current_red = sum (current_red, a[element])
        else
            result[element] = merge[element]
    result[0] = current_red
    result[1 : VLMAX] = 0
```

## 2.7.65 Integer vector bitwise-xor reduction

Instruction: ['vredxor.vs']

Prototypes:

- `int16xm1_t vredxorvs_int16xm1(int16xm1_t a, int16xm1_t b, unsigned int gvl)`
- `int16xm2_t vredxorvs_int16xm2(int16xm2_t a, int16xm2_t b, unsigned int gvl)`
- `int16xm4_t vredxorvs_int16xm4(int16xm4_t a, int16xm4_t b, unsigned int gvl)`
- `int16xm8_t vredxorvs_int16xm8(int16xm8_t a, int16xm8_t b, unsigned int gvl)`
- `int32xm1_t vredxorvs_int32xm1(int32xm1_t a, int32xm1_t b, unsigned int gvl)`
- `int32xm2_t vredxorvs_int32xm2(int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int32xm4_t vredxorvs_int32xm4(int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vredxorvs_int32xm8(int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vredxorvs_int64xm1(int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vredxorvs_int64xm2(int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vredxorvs_int64xm4(int64xm4_t a, int64xm4_t b, unsigned int gvl)`
- `int64xm8_t vredxorvs_int64xm8(int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vredxorvs_int8xm1(int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vredxorvs_int8xm2(int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vredxorvs_int8xm4(int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vredxorvs_int8xm8(int8xm8_t a, int8xm8_t b, unsigned int gvl)`
- `uint16xm1_t vredxorvs_uint16xm1(uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vredxorvs_uint16xm2(uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`

- `uint16xm4_t vredxorvs_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vredxorvs_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vredxorvs_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vredxorvs_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vredxorvs_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vredxorvs_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vredxorvs_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vredxorvs_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vredxorvs_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vredxorvs_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vredxorvs_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vredxorvs_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vredxorvs_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vredxorvs_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        current_red = bitwise_xor (current_red, a[element])
    result[0] = current_red
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vredxorvs_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vredxorvs_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vredxorvs_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vredxorvs_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vredxorvs_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vredxorvs_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vredxorvs_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vredxorvs_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vredxorvs_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vredxorvs_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`

- `int64xm4_t vredxorvs_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, `int64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)
- `int64xm8_t vredxorvs_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, `int64xm8_t b`, `e64xm8_t mask`, unsigned int gvl)
- `int8xm1_t vredxorvs_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, `int8xm1_t b`, `e8xm1_t mask`, unsigned int gvl)
- `int8xm2_t vredxorvs_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, `int8xm2_t b`, `e8xm2_t mask`, unsigned int gvl)
- `int8xm4_t vredxorvs_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, `int8xm4_t b`, `e8xm4_t mask`, unsigned int gvl)
- `int8xm8_t vredxorvs_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, `int8xm8_t b`, `e8xm8_t mask`, unsigned int gvl)
- `uint16xm1_t vredxorvs_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, `uint16xm1_t b`, `e16xm1_t mask`, unsigned int gvl)
- `uint16xm2_t vredxorvs_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, `uint16xm2_t b`, `e16xm2_t mask`, unsigned int gvl)
- `uint16xm4_t vredxorvs_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, `uint16xm4_t b`, `e16xm4_t mask`, unsigned int gvl)
- `uint16xm8_t vredxorvs_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, `uint16xm8_t b`, `e16xm8_t mask`, unsigned int gvl)
- `uint32xm1_t vredxorvs_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, `uint32xm1_t b`, `e32xm1_t mask`, unsigned int gvl)
- `uint32xm2_t vredxorvs_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, `uint32xm2_t b`, `e32xm2_t mask`, unsigned int gvl)
- `uint32xm4_t vredxorvs_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, `uint32xm4_t b`, `e32xm4_t mask`, unsigned int gvl)
- `uint32xm8_t vredxorvs_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, `uint32xm8_t b`, `e32xm8_t mask`, unsigned int gvl)
- `uint64xm1_t vredxorvs_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, `uint64xm1_t b`, `e64xm1_t mask`, unsigned int gvl)
- `uint64xm2_t vredxorvs_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, `uint64xm2_t b`, `e64xm2_t mask`, unsigned int gvl)
- `uint64xm4_t vredxorvs_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, `uint64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)
- `uint64xm8_t vredxorvs_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, `uint64xm8_t b`, `e64xm8_t mask`, unsigned int gvl)
- `uint8xm1_t vredxorvs_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int gvl)
- `uint8xm2_t vredxorvs_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int gvl)
- `uint8xm4_t vredxorvs_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int gvl)
- `uint8xm8_t vredxorvs_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, `uint8xm8_t b`, `e8xm8_t mask`, unsigned int gvl)

Masked operation:



```

>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        if mask[element] then
            current_red = bitwise_xor (current_red, a[element])
        else
            result[element] = merge[element]
    result[0] = current_red
    result[1 : VLMAX] = 0

```

## 2.7.66 Elementwise signed vector-vector division remainder

**Instruction:** ['vrem.vv']

**Prototypes:**

- *int16xm1\_t* **vremvv\_int16xm1** (*int16xm1\_t* a, *int16xm1\_t* b, unsigned int gvl)
- *int16xm2\_t* **vremvv\_int16xm2** (*int16xm2\_t* a, *int16xm2\_t* b, unsigned int gvl)
- *int16xm4\_t* **vremvv\_int16xm4** (*int16xm4\_t* a, *int16xm4\_t* b, unsigned int gvl)
- *int16xm8\_t* **vremvv\_int16xm8** (*int16xm8\_t* a, *int16xm8\_t* b, unsigned int gvl)
- *int32xm1\_t* **vremvv\_int32xm1** (*int32xm1\_t* a, *int32xm1\_t* b, unsigned int gvl)
- *int32xm2\_t* **vremvv\_int32xm2** (*int32xm2\_t* a, *int32xm2\_t* b, unsigned int gvl)
- *int32xm4\_t* **vremvv\_int32xm4** (*int32xm4\_t* a, *int32xm4\_t* b, unsigned int gvl)
- *int32xm8\_t* **vremvv\_int32xm8** (*int32xm8\_t* a, *int32xm8\_t* b, unsigned int gvl)
- *int64xm1\_t* **vremvv\_int64xm1** (*int64xm1\_t* a, *int64xm1\_t* b, unsigned int gvl)
- *int64xm2\_t* **vremvv\_int64xm2** (*int64xm2\_t* a, *int64xm2\_t* b, unsigned int gvl)
- *int64xm4\_t* **vremvv\_int64xm4** (*int64xm4\_t* a, *int64xm4\_t* b, unsigned int gvl)
- *int64xm8\_t* **vremvv\_int64xm8** (*int64xm8\_t* a, *int64xm8\_t* b, unsigned int gvl)
- *int8xm1\_t* **vremvv\_int8xm1** (*int8xm1\_t* a, *int8xm1\_t* b, unsigned int gvl)
- *int8xm2\_t* **vremvv\_int8xm2** (*int8xm2\_t* a, *int8xm2\_t* b, unsigned int gvl)
- *int8xm4\_t* **vremvv\_int8xm4** (*int8xm4\_t* a, *int8xm4\_t* b, unsigned int gvl)
- *int8xm8\_t* **vremvv\_int8xm8** (*int8xm8\_t* a, *int8xm8\_t* b, unsigned int gvl)

**Operation:**

```

>>> for element = 0 to gvl - 1
    result[element] = rem (a[element], b[element])
result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- *int16xm1\_t* **vremvv\_mask\_int16xm1** (*int16xm1\_t* merge, *int16xm1\_t* a, *int16xm1\_t* b, *int16xm1\_t* mask, unsigned int gvl)
- *int16xm2\_t* **vremvv\_mask\_int16xm2** (*int16xm2\_t* merge, *int16xm2\_t* a, *int16xm2\_t* b, *int16xm2\_t* mask, unsigned int gvl)
- *int16xm4\_t* **vremvv\_mask\_int16xm4** (*int16xm4\_t* merge, *int16xm4\_t* a, *int16xm4\_t* b, *int16xm4\_t* mask, unsigned int gvl)

- `int16xm8_t vremvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vremvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vremvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vremvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vremvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vremvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vremvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vremvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vremvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vremvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vremvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vremvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vremvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = rem (a[element], b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.67 Elementwise signed vector-scalar division remainder

**Instruction:** ['vrem.vx']

**Prototypes:**

- `int16xm1_t vremvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int16xm2_t vremvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vremvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vremvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vremvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vremvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`

- `int32xm4_t vremvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vremvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vremvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vremvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vremvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vremvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vremvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vremvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vremvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vremvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = rem (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vremvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vremvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vremvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vremvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vremvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vremvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vremvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vremvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vremvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vremvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vremvx_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vremvx_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vremvx_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vremvx_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl)`

- `int8xm4_t vremvx_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, signed char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int8xm8_t vremvx_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, signed char `b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = rem (a[element], b)
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.7.68 Elementwise unsigned vector-vector division remainder

Instruction: ['vremu.vv']

Prototypes:

- `uint16xm1_t vremuvv_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, unsigned int `gvl`)
- `uint16xm2_t vremuvv_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, unsigned int `gvl`)
- `uint16xm4_t vremuvv_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, unsigned int `gvl`)
- `uint16xm8_t vremuvv_uint16xm8` (`uint16xm8_t a`, `uint16xm8_t b`, unsigned int `gvl`)
- `uint32xm1_t vremuvv_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, unsigned int `gvl`)
- `uint32xm2_t vremuvv_uint32xm2` (`uint32xm2_t a`, `uint32xm2_t b`, unsigned int `gvl`)
- `uint32xm4_t vremuvv_uint32xm4` (`uint32xm4_t a`, `uint32xm4_t b`, unsigned int `gvl`)
- `uint32xm8_t vremuvv_uint32xm8` (`uint32xm8_t a`, `uint32xm8_t b`, unsigned int `gvl`)
- `uint64xm1_t vremuvv_uint64xm1` (`uint64xm1_t a`, `uint64xm1_t b`, unsigned int `gvl`)
- `uint64xm2_t vremuvv_uint64xm2` (`uint64xm2_t a`, `uint64xm2_t b`, unsigned int `gvl`)
- `uint64xm4_t vremuvv_uint64xm4` (`uint64xm4_t a`, `uint64xm4_t b`, unsigned int `gvl`)
- `uint64xm8_t vremuvv_uint64xm8` (`uint64xm8_t a`, `uint64xm8_t b`, unsigned int `gvl`)
- `uint8xm1_t vremuvv_uint8xm1` (`uint8xm1_t a`, `uint8xm1_t b`, unsigned int `gvl`)
- `uint8xm2_t vremuvv_uint8xm2` (`uint8xm2_t a`, `uint8xm2_t b`, unsigned int `gvl`)
- `uint8xm4_t vremuvv_uint8xm4` (`uint8xm4_t a`, `uint8xm4_t b`, unsigned int `gvl`)
- `uint8xm8_t vremuvv_uint8xm8` (`uint8xm8_t a`, `uint8xm8_t b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = rem (a[element], b[element])
    result[gvl : VLMAX] = 0
```

Masked prototypes:

- `uint16xm1_t vremuvv_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, `uint16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)

- `uint16xm2_t vremuvv_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vremuvv_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vremuvv_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vremuvv_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vremuvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vremuvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vremuvv_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vremuvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vremuvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vremuvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vremuvv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vremuvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vremuvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vremuvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vremuvv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = rem (a[element], b[element])
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.7.69 Elementwise unsigned vector-scalar division remainder****Instruction:** [`'vremu.vx'`]**Prototypes:**

- `uint16xm1_t vremuvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vremuvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vremuvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`

- `uint16xm8_t vremuvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vremuvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vremuvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vremuvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vremuvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vremuvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vremuvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vremuvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vremuvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vremuvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vremuvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vremuvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vremuvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = rem (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vremuvx_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vremuvx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vremuvx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vremuvx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vremuvx_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vremuvx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vremuvx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vremuvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vremuvx_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vremuvx_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vremuvx_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vremuvx_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`



- `uint8xm1_t vremuvx_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vremuvx_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vremuvx_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vremuvx_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = rem (a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.70 Elementwise vector-immediate integer reverse subtraction

Instruction: ['vrsbvi.vi']

Prototypes:

- `int16xm1_t vrsbvi_int16xm1 (int16xm1_t a, const short b, unsigned int gvl)`
- `int16xm2_t vrsbvi_int16xm2 (int16xm2_t a, const short b, unsigned int gvl)`
- `int16xm4_t vrsbvi_int16xm4 (int16xm4_t a, const short b, unsigned int gvl)`
- `int16xm8_t vrsbvi_int16xm8 (int16xm8_t a, const short b, unsigned int gvl)`
- `int32xm1_t vrsbvi_int32xm1 (int32xm1_t a, const int b, unsigned int gvl)`
- `int32xm2_t vrsbvi_int32xm2 (int32xm2_t a, const int b, unsigned int gvl)`
- `int32xm4_t vrsbvi_int32xm4 (int32xm4_t a, const int b, unsigned int gvl)`
- `int32xm8_t vrsbvi_int32xm8 (int32xm8_t a, const int b, unsigned int gvl)`
- `int64xm1_t vrsbvi_int64xm1 (int64xm1_t a, const long b, unsigned int gvl)`
- `int64xm2_t vrsbvi_int64xm2 (int64xm2_t a, const long b, unsigned int gvl)`
- `int64xm4_t vrsbvi_int64xm4 (int64xm4_t a, const long b, unsigned int gvl)`
- `int64xm8_t vrsbvi_int64xm8 (int64xm8_t a, const long b, unsigned int gvl)`
- `int8xm1_t vrsbvi_int8xm1 (int8xm1_t a, const signed char b, unsigned int gvl)`
- `int8xm2_t vrsbvi_int8xm2 (int8xm2_t a, const signed char b, unsigned int gvl)`
- `int8xm4_t vrsbvi_int8xm4 (int8xm4_t a, const signed char b, unsigned int gvl)`
- `int8xm8_t vrsbvi_int8xm8 (int8xm8_t a, const signed char b, unsigned int gvl)`
- `uint16xm1_t vrsbvi_uint16xm1 (uint16xm1_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm2_t vrsbvi_uint16xm2 (uint16xm2_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm4_t vrsbvi_uint16xm4 (uint16xm4_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm8_t vrsbvi_uint16xm8 (uint16xm8_t a, const unsigned short b, unsigned int gvl)`

- `uint32xm1_t vrsubvi_uint32xm1 (uint32xm1_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm2_t vrsubvi_uint32xm2 (uint32xm2_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm4_t vrsubvi_uint32xm4 (uint32xm4_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm8_t vrsubvi_uint32xm8 (uint32xm8_t a, const unsigned int b, unsigned int gvl)`
- `uint64xm1_t vrsubvi_uint64xm1 (uint64xm1_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm2_t vrsubvi_uint64xm2 (uint64xm2_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm4_t vrsubvi_uint64xm4 (uint64xm4_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm8_t vrsubvi_uint64xm8 (uint64xm8_t a, const unsigned long b, unsigned int gvl)`
- `uint8xm1_t vrsubvi_uint8xm1 (uint8xm1_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm2_t vrsubvi_uint8xm2 (uint8xm2_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm4_t vrsubvi_uint8xm4 (uint8xm4_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm8_t vrsubvi_uint8xm8 (uint8xm8_t a, const unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = rsub (a[element], b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vrsubvi_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, const short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vrsubvi_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, const short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vrsubvi_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, const short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vrsubvi_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, const short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vrsubvi_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, const int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vrsubvi_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, const int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vrsubvi_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, const int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vrsubvi_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, const int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vrsubvi_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, const long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vrsubvi_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, const long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vrsubvi_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, const long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vrsubvi_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, const long b, e64xm8_t mask, unsigned int gvl)`

- `int8xm1_t vrsubvi_mask_int8xm1(int8xm1_t merge, int8xm1_t a, const signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vrsubvi_mask_int8xm2(int8xm2_t merge, int8xm2_t a, const signed char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vrsubvi_mask_int8xm4(int8xm4_t merge, int8xm4_t a, const signed char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vrsubvi_mask_int8xm8(int8xm8_t merge, int8xm8_t a, const signed char b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vrsubvi_mask_uint16xm1(uint16xm1_t merge, uint16xm1_t a, const unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vrsubvi_mask_uint16xm2(uint16xm2_t merge, uint16xm2_t a, const unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vrsubvi_mask_uint16xm4(uint16xm4_t merge, uint16xm4_t a, const unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vrsubvi_mask_uint16xm8(uint16xm8_t merge, uint16xm8_t a, const unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vrsubvi_mask_uint32xm1(uint32xm1_t merge, uint32xm1_t a, const unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vrsubvi_mask_uint32xm2(uint32xm2_t merge, uint32xm2_t a, const unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vrsubvi_mask_uint32xm4(uint32xm4_t merge, uint32xm4_t a, const unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vrsubvi_mask_uint32xm8(uint32xm8_t merge, uint32xm8_t a, const unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vrsubvi_mask_uint64xm1(uint64xm1_t merge, uint64xm1_t a, const unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vrsubvi_mask_uint64xm2(uint64xm2_t merge, uint64xm2_t a, const unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vrsubvi_mask_uint64xm4(uint64xm4_t merge, uint64xm4_t a, const unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vrsubvi_mask_uint64xm8(uint64xm8_t merge, uint64xm8_t a, const unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vrsubvi_mask_uint8xm1(uint8xm1_t merge, uint8xm1_t a, const unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vrsubvi_mask_uint8xm2(uint8xm2_t merge, uint8xm2_t a, const unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vrsubvi_mask_uint8xm4(uint8xm4_t merge, uint8xm4_t a, const unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vrsubvi_mask_uint8xm8(uint8xm8_t merge, uint8xm8_t a, const unsigned char b, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = rsub (a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.71 Elementwise vector-scalar integer reverse subtraction

**Instruction:** ['vrsub.vx']

**Prototypes:**

- *int16xm1\_t vrsubvx\_int16xm1* (*int16xm1\_t a*, short *b*, unsigned int *gvl*)
- *int16xm2\_t vrsubvx\_int16xm2* (*int16xm2\_t a*, short *b*, unsigned int *gvl*)
- *int16xm4\_t vrsubvx\_int16xm4* (*int16xm4\_t a*, short *b*, unsigned int *gvl*)
- *int16xm8\_t vrsubvx\_int16xm8* (*int16xm8\_t a*, short *b*, unsigned int *gvl*)
- *int32xm1\_t vrsubvx\_int32xm1* (*int32xm1\_t a*, int *b*, unsigned int *gvl*)
- *int32xm2\_t vrsubvx\_int32xm2* (*int32xm2\_t a*, int *b*, unsigned int *gvl*)
- *int32xm4\_t vrsubvx\_int32xm4* (*int32xm4\_t a*, int *b*, unsigned int *gvl*)
- *int32xm8\_t vrsubvx\_int32xm8* (*int32xm8\_t a*, int *b*, unsigned int *gvl*)
- *int64xm1\_t vrsubvx\_int64xm1* (*int64xm1\_t a*, long *b*, unsigned int *gvl*)
- *int64xm2\_t vrsubvx\_int64xm2* (*int64xm2\_t a*, long *b*, unsigned int *gvl*)
- *int64xm4\_t vrsubvx\_int64xm4* (*int64xm4\_t a*, long *b*, unsigned int *gvl*)
- *int64xm8\_t vrsubvx\_int64xm8* (*int64xm8\_t a*, long *b*, unsigned int *gvl*)
- *int8xm1\_t vrsubvx\_int8xm1* (*int8xm1\_t a*, signed char *b*, unsigned int *gvl*)
- *int8xm2\_t vrsubvx\_int8xm2* (*int8xm2\_t a*, signed char *b*, unsigned int *gvl*)
- *int8xm4\_t vrsubvx\_int8xm4* (*int8xm4\_t a*, signed char *b*, unsigned int *gvl*)
- *int8xm8\_t vrsubvx\_int8xm8* (*int8xm8\_t a*, signed char *b*, unsigned int *gvl*)
- *uint16xm1\_t vrsubvx\_uint16xm1* (*uint16xm1\_t a*, unsigned short *b*, unsigned int *gvl*)
- *uint16xm2\_t vrsubvx\_uint16xm2* (*uint16xm2\_t a*, unsigned short *b*, unsigned int *gvl*)
- *uint16xm4\_t vrsubvx\_uint16xm4* (*uint16xm4\_t a*, unsigned short *b*, unsigned int *gvl*)
- *uint16xm8\_t vrsubvx\_uint16xm8* (*uint16xm8\_t a*, unsigned short *b*, unsigned int *gvl*)
- *uint32xm1\_t vrsubvx\_uint32xm1* (*uint32xm1\_t a*, unsigned int *b*, unsigned int *gvl*)
- *uint32xm2\_t vrsubvx\_uint32xm2* (*uint32xm2\_t a*, unsigned int *b*, unsigned int *gvl*)
- *uint32xm4\_t vrsubvx\_uint32xm4* (*uint32xm4\_t a*, unsigned int *b*, unsigned int *gvl*)
- *uint32xm8\_t vrsubvx\_uint32xm8* (*uint32xm8\_t a*, unsigned int *b*, unsigned int *gvl*)
- *uint64xm1\_t vrsubvx\_uint64xm1* (*uint64xm1\_t a*, unsigned long *b*, unsigned int *gvl*)
- *uint64xm2\_t vrsubvx\_uint64xm2* (*uint64xm2\_t a*, unsigned long *b*, unsigned int *gvl*)
- *uint64xm4\_t vrsubvx\_uint64xm4* (*uint64xm4\_t a*, unsigned long *b*, unsigned int *gvl*)
- *uint64xm8\_t vrsubvx\_uint64xm8* (*uint64xm8\_t a*, unsigned long *b*, unsigned int *gvl*)
- *uint8xm1\_t vrsubvx\_uint8xm1* (*uint8xm1\_t a*, unsigned char *b*, unsigned int *gvl*)
- *uint8xm2\_t vrsubvx\_uint8xm2* (*uint8xm2\_t a*, unsigned char *b*, unsigned int *gvl*)
- *uint8xm4\_t vrsubvx\_uint8xm4* (*uint8xm4\_t a*, unsigned char *b*, unsigned int *gvl*)
- *uint8xm8\_t vrsubvx\_uint8xm8* (*uint8xm8\_t a*, unsigned char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = rsub (a[element], b)
      result[gvl : VLMAX] = 0
```

### Masked prototypes:

- *int16xm1\_t vrsubvx\_mask\_int16xm1* (*int16xm1\_t merge, int16xm1\_t a, short b, e16xm1\_t mask, unsigned int gvl*)
- *int16xm2\_t vrsubvx\_mask\_int16xm2* (*int16xm2\_t merge, int16xm2\_t a, short b, e16xm2\_t mask, unsigned int gvl*)
- *int16xm4\_t vrsubvx\_mask\_int16xm4* (*int16xm4\_t merge, int16xm4\_t a, short b, e16xm4\_t mask, unsigned int gvl*)
- *int16xm8\_t vrsubvx\_mask\_int16xm8* (*int16xm8\_t merge, int16xm8\_t a, short b, e16xm8\_t mask, unsigned int gvl*)
- *int32xm1\_t vrsubvx\_mask\_int32xm1* (*int32xm1\_t merge, int32xm1\_t a, int b, e32xm1\_t mask, unsigned int gvl*)
- *int32xm2\_t vrsubvx\_mask\_int32xm2* (*int32xm2\_t merge, int32xm2\_t a, int b, e32xm2\_t mask, unsigned int gvl*)
- *int32xm4\_t vrsubvx\_mask\_int32xm4* (*int32xm4\_t merge, int32xm4\_t a, int b, e32xm4\_t mask, unsigned int gvl*)
- *int32xm8\_t vrsubvx\_mask\_int32xm8* (*int32xm8\_t merge, int32xm8\_t a, int b, e32xm8\_t mask, unsigned int gvl*)
- *int64xm1\_t vrsubvx\_mask\_int64xm1* (*int64xm1\_t merge, int64xm1\_t a, long b, e64xm1\_t mask, unsigned int gvl*)
- *int64xm2\_t vrsubvx\_mask\_int64xm2* (*int64xm2\_t merge, int64xm2\_t a, long b, e64xm2\_t mask, unsigned int gvl*)
- *int64xm4\_t vrsubvx\_mask\_int64xm4* (*int64xm4\_t merge, int64xm4\_t a, long b, e64xm4\_t mask, unsigned int gvl*)
- *int64xm8\_t vrsubvx\_mask\_int64xm8* (*int64xm8\_t merge, int64xm8\_t a, long b, e64xm8\_t mask, unsigned int gvl*)
- *int8xm1\_t vrsubvx\_mask\_int8xm1* (*int8xm1\_t merge, int8xm1\_t a, signed char b, e8xm1\_t mask, unsigned int gvl*)
- *int8xm2\_t vrsubvx\_mask\_int8xm2* (*int8xm2\_t merge, int8xm2\_t a, signed char b, e8xm2\_t mask, unsigned int gvl*)
- *int8xm4\_t vrsubvx\_mask\_int8xm4* (*int8xm4\_t merge, int8xm4\_t a, signed char b, e8xm4\_t mask, unsigned int gvl*)
- *int8xm8\_t vrsubvx\_mask\_int8xm8* (*int8xm8\_t merge, int8xm8\_t a, signed char b, e8xm8\_t mask, unsigned int gvl*)
- *uint16xm1\_t vrsubvx\_mask\_uint16xm1* (*uint16xm1\_t merge, uint16xm1\_t a, unsigned short b, e16xm1\_t mask, unsigned int gvl*)
- *uint16xm2\_t vrsubvx\_mask\_uint16xm2* (*uint16xm2\_t merge, uint16xm2\_t a, unsigned short b, e16xm2\_t mask, unsigned int gvl*)
- *uint16xm4\_t vrsubvx\_mask\_uint16xm4* (*uint16xm4\_t merge, uint16xm4\_t a, unsigned short b, e16xm4\_t mask, unsigned int gvl*)
- *uint16xm8\_t vrsubvx\_mask\_uint16xm8* (*uint16xm8\_t merge, uint16xm8\_t a, unsigned short b, e16xm8\_t mask, unsigned int gvl*)
- *uint32xm1\_t vrsubvx\_mask\_uint32xm1* (*uint32xm1\_t merge, uint32xm1\_t a, unsigned int b, e32xm1\_t mask, unsigned int gvl*)



- `uint32xm2_t vrsubvx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vrsubvx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vrsubvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vrsubvx_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vrsubvx_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vrsubvx_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vrsubvx_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vrsubvx_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vrsubvx_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vrsubvx_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vrsubvx_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = rsub (a[element], b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

### 2.7.72 Elementwise signed vector-immediate addition with saturation

**Instruction:** [`vsadd.vi`']

#### Prototypes:

- `int16xm1_t vsaddvi_int16xm1 (int16xm1_t a, const short b, unsigned int gvl)`
- `int16xm2_t vsaddvi_int16xm2 (int16xm2_t a, const short b, unsigned int gvl)`
- `int16xm4_t vsaddvi_int16xm4 (int16xm4_t a, const short b, unsigned int gvl)`
- `int16xm8_t vsaddvi_int16xm8 (int16xm8_t a, const short b, unsigned int gvl)`
- `int32xm1_t vsaddvi_int32xm1 (int32xm1_t a, const int b, unsigned int gvl)`
- `int32xm2_t vsaddvi_int32xm2 (int32xm2_t a, const int b, unsigned int gvl)`
- `int32xm4_t vsaddvi_int32xm4 (int32xm4_t a, const int b, unsigned int gvl)`
- `int32xm8_t vsaddvi_int32xm8 (int32xm8_t a, const int b, unsigned int gvl)`
- `int64xm1_t vsaddvi_int64xm1 (int64xm1_t a, const long b, unsigned int gvl)`



- `int64xm2_t vsaddvi_int64xm2 (int64xm2_t a, const long b, unsigned int gvl)`
- `int64xm4_t vsaddvi_int64xm4 (int64xm4_t a, const long b, unsigned int gvl)`
- `int64xm8_t vsaddvi_int64xm8 (int64xm8_t a, const long b, unsigned int gvl)`
- `int8xm1_t vsaddvi_int8xm1 (int8xm1_t a, const signed char b, unsigned int gvl)`
- `int8xm2_t vsaddvi_int8xm2 (int8xm2_t a, const signed char b, unsigned int gvl)`
- `int8xm4_t vsaddvi_int8xm4 (int8xm4_t a, const signed char b, unsigned int gvl)`
- `int8xm8_t vsaddvi_int8xm8 (int8xm8_t a, const signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = saturat(a[element] + b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vsaddvi_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, const short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vsaddvi_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, const short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vsaddvi_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, const short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vsaddvi_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, const short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vsaddvi_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, const int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsaddvi_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, const int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsaddvi_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, const int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vsaddvi_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, const int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vsaddvi_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, const long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vsaddvi_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, const long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vsaddvi_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, const long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vsaddvi_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, const long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vsaddvi_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, const signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vsaddvi_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, const signed char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vsaddvi_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, const signed char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vsaddvi_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, const signed char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = saturat(a[element] + b)
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.73 Elementwise signed vector-vector addition with saturation****Instruction:** ['vsadd.vv']**Prototypes:**

- *int16xm1\_t vsaddvv\_int16xm1* (*int16xm1\_t a*, *int16xm1\_t b*, unsigned int *gvl*)
- *int16xm2\_t vsaddvv\_int16xm2* (*int16xm2\_t a*, *int16xm2\_t b*, unsigned int *gvl*)
- *int16xm4\_t vsaddvv\_int16xm4* (*int16xm4\_t a*, *int16xm4\_t b*, unsigned int *gvl*)
- *int16xm8\_t vsaddvv\_int16xm8* (*int16xm8\_t a*, *int16xm8\_t b*, unsigned int *gvl*)
- *int32xm1\_t vsaddvv\_int32xm1* (*int32xm1\_t a*, *int32xm1\_t b*, unsigned int *gvl*)
- *int32xm2\_t vsaddvv\_int32xm2* (*int32xm2\_t a*, *int32xm2\_t b*, unsigned int *gvl*)
- *int32xm4\_t vsaddvv\_int32xm4* (*int32xm4\_t a*, *int32xm4\_t b*, unsigned int *gvl*)
- *int32xm8\_t vsaddvv\_int32xm8* (*int32xm8\_t a*, *int32xm8\_t b*, unsigned int *gvl*)
- *int64xm1\_t vsaddvv\_int64xm1* (*int64xm1\_t a*, *int64xm1\_t b*, unsigned int *gvl*)
- *int64xm2\_t vsaddvv\_int64xm2* (*int64xm2\_t a*, *int64xm2\_t b*, unsigned int *gvl*)
- *int64xm4\_t vsaddvv\_int64xm4* (*int64xm4\_t a*, *int64xm4\_t b*, unsigned int *gvl*)
- *int64xm8\_t vsaddvv\_int64xm8* (*int64xm8\_t a*, *int64xm8\_t b*, unsigned int *gvl*)
- *int8xm1\_t vsaddvv\_int8xm1* (*int8xm1\_t a*, *int8xm1\_t b*, unsigned int *gvl*)
- *int8xm2\_t vsaddvv\_int8xm2* (*int8xm2\_t a*, *int8xm2\_t b*, unsigned int *gvl*)
- *int8xm4\_t vsaddvv\_int8xm4* (*int8xm4\_t a*, *int8xm4\_t b*, unsigned int *gvl*)
- *int8xm8\_t vsaddvv\_int8xm8* (*int8xm8\_t a*, *int8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = saturat(a[element] + b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vsaddvv\_mask\_int16xm1* (*int16xm1\_t merge*, *int16xm1\_t a*, *int16xm1\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vsaddvv\_mask\_int16xm2* (*int16xm2\_t merge*, *int16xm2\_t a*, *int16xm2\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vsaddvv\_mask\_int16xm4* (*int16xm4\_t merge*, *int16xm4\_t a*, *int16xm4\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)

- `int16xm8_t vsaddvv_mask_int16xm8` (`int16xm8_t merge`, `int16xm8_t a`, `int16xm8_t b`, `e16xm8_t mask`, unsigned int gvl)
- `int32xm1_t vsaddvv_mask_int32xm1` (`int32xm1_t merge`, `int32xm1_t a`, `int32xm1_t b`, `e32xm1_t mask`, unsigned int gvl)
- `int32xm2_t vsaddvv_mask_int32xm2` (`int32xm2_t merge`, `int32xm2_t a`, `int32xm2_t b`, `e32xm2_t mask`, unsigned int gvl)
- `int32xm4_t vsaddvv_mask_int32xm4` (`int32xm4_t merge`, `int32xm4_t a`, `int32xm4_t b`, `e32xm4_t mask`, unsigned int gvl)
- `int32xm8_t vsaddvv_mask_int32xm8` (`int32xm8_t merge`, `int32xm8_t a`, `int32xm8_t b`, `e32xm8_t mask`, unsigned int gvl)
- `int64xm1_t vsaddvv_mask_int64xm1` (`int64xm1_t merge`, `int64xm1_t a`, `int64xm1_t b`, `e64xm1_t mask`, unsigned int gvl)
- `int64xm2_t vsaddvv_mask_int64xm2` (`int64xm2_t merge`, `int64xm2_t a`, `int64xm2_t b`, `e64xm2_t mask`, unsigned int gvl)
- `int64xm4_t vsaddvv_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, `int64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)
- `int64xm8_t vsaddvv_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, `int64xm8_t b`, `e64xm8_t mask`, unsigned int gvl)
- `int8xm1_t vsaddvv_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, `int8xm1_t b`, `e8xm1_t mask`, unsigned int gvl)
- `int8xm2_t vsaddvv_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, `int8xm2_t b`, `e8xm2_t mask`, unsigned int gvl)
- `int8xm4_t vsaddvv_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, `int8xm4_t b`, `e8xm4_t mask`, unsigned int gvl)
- `int8xm8_t vsaddvv_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, `int8xm8_t b`, `e8xm8_t mask`, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = saturat(a[element] + b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.74 Elementwise signed vector-scalar addition with saturation

Instruction: ['vsadd.vx']

Prototypes:

- `int16xm1_t vsaddvx_int16xm1` (`int16xm1_t a`, short `b`, unsigned int gvl)
- `int16xm2_t vsaddvx_int16xm2` (`int16xm2_t a`, short `b`, unsigned int gvl)
- `int16xm4_t vsaddvx_int16xm4` (`int16xm4_t a`, short `b`, unsigned int gvl)
- `int16xm8_t vsaddvx_int16xm8` (`int16xm8_t a`, short `b`, unsigned int gvl)
- `int32xm1_t vsaddvx_int32xm1` (`int32xm1_t a`, int `b`, unsigned int gvl)
- `int32xm2_t vsaddvx_int32xm2` (`int32xm2_t a`, int `b`, unsigned int gvl)

- `int32xm4_t vsaddvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vsaddvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vsaddvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vsaddvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vsaddvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vsaddvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vsaddvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vsaddvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vsaddvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vsaddvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = saturat(a[element] + b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vsaddvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vsaddvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vsaddvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vsaddvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vsaddvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsaddvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsaddvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vsaddvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vsaddvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vsaddvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vsaddvx_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vsaddvx_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vsaddvx_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vsaddvx_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl)`

- `int8xm4_t vsaddvx_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, signed char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int8xm8_t vsaddvx_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, signed char `b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = saturat(a[element] + b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.75 Elementwise unsigned vector-immediate addition with saturation

Instruction: [`vsaddu.vi`]

Prototypes:

- `uint16xm1_t vsadduvi_uint16xm1` (`uint16xm1_t a`, const unsigned short `b`, unsigned int `gvl`)
- `uint16xm2_t vsadduvi_uint16xm2` (`uint16xm2_t a`, const unsigned short `b`, unsigned int `gvl`)
- `uint16xm4_t vsadduvi_uint16xm4` (`uint16xm4_t a`, const unsigned short `b`, unsigned int `gvl`)
- `uint16xm8_t vsadduvi_uint16xm8` (`uint16xm8_t a`, const unsigned short `b`, unsigned int `gvl`)
- `uint32xm1_t vsadduvi_uint32xm1` (`uint32xm1_t a`, const unsigned int `b`, unsigned int `gvl`)
- `uint32xm2_t vsadduvi_uint32xm2` (`uint32xm2_t a`, const unsigned int `b`, unsigned int `gvl`)
- `uint32xm4_t vsadduvi_uint32xm4` (`uint32xm4_t a`, const unsigned int `b`, unsigned int `gvl`)
- `uint32xm8_t vsadduvi_uint32xm8` (`uint32xm8_t a`, const unsigned int `b`, unsigned int `gvl`)
- `uint64xm1_t vsadduvi_uint64xm1` (`uint64xm1_t a`, const unsigned long `b`, unsigned int `gvl`)
- `uint64xm2_t vsadduvi_uint64xm2` (`uint64xm2_t a`, const unsigned long `b`, unsigned int `gvl`)
- `uint64xm4_t vsadduvi_uint64xm4` (`uint64xm4_t a`, const unsigned long `b`, unsigned int `gvl`)
- `uint64xm8_t vsadduvi_uint64xm8` (`uint64xm8_t a`, const unsigned long `b`, unsigned int `gvl`)
- `uint8xm1_t vsadduvi_uint8xm1` (`uint8xm1_t a`, const unsigned char `b`, unsigned int `gvl`)
- `uint8xm2_t vsadduvi_uint8xm2` (`uint8xm2_t a`, const unsigned char `b`, unsigned int `gvl`)
- `uint8xm4_t vsadduvi_uint8xm4` (`uint8xm4_t a`, const unsigned char `b`, unsigned int `gvl`)
- `uint8xm8_t vsadduvi_uint8xm8` (`uint8xm8_t a`, const unsigned char `b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = saturat(a[element] + b)
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `uint16xm1_t vsadduvi_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, const unsigned short `b`, `e16xm1_t mask`, unsigned int `gvl`)

- `uint16xm2_t vsadduvi_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, const unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vsadduvi_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, const unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vsadduvi_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, const unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vsadduvi_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, const unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vsadduvi_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, const unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vsadduvi_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, const unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vsadduvi_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, const unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vsadduvi_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, const unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vsadduvi_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, const unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vsadduvi_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, const unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vsadduvi_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, const unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vsadduvi_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, const unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vsadduvi_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, const unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vsadduvi_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, const unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vsadduvi_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, const unsigned char b, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = saturat(a[element] + b)
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.7.76 Elementwise unsigned vector-vector addition with saturation

**Instruction:** [`'vsaddu.vv'`]

**Prototypes:**

- `uint16xm1_t vsadduvv_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vsadduvv_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vsadduvv_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`



- `uint16xm8_t vsadduvv_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vsadduvv_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vsadduvv_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vsadduvv_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vsadduvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vsadduvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vsadduvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vsadduvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vsadduvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vsadduvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vsadduvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vsadduvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vsadduvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = saturat(a[element] + b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vsadduvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vsadduvv_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vsadduvv_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vsadduvv_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vsadduvv_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vsadduvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vsadduvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vsadduvv_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vsadduvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vsadduvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vsadduvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vsadduvv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`

- `uint8xm1_t vsadduvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vsadduvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vsadduvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vsadduvv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = saturat(a[element] + b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.77 Elementwise unsigned vector-scalar addition with saturation

Instruction: ['vsaddu.vx']

Prototypes:

- `uint16xm1_t vsadduvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vsadduvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vsadduvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vsadduvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vsadduvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vsadduvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vsadduvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vsadduvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vsadduvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vsadduvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vsadduvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vsadduvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vsadduvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vsadduvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vsadduvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vsadduvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = saturat(a[element] + b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vsadduvx_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, unsigned short `b`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint16xm2_t vsadduvx_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, unsigned short `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint16xm4_t vsadduvx_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, unsigned short `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vsadduvx_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, unsigned short `b`, `e16xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vsadduvx_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, unsigned int `b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vsadduvx_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, unsigned int `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vsadduvx_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, unsigned int `b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint32xm8_t vsadduvx_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, unsigned int `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint64xm1_t vsadduvx_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, unsigned long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vsadduvx_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, unsigned long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vsadduvx_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, unsigned long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vsadduvx_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, unsigned long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vsadduvx_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vsadduvx_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vsadduvx_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vsadduvx_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = saturat(a[element] + b)
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

**2.7.78 Elementwise vector-vector integer addition with borrow****Instruction:** [`'vsbc.vvm'`]**Masked prototypes:**

- `int16xm1_t vsbcbvm_mask_int16xm1(int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vsbcbvm_mask_int16xm2(int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vsbcbvm_mask_int16xm4(int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vsbcbvm_mask_int16xm8(int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vsbcbvm_mask_int32xm1(int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsbcbvm_mask_int32xm2(int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsbcbvm_mask_int32xm4(int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vsbcbvm_mask_int32xm8(int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vsbcbvm_mask_int64xm1(int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vsbcbvm_mask_int64xm2(int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vsbcbvm_mask_int64xm4(int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vsbcbvm_mask_int64xm8(int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vsbcbvm_mask_int8xm1(int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vsbcbvm_mask_int8xm2(int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vsbcbvm_mask_int8xm4(int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vsbcbvm_mask_int8xm8(int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vsbcbvm_mask_uint16xm1(uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vsbcbvm_mask_uint16xm2(uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vsbcbvm_mask_uint16xm4(uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vsbcbvm_mask_uint16xm8(uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vsbcbvm_mask_uint32xm1(uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vsbcbvm_mask_uint32xm2(uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vsbcbvm_mask_uint32xm4(uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vsbcbvm_mask_uint32xm8(uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vsbcbvm_mask_uint64xm1(uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`

- `uint64xm2_t vsbcvmm_mask_uint64xm2 (uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vsbcvmm_mask_uint64xm4 (uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vsbcvmm_mask_uint64xm8 (uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vsbcvmm_mask_uint8xm1 (uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vsbcvmm_mask_uint8xm2 (uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vsbcvmm_mask_uint8xm4 (uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vsbcvmm_mask_uint8xm8 (uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] - b[element] - mask[element]
result[gvl : VLMAX] = 0
```

## 2.7.79 Elementwise vector-scalar integer addition with borrow

Instruction: `['vsbcvxm']`

Masked prototypes:

- `int16xm1_t vsbcvxm_mask_int16xm1 (int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vsbcvxm_mask_int16xm2 (int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vsbcvxm_mask_int16xm4 (int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vsbcvxm_mask_int16xm8 (int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vsbcvxm_mask_int32xm1 (int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsbcvxm_mask_int32xm2 (int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsbcvxm_mask_int32xm4 (int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vsbcvxm_mask_int32xm8 (int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vsbcvxm_mask_int64xm1 (int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vsbcvxm_mask_int64xm2 (int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vsbcvxm_mask_int64xm4 (int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vsbcvxm_mask_int64xm8 (int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vsbcvxm_mask_int8xm1 (int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vsbcvxm_mask_int8xm2 (int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vsbcvxm_mask_int8xm4 (int8xm4_t a, signed char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vsbcvxm_mask_int8xm8 (int8xm8_t a, signed char b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vsbcvxm_mask_uint16xm1 (uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vsbcvxm_mask_uint16xm2 (uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`

- `uint16xm4_t vsbcvxm_mask_uint16xm4` (`uint16xm4_t a`, unsigned short `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vsbcvxm_mask_uint16xm8` (`uint16xm8_t a`, unsigned short `b`, `e16xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vsbcvxm_mask_uint32xm1` (`uint32xm1_t a`, unsigned int `b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vsbcvxm_mask_uint32xm2` (`uint32xm2_t a`, unsigned int `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vsbcvxm_mask_uint32xm4` (`uint32xm4_t a`, unsigned int `b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint32xm8_t vsbcvxm_mask_uint32xm8` (`uint32xm8_t a`, unsigned int `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint64xm1_t vsbcvxm_mask_uint64xm1` (`uint64xm1_t a`, unsigned long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vsbcvxm_mask_uint64xm2` (`uint64xm2_t a`, unsigned long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vsbcvxm_mask_uint64xm4` (`uint64xm4_t a`, unsigned long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vsbcvxm_mask_uint64xm8` (`uint64xm8_t a`, unsigned long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vsbcvxm_mask_uint8xm1` (`uint8xm1_t a`, unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vsbcvxm_mask_uint8xm2` (`uint8xm2_t a`, unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vsbcvxm_mask_uint8xm4` (`uint8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vsbcvxm_mask_uint8xm8` (`uint8xm8_t a`, unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] - b - mask[elemet]
result[gvl : VLMAX] = 0
```

## 2.7.80 Elementwise vector-vector multiply with rounding and saturation

**Instruction:** [`'vsmul.vv'`]

**Prototypes:**

- `int16xm1_t vsmulvv_int16xm1` (`int16xm1_t a`, `int16xm1_t b`, unsigned int `gvl`)
- `int16xm2_t vsmulvv_int16xm2` (`int16xm2_t a`, `int16xm2_t b`, unsigned int `gvl`)
- `int16xm4_t vsmulvv_int16xm4` (`int16xm4_t a`, `int16xm4_t b`, unsigned int `gvl`)
- `int16xm8_t vsmulvv_int16xm8` (`int16xm8_t a`, `int16xm8_t b`, unsigned int `gvl`)
- `int32xm1_t vsmulvv_int32xm1` (`int32xm1_t a`, `int32xm1_t b`, unsigned int `gvl`)
- `int32xm2_t vsmulvv_int32xm2` (`int32xm2_t a`, `int32xm2_t b`, unsigned int `gvl`)



- `int32xm4_t vsmulvv_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vsmulvv_int32xm8 (int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vsmulvv_int64xm1 (int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vsmulvv_int64xm2 (int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vsmulvv_int64xm4 (int64xm4_t a, int64xm4_t b, unsigned int gvl)`
- `int64xm8_t vsmulvv_int64xm8 (int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vsmulvv_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vsmulvv_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vsmulvv_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vsmulvv_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = clip(roundoff(a[element] * b[element], SEW - 1))
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vsmulvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vsmulvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vsmulvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vsmulvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vsmulvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsmulvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsmulvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vsmulvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vsmulvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vsmulvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vsmulvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vsmulvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vsmulvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vsmulvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`

- `int8xm4_t vsmulvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vsmulvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = clip(roundoff(a[element] * b[element], SEW - 1))
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.81 Elementwise vector-scalar multiply with rounding and saturation

**Instruction:** [`vsmul.vx`']

**Prototypes:**

- `int16xm1_t vsmulvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int16xm2_t vsmulvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vsmulvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vsmulvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vsmulvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vsmulvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vsmulvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vsmulvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vsmulvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vsmulvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vsmulvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vsmulvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vsmulvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vsmulvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vsmulvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vsmulvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = clip(roundoff(a[element] * b, SEW - 1))
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vsmulvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`

- `int16xm2_t vsmulvx_mask_int16xm2` (`int16xm2_t merge`, `int16xm2_t a`, short `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `int16xm4_t vsmulvx_mask_int16xm4` (`int16xm4_t merge`, `int16xm4_t a`, short `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `int16xm8_t vsmulvx_mask_int16xm8` (`int16xm8_t merge`, `int16xm8_t a`, short `b`, `e16xm8_t mask`, unsigned int `gvl`)
- `int32xm1_t vsmulvx_mask_int32xm1` (`int32xm1_t merge`, `int32xm1_t a`, int `b`, `e32xm1_t mask`, unsigned int `gvl`)
- `int32xm2_t vsmulvx_mask_int32xm2` (`int32xm2_t merge`, `int32xm2_t a`, int `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `int32xm4_t vsmulvx_mask_int32xm4` (`int32xm4_t merge`, `int32xm4_t a`, int `b`, `e32xm4_t mask`, unsigned int `gvl`)
- `int32xm8_t vsmulvx_mask_int32xm8` (`int32xm8_t merge`, `int32xm8_t a`, int `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `int64xm1_t vsmulvx_mask_int64xm1` (`int64xm1_t merge`, `int64xm1_t a`, long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `int64xm2_t vsmulvx_mask_int64xm2` (`int64xm2_t merge`, `int64xm2_t a`, long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `int64xm4_t vsmulvx_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `int64xm8_t vsmulvx_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `int8xm1_t vsmulvx_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, signed char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8xm2_t vsmulvx_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, signed char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `int8xm4_t vsmulvx_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, signed char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int8xm8_t vsmulvx_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, signed char `b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = clip(roundoff(a[element] * b, SEW - 1))
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.7.82 Elementwise signed vector-immediate signed scaling shift****Instruction:** [`'vssra.vi'`]**Prototypes:**

- `int16xm1_t vssravi_int16xm1` (`int16xm1_t a`, const unsigned short `b`, unsigned int `gvl`)
- `int16xm2_t vssravi_int16xm2` (`int16xm2_t a`, const unsigned short `b`, unsigned int `gvl`)
- `int16xm4_t vssravi_int16xm4` (`int16xm4_t a`, const unsigned short `b`, unsigned int `gvl`)

- `int16xm8_t vssravi_int16xm8 (int16xm8_t a, const unsigned short b, unsigned int gvl)`
- `int32xm1_t vssravi_int32xm1 (int32xm1_t a, const unsigned int b, unsigned int gvl)`
- `int32xm2_t vssravi_int32xm2 (int32xm2_t a, const unsigned int b, unsigned int gvl)`
- `int32xm4_t vssravi_int32xm4 (int32xm4_t a, const unsigned int b, unsigned int gvl)`
- `int32xm8_t vssravi_int32xm8 (int32xm8_t a, const unsigned int b, unsigned int gvl)`
- `int64xm1_t vssravi_int64xm1 (int64xm1_t a, const unsigned long b, unsigned int gvl)`
- `int64xm2_t vssravi_int64xm2 (int64xm2_t a, const unsigned long b, unsigned int gvl)`
- `int64xm4_t vssravi_int64xm4 (int64xm4_t a, const unsigned long b, unsigned int gvl)`
- `int64xm8_t vssravi_int64xm8 (int64xm8_t a, const unsigned long b, unsigned int gvl)`
- `int8xm1_t vssravi_int8xm1 (int8xm1_t a, const unsigned char b, unsigned int gvl)`
- `int8xm2_t vssravi_int8xm2 (int8xm2_t a, const unsigned char b, unsigned int gvl)`
- `int8xm4_t vssravi_int8xm4 (int8xm4_t a, const unsigned char b, unsigned int gvl)`
- `int8xm8_t vssravi_int8xm8 (int8xm8_t a, const unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = (a[element] + (1 << b - 1)) >> b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vssravi_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, const unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vssravi_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, const unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vssravi_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, const unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vssravi_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, const unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vssravi_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, const unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vssravi_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, const unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vssravi_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, const unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vssravi_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, const unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vssravi_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, const unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vssravi_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, const unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vssravi_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, const unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vssravi_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, const unsigned long b, e64xm8_t mask, unsigned int gvl)`

- `int8xm1_t vssravi_mask_int8xm1(int8xm1_t merge, int8xm1_t a, const unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vssravi_mask_int8xm2(int8xm2_t merge, int8xm2_t a, const unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vssravi_mask_int8xm4(int8xm4_t merge, int8xm4_t a, const unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vssravi_mask_int8xm8(int8xm8_t merge, int8xm8_t a, const unsigned char b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[elemet] + (1 << b - 1)) >> b[elemet]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.83 Elementwise signed vector-vector signed scaling shift

Instruction: ['vssra.vv']

Prototypes:

- `int16xm1_t vssravv_int16xm1_uint16xm1(int16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `int16xm2_t vssravv_int16xm2_uint16xm2(int16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `int16xm4_t vssravv_int16xm4_uint16xm4(int16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `int16xm8_t vssravv_int16xm8_uint16xm8(int16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `int32xm1_t vssravv_int32xm1_uint32xm1(int32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `int32xm2_t vssravv_int32xm2_uint32xm2(int32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `int32xm4_t vssravv_int32xm4_uint32xm4(int32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `int32xm8_t vssravv_int32xm8_uint32xm8(int32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `int64xm1_t vssravv_int64xm1_uint64xm1(int64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `int64xm2_t vssravv_int64xm2_uint64xm2(int64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `int64xm4_t vssravv_int64xm4_uint64xm4(int64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `int64xm8_t vssravv_int64xm8_uint64xm8(int64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `int8xm1_t vssravv_int8xm1_uint8xm1(int8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `int8xm2_t vssravv_int8xm2_uint8xm2(int8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `int8xm4_t vssravv_int8xm4_uint8xm4(int8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `int8xm8_t vssravv_int8xm8_uint8xm8(int8xm8_t a, uint8xm8_t b, unsigned int gvl)`

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = (a[elemet] + (1 << b[element] - 1)) >> b[elemet]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vssravv\_mask\_int16xm1\_uint16xm1* (*int16xm1\_t* merge, *int16xm1\_t* a, *uint16xm1\_t* b, *e16xm1\_t* mask, unsigned int gvl)
- *int16xm2\_t vssravv\_mask\_int16xm2\_uint16xm2* (*int16xm2\_t* merge, *int16xm2\_t* a, *uint16xm2\_t* b, *e16xm2\_t* mask, unsigned int gvl)
- *int16xm4\_t vssravv\_mask\_int16xm4\_uint16xm4* (*int16xm4\_t* merge, *int16xm4\_t* a, *uint16xm4\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *int16xm8\_t vssravv\_mask\_int16xm8\_uint16xm8* (*int16xm8\_t* merge, *int16xm8\_t* a, *uint16xm8\_t* b, *e16xm8\_t* mask, unsigned int gvl)
- *int32xm1\_t vssravv\_mask\_int32xm1\_uint32xm1* (*int32xm1\_t* merge, *int32xm1\_t* a, *uint32xm1\_t* b, *e32xm1\_t* mask, unsigned int gvl)
- *int32xm2\_t vssravv\_mask\_int32xm2\_uint32xm2* (*int32xm2\_t* merge, *int32xm2\_t* a, *uint32xm2\_t* b, *e32xm2\_t* mask, unsigned int gvl)
- *int32xm4\_t vssravv\_mask\_int32xm4\_uint32xm4* (*int32xm4\_t* merge, *int32xm4\_t* a, *uint32xm4\_t* b, *e32xm4\_t* mask, unsigned int gvl)
- *int32xm8\_t vssravv\_mask\_int32xm8\_uint32xm8* (*int32xm8\_t* merge, *int32xm8\_t* a, *uint32xm8\_t* b, *e32xm8\_t* mask, unsigned int gvl)
- *int64xm1\_t vssravv\_mask\_int64xm1\_uint64xm1* (*int64xm1\_t* merge, *int64xm1\_t* a, *uint64xm1\_t* b, *e64xm1\_t* mask, unsigned int gvl)
- *int64xm2\_t vssravv\_mask\_int64xm2\_uint64xm2* (*int64xm2\_t* merge, *int64xm2\_t* a, *uint64xm2\_t* b, *e64xm2\_t* mask, unsigned int gvl)
- *int64xm4\_t vssravv\_mask\_int64xm4\_uint64xm4* (*int64xm4\_t* merge, *int64xm4\_t* a, *uint64xm4\_t* b, *e64xm4\_t* mask, unsigned int gvl)
- *int64xm8\_t vssravv\_mask\_int64xm8\_uint64xm8* (*int64xm8\_t* merge, *int64xm8\_t* a, *uint64xm8\_t* b, *e64xm8\_t* mask, unsigned int gvl)
- *int8xm1\_t vssravv\_mask\_int8xm1\_uint8xm1* (*int8xm1\_t* merge, *int8xm1\_t* a, *uint8xm1\_t* b, *e8xm1\_t* mask, unsigned int gvl)
- *int8xm2\_t vssravv\_mask\_int8xm2\_uint8xm2* (*int8xm2\_t* merge, *int8xm2\_t* a, *uint8xm2\_t* b, *e8xm2\_t* mask, unsigned int gvl)
- *int8xm4\_t vssravv\_mask\_int8xm4\_uint8xm4* (*int8xm4\_t* merge, *int8xm4\_t* a, *uint8xm4\_t* b, *e8xm4\_t* mask, unsigned int gvl)
- *int8xm8\_t vssravv\_mask\_int8xm8\_uint8xm8* (*int8xm8\_t* merge, *int8xm8\_t* a, *uint8xm8\_t* b, *e8xm8\_t* mask, unsigned int gvl)

**Masked operation:**



```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = (a[elemet] + (1 << b[element] - 1)) >> b[elemet]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.84 Elementwise signed vector-scalar signed scaling shift

**Instruction:** ['vssra.vx']

**Prototypes:**

- *int16xm1\_t vssravx\_int16xm1* (*int16xm1\_t a*, unsigned short *b*, unsigned int *gvl*)
- *int16xm2\_t vssravx\_int16xm2* (*int16xm2\_t a*, unsigned short *b*, unsigned int *gvl*)
- *int16xm4\_t vssravx\_int16xm4* (*int16xm4\_t a*, unsigned short *b*, unsigned int *gvl*)
- *int16xm8\_t vssravx\_int16xm8* (*int16xm8\_t a*, unsigned short *b*, unsigned int *gvl*)
- *int32xm1\_t vssravx\_int32xm1* (*int32xm1\_t a*, unsigned int *b*, unsigned int *gvl*)
- *int32xm2\_t vssravx\_int32xm2* (*int32xm2\_t a*, unsigned int *b*, unsigned int *gvl*)
- *int32xm4\_t vssravx\_int32xm4* (*int32xm4\_t a*, unsigned int *b*, unsigned int *gvl*)
- *int32xm8\_t vssravx\_int32xm8* (*int32xm8\_t a*, unsigned int *b*, unsigned int *gvl*)
- *int64xm1\_t vssravx\_int64xm1* (*int64xm1\_t a*, unsigned long *b*, unsigned int *gvl*)
- *int64xm2\_t vssravx\_int64xm2* (*int64xm2\_t a*, unsigned long *b*, unsigned int *gvl*)
- *int64xm4\_t vssravx\_int64xm4* (*int64xm4\_t a*, unsigned long *b*, unsigned int *gvl*)
- *int64xm8\_t vssravx\_int64xm8* (*int64xm8\_t a*, unsigned long *b*, unsigned int *gvl*)
- *int8xm1\_t vssravx\_int8xm1* (*int8xm1\_t a*, unsigned char *b*, unsigned int *gvl*)
- *int8xm2\_t vssravx\_int8xm2* (*int8xm2\_t a*, unsigned char *b*, unsigned int *gvl*)
- *int8xm4\_t vssravx\_int8xm4* (*int8xm4\_t a*, unsigned char *b*, unsigned int *gvl*)
- *int8xm8\_t vssravx\_int8xm8* (*int8xm8\_t a*, unsigned char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = (a[elemet] + (1 << b - 1)) >> b[elemet]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vssravx\_mask\_int16xm1* (*int16xm1\_t merge*, *int16xm1\_t a*, unsigned short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vssravx\_mask\_int16xm2* (*int16xm2\_t merge*, *int16xm2\_t a*, unsigned short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vssravx\_mask\_int16xm4* (*int16xm4\_t merge*, *int16xm4\_t a*, unsigned short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vssravx\_mask\_int16xm8* (*int16xm8\_t merge*, *int16xm8\_t a*, unsigned short *b*, *e16xm8\_t mask*, unsigned int *gvl*)

- `int32xm1_t vssravx_mask_int32xm1(int32xm1_t merge, int32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vssravx_mask_int32xm2(int32xm2_t merge, int32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vssravx_mask_int32xm4(int32xm4_t merge, int32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vssravx_mask_int32xm8(int32xm8_t merge, int32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vssravx_mask_int64xm1(int64xm1_t merge, int64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vssravx_mask_int64xm2(int64xm2_t merge, int64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vssravx_mask_int64xm4(int64xm4_t merge, int64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vssravx_mask_int64xm8(int64xm8_t merge, int64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vssravx_mask_int8xm1(int8xm1_t merge, int8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vssravx_mask_int8xm2(int8xm2_t merge, int8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vssravx_mask_int8xm4(int8xm4_t merge, int8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vssravx_mask_int8xm8(int8xm8_t merge, int8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = (a[elemet] + (1 << b - 1)) >> b[elemet]
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.7.85 Elementwise unsigned vector-immediate unsigned scaling shift

Instruction: [`'vssrlvi'`]

Prototypes:

- `uint16xm1_t vssrlvi_uint16xm1(uint16xm1_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm2_t vssrlvi_uint16xm2(uint16xm2_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm4_t vssrlvi_uint16xm4(uint16xm4_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm8_t vssrlvi_uint16xm8(uint16xm8_t a, const unsigned short b, unsigned int gvl)`
- `uint32xm1_t vssrlvi_uint32xm1(uint32xm1_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm2_t vssrlvi_uint32xm2(uint32xm2_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm4_t vssrlvi_uint32xm4(uint32xm4_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm8_t vssrlvi_uint32xm8(uint32xm8_t a, const unsigned int b, unsigned int gvl)`

- `uint64xm1_t vssrlvi_uint64xm1 (uint64xm1_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm2_t vssrlvi_uint64xm2 (uint64xm2_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm4_t vssrlvi_uint64xm4 (uint64xm4_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm8_t vssrlvi_uint64xm8 (uint64xm8_t a, const unsigned long b, unsigned int gvl)`
- `uint8xm1_t vssrlvi_uint8xm1 (uint8xm1_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm2_t vssrlvi_uint8xm2 (uint8xm2_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm4_t vssrlvi_uint8xm4 (uint8xm4_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm8_t vssrlvi_uint8xm8 (uint8xm8_t a, const unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] + (1 << b - 1)) >> b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vssrlvi_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, const unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vssrlvi_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, const unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vssrlvi_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, const unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vssrlvi_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, const unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vssrlvi_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, const unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vssrlvi_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, const unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vssrlvi_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, const unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vssrlvi_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, const unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vssrlvi_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, const unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vssrlvi_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, const unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vssrlvi_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, const unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vssrlvi_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, const unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vssrlvi_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, const unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vssrlvi_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, const unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vssrlvi_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, const unsigned char b, e8xm4_t mask, unsigned int gvl)`

- `uint8xm8_t vssrlvi_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, const unsigned char b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = (a[elemet] + (1 << b - 1)) >> b[elemet]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.86 Elementwise unsigned vector-vector unsigned scaling shift

Instruction: ['vssrl.vv']

Prototypes:

- `uint16xm1_t vssrlvv_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vssrlvv_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vssrlvv_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vssrlvv_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vssrlvv_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vssrlvv_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vssrlvv_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vssrlvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vssrlvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vssrlvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vssrlvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vssrlvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vssrlvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vssrlvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vssrlvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vssrlvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

Operation:

```
>>> for element = 0 to gvl - 1
    result[element] = (a[elemet] + (1 << b[element] - 1)) >> b[elemet]
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `uint16xm1_t vssrlvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vssrlvv_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`

- `uint16xm4_t vssrlvv_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vssrlvv_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vssrlvv_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vssrlvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vssrlvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vssrlvv_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vssrlvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vssrlvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vssrlvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vssrlvv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vssrlvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vssrlvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vssrlvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vssrlvv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = (a[element] + (1 << b[element] - 1)) >> b[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.87 Elementwise unsigned vector-scalar unsigned scaling shift

**Instruction:** ['vssrlv.vx']

#### Prototypes:

- `uint16xm1_t vssrlvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vssrlvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vssrlvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vssrlvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vssrlvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`

- `uint32xm2_t vssrlvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vssrlvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vssrlvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vssrlvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vssrlvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vssrlvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vssrlvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vssrlvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vssrlvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vssrlvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vssrlvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = (a[element] + (1 << b - 1)) >> b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vssrlvx_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vssrlvx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vssrlvx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vssrlvx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vssrlvx_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vssrlvx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vssrlvx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vssrlvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vssrlvx_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vssrlvx_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vssrlvx_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vssrlvx_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vssrlvx_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`



- `uint8xm2_t vssrlvx_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vssrlvx_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vssrlvx_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = (a[element] + (1 << b - 1)) >> b[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.88 Elementwise signed vector-vector subtraction with saturation

**Instruction:** ['vssub.vv']

**Prototypes:**

- `int16xm1_t vssubvv_int16xm1 (int16xm1_t a, int16xm1_t b, unsigned int gvl)`
- `int16xm2_t vssubvv_int16xm2 (int16xm2_t a, int16xm2_t b, unsigned int gvl)`
- `int16xm4_t vssubvv_int16xm4 (int16xm4_t a, int16xm4_t b, unsigned int gvl)`
- `int16xm8_t vssubvv_int16xm8 (int16xm8_t a, int16xm8_t b, unsigned int gvl)`
- `int32xm1_t vssubvv_int32xm1 (int32xm1_t a, int32xm1_t b, unsigned int gvl)`
- `int32xm2_t vssubvv_int32xm2 (int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int32xm4_t vssubvv_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`
- `int32xm8_t vssubvv_int32xm8 (int32xm8_t a, int32xm8_t b, unsigned int gvl)`
- `int64xm1_t vssubvv_int64xm1 (int64xm1_t a, int64xm1_t b, unsigned int gvl)`
- `int64xm2_t vssubvv_int64xm2 (int64xm2_t a, int64xm2_t b, unsigned int gvl)`
- `int64xm4_t vssubvv_int64xm4 (int64xm4_t a, int64xm4_t b, unsigned int gvl)`
- `int64xm8_t vssubvv_int64xm8 (int64xm8_t a, int64xm8_t b, unsigned int gvl)`
- `int8xm1_t vssubvv_int8xm1 (int8xm1_t a, int8xm1_t b, unsigned int gvl)`
- `int8xm2_t vssubvv_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int8xm4_t vssubvv_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vssubvv_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = saturat(a[element] - b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vssubvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vssubvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vssubvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vssubvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vssubvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vssubvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vssubvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vssubvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vssubvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vssubvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vssubvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vssubvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vssubvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vssubvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vssubvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vssubvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = saturat(a[element] - b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.89 Elementwise signed vector-scalar subtraction with saturation

**Instruction:** ['vssub.vx']

**Prototypes:**

- `int16xm1_t vssubvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`

- *int16xm2\_t vssubvx\_int16xm2* (*int16xm2\_t a*, short *b*, unsigned int *gvl*)
- *int16xm4\_t vssubvx\_int16xm4* (*int16xm4\_t a*, short *b*, unsigned int *gvl*)
- *int16xm8\_t vssubvx\_int16xm8* (*int16xm8\_t a*, short *b*, unsigned int *gvl*)
- *int32xm1\_t vssubvx\_int32xm1* (*int32xm1\_t a*, int *b*, unsigned int *gvl*)
- *int32xm2\_t vssubvx\_int32xm2* (*int32xm2\_t a*, int *b*, unsigned int *gvl*)
- *int32xm4\_t vssubvx\_int32xm4* (*int32xm4\_t a*, int *b*, unsigned int *gvl*)
- *int32xm8\_t vssubvx\_int32xm8* (*int32xm8\_t a*, int *b*, unsigned int *gvl*)
- *int64xm1\_t vssubvx\_int64xm1* (*int64xm1\_t a*, long *b*, unsigned int *gvl*)
- *int64xm2\_t vssubvx\_int64xm2* (*int64xm2\_t a*, long *b*, unsigned int *gvl*)
- *int64xm4\_t vssubvx\_int64xm4* (*int64xm4\_t a*, long *b*, unsigned int *gvl*)
- *int64xm8\_t vssubvx\_int64xm8* (*int64xm8\_t a*, long *b*, unsigned int *gvl*)
- *int8xm1\_t vssubvx\_int8xm1* (*int8xm1\_t a*, signed char *b*, unsigned int *gvl*)
- *int8xm2\_t vssubvx\_int8xm2* (*int8xm2\_t a*, signed char *b*, unsigned int *gvl*)
- *int8xm4\_t vssubvx\_int8xm4* (*int8xm4\_t a*, signed char *b*, unsigned int *gvl*)
- *int8xm8\_t vssubvx\_int8xm8* (*int8xm8\_t a*, signed char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = saturat(a[element] - b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vssubvx\_mask\_int16xm1* (*int16xm1\_t merge*, *int16xm1\_t a*, short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vssubvx\_mask\_int16xm2* (*int16xm2\_t merge*, *int16xm2\_t a*, short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vssubvx\_mask\_int16xm4* (*int16xm4\_t merge*, *int16xm4\_t a*, short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vssubvx\_mask\_int16xm8* (*int16xm8\_t merge*, *int16xm8\_t a*, short *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *int32xm1\_t vssubvx\_mask\_int32xm1* (*int32xm1\_t merge*, *int32xm1\_t a*, int *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vssubvx\_mask\_int32xm2* (*int32xm2\_t merge*, *int32xm2\_t a*, int *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vssubvx\_mask\_int32xm4* (*int32xm4\_t merge*, *int32xm4\_t a*, int *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *int32xm8\_t vssubvx\_mask\_int32xm8* (*int32xm8\_t merge*, *int32xm8\_t a*, int *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *int64xm1\_t vssubvx\_mask\_int64xm1* (*int64xm1\_t merge*, *int64xm1\_t a*, long *b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *int64xm2\_t vssubvx\_mask\_int64xm2* (*int64xm2\_t merge*, *int64xm2\_t a*, long *b*, *e64xm2\_t mask*, unsigned int *gvl*)

- `int64xm4_t vssubvx_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `int64xm8_t vssubvx_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `int8xm1_t vssubvx_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, signed char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8xm2_t vssubvx_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, signed char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `int8xm4_t vssubvx_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, signed char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int8xm8_t vssubvx_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, signed char `b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = saturat(a[element] - b)
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.90 Elementwise unsigned vector-vector subtraction with saturation****Instruction:** [`'vssubu.vv'`]**Prototypes:**

- `uint16xm1_t vssubuvv_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, unsigned int `gvl`)
- `uint16xm2_t vssubuvv_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, unsigned int `gvl`)
- `uint16xm4_t vssubuvv_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, unsigned int `gvl`)
- `uint16xm8_t vssubuvv_uint16xm8` (`uint16xm8_t a`, `uint16xm8_t b`, unsigned int `gvl`)
- `uint32xm1_t vssubuvv_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, unsigned int `gvl`)
- `uint32xm2_t vssubuvv_uint32xm2` (`uint32xm2_t a`, `uint32xm2_t b`, unsigned int `gvl`)
- `uint32xm4_t vssubuvv_uint32xm4` (`uint32xm4_t a`, `uint32xm4_t b`, unsigned int `gvl`)
- `uint32xm8_t vssubuvv_uint32xm8` (`uint32xm8_t a`, `uint32xm8_t b`, unsigned int `gvl`)
- `uint64xm1_t vssubuvv_uint64xm1` (`uint64xm1_t a`, `uint64xm1_t b`, unsigned int `gvl`)
- `uint64xm2_t vssubuvv_uint64xm2` (`uint64xm2_t a`, `uint64xm2_t b`, unsigned int `gvl`)
- `uint64xm4_t vssubuvv_uint64xm4` (`uint64xm4_t a`, `uint64xm4_t b`, unsigned int `gvl`)
- `uint64xm8_t vssubuvv_uint64xm8` (`uint64xm8_t a`, `uint64xm8_t b`, unsigned int `gvl`)
- `uint8xm1_t vssubuvv_uint8xm1` (`uint8xm1_t a`, `uint8xm1_t b`, unsigned int `gvl`)
- `uint8xm2_t vssubuvv_uint8xm2` (`uint8xm2_t a`, `uint8xm2_t b`, unsigned int `gvl`)
- `uint8xm4_t vssubuvv_uint8xm4` (`uint8xm4_t a`, `uint8xm4_t b`, unsigned int `gvl`)
- `uint8xm8_t vssubuvv_uint8xm8` (`uint8xm8_t a`, `uint8xm8_t b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = saturat(a[element] - b[element])
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *uint16xm1\_t vssubuvv\_mask\_uint16xm1* (*uint16xm1\_t merge, uint16xm1\_t a, uint16xm1\_t b, e16xm1\_t mask*, unsigned int gvl)
- *uint16xm2\_t vssubuvv\_mask\_uint16xm2* (*uint16xm2\_t merge, uint16xm2\_t a, uint16xm2\_t b, e16xm2\_t mask*, unsigned int gvl)
- *uint16xm4\_t vssubuvv\_mask\_uint16xm4* (*uint16xm4\_t merge, uint16xm4\_t a, uint16xm4\_t b, e16xm4\_t mask*, unsigned int gvl)
- *uint16xm8\_t vssubuvv\_mask\_uint16xm8* (*uint16xm8\_t merge, uint16xm8\_t a, uint16xm8\_t b, e16xm8\_t mask*, unsigned int gvl)
- *uint32xm1\_t vssubuvv\_mask\_uint32xm1* (*uint32xm1\_t merge, uint32xm1\_t a, uint32xm1\_t b, e32xm1\_t mask*, unsigned int gvl)
- *uint32xm2\_t vssubuvv\_mask\_uint32xm2* (*uint32xm2\_t merge, uint32xm2\_t a, uint32xm2\_t b, e32xm2\_t mask*, unsigned int gvl)
- *uint32xm4\_t vssubuvv\_mask\_uint32xm4* (*uint32xm4\_t merge, uint32xm4\_t a, uint32xm4\_t b, e32xm4\_t mask*, unsigned int gvl)
- *uint32xm8\_t vssubuvv\_mask\_uint32xm8* (*uint32xm8\_t merge, uint32xm8\_t a, uint32xm8\_t b, e32xm8\_t mask*, unsigned int gvl)
- *uint64xm1\_t vssubuvv\_mask\_uint64xm1* (*uint64xm1\_t merge, uint64xm1\_t a, uint64xm1\_t b, e64xm1\_t mask*, unsigned int gvl)
- *uint64xm2\_t vssubuvv\_mask\_uint64xm2* (*uint64xm2\_t merge, uint64xm2\_t a, uint64xm2\_t b, e64xm2\_t mask*, unsigned int gvl)
- *uint64xm4\_t vssubuvv\_mask\_uint64xm4* (*uint64xm4\_t merge, uint64xm4\_t a, uint64xm4\_t b, e64xm4\_t mask*, unsigned int gvl)
- *uint64xm8\_t vssubuvv\_mask\_uint64xm8* (*uint64xm8\_t merge, uint64xm8\_t a, uint64xm8\_t b, e64xm8\_t mask*, unsigned int gvl)
- *uint8xm1\_t vssubuvv\_mask\_uint8xm1* (*uint8xm1\_t merge, uint8xm1\_t a, uint8xm1\_t b, e8xm1\_t mask*, unsigned int gvl)
- *uint8xm2\_t vssubuvv\_mask\_uint8xm2* (*uint8xm2\_t merge, uint8xm2\_t a, uint8xm2\_t b, e8xm2\_t mask*, unsigned int gvl)
- *uint8xm4\_t vssubuvv\_mask\_uint8xm4* (*uint8xm4\_t merge, uint8xm4\_t a, uint8xm4\_t b, e8xm4\_t mask*, unsigned int gvl)
- *uint8xm8\_t vssubuvv\_mask\_uint8xm8* (*uint8xm8\_t merge, uint8xm8\_t a, uint8xm8\_t b, e8xm8\_t mask*, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = saturat(a[element] - b[element])
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.7.91 Elementwise unsigned vector-scalar subtraction with saturation

**Instruction:** [`'vssubu.vx'`]

**Prototypes:**

- `uint16xm1_t vssubuvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vssubuvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vssubuvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vssubuvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vssubuvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vssubuvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vssubuvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vssubuvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vssubuvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vssubuvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vssubuvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vssubuvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vssubuvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vssubuvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vssubuvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vssubuvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = saturat(a[element] - b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vssubuvx_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vssubuvx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vssubuvx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vssubuvx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vssubuvx_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vssubuvx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vssubuvx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vssubuvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`



- `uint64xm1_t vssubuvx_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, unsigned long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vssubuvx_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, unsigned long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vssubuvx_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, unsigned long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vssubuvx_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, unsigned long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vssubuvx_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vssubuvx_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vssubuvx_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vssubuvx_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = saturat(a[element] - b)
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.7.92 Elementwise vector-vector integer subtraction

Instruction: ['vsub.vv']

Prototypes:

- `int16xm1_t vsubvv_int16xm1` (`int16xm1_t a`, `int16xm1_t b`, unsigned int `gvl`)
- `int16xm2_t vsubvv_int16xm2` (`int16xm2_t a`, `int16xm2_t b`, unsigned int `gvl`)
- `int16xm4_t vsubvv_int16xm4` (`int16xm4_t a`, `int16xm4_t b`, unsigned int `gvl`)
- `int16xm8_t vsubvv_int16xm8` (`int16xm8_t a`, `int16xm8_t b`, unsigned int `gvl`)
- `int32xm1_t vsubvv_int32xm1` (`int32xm1_t a`, `int32xm1_t b`, unsigned int `gvl`)
- `int32xm2_t vsubvv_int32xm2` (`int32xm2_t a`, `int32xm2_t b`, unsigned int `gvl`)
- `int32xm4_t vsubvv_int32xm4` (`int32xm4_t a`, `int32xm4_t b`, unsigned int `gvl`)
- `int32xm8_t vsubvv_int32xm8` (`int32xm8_t a`, `int32xm8_t b`, unsigned int `gvl`)
- `int64xm1_t vsubvv_int64xm1` (`int64xm1_t a`, `int64xm1_t b`, unsigned int `gvl`)
- `int64xm2_t vsubvv_int64xm2` (`int64xm2_t a`, `int64xm2_t b`, unsigned int `gvl`)
- `int64xm4_t vsubvv_int64xm4` (`int64xm4_t a`, `int64xm4_t b`, unsigned int `gvl`)
- `int64xm8_t vsubvv_int64xm8` (`int64xm8_t a`, `int64xm8_t b`, unsigned int `gvl`)
- `int8xm1_t vsubvv_int8xm1` (`int8xm1_t a`, `int8xm1_t b`, unsigned int `gvl`)
- `int8xm2_t vsubvv_int8xm2` (`int8xm2_t a`, `int8xm2_t b`, unsigned int `gvl`)

- `int8xm4_t vsubvv_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int8xm8_t vsubvv_int8xm8 (int8xm8_t a, int8xm8_t b, unsigned int gvl)`
- `uint16xm1_t vsubvv_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vsubvv_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vsubvv_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vsubvv_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vsubvv_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vsubvv_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vsubvv_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vsubvv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vsubvv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vsubvv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vsubvv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vsubvv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vsubvv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vsubvv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vsubvv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vsubvv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] - b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vsubvv_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vsubvv_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vsubvv_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vsubvv_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, int16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vsubvv_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsubvv_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsubvv_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vsubvv_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int32xm8_t b, e32xm8_t mask, unsigned int gvl)`

- `int64xm1_t vsubvv_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, int64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vsubvv_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, int64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vsubvv_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, int64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vsubvv_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, int64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vsubvv_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vsubvv_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vsubvv_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vsubvv_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, int8xm8_t b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vsubvv_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vsubvv_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vsubvv_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vsubvv_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vsubvv_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vsubvv_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vsubvv_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vsubvv_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vsubvv_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, uint64xm1_t b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vsubvv_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, uint64xm2_t b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vsubvv_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, uint64xm4_t b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vsubvv_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vsubvv_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vsubvv_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vsubvv_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`

- `uint8xm8_t vsubvv_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] - b[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.93 Elementwise vector-scalar integer subtraction

Instruction: ['vsub.vx']

Prototypes:

- `int16xm1_t vsubvx_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int16xm2_t vsubvx_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int16xm4_t vsubvx_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int16xm8_t vsubvx_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `int32xm1_t vsubvx_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int32xm2_t vsubvx_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int32xm4_t vsubvx_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `int32xm8_t vsubvx_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `int64xm1_t vsubvx_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `int64xm2_t vsubvx_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `int64xm4_t vsubvx_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `int64xm8_t vsubvx_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `int8xm1_t vsubvx_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int8xm2_t vsubvx_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int8xm4_t vsubvx_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int8xm8_t vsubvx_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`
- `uint16xm1_t vsubvx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vsubvx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vsubvx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vsubvx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vsubvx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vsubvx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vsubvx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vsubvx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vsubvx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`

- `uint64xm2_t vsubvx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vsubvx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vsubvx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vsubvx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vsubvx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vsubvx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vsubvx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] - b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vsubvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vsubvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vsubvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vsubvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vsubvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vsubvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vsubvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vsubvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vsubvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vsubvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vsubvx_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vsubvx_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vsubvx_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vsubvx_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vsubvx_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, signed char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vsubvx_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, signed char b, e8xm8_t mask, unsigned int gvl)`

- `uint16xm1_t vsubvx_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, unsigned short `b`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint16xm2_t vsubvx_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, unsigned short `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint16xm4_t vsubvx_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, unsigned short `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vsubvx_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, unsigned short `b`, `e16xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vsubvx_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, unsigned int `b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vsubvx_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, unsigned int `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vsubvx_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, unsigned int `b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint32xm8_t vsubvx_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, unsigned int `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint64xm1_t vsubvx_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, unsigned long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vsubvx_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, unsigned long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vsubvx_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, unsigned long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vsubvx_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, unsigned long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vsubvx_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vsubvx_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vsubvx_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vsubvx_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] - b
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.7.94 Widening elementwise signed vector-vector addition

**Instruction:** [`'vwadd.vv'`]

**Prototypes:**

- `int16xm2_t vwaddvv_int16xm2_int8xm1` (`int8xm1_t a`, `int8xm1_t b`, unsigned int `gvl`)



- `int16xm4_t vwaddvv_int16xm4_int8xm2 (int8xm2_t a, int8xm2_t b, unsigned int gvl)`
- `int16xm8_t vwaddvv_int16xm8_int8xm4 (int8xm4_t a, int8xm4_t b, unsigned int gvl)`
- `int32xm2_t vwaddvv_int32xm2_int16xm1 (int16xm1_t a, int16xm1_t b, unsigned int gvl)`
- `int32xm4_t vwaddvv_int32xm4_int16xm2 (int16xm2_t a, int16xm2_t b, unsigned int gvl)`
- `int32xm8_t vwaddvv_int32xm8_int16xm4 (int16xm4_t a, int16xm4_t b, unsigned int gvl)`
- `int64xm2_t vwaddvv_int64xm2_int32xm1 (int32xm1_t a, int32xm1_t b, unsigned int gvl)`
- `int64xm4_t vwaddvv_int64xm4_int32xm2 (int32xm2_t a, int32xm2_t b, unsigned int gvl)`
- `int64xm8_t vwaddvv_int64xm8_int32xm4 (int32xm4_t a, int32xm4_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = widen_integer (a[element]) + widen_integer (b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm2_t vwaddvv_mask_int16xm2_int8xm1 (int16xm2_t merge, int8xm1_t a, int8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `int16xm4_t vwaddvv_mask_int16xm4_int8xm2 (int16xm4_t merge, int8xm2_t a, int8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `int16xm8_t vwaddvv_mask_int16xm8_int8xm4 (int16xm8_t merge, int8xm4_t a, int8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `int32xm2_t vwaddvv_mask_int32xm2_int16xm1 (int32xm2_t merge, int16xm1_t a, int16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `int32xm4_t vwaddvv_mask_int32xm4_int16xm2 (int32xm4_t merge, int16xm2_t a, int16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `int32xm8_t vwaddvv_mask_int32xm8_int16xm4 (int32xm8_t merge, int16xm4_t a, int16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int64xm2_t vwaddvv_mask_int64xm2_int32xm1 (int64xm2_t merge, int32xm1_t a, int32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int64xm4_t vwaddvv_mask_int64xm4_int32xm2 (int64xm4_t merge, int32xm2_t a, int32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int64xm8_t vwaddvv_mask_int64xm8_int32xm4 (int64xm8_t merge, int32xm4_t a, int32xm4_t b, e32xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = widen_integer (a[element]) + widen_integer (b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.95 Widening elementwise signed vector-scalar addition****Instruction:** ['vwadd.vx']**Prototypes:**

- `int16xm2_t vwaddvx_int16xm2_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int16xm4_t vwaddvx_int16xm4_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int16xm8_t vwaddvx_int16xm8_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int32xm2_t vwaddvx_int32xm2_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int32xm4_t vwaddvx_int32xm4_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int32xm8_t vwaddvx_int32xm8_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int64xm2_t vwaddvx_int64xm2_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int64xm4_t vwaddvx_int64xm4_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int64xm8_t vwaddvx_int64xm8_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = widen_integer (a[element]) + widen_integer (b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm2_t vwaddvx_mask_int16xm2_int8xm1 (int16xm2_t merge, int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl)`
- `int16xm4_t vwaddvx_mask_int16xm4_int8xm2 (int16xm4_t merge, int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl)`
- `int16xm8_t vwaddvx_mask_int16xm8_int8xm4 (int16xm8_t merge, int8xm4_t a, signed char b, e8xm4_t mask, unsigned int gvl)`
- `int32xm2_t vwaddvx_mask_int32xm2_int16xm1 (int32xm2_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int32xm4_t vwaddvx_mask_int32xm4_int16xm2 (int32xm4_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int32xm8_t vwaddvx_mask_int32xm8_int16xm4 (int32xm8_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int64xm2_t vwaddvx_mask_int64xm2_int32xm1 (int64xm2_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int64xm4_t vwaddvx_mask_int64xm4_int32xm2 (int64xm4_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int64xm8_t vwaddvx_mask_int64xm8_int32xm4 (int64xm8_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = widen_integer (a[element]) + widen_integer (b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.96 Widening elementwise (Widening)signed vector-vector addition

**Instruction:** ['vwadd.wv']

**Prototypes:**

- *int16xm2\_t* vwaddwv\_int16xm2\_int8xm1 (*int16xm2\_t* a, *int8xm1\_t* b, unsigned int gvl)
- *int16xm4\_t* vwaddwv\_int16xm4\_int8xm2 (*int16xm4\_t* a, *int8xm2\_t* b, unsigned int gvl)
- *int16xm8\_t* vwaddwv\_int16xm8\_int8xm4 (*int16xm8\_t* a, *int8xm4\_t* b, unsigned int gvl)
- *int32xm2\_t* vwaddwv\_int32xm2\_int16xm1 (*int32xm2\_t* a, *int16xm1\_t* b, unsigned int gvl)
- *int32xm4\_t* vwaddwv\_int32xm4\_int16xm2 (*int32xm4\_t* a, *int16xm2\_t* b, unsigned int gvl)
- *int32xm8\_t* vwaddwv\_int32xm8\_int16xm4 (*int32xm8\_t* a, *int16xm4\_t* b, unsigned int gvl)
- *int64xm2\_t* vwaddwv\_int64xm2\_int32xm1 (*int64xm2\_t* a, *int32xm1\_t* b, unsigned int gvl)
- *int64xm4\_t* vwaddwv\_int64xm4\_int32xm2 (*int64xm4\_t* a, *int32xm2\_t* b, unsigned int gvl)
- *int64xm8\_t* vwaddwv\_int64xm8\_int32xm4 (*int64xm8\_t* a, *int32xm4\_t* b, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + widen_integer (b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm2\_t* vwaddwv\_mask\_int16xm2\_int8xm1 (*int16xm2\_t* merge, *int16xm2\_t* a, *int8xm1\_t* b, *e8xm1\_t* mask, unsigned int gvl)
- *int16xm4\_t* vwaddwv\_mask\_int16xm4\_int8xm2 (*int16xm4\_t* merge, *int16xm4\_t* a, *int8xm2\_t* b, *e8xm2\_t* mask, unsigned int gvl)
- *int16xm8\_t* vwaddwv\_mask\_int16xm8\_int8xm4 (*int16xm8\_t* merge, *int16xm8\_t* a, *int8xm4\_t* b, *e8xm4\_t* mask, unsigned int gvl)
- *int32xm2\_t* vwaddwv\_mask\_int32xm2\_int16xm1 (*int32xm2\_t* merge, *int32xm2\_t* a, *int16xm1\_t* b, *e16xm1\_t* mask, unsigned int gvl)
- *int32xm4\_t* vwaddwv\_mask\_int32xm4\_int16xm2 (*int32xm4\_t* merge, *int32xm4\_t* a, *int16xm2\_t* b, *e16xm2\_t* mask, unsigned int gvl)
- *int32xm8\_t* vwaddwv\_mask\_int32xm8\_int16xm4 (*int32xm8\_t* merge, *int32xm8\_t* a, *int16xm4\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *int64xm2\_t* vwaddwv\_mask\_int64xm2\_int32xm1 (*int64xm2\_t* merge, *int64xm2\_t* a, *int32xm1\_t* b, *e32xm1\_t* mask, unsigned int gvl)
- *int64xm4\_t* vwaddwv\_mask\_int64xm4\_int32xm2 (*int64xm4\_t* merge, *int64xm4\_t* a, *int32xm2\_t* b, *e32xm2\_t* mask, unsigned int gvl)
- *int64xm8\_t* vwaddwv\_mask\_int64xm8\_int32xm4 (*int64xm8\_t* merge, *int64xm8\_t* a, *int32xm4\_t* b, *e32xm4\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] + widen_integer (b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.97 Widening elementwise (Widening)signed vector-scalar addition

**Instruction:** ['vwadd.wx']

**Prototypes:**

- *int16xm2\_t* **vwaddwx\_int16xm2** (*int16xm2\_t* *a*, signed char *b*, unsigned int *gvl*)
- *int16xm4\_t* **vwaddwx\_int16xm4** (*int16xm4\_t* *a*, signed char *b*, unsigned int *gvl*)
- *int16xm8\_t* **vwaddwx\_int16xm8** (*int16xm8\_t* *a*, signed char *b*, unsigned int *gvl*)
- *int32xm2\_t* **vwaddwx\_int32xm2** (*int32xm2\_t* *a*, short *b*, unsigned int *gvl*)
- *int32xm4\_t* **vwaddwx\_int32xm4** (*int32xm4\_t* *a*, short *b*, unsigned int *gvl*)
- *int32xm8\_t* **vwaddwx\_int32xm8** (*int32xm8\_t* *a*, short *b*, unsigned int *gvl*)
- *int64xm2\_t* **vwaddwx\_int64xm2** (*int64xm2\_t* *a*, int *b*, unsigned int *gvl*)
- *int64xm4\_t* **vwaddwx\_int64xm4** (*int64xm4\_t* *a*, int *b*, unsigned int *gvl*)
- *int64xm8\_t* **vwaddwx\_int64xm8** (*int64xm8\_t* *a*, int *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + widen_integer (b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm2\_t* **vwaddwx\_mask\_int16xm2** (*int16xm2\_t* *merge*, *int16xm2\_t* *a*, signed char *b*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *int16xm4\_t* **vwaddwx\_mask\_int16xm4** (*int16xm4\_t* *merge*, *int16xm4\_t* *a*, signed char *b*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- *int16xm8\_t* **vwaddwx\_mask\_int16xm8** (*int16xm8\_t* *merge*, *int16xm8\_t* *a*, signed char *b*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- *int32xm2\_t* **vwaddwx\_mask\_int32xm2** (*int32xm2\_t* *merge*, *int32xm2\_t* *a*, short *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *int32xm4\_t* **vwaddwx\_mask\_int32xm4** (*int32xm4\_t* *merge*, *int32xm4\_t* *a*, short *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *int32xm8\_t* **vwaddwx\_mask\_int32xm8** (*int32xm8\_t* *merge*, *int32xm8\_t* *a*, short *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *int64xm2\_t* **vwaddwx\_mask\_int64xm2** (*int64xm2\_t* *merge*, *int64xm2\_t* *a*, int *b*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *int64xm4\_t* **vwaddwx\_mask\_int64xm4** (*int64xm4\_t* *merge*, *int64xm4\_t* *a*, int *b*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *int64xm8\_t* **vwaddwx\_mask\_int64xm8** (*int64xm8\_t* *merge*, *int64xm8\_t* *a*, int *b*, *e32xm4\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] + widen_integer (b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.98 Widening elementwise unsigned vector-vector addition

**Instruction:** ['vwaddu.vv']

**Prototypes:**

- *uint16xm2\_t* **vwadduvv\_uint16xm2\_uint8xm1** (*uint8xm1\_t* *a*, *uint8xm1\_t* *b*, unsigned int *gvl*)
- *uint16xm4\_t* **vwadduvv\_uint16xm4\_uint8xm2** (*uint8xm2\_t* *a*, *uint8xm2\_t* *b*, unsigned int *gvl*)
- *uint16xm8\_t* **vwadduvv\_uint16xm8\_uint8xm4** (*uint8xm4\_t* *a*, *uint8xm4\_t* *b*, unsigned int *gvl*)
- *uint32xm2\_t* **vwadduvv\_uint32xm2\_uint16xm1** (*uint16xm1\_t* *a*, *uint16xm1\_t* *b*, unsigned int *gvl*)
- *uint32xm4\_t* **vwadduvv\_uint32xm4\_uint16xm2** (*uint16xm2\_t* *a*, *uint16xm2\_t* *b*, unsigned int *gvl*)
- *uint32xm8\_t* **vwadduvv\_uint32xm8\_uint16xm4** (*uint16xm4\_t* *a*, *uint16xm4\_t* *b*, unsigned int *gvl*)
- *uint64xm2\_t* **vwadduvv\_uint64xm2\_uint32xm1** (*uint32xm1\_t* *a*, *uint32xm1\_t* *b*, unsigned int *gvl*)
- *uint64xm4\_t* **vwadduvv\_uint64xm4\_uint32xm2** (*uint32xm2\_t* *a*, *uint32xm2\_t* *b*, unsigned int *gvl*)
- *uint64xm8\_t* **vwadduvv\_uint64xm8\_uint32xm4** (*uint32xm4\_t* *a*, *uint32xm4\_t* *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = widen_integer (a[element]) + widen_integer (b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *uint16xm2\_t* **vwadduvv\_mask\_uint16xm2\_uint8xm1** (*uint16xm2\_t* *merge*, *uint8xm1\_t* *a*, *uint8xm1\_t* *b*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *uint16xm4\_t* **vwadduvv\_mask\_uint16xm4\_uint8xm2** (*uint16xm4\_t* *merge*, *uint8xm2\_t* *a*, *uint8xm2\_t* *b*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- *uint16xm8\_t* **vwadduvv\_mask\_uint16xm8\_uint8xm4** (*uint16xm8\_t* *merge*, *uint8xm4\_t* *a*, *uint8xm4\_t* *b*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- *uint32xm2\_t* **vwadduvv\_mask\_uint32xm2\_uint16xm1** (*uint32xm2\_t* *merge*, *uint16xm1\_t* *a*, *uint16xm1\_t* *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *uint32xm4\_t* **vwadduvv\_mask\_uint32xm4\_uint16xm2** (*uint32xm4\_t* *merge*, *uint16xm2\_t* *a*, *uint16xm2\_t* *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *uint32xm8\_t* **vwadduvv\_mask\_uint32xm8\_uint16xm4** (*uint32xm8\_t* *merge*, *uint16xm4\_t* *a*, *uint16xm4\_t* *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *uint64xm2\_t* **vwadduvv\_mask\_uint64xm2\_uint32xm1** (*uint64xm2\_t* *merge*, *uint32xm1\_t* *a*, *uint32xm1\_t* *b*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *uint64xm4\_t* **vwadduvv\_mask\_uint64xm4\_uint32xm2** (*uint64xm4\_t* *merge*, *uint32xm2\_t* *a*, *uint32xm2\_t* *b*, *e32xm2\_t* *mask*, unsigned int *gvl*)

- `uint64xm8_t vwadduvv_mask_uint64xm8_uint32xm4 (uint64xm8_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = widen_integer (a[element]) + widen_integer (b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.99 Widening elementwise unsigned vector-scalar addition

Instruction: ['vwaddu.vx']

Prototypes:

- `uint16xm2_t vwadduvx_uint16xm2_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint16xm4_t vwadduvx_uint16xm4_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint16xm8_t vwadduvx_uint16xm8_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint32xm2_t vwadduvx_uint32xm2_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint32xm4_t vwadduvx_uint32xm4_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint32xm8_t vwadduvx_uint32xm8_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint64xm2_t vwadduvx_uint64xm2_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint64xm4_t vwadduvx_uint64xm4_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint64xm8_t vwadduvx_uint64xm8_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`

Operation:

```
>>> for element = 0 to gvl - 1
    result[element] = widen_integer (a[element]) + widen_integer (b)
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `uint16xm2_t vwadduvx_mask_uint16xm2_uint8xm1 (uint16xm2_t merge, uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint16xm4_t vwadduvx_mask_uint16xm4_uint8xm2 (uint16xm4_t merge, uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint16xm8_t vwadduvx_mask_uint16xm8_uint8xm4 (uint16xm8_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint32xm2_t vwadduvx_mask_uint32xm2_uint16xm1 (uint32xm2_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`



- `uint32xm4_t vwadduvx_mask_uint32xm4_uint16xm2` (`uint32xm4_t` merge, `uint16xm2_t` *a*, unsigned short *b*, `e16xm2_t` mask, unsigned int *gvl*)
- `uint32xm8_t vwadduvx_mask_uint32xm8_uint16xm4` (`uint32xm8_t` merge, `uint16xm4_t` *a*, unsigned short *b*, `e16xm4_t` mask, unsigned int *gvl*)
- `uint64xm2_t vwadduvx_mask_uint64xm2_uint32xm1` (`uint64xm2_t` merge, `uint32xm1_t` *a*, unsigned int *b*, `e32xm1_t` mask, unsigned int *gvl*)
- `uint64xm4_t vwadduvx_mask_uint64xm4_uint32xm2` (`uint64xm4_t` merge, `uint32xm2_t` *a*, unsigned int *b*, `e32xm2_t` mask, unsigned int *gvl*)
- `uint64xm8_t vwadduvx_mask_uint64xm8_uint32xm4` (`uint64xm8_t` merge, `uint32xm4_t` *a*, unsigned int *b*, `e32xm4_t` mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = widen_integer (a[element]) + widen_integer (b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.100 Widening elementwise (Widening)unsigned vector-vector addition****Instruction:** ['vwaddu.wv']**Prototypes:**

- `uint16xm2_t vwadduwv_uint16xm2_uint8xm1` (`uint16xm2_t` *a*, `uint8xm1_t` *b*, unsigned int *gvl*)
- `uint16xm4_t vwadduwv_uint16xm4_uint8xm2` (`uint16xm4_t` *a*, `uint8xm2_t` *b*, unsigned int *gvl*)
- `uint16xm8_t vwadduwv_uint16xm8_uint8xm4` (`uint16xm8_t` *a*, `uint8xm4_t` *b*, unsigned int *gvl*)
- `uint32xm2_t vwadduwv_uint32xm2_uint16xm1` (`uint32xm2_t` *a*, `uint16xm1_t` *b*, unsigned int *gvl*)
- `uint32xm4_t vwadduwv_uint32xm4_uint16xm2` (`uint32xm4_t` *a*, `uint16xm2_t` *b*, unsigned int *gvl*)
- `uint32xm8_t vwadduwv_uint32xm8_uint16xm4` (`uint32xm8_t` *a*, `uint16xm4_t` *b*, unsigned int *gvl*)
- `uint64xm2_t vwadduwv_uint64xm2_uint32xm1` (`uint64xm2_t` *a*, `uint32xm1_t` *b*, unsigned int *gvl*)
- `uint64xm4_t vwadduwv_uint64xm4_uint32xm2` (`uint64xm4_t` *a*, `uint32xm2_t` *b*, unsigned int *gvl*)
- `uint64xm8_t vwadduwv_uint64xm8_uint32xm4` (`uint64xm8_t` *a*, `uint32xm4_t` *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + widen_integer (b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm2_t vwadduwv_mask_uint16xm2_uint8xm1` (`uint16xm2_t merge`, `uint16xm2_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint16xm4_t vwadduwv_mask_uint16xm4_uint8xm2` (`uint16xm4_t merge`, `uint16xm4_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint16xm8_t vwadduwv_mask_uint16xm8_uint8xm4` (`uint16xm8_t merge`, `uint16xm8_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint32xm2_t vwadduwv_mask_uint32xm2_uint16xm1` (`uint32xm2_t merge`, `uint32xm2_t a`, `uint16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint32xm4_t vwadduwv_mask_uint32xm4_uint16xm2` (`uint32xm4_t merge`, `uint32xm4_t a`, `uint16xm2_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint32xm8_t vwadduwv_mask_uint32xm8_uint16xm4` (`uint32xm8_t merge`, `uint32xm8_t a`, `uint16xm4_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint64xm2_t vwadduwv_mask_uint64xm2_uint32xm1` (`uint64xm2_t merge`, `uint64xm2_t a`, `uint32xm1_t b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint64xm4_t vwadduwv_mask_uint64xm4_uint32xm2` (`uint64xm4_t merge`, `uint64xm4_t a`, `uint32xm2_t b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint64xm8_t vwadduwv_mask_uint64xm8_uint32xm4` (`uint64xm8_t merge`, `uint64xm8_t a`, `uint32xm4_t b`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] + widen_integer (b[element])
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.7.101 Widening elementwise (Widening)unsigned vector-scalar addition****Instruction:** [`'vwaddu.wx'`]**Prototypes:**

- `uint16xm2_t vwadduwx_uint16xm2` (`uint16xm2_t a`, unsigned char `b`, unsigned int `gvl`)
- `uint16xm4_t vwadduwx_uint16xm4` (`uint16xm4_t a`, unsigned char `b`, unsigned int `gvl`)
- `uint16xm8_t vwadduwx_uint16xm8` (`uint16xm8_t a`, unsigned char `b`, unsigned int `gvl`)
- `uint32xm2_t vwadduwx_uint32xm2` (`uint32xm2_t a`, unsigned short `b`, unsigned int `gvl`)
- `uint32xm4_t vwadduwx_uint32xm4` (`uint32xm4_t a`, unsigned short `b`, unsigned int `gvl`)
- `uint32xm8_t vwadduwx_uint32xm8` (`uint32xm8_t a`, unsigned short `b`, unsigned int `gvl`)

- `uint64xm2_t vwadduwx_uint64xm2 (uint64xm2_t a, unsigned int b, unsigned int gvl)`
- `uint64xm4_t vwadduwx_uint64xm4 (uint64xm4_t a, unsigned int b, unsigned int gvl)`
- `uint64xm8_t vwadduwx_uint64xm8 (uint64xm8_t a, unsigned int b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] + widen_integer (b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm2_t vwadduwx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint16xm4_t vwadduwx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint16xm8_t vwadduwx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint32xm2_t vwadduwx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint32xm4_t vwadduwx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint32xm8_t vwadduwx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint64xm2_t vwadduwx_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint64xm4_t vwadduwx_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint64xm8_t vwadduwx_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] + widen_integer (b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.102 Widening elementwise signed vector-vector multiply-add, overwrite addend****Instruction:** ['vwmaccvv.vv']**Prototypes:**

- `int16xm2_t vwmaccvv_int16xm2_int8xm1 (int8xm1_t a, int8xm1_t b, int16xm2_t result, unsigned int gvl)`
- `int16xm4_t vwmaccvv_int16xm4_int8xm2 (int8xm2_t a, int8xm2_t b, int16xm4_t result, unsigned int gvl)`

- *int16xm8\_t vwmaccvv\_int16xm8\_int8xm4* (*int8xm4\_t a*, *int8xm4\_t b*, *int16xm8\_t result*, unsigned int gvl)
- *int32xm2\_t vwmaccvv\_int32xm2\_int16xm1* (*int16xm1\_t a*, *int16xm1\_t b*, *int32xm2\_t result*, unsigned int gvl)
- *int32xm4\_t vwmaccvv\_int32xm4\_int16xm2* (*int16xm2\_t a*, *int16xm2\_t b*, *int32xm4\_t result*, unsigned int gvl)
- *int32xm8\_t vwmaccvv\_int32xm8\_int16xm4* (*int16xm4\_t a*, *int16xm4\_t b*, *int32xm8\_t result*, unsigned int gvl)
- *int64xm2\_t vwmaccvv\_int64xm2\_int32xm1* (*int32xm1\_t a*, *int32xm1\_t b*, *int64xm2\_t result*, unsigned int gvl)
- *int64xm4\_t vwmaccvv\_int64xm4\_int32xm2* (*int32xm2\_t a*, *int32xm2\_t b*, *int64xm4\_t result*, unsigned int gvl)
- *int64xm8\_t vwmaccvv\_int64xm8\_int32xm4* (*int32xm4\_t a*, *int32xm4\_t b*, *int64xm8\_t result*, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = +widen_integer(a[element] * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm2\_t vwmaccvv\_mask\_int16xm2\_int8xm1* (*int16xm2\_t merge*, *int8xm1\_t a*, *int8xm1\_t b*, *int16xm2\_t result*, *e8xm1\_t mask*, unsigned int gvl)
- *int16xm4\_t vwmaccvv\_mask\_int16xm4\_int8xm2* (*int16xm4\_t merge*, *int8xm2\_t a*, *int8xm2\_t b*, *int16xm4\_t result*, *e8xm2\_t mask*, unsigned int gvl)
- *int16xm8\_t vwmaccvv\_mask\_int16xm8\_int8xm4* (*int16xm8\_t merge*, *int8xm4\_t a*, *int8xm4\_t b*, *int16xm8\_t result*, *e8xm4\_t mask*, unsigned int gvl)
- *int32xm2\_t vwmaccvv\_mask\_int32xm2\_int16xm1* (*int32xm2\_t merge*, *int16xm1\_t a*, *int16xm1\_t b*, *int32xm2\_t result*, *e16xm1\_t mask*, unsigned int gvl)
- *int32xm4\_t vwmaccvv\_mask\_int32xm4\_int16xm2* (*int32xm4\_t merge*, *int16xm2\_t a*, *int16xm2\_t b*, *int32xm4\_t result*, *e16xm2\_t mask*, unsigned int gvl)
- *int32xm8\_t vwmaccvv\_mask\_int32xm8\_int16xm4* (*int32xm8\_t merge*, *int16xm4\_t a*, *int16xm4\_t b*, *int32xm8\_t result*, *e16xm4\_t mask*, unsigned int gvl)
- *int64xm2\_t vwmaccvv\_mask\_int64xm2\_int32xm1* (*int64xm2\_t merge*, *int32xm1\_t a*, *int32xm1\_t b*, *int64xm2\_t result*, *e32xm1\_t mask*, unsigned int gvl)
- *int64xm4\_t vwmaccvv\_mask\_int64xm4\_int32xm2* (*int64xm4\_t merge*, *int32xm2\_t a*, *int32xm2\_t b*, *int64xm4\_t result*, *e32xm2\_t mask*, unsigned int gvl)
- *int64xm8\_t vwmaccvv\_mask\_int64xm8\_int32xm4* (*int64xm8\_t merge*, *int32xm4\_t a*, *int32xm4\_t b*, *int64xm8\_t result*, *e32xm4\_t mask*, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = +widen_integer(a[element] * b[element]) + result[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.103 Widening elementwise signed vector-scalar multiply-add, overwrite addend

**Instruction:** ['vwmaccvx']

**Prototypes:**

- *int16xm2\_t vwmaccvx\_int16xm2\_int8xm1* (signed char *a*, *int8xm1\_t b*, *int16xm2\_t result*, unsigned int *gvl*)
- *int16xm4\_t vwmaccvx\_int16xm4\_int8xm2* (signed char *a*, *int8xm2\_t b*, *int16xm4\_t result*, unsigned int *gvl*)
- *int16xm8\_t vwmaccvx\_int16xm8\_int8xm4* (signed char *a*, *int8xm4\_t b*, *int16xm8\_t result*, unsigned int *gvl*)
- *int32xm2\_t vwmaccvx\_int32xm2\_int16xm1* (short *a*, *int16xm1\_t b*, *int32xm2\_t result*, unsigned int *gvl*)
- *int32xm4\_t vwmaccvx\_int32xm4\_int16xm2* (short *a*, *int16xm2\_t b*, *int32xm4\_t result*, unsigned int *gvl*)
- *int32xm8\_t vwmaccvx\_int32xm8\_int16xm4* (short *a*, *int16xm4\_t b*, *int32xm8\_t result*, unsigned int *gvl*)
- *int64xm2\_t vwmaccvx\_int64xm2\_int32xm1* (int *a*, *int32xm1\_t b*, *int64xm2\_t result*, unsigned int *gvl*)
- *int64xm4\_t vwmaccvx\_int64xm4\_int32xm2* (int *a*, *int32xm2\_t b*, *int64xm4\_t result*, unsigned int *gvl*)
- *int64xm8\_t vwmaccvx\_int64xm8\_int32xm4* (int *a*, *int32xm4\_t b*, *int64xm8\_t result*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = +widen_integer(a * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm2\_t vwmaccvx\_mask\_int16xm2\_int8xm1* (*int16xm2\_t merge*, signed char *a*, *int8xm1\_t b*, *int16xm2\_t result*, *e8xm1\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vwmaccvx\_mask\_int16xm4\_int8xm2* (*int16xm4\_t merge*, signed char *a*, *int8xm2\_t b*, *int16xm4\_t result*, *e8xm2\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vwmaccvx\_mask\_int16xm8\_int8xm4* (*int16xm8\_t merge*, signed char *a*, *int8xm4\_t b*, *int16xm8\_t result*, *e8xm4\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vwmaccvx\_mask\_int32xm2\_int16xm1* (*int32xm2\_t merge*, short *a*, *int16xm1\_t b*, *int32xm2\_t result*, *e16xm1\_t mask*, unsigned int *gvl*)

- `int32xm4_t vwmaccvx_mask_int32xm4_int16xm2` (`int32xm4_t merge`, `short a`, `int16xm2_t b`, `int32xm4_t result`, `e16xm2_t mask`, unsigned `int gvl`)
- `int32xm8_t vwmaccvx_mask_int32xm8_int16xm4` (`int32xm8_t merge`, `short a`, `int16xm4_t b`, `int32xm8_t result`, `e16xm4_t mask`, unsigned `int gvl`)
- `int64xm2_t vwmaccvx_mask_int64xm2_int32xm1` (`int64xm2_t merge`, `int a`, `int32xm1_t b`, `int64xm2_t result`, `e32xm1_t mask`, unsigned `int gvl`)
- `int64xm4_t vwmaccvx_mask_int64xm4_int32xm2` (`int64xm4_t merge`, `int a`, `int32xm2_t b`, `int64xm4_t result`, `e32xm2_t mask`, unsigned `int gvl`)
- `int64xm8_t vwmaccvx_mask_int64xm8_int32xm4` (`int64xm8_t merge`, `int a`, `int32xm4_t b`, `int64xm8_t result`, `e32xm4_t mask`, unsigned `int gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = +widen_integer(a * b[element]) + result[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.104 Widening elementwise vector-vector signed-unsigned integer multiply-sub, overwrite addend****Instruction:** [`'vwmaccsu.vv'`]**Prototypes:**

- `int16xm2_t vwmaccsuvv_int16xm2_int8xm1_uint8xm1` (`int8xm1_t a`, `uint8xm1_t b`, `int16xm2_t result`, unsigned `int gvl`)
- `int16xm4_t vwmaccsuvv_int16xm4_int8xm2_uint8xm2` (`int8xm2_t a`, `uint8xm2_t b`, `int16xm4_t result`, unsigned `int gvl`)
- `int16xm8_t vwmaccsuvv_int16xm8_int8xm4_uint8xm4` (`int8xm4_t a`, `uint8xm4_t b`, `int16xm8_t result`, unsigned `int gvl`)
- `int32xm2_t vwmaccsuvv_int32xm2_int16xm1_uint16xm1` (`int16xm1_t a`, `uint16xm1_t b`, `int32xm2_t result`, unsigned `int gvl`)
- `int32xm4_t vwmaccsuvv_int32xm4_int16xm2_uint16xm2` (`int16xm2_t a`, `uint16xm2_t b`, `int32xm4_t result`, unsigned `int gvl`)
- `int32xm8_t vwmaccsuvv_int32xm8_int16xm4_uint16xm4` (`int16xm4_t a`, `uint16xm4_t b`, `int32xm8_t result`, unsigned `int gvl`)
- `int64xm2_t vwmaccsuvv_int64xm2_int32xm1_uint32xm1` (`int32xm1_t a`, `uint32xm1_t b`, `int64xm2_t result`, unsigned `int gvl`)



- `int64xm4_t vwmaccsuvv_int64xm4_int32xm2_uint32xm2 (int32xm2_t a, uint32xm2_t b, int64xm4_t result, unsigned int gvl)`
- `int64xm8_t vwmaccsuvv_int64xm8_int32xm4_uint32xm4 (int32xm4_t a, uint32xm4_t b, int64xm8_t result, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = +widen_integer (a[element] * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm2_t vwmaccsuvv_mask_int16xm2_int8xm1_uint8xm1 (int16xm2_t merge, int8xm1_t a, uint8xm1_t b, int16xm2_t result, e8xm1_t mask, unsigned int gvl)`
- `int16xm4_t vwmaccsuvv_mask_int16xm4_int8xm2_uint8xm2 (int16xm4_t merge, int8xm2_t a, uint8xm2_t b, int16xm4_t result, e8xm2_t mask, unsigned int gvl)`
- `int16xm8_t vwmaccsuvv_mask_int16xm8_int8xm4_uint8xm4 (int16xm8_t merge, int8xm4_t a, uint8xm4_t b, int16xm8_t result, e8xm4_t mask, unsigned int gvl)`
- `int32xm2_t vwmaccsuvv_mask_int32xm2_int16xm1_uint16xm1 (int32xm2_t merge, int16xm1_t a, uint16xm1_t b, int32xm2_t result, e16xm1_t mask, unsigned int gvl)`
- `int32xm4_t vwmaccsuvv_mask_int32xm4_int16xm2_uint16xm2 (int32xm4_t merge, int16xm2_t a, uint16xm2_t b, int32xm4_t result, e16xm2_t mask, unsigned int gvl)`
- `int32xm8_t vwmaccsuvv_mask_int32xm8_int16xm4_uint16xm4 (int32xm8_t merge, int16xm4_t a, uint16xm4_t b, int32xm8_t result, e16xm4_t mask, unsigned int gvl)`
- `int64xm2_t vwmaccsuvv_mask_int64xm2_int32xm1_uint32xm1 (int64xm2_t merge, int32xm1_t a, uint32xm1_t b, int64xm2_t result, e32xm1_t mask, unsigned int gvl)`

- `int64xm4_t vwmaccsuvv_mask_int64xm4_int32xm2_uint32xm2` (`int64xm4_t` *merge*,  
`int32xm2_t` *a*,  
`uint32xm2_t` *b*,  
`int64xm4_t` *result*,  
`e32xm2_t` *mask*, unsigned  
`int gvl`)
- `int64xm8_t vwmaccsuvv_mask_int64xm8_int32xm4_uint32xm4` (`int64xm8_t` *merge*,  
`int32xm4_t` *a*,  
`uint32xm4_t` *b*,  
`int64xm8_t` *result*,  
`e32xm4_t` *mask*, unsigned  
`int gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = +widen_integer (a[element] * b[element]) + result[element]
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

**2.7.105 Widening elementwise vector-scalar signed-unsigned integer multiply-sub, overwrite addend****Instruction:** [`'vwmaccsu.vx'`]**Prototypes:**

- `int16xm2_t vwmaccsuvx_int16xm2_uint8xm1` (signed char *a*, `uint8xm1_t` *b*, `int16xm2_t` *result*, unsigned int *gvl*)
- `int16xm4_t vwmaccsuvx_int16xm4_uint8xm2` (signed char *a*, `uint8xm2_t` *b*, `int16xm4_t` *result*, unsigned int *gvl*)
- `int16xm8_t vwmaccsuvx_int16xm8_uint8xm4` (signed char *a*, `uint8xm4_t` *b*, `int16xm8_t` *result*, unsigned int *gvl*)
- `int32xm2_t vwmaccsuvx_int32xm2_uint16xm1` (short *a*, `uint16xm1_t` *b*, `int32xm2_t` *result*, unsigned int *gvl*)
- `int32xm4_t vwmaccsuvx_int32xm4_uint16xm2` (short *a*, `uint16xm2_t` *b*, `int32xm4_t` *result*, unsigned int *gvl*)
- `int32xm8_t vwmaccsuvx_int32xm8_uint16xm4` (short *a*, `uint16xm4_t` *b*, `int32xm8_t` *result*, unsigned int *gvl*)
- `int64xm2_t vwmaccsuvx_int64xm2_uint32xm1` (int *a*, `uint32xm1_t` *b*, `int64xm2_t` *result*, unsigned int *gvl*)
- `int64xm4_t vwmaccsuvx_int64xm4_uint32xm2` (int *a*, `uint32xm2_t` *b*, `int64xm4_t` *result*, unsigned int *gvl*)
- `int64xm8_t vwmaccsuvx_int64xm8_uint32xm4` (int *a*, `uint32xm4_t` *b*, `int64xm8_t` *result*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = +widen_integer (a * b[element]) + result[element]
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm2_t vwmaccsuvx_mask_int16xm2_uint8xm1` (`int16xm2_t merge`, signed char `a`, `uint8xm1_t b`, `int16xm2_t result`, `e8xm1_t mask`, unsigned int `gvl`)
- `int16xm4_t vwmaccsuvx_mask_int16xm4_uint8xm2` (`int16xm4_t merge`, signed char `a`, `uint8xm2_t b`, `int16xm4_t result`, `e8xm2_t mask`, unsigned int `gvl`)
- `int16xm8_t vwmaccsuvx_mask_int16xm8_uint8xm4` (`int16xm8_t merge`, signed char `a`, `uint8xm4_t b`, `int16xm8_t result`, `e8xm4_t mask`, unsigned int `gvl`)
- `int32xm2_t vwmaccsuvx_mask_int32xm2_uint16xm1` (`int32xm2_t merge`, short `a`, `uint16xm1_t b`, `int32xm2_t result`, `e16xm1_t mask`, unsigned int `gvl`)
- `int32xm4_t vwmaccsuvx_mask_int32xm4_uint16xm2` (`int32xm4_t merge`, short `a`, `uint16xm2_t b`, `int32xm4_t result`, `e16xm2_t mask`, unsigned int `gvl`)
- `int32xm8_t vwmaccsuvx_mask_int32xm8_uint16xm4` (`int32xm8_t merge`, short `a`, `uint16xm4_t b`, `int32xm8_t result`, `e16xm4_t mask`, unsigned int `gvl`)
- `int64xm2_t vwmaccsuvx_mask_int64xm2_uint32xm1` (`int64xm2_t merge`, int `a`, `uint32xm1_t b`, `int64xm2_t result`, `e32xm1_t mask`, unsigned int `gvl`)
- `int64xm4_t vwmaccsuvx_mask_int64xm4_uint32xm2` (`int64xm4_t merge`, int `a`, `uint32xm2_t b`, `int64xm4_t result`, `e32xm2_t mask`, unsigned int `gvl`)
- `int64xm8_t vwmaccsuvx_mask_int64xm8_uint32xm4` (`int64xm8_t merge`, int `a`, `uint32xm4_t b`, `int64xm8_t result`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = +widen_integer (a * b[element]) + result[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.106 Widening elementwise unsigned vector-vector multiply-add, overwrite addend****Instruction:** [`'vwmaccuvv'`]**Prototypes:**

- `uint16xm2_t vwmaccuvv_uint16xm2_uint8xm1` (`uint8xm1_t a`, `uint8xm1_t b`, `uint16xm2_t result`, unsigned int `gvl`)
- `uint16xm4_t vwmaccuvv_uint16xm4_uint8xm2` (`uint8xm2_t a`, `uint8xm2_t b`, `uint16xm4_t result`, unsigned int `gvl`)
- `uint16xm8_t vwmaccuvv_uint16xm8_uint8xm4` (`uint8xm4_t a`, `uint8xm4_t b`, `uint16xm8_t result`, unsigned int `gvl`)

- `uint32xm2_t vwmaccuvv_uint32xm2_uint16xm1 (uint16xm1_t a, uint16xm1_t b, uint32xm2_t result, unsigned int gvl)`
- `uint32xm4_t vwmaccuvv_uint32xm4_uint16xm2 (uint16xm2_t a, uint16xm2_t b, uint32xm4_t result, unsigned int gvl)`
- `uint32xm8_t vwmaccuvv_uint32xm8_uint16xm4 (uint16xm4_t a, uint16xm4_t b, uint32xm8_t result, unsigned int gvl)`
- `uint64xm2_t vwmaccuvv_uint64xm2_uint32xm1 (uint32xm1_t a, uint32xm1_t b, uint64xm2_t result, unsigned int gvl)`
- `uint64xm4_t vwmaccuvv_uint64xm4_uint32xm2 (uint32xm2_t a, uint32xm2_t b, uint64xm4_t result, unsigned int gvl)`
- `uint64xm8_t vwmaccuvv_uint64xm8_uint32xm4 (uint32xm4_t a, uint32xm4_t b, uint64xm8_t result, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = +widen_integer(a[element] * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm2_t vwmaccuvv_mask_uint16xm2_uint8xm1 (uint16xm2_t merge, uint8xm1_t a, uint8xm1_t b, uint16xm2_t result, e8xm1_t mask, unsigned int gvl)`
- `uint16xm4_t vwmaccuvv_mask_uint16xm4_uint8xm2 (uint16xm4_t merge, uint8xm2_t a, uint8xm2_t b, uint16xm4_t result, e8xm2_t mask, unsigned int gvl)`
- `uint16xm8_t vwmaccuvv_mask_uint16xm8_uint8xm4 (uint16xm8_t merge, uint8xm4_t a, uint8xm4_t b, uint16xm8_t result, e8xm4_t mask, unsigned int gvl)`
- `uint32xm2_t vwmaccuvv_mask_uint32xm2_uint16xm1 (uint32xm2_t merge, uint16xm1_t a, uint16xm1_t b, uint32xm2_t result, e16xm1_t mask, unsigned int gvl)`
- `uint32xm4_t vwmaccuvv_mask_uint32xm4_uint16xm2 (uint32xm4_t merge, uint16xm2_t a, uint16xm2_t b, uint32xm4_t result, e16xm2_t mask, unsigned int gvl)`
- `uint32xm8_t vwmaccuvv_mask_uint32xm8_uint16xm4 (uint32xm8_t merge, uint16xm4_t a, uint16xm4_t b, uint32xm8_t result, e16xm4_t mask, unsigned int gvl)`
- `uint64xm2_t vwmaccuvv_mask_uint64xm2_uint32xm1 (uint64xm2_t merge, uint32xm1_t a, uint32xm1_t b, uint64xm2_t result, e32xm1_t mask, unsigned int gvl)`
- `uint64xm4_t vwmaccuvv_mask_uint64xm4_uint32xm2 (uint64xm4_t merge, uint32xm2_t a, uint32xm2_t b, uint64xm4_t result, e32xm2_t mask, unsigned int gvl)`
- `uint64xm8_t vwmaccuvv_mask_uint64xm8_uint32xm4 (uint64xm8_t merge, uint32xm4_t a, uint32xm4_t b, uint64xm8_t result, e32xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = +widen_integer(a[element] * b[element]) + result[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.107 Widening elementwise unsigned vector-scalar multiply-add, overwrite addend

**Instruction:** ['vwmaccu.vx']

**Prototypes:**

- *uint16xm2\_t* **vwmaccu.vx** *uint16xm2\_uint8xm1* (unsigned char *a*, *uint8xm1\_t* *b*, *uint16xm2\_t* *result*, unsigned int *gvl*)
- *uint16xm4\_t* **vwmaccu.vx** *uint16xm4\_uint8xm2* (unsigned char *a*, *uint8xm2\_t* *b*, *uint16xm4\_t* *result*, unsigned int *gvl*)
- *uint16xm8\_t* **vwmaccu.vx** *uint16xm8\_uint8xm4* (unsigned char *a*, *uint8xm4\_t* *b*, *uint16xm8\_t* *result*, unsigned int *gvl*)
- *uint32xm2\_t* **vwmaccu.vx** *uint32xm2\_uint16xm1* (unsigned short *a*, *uint16xm1\_t* *b*, *uint32xm2\_t* *result*, unsigned int *gvl*)
- *uint32xm4\_t* **vwmaccu.vx** *uint32xm4\_uint16xm2* (unsigned short *a*, *uint16xm2\_t* *b*, *uint32xm4\_t* *result*, unsigned int *gvl*)
- *uint32xm8\_t* **vwmaccu.vx** *uint32xm8\_uint16xm4* (unsigned short *a*, *uint16xm4\_t* *b*, *uint32xm8\_t* *result*, unsigned int *gvl*)
- *uint64xm2\_t* **vwmaccu.vx** *uint64xm2\_uint32xm1* (unsigned int *a*, *uint32xm1\_t* *b*, *uint64xm2\_t* *result*, unsigned int *gvl*)
- *uint64xm4\_t* **vwmaccu.vx** *uint64xm4\_uint32xm2* (unsigned int *a*, *uint32xm2\_t* *b*, *uint64xm4\_t* *result*, unsigned int *gvl*)
- *uint64xm8\_t* **vwmaccu.vx** *uint64xm8\_uint32xm4* (unsigned int *a*, *uint32xm4\_t* *b*, *uint64xm8\_t* *result*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = +widen_integer(a * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *uint16xm2\_t* **vwmaccu.vx\_mask** *uint16xm2\_uint8xm1* (*uint16xm2\_t* *merge*, unsigned char *a*, *uint8xm1\_t* *b*, *uint16xm2\_t* *result*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *uint16xm4\_t* **vwmaccu.vx\_mask** *uint16xm4\_uint8xm2* (*uint16xm4\_t* *merge*, unsigned char *a*, *uint8xm2\_t* *b*, *uint16xm4\_t* *result*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- *uint16xm8\_t* **vwmaccu.vx\_mask** *uint16xm8\_uint8xm4* (*uint16xm8\_t* *merge*, unsigned char *a*, *uint8xm4\_t* *b*, *uint16xm8\_t* *result*, *e8xm4\_t* *mask*, unsigned int *gvl*)

- `uint32xm2_t vwmaccuvx_mask_uint32xm2_uint16xm1` (`uint32xm2_t merge`, unsigned short `a`, `uint16xm1_t b`, `uint32xm2_t result`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint32xm4_t vwmaccuvx_mask_uint32xm4_uint16xm2` (`uint32xm4_t merge`, unsigned short `a`, `uint16xm2_t b`, `uint32xm4_t result`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint32xm8_t vwmaccuvx_mask_uint32xm8_uint16xm4` (`uint32xm8_t merge`, unsigned short `a`, `uint16xm4_t b`, `uint32xm8_t result`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint64xm2_t vwmaccuvx_mask_uint64xm2_uint32xm1` (`uint64xm2_t merge`, unsigned int `a`, `uint32xm1_t b`, `uint64xm2_t result`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint64xm4_t vwmaccuvx_mask_uint64xm4_uint32xm2` (`uint64xm4_t merge`, unsigned int `a`, `uint32xm2_t b`, `uint64xm4_t result`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint64xm8_t vwmaccuvx_mask_uint64xm8_uint32xm4` (`uint64xm8_t merge`, unsigned int `a`, `uint32xm4_t b`, `uint64xm8_t result`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = +widen_integer(a * b[element]) + result[element]
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.7.108 Widening elementwise vector-scalar unsigned-signed integer multiply-sub, overwrite addend****Instruction:** [`'vwmaccus.vx'`]**Prototypes:**

- `int16xm2_t vwmaccusvx_int16xm2_int8xm1` (unsigned char `a`, `int8xm1_t b`, `int16xm2_t result`, unsigned int `gvl`)
- `int16xm4_t vwmaccusvx_int16xm4_int8xm2` (unsigned char `a`, `int8xm2_t b`, `int16xm4_t result`, unsigned int `gvl`)
- `int16xm8_t vwmaccusvx_int16xm8_int8xm4` (unsigned char `a`, `int8xm4_t b`, `int16xm8_t result`, unsigned int `gvl`)
- `int32xm2_t vwmaccusvx_int32xm2_int16xm1` (unsigned short `a`, `int16xm1_t b`, `int32xm2_t result`, unsigned int `gvl`)
- `int32xm4_t vwmaccusvx_int32xm4_int16xm2` (unsigned short `a`, `int16xm2_t b`, `int32xm4_t result`, unsigned int `gvl`)
- `int32xm8_t vwmaccusvx_int32xm8_int16xm4` (unsigned short `a`, `int16xm4_t b`, `int32xm8_t result`, unsigned int `gvl`)
- `int64xm2_t vwmaccusvx_int64xm2_int32xm1` (unsigned int `a`, `int32xm1_t b`, `int64xm2_t result`, unsigned int `gvl`)
- `int64xm4_t vwmaccusvx_int64xm4_int32xm2` (unsigned int `a`, `int32xm2_t b`, `int64xm4_t result`, unsigned int `gvl`)



- `int64xm8_t vwmaccusvx_int64xm8_int32xm4` (unsigned int *a*, `int32xm4_t` *b*, `int64xm8_t` *result*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = +widen_integer (a * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm2_t vwmaccusvx_mask_int16xm2_int8xm1` (`int16xm2_t` *merge*, unsigned char *a*, `int8xm1_t` *b*, `int16xm2_t` *result*, `e8xm1_t` *mask*, unsigned int *gvl*)
- `int16xm4_t vwmaccusvx_mask_int16xm4_int8xm2` (`int16xm4_t` *merge*, unsigned char *a*, `int8xm2_t` *b*, `int16xm4_t` *result*, `e8xm2_t` *mask*, unsigned int *gvl*)
- `int16xm8_t vwmaccusvx_mask_int16xm8_int8xm4` (`int16xm8_t` *merge*, unsigned char *a*, `int8xm4_t` *b*, `int16xm8_t` *result*, `e8xm4_t` *mask*, unsigned int *gvl*)
- `int32xm2_t vwmaccusvx_mask_int32xm2_int16xm1` (`int32xm2_t` *merge*, unsigned short *a*, `int16xm1_t` *b*, `int32xm2_t` *result*, `e16xm1_t` *mask*, unsigned int *gvl*)
- `int32xm4_t vwmaccusvx_mask_int32xm4_int16xm2` (`int32xm4_t` *merge*, unsigned short *a*, `int16xm2_t` *b*, `int32xm4_t` *result*, `e16xm2_t` *mask*, unsigned int *gvl*)
- `int32xm8_t vwmaccusvx_mask_int32xm8_int16xm4` (`int32xm8_t` *merge*, unsigned short *a*, `int16xm4_t` *b*, `int32xm8_t` *result*, `e16xm4_t` *mask*, unsigned int *gvl*)
- `int64xm2_t vwmaccusvx_mask_int64xm2_int32xm1` (`int64xm2_t` *merge*, unsigned int *a*, `int32xm1_t` *b*, `int64xm2_t` *result*, `e32xm1_t` *mask*, unsigned int *gvl*)
- `int64xm4_t vwmaccusvx_mask_int64xm4_int32xm2` (`int64xm4_t` *merge*, unsigned int *a*, `int32xm2_t` *b*, `int64xm4_t` *result*, `e32xm2_t` *mask*, unsigned int *gvl*)
- `int64xm8_t vwmaccusvx_mask_int64xm8_int32xm4` (`int64xm8_t` *merge*, unsigned int *a*, `int32xm4_t` *b*, `int64xm8_t` *result*, `e32xm4_t` *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = +widen_integer (a * b[element]) + result[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.109 Widening elementwise signed vector-vector multiplication****Instruction:** [`'vwmul.vv'`]**Prototypes:**

- *int16xm2\_t vwmulvv\_int16xm2\_int8xm1* (*int8xm1\_t a*, *int8xm1\_t b*, unsigned int *gvl*)
- *int16xm4\_t vwmulvv\_int16xm4\_int8xm2* (*int8xm2\_t a*, *int8xm2\_t b*, unsigned int *gvl*)
- *int16xm8\_t vwmulvv\_int16xm8\_int8xm4* (*int8xm4\_t a*, *int8xm4\_t b*, unsigned int *gvl*)
- *int32xm2\_t vwmulvv\_int32xm2\_int16xm1* (*int16xm1\_t a*, *int16xm1\_t b*, unsigned int *gvl*)
- *int32xm4\_t vwmulvv\_int32xm4\_int16xm2* (*int16xm2\_t a*, *int16xm2\_t b*, unsigned int *gvl*)
- *int32xm8\_t vwmulvv\_int32xm8\_int16xm4* (*int16xm4\_t a*, *int16xm4\_t b*, unsigned int *gvl*)
- *int64xm2\_t vwmulvv\_int64xm2\_int32xm1* (*int32xm1\_t a*, *int32xm1\_t b*, unsigned int *gvl*)
- *int64xm4\_t vwmulvv\_int64xm4\_int32xm2* (*int32xm2\_t a*, *int32xm2\_t b*, unsigned int *gvl*)
- *int64xm8\_t vwmulvv\_int64xm8\_int32xm4* (*int32xm4\_t a*, *int32xm4\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = widen_integer (a[element]) * widen_integer (b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm2\_t vwmulvv\_mask\_int16xm2\_int8xm1* (*int16xm2\_t merge*, *int8xm1\_t a*, *int8xm1\_t b*, *e8xm1\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vwmulvv\_mask\_int16xm4\_int8xm2* (*int16xm4\_t merge*, *int8xm2\_t a*, *int8xm2\_t b*, *e8xm2\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vwmulvv\_mask\_int16xm8\_int8xm4* (*int16xm8\_t merge*, *int8xm4\_t a*, *int8xm4\_t b*, *e8xm4\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vwmulvv\_mask\_int32xm2\_int16xm1* (*int32xm2\_t merge*, *int16xm1\_t a*, *int16xm1\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vwmulvv\_mask\_int32xm4\_int16xm2* (*int32xm4\_t merge*, *int16xm2\_t a*, *int16xm2\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int32xm8\_t vwmulvv\_mask\_int32xm8\_int16xm4* (*int32xm8\_t merge*, *int16xm4\_t a*, *int16xm4\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int64xm2\_t vwmulvv\_mask\_int64xm2\_int32xm1* (*int64xm2\_t merge*, *int32xm1\_t a*, *int32xm1\_t b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *int64xm4\_t vwmulvv\_mask\_int64xm4\_int32xm2* (*int64xm4\_t merge*, *int32xm2\_t a*, *int32xm2\_t b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *int64xm8\_t vwmulvv\_mask\_int64xm8\_int32xm4* (*int64xm8\_t merge*, *int32xm4\_t a*, *int32xm4\_t b*, *e32xm4\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = widen_integer (a[element]) * widen_integer (b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.110 Widening elementwise signed vector-scalar multiplication

**Instruction:** [`'vwmul.vx'`]

**Prototypes:**

- `int16xm2_t vwmulvx_int16xm2_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `int16xm4_t vwmulvx_int16xm4_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `int16xm8_t vwmulvx_int16xm8_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `int32xm2_t vwmulvx_int32xm2_int16xm1 (int16xm1_t a, short b, unsigned int gvl)`
- `int32xm4_t vwmulvx_int32xm4_int16xm2 (int16xm2_t a, short b, unsigned int gvl)`
- `int32xm8_t vwmulvx_int32xm8_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `int64xm2_t vwmulvx_int64xm2_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `int64xm4_t vwmulvx_int64xm4_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `int64xm8_t vwmulvx_int64xm8_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = widen_integer (a[element]) * widen_integer (b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm2_t vwmulvx_mask_int16xm2_int8xm1 (int16xm2_t merge, int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl)`
- `int16xm4_t vwmulvx_mask_int16xm4_int8xm2 (int16xm4_t merge, int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl)`
- `int16xm8_t vwmulvx_mask_int16xm8_int8xm4 (int16xm8_t merge, int8xm4_t a, signed char b, e8xm4_t mask, unsigned int gvl)`
- `int32xm2_t vwmulvx_mask_int32xm2_int16xm1 (int32xm2_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int32xm4_t vwmulvx_mask_int32xm4_int16xm2 (int32xm4_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int32xm8_t vwmulvx_mask_int32xm8_int16xm4 (int32xm8_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int64xm2_t vwmulvx_mask_int64xm2_int32xm1 (int64xm2_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int64xm4_t vwmulvx_mask_int64xm4_int32xm2 (int64xm4_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int64xm8_t vwmulvx_mask_int64xm8_int32xm4 (int64xm8_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = widen_integer (a[element]) * widen_integer (b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.111 Widening elementwise vector-vector signed-unsigned integer multiplication

**Instruction:** ['vwmulsuvv.vv']

**Prototypes:**

- *int16xm2\_t vwmulsuvv\_int16xm2\_int8xm1\_uint8xm1* (*int8xm1\_t a*, *uint8xm1\_t b*, unsigned int gvl)
- *int16xm4\_t vwmulsuvv\_int16xm4\_int8xm2\_uint8xm2* (*int8xm2\_t a*, *uint8xm2\_t b*, unsigned int gvl)
- *int16xm8\_t vwmulsuvv\_int16xm8\_int8xm4\_uint8xm4* (*int8xm4\_t a*, *uint8xm4\_t b*, unsigned int gvl)
- *int32xm2\_t vwmulsuvv\_int32xm2\_int16xm1\_uint16xm1* (*int16xm1\_t a*, *uint16xm1\_t b*, unsigned int gvl)
- *int32xm4\_t vwmulsuvv\_int32xm4\_int16xm2\_uint16xm2* (*int16xm2\_t a*, *uint16xm2\_t b*, unsigned int gvl)
- *int32xm8\_t vwmulsuvv\_int32xm8\_int16xm4\_uint16xm4* (*int16xm4\_t a*, *uint16xm4\_t b*, unsigned int gvl)
- *int64xm2\_t vwmulsuvv\_int64xm2\_int32xm1\_uint32xm1* (*int32xm1\_t a*, *uint32xm1\_t b*, unsigned int gvl)
- *int64xm4\_t vwmulsuvv\_int64xm4\_int32xm2\_uint32xm2* (*int32xm2\_t a*, *uint32xm2\_t b*, unsigned int gvl)
- *int64xm8\_t vwmulsuvv\_int64xm8\_int32xm4\_uint32xm4* (*int32xm4\_t a*, *uint32xm4\_t b*, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = widen_integer (a[element] * b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm2\_t vwmulsuvv\_mask\_int16xm2\_int8xm1\_uint8xm1* (*int16xm2\_t merge*, *int8xm1\_t a*, *uint8xm1\_t b*, *e8xm1\_t mask*, unsigned int gvl)
- *int16xm4\_t vwmulsuvv\_mask\_int16xm4\_int8xm2\_uint8xm2* (*int16xm4\_t merge*, *int8xm2\_t a*, *uint8xm2\_t b*, *e8xm2\_t mask*, unsigned int gvl)
- *int16xm8\_t vwmulsuvv\_mask\_int16xm8\_int8xm4\_uint8xm4* (*int16xm8\_t merge*, *int8xm4\_t a*, *uint8xm4\_t b*, *e8xm4\_t mask*, unsigned int gvl)
- *int32xm2\_t vwmulsuvv\_mask\_int32xm2\_int16xm1\_uint16xm1* (*int32xm2\_t merge*, *int16xm1\_t a*, *uint16xm1\_t b*, *e16xm1\_t mask*, unsigned int gvl)
- *int32xm4\_t vwmulsuvv\_mask\_int32xm4\_int16xm2\_uint16xm2* (*int32xm4\_t merge*, *int16xm2\_t a*, *uint16xm2\_t b*, *e16xm2\_t mask*, unsigned int gvl)

- `int32xm8_t vwmulsuvv_mask_int32xm8_int16xm4_uint16xm4 (int32xm8_t merge, int16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `int64xm2_t vwmulsuvv_mask_int64xm2_int32xm1_uint32xm1 (int64xm2_t merge, int32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `int64xm4_t vwmulsuvv_mask_int64xm4_int32xm2_uint32xm2 (int64xm4_t merge, int32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `int64xm8_t vwmulsuvv_mask_int64xm8_int32xm4_uint32xm4 (int64xm8_t merge, int32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = widen_integer (a[element] * b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.112 Widening elementwise vector-scalar signed-unsigned integer multiplication****Instruction:** [`'vwmulsu.vx'`]**Prototypes:**

- `int16xm2_t vwmulsuvx_int16xm2_int8xm1 (int8xm1_t a, unsigned char b, unsigned int gvl)`
- `int16xm4_t vwmulsuvx_int16xm4_int8xm2 (int8xm2_t a, unsigned char b, unsigned int gvl)`
- `int16xm8_t vwmulsuvx_int16xm8_int8xm4 (int8xm4_t a, unsigned char b, unsigned int gvl)`
- `int32xm2_t vwmulsuvx_int32xm2_int16xm1 (int16xm1_t a, unsigned short b, unsigned int gvl)`
- `int32xm4_t vwmulsuvx_int32xm4_int16xm2 (int16xm2_t a, unsigned short b, unsigned int gvl)`
- `int32xm8_t vwmulsuvx_int32xm8_int16xm4 (int16xm4_t a, unsigned short b, unsigned int gvl)`
- `int64xm2_t vwmulsuvx_int64xm2_int32xm1 (int32xm1_t a, unsigned int b, unsigned int gvl)`
- `int64xm4_t vwmulsuvx_int64xm4_int32xm2 (int32xm2_t a, unsigned int b, unsigned int gvl)`
- `int64xm8_t vwmulsuvx_int64xm8_int32xm4 (int32xm4_t a, unsigned int b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = widen_integer (a[element] * b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm2_t vwmulsuvx_mask_int16xm2_int8xm1` (`int16xm2_t merge`, `int8xm1_t a`, unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `int16xm4_t vwmulsuvx_mask_int16xm4_int8xm2` (`int16xm4_t merge`, `int8xm2_t a`, unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `int16xm8_t vwmulsuvx_mask_int16xm8_int8xm4` (`int16xm8_t merge`, `int8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int32xm2_t vwmulsuvx_mask_int32xm2_int16xm1` (`int32xm2_t merge`, `int16xm1_t a`, unsigned short `b`, `e16xm1_t mask`, unsigned int `gvl`)
- `int32xm4_t vwmulsuvx_mask_int32xm4_int16xm2` (`int32xm4_t merge`, `int16xm2_t a`, unsigned short `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `int32xm8_t vwmulsuvx_mask_int32xm8_int16xm4` (`int32xm8_t merge`, `int16xm4_t a`, unsigned short `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `int64xm2_t vwmulsuvx_mask_int64xm2_int32xm1` (`int64xm2_t merge`, `int32xm1_t a`, unsigned int `b`, `e32xm1_t mask`, unsigned int `gvl`)
- `int64xm4_t vwmulsuvx_mask_int64xm4_int32xm2` (`int64xm4_t merge`, `int32xm2_t a`, unsigned int `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `int64xm8_t vwmulsuvx_mask_int64xm8_int32xm4` (`int64xm8_t merge`, `int32xm4_t a`, unsigned int `b`, `e32xm4_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = widen_integer (a[element] * b)
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.7.113 Widening elementwise unsigned vector-vector multiplition

Instruction: [`'vwmulu.vv'`]

Prototypes:

- `uint16xm2_t vwmuluvv_uint16xm2_uint8xm1` (`uint8xm1_t a`, `uint8xm1_t b`, unsigned int `gvl`)
- `uint16xm4_t vwmuluvv_uint16xm4_uint8xm2` (`uint8xm2_t a`, `uint8xm2_t b`, unsigned int `gvl`)
- `uint16xm8_t vwmuluvv_uint16xm8_uint8xm4` (`uint8xm4_t a`, `uint8xm4_t b`, unsigned int `gvl`)
- `uint32xm2_t vwmuluvv_uint32xm2_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, unsigned int `gvl`)
- `uint32xm4_t vwmuluvv_uint32xm4_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, unsigned int `gvl`)
- `uint32xm8_t vwmuluvv_uint32xm8_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, unsigned int `gvl`)
- `uint64xm2_t vwmuluvv_uint64xm2_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, unsigned int `gvl`)
- `uint64xm4_t vwmuluvv_uint64xm4_uint32xm2` (`uint32xm2_t a`, `uint32xm2_t b`, unsigned int `gvl`)
- `uint64xm8_t vwmuluvv_uint64xm8_uint32xm4` (`uint32xm4_t a`, `uint32xm4_t b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = widen_integer (a[element]) * widen_integer (b[element])
    result[gvl : VLMAX] = 0
```



**Masked prototypes:**

- `uint16xm2_t vwmuluvv_mask_uint16xm2_uint8xm1 (uint16xm2_t merge, uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint16xm4_t vwmuluvv_mask_uint16xm4_uint8xm2 (uint16xm4_t merge, uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint16xm8_t vwmuluvv_mask_uint16xm8_uint8xm4 (uint16xm8_t merge, uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint32xm2_t vwmuluvv_mask_uint32xm2_uint16xm1 (uint32xm2_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint32xm4_t vwmuluvv_mask_uint32xm4_uint16xm2 (uint32xm4_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint32xm8_t vwmuluvv_mask_uint32xm8_uint16xm4 (uint32xm8_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint64xm2_t vwmuluvv_mask_uint64xm2_uint32xm1 (uint64xm2_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint64xm4_t vwmuluvv_mask_uint64xm4_uint32xm2 (uint64xm4_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint64xm8_t vwmuluvv_mask_uint64xm8_uint32xm4 (uint64xm8_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = widen_integer (a[element]) * widen_integer (b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.114 Widening elementwise unsigned vector-scalar multiplition****Instruction:** [`'vwmulu.vx'`]**Prototypes:**

- `uint16xm2_t vwmuluvx_uint16xm2_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint16xm4_t vwmuluvx_uint16xm4_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint16xm8_t vwmuluvx_uint16xm8_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint32xm2_t vwmuluvx_uint32xm2_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`

- `uint32xm4_t vwmuluvx_uint32xm4_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint32xm8_t vwmuluvx_uint32xm8_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint64xm2_t vwmuluvx_uint64xm2_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint64xm4_t vwmuluvx_uint64xm4_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint64xm8_t vwmuluvx_uint64xm8_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = widen_integer (a[element]) * widen_integer (b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm2_t vwmuluvx_mask_uint16xm2_uint8xm1 (uint16xm2_t merge, uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint16xm4_t vwmuluvx_mask_uint16xm4_uint8xm2 (uint16xm4_t merge, uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint16xm8_t vwmuluvx_mask_uint16xm8_uint8xm4 (uint16xm8_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint32xm2_t vwmuluvx_mask_uint32xm2_uint16xm1 (uint32xm2_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint32xm4_t vwmuluvx_mask_uint32xm4_uint16xm2 (uint32xm4_t merge, uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint32xm8_t vwmuluvx_mask_uint32xm8_uint16xm4 (uint32xm8_t merge, uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint64xm2_t vwmuluvx_mask_uint64xm2_uint32xm1 (uint64xm2_t merge, uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint64xm4_t vwmuluvx_mask_uint64xm4_uint32xm2 (uint64xm4_t merge, uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint64xm8_t vwmuluvx_mask_uint64xm8_uint32xm4 (uint64xm8_t merge, uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = widen_integer (a[element]) * widen_integer (b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.115 Widening signed integer vector sum reduction

**Instruction:** ['vwredsum.vs']

**Prototypes:**

- *int16xm2\_t* **vwredsumvs\_int16xm2\_int8xm1** (*int8xm1\_t* a, *int16xm2\_t* b, unsigned int gvl)
- *int16xm4\_t* **vwredsumvs\_int16xm4\_int8xm2** (*int8xm2\_t* a, *int16xm4\_t* b, unsigned int gvl)
- *int16xm8\_t* **vwredsumvs\_int16xm8\_int8xm4** (*int8xm4\_t* a, *int16xm8\_t* b, unsigned int gvl)
- *int32xm2\_t* **vwredsumvs\_int32xm2\_int16xm1** (*int16xm1\_t* a, *int32xm2\_t* b, unsigned int gvl)
- *int32xm4\_t* **vwredsumvs\_int32xm4\_int16xm2** (*int16xm2\_t* a, *int32xm4\_t* b, unsigned int gvl)
- *int32xm8\_t* **vwredsumvs\_int32xm8\_int16xm4** (*int16xm4\_t* a, *int32xm8\_t* b, unsigned int gvl)
- *int64xm2\_t* **vwredsumvs\_int64xm2\_int32xm1** (*int32xm1\_t* a, *int64xm2\_t* b, unsigned int gvl)
- *int64xm4\_t* **vwredsumvs\_int64xm4\_int32xm2** (*int32xm2\_t* a, *int64xm4\_t* b, unsigned int gvl)
- *int64xm8\_t* **vwredsumvs\_int64xm8\_int32xm4** (*int32xm4\_t* a, *int64xm8\_t* b, unsigned int gvl)

**Operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        current_red = sum (current_red, widen_integer (a[element]))
    result[0] = current_red
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm2\_t* **vwredsumvs\_mask\_int16xm2\_int8xm1** (*int16xm2\_t* merge, *int8xm1\_t* a, *int16xm2\_t* b, *e8xm1\_t* mask, unsigned int gvl)
- *int16xm4\_t* **vwredsumvs\_mask\_int16xm4\_int8xm2** (*int16xm4\_t* merge, *int8xm2\_t* a, *int16xm4\_t* b, *e8xm2\_t* mask, unsigned int gvl)
- *int16xm8\_t* **vwredsumvs\_mask\_int16xm8\_int8xm4** (*int16xm8\_t* merge, *int8xm4\_t* a, *int16xm8\_t* b, *e8xm4\_t* mask, unsigned int gvl)
- *int32xm2\_t* **vwredsumvs\_mask\_int32xm2\_int16xm1** (*int32xm2\_t* merge, *int16xm1\_t* a, *int32xm2\_t* b, *e16xm1\_t* mask, unsigned int gvl)
- *int32xm4\_t* **vwredsumvs\_mask\_int32xm4\_int16xm2** (*int32xm4\_t* merge, *int16xm2\_t* a, *int32xm4\_t* b, *e16xm2\_t* mask, unsigned int gvl)
- *int32xm8\_t* **vwredsumvs\_mask\_int32xm8\_int16xm4** (*int32xm8\_t* merge, *int16xm4\_t* a, *int32xm8\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *int64xm2\_t* **vwredsumvs\_mask\_int64xm2\_int32xm1** (*int64xm2\_t* merge, *int32xm1\_t* a, *int64xm2\_t* b, *e32xm1\_t* mask, unsigned int gvl)
- *int64xm4\_t* **vwredsumvs\_mask\_int64xm4\_int32xm2** (*int64xm4\_t* merge, *int32xm2\_t* a, *int64xm4\_t* b, *e32xm2\_t* mask, unsigned int gvl)

- `int64xm8_t vwredsumvs_mask_int64xm8_int32xm4` (`int64xm8_t merge`, `int32xm4_t a`, `int64xm8_t b`, `e32xm4_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        if mask[element] then
            current_red = sum (current_red, widen_integer (a[element]))
        else
            result[element] = merge[element]
    result[0] = current_red
    result[1 : VLMAX] = 0
```

## 2.7.116 Widening unsigned integer vector sum reduction

Instruction: ['vwredsumu.vs']

Prototypes:

- `uint16xm2_t vwredsumuvs_uint16xm2_uint8xm1` (`uint8xm1_t a`, `uint16xm2_t b`, unsigned int `gvl`)
- `uint16xm4_t vwredsumuvs_uint16xm4_uint8xm2` (`uint8xm2_t a`, `uint16xm4_t b`, unsigned int `gvl`)
- `uint16xm8_t vwredsumuvs_uint16xm8_uint8xm4` (`uint8xm4_t a`, `uint16xm8_t b`, unsigned int `gvl`)
- `uint32xm2_t vwredsumuvs_uint32xm2_uint16xm1` (`uint16xm1_t a`, `uint32xm2_t b`, unsigned int `gvl`)
- `uint32xm4_t vwredsumuvs_uint32xm4_uint16xm2` (`uint16xm2_t a`, `uint32xm4_t b`, unsigned int `gvl`)
- `uint32xm8_t vwredsumuvs_uint32xm8_uint16xm4` (`uint16xm4_t a`, `uint32xm8_t b`, unsigned int `gvl`)
- `uint64xm2_t vwredsumuvs_uint64xm2_uint32xm1` (`uint32xm1_t a`, `uint64xm2_t b`, unsigned int `gvl`)
- `uint64xm4_t vwredsumuvs_uint64xm4_uint32xm2` (`uint32xm2_t a`, `uint64xm4_t b`, unsigned int `gvl`)
- `uint64xm8_t vwredsumuvs_uint64xm8_uint32xm4` (`uint32xm4_t a`, `uint64xm8_t b`, unsigned int `gvl`)

Operation:

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        current_red = sum (current_red, widen_integer (a[element]))
    result[0] = current_red
    result[gvl : VLMAX] = 0
```

Masked prototypes:

- `uint16xm2_t vwredsumuvs_mask_uint16xm2_uint8xm1` (`uint16xm2_t merge`, `uint8xm1_t a`, `uint16xm2_t b`, `e8xm1_t mask`, unsigned int `gvl`)

- `uint16xm4_t vwredsumuvs_mask_uint16xm4_uint8xm2` (`uint16xm4_t merge`, `uint8xm2_t a`, `uint16xm4_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint16xm8_t vwredsumuvs_mask_uint16xm8_uint8xm4` (`uint16xm8_t merge`, `uint8xm4_t a`, `uint16xm8_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint32xm2_t vwredsumuvs_mask_uint32xm2_uint16xm1` (`uint32xm2_t merge`, `uint16xm1_t a`, `uint32xm2_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint32xm4_t vwredsumuvs_mask_uint32xm4_uint16xm2` (`uint32xm4_t merge`, `uint16xm2_t a`, `uint32xm4_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint32xm8_t vwredsumuvs_mask_uint32xm8_uint16xm4` (`uint32xm8_t merge`, `uint16xm4_t a`, `uint32xm8_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint64xm2_t vwredsumuvs_mask_uint64xm2_uint32xm1` (`uint64xm2_t merge`, `uint32xm1_t a`, `uint64xm2_t b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint64xm4_t vwredsumuvs_mask_uint64xm4_uint32xm2` (`uint64xm4_t merge`, `uint32xm2_t a`, `uint64xm4_t b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint64xm8_t vwredsumuvs_mask_uint64xm8_uint32xm4` (`uint64xm8_t merge`, `uint32xm4_t a`, `uint64xm8_t b`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> if gvl > 0:
    current_red = b[0]
    for element = 0 to gvl - 1
        if mask[element] then
            current_red = sum (current_red, widen_integer (a[element]))
        else
            result[element] = merge[element]
    result[0] = current_red
    result[1 : VLMAX] = 0
```

**2.7.117 Widening elementwise signed vector-vector multiply-add, overwrite addend, with round and saturation****Instruction:** ['vwsmacc.vv']**Prototypes:**

- `int16xm2_t vwsmaccvv_int16xm2_int8xm1` (`int8xm1_t a`, `int8xm1_t b`, `int16xm2_t result`, unsigned int `gvl`)
- `int16xm4_t vwsmaccvv_int16xm4_int8xm2` (`int8xm2_t a`, `int8xm2_t b`, `int16xm4_t result`, unsigned int `gvl`)
- `int16xm8_t vwsmaccvv_int16xm8_int8xm4` (`int8xm4_t a`, `int8xm4_t b`, `int16xm8_t result`, unsigned int `gvl`)
- `int32xm2_t vwsmaccvv_int32xm2_int16xm1` (`int16xm1_t a`, `int16xm1_t b`, `int32xm2_t result`, unsigned int `gvl`)

- `int32xm4_t vwsmaaccvv_int32xm4_int16xm2` (`int16xm2_t a`, `int16xm2_t b`, `int32xm4_t result`, unsigned int `gvl`)
- `int32xm8_t vwsmaaccvv_int32xm8_int16xm4` (`int16xm4_t a`, `int16xm4_t b`, `int32xm8_t result`, unsigned int `gvl`)
- `int64xm2_t vwsmaaccvv_int64xm2_int32xm1` (`int32xm1_t a`, `int32xm1_t b`, `int64xm2_t result`, unsigned int `gvl`)
- `int64xm4_t vwsmaaccvv_int64xm4_int32xm2` (`int32xm2_t a`, `int32xm2_t b`, `int64xm4_t result`, unsigned int `gvl`)
- `int64xm8_t vwsmaaccvv_int64xm8_int32xm4` (`int32xm4_t a`, `int32xm4_t b`, `int64xm8_t result`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = clip(((a[element] * c[element] + round) >> (sew / 2)) +
↪result[element])
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm2_t vwsmaaccvv_mask_int16xm2_int8xm1` (`int16xm2_t merge`, `int8xm1_t a`, `int8xm1_t b`, `int16xm2_t result`, `e8xm1_t mask`, unsigned int `gvl`)
- `int16xm4_t vwsmaaccvv_mask_int16xm4_int8xm2` (`int16xm4_t merge`, `int8xm2_t a`, `int8xm2_t b`, `int16xm4_t result`, `e8xm2_t mask`, unsigned int `gvl`)
- `int16xm8_t vwsmaaccvv_mask_int16xm8_int8xm4` (`int16xm8_t merge`, `int8xm4_t a`, `int8xm4_t b`, `int16xm8_t result`, `e8xm4_t mask`, unsigned int `gvl`)
- `int32xm2_t vwsmaaccvv_mask_int32xm2_int16xm1` (`int32xm2_t merge`, `int16xm1_t a`, `int16xm1_t b`, `int32xm2_t result`, `e16xm1_t mask`, unsigned int `gvl`)
- `int32xm4_t vwsmaaccvv_mask_int32xm4_int16xm2` (`int32xm4_t merge`, `int16xm2_t a`, `int16xm2_t b`, `int32xm4_t result`, `e16xm2_t mask`, unsigned int `gvl`)
- `int32xm8_t vwsmaaccvv_mask_int32xm8_int16xm4` (`int32xm8_t merge`, `int16xm4_t a`, `int16xm4_t b`, `int32xm8_t result`, `e16xm4_t mask`, unsigned int `gvl`)
- `int64xm2_t vwsmaaccvv_mask_int64xm2_int32xm1` (`int64xm2_t merge`, `int32xm1_t a`, `int32xm1_t b`, `int64xm2_t result`, `e32xm1_t mask`, unsigned int `gvl`)
- `int64xm4_t vwsmaaccvv_mask_int64xm4_int32xm2` (`int64xm4_t merge`, `int32xm2_t a`, `int32xm2_t b`, `int64xm4_t result`, `e32xm2_t mask`, unsigned int `gvl`)
- `int64xm8_t vwsmaaccvv_mask_int64xm8_int32xm4` (`int64xm8_t merge`, `int32xm4_t a`, `int32xm4_t b`, `int64xm8_t result`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**



```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = clip(((a[element] * c[element] + round) >> (sew / 2)) +
    ↪ result[element])
    else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.7.118 Widening elementwise signed vector-scalar multiply-add, overwrite addend, with round and saturation

**Instruction:** ['vwsmacc.vx']

**Prototypes:**

- *int16xm2\_t vwsmaccvx\_int16xm2\_int8xm1* (signed char *a*, *int8xm1\_t b*, *int16xm2\_t result*, unsigned int *gvl*)
- *int16xm4\_t vwsmaccvx\_int16xm4\_int8xm2* (signed char *a*, *int8xm2\_t b*, *int16xm4\_t result*, unsigned int *gvl*)
- *int16xm8\_t vwsmaccvx\_int16xm8\_int8xm4* (signed char *a*, *int8xm4\_t b*, *int16xm8\_t result*, unsigned int *gvl*)
- *int32xm2\_t vwsmaccvx\_int32xm2\_int16xm1* (short *a*, *int16xm1\_t b*, *int32xm2\_t result*, unsigned int *gvl*)
- *int32xm4\_t vwsmaccvx\_int32xm4\_int16xm2* (short *a*, *int16xm2\_t b*, *int32xm4\_t result*, unsigned int *gvl*)
- *int32xm8\_t vwsmaccvx\_int32xm8\_int16xm4* (short *a*, *int16xm4\_t b*, *int32xm8\_t result*, unsigned int *gvl*)
- *int64xm2\_t vwsmaccvx\_int64xm2\_int32xm1* (int *a*, *int32xm1\_t b*, *int64xm2\_t result*, unsigned int *gvl*)
- *int64xm4\_t vwsmaccvx\_int64xm4\_int32xm2* (int *a*, *int32xm2\_t b*, *int64xm4\_t result*, unsigned int *gvl*)
- *int64xm8\_t vwsmaccvx\_int64xm8\_int32xm4* (int *a*, *int32xm4\_t b*, *int64xm8\_t result*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = +widen_integer(a * b[element]) + result[element]
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm2\_t vwsmaccvx\_mask\_int16xm2\_int8xm1* (*int16xm2\_t merge*, signed char *a*, *int8xm1\_t b*, *int16xm2\_t result*, *e8xm1\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vwsmaccvx\_mask\_int16xm4\_int8xm2* (*int16xm4\_t merge*, signed char *a*, *int8xm2\_t b*, *int16xm4\_t result*, *e8xm2\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vwsmaccvx\_mask\_int16xm8\_int8xm4* (*int16xm8\_t merge*, signed char *a*, *int8xm4\_t b*, *int16xm8\_t result*, *e8xm4\_t mask*, unsigned int *gvl*)

- `int32xm2_t vwsmaaccvx_mask_int32xm2_int16xm1` (`int32xm2_t merge`, short `a`, `int16xm1_t b`, `int32xm2_t result`, `e16xm1_t mask`, unsigned int `gvl`)
- `int32xm4_t vwsmaaccvx_mask_int32xm4_int16xm2` (`int32xm4_t merge`, short `a`, `int16xm2_t b`, `int32xm4_t result`, `e16xm2_t mask`, unsigned int `gvl`)
- `int32xm8_t vwsmaaccvx_mask_int32xm8_int16xm4` (`int32xm8_t merge`, short `a`, `int16xm4_t b`, `int32xm8_t result`, `e16xm4_t mask`, unsigned int `gvl`)
- `int64xm2_t vwsmaaccvx_mask_int64xm2_int32xm1` (`int64xm2_t merge`, int `a`, `int32xm1_t b`, `int64xm2_t result`, `e32xm1_t mask`, unsigned int `gvl`)
- `int64xm4_t vwsmaaccvx_mask_int64xm4_int32xm2` (`int64xm4_t merge`, int `a`, `int32xm2_t b`, `int64xm4_t result`, `e32xm2_t mask`, unsigned int `gvl`)
- `int64xm8_t vwsmaaccvx_mask_int64xm8_int32xm4` (`int64xm8_t merge`, int `a`, `int32xm4_t b`, `int64xm8_t result`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = +widen_integer(a * b[element]) + result[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.119 Widening elementwise vector-vector signed-unsigned integer multiply-sub, overwrite addend, with round and saturation****Instruction:** [`'vwsmaaccsu.vv'`]**Prototypes:**

- `int16xm2_t vwsmaaccsuvv_int16xm2_int8xm1_uint8xm1` (`int8xm1_t a`, `uint8xm1_t b`, `int16xm2_t result`, unsigned int `gvl`)
- `int16xm4_t vwsmaaccsuvv_int16xm4_int8xm2_uint8xm2` (`int8xm2_t a`, `uint8xm2_t b`, `int16xm4_t result`, unsigned int `gvl`)
- `int16xm8_t vwsmaaccsuvv_int16xm8_int8xm4_uint8xm4` (`int8xm4_t a`, `uint8xm4_t b`, `int16xm8_t result`, unsigned int `gvl`)
- `int32xm2_t vwsmaaccsuvv_int32xm2_int16xm1_uint16xm1` (`int16xm1_t a`, `uint16xm1_t b`, `int32xm2_t result`, unsigned int `gvl`)
- `int32xm4_t vwsmaaccsuvv_int32xm4_int16xm2_uint16xm2` (`int16xm2_t a`, `uint16xm2_t b`, `int32xm4_t result`, unsigned int `gvl`)
- `int32xm8_t vwsmaaccsuvv_int32xm8_int16xm4_uint16xm4` (`int16xm4_t a`, `uint16xm4_t b`, `int32xm8_t result`, unsigned int `gvl`)

- `int64xm2_t vwsmaaccsuvv_int64xm2_int32xm1_uint32xm1` (`int32xm1_t a`, `uint32xm1_t b`, `int64xm2_t result`, unsigned int `gvl`)
- `int64xm4_t vwsmaaccsuvv_int64xm4_int32xm2_uint32xm2` (`int32xm2_t a`, `uint32xm2_t b`, `int64xm4_t result`, unsigned int `gvl`)
- `int64xm8_t vwsmaaccsuvv_int64xm8_int32xm4_uint32xm4` (`int32xm4_t a`, `uint32xm4_t b`, `int64xm8_t result`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = clip(((a[element] * b[elemet] + round) >> (sew / 2)) +
↪ result[element])
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm2_t vwsmaaccsuvv_mask_int16xm2_int8xm1_uint8xm1` (`int16xm2_t merge`, `int8xm1_t a`, `uint8xm1_t b`, `int16xm2_t result`, `e8xm1_t mask`, unsigned int `gvl`)
- `int16xm4_t vwsmaaccsuvv_mask_int16xm4_int8xm2_uint8xm2` (`int16xm4_t merge`, `int8xm2_t a`, `uint8xm2_t b`, `int16xm4_t result`, `e8xm2_t mask`, unsigned int `gvl`)
- `int16xm8_t vwsmaaccsuvv_mask_int16xm8_int8xm4_uint8xm4` (`int16xm8_t merge`, `int8xm4_t a`, `uint8xm4_t b`, `int16xm8_t result`, `e8xm4_t mask`, unsigned int `gvl`)
- `int32xm2_t vwsmaaccsuvv_mask_int32xm2_int16xm1_uint16xm1` (`int32xm2_t merge`, `int16xm1_t a`, `uint16xm1_t b`, `int32xm2_t result`, `e16xm1_t mask`, unsigned int `gvl`)
- `int32xm4_t vwsmaaccsuvv_mask_int32xm4_int16xm2_uint16xm2` (`int32xm4_t merge`, `int16xm2_t a`, `uint16xm2_t b`, `int32xm4_t result`, `e16xm2_t mask`, unsigned int `gvl`)
- `int32xm8_t vwsmaaccsuvv_mask_int32xm8_int16xm4_uint16xm4` (`int32xm8_t merge`, `int16xm4_t a`, `uint16xm4_t b`, `int32xm8_t result`, `e16xm4_t mask`, unsigned int `gvl`)

- `int64xm2_t vsmacccsuvv_mask_int64xm2_int32xm1_uint32xm1` (`int64xm2_t` *merge*,  
`int32xm1_t` *a*,  
`uint32xm1_t` *b*,  
`int64xm2_t` *result*,  
`e32xm1_t` *mask*, unsigned int *gvl*)
- `int64xm4_t vsmacccsuvv_mask_int64xm4_int32xm2_uint32xm2` (`int64xm4_t` *merge*,  
`int32xm2_t` *a*,  
`uint32xm2_t` *b*,  
`int64xm4_t` *result*,  
`e32xm2_t` *mask*, unsigned int *gvl*)
- `int64xm8_t vsmacccsuvv_mask_int64xm8_int32xm4_uint32xm4` (`int64xm8_t` *merge*,  
`int32xm4_t` *a*,  
`uint32xm4_t` *b*,  
`int64xm8_t` *result*,  
`e32xm4_t` *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = clip(((a[element] * b[elemet] + round) >> (sew / 2)) +
↪result[element])
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

**2.7.120 Widening elementwise vector-scalar signed-unsigned integer multiply-sub, overwrite addend, with round and saturation****Instruction:** ['vsmacccsu.vx']**Prototypes:**

- `int16xm2_t vsmacccsuvx_int16xm2_uint8xm1` (signed char *a*, `uint8xm1_t` *b*, `int16xm2_t` *result*, unsigned int *gvl*)
- `int16xm4_t vsmacccsuvx_int16xm4_uint8xm2` (signed char *a*, `uint8xm2_t` *b*, `int16xm4_t` *result*, unsigned int *gvl*)
- `int16xm8_t vsmacccsuvx_int16xm8_uint8xm4` (signed char *a*, `uint8xm4_t` *b*, `int16xm8_t` *result*, unsigned int *gvl*)
- `int32xm2_t vsmacccsuvx_int32xm2_uint16xm1` (short *a*, `uint16xm1_t` *b*, `int32xm2_t` *result*, unsigned int *gvl*)
- `int32xm4_t vsmacccsuvx_int32xm4_uint16xm2` (short *a*, `uint16xm2_t` *b*, `int32xm4_t` *result*, unsigned int *gvl*)
- `int32xm8_t vsmacccsuvx_int32xm8_uint16xm4` (short *a*, `uint16xm4_t` *b*, `int32xm8_t` *result*, unsigned int *gvl*)
- `int64xm2_t vsmacccsuvx_int64xm2_uint32xm1` (int *a*, `uint32xm1_t` *b*, `int64xm2_t` *result*, unsigned int *gvl*)
- `int64xm4_t vsmacccsuvx_int64xm4_uint32xm2` (int *a*, `uint32xm2_t` *b*, `int64xm4_t` *result*, unsigned int *gvl*)

- `int64xm8_t vsmacccsuvx_int64xm8_uint32xm4` (`int a`, `uint32xm4_t b`, `int64xm8_t result`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = +widen_integer (a * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm2_t vsmacccsuvx_mask_int16xm2_uint8xm1` (`int16xm2_t merge`, signed char `a`, `uint8xm1_t b`, `int16xm2_t result`, `e8xm1_t mask`, unsigned int `gvl`)
- `int16xm4_t vsmacccsuvx_mask_int16xm4_uint8xm2` (`int16xm4_t merge`, signed char `a`, `uint8xm2_t b`, `int16xm4_t result`, `e8xm2_t mask`, unsigned int `gvl`)
- `int16xm8_t vsmacccsuvx_mask_int16xm8_uint8xm4` (`int16xm8_t merge`, signed char `a`, `uint8xm4_t b`, `int16xm8_t result`, `e8xm4_t mask`, unsigned int `gvl`)
- `int32xm2_t vsmacccsuvx_mask_int32xm2_uint16xm1` (`int32xm2_t merge`, short `a`, `uint16xm1_t b`, `int32xm2_t result`, `e16xm1_t mask`, unsigned int `gvl`)
- `int32xm4_t vsmacccsuvx_mask_int32xm4_uint16xm2` (`int32xm4_t merge`, short `a`, `uint16xm2_t b`, `int32xm4_t result`, `e16xm2_t mask`, unsigned int `gvl`)
- `int32xm8_t vsmacccsuvx_mask_int32xm8_uint16xm4` (`int32xm8_t merge`, short `a`, `uint16xm4_t b`, `int32xm8_t result`, `e16xm4_t mask`, unsigned int `gvl`)
- `int64xm2_t vsmacccsuvx_mask_int64xm2_uint32xm1` (`int64xm2_t merge`, int `a`, `uint32xm1_t b`, `int64xm2_t result`, `e32xm1_t mask`, unsigned int `gvl`)
- `int64xm4_t vsmacccsuvx_mask_int64xm4_uint32xm2` (`int64xm4_t merge`, int `a`, `uint32xm2_t b`, `int64xm4_t result`, `e32xm2_t mask`, unsigned int `gvl`)
- `int64xm8_t vsmacccsuvx_mask_int64xm8_uint32xm4` (`int64xm8_t merge`, int `a`, `uint32xm4_t b`, `int64xm8_t result`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = +widen_integer (a * b[element]) + result[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.121 Widening elementwise unsigned vector-vector multiply-add, overwrite addend, with round and saturation

**Instruction:** ['vsmaccu.vv']

**Prototypes:**

- `uint16xm2_t vwsmaaccuvv_uint16xm2_uint8xm1` (`uint8xm1_t a`, `uint8xm1_t b`, `uint16xm2_t result`, unsigned int `gvl`)
- `uint16xm4_t vwsmaaccuvv_uint16xm4_uint8xm2` (`uint8xm2_t a`, `uint8xm2_t b`, `uint16xm4_t result`, unsigned int `gvl`)
- `uint16xm8_t vwsmaaccuvv_uint16xm8_uint8xm4` (`uint8xm4_t a`, `uint8xm4_t b`, `uint16xm8_t result`, unsigned int `gvl`)
- `uint32xm2_t vwsmaaccuvv_uint32xm2_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, `uint32xm2_t result`, unsigned int `gvl`)
- `uint32xm4_t vwsmaaccuvv_uint32xm4_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, `uint32xm4_t result`, unsigned int `gvl`)
- `uint32xm8_t vwsmaaccuvv_uint32xm8_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, `uint32xm8_t result`, unsigned int `gvl`)
- `uint64xm2_t vwsmaaccuvv_uint64xm2_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, `uint64xm2_t result`, unsigned int `gvl`)
- `uint64xm4_t vwsmaaccuvv_uint64xm4_uint32xm2` (`uint32xm2_t a`, `uint32xm2_t b`, `uint64xm4_t result`, unsigned int `gvl`)
- `uint64xm8_t vwsmaaccuvv_uint64xm8_uint32xm4` (`uint32xm4_t a`, `uint32xm4_t b`, `uint64xm8_t result`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = +widen_integer(a[element] * b[element]) + result[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm2_t vwsmaaccuvv_mask_uint16xm2_uint8xm1` (`uint16xm2_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `uint16xm2_t result`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint16xm4_t vwsmaaccuvv_mask_uint16xm4_uint8xm2` (`uint16xm4_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `uint16xm4_t result`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint16xm8_t vwsmaaccuvv_mask_uint16xm8_uint8xm4` (`uint16xm8_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `uint16xm8_t result`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint32xm2_t vwsmaaccuvv_mask_uint32xm2_uint16xm1` (`uint32xm2_t merge`, `uint16xm1_t a`, `uint16xm1_t b`, `uint32xm2_t result`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint32xm4_t vwsmaaccuvv_mask_uint32xm4_uint16xm2` (`uint32xm4_t merge`, `uint16xm2_t a`, `uint16xm2_t b`, `uint32xm4_t result`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint32xm8_t vwsmaaccuvv_mask_uint32xm8_uint16xm4` (`uint32xm8_t merge`, `uint16xm4_t a`, `uint16xm4_t b`, `uint32xm8_t result`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint64xm2_t vwsmaaccuvv_mask_uint64xm2_uint32xm1` (`uint64xm2_t merge`, `uint32xm1_t a`, `uint32xm1_t b`, `uint64xm2_t result`, `e32xm1_t mask`, unsigned int `gvl`)



- `uint64xm4_t vwsmaaccuvv_mask_uint64xm4_uint32xm2` (`uint64xm4_t merge`, `uint32xm2_t a`, `uint32xm2_t b`, `uint64xm4_t result`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint64xm8_t vwsmaaccuvv_mask_uint64xm8_uint32xm4` (`uint64xm8_t merge`, `uint32xm4_t a`, `uint32xm4_t b`, `uint64xm8_t result`, `e32xm4_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = +widen_integer(a[element] * b[element]) + result[element]
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.7.122 Widening elementwise unsigned vector-scalar multiply-add, overwrite addend, with round and saturation

Instruction: ['vwsmaaccu.vx']

Prototypes:

- `uint16xm2_t vwsmaaccuvx_uint16xm2_uint8xm1` (unsigned char `a`, `uint8xm1_t b`, `uint16xm2_t result`, unsigned int `gvl`)
- `uint16xm4_t vwsmaaccuvx_uint16xm4_uint8xm2` (unsigned char `a`, `uint8xm2_t b`, `uint16xm4_t result`, unsigned int `gvl`)
- `uint16xm8_t vwsmaaccuvx_uint16xm8_uint8xm4` (unsigned char `a`, `uint8xm4_t b`, `uint16xm8_t result`, unsigned int `gvl`)
- `uint32xm2_t vwsmaaccuvx_uint32xm2_uint16xm1` (unsigned short `a`, `uint16xm1_t b`, `uint32xm2_t result`, unsigned int `gvl`)
- `uint32xm4_t vwsmaaccuvx_uint32xm4_uint16xm2` (unsigned short `a`, `uint16xm2_t b`, `uint32xm4_t result`, unsigned int `gvl`)
- `uint32xm8_t vwsmaaccuvx_uint32xm8_uint16xm4` (unsigned short `a`, `uint16xm4_t b`, `uint32xm8_t result`, unsigned int `gvl`)
- `uint64xm2_t vwsmaaccuvx_uint64xm2_uint32xm1` (unsigned int `a`, `uint32xm1_t b`, `uint64xm2_t result`, unsigned int `gvl`)
- `uint64xm4_t vwsmaaccuvx_uint64xm4_uint32xm2` (unsigned int `a`, `uint32xm2_t b`, `uint64xm4_t result`, unsigned int `gvl`)
- `uint64xm8_t vwsmaaccuvx_uint64xm8_uint32xm4` (unsigned int `a`, `uint32xm4_t b`, `uint64xm8_t result`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = +widen_integer(a * b[element]) + result[element]
      result[gvl : VLMAX] = 0
```

Masked prototypes:

- `uint16xm2_t vwsmaaccuvx_mask_uint16xm2_uint8xm1` (`uint16xm2_t merge`, unsigned char `a`, `uint8xm1_t b`, `uint16xm2_t result`, `e8xm1_t mask`, unsigned int `gvl`)

- `uint16xm4_t vwsmaaccuvx_mask_uint16xm4_uint8xm2` (`uint16xm4_t merge`, unsigned char `a`, `uint8xm2_t b`, `uint16xm4_t result`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint16xm8_t vwsmaaccuvx_mask_uint16xm8_uint8xm4` (`uint16xm8_t merge`, unsigned char `a`, `uint8xm4_t b`, `uint16xm8_t result`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint32xm2_t vwsmaaccuvx_mask_uint32xm2_uint16xm1` (`uint32xm2_t merge`, unsigned short `a`, `uint16xm1_t b`, `uint32xm2_t result`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint32xm4_t vwsmaaccuvx_mask_uint32xm4_uint16xm2` (`uint32xm4_t merge`, unsigned short `a`, `uint16xm2_t b`, `uint32xm4_t result`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint32xm8_t vwsmaaccuvx_mask_uint32xm8_uint16xm4` (`uint32xm8_t merge`, unsigned short `a`, `uint16xm4_t b`, `uint32xm8_t result`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint64xm2_t vwsmaaccuvx_mask_uint64xm2_uint32xm1` (`uint64xm2_t merge`, unsigned int `a`, `uint32xm1_t b`, `uint64xm2_t result`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint64xm4_t vwsmaaccuvx_mask_uint64xm4_uint32xm2` (`uint64xm4_t merge`, unsigned int `a`, `uint32xm2_t b`, `uint64xm4_t result`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint64xm8_t vwsmaaccuvx_mask_uint64xm8_uint32xm4` (`uint64xm8_t merge`, unsigned int `a`, `uint32xm4_t b`, `uint64xm8_t result`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = +widen_integer(a * b[element]) + result[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.123 Widening elementwise vector-scalar unsigned-signed integer multiply-sub, overwrite addend, with round and saturation****Instruction:** [`'vwsmaaccus.vx'`]**Prototypes:**

- `int16xm2_t vwsmaaccusvx_int16xm2_int8xm1` (unsigned char `a`, `int8xm1_t b`, `int16xm2_t result`, unsigned int `gvl`)
- `int16xm4_t vwsmaaccusvx_int16xm4_int8xm2` (unsigned char `a`, `int8xm2_t b`, `int16xm4_t result`, unsigned int `gvl`)
- `int16xm8_t vwsmaaccusvx_int16xm8_int8xm4` (unsigned char `a`, `int8xm4_t b`, `int16xm8_t result`, unsigned int `gvl`)
- `int32xm2_t vwsmaaccusvx_int32xm2_int16xm1` (unsigned short `a`, `int16xm1_t b`, `int32xm2_t result`, unsigned int `gvl`)
- `int32xm4_t vwsmaaccusvx_int32xm4_int16xm2` (unsigned short `a`, `int16xm2_t b`, `int32xm4_t result`, unsigned int `gvl`)

- `int32xm8_t vsmaccusvx_int32xm8_int16xm4` (unsigned short *a*, `int16xm4_t` *b*, `int32xm8_t` *result*, unsigned int *gvl*)
- `int64xm2_t vsmaccusvx_int64xm2_int32xm1` (unsigned int *a*, `int32xm1_t` *b*, `int64xm2_t` *result*, unsigned int *gvl*)
- `int64xm4_t vsmaccusvx_int64xm4_int32xm2` (unsigned int *a*, `int32xm2_t` *b*, `int64xm4_t` *result*, unsigned int *gvl*)
- `int64xm8_t vsmaccusvx_int64xm8_int32xm4` (unsigned int *a*, `int32xm4_t` *b*, `int64xm8_t` *result*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = clip(((a * b[elemet] + round) >> (sew / 2)) + result[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm2_t vsmaccusvx_mask_int16xm2_int8xm1` (`int16xm2_t` *merge*, unsigned char *a*, `int8xm1_t` *b*, `int16xm2_t` *result*, `e8xm1_t` *mask*, unsigned int *gvl*)
- `int16xm4_t vsmaccusvx_mask_int16xm4_int8xm2` (`int16xm4_t` *merge*, unsigned char *a*, `int8xm2_t` *b*, `int16xm4_t` *result*, `e8xm2_t` *mask*, unsigned int *gvl*)
- `int16xm8_t vsmaccusvx_mask_int16xm8_int8xm4` (`int16xm8_t` *merge*, unsigned char *a*, `int8xm4_t` *b*, `int16xm8_t` *result*, `e8xm4_t` *mask*, unsigned int *gvl*)
- `int32xm2_t vsmaccusvx_mask_int32xm2_int16xm1` (`int32xm2_t` *merge*, unsigned short *a*, `int16xm1_t` *b*, `int32xm2_t` *result*, `e16xm1_t` *mask*, unsigned int *gvl*)
- `int32xm4_t vsmaccusvx_mask_int32xm4_int16xm2` (`int32xm4_t` *merge*, unsigned short *a*, `int16xm2_t` *b*, `int32xm4_t` *result*, `e16xm2_t` *mask*, unsigned int *gvl*)
- `int32xm8_t vsmaccusvx_mask_int32xm8_int16xm4` (`int32xm8_t` *merge*, unsigned short *a*, `int16xm4_t` *b*, `int32xm8_t` *result*, `e16xm4_t` *mask*, unsigned int *gvl*)
- `int64xm2_t vsmaccusvx_mask_int64xm2_int32xm1` (`int64xm2_t` *merge*, unsigned int *a*, `int32xm1_t` *b*, `int64xm2_t` *result*, `e32xm1_t` *mask*, unsigned int *gvl*)
- `int64xm4_t vsmaccusvx_mask_int64xm4_int32xm2` (`int64xm4_t` *merge*, unsigned int *a*, `int32xm2_t` *b*, `int64xm4_t` *result*, `e32xm2_t` *mask*, unsigned int *gvl*)
- `int64xm8_t vsmaccusvx_mask_int64xm8_int32xm4` (`int64xm8_t` *merge*, unsigned int *a*, `int32xm4_t` *b*, `int64xm8_t` *result*, `e32xm4_t` *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = clip(((a * b[elemet] + round) >> (sew / 2)) +
↪ result[element])
      else
```

(continues on next page)

(continued from previous page)

```
result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.124 Widening elementwise signed vector-vector subtraction

**Instruction:** [`vsub.vv`']

**Prototypes:**

- `int16xm2_t vsubvv_int16xm2_int8xm1` (`int8xm1_t a`, `int8xm1_t b`, unsigned int `gvl`)
- `int16xm4_t vsubvv_int16xm4_int8xm2` (`int8xm2_t a`, `int8xm2_t b`, unsigned int `gvl`)
- `int16xm8_t vsubvv_int16xm8_int8xm4` (`int8xm4_t a`, `int8xm4_t b`, unsigned int `gvl`)
- `int32xm2_t vsubvv_int32xm2_int16xm1` (`int16xm1_t a`, `int16xm1_t b`, unsigned int `gvl`)
- `int32xm4_t vsubvv_int32xm4_int16xm2` (`int16xm2_t a`, `int16xm2_t b`, unsigned int `gvl`)
- `int32xm8_t vsubvv_int32xm8_int16xm4` (`int16xm4_t a`, `int16xm4_t b`, unsigned int `gvl`)
- `int64xm2_t vsubvv_int64xm2_int32xm1` (`int32xm1_t a`, `int32xm1_t b`, unsigned int `gvl`)
- `int64xm4_t vsubvv_int64xm4_int32xm2` (`int32xm2_t a`, `int32xm2_t b`, unsigned int `gvl`)
- `int64xm8_t vsubvv_int64xm8_int32xm4` (`int32xm4_t a`, `int32xm4_t b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = widen_integer (a[element]) - widen_integer (b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm2_t vsubvv_mask_int16xm2_int8xm1` (`int16xm2_t merge`, `int8xm1_t a`, `int8xm1_t b`, `e8xm1_t mask`, unsigned int `gvl`)
- `int16xm4_t vsubvv_mask_int16xm4_int8xm2` (`int16xm4_t merge`, `int8xm2_t a`, `int8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `int16xm8_t vsubvv_mask_int16xm8_int8xm4` (`int16xm8_t merge`, `int8xm4_t a`, `int8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int32xm2_t vsubvv_mask_int32xm2_int16xm1` (`int32xm2_t merge`, `int16xm1_t a`, `int16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `int32xm4_t vsubvv_mask_int32xm4_int16xm2` (`int32xm4_t merge`, `int16xm2_t a`, `int16xm2_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `int32xm8_t vsubvv_mask_int32xm8_int16xm4` (`int32xm8_t merge`, `int16xm4_t a`, `int16xm4_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `int64xm2_t vsubvv_mask_int64xm2_int32xm1` (`int64xm2_t merge`, `int32xm1_t a`, `int32xm1_t b`, `e32xm1_t mask`, unsigned int `gvl`)
- `int64xm4_t vsubvv_mask_int64xm4_int32xm2` (`int64xm4_t merge`, `int32xm2_t a`, `int32xm2_t b`, `e32xm2_t mask`, unsigned int `gvl`)
- `int64xm8_t vsubvv_mask_int64xm8_int32xm4` (`int64xm8_t merge`, `int32xm4_t a`, `int32xm4_t b`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = widen_integer (a[element]) - widen_integer (b[element])
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.7.125 Widening elementwise signed vector-scalar subtraction

**Instruction:** ['vwsbvx']

**Prototypes:**

- *int16xm2\_t vwsbvx\_int16xm2\_int8xm1* (*int8xm1\_t a*, signed char *b*, unsigned int *gvl*)
- *int16xm4\_t vwsbvx\_int16xm4\_int8xm2* (*int8xm2\_t a*, signed char *b*, unsigned int *gvl*)
- *int16xm8\_t vwsbvx\_int16xm8\_int8xm4* (*int8xm4\_t a*, signed char *b*, unsigned int *gvl*)
- *int32xm2\_t vwsbvx\_int32xm2\_int16xm1* (*int16xm1\_t a*, short *b*, unsigned int *gvl*)
- *int32xm4\_t vwsbvx\_int32xm4\_int16xm2* (*int16xm2\_t a*, short *b*, unsigned int *gvl*)
- *int32xm8\_t vwsbvx\_int32xm8\_int16xm4* (*int16xm4\_t a*, short *b*, unsigned int *gvl*)
- *int64xm2\_t vwsbvx\_int64xm2\_int32xm1* (*int32xm1\_t a*, int *b*, unsigned int *gvl*)
- *int64xm4\_t vwsbvx\_int64xm4\_int32xm2* (*int32xm2\_t a*, int *b*, unsigned int *gvl*)
- *int64xm8\_t vwsbvx\_int64xm8\_int32xm4* (*int32xm4\_t a*, int *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = widen_integer (a[element]) - widen_integer (b)
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm2\_t vwsbvx\_mask\_int16xm2\_int8xm1* (*int16xm2\_t merge*, *int8xm1\_t a*, signed char *b*, *e8xm1\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vwsbvx\_mask\_int16xm4\_int8xm2* (*int16xm4\_t merge*, *int8xm2\_t a*, signed char *b*, *e8xm2\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vwsbvx\_mask\_int16xm8\_int8xm4* (*int16xm8\_t merge*, *int8xm4\_t a*, signed char *b*, *e8xm4\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vwsbvx\_mask\_int32xm2\_int16xm1* (*int32xm2\_t merge*, *int16xm1\_t a*, short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vwsbvx\_mask\_int32xm4\_int16xm2* (*int32xm4\_t merge*, *int16xm2\_t a*, short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int32xm8\_t vwsbvx\_mask\_int32xm8\_int16xm4* (*int32xm8\_t merge*, *int16xm4\_t a*, short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int64xm2\_t vwsbvx\_mask\_int64xm2\_int32xm1* (*int64xm2\_t merge*, *int32xm1\_t a*, int *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *int64xm4\_t vwsbvx\_mask\_int64xm4\_int32xm2* (*int64xm4\_t merge*, *int32xm2\_t a*, int *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *int64xm8\_t vwsbvx\_mask\_int64xm8\_int32xm4* (*int64xm8\_t merge*, *int32xm4\_t a*, int *b*, *e32xm4\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = widen_integer (a[element]) - widen_integer (b)
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.126 Widening elementwise (Widening)signed vector-vector subtraction****Instruction:** ['vwsbvwv']**Prototypes:**

- *int16xm2\_t vwsbvwv\_int16xm2\_int8xm1* (*int16xm2\_t a*, *int8xm1\_t b*, unsigned int *gvl*)
- *int16xm4\_t vwsbvwv\_int16xm4\_int8xm2* (*int16xm4\_t a*, *int8xm2\_t b*, unsigned int *gvl*)
- *int16xm8\_t vwsbvwv\_int16xm8\_int8xm4* (*int16xm8\_t a*, *int8xm4\_t b*, unsigned int *gvl*)
- *int32xm2\_t vwsbvwv\_int32xm2\_int16xm1* (*int32xm2\_t a*, *int16xm1\_t b*, unsigned int *gvl*)
- *int32xm4\_t vwsbvwv\_int32xm4\_int16xm2* (*int32xm4\_t a*, *int16xm2\_t b*, unsigned int *gvl*)
- *int32xm8\_t vwsbvwv\_int32xm8\_int16xm4* (*int32xm8\_t a*, *int16xm4\_t b*, unsigned int *gvl*)
- *int64xm2\_t vwsbvwv\_int64xm2\_int32xm1* (*int64xm2\_t a*, *int32xm1\_t b*, unsigned int *gvl*)
- *int64xm4\_t vwsbvwv\_int64xm4\_int32xm2* (*int64xm4\_t a*, *int32xm2\_t b*, unsigned int *gvl*)
- *int64xm8\_t vwsbvwv\_int64xm8\_int32xm4* (*int64xm8\_t a*, *int32xm4\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = a[element] - widen_integer (b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm2\_t vwsbvwv\_mask\_int16xm2\_int8xm1* (*int16xm2\_t merge*, *int16xm2\_t a*, *int8xm1\_t b*, *e8xm1\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vwsbvwv\_mask\_int16xm4\_int8xm2* (*int16xm4\_t merge*, *int16xm4\_t a*, *int8xm2\_t b*, *e8xm2\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vwsbvwv\_mask\_int16xm8\_int8xm4* (*int16xm8\_t merge*, *int16xm8\_t a*, *int8xm4\_t b*, *e8xm4\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vwsbvwv\_mask\_int32xm2\_int16xm1* (*int32xm2\_t merge*, *int32xm2\_t a*, *int16xm1\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vwsbvwv\_mask\_int32xm4\_int16xm2* (*int32xm4\_t merge*, *int32xm4\_t a*, *int16xm2\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int32xm8\_t vwsbvwv\_mask\_int32xm8\_int16xm4* (*int32xm8\_t merge*, *int32xm8\_t a*, *int16xm4\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int64xm2\_t vwsbvwv\_mask\_int64xm2\_int32xm1* (*int64xm2\_t merge*, *int64xm2\_t a*, *int32xm1\_t b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *int64xm4\_t vwsbvwv\_mask\_int64xm4\_int32xm2* (*int64xm4\_t merge*, *int64xm4\_t a*, *int32xm2\_t b*, *e32xm2\_t mask*, unsigned int *gvl*)



- `int64xm8_t vsubwv_mask_int64xm8_int32xm4` (`int64xm8_t merge`, `int64xm8_t a`, `int32xm4_t b`, `e32xm4_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] - widen_integer (b[element])
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.7.127 Widening elementwise (Widening)signed vector-scalar subtraction

Instruction: ['vsub.wx']

Prototypes:

- `int16xm2_t vsubwx_int16xm2` (`int16xm2_t a`, signed char `b`, unsigned int `gvl`)
- `int16xm4_t vsubwx_int16xm4` (`int16xm4_t a`, signed char `b`, unsigned int `gvl`)
- `int16xm8_t vsubwx_int16xm8` (`int16xm8_t a`, signed char `b`, unsigned int `gvl`)
- `int32xm2_t vsubwx_int32xm2` (`int32xm2_t a`, short `b`, unsigned int `gvl`)
- `int32xm4_t vsubwx_int32xm4` (`int32xm4_t a`, short `b`, unsigned int `gvl`)
- `int32xm8_t vsubwx_int32xm8` (`int32xm8_t a`, short `b`, unsigned int `gvl`)
- `int64xm2_t vsubwx_int64xm2` (`int64xm2_t a`, int `b`, unsigned int `gvl`)
- `int64xm4_t vsubwx_int64xm4` (`int64xm4_t a`, int `b`, unsigned int `gvl`)
- `int64xm8_t vsubwx_int64xm8` (`int64xm8_t a`, int `b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] - widen_integer (b)
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `int16xm2_t vsubwx_mask_int16xm2` (`int16xm2_t merge`, `int16xm2_t a`, signed char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `int16xm4_t vsubwx_mask_int16xm4` (`int16xm4_t merge`, `int16xm4_t a`, signed char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `int16xm8_t vsubwx_mask_int16xm8` (`int16xm8_t merge`, `int16xm8_t a`, signed char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int32xm2_t vsubwx_mask_int32xm2` (`int32xm2_t merge`, `int32xm2_t a`, short `b`, `e16xm1_t mask`, unsigned int `gvl`)
- `int32xm4_t vsubwx_mask_int32xm4` (`int32xm4_t merge`, `int32xm4_t a`, short `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `int32xm8_t vsubwx_mask_int32xm8` (`int32xm8_t merge`, `int32xm8_t a`, short `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `int64xm2_t vsubwx_mask_int64xm2` (`int64xm2_t merge`, `int64xm2_t a`, int `b`, `e32xm1_t mask`, unsigned int `gvl`)

- `int64xm4_t vsubwx_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, `int b`, `e32xm2_t mask`, unsigned int `gvl`)
- `int64xm8_t vsubwx_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, `int b`, `e32xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] - widen_integer (b)
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.7.128 Widening elementwise unsigned vector-vector subtraction

**Instruction:** [`'vsubu.vv'`]

**Prototypes:**

- `uint16xm2_t vsubuvv_uint16xm2_uint8xm1` (`uint8xm1_t a`, `uint8xm1_t b`, unsigned int `gvl`)
- `uint16xm4_t vsubuvv_uint16xm4_uint8xm2` (`uint8xm2_t a`, `uint8xm2_t b`, unsigned int `gvl`)
- `uint16xm8_t vsubuvv_uint16xm8_uint8xm4` (`uint8xm4_t a`, `uint8xm4_t b`, unsigned int `gvl`)
- `uint32xm2_t vsubuvv_uint32xm2_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, unsigned int `gvl`)
- `uint32xm4_t vsubuvv_uint32xm4_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, unsigned int `gvl`)
- `uint32xm8_t vsubuvv_uint32xm8_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, unsigned int `gvl`)
- `uint64xm2_t vsubuvv_uint64xm2_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, unsigned int `gvl`)
- `uint64xm4_t vsubuvv_uint64xm4_uint32xm2` (`uint32xm2_t a`, `uint32xm2_t b`, unsigned int `gvl`)
- `uint64xm8_t vsubuvv_uint64xm8_uint32xm4` (`uint32xm4_t a`, `uint32xm4_t b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = widen_integer (a[element]) - widen_integer (b[element])
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm2_t vsubuvv_mask_uint16xm2_uint8xm1` (`uint16xm2_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint16xm4_t vsubuvv_mask_uint16xm4_uint8xm2` (`uint16xm4_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint16xm8_t vsubuvv_mask_uint16xm8_uint8xm4` (`uint16xm8_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint32xm2_t vsubuvv_mask_uint32xm2_uint16xm1` (`uint32xm2_t merge`, `uint16xm1_t a`, `uint16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)

- `uint32xm4_t vwsbuvv_mask_uint32xm4_uint16xm2 (uint32xm4_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint32xm8_t vwsbuvv_mask_uint32xm8_uint16xm4 (uint32xm8_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint64xm2_t vwsbuvv_mask_uint64xm2_uint32xm1 (uint64xm2_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint64xm4_t vwsbuvv_mask_uint64xm4_uint32xm2 (uint64xm4_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint64xm8_t vwsbuvv_mask_uint64xm8_uint32xm4 (uint64xm8_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = widen_integer (a[element]) - widen_integer (b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.129 Widening elementwise unsigned vector-scalar subtraction****Instruction:** ['vwsbu.vx']**Prototypes:**

- `uint16xm2_t vwsbuvx_uint16xm2_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint16xm4_t vwsbuvx_uint16xm4_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint16xm8_t vwsbuvx_uint16xm8_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint32xm2_t vwsbuvx_uint32xm2_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint32xm4_t vwsbuvx_uint32xm4_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint32xm8_t vwsbuvx_uint32xm8_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint64xm2_t vwsbuvx_uint64xm2_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint64xm4_t vwsbuvx_uint64xm4_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint64xm8_t vwsbuvx_uint64xm8_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = widen_integer (a[element]) - widen_integer (b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm2_t vwsbuvx_mask_uint16xm2_uint8xm1` (`uint16xm2_t` merge, `uint8xm1_t` *a*, unsigned char *b*, `e8xm1_t` mask, unsigned int *gvl*)
- `uint16xm4_t vwsbuvx_mask_uint16xm4_uint8xm2` (`uint16xm4_t` merge, `uint8xm2_t` *a*, unsigned char *b*, `e8xm2_t` mask, unsigned int *gvl*)
- `uint16xm8_t vwsbuvx_mask_uint16xm8_uint8xm4` (`uint16xm8_t` merge, `uint8xm4_t` *a*, unsigned char *b*, `e8xm4_t` mask, unsigned int *gvl*)
- `uint32xm2_t vwsbuvx_mask_uint32xm2_uint16xm1` (`uint32xm2_t` merge, `uint16xm1_t` *a*, unsigned short *b*, `e16xm1_t` mask, unsigned int *gvl*)
- `uint32xm4_t vwsbuvx_mask_uint32xm4_uint16xm2` (`uint32xm4_t` merge, `uint16xm2_t` *a*, unsigned short *b*, `e16xm2_t` mask, unsigned int *gvl*)
- `uint32xm8_t vwsbuvx_mask_uint32xm8_uint16xm4` (`uint32xm8_t` merge, `uint16xm4_t` *a*, unsigned short *b*, `e16xm4_t` mask, unsigned int *gvl*)
- `uint64xm2_t vwsbuvx_mask_uint64xm2_uint32xm1` (`uint64xm2_t` merge, `uint32xm1_t` *a*, unsigned int *b*, `e32xm1_t` mask, unsigned int *gvl*)
- `uint64xm4_t vwsbuvx_mask_uint64xm4_uint32xm2` (`uint64xm4_t` merge, `uint32xm2_t` *a*, unsigned int *b*, `e32xm2_t` mask, unsigned int *gvl*)
- `uint64xm8_t vwsbuvx_mask_uint64xm8_uint32xm4` (`uint64xm8_t` merge, `uint32xm4_t` *a*, unsigned int *b*, `e32xm4_t` mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = widen_integer (a[element]) - widen_integer (b)
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.7.130 Widening elementwise (Widening)unsigned vector-vector subtraction****Instruction:** [`'vwsbuwv'`]**Prototypes:**

- `uint16xm2_t vwsbuwv_uint16xm2_uint8xm1` (`uint16xm2_t` *a*, `uint8xm1_t` *b*, unsigned int *gvl*)
- `uint16xm4_t vwsbuwv_uint16xm4_uint8xm2` (`uint16xm4_t` *a*, `uint8xm2_t` *b*, unsigned int *gvl*)
- `uint16xm8_t vwsbuwv_uint16xm8_uint8xm4` (`uint16xm8_t` *a*, `uint8xm4_t` *b*, unsigned int *gvl*)
- `uint32xm2_t vwsbuwv_uint32xm2_uint16xm1` (`uint32xm2_t` *a*, `uint16xm1_t` *b*, unsigned int *gvl*)
- `uint32xm4_t vwsbuwv_uint32xm4_uint16xm2` (`uint32xm4_t` *a*, `uint16xm2_t` *b*, unsigned int *gvl*)
- `uint32xm8_t vwsbuwv_uint32xm8_uint16xm4` (`uint32xm8_t` *a*, `uint16xm4_t` *b*, unsigned int *gvl*)

- `uint64xm2_t vwsuwv_uint64xm2_uint32xm1 (uint64xm2_t a, uint32xm1_t b, unsigned int gvl)`
- `uint64xm4_t vwsuwv_uint64xm4_uint32xm2 (uint64xm4_t a, uint32xm2_t b, unsigned int gvl)`
- `uint64xm8_t vwsuwv_uint64xm8_uint32xm4 (uint64xm8_t a, uint32xm4_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = a[element] - widen_integer (b[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm2_t vwsuwv_mask_uint16xm2_uint8xm1 (uint16xm2_t merge, uint16xm2_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint16xm4_t vwsuwv_mask_uint16xm4_uint8xm2 (uint16xm4_t merge, uint16xm4_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint16xm8_t vwsuwv_mask_uint16xm8_uint8xm4 (uint16xm8_t merge, uint16xm8_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint32xm2_t vwsuwv_mask_uint32xm2_uint16xm1 (uint32xm2_t merge, uint32xm2_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `uint32xm4_t vwsuwv_mask_uint32xm4_uint16xm2 (uint32xm4_t merge, uint32xm4_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `uint32xm8_t vwsuwv_mask_uint32xm8_uint16xm4 (uint32xm8_t merge, uint32xm8_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `uint64xm2_t vwsuwv_mask_uint64xm2_uint32xm1 (uint64xm2_t merge, uint64xm2_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `uint64xm4_t vwsuwv_mask_uint64xm4_uint32xm2 (uint64xm4_t merge, uint64xm4_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `uint64xm8_t vwsuwv_mask_uint64xm8_uint32xm4 (uint64xm8_t merge, uint64xm8_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = a[element] - widen_integer (b[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.7.131 Widening elementwise (Widening)unsigned vector-scalar subtraction****Instruction:** [`'vwsu.w'`]

**Prototypes:**

- `uint16xm2_t vwsbuwx_uint16xm2 (uint16xm2_t a, unsigned char b, unsigned int gvl)`
- `uint16xm4_t vwsbuwx_uint16xm4 (uint16xm4_t a, unsigned char b, unsigned int gvl)`
- `uint16xm8_t vwsbuwx_uint16xm8 (uint16xm8_t a, unsigned char b, unsigned int gvl)`
- `uint32xm2_t vwsbuwx_uint32xm2 (uint32xm2_t a, unsigned short b, unsigned int gvl)`
- `uint32xm4_t vwsbuwx_uint32xm4 (uint32xm4_t a, unsigned short b, unsigned int gvl)`
- `uint32xm8_t vwsbuwx_uint32xm8 (uint32xm8_t a, unsigned short b, unsigned int gvl)`
- `uint64xm2_t vwsbuwx_uint64xm2 (uint64xm2_t a, unsigned int b, unsigned int gvl)`
- `uint64xm4_t vwsbuwx_uint64xm4 (uint64xm4_t a, unsigned int b, unsigned int gvl)`
- `uint64xm8_t vwsbuwx_uint64xm8 (uint64xm8_t a, unsigned int b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] - widen_integer (b)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm2_t vwsbuwx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `uint16xm4_t vwsbuwx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `uint16xm8_t vwsbuwx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `uint32xm2_t vwsbuwx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint32xm4_t vwsbuwx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `uint32xm8_t vwsbuwx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `uint64xm2_t vwsbuwx_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `uint64xm4_t vwsbuwx_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `uint64xm8_t vwsbuwx_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] - widen_integer (b)
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```



## 2.8 Integer relational operations

### 2.8.1 Compare elementwise vector-immediate for equality

**Instruction:** [`vmseq.vi`']

**Prototypes:**

- `e16xm1_t vmseqvi_e16xm1_int16xm1 (int16xm1_t a, const short b, unsigned int gvl)`
- `e16xm1_t vmseqvi_e16xm1_uint16xm1 (uint16xm1_t a, const unsigned short b, unsigned int gvl)`
- `e16xm2_t vmseqvi_e16xm2_int16xm2 (int16xm2_t a, const short b, unsigned int gvl)`
- `e16xm2_t vmseqvi_e16xm2_uint16xm2 (uint16xm2_t a, const unsigned short b, unsigned int gvl)`
- `e16xm4_t vmseqvi_e16xm4_int16xm4 (int16xm4_t a, const short b, unsigned int gvl)`
- `e16xm4_t vmseqvi_e16xm4_uint16xm4 (uint16xm4_t a, const unsigned short b, unsigned int gvl)`
- `e16xm8_t vmseqvi_e16xm8_int16xm8 (int16xm8_t a, const short b, unsigned int gvl)`
- `e16xm8_t vmseqvi_e16xm8_uint16xm8 (uint16xm8_t a, const unsigned short b, unsigned int gvl)`
- `e32xm1_t vmseqvi_e32xm1_int32xm1 (int32xm1_t a, const int b, unsigned int gvl)`
- `e32xm1_t vmseqvi_e32xm1_uint32xm1 (uint32xm1_t a, const unsigned int b, unsigned int gvl)`
- `e32xm2_t vmseqvi_e32xm2_int32xm2 (int32xm2_t a, const int b, unsigned int gvl)`
- `e32xm2_t vmseqvi_e32xm2_uint32xm2 (uint32xm2_t a, const unsigned int b, unsigned int gvl)`
- `e32xm4_t vmseqvi_e32xm4_int32xm4 (int32xm4_t a, const int b, unsigned int gvl)`
- `e32xm4_t vmseqvi_e32xm4_uint32xm4 (uint32xm4_t a, const unsigned int b, unsigned int gvl)`
- `e32xm8_t vmseqvi_e32xm8_int32xm8 (int32xm8_t a, const int b, unsigned int gvl)`
- `e32xm8_t vmseqvi_e32xm8_uint32xm8 (uint32xm8_t a, const unsigned int b, unsigned int gvl)`
- `e64xm1_t vmseqvi_e64xm1_int64xm1 (int64xm1_t a, const long b, unsigned int gvl)`
- `e64xm1_t vmseqvi_e64xm1_uint64xm1 (uint64xm1_t a, const unsigned long b, unsigned int gvl)`
- `e64xm2_t vmseqvi_e64xm2_int64xm2 (int64xm2_t a, const long b, unsigned int gvl)`
- `e64xm2_t vmseqvi_e64xm2_uint64xm2 (uint64xm2_t a, const unsigned long b, unsigned int gvl)`
- `e64xm4_t vmseqvi_e64xm4_int64xm4 (int64xm4_t a, const long b, unsigned int gvl)`
- `e64xm4_t vmseqvi_e64xm4_uint64xm4 (uint64xm4_t a, const unsigned long b, unsigned int gvl)`
- `e64xm8_t vmseqvi_e64xm8_int64xm8 (int64xm8_t a, const long b, unsigned int gvl)`
- `e64xm8_t vmseqvi_e64xm8_uint64xm8 (uint64xm8_t a, const unsigned long b, unsigned int gvl)`
- `e8xm1_t vmseqvi_e8xm1_int8xm1 (int8xm1_t a, const signed char b, unsigned int gvl)`
- `e8xm1_t vmseqvi_e8xm1_uint8xm1 (uint8xm1_t a, const unsigned char b, unsigned int gvl)`
- `e8xm2_t vmseqvi_e8xm2_int8xm2 (int8xm2_t a, const signed char b, unsigned int gvl)`
- `e8xm2_t vmseqvi_e8xm2_uint8xm2 (uint8xm2_t a, const unsigned char b, unsigned int gvl)`
- `e8xm4_t vmseqvi_e8xm4_int8xm4 (int8xm4_t a, const signed char b, unsigned int gvl)`
- `e8xm4_t vmseqvi_e8xm4_uint8xm4 (uint8xm4_t a, const unsigned char b, unsigned int gvl)`
- `e8xm8_t vmseqvi_e8xm8_int8xm8 (int8xm8_t a, const signed char b, unsigned int gvl)`

- *e8xm8\_t vmseqvi\_e8xm8\_uint8xm8* (*uint8xm8\_t a*, const unsigned char *b*, unsigned int *gvl*)

#### Operation:

```
>>> for element = 0 to gvl - 1
    result[element] = a[element] == b
result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- *e16xm1\_t vmseqvi\_mask\_e16xm1\_int16xm1* (*e16xm1\_t merge*, *int16xm1\_t a*, const short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm1\_t vmseqvi\_mask\_e16xm1\_uint16xm1* (*e16xm1\_t merge*, *uint16xm1\_t a*, const unsigned short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmseqvi\_mask\_e16xm2\_int16xm2* (*e16xm2\_t merge*, *int16xm2\_t a*, const short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmseqvi\_mask\_e16xm2\_uint16xm2* (*e16xm2\_t merge*, *uint16xm2\_t a*, const unsigned short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmseqvi\_mask\_e16xm4\_int16xm4* (*e16xm4\_t merge*, *int16xm4\_t a*, const short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmseqvi\_mask\_e16xm4\_uint16xm4* (*e16xm4\_t merge*, *uint16xm4\_t a*, const unsigned short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmseqvi\_mask\_e16xm8\_int16xm8* (*e16xm8\_t merge*, *int16xm8\_t a*, const short *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmseqvi\_mask\_e16xm8\_uint16xm8* (*e16xm8\_t merge*, *uint16xm8\_t a*, const unsigned short *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmseqvi\_mask\_e32xm1\_int32xm1* (*e32xm1\_t merge*, *int32xm1\_t a*, const int *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmseqvi\_mask\_e32xm1\_uint32xm1* (*e32xm1\_t merge*, *uint32xm1\_t a*, const unsigned int *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm2\_t vmseqvi\_mask\_e32xm2\_int32xm2* (*e32xm2\_t merge*, *int32xm2\_t a*, const int *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *e32xm2\_t vmseqvi\_mask\_e32xm2\_uint32xm2* (*e32xm2\_t merge*, *uint32xm2\_t a*, const unsigned int *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *e32xm4\_t vmseqvi\_mask\_e32xm4\_int32xm4* (*e32xm4\_t merge*, *int32xm4\_t a*, const int *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *e32xm4\_t vmseqvi\_mask\_e32xm4\_uint32xm4* (*e32xm4\_t merge*, *uint32xm4\_t a*, const unsigned int *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *e32xm8\_t vmseqvi\_mask\_e32xm8\_int32xm8* (*e32xm8\_t merge*, *int32xm8\_t a*, const int *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *e32xm8\_t vmseqvi\_mask\_e32xm8\_uint32xm8* (*e32xm8\_t merge*, *uint32xm8\_t a*, const unsigned int *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *e64xm1\_t vmseqvi\_mask\_e64xm1\_int64xm1* (*e64xm1\_t merge*, *int64xm1\_t a*, const long *b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *e64xm1\_t vmseqvi\_mask\_e64xm1\_uint64xm1* (*e64xm1\_t merge*, *uint64xm1\_t a*, const unsigned long *b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *e64xm2\_t vmseqvi\_mask\_e64xm2\_int64xm2* (*e64xm2\_t merge*, *int64xm2\_t a*, const long *b*, *e64xm2\_t mask*, unsigned int *gvl*)

- *e64xm2\_t vmseqvi\_mask\_e64xm2\_uint64xm2* (*e64xm2\_t merge*, *uint64xm2\_t a*, const unsigned long *b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *e64xm4\_t vmseqvi\_mask\_e64xm4\_int64xm4* (*e64xm4\_t merge*, *int64xm4\_t a*, const long *b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *e64xm4\_t vmseqvi\_mask\_e64xm4\_uint64xm4* (*e64xm4\_t merge*, *uint64xm4\_t a*, const unsigned long *b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *e64xm8\_t vmseqvi\_mask\_e64xm8\_int64xm8* (*e64xm8\_t merge*, *int64xm8\_t a*, const long *b*, *e64xm8\_t mask*, unsigned int *gvl*)
- *e64xm8\_t vmseqvi\_mask\_e64xm8\_uint64xm8* (*e64xm8\_t merge*, *uint64xm8\_t a*, const unsigned long *b*, *e64xm8\_t mask*, unsigned int *gvl*)
- *e8xm1\_t vmseqvi\_mask\_e8xm1\_int8xm1* (*e8xm1\_t merge*, *int8xm1\_t a*, const signed char *b*, *e8xm1\_t mask*, unsigned int *gvl*)
- *e8xm1\_t vmseqvi\_mask\_e8xm1\_uint8xm1* (*e8xm1\_t merge*, *uint8xm1\_t a*, const unsigned char *b*, *e8xm1\_t mask*, unsigned int *gvl*)
- *e8xm2\_t vmseqvi\_mask\_e8xm2\_int8xm2* (*e8xm2\_t merge*, *int8xm2\_t a*, const signed char *b*, *e8xm2\_t mask*, unsigned int *gvl*)
- *e8xm2\_t vmseqvi\_mask\_e8xm2\_uint8xm2* (*e8xm2\_t merge*, *uint8xm2\_t a*, const unsigned char *b*, *e8xm2\_t mask*, unsigned int *gvl*)
- *e8xm4\_t vmseqvi\_mask\_e8xm4\_int8xm4* (*e8xm4\_t merge*, *int8xm4\_t a*, const signed char *b*, *e8xm4\_t mask*, unsigned int *gvl*)
- *e8xm4\_t vmseqvi\_mask\_e8xm4\_uint8xm4* (*e8xm4\_t merge*, *uint8xm4\_t a*, const unsigned char *b*, *e8xm4\_t mask*, unsigned int *gvl*)
- *e8xm8\_t vmseqvi\_mask\_e8xm8\_int8xm8* (*e8xm8\_t merge*, *int8xm8\_t a*, const signed char *b*, *e8xm8\_t mask*, unsigned int *gvl*)
- *e8xm8\_t vmseqvi\_mask\_e8xm8\_uint8xm8* (*e8xm8\_t merge*, *uint8xm8\_t a*, const unsigned char *b*, *e8xm8\_t mask*, unsigned int *gvl*)

Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = a[element] == b
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.8.2 Compare elementwise vector-vector for equality

Instruction: ['vmseq.vv']

Prototypes:

- *e16xm1\_t vmseqvv\_e16xm1\_int16xm1* (*int16xm1\_t a*, *int16xm1\_t b*, unsigned int *gvl*)
- *e16xm1\_t vmseqvv\_e16xm1\_uint16xm1* (*uint16xm1\_t a*, *uint16xm1\_t b*, unsigned int *gvl*)
- *e16xm2\_t vmseqvv\_e16xm2\_int16xm2* (*int16xm2\_t a*, *int16xm2\_t b*, unsigned int *gvl*)
- *e16xm2\_t vmseqvv\_e16xm2\_uint16xm2* (*uint16xm2\_t a*, *uint16xm2\_t b*, unsigned int *gvl*)
- *e16xm4\_t vmseqvv\_e16xm4\_int16xm4* (*int16xm4\_t a*, *int16xm4\_t b*, unsigned int *gvl*)
- *e16xm4\_t vmseqvv\_e16xm4\_uint16xm4* (*uint16xm4\_t a*, *uint16xm4\_t b*, unsigned int *gvl*)

- *e16xm8\_t vmseqvv\_e16xm8\_int16xm8* (*int16xm8\_t a, int16xm8\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmseqvv\_e16xm8\_uint16xm8* (*uint16xm8\_t a, uint16xm8\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmseqvv\_e32xm1\_int32xm1* (*int32xm1\_t a, int32xm1\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmseqvv\_e32xm1\_uint32xm1* (*uint32xm1\_t a, uint32xm1\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmseqvv\_e32xm2\_int32xm2* (*int32xm2\_t a, int32xm2\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmseqvv\_e32xm2\_uint32xm2* (*uint32xm2\_t a, uint32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmseqvv\_e32xm4\_int32xm4* (*int32xm4\_t a, int32xm4\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmseqvv\_e32xm4\_uint32xm4* (*uint32xm4\_t a, uint32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmseqvv\_e32xm8\_int32xm8* (*int32xm8\_t a, int32xm8\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmseqvv\_e32xm8\_uint32xm8* (*uint32xm8\_t a, uint32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmseqvv\_e64xm1\_int64xm1* (*int64xm1\_t a, int64xm1\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmseqvv\_e64xm1\_uint64xm1* (*uint64xm1\_t a, uint64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmseqvv\_e64xm2\_int64xm2* (*int64xm2\_t a, int64xm2\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmseqvv\_e64xm2\_uint64xm2* (*uint64xm2\_t a, uint64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmseqvv\_e64xm4\_int64xm4* (*int64xm4\_t a, int64xm4\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmseqvv\_e64xm4\_uint64xm4* (*uint64xm4\_t a, uint64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmseqvv\_e64xm8\_int64xm8* (*int64xm8\_t a, int64xm8\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmseqvv\_e64xm8\_uint64xm8* (*uint64xm8\_t a, uint64xm8\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmseqvv\_e8xm1\_int8xm1* (*int8xm1\_t a, int8xm1\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmseqvv\_e8xm1\_uint8xm1* (*uint8xm1\_t a, uint8xm1\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmseqvv\_e8xm2\_int8xm2* (*int8xm2\_t a, int8xm2\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmseqvv\_e8xm2\_uint8xm2* (*uint8xm2\_t a, uint8xm2\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmseqvv\_e8xm4\_int8xm4* (*int8xm4\_t a, int8xm4\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmseqvv\_e8xm4\_uint8xm4* (*uint8xm4\_t a, uint8xm4\_t b*, unsigned int *gvl*)
- *e8xm8\_t vmseqvv\_e8xm8\_int8xm8* (*int8xm8\_t a, int8xm8\_t b*, unsigned int *gvl*)
- *e8xm8\_t vmseqvv\_e8xm8\_uint8xm8* (*uint8xm8\_t a, uint8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] == b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmseqvv\_mask\_e16xm1\_int16xm1* (*e16xm1\_t merge, int16xm1\_t a, int16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm1\_t vmseqvv\_mask\_e16xm1\_uint16xm1* (*e16xm1\_t merge, uint16xm1\_t a, uint16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmseqvv\_mask\_e16xm2\_int16xm2* (*e16xm2\_t merge, int16xm2\_t a, int16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)

- *e16xm2\_t vmseqvv\_mask\_e16xm2\_uint16xm2* (*e16xm2\_t* merge, *uint16xm2\_t* a, *uint16xm2\_t* b, *e16xm2\_t* mask, unsigned int gvl)
- *e16xm4\_t vmseqvv\_mask\_e16xm4\_int16xm4* (*e16xm4\_t* merge, *int16xm4\_t* a, *int16xm4\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *e16xm4\_t vmseqvv\_mask\_e16xm4\_uint16xm4* (*e16xm4\_t* merge, *uint16xm4\_t* a, *uint16xm4\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *e16xm8\_t vmseqvv\_mask\_e16xm8\_int16xm8* (*e16xm8\_t* merge, *int16xm8\_t* a, *int16xm8\_t* b, *e16xm8\_t* mask, unsigned int gvl)
- *e16xm8\_t vmseqvv\_mask\_e16xm8\_uint16xm8* (*e16xm8\_t* merge, *uint16xm8\_t* a, *uint16xm8\_t* b, *e16xm8\_t* mask, unsigned int gvl)
- *e32xm1\_t vmseqvv\_mask\_e32xm1\_int32xm1* (*e32xm1\_t* merge, *int32xm1\_t* a, *int32xm1\_t* b, *e32xm1\_t* mask, unsigned int gvl)
- *e32xm1\_t vmseqvv\_mask\_e32xm1\_uint32xm1* (*e32xm1\_t* merge, *uint32xm1\_t* a, *uint32xm1\_t* b, *e32xm1\_t* mask, unsigned int gvl)
- *e32xm2\_t vmseqvv\_mask\_e32xm2\_int32xm2* (*e32xm2\_t* merge, *int32xm2\_t* a, *int32xm2\_t* b, *e32xm2\_t* mask, unsigned int gvl)
- *e32xm2\_t vmseqvv\_mask\_e32xm2\_uint32xm2* (*e32xm2\_t* merge, *uint32xm2\_t* a, *uint32xm2\_t* b, *e32xm2\_t* mask, unsigned int gvl)
- *e32xm4\_t vmseqvv\_mask\_e32xm4\_int32xm4* (*e32xm4\_t* merge, *int32xm4\_t* a, *int32xm4\_t* b, *e32xm4\_t* mask, unsigned int gvl)
- *e32xm4\_t vmseqvv\_mask\_e32xm4\_uint32xm4* (*e32xm4\_t* merge, *uint32xm4\_t* a, *uint32xm4\_t* b, *e32xm4\_t* mask, unsigned int gvl)
- *e32xm8\_t vmseqvv\_mask\_e32xm8\_int32xm8* (*e32xm8\_t* merge, *int32xm8\_t* a, *int32xm8\_t* b, *e32xm8\_t* mask, unsigned int gvl)
- *e32xm8\_t vmseqvv\_mask\_e32xm8\_uint32xm8* (*e32xm8\_t* merge, *uint32xm8\_t* a, *uint32xm8\_t* b, *e32xm8\_t* mask, unsigned int gvl)
- *e64xm1\_t vmseqvv\_mask\_e64xm1\_int64xm1* (*e64xm1\_t* merge, *int64xm1\_t* a, *int64xm1\_t* b, *e64xm1\_t* mask, unsigned int gvl)
- *e64xm1\_t vmseqvv\_mask\_e64xm1\_uint64xm1* (*e64xm1\_t* merge, *uint64xm1\_t* a, *uint64xm1\_t* b, *e64xm1\_t* mask, unsigned int gvl)
- *e64xm2\_t vmseqvv\_mask\_e64xm2\_int64xm2* (*e64xm2\_t* merge, *int64xm2\_t* a, *int64xm2\_t* b, *e64xm2\_t* mask, unsigned int gvl)
- *e64xm2\_t vmseqvv\_mask\_e64xm2\_uint64xm2* (*e64xm2\_t* merge, *uint64xm2\_t* a, *uint64xm2\_t* b, *e64xm2\_t* mask, unsigned int gvl)
- *e64xm4\_t vmseqvv\_mask\_e64xm4\_int64xm4* (*e64xm4\_t* merge, *int64xm4\_t* a, *int64xm4\_t* b, *e64xm4\_t* mask, unsigned int gvl)
- *e64xm4\_t vmseqvv\_mask\_e64xm4\_uint64xm4* (*e64xm4\_t* merge, *uint64xm4\_t* a, *uint64xm4\_t* b, *e64xm4\_t* mask, unsigned int gvl)
- *e64xm8\_t vmseqvv\_mask\_e64xm8\_int64xm8* (*e64xm8\_t* merge, *int64xm8\_t* a, *int64xm8\_t* b, *e64xm8\_t* mask, unsigned int gvl)
- *e64xm8\_t vmseqvv\_mask\_e64xm8\_uint64xm8* (*e64xm8\_t* merge, *uint64xm8\_t* a, *uint64xm8\_t* b, *e64xm8\_t* mask, unsigned int gvl)
- *e8xm1\_t vmseqvv\_mask\_e8xm1\_int8xm1* (*e8xm1\_t* merge, *int8xm1\_t* a, *int8xm1\_t* b, *e8xm1\_t* mask, unsigned int gvl)
- *e8xm1\_t vmseqvv\_mask\_e8xm1\_uint8xm1* (*e8xm1\_t* merge, *uint8xm1\_t* a, *uint8xm1\_t* b, *e8xm1\_t* mask, unsigned int gvl)



- `e8xm2_t vmseqvv_mask_e8xm2_int8xm2` (`e8xm2_t merge`, `int8xm2_t a`, `int8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `e8xm2_t vmseqvv_mask_e8xm2_uint8xm2` (`e8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `e8xm4_t vmseqvv_mask_e8xm4_int8xm4` (`e8xm4_t merge`, `int8xm4_t a`, `int8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `e8xm4_t vmseqvv_mask_e8xm4_uint8xm4` (`e8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `e8xm8_t vmseqvv_mask_e8xm8_int8xm8` (`e8xm8_t merge`, `int8xm8_t a`, `int8xm8_t b`, `e8xm8_t mask`, unsigned int `gvl`)
- `e8xm8_t vmseqvv_mask_e8xm8_uint8xm8` (`e8xm8_t merge`, `uint8xm8_t a`, `uint8xm8_t b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = a[element] == b[element]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.8.3 Compare elementwise vector-scalar for equality

**Instruction:** [`'vmseq.vx'`]**Prototypes:**

- `e16xm1_t vmseqvx_e16xm1_int16xm1` (`int16xm1_t a`, short `b`, unsigned int `gvl`)
- `e16xm1_t vmseqvx_e16xm1_uint16xm1` (`uint16xm1_t a`, unsigned short `b`, unsigned int `gvl`)
- `e16xm2_t vmseqvx_e16xm2_int16xm2` (`int16xm2_t a`, short `b`, unsigned int `gvl`)
- `e16xm2_t vmseqvx_e16xm2_uint16xm2` (`uint16xm2_t a`, unsigned short `b`, unsigned int `gvl`)
- `e16xm4_t vmseqvx_e16xm4_int16xm4` (`int16xm4_t a`, short `b`, unsigned int `gvl`)
- `e16xm4_t vmseqvx_e16xm4_uint16xm4` (`uint16xm4_t a`, unsigned short `b`, unsigned int `gvl`)
- `e16xm8_t vmseqvx_e16xm8_int16xm8` (`int16xm8_t a`, short `b`, unsigned int `gvl`)
- `e16xm8_t vmseqvx_e16xm8_uint16xm8` (`uint16xm8_t a`, unsigned short `b`, unsigned int `gvl`)
- `e32xm1_t vmseqvx_e32xm1_int32xm1` (`int32xm1_t a`, int `b`, unsigned int `gvl`)
- `e32xm1_t vmseqvx_e32xm1_uint32xm1` (`uint32xm1_t a`, unsigned int `b`, unsigned int `gvl`)
- `e32xm2_t vmseqvx_e32xm2_int32xm2` (`int32xm2_t a`, int `b`, unsigned int `gvl`)
- `e32xm2_t vmseqvx_e32xm2_uint32xm2` (`uint32xm2_t a`, unsigned int `b`, unsigned int `gvl`)
- `e32xm4_t vmseqvx_e32xm4_int32xm4` (`int32xm4_t a`, int `b`, unsigned int `gvl`)
- `e32xm4_t vmseqvx_e32xm4_uint32xm4` (`uint32xm4_t a`, unsigned int `b`, unsigned int `gvl`)
- `e32xm8_t vmseqvx_e32xm8_int32xm8` (`int32xm8_t a`, int `b`, unsigned int `gvl`)
- `e32xm8_t vmseqvx_e32xm8_uint32xm8` (`uint32xm8_t a`, unsigned int `b`, unsigned int `gvl`)
- `e64xm1_t vmseqvx_e64xm1_int64xm1` (`int64xm1_t a`, long `b`, unsigned int `gvl`)



- *e64xm1\_t vmseqvx\_e64xm1\_uint64xm1* (*uint64xm1\_t a*, unsigned long *b*, unsigned int *gvl*)
- *e64xm2\_t vmseqvx\_e64xm2\_int64xm2* (*int64xm2\_t a*, long *b*, unsigned int *gvl*)
- *e64xm2\_t vmseqvx\_e64xm2\_uint64xm2* (*uint64xm2\_t a*, unsigned long *b*, unsigned int *gvl*)
- *e64xm4\_t vmseqvx\_e64xm4\_int64xm4* (*int64xm4\_t a*, long *b*, unsigned int *gvl*)
- *e64xm4\_t vmseqvx\_e64xm4\_uint64xm4* (*uint64xm4\_t a*, unsigned long *b*, unsigned int *gvl*)
- *e64xm8\_t vmseqvx\_e64xm8\_int64xm8* (*int64xm8\_t a*, long *b*, unsigned int *gvl*)
- *e64xm8\_t vmseqvx\_e64xm8\_uint64xm8* (*uint64xm8\_t a*, unsigned long *b*, unsigned int *gvl*)
- *e8xm1\_t vmseqvx\_e8xm1\_int8xm1* (*int8xm1\_t a*, signed char *b*, unsigned int *gvl*)
- *e8xm1\_t vmseqvx\_e8xm1\_uint8xm1* (*uint8xm1\_t a*, unsigned char *b*, unsigned int *gvl*)
- *e8xm2\_t vmseqvx\_e8xm2\_int8xm2* (*int8xm2\_t a*, signed char *b*, unsigned int *gvl*)
- *e8xm2\_t vmseqvx\_e8xm2\_uint8xm2* (*uint8xm2\_t a*, unsigned char *b*, unsigned int *gvl*)
- *e8xm4\_t vmseqvx\_e8xm4\_int8xm4* (*int8xm4\_t a*, signed char *b*, unsigned int *gvl*)
- *e8xm4\_t vmseqvx\_e8xm4\_uint8xm4* (*uint8xm4\_t a*, unsigned char *b*, unsigned int *gvl*)
- *e8xm8\_t vmseqvx\_e8xm8\_int8xm8* (*int8xm8\_t a*, signed char *b*, unsigned int *gvl*)
- *e8xm8\_t vmseqvx\_e8xm8\_uint8xm8* (*uint8xm8\_t a*, unsigned char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] == b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmseqvx\_mask\_e16xm1\_int16xm1* (*e16xm1\_t merge*, *int16xm1\_t a*, short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm1\_t vmseqvx\_mask\_e16xm1\_uint16xm1* (*e16xm1\_t merge*, *uint16xm1\_t a*, unsigned short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmseqvx\_mask\_e16xm2\_int16xm2* (*e16xm2\_t merge*, *int16xm2\_t a*, short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmseqvx\_mask\_e16xm2\_uint16xm2* (*e16xm2\_t merge*, *uint16xm2\_t a*, unsigned short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmseqvx\_mask\_e16xm4\_int16xm4* (*e16xm4\_t merge*, *int16xm4\_t a*, short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmseqvx\_mask\_e16xm4\_uint16xm4* (*e16xm4\_t merge*, *uint16xm4\_t a*, unsigned short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmseqvx\_mask\_e16xm8\_int16xm8* (*e16xm8\_t merge*, *int16xm8\_t a*, short *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmseqvx\_mask\_e16xm8\_uint16xm8* (*e16xm8\_t merge*, *uint16xm8\_t a*, unsigned short *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmseqvx\_mask\_e32xm1\_int32xm1* (*e32xm1\_t merge*, *int32xm1\_t a*, int *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmseqvx\_mask\_e32xm1\_uint32xm1* (*e32xm1\_t merge*, *uint32xm1\_t a*, unsigned int *b*, *e32xm1\_t mask*, unsigned int *gvl*)

- `e32xm2_t vmseqvx_mask_e32xm2_int32xm2` (`e32xm2_t` merge, `int32xm2_t` *a*, `int` *b*, `e32xm2_t` mask, unsigned int gvl)
- `e32xm2_t vmseqvx_mask_e32xm2_uint32xm2` (`e32xm2_t` merge, `uint32xm2_t` *a*, unsigned int *b*, `e32xm2_t` mask, unsigned int gvl)
- `e32xm4_t vmseqvx_mask_e32xm4_int32xm4` (`e32xm4_t` merge, `int32xm4_t` *a*, `int` *b*, `e32xm4_t` mask, unsigned int gvl)
- `e32xm4_t vmseqvx_mask_e32xm4_uint32xm4` (`e32xm4_t` merge, `uint32xm4_t` *a*, unsigned int *b*, `e32xm4_t` mask, unsigned int gvl)
- `e32xm8_t vmseqvx_mask_e32xm8_int32xm8` (`e32xm8_t` merge, `int32xm8_t` *a*, `int` *b*, `e32xm8_t` mask, unsigned int gvl)
- `e32xm8_t vmseqvx_mask_e32xm8_uint32xm8` (`e32xm8_t` merge, `uint32xm8_t` *a*, unsigned int *b*, `e32xm8_t` mask, unsigned int gvl)
- `e64xm1_t vmseqvx_mask_e64xm1_int64xm1` (`e64xm1_t` merge, `int64xm1_t` *a*, `long` *b*, `e64xm1_t` mask, unsigned int gvl)
- `e64xm1_t vmseqvx_mask_e64xm1_uint64xm1` (`e64xm1_t` merge, `uint64xm1_t` *a*, unsigned long *b*, `e64xm1_t` mask, unsigned int gvl)
- `e64xm2_t vmseqvx_mask_e64xm2_int64xm2` (`e64xm2_t` merge, `int64xm2_t` *a*, `long` *b*, `e64xm2_t` mask, unsigned int gvl)
- `e64xm2_t vmseqvx_mask_e64xm2_uint64xm2` (`e64xm2_t` merge, `uint64xm2_t` *a*, unsigned long *b*, `e64xm2_t` mask, unsigned int gvl)
- `e64xm4_t vmseqvx_mask_e64xm4_int64xm4` (`e64xm4_t` merge, `int64xm4_t` *a*, `long` *b*, `e64xm4_t` mask, unsigned int gvl)
- `e64xm4_t vmseqvx_mask_e64xm4_uint64xm4` (`e64xm4_t` merge, `uint64xm4_t` *a*, unsigned long *b*, `e64xm4_t` mask, unsigned int gvl)
- `e64xm8_t vmseqvx_mask_e64xm8_int64xm8` (`e64xm8_t` merge, `int64xm8_t` *a*, `long` *b*, `e64xm8_t` mask, unsigned int gvl)
- `e64xm8_t vmseqvx_mask_e64xm8_uint64xm8` (`e64xm8_t` merge, `uint64xm8_t` *a*, unsigned long *b*, `e64xm8_t` mask, unsigned int gvl)
- `e8xm1_t vmseqvx_mask_e8xm1_int8xm1` (`e8xm1_t` merge, `int8xm1_t` *a*, `signed char` *b*, `e8xm1_t` mask, unsigned int gvl)
- `e8xm1_t vmseqvx_mask_e8xm1_uint8xm1` (`e8xm1_t` merge, `uint8xm1_t` *a*, unsigned char *b*, `e8xm1_t` mask, unsigned int gvl)
- `e8xm2_t vmseqvx_mask_e8xm2_int8xm2` (`e8xm2_t` merge, `int8xm2_t` *a*, `signed char` *b*, `e8xm2_t` mask, unsigned int gvl)
- `e8xm2_t vmseqvx_mask_e8xm2_uint8xm2` (`e8xm2_t` merge, `uint8xm2_t` *a*, unsigned char *b*, `e8xm2_t` mask, unsigned int gvl)
- `e8xm4_t vmseqvx_mask_e8xm4_int8xm4` (`e8xm4_t` merge, `int8xm4_t` *a*, `signed char` *b*, `e8xm4_t` mask, unsigned int gvl)
- `e8xm4_t vmseqvx_mask_e8xm4_uint8xm4` (`e8xm4_t` merge, `uint8xm4_t` *a*, unsigned char *b*, `e8xm4_t` mask, unsigned int gvl)
- `e8xm8_t vmseqvx_mask_e8xm8_int8xm8` (`e8xm8_t` merge, `int8xm8_t` *a*, `signed char` *b*, `e8xm8_t` mask, unsigned int gvl)
- `e8xm8_t vmseqvx_mask_e8xm8_uint8xm8` (`e8xm8_t` merge, `uint8xm8_t` *a*, unsigned char *b*, `e8xm8_t` mask, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] == b
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.8.4 Compare elementwise signed integer one vector and one scalar for greater-than-or-equal

Instruction: ['vmsge.vx']

Prototypes:

- *e16xm1\_t vmsgevx\_e16xm1\_int16xm1* (*int16xm1\_t a*, short *b*, unsigned int *gvl*)
- *e16xm2\_t vmsgevx\_e16xm2\_int16xm2* (*int16xm2\_t a*, short *b*, unsigned int *gvl*)
- *e16xm4\_t vmsgevx\_e16xm4\_int16xm4* (*int16xm4\_t a*, short *b*, unsigned int *gvl*)
- *e16xm8\_t vmsgevx\_e16xm8\_int16xm8* (*int16xm8\_t a*, short *b*, unsigned int *gvl*)
- *e32xm1\_t vmsgevx\_e32xm1\_int32xm1* (*int32xm1\_t a*, int *b*, unsigned int *gvl*)
- *e32xm2\_t vmsgevx\_e32xm2\_int32xm2* (*int32xm2\_t a*, int *b*, unsigned int *gvl*)
- *e32xm4\_t vmsgevx\_e32xm4\_int32xm4* (*int32xm4\_t a*, int *b*, unsigned int *gvl*)
- *e32xm8\_t vmsgevx\_e32xm8\_int32xm8* (*int32xm8\_t a*, int *b*, unsigned int *gvl*)
- *e64xm1\_t vmsgevx\_e64xm1\_int64xm1* (*int64xm1\_t a*, long *b*, unsigned int *gvl*)
- *e64xm2\_t vmsgevx\_e64xm2\_int64xm2* (*int64xm2\_t a*, long *b*, unsigned int *gvl*)
- *e64xm4\_t vmsgevx\_e64xm4\_int64xm4* (*int64xm4\_t a*, long *b*, unsigned int *gvl*)
- *e64xm8\_t vmsgevx\_e64xm8\_int64xm8* (*int64xm8\_t a*, long *b*, unsigned int *gvl*)
- *e8xm1\_t vmsgevx\_e8xm1\_int8xm1* (*int8xm1\_t a*, signed char *b*, unsigned int *gvl*)
- *e8xm2\_t vmsgevx\_e8xm2\_int8xm2* (*int8xm2\_t a*, signed char *b*, unsigned int *gvl*)
- *e8xm4\_t vmsgevx\_e8xm4\_int8xm4* (*int8xm4\_t a*, signed char *b*, unsigned int *gvl*)
- *e8xm8\_t vmsgevx\_e8xm8\_int8xm8* (*int8xm8\_t a*, signed char *b*, unsigned int *gvl*)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] >= b
      result[gvl : VLMAX] = 0
```

Masked prototypes:

- *e16xm1\_t vmsgevx\_mask\_e16xm1\_int16xm1* (*e16xm1\_t merge*, *int16xm1\_t a*, short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmsgevx\_mask\_e16xm2\_int16xm2* (*e16xm2\_t merge*, *int16xm2\_t a*, short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmsgevx\_mask\_e16xm4\_int16xm4* (*e16xm4\_t merge*, *int16xm4\_t a*, short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmsgevx\_mask\_e16xm8\_int16xm8* (*e16xm8\_t merge*, *int16xm8\_t a*, short *b*, *e16xm8\_t mask*, unsigned int *gvl*)

- `e32xm1_t vmsgevx_mask_e32xm1_int32xm1` (`e32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl`)
- `e32xm2_t vmsgevx_mask_e32xm2_int32xm2` (`e32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl`)
- `e32xm4_t vmsgevx_mask_e32xm4_int32xm4` (`e32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl`)
- `e32xm8_t vmsgevx_mask_e32xm8_int32xm8` (`e32xm8_t merge, int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl`)
- `e64xm1_t vmsgevx_mask_e64xm1_int64xm1` (`e64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl`)
- `e64xm2_t vmsgevx_mask_e64xm2_int64xm2` (`e64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl`)
- `e64xm4_t vmsgevx_mask_e64xm4_int64xm4` (`e64xm4_t merge, int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl`)
- `e64xm8_t vmsgevx_mask_e64xm8_int64xm8` (`e64xm8_t merge, int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl`)
- `e8xm1_t vmsgevx_mask_e8xm1_int8xm1` (`e8xm1_t merge, int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl`)
- `e8xm2_t vmsgevx_mask_e8xm2_int8xm2` (`e8xm2_t merge, int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl`)
- `e8xm4_t vmsgevx_mask_e8xm4_int8xm4` (`e8xm4_t merge, int8xm4_t a, signed char b, e8xm4_t mask, unsigned int gvl`)
- `e8xm8_t vmsgevx_mask_e8xm8_int8xm8` (`e8xm8_t merge, int8xm8_t a, signed char b, e8xm8_t mask, unsigned int gvl`)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = a[element] >= b
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.8.5 Compare elementwise unsigned integer one vector and one scalar for greater-than-or-equal

**Instruction:** [`'vmsgeu.vx'`]

#### Prototypes:

- `e16xm1_t vmsgeuvx_e16xm1_uint16xm1` (`uint16xm1_t a, unsigned short b, unsigned int gvl`)
- `e16xm2_t vmsgeuvx_e16xm2_uint16xm2` (`uint16xm2_t a, unsigned short b, unsigned int gvl`)
- `e16xm4_t vmsgeuvx_e16xm4_uint16xm4` (`uint16xm4_t a, unsigned short b, unsigned int gvl`)
- `e16xm8_t vmsgeuvx_e16xm8_uint16xm8` (`uint16xm8_t a, unsigned short b, unsigned int gvl`)
- `e32xm1_t vmsgeuvx_e32xm1_uint32xm1` (`uint32xm1_t a, unsigned int b, unsigned int gvl`)
- `e32xm2_t vmsgeuvx_e32xm2_uint32xm2` (`uint32xm2_t a, unsigned int b, unsigned int gvl`)
- `e32xm4_t vmsgeuvx_e32xm4_uint32xm4` (`uint32xm4_t a, unsigned int b, unsigned int gvl`)

- *e32xm8\_t vmsgeuvx\_e32xm8\_uint32xm8* (*uint32xm8\_t a*, unsigned int *b*, unsigned int *gvl*)
- *e64xm1\_t vmsgeuvx\_e64xm1\_uint64xm1* (*uint64xm1\_t a*, unsigned long *b*, unsigned int *gvl*)
- *e64xm2\_t vmsgeuvx\_e64xm2\_uint64xm2* (*uint64xm2\_t a*, unsigned long *b*, unsigned int *gvl*)
- *e64xm4\_t vmsgeuvx\_e64xm4\_uint64xm4* (*uint64xm4\_t a*, unsigned long *b*, unsigned int *gvl*)
- *e64xm8\_t vmsgeuvx\_e64xm8\_uint64xm8* (*uint64xm8\_t a*, unsigned long *b*, unsigned int *gvl*)
- *e8xm1\_t vmsgeuvx\_e8xm1\_uint8xm1* (*uint8xm1\_t a*, unsigned char *b*, unsigned int *gvl*)
- *e8xm2\_t vmsgeuvx\_e8xm2\_uint8xm2* (*uint8xm2\_t a*, unsigned char *b*, unsigned int *gvl*)
- *e8xm4\_t vmsgeuvx\_e8xm4\_uint8xm4* (*uint8xm4\_t a*, unsigned char *b*, unsigned int *gvl*)
- *e8xm8\_t vmsgeuvx\_e8xm8\_uint8xm8* (*uint8xm8\_t a*, unsigned char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] >= b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmsgeuvx\_mask\_e16xm1\_uint16xm1* (*e16xm1\_t merge*, *uint16xm1\_t a*, unsigned short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmsgeuvx\_mask\_e16xm2\_uint16xm2* (*e16xm2\_t merge*, *uint16xm2\_t a*, unsigned short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmsgeuvx\_mask\_e16xm4\_uint16xm4* (*e16xm4\_t merge*, *uint16xm4\_t a*, unsigned short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmsgeuvx\_mask\_e16xm8\_uint16xm8* (*e16xm8\_t merge*, *uint16xm8\_t a*, unsigned short *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmsgeuvx\_mask\_e32xm1\_uint32xm1* (*e32xm1\_t merge*, *uint32xm1\_t a*, unsigned int *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm2\_t vmsgeuvx\_mask\_e32xm2\_uint32xm2* (*e32xm2\_t merge*, *uint32xm2\_t a*, unsigned int *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *e32xm4\_t vmsgeuvx\_mask\_e32xm4\_uint32xm4* (*e32xm4\_t merge*, *uint32xm4\_t a*, unsigned int *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *e32xm8\_t vmsgeuvx\_mask\_e32xm8\_uint32xm8* (*e32xm8\_t merge*, *uint32xm8\_t a*, unsigned int *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *e64xm1\_t vmsgeuvx\_mask\_e64xm1\_uint64xm1* (*e64xm1\_t merge*, *uint64xm1\_t a*, unsigned long *b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *e64xm2\_t vmsgeuvx\_mask\_e64xm2\_uint64xm2* (*e64xm2\_t merge*, *uint64xm2\_t a*, unsigned long *b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *e64xm4\_t vmsgeuvx\_mask\_e64xm4\_uint64xm4* (*e64xm4\_t merge*, *uint64xm4\_t a*, unsigned long *b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *e64xm8\_t vmsgeuvx\_mask\_e64xm8\_uint64xm8* (*e64xm8\_t merge*, *uint64xm8\_t a*, unsigned long *b*, *e64xm8\_t mask*, unsigned int *gvl*)
- *e8xm1\_t vmsgeuvx\_mask\_e8xm1\_uint8xm1* (*e8xm1\_t merge*, *uint8xm1\_t a*, unsigned char *b*, *e8xm1\_t mask*, unsigned int *gvl*)
- *e8xm2\_t vmsgeuvx\_mask\_e8xm2\_uint8xm2* (*e8xm2\_t merge*, *uint8xm2\_t a*, unsigned char *b*, *e8xm2\_t mask*, unsigned int *gvl*)

- `e8xm4_t vmsgeuvx_mask_e8xm4_uint8xm4` (`e8xm4_t merge`, `uint8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `e8xm8_t vmsgeuvx_mask_e8xm8_uint8xm8` (`e8xm8_t merge`, `uint8xm8_t a`, unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] >= b
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.8.6 Compare elementwise signed integer one vector and one scalar immediate for greater-than

Instruction: ['vmsgt.vi']

Prototypes:

- `e16xm1_t vmsgtvi_e16xm1_int16xm1` (`int16xm1_t a`, const short `b`, unsigned int `gvl`)
- `e16xm2_t vmsgtvi_e16xm2_int16xm2` (`int16xm2_t a`, const short `b`, unsigned int `gvl`)
- `e16xm4_t vmsgtvi_e16xm4_int16xm4` (`int16xm4_t a`, const short `b`, unsigned int `gvl`)
- `e16xm8_t vmsgtvi_e16xm8_int16xm8` (`int16xm8_t a`, const short `b`, unsigned int `gvl`)
- `e32xm1_t vmsgtvi_e32xm1_int32xm1` (`int32xm1_t a`, const int `b`, unsigned int `gvl`)
- `e32xm2_t vmsgtvi_e32xm2_int32xm2` (`int32xm2_t a`, const int `b`, unsigned int `gvl`)
- `e32xm4_t vmsgtvi_e32xm4_int32xm4` (`int32xm4_t a`, const int `b`, unsigned int `gvl`)
- `e32xm8_t vmsgtvi_e32xm8_int32xm8` (`int32xm8_t a`, const int `b`, unsigned int `gvl`)
- `e64xm1_t vmsgtvi_e64xm1_int64xm1` (`int64xm1_t a`, const long `b`, unsigned int `gvl`)
- `e64xm2_t vmsgtvi_e64xm2_int64xm2` (`int64xm2_t a`, const long `b`, unsigned int `gvl`)
- `e64xm4_t vmsgtvi_e64xm4_int64xm4` (`int64xm4_t a`, const long `b`, unsigned int `gvl`)
- `e64xm8_t vmsgtvi_e64xm8_int64xm8` (`int64xm8_t a`, const long `b`, unsigned int `gvl`)
- `e8xm1_t vmsgtvi_e8xm1_int8xm1` (`int8xm1_t a`, const signed char `b`, unsigned int `gvl`)
- `e8xm2_t vmsgtvi_e8xm2_int8xm2` (`int8xm2_t a`, const signed char `b`, unsigned int `gvl`)
- `e8xm4_t vmsgtvi_e8xm4_int8xm4` (`int8xm4_t a`, const signed char `b`, unsigned int `gvl`)
- `e8xm8_t vmsgtvi_e8xm8_int8xm8` (`int8xm8_t a`, const signed char `b`, unsigned int `gvl`)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] > b
      result[gvl : VLMAX] = 0
```

Masked prototypes:

- `e16xm1_t vmsgtvi_mask_e16xm1_int16xm1` (`e16xm1_t merge`, `int16xm1_t a`, const short `b`, `e16xm1_t mask`, unsigned int `gvl`)



- `e16xm2_t vmsgtvi_mask_e16xm2_int16xm2` (`e16xm2_t` merge, `int16xm2_t` *a*, const short *b*, `e16xm2_t` mask, unsigned int *gvl*)
- `e16xm4_t vmsgtvi_mask_e16xm4_int16xm4` (`e16xm4_t` merge, `int16xm4_t` *a*, const short *b*, `e16xm4_t` mask, unsigned int *gvl*)
- `e16xm8_t vmsgtvi_mask_e16xm8_int16xm8` (`e16xm8_t` merge, `int16xm8_t` *a*, const short *b*, `e16xm8_t` mask, unsigned int *gvl*)
- `e32xm1_t vmsgtvi_mask_e32xm1_int32xm1` (`e32xm1_t` merge, `int32xm1_t` *a*, const int *b*, `e32xm1_t` mask, unsigned int *gvl*)
- `e32xm2_t vmsgtvi_mask_e32xm2_int32xm2` (`e32xm2_t` merge, `int32xm2_t` *a*, const int *b*, `e32xm2_t` mask, unsigned int *gvl*)
- `e32xm4_t vmsgtvi_mask_e32xm4_int32xm4` (`e32xm4_t` merge, `int32xm4_t` *a*, const int *b*, `e32xm4_t` mask, unsigned int *gvl*)
- `e32xm8_t vmsgtvi_mask_e32xm8_int32xm8` (`e32xm8_t` merge, `int32xm8_t` *a*, const int *b*, `e32xm8_t` mask, unsigned int *gvl*)
- `e64xm1_t vmsgtvi_mask_e64xm1_int64xm1` (`e64xm1_t` merge, `int64xm1_t` *a*, const long *b*, `e64xm1_t` mask, unsigned int *gvl*)
- `e64xm2_t vmsgtvi_mask_e64xm2_int64xm2` (`e64xm2_t` merge, `int64xm2_t` *a*, const long *b*, `e64xm2_t` mask, unsigned int *gvl*)
- `e64xm4_t vmsgtvi_mask_e64xm4_int64xm4` (`e64xm4_t` merge, `int64xm4_t` *a*, const long *b*, `e64xm4_t` mask, unsigned int *gvl*)
- `e64xm8_t vmsgtvi_mask_e64xm8_int64xm8` (`e64xm8_t` merge, `int64xm8_t` *a*, const long *b*, `e64xm8_t` mask, unsigned int *gvl*)
- `e8xm1_t vmsgtvi_mask_e8xm1_int8xm1` (`e8xm1_t` merge, `int8xm1_t` *a*, const signed char *b*, `e8xm1_t` mask, unsigned int *gvl*)
- `e8xm2_t vmsgtvi_mask_e8xm2_int8xm2` (`e8xm2_t` merge, `int8xm2_t` *a*, const signed char *b*, `e8xm2_t` mask, unsigned int *gvl*)
- `e8xm4_t vmsgtvi_mask_e8xm4_int8xm4` (`e8xm4_t` merge, `int8xm4_t` *a*, const signed char *b*, `e8xm4_t` mask, unsigned int *gvl*)
- `e8xm8_t vmsgtvi_mask_e8xm8_int8xm8` (`e8xm8_t` merge, `int8xm8_t` *a*, const signed char *b*, `e8xm8_t` mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] > b
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

**2.8.7 Compare elementwise signed integer one vector and one scalar for greater-than****Instruction:** [`'vmsgt.vx'`]**Prototypes:**

- `e16xm1_t vmsgtvx_e16xm1_int16xm1` (`int16xm1_t` *a*, short *b*, unsigned int *gvl*)
- `e16xm2_t vmsgtvx_e16xm2_int16xm2` (`int16xm2_t` *a*, short *b*, unsigned int *gvl*)

- `e16xm4_t vmsgtvx_e16xm4_int16xm4 (int16xm4_t a, short b, unsigned int gvl)`
- `e16xm8_t vmsgtvx_e16xm8_int16xm8 (int16xm8_t a, short b, unsigned int gvl)`
- `e32xm1_t vmsgtvx_e32xm1_int32xm1 (int32xm1_t a, int b, unsigned int gvl)`
- `e32xm2_t vmsgtvx_e32xm2_int32xm2 (int32xm2_t a, int b, unsigned int gvl)`
- `e32xm4_t vmsgtvx_e32xm4_int32xm4 (int32xm4_t a, int b, unsigned int gvl)`
- `e32xm8_t vmsgtvx_e32xm8_int32xm8 (int32xm8_t a, int b, unsigned int gvl)`
- `e64xm1_t vmsgtvx_e64xm1_int64xm1 (int64xm1_t a, long b, unsigned int gvl)`
- `e64xm2_t vmsgtvx_e64xm2_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `e64xm4_t vmsgtvx_e64xm4_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `e64xm8_t vmsgtvx_e64xm8_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `e8xm1_t vmsgtvx_e8xm1_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `e8xm2_t vmsgtvx_e8xm2_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `e8xm4_t vmsgtvx_e8xm4_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `e8xm8_t vmsgtvx_e8xm8_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] > b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `e16xm1_t vmsgtvx_mask_e16xm1_int16xm1 (e16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `e16xm2_t vmsgtvx_mask_e16xm2_int16xm2 (e16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `e16xm4_t vmsgtvx_mask_e16xm4_int16xm4 (e16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `e16xm8_t vmsgtvx_mask_e16xm8_int16xm8 (e16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `e32xm1_t vmsgtvx_mask_e32xm1_int32xm1 (e32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `e32xm2_t vmsgtvx_mask_e32xm2_int32xm2 (e32xm2_t merge, int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `e32xm4_t vmsgtvx_mask_e32xm4_int32xm4 (e32xm4_t merge, int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `e32xm8_t vmsgtvx_mask_e32xm8_int32xm8 (e32xm8_t merge, int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `e64xm1_t vmsgtvx_mask_e64xm1_int64xm1 (e64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `e64xm2_t vmsgtvx_mask_e64xm2_int64xm2 (e64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `e64xm4_t vmsgtvx_mask_e64xm4_int64xm4 (e64xm4_t merge, int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`

- `e64xm8_t vmsgtvx_mask_e64xm8_int64xm8` (`e64xm8_t merge`, `int64xm8_t a`, long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `e8xm1_t vmsgtvx_mask_e8xm1_int8xm1` (`e8xm1_t merge`, `int8xm1_t a`, signed char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `e8xm2_t vmsgtvx_mask_e8xm2_int8xm2` (`e8xm2_t merge`, `int8xm2_t a`, signed char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `e8xm4_t vmsgtvx_mask_e8xm4_int8xm4` (`e8xm4_t merge`, `int8xm4_t a`, signed char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `e8xm8_t vmsgtvx_mask_e8xm8_int8xm8` (`e8xm8_t merge`, `int8xm8_t a`, signed char `b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] > b
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

**2.8.8 Compare elementwise unsigned integer one vector and one scalar immediate for greater-than****Instruction:** [`vmsgtuvi`]**Prototypes:**

- `e16xm1_t vmsgtuvi_e16xm1_uint16xm1` (`uint16xm1_t a`, const unsigned short `b`, unsigned int `gvl`)
- `e16xm2_t vmsgtuvi_e16xm2_uint16xm2` (`uint16xm2_t a`, const unsigned short `b`, unsigned int `gvl`)
- `e16xm4_t vmsgtuvi_e16xm4_uint16xm4` (`uint16xm4_t a`, const unsigned short `b`, unsigned int `gvl`)
- `e16xm8_t vmsgtuvi_e16xm8_uint16xm8` (`uint16xm8_t a`, const unsigned short `b`, unsigned int `gvl`)
- `e32xm1_t vmsgtuvi_e32xm1_uint32xm1` (`uint32xm1_t a`, const unsigned int `b`, unsigned int `gvl`)
- `e32xm2_t vmsgtuvi_e32xm2_uint32xm2` (`uint32xm2_t a`, const unsigned int `b`, unsigned int `gvl`)
- `e32xm4_t vmsgtuvi_e32xm4_uint32xm4` (`uint32xm4_t a`, const unsigned int `b`, unsigned int `gvl`)
- `e32xm8_t vmsgtuvi_e32xm8_uint32xm8` (`uint32xm8_t a`, const unsigned int `b`, unsigned int `gvl`)
- `e64xm1_t vmsgtuvi_e64xm1_uint64xm1` (`uint64xm1_t a`, const unsigned long `b`, unsigned int `gvl`)
- `e64xm2_t vmsgtuvi_e64xm2_uint64xm2` (`uint64xm2_t a`, const unsigned long `b`, unsigned int `gvl`)
- `e64xm4_t vmsgtuvi_e64xm4_uint64xm4` (`uint64xm4_t a`, const unsigned long `b`, unsigned int `gvl`)
- `e64xm8_t vmsgtuvi_e64xm8_uint64xm8` (`uint64xm8_t a`, const unsigned long `b`, unsigned int `gvl`)
- `e8xm1_t vmsgtuvi_e8xm1_uint8xm1` (`uint8xm1_t a`, const unsigned char `b`, unsigned int `gvl`)
- `e8xm2_t vmsgtuvi_e8xm2_uint8xm2` (`uint8xm2_t a`, const unsigned char `b`, unsigned int `gvl`)
- `e8xm4_t vmsgtuvi_e8xm4_uint8xm4` (`uint8xm4_t a`, const unsigned char `b`, unsigned int `gvl`)
- `e8xm8_t vmsgtuvi_e8xm8_uint8xm8` (`uint8xm8_t a`, const unsigned char `b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] > b
      result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- *e16xm1\_t vmsgtuvi\_mask\_e16xm1\_uint16xm1* (*e16xm1\_t* merge, *uint16xm1\_t* *a*, const unsigned short *b*, *e16xm1\_t* mask, unsigned int *gvl*)
- *e16xm2\_t vmsgtuvi\_mask\_e16xm2\_uint16xm2* (*e16xm2\_t* merge, *uint16xm2\_t* *a*, const unsigned short *b*, *e16xm2\_t* mask, unsigned int *gvl*)
- *e16xm4\_t vmsgtuvi\_mask\_e16xm4\_uint16xm4* (*e16xm4\_t* merge, *uint16xm4\_t* *a*, const unsigned short *b*, *e16xm4\_t* mask, unsigned int *gvl*)
- *e16xm8\_t vmsgtuvi\_mask\_e16xm8\_uint16xm8* (*e16xm8\_t* merge, *uint16xm8\_t* *a*, const unsigned short *b*, *e16xm8\_t* mask, unsigned int *gvl*)
- *e32xm1\_t vmsgtuvi\_mask\_e32xm1\_uint32xm1* (*e32xm1\_t* merge, *uint32xm1\_t* *a*, const unsigned int *b*, *e32xm1\_t* mask, unsigned int *gvl*)
- *e32xm2\_t vmsgtuvi\_mask\_e32xm2\_uint32xm2* (*e32xm2\_t* merge, *uint32xm2\_t* *a*, const unsigned int *b*, *e32xm2\_t* mask, unsigned int *gvl*)
- *e32xm4\_t vmsgtuvi\_mask\_e32xm4\_uint32xm4* (*e32xm4\_t* merge, *uint32xm4\_t* *a*, const unsigned int *b*, *e32xm4\_t* mask, unsigned int *gvl*)
- *e32xm8\_t vmsgtuvi\_mask\_e32xm8\_uint32xm8* (*e32xm8\_t* merge, *uint32xm8\_t* *a*, const unsigned int *b*, *e32xm8\_t* mask, unsigned int *gvl*)
- *e64xm1\_t vmsgtuvi\_mask\_e64xm1\_uint64xm1* (*e64xm1\_t* merge, *uint64xm1\_t* *a*, const unsigned long *b*, *e64xm1\_t* mask, unsigned int *gvl*)
- *e64xm2\_t vmsgtuvi\_mask\_e64xm2\_uint64xm2* (*e64xm2\_t* merge, *uint64xm2\_t* *a*, const unsigned long *b*, *e64xm2\_t* mask, unsigned int *gvl*)
- *e64xm4\_t vmsgtuvi\_mask\_e64xm4\_uint64xm4* (*e64xm4\_t* merge, *uint64xm4\_t* *a*, const unsigned long *b*, *e64xm4\_t* mask, unsigned int *gvl*)
- *e64xm8\_t vmsgtuvi\_mask\_e64xm8\_uint64xm8* (*e64xm8\_t* merge, *uint64xm8\_t* *a*, const unsigned long *b*, *e64xm8\_t* mask, unsigned int *gvl*)
- *e8xm1\_t vmsgtuvi\_mask\_e8xm1\_uint8xm1* (*e8xm1\_t* merge, *uint8xm1\_t* *a*, const unsigned char *b*, *e8xm1\_t* mask, unsigned int *gvl*)
- *e8xm2\_t vmsgtuvi\_mask\_e8xm2\_uint8xm2* (*e8xm2\_t* merge, *uint8xm2\_t* *a*, const unsigned char *b*, *e8xm2\_t* mask, unsigned int *gvl*)
- *e8xm4\_t vmsgtuvi\_mask\_e8xm4\_uint8xm4* (*e8xm4\_t* merge, *uint8xm4\_t* *a*, const unsigned char *b*, *e8xm4\_t* mask, unsigned int *gvl*)
- *e8xm8\_t vmsgtuvi\_mask\_e8xm8\_uint8xm8* (*e8xm8\_t* merge, *uint8xm8\_t* *a*, const unsigned char *b*, *e8xm8\_t* mask, unsigned int *gvl*)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] > b
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.8.9 Compare elementwise unsigned integer one vector and one scalar for greater-than

**Instruction:** ['vmsgtuvx']

**Prototypes:**

- *e16xm1\_t vmsgtuvx\_e16xm1\_uint16xm1 (uint16xm1\_t a, unsigned short b, unsigned int gvl)*
- *e16xm2\_t vmsgtuvx\_e16xm2\_uint16xm2 (uint16xm2\_t a, unsigned short b, unsigned int gvl)*
- *e16xm4\_t vmsgtuvx\_e16xm4\_uint16xm4 (uint16xm4\_t a, unsigned short b, unsigned int gvl)*
- *e16xm8\_t vmsgtuvx\_e16xm8\_uint16xm8 (uint16xm8\_t a, unsigned short b, unsigned int gvl)*
- *e32xm1\_t vmsgtuvx\_e32xm1\_uint32xm1 (uint32xm1\_t a, unsigned int b, unsigned int gvl)*
- *e32xm2\_t vmsgtuvx\_e32xm2\_uint32xm2 (uint32xm2\_t a, unsigned int b, unsigned int gvl)*
- *e32xm4\_t vmsgtuvx\_e32xm4\_uint32xm4 (uint32xm4\_t a, unsigned int b, unsigned int gvl)*
- *e32xm8\_t vmsgtuvx\_e32xm8\_uint32xm8 (uint32xm8\_t a, unsigned int b, unsigned int gvl)*
- *e64xm1\_t vmsgtuvx\_e64xm1\_uint64xm1 (uint64xm1\_t a, unsigned long b, unsigned int gvl)*
- *e64xm2\_t vmsgtuvx\_e64xm2\_uint64xm2 (uint64xm2\_t a, unsigned long b, unsigned int gvl)*
- *e64xm4\_t vmsgtuvx\_e64xm4\_uint64xm4 (uint64xm4\_t a, unsigned long b, unsigned int gvl)*
- *e64xm8\_t vmsgtuvx\_e64xm8\_uint64xm8 (uint64xm8\_t a, unsigned long b, unsigned int gvl)*
- *e8xm1\_t vmsgtuvx\_e8xm1\_uint8xm1 (uint8xm1\_t a, unsigned char b, unsigned int gvl)*
- *e8xm2\_t vmsgtuvx\_e8xm2\_uint8xm2 (uint8xm2\_t a, unsigned char b, unsigned int gvl)*
- *e8xm4\_t vmsgtuvx\_e8xm4\_uint8xm4 (uint8xm4\_t a, unsigned char b, unsigned int gvl)*
- *e8xm8\_t vmsgtuvx\_e8xm8\_uint8xm8 (uint8xm8\_t a, unsigned char b, unsigned int gvl)*

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] > b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmsgtuvx\_mask\_e16xm1\_uint16xm1 (e16xm1\_t merge, uint16xm1\_t a, unsigned short b, e16xm1\_t mask, unsigned int gvl)*
- *e16xm2\_t vmsgtuvx\_mask\_e16xm2\_uint16xm2 (e16xm2\_t merge, uint16xm2\_t a, unsigned short b, e16xm2\_t mask, unsigned int gvl)*
- *e16xm4\_t vmsgtuvx\_mask\_e16xm4\_uint16xm4 (e16xm4\_t merge, uint16xm4\_t a, unsigned short b, e16xm4\_t mask, unsigned int gvl)*
- *e16xm8\_t vmsgtuvx\_mask\_e16xm8\_uint16xm8 (e16xm8\_t merge, uint16xm8\_t a, unsigned short b, e16xm8\_t mask, unsigned int gvl)*
- *e32xm1\_t vmsgtuvx\_mask\_e32xm1\_uint32xm1 (e32xm1\_t merge, uint32xm1\_t a, unsigned int b, e32xm1\_t mask, unsigned int gvl)*
- *e32xm2\_t vmsgtuvx\_mask\_e32xm2\_uint32xm2 (e32xm2\_t merge, uint32xm2\_t a, unsigned int b, e32xm2\_t mask, unsigned int gvl)*
- *e32xm4\_t vmsgtuvx\_mask\_e32xm4\_uint32xm4 (e32xm4\_t merge, uint32xm4\_t a, unsigned int b, e32xm4\_t mask, unsigned int gvl)*

- *e32xm8\_t vmsgtuvx\_mask\_e32xm8\_uint32xm8* (*e32xm8\_t* merge, *uint32xm8\_t* *a*, unsigned int *b*, *e32xm8\_t* mask, unsigned int *gvl*)
- *e64xm1\_t vmsgtuvx\_mask\_e64xm1\_uint64xm1* (*e64xm1\_t* merge, *uint64xm1\_t* *a*, unsigned long *b*, *e64xm1\_t* mask, unsigned int *gvl*)
- *e64xm2\_t vmsgtuvx\_mask\_e64xm2\_uint64xm2* (*e64xm2\_t* merge, *uint64xm2\_t* *a*, unsigned long *b*, *e64xm2\_t* mask, unsigned int *gvl*)
- *e64xm4\_t vmsgtuvx\_mask\_e64xm4\_uint64xm4* (*e64xm4\_t* merge, *uint64xm4\_t* *a*, unsigned long *b*, *e64xm4\_t* mask, unsigned int *gvl*)
- *e64xm8\_t vmsgtuvx\_mask\_e64xm8\_uint64xm8* (*e64xm8\_t* merge, *uint64xm8\_t* *a*, unsigned long *b*, *e64xm8\_t* mask, unsigned int *gvl*)
- *e8xm1\_t vmsgtuvx\_mask\_e8xm1\_uint8xm1* (*e8xm1\_t* merge, *uint8xm1\_t* *a*, unsigned char *b*, *e8xm1\_t* mask, unsigned int *gvl*)
- *e8xm2\_t vmsgtuvx\_mask\_e8xm2\_uint8xm2* (*e8xm2\_t* merge, *uint8xm2\_t* *a*, unsigned char *b*, *e8xm2\_t* mask, unsigned int *gvl*)
- *e8xm4\_t vmsgtuvx\_mask\_e8xm4\_uint8xm4* (*e8xm4\_t* merge, *uint8xm4\_t* *a*, unsigned char *b*, *e8xm4\_t* mask, unsigned int *gvl*)
- *e8xm8\_t vmsgtuvx\_mask\_e8xm8\_uint8xm8* (*e8xm8\_t* merge, *uint8xm8\_t* *a*, unsigned char *b*, *e8xm8\_t* mask, unsigned int *gvl*)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = a[element] > b
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

### 2.8.10 Compare elementwise signed integer one vector and one scalar immediate for lower-than-or-equal

**Instruction:** [*'vmslevi.vi'*]

#### Prototypes:

- *e16xm1\_t vmslevi\_e16xm1\_int16xm1* (*int16xm1\_t* *a*, const short *b*, unsigned int *gvl*)
- *e16xm2\_t vmslevi\_e16xm2\_int16xm2* (*int16xm2\_t* *a*, const short *b*, unsigned int *gvl*)
- *e16xm4\_t vmslevi\_e16xm4\_int16xm4* (*int16xm4\_t* *a*, const short *b*, unsigned int *gvl*)
- *e16xm8\_t vmslevi\_e16xm8\_int16xm8* (*int16xm8\_t* *a*, const short *b*, unsigned int *gvl*)
- *e32xm1\_t vmslevi\_e32xm1\_int32xm1* (*int32xm1\_t* *a*, const int *b*, unsigned int *gvl*)
- *e32xm2\_t vmslevi\_e32xm2\_int32xm2* (*int32xm2\_t* *a*, const int *b*, unsigned int *gvl*)
- *e32xm4\_t vmslevi\_e32xm4\_int32xm4* (*int32xm4\_t* *a*, const int *b*, unsigned int *gvl*)
- *e32xm8\_t vmslevi\_e32xm8\_int32xm8* (*int32xm8\_t* *a*, const int *b*, unsigned int *gvl*)
- *e64xm1\_t vmslevi\_e64xm1\_int64xm1* (*int64xm1\_t* *a*, const long *b*, unsigned int *gvl*)
- *e64xm2\_t vmslevi\_e64xm2\_int64xm2* (*int64xm2\_t* *a*, const long *b*, unsigned int *gvl*)
- *e64xm4\_t vmslevi\_e64xm4\_int64xm4* (*int64xm4\_t* *a*, const long *b*, unsigned int *gvl*)
- *e64xm8\_t vmslevi\_e64xm8\_int64xm8* (*int64xm8\_t* *a*, const long *b*, unsigned int *gvl*)



- `e8xm1_t vmslevi_e8xm1_int8xm1 (int8xm1_t a, const signed char b, unsigned int gvl)`
- `e8xm2_t vmslevi_e8xm2_int8xm2 (int8xm2_t a, const signed char b, unsigned int gvl)`
- `e8xm4_t vmslevi_e8xm4_int8xm4 (int8xm4_t a, const signed char b, unsigned int gvl)`
- `e8xm8_t vmslevi_e8xm8_int8xm8 (int8xm8_t a, const signed char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] <= b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `e16xm1_t vmslevi_mask_e16xm1_int16xm1 (e16xm1_t merge, int16xm1_t a, const short b, e16xm1_t mask, unsigned int gvl)`
- `e16xm2_t vmslevi_mask_e16xm2_int16xm2 (e16xm2_t merge, int16xm2_t a, const short b, e16xm2_t mask, unsigned int gvl)`
- `e16xm4_t vmslevi_mask_e16xm4_int16xm4 (e16xm4_t merge, int16xm4_t a, const short b, e16xm4_t mask, unsigned int gvl)`
- `e16xm8_t vmslevi_mask_e16xm8_int16xm8 (e16xm8_t merge, int16xm8_t a, const short b, e16xm8_t mask, unsigned int gvl)`
- `e32xm1_t vmslevi_mask_e32xm1_int32xm1 (e32xm1_t merge, int32xm1_t a, const int b, e32xm1_t mask, unsigned int gvl)`
- `e32xm2_t vmslevi_mask_e32xm2_int32xm2 (e32xm2_t merge, int32xm2_t a, const int b, e32xm2_t mask, unsigned int gvl)`
- `e32xm4_t vmslevi_mask_e32xm4_int32xm4 (e32xm4_t merge, int32xm4_t a, const int b, e32xm4_t mask, unsigned int gvl)`
- `e32xm8_t vmslevi_mask_e32xm8_int32xm8 (e32xm8_t merge, int32xm8_t a, const int b, e32xm8_t mask, unsigned int gvl)`
- `e64xm1_t vmslevi_mask_e64xm1_int64xm1 (e64xm1_t merge, int64xm1_t a, const long b, e64xm1_t mask, unsigned int gvl)`
- `e64xm2_t vmslevi_mask_e64xm2_int64xm2 (e64xm2_t merge, int64xm2_t a, const long b, e64xm2_t mask, unsigned int gvl)`
- `e64xm4_t vmslevi_mask_e64xm4_int64xm4 (e64xm4_t merge, int64xm4_t a, const long b, e64xm4_t mask, unsigned int gvl)`
- `e64xm8_t vmslevi_mask_e64xm8_int64xm8 (e64xm8_t merge, int64xm8_t a, const long b, e64xm8_t mask, unsigned int gvl)`
- `e8xm1_t vmslevi_mask_e8xm1_int8xm1 (e8xm1_t merge, int8xm1_t a, const signed char b, e8xm1_t mask, unsigned int gvl)`
- `e8xm2_t vmslevi_mask_e8xm2_int8xm2 (e8xm2_t merge, int8xm2_t a, const signed char b, e8xm2_t mask, unsigned int gvl)`
- `e8xm4_t vmslevi_mask_e8xm4_int8xm4 (e8xm4_t merge, int8xm4_t a, const signed char b, e8xm4_t mask, unsigned int gvl)`
- `e8xm8_t vmslevi_mask_e8xm8_int8xm8 (e8xm8_t merge, int8xm8_t a, const signed char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] <= b
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.8.11 Compare elementwise signed vector-vector for lower-than-or-equal

**Instruction:** ['vmsle.vv']

**Prototypes:**

- *e16xm1\_t vmslevv\_e16xm1\_int16xm1* (*int16xm1\_t a*, *int16xm1\_t b*, unsigned int *gvl*)
- *e16xm2\_t vmslevv\_e16xm2\_int16xm2* (*int16xm2\_t a*, *int16xm2\_t b*, unsigned int *gvl*)
- *e16xm4\_t vmslevv\_e16xm4\_int16xm4* (*int16xm4\_t a*, *int16xm4\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmslevv\_e16xm8\_int16xm8* (*int16xm8\_t a*, *int16xm8\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmslevv\_e32xm1\_int32xm1* (*int32xm1\_t a*, *int32xm1\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmslevv\_e32xm2\_int32xm2* (*int32xm2\_t a*, *int32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmslevv\_e32xm4\_int32xm4* (*int32xm4\_t a*, *int32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmslevv\_e32xm8\_int32xm8* (*int32xm8\_t a*, *int32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmslevv\_e64xm1\_int64xm1* (*int64xm1\_t a*, *int64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmslevv\_e64xm2\_int64xm2* (*int64xm2\_t a*, *int64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmslevv\_e64xm4\_int64xm4* (*int64xm4\_t a*, *int64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmslevv\_e64xm8\_int64xm8* (*int64xm8\_t a*, *int64xm8\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmslevv\_e8xm1\_int8xm1* (*int8xm1\_t a*, *int8xm1\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmslevv\_e8xm2\_int8xm2* (*int8xm2\_t a*, *int8xm2\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmslevv\_e8xm4\_int8xm4* (*int8xm4\_t a*, *int8xm4\_t b*, unsigned int *gvl*)
- *e8xm8\_t vmslevv\_e8xm8\_int8xm8* (*int8xm8\_t a*, *int8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] <= b[element]
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmslevv\_mask\_e16xm1\_int16xm1* (*e16xm1\_t merge*, *int16xm1\_t a*, *int16xm1\_t b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmslevv\_mask\_e16xm2\_int16xm2* (*e16xm2\_t merge*, *int16xm2\_t a*, *int16xm2\_t b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmslevv\_mask\_e16xm4\_int16xm4* (*e16xm4\_t merge*, *int16xm4\_t a*, *int16xm4\_t b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmslevv\_mask\_e16xm8\_int16xm8* (*e16xm8\_t merge*, *int16xm8\_t a*, *int16xm8\_t b*, *e16xm8\_t mask*, unsigned int *gvl*)

- `e32xm1_t vmslevv_mask_e32xm1_int32xm1` (`e32xm1_t merge`, `int32xm1_t a`, `int32xm1_t b`, `e32xm1_t mask`, unsigned int gvl)
- `e32xm2_t vmslevv_mask_e32xm2_int32xm2` (`e32xm2_t merge`, `int32xm2_t a`, `int32xm2_t b`, `e32xm2_t mask`, unsigned int gvl)
- `e32xm4_t vmslevv_mask_e32xm4_int32xm4` (`e32xm4_t merge`, `int32xm4_t a`, `int32xm4_t b`, `e32xm4_t mask`, unsigned int gvl)
- `e32xm8_t vmslevv_mask_e32xm8_int32xm8` (`e32xm8_t merge`, `int32xm8_t a`, `int32xm8_t b`, `e32xm8_t mask`, unsigned int gvl)
- `e64xm1_t vmslevv_mask_e64xm1_int64xm1` (`e64xm1_t merge`, `int64xm1_t a`, `int64xm1_t b`, `e64xm1_t mask`, unsigned int gvl)
- `e64xm2_t vmslevv_mask_e64xm2_int64xm2` (`e64xm2_t merge`, `int64xm2_t a`, `int64xm2_t b`, `e64xm2_t mask`, unsigned int gvl)
- `e64xm4_t vmslevv_mask_e64xm4_int64xm4` (`e64xm4_t merge`, `int64xm4_t a`, `int64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)
- `e64xm8_t vmslevv_mask_e64xm8_int64xm8` (`e64xm8_t merge`, `int64xm8_t a`, `int64xm8_t b`, `e64xm8_t mask`, unsigned int gvl)
- `e8xm1_t vmslevv_mask_e8xm1_int8xm1` (`e8xm1_t merge`, `int8xm1_t a`, `int8xm1_t b`, `e8xm1_t mask`, unsigned int gvl)
- `e8xm2_t vmslevv_mask_e8xm2_int8xm2` (`e8xm2_t merge`, `int8xm2_t a`, `int8xm2_t b`, `e8xm2_t mask`, unsigned int gvl)
- `e8xm4_t vmslevv_mask_e8xm4_int8xm4` (`e8xm4_t merge`, `int8xm4_t a`, `int8xm4_t b`, `e8xm4_t mask`, unsigned int gvl)
- `e8xm8_t vmslevv_mask_e8xm8_int8xm8` (`e8xm8_t merge`, `int8xm8_t a`, `int8xm8_t b`, `e8xm8_t mask`, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] <= b[element]
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

**2.8.12 Compare elementwise signed vector-scalar for lower-than-or-equal****Instruction:** [`'vmsle.vx'`]**Prototypes:**

- `e16xm1_t vmslevx_e16xm1_int16xm1` (`int16xm1_t a`, short `b`, unsigned int gvl)
- `e16xm2_t vmslevx_e16xm2_int16xm2` (`int16xm2_t a`, short `b`, unsigned int gvl)
- `e16xm4_t vmslevx_e16xm4_int16xm4` (`int16xm4_t a`, short `b`, unsigned int gvl)
- `e16xm8_t vmslevx_e16xm8_int16xm8` (`int16xm8_t a`, short `b`, unsigned int gvl)
- `e32xm1_t vmslevx_e32xm1_int32xm1` (`int32xm1_t a`, int `b`, unsigned int gvl)
- `e32xm2_t vmslevx_e32xm2_int32xm2` (`int32xm2_t a`, int `b`, unsigned int gvl)
- `e32xm4_t vmslevx_e32xm4_int32xm4` (`int32xm4_t a`, int `b`, unsigned int gvl)
- `e32xm8_t vmslevx_e32xm8_int32xm8` (`int32xm8_t a`, int `b`, unsigned int gvl)

- *e64xm1\_t* vmslevx\_e64xm1\_int64xm1 (*int64xm1\_t* *a*, long *b*, unsigned int *gvl*)
- *e64xm2\_t* vmslevx\_e64xm2\_int64xm2 (*int64xm2\_t* *a*, long *b*, unsigned int *gvl*)
- *e64xm4\_t* vmslevx\_e64xm4\_int64xm4 (*int64xm4\_t* *a*, long *b*, unsigned int *gvl*)
- *e64xm8\_t* vmslevx\_e64xm8\_int64xm8 (*int64xm8\_t* *a*, long *b*, unsigned int *gvl*)
- *e8xm1\_t* vmslevx\_e8xm1\_int8xm1 (*int8xm1\_t* *a*, signed char *b*, unsigned int *gvl*)
- *e8xm2\_t* vmslevx\_e8xm2\_int8xm2 (*int8xm2\_t* *a*, signed char *b*, unsigned int *gvl*)
- *e8xm4\_t* vmslevx\_e8xm4\_int8xm4 (*int8xm4\_t* *a*, signed char *b*, unsigned int *gvl*)
- *e8xm8\_t* vmslevx\_e8xm8\_int8xm8 (*int8xm8\_t* *a*, signed char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] <= b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t* vmslevx\_mask\_e16xm1\_int16xm1 (*e16xm1\_t* *merge*, *int16xm1\_t* *a*, short *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *e16xm2\_t* vmslevx\_mask\_e16xm2\_int16xm2 (*e16xm2\_t* *merge*, *int16xm2\_t* *a*, short *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *e16xm4\_t* vmslevx\_mask\_e16xm4\_int16xm4 (*e16xm4\_t* *merge*, *int16xm4\_t* *a*, short *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *e16xm8\_t* vmslevx\_mask\_e16xm8\_int16xm8 (*e16xm8\_t* *merge*, *int16xm8\_t* *a*, short *b*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- *e32xm1\_t* vmslevx\_mask\_e32xm1\_int32xm1 (*e32xm1\_t* *merge*, *int32xm1\_t* *a*, int *b*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *e32xm2\_t* vmslevx\_mask\_e32xm2\_int32xm2 (*e32xm2\_t* *merge*, *int32xm2\_t* *a*, int *b*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *e32xm4\_t* vmslevx\_mask\_e32xm4\_int32xm4 (*e32xm4\_t* *merge*, *int32xm4\_t* *a*, int *b*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *e32xm8\_t* vmslevx\_mask\_e32xm8\_int32xm8 (*e32xm8\_t* *merge*, *int32xm8\_t* *a*, int *b*, *e32xm8\_t* *mask*, unsigned int *gvl*)
- *e64xm1\_t* vmslevx\_mask\_e64xm1\_int64xm1 (*e64xm1\_t* *merge*, *int64xm1\_t* *a*, long *b*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *e64xm2\_t* vmslevx\_mask\_e64xm2\_int64xm2 (*e64xm2\_t* *merge*, *int64xm2\_t* *a*, long *b*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *e64xm4\_t* vmslevx\_mask\_e64xm4\_int64xm4 (*e64xm4\_t* *merge*, *int64xm4\_t* *a*, long *b*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *e64xm8\_t* vmslevx\_mask\_e64xm8\_int64xm8 (*e64xm8\_t* *merge*, *int64xm8\_t* *a*, long *b*, *e64xm8\_t* *mask*, unsigned int *gvl*)
- *e8xm1\_t* vmslevx\_mask\_e8xm1\_int8xm1 (*e8xm1\_t* *merge*, *int8xm1\_t* *a*, signed char *b*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *e8xm2\_t* vmslevx\_mask\_e8xm2\_int8xm2 (*e8xm2\_t* *merge*, *int8xm2\_t* *a*, signed char *b*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- *e8xm4\_t* vmslevx\_mask\_e8xm4\_int8xm4 (*e8xm4\_t* *merge*, *int8xm4\_t* *a*, signed char *b*, *e8xm4\_t* *mask*, unsigned int *gvl*)

- *e8xm8\_t vmslevx\_mask\_e8xm8\_int8xm8* (*e8xm8\_t merge*, *int8xm8\_t a*, signed char *b*, *e8xm8\_t mask*, unsigned int *gvl*)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] <= b
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.8.13 Compare elementwise unsigned integer one vector and one scalar immediate for lower-than-or-equal

Instruction: ['vmsleu.vi']

Prototypes:

- *e16xm1\_t vmsleuvi\_e16xm1\_uint16xm1* (*uint16xm1\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *e16xm2\_t vmsleuvi\_e16xm2\_uint16xm2* (*uint16xm2\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *e16xm4\_t vmsleuvi\_e16xm4\_uint16xm4* (*uint16xm4\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *e16xm8\_t vmsleuvi\_e16xm8\_uint16xm8* (*uint16xm8\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *e32xm1\_t vmsleuvi\_e32xm1\_uint32xm1* (*uint32xm1\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *e32xm2\_t vmsleuvi\_e32xm2\_uint32xm2* (*uint32xm2\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *e32xm4\_t vmsleuvi\_e32xm4\_uint32xm4* (*uint32xm4\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *e32xm8\_t vmsleuvi\_e32xm8\_uint32xm8* (*uint32xm8\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *e64xm1\_t vmsleuvi\_e64xm1\_uint64xm1* (*uint64xm1\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *e64xm2\_t vmsleuvi\_e64xm2\_uint64xm2* (*uint64xm2\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *e64xm4\_t vmsleuvi\_e64xm4\_uint64xm4* (*uint64xm4\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *e64xm8\_t vmsleuvi\_e64xm8\_uint64xm8* (*uint64xm8\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *e8xm1\_t vmsleuvi\_e8xm1\_uint8xm1* (*uint8xm1\_t a*, const unsigned char *b*, unsigned int *gvl*)
- *e8xm2\_t vmsleuvi\_e8xm2\_uint8xm2* (*uint8xm2\_t a*, const unsigned char *b*, unsigned int *gvl*)
- *e8xm4\_t vmsleuvi\_e8xm4\_uint8xm4* (*uint8xm4\_t a*, const unsigned char *b*, unsigned int *gvl*)
- *e8xm8\_t vmsleuvi\_e8xm8\_uint8xm8* (*uint8xm8\_t a*, const unsigned char *b*, unsigned int *gvl*)

Operation:

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] <= b
result[gvl : VLMAX] = 0
```

Masked prototypes:

- *e16xm1\_t vmsleuvi\_mask\_e16xm1\_uint16xm1* (*e16xm1\_t merge*, *uint16xm1\_t a*, const unsigned short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmsleuvi\_mask\_e16xm2\_uint16xm2* (*e16xm2\_t merge*, *uint16xm2\_t a*, const unsigned short *b*, *e16xm2\_t mask*, unsigned int *gvl*)

- `e16xm4_t vmsleuvi_mask_e16xm4_uint16xm4` (`e16xm4_t` merge, `uint16xm4_t a`, const unsigned short `b`, `e16xm4_t` mask, unsigned int `gvl`)
- `e16xm8_t vmsleuvi_mask_e16xm8_uint16xm8` (`e16xm8_t` merge, `uint16xm8_t a`, const unsigned short `b`, `e16xm8_t` mask, unsigned int `gvl`)
- `e32xm1_t vmsleuvi_mask_e32xm1_uint32xm1` (`e32xm1_t` merge, `uint32xm1_t a`, const unsigned int `b`, `e32xm1_t` mask, unsigned int `gvl`)
- `e32xm2_t vmsleuvi_mask_e32xm2_uint32xm2` (`e32xm2_t` merge, `uint32xm2_t a`, const unsigned int `b`, `e32xm2_t` mask, unsigned int `gvl`)
- `e32xm4_t vmsleuvi_mask_e32xm4_uint32xm4` (`e32xm4_t` merge, `uint32xm4_t a`, const unsigned int `b`, `e32xm4_t` mask, unsigned int `gvl`)
- `e32xm8_t vmsleuvi_mask_e32xm8_uint32xm8` (`e32xm8_t` merge, `uint32xm8_t a`, const unsigned int `b`, `e32xm8_t` mask, unsigned int `gvl`)
- `e64xm1_t vmsleuvi_mask_e64xm1_uint64xm1` (`e64xm1_t` merge, `uint64xm1_t a`, const unsigned long `b`, `e64xm1_t` mask, unsigned int `gvl`)
- `e64xm2_t vmsleuvi_mask_e64xm2_uint64xm2` (`e64xm2_t` merge, `uint64xm2_t a`, const unsigned long `b`, `e64xm2_t` mask, unsigned int `gvl`)
- `e64xm4_t vmsleuvi_mask_e64xm4_uint64xm4` (`e64xm4_t` merge, `uint64xm4_t a`, const unsigned long `b`, `e64xm4_t` mask, unsigned int `gvl`)
- `e64xm8_t vmsleuvi_mask_e64xm8_uint64xm8` (`e64xm8_t` merge, `uint64xm8_t a`, const unsigned long `b`, `e64xm8_t` mask, unsigned int `gvl`)
- `e8xm1_t vmsleuvi_mask_e8xm1_uint8xm1` (`e8xm1_t` merge, `uint8xm1_t a`, const unsigned char `b`, `e8xm1_t` mask, unsigned int `gvl`)
- `e8xm2_t vmsleuvi_mask_e8xm2_uint8xm2` (`e8xm2_t` merge, `uint8xm2_t a`, const unsigned char `b`, `e8xm2_t` mask, unsigned int `gvl`)
- `e8xm4_t vmsleuvi_mask_e8xm4_uint8xm4` (`e8xm4_t` merge, `uint8xm4_t a`, const unsigned char `b`, `e8xm4_t` mask, unsigned int `gvl`)
- `e8xm8_t vmsleuvi_mask_e8xm8_uint8xm8` (`e8xm8_t` merge, `uint8xm8_t a`, const unsigned char `b`, `e8xm8_t` mask, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] <= b
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.8.14 Compare elementwise unsigned vector-vector for lower-than-or-equal

Instruction: ['vmsleu.vv']

Prototypes:

- `e16xm1_t vmsleuvv_e16xm1_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, unsigned int `gvl`)
- `e16xm2_t vmsleuvv_e16xm2_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, unsigned int `gvl`)
- `e16xm4_t vmsleuvv_e16xm4_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, unsigned int `gvl`)
- `e16xm8_t vmsleuvv_e16xm8_uint16xm8` (`uint16xm8_t a`, `uint16xm8_t b`, unsigned int `gvl`)
- `e32xm1_t vmsleuvv_e32xm1_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, unsigned int `gvl`)



- *e32xm2\_t vmsleuvv\_e32xm2\_uint32xm2* (*uint32xm2\_t a, uint32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmsleuvv\_e32xm4\_uint32xm4* (*uint32xm4\_t a, uint32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmsleuvv\_e32xm8\_uint32xm8* (*uint32xm8\_t a, uint32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmsleuvv\_e64xm1\_uint64xm1* (*uint64xm1\_t a, uint64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmsleuvv\_e64xm2\_uint64xm2* (*uint64xm2\_t a, uint64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmsleuvv\_e64xm4\_uint64xm4* (*uint64xm4\_t a, uint64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmsleuvv\_e64xm8\_uint64xm8* (*uint64xm8\_t a, uint64xm8\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmsleuvv\_e8xm1\_uint8xm1* (*uint8xm1\_t a, uint8xm1\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmsleuvv\_e8xm2\_uint8xm2* (*uint8xm2\_t a, uint8xm2\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmsleuvv\_e8xm4\_uint8xm4* (*uint8xm4\_t a, uint8xm4\_t b*, unsigned int *gvl*)
- *e8xm8\_t vmsleuvv\_e8xm8\_uint8xm8* (*uint8xm8\_t a, uint8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] <= b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmsleuvv\_mask\_e16xm1\_uint16xm1* (*e16xm1\_t merge, uint16xm1\_t a, uint16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmsleuvv\_mask\_e16xm2\_uint16xm2* (*e16xm2\_t merge, uint16xm2\_t a, uint16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmsleuvv\_mask\_e16xm4\_uint16xm4* (*e16xm4\_t merge, uint16xm4\_t a, uint16xm4\_t b, e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmsleuvv\_mask\_e16xm8\_uint16xm8* (*e16xm8\_t merge, uint16xm8\_t a, uint16xm8\_t b, e16xm8\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmsleuvv\_mask\_e32xm1\_uint32xm1* (*e32xm1\_t merge, uint32xm1\_t a, uint32xm1\_t b, e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm2\_t vmsleuvv\_mask\_e32xm2\_uint32xm2* (*e32xm2\_t merge, uint32xm2\_t a, uint32xm2\_t b, e32xm2\_t mask*, unsigned int *gvl*)
- *e32xm4\_t vmsleuvv\_mask\_e32xm4\_uint32xm4* (*e32xm4\_t merge, uint32xm4\_t a, uint32xm4\_t b, e32xm4\_t mask*, unsigned int *gvl*)
- *e32xm8\_t vmsleuvv\_mask\_e32xm8\_uint32xm8* (*e32xm8\_t merge, uint32xm8\_t a, uint32xm8\_t b, e32xm8\_t mask*, unsigned int *gvl*)
- *e64xm1\_t vmsleuvv\_mask\_e64xm1\_uint64xm1* (*e64xm1\_t merge, uint64xm1\_t a, uint64xm1\_t b, e64xm1\_t mask*, unsigned int *gvl*)
- *e64xm2\_t vmsleuvv\_mask\_e64xm2\_uint64xm2* (*e64xm2\_t merge, uint64xm2\_t a, uint64xm2\_t b, e64xm2\_t mask*, unsigned int *gvl*)
- *e64xm4\_t vmsleuvv\_mask\_e64xm4\_uint64xm4* (*e64xm4\_t merge, uint64xm4\_t a, uint64xm4\_t b, e64xm4\_t mask*, unsigned int *gvl*)
- *e64xm8\_t vmsleuvv\_mask\_e64xm8\_uint64xm8* (*e64xm8\_t merge, uint64xm8\_t a, uint64xm8\_t b, e64xm8\_t mask*, unsigned int *gvl*)
- *e8xm1\_t vmsleuvv\_mask\_e8xm1\_uint8xm1* (*e8xm1\_t merge, uint8xm1\_t a, uint8xm1\_t b, e8xm1\_t mask*, unsigned int *gvl*)

- `e8xm2_t vmsleuvv_mask_e8xm2_uint8xm2` (`e8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `e8xm4_t vmsleuvv_mask_e8xm4_uint8xm4` (`e8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `e8xm8_t vmsleuvv_mask_e8xm8_uint8xm8` (`e8xm8_t merge`, `uint8xm8_t a`, `uint8xm8_t b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] <= b[element]
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.8.15 Compare elementwise unsigned vector-scalar for lower-than-or-equal

**Instruction:** ['vmsleu.vx']

**Prototypes:**

- `e16xm1_t vmsleuvx_e16xm1_uint16xm1` (`uint16xm1_t a`, unsigned short `b`, unsigned int `gvl`)
- `e16xm2_t vmsleuvx_e16xm2_uint16xm2` (`uint16xm2_t a`, unsigned short `b`, unsigned int `gvl`)
- `e16xm4_t vmsleuvx_e16xm4_uint16xm4` (`uint16xm4_t a`, unsigned short `b`, unsigned int `gvl`)
- `e16xm8_t vmsleuvx_e16xm8_uint16xm8` (`uint16xm8_t a`, unsigned short `b`, unsigned int `gvl`)
- `e32xm1_t vmsleuvx_e32xm1_uint32xm1` (`uint32xm1_t a`, unsigned int `b`, unsigned int `gvl`)
- `e32xm2_t vmsleuvx_e32xm2_uint32xm2` (`uint32xm2_t a`, unsigned int `b`, unsigned int `gvl`)
- `e32xm4_t vmsleuvx_e32xm4_uint32xm4` (`uint32xm4_t a`, unsigned int `b`, unsigned int `gvl`)
- `e32xm8_t vmsleuvx_e32xm8_uint32xm8` (`uint32xm8_t a`, unsigned int `b`, unsigned int `gvl`)
- `e64xm1_t vmsleuvx_e64xm1_uint64xm1` (`uint64xm1_t a`, unsigned long `b`, unsigned int `gvl`)
- `e64xm2_t vmsleuvx_e64xm2_uint64xm2` (`uint64xm2_t a`, unsigned long `b`, unsigned int `gvl`)
- `e64xm4_t vmsleuvx_e64xm4_uint64xm4` (`uint64xm4_t a`, unsigned long `b`, unsigned int `gvl`)
- `e64xm8_t vmsleuvx_e64xm8_uint64xm8` (`uint64xm8_t a`, unsigned long `b`, unsigned int `gvl`)
- `e8xm1_t vmsleuvx_e8xm1_uint8xm1` (`uint8xm1_t a`, unsigned char `b`, unsigned int `gvl`)
- `e8xm2_t vmsleuvx_e8xm2_uint8xm2` (`uint8xm2_t a`, unsigned char `b`, unsigned int `gvl`)
- `e8xm4_t vmsleuvx_e8xm4_uint8xm4` (`uint8xm4_t a`, unsigned char `b`, unsigned int `gvl`)
- `e8xm8_t vmsleuvx_e8xm8_uint8xm8` (`uint8xm8_t a`, unsigned char `b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] <= b
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `e16xm1_t vmsleuvx_mask_e16xm1_uint16xm1` (`e16xm1_t` merge, `uint16xm1_t` *a*, unsigned short *b*, `e16xm1_t` mask, unsigned int *gvl*)
- `e16xm2_t vmsleuvx_mask_e16xm2_uint16xm2` (`e16xm2_t` merge, `uint16xm2_t` *a*, unsigned short *b*, `e16xm2_t` mask, unsigned int *gvl*)
- `e16xm4_t vmsleuvx_mask_e16xm4_uint16xm4` (`e16xm4_t` merge, `uint16xm4_t` *a*, unsigned short *b*, `e16xm4_t` mask, unsigned int *gvl*)
- `e16xm8_t vmsleuvx_mask_e16xm8_uint16xm8` (`e16xm8_t` merge, `uint16xm8_t` *a*, unsigned short *b*, `e16xm8_t` mask, unsigned int *gvl*)
- `e32xm1_t vmsleuvx_mask_e32xm1_uint32xm1` (`e32xm1_t` merge, `uint32xm1_t` *a*, unsigned int *b*, `e32xm1_t` mask, unsigned int *gvl*)
- `e32xm2_t vmsleuvx_mask_e32xm2_uint32xm2` (`e32xm2_t` merge, `uint32xm2_t` *a*, unsigned int *b*, `e32xm2_t` mask, unsigned int *gvl*)
- `e32xm4_t vmsleuvx_mask_e32xm4_uint32xm4` (`e32xm4_t` merge, `uint32xm4_t` *a*, unsigned int *b*, `e32xm4_t` mask, unsigned int *gvl*)
- `e32xm8_t vmsleuvx_mask_e32xm8_uint32xm8` (`e32xm8_t` merge, `uint32xm8_t` *a*, unsigned int *b*, `e32xm8_t` mask, unsigned int *gvl*)
- `e64xm1_t vmsleuvx_mask_e64xm1_uint64xm1` (`e64xm1_t` merge, `uint64xm1_t` *a*, unsigned long *b*, `e64xm1_t` mask, unsigned int *gvl*)
- `e64xm2_t vmsleuvx_mask_e64xm2_uint64xm2` (`e64xm2_t` merge, `uint64xm2_t` *a*, unsigned long *b*, `e64xm2_t` mask, unsigned int *gvl*)
- `e64xm4_t vmsleuvx_mask_e64xm4_uint64xm4` (`e64xm4_t` merge, `uint64xm4_t` *a*, unsigned long *b*, `e64xm4_t` mask, unsigned int *gvl*)
- `e64xm8_t vmsleuvx_mask_e64xm8_uint64xm8` (`e64xm8_t` merge, `uint64xm8_t` *a*, unsigned long *b*, `e64xm8_t` mask, unsigned int *gvl*)
- `e8xm1_t vmsleuvx_mask_e8xm1_uint8xm1` (`e8xm1_t` merge, `uint8xm1_t` *a*, unsigned char *b*, `e8xm1_t` mask, unsigned int *gvl*)
- `e8xm2_t vmsleuvx_mask_e8xm2_uint8xm2` (`e8xm2_t` merge, `uint8xm2_t` *a*, unsigned char *b*, `e8xm2_t` mask, unsigned int *gvl*)
- `e8xm4_t vmsleuvx_mask_e8xm4_uint8xm4` (`e8xm4_t` merge, `uint8xm4_t` *a*, unsigned char *b*, `e8xm4_t` mask, unsigned int *gvl*)
- `e8xm8_t vmsleuvx_mask_e8xm8_uint8xm8` (`e8xm8_t` merge, `uint8xm8_t` *a*, unsigned char *b*, `e8xm8_t` mask, unsigned int *gvl*)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] <= b
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.8.16 Compare elementwise signed vector-vector for lower-than

**Instruction:** [`'vmslt.vv'`]

**Prototypes:**

- `e16xm1_t vmsltvv_e16xm1_int16xm1` (`int16xm1_t` *a*, `int16xm1_t` *b*, unsigned int *gvl*)

- *e16xm2\_t vmsltvv\_e16xm2\_int16xm2* (*int16xm2\_t a, int16xm2\_t b*, unsigned int *gvl*)
- *e16xm4\_t vmsltvv\_e16xm4\_int16xm4* (*int16xm4\_t a, int16xm4\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmsltvv\_e16xm8\_int16xm8* (*int16xm8\_t a, int16xm8\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmsltvv\_e32xm1\_int32xm1* (*int32xm1\_t a, int32xm1\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmsltvv\_e32xm2\_int32xm2* (*int32xm2\_t a, int32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmsltvv\_e32xm4\_int32xm4* (*int32xm4\_t a, int32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmsltvv\_e32xm8\_int32xm8* (*int32xm8\_t a, int32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmsltvv\_e64xm1\_int64xm1* (*int64xm1\_t a, int64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmsltvv\_e64xm2\_int64xm2* (*int64xm2\_t a, int64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmsltvv\_e64xm4\_int64xm4* (*int64xm4\_t a, int64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmsltvv\_e64xm8\_int64xm8* (*int64xm8\_t a, int64xm8\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmsltvv\_e8xm1\_int8xm1* (*int8xm1\_t a, int8xm1\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmsltvv\_e8xm2\_int8xm2* (*int8xm2\_t a, int8xm2\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmsltvv\_e8xm4\_int8xm4* (*int8xm4\_t a, int8xm4\_t b*, unsigned int *gvl*)
- *e8xm8\_t vmsltvv\_e8xm8\_int8xm8* (*int8xm8\_t a, int8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] < b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmsltvv\_mask\_e16xm1\_int16xm1* (*e16xm1\_t merge, int16xm1\_t a, int16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmsltvv\_mask\_e16xm2\_int16xm2* (*e16xm2\_t merge, int16xm2\_t a, int16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmsltvv\_mask\_e16xm4\_int16xm4* (*e16xm4\_t merge, int16xm4\_t a, int16xm4\_t b, e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmsltvv\_mask\_e16xm8\_int16xm8* (*e16xm8\_t merge, int16xm8\_t a, int16xm8\_t b, e16xm8\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmsltvv\_mask\_e32xm1\_int32xm1* (*e32xm1\_t merge, int32xm1\_t a, int32xm1\_t b, e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm2\_t vmsltvv\_mask\_e32xm2\_int32xm2* (*e32xm2\_t merge, int32xm2\_t a, int32xm2\_t b, e32xm2\_t mask*, unsigned int *gvl*)
- *e32xm4\_t vmsltvv\_mask\_e32xm4\_int32xm4* (*e32xm4\_t merge, int32xm4\_t a, int32xm4\_t b, e32xm4\_t mask*, unsigned int *gvl*)
- *e32xm8\_t vmsltvv\_mask\_e32xm8\_int32xm8* (*e32xm8\_t merge, int32xm8\_t a, int32xm8\_t b, e32xm8\_t mask*, unsigned int *gvl*)
- *e64xm1\_t vmsltvv\_mask\_e64xm1\_int64xm1* (*e64xm1\_t merge, int64xm1\_t a, int64xm1\_t b, e64xm1\_t mask*, unsigned int *gvl*)
- *e64xm2\_t vmsltvv\_mask\_e64xm2\_int64xm2* (*e64xm2\_t merge, int64xm2\_t a, int64xm2\_t b, e64xm2\_t mask*, unsigned int *gvl*)

- `e64xm4_t vmsltvv_mask_e64xm4_int64xm4` (`e64xm4_t` merge, `int64xm4_t` *a*, `int64xm4_t` *b*, `e64xm4_t` mask, unsigned int *gvl*)
- `e64xm8_t vmsltvv_mask_e64xm8_int64xm8` (`e64xm8_t` merge, `int64xm8_t` *a*, `int64xm8_t` *b*, `e64xm8_t` mask, unsigned int *gvl*)
- `e8xm1_t vmsltvv_mask_e8xm1_int8xm1` (`e8xm1_t` merge, `int8xm1_t` *a*, `int8xm1_t` *b*, `e8xm1_t` mask, unsigned int *gvl*)
- `e8xm2_t vmsltvv_mask_e8xm2_int8xm2` (`e8xm2_t` merge, `int8xm2_t` *a*, `int8xm2_t` *b*, `e8xm2_t` mask, unsigned int *gvl*)
- `e8xm4_t vmsltvv_mask_e8xm4_int8xm4` (`e8xm4_t` merge, `int8xm4_t` *a*, `int8xm4_t` *b*, `e8xm4_t` mask, unsigned int *gvl*)
- `e8xm8_t vmsltvv_mask_e8xm8_int8xm8` (`e8xm8_t` merge, `int8xm8_t` *a*, `int8xm8_t` *b*, `e8xm8_t` mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] < b[element]
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.8.17 Compare elementwise signed vector-scalar for lower-than****Instruction:** [`'vmslt.vx'`]**Prototypes:**

- `e16xm1_t vmsltvx_e16xm1_int16xm1` (`int16xm1_t` *a*, short *b*, unsigned int *gvl*)
- `e16xm2_t vmsltvx_e16xm2_int16xm2` (`int16xm2_t` *a*, short *b*, unsigned int *gvl*)
- `e16xm4_t vmsltvx_e16xm4_int16xm4` (`int16xm4_t` *a*, short *b*, unsigned int *gvl*)
- `e16xm8_t vmsltvx_e16xm8_int16xm8` (`int16xm8_t` *a*, short *b*, unsigned int *gvl*)
- `e32xm1_t vmsltvx_e32xm1_int32xm1` (`int32xm1_t` *a*, int *b*, unsigned int *gvl*)
- `e32xm2_t vmsltvx_e32xm2_int32xm2` (`int32xm2_t` *a*, int *b*, unsigned int *gvl*)
- `e32xm4_t vmsltvx_e32xm4_int32xm4` (`int32xm4_t` *a*, int *b*, unsigned int *gvl*)
- `e32xm8_t vmsltvx_e32xm8_int32xm8` (`int32xm8_t` *a*, int *b*, unsigned int *gvl*)
- `e64xm1_t vmsltvx_e64xm1_int64xm1` (`int64xm1_t` *a*, long *b*, unsigned int *gvl*)
- `e64xm2_t vmsltvx_e64xm2_int64xm2` (`int64xm2_t` *a*, long *b*, unsigned int *gvl*)
- `e64xm4_t vmsltvx_e64xm4_int64xm4` (`int64xm4_t` *a*, long *b*, unsigned int *gvl*)
- `e64xm8_t vmsltvx_e64xm8_int64xm8` (`int64xm8_t` *a*, long *b*, unsigned int *gvl*)
- `e8xm1_t vmsltvx_e8xm1_int8xm1` (`int8xm1_t` *a*, signed char *b*, unsigned int *gvl*)
- `e8xm2_t vmsltvx_e8xm2_int8xm2` (`int8xm2_t` *a*, signed char *b*, unsigned int *gvl*)
- `e8xm4_t vmsltvx_e8xm4_int8xm4` (`int8xm4_t` *a*, signed char *b*, unsigned int *gvl*)
- `e8xm8_t vmsltvx_e8xm8_int8xm8` (`int8xm8_t` *a*, signed char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] < b
      result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- *e16xm1\_t vmsltvx\_mask\_e16xm1\_int16xm1* (*e16xm1\_t* merge, *int16xm1\_t* *a*, short *b*, *e16xm1\_t* mask, unsigned int *gvl*)
- *e16xm2\_t vmsltvx\_mask\_e16xm2\_int16xm2* (*e16xm2\_t* merge, *int16xm2\_t* *a*, short *b*, *e16xm2\_t* mask, unsigned int *gvl*)
- *e16xm4\_t vmsltvx\_mask\_e16xm4\_int16xm4* (*e16xm4\_t* merge, *int16xm4\_t* *a*, short *b*, *e16xm4\_t* mask, unsigned int *gvl*)
- *e16xm8\_t vmsltvx\_mask\_e16xm8\_int16xm8* (*e16xm8\_t* merge, *int16xm8\_t* *a*, short *b*, *e16xm8\_t* mask, unsigned int *gvl*)
- *e32xm1\_t vmsltvx\_mask\_e32xm1\_int32xm1* (*e32xm1\_t* merge, *int32xm1\_t* *a*, int *b*, *e32xm1\_t* mask, unsigned int *gvl*)
- *e32xm2\_t vmsltvx\_mask\_e32xm2\_int32xm2* (*e32xm2\_t* merge, *int32xm2\_t* *a*, int *b*, *e32xm2\_t* mask, unsigned int *gvl*)
- *e32xm4\_t vmsltvx\_mask\_e32xm4\_int32xm4* (*e32xm4\_t* merge, *int32xm4\_t* *a*, int *b*, *e32xm4\_t* mask, unsigned int *gvl*)
- *e32xm8\_t vmsltvx\_mask\_e32xm8\_int32xm8* (*e32xm8\_t* merge, *int32xm8\_t* *a*, int *b*, *e32xm8\_t* mask, unsigned int *gvl*)
- *e64xm1\_t vmsltvx\_mask\_e64xm1\_int64xm1* (*e64xm1\_t* merge, *int64xm1\_t* *a*, long *b*, *e64xm1\_t* mask, unsigned int *gvl*)
- *e64xm2\_t vmsltvx\_mask\_e64xm2\_int64xm2* (*e64xm2\_t* merge, *int64xm2\_t* *a*, long *b*, *e64xm2\_t* mask, unsigned int *gvl*)
- *e64xm4\_t vmsltvx\_mask\_e64xm4\_int64xm4* (*e64xm4\_t* merge, *int64xm4\_t* *a*, long *b*, *e64xm4\_t* mask, unsigned int *gvl*)
- *e64xm8\_t vmsltvx\_mask\_e64xm8\_int64xm8* (*e64xm8\_t* merge, *int64xm8\_t* *a*, long *b*, *e64xm8\_t* mask, unsigned int *gvl*)
- *e8xm1\_t vmsltvx\_mask\_e8xm1\_int8xm1* (*e8xm1\_t* merge, *int8xm1\_t* *a*, signed char *b*, *e8xm1\_t* mask, unsigned int *gvl*)
- *e8xm2\_t vmsltvx\_mask\_e8xm2\_int8xm2* (*e8xm2\_t* merge, *int8xm2\_t* *a*, signed char *b*, *e8xm2\_t* mask, unsigned int *gvl*)
- *e8xm4\_t vmsltvx\_mask\_e8xm4\_int8xm4* (*e8xm4\_t* merge, *int8xm4\_t* *a*, signed char *b*, *e8xm4\_t* mask, unsigned int *gvl*)
- *e8xm8\_t vmsltvx\_mask\_e8xm8\_int8xm8* (*e8xm8\_t* merge, *int8xm8\_t* *a*, signed char *b*, *e8xm8\_t* mask, unsigned int *gvl*)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] < b
      else
        result[element] = merge[element]
      result[gvl : VLMAX] = 0
```



## 2.8.18 Compare elementwise unsigned vector-vector for lower-than

**Instruction:** [`'vmsltu.vv'`]

**Prototypes:**

- `e16xm1_t vmsltuvv_e16xm1_uint16xm1 (uint16xm1_t a, uint16xm1_t b, unsigned int gvl)`
- `e16xm2_t vmsltuvv_e16xm2_uint16xm2 (uint16xm2_t a, uint16xm2_t b, unsigned int gvl)`
- `e16xm4_t vmsltuvv_e16xm4_uint16xm4 (uint16xm4_t a, uint16xm4_t b, unsigned int gvl)`
- `e16xm8_t vmsltuvv_e16xm8_uint16xm8 (uint16xm8_t a, uint16xm8_t b, unsigned int gvl)`
- `e32xm1_t vmsltuvv_e32xm1_uint32xm1 (uint32xm1_t a, uint32xm1_t b, unsigned int gvl)`
- `e32xm2_t vmsltuvv_e32xm2_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `e32xm4_t vmsltuvv_e32xm4_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `e32xm8_t vmsltuvv_e32xm8_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `e64xm1_t vmsltuvv_e64xm1_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `e64xm2_t vmsltuvv_e64xm2_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `e64xm4_t vmsltuvv_e64xm4_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `e64xm8_t vmsltuvv_e64xm8_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `e8xm1_t vmsltuvv_e8xm1_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `e8xm2_t vmsltuvv_e8xm2_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `e8xm4_t vmsltuvv_e8xm4_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `e8xm8_t vmsltuvv_e8xm8_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] < b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `e16xm1_t vmsltuvv_mask_e16xm1_uint16xm1 (e16xm1_t merge, uint16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `e16xm2_t vmsltuvv_mask_e16xm2_uint16xm2 (e16xm2_t merge, uint16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `e16xm4_t vmsltuvv_mask_e16xm4_uint16xm4 (e16xm4_t merge, uint16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `e16xm8_t vmsltuvv_mask_e16xm8_uint16xm8 (e16xm8_t merge, uint16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `e32xm1_t vmsltuvv_mask_e32xm1_uint32xm1 (e32xm1_t merge, uint32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `e32xm2_t vmsltuvv_mask_e32xm2_uint32xm2 (e32xm2_t merge, uint32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `e32xm4_t vmsltuvv_mask_e32xm4_uint32xm4 (e32xm4_t merge, uint32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `e32xm8_t vmsltuvv_mask_e32xm8_uint32xm8 (e32xm8_t merge, uint32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`

- `e64xm1_t vmsltuvv_mask_e64xm1_uint64xm1` (`e64xm1_t merge`, `uint64xm1_t a`, `uint64xm1_t b`, `e64xm1_t mask`, unsigned int `gvl`)
- `e64xm2_t vmsltuvv_mask_e64xm2_uint64xm2` (`e64xm2_t merge`, `uint64xm2_t a`, `uint64xm2_t b`, `e64xm2_t mask`, unsigned int `gvl`)
- `e64xm4_t vmsltuvv_mask_e64xm4_uint64xm4` (`e64xm4_t merge`, `uint64xm4_t a`, `uint64xm4_t b`, `e64xm4_t mask`, unsigned int `gvl`)
- `e64xm8_t vmsltuvv_mask_e64xm8_uint64xm8` (`e64xm8_t merge`, `uint64xm8_t a`, `uint64xm8_t b`, `e64xm8_t mask`, unsigned int `gvl`)
- `e8xm1_t vmsltuvv_mask_e8xm1_uint8xm1` (`e8xm1_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int `gvl`)
- `e8xm2_t vmsltuvv_mask_e8xm2_uint8xm2` (`e8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `e8xm4_t vmsltuvv_mask_e8xm4_uint8xm4` (`e8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `e8xm8_t vmsltuvv_mask_e8xm8_uint8xm8` (`e8xm8_t merge`, `uint8xm8_t a`, `uint8xm8_t b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] < b[element]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.8.19 Compare elementwise unsigned vector-scalar for lower-than

Instruction: ['vmsltu.vx']

Prototypes:

- `e16xm1_t vmsltuvx_e16xm1_uint16xm1` (`uint16xm1_t a`, unsigned short `b`, unsigned int `gvl`)
- `e16xm2_t vmsltuvx_e16xm2_uint16xm2` (`uint16xm2_t a`, unsigned short `b`, unsigned int `gvl`)
- `e16xm4_t vmsltuvx_e16xm4_uint16xm4` (`uint16xm4_t a`, unsigned short `b`, unsigned int `gvl`)
- `e16xm8_t vmsltuvx_e16xm8_uint16xm8` (`uint16xm8_t a`, unsigned short `b`, unsigned int `gvl`)
- `e32xm1_t vmsltuvx_e32xm1_uint32xm1` (`uint32xm1_t a`, unsigned int `b`, unsigned int `gvl`)
- `e32xm2_t vmsltuvx_e32xm2_uint32xm2` (`uint32xm2_t a`, unsigned int `b`, unsigned int `gvl`)
- `e32xm4_t vmsltuvx_e32xm4_uint32xm4` (`uint32xm4_t a`, unsigned int `b`, unsigned int `gvl`)
- `e32xm8_t vmsltuvx_e32xm8_uint32xm8` (`uint32xm8_t a`, unsigned int `b`, unsigned int `gvl`)
- `e64xm1_t vmsltuvx_e64xm1_uint64xm1` (`uint64xm1_t a`, unsigned long `b`, unsigned int `gvl`)
- `e64xm2_t vmsltuvx_e64xm2_uint64xm2` (`uint64xm2_t a`, unsigned long `b`, unsigned int `gvl`)
- `e64xm4_t vmsltuvx_e64xm4_uint64xm4` (`uint64xm4_t a`, unsigned long `b`, unsigned int `gvl`)
- `e64xm8_t vmsltuvx_e64xm8_uint64xm8` (`uint64xm8_t a`, unsigned long `b`, unsigned int `gvl`)
- `e8xm1_t vmsltuvx_e8xm1_uint8xm1` (`uint8xm1_t a`, unsigned char `b`, unsigned int `gvl`)
- `e8xm2_t vmsltuvx_e8xm2_uint8xm2` (`uint8xm2_t a`, unsigned char `b`, unsigned int `gvl`)

- `e8xm4_t vmsltuvx_e8xm4_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `e8xm8_t vmsltuvx_e8xm8_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] < b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `e16xm1_t vmsltuvx_mask_e16xm1_uint16xm1 (e16xm1_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `e16xm2_t vmsltuvx_mask_e16xm2_uint16xm2 (e16xm2_t merge, uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `e16xm4_t vmsltuvx_mask_e16xm4_uint16xm4 (e16xm4_t merge, uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `e16xm8_t vmsltuvx_mask_e16xm8_uint16xm8 (e16xm8_t merge, uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `e32xm1_t vmsltuvx_mask_e32xm1_uint32xm1 (e32xm1_t merge, uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `e32xm2_t vmsltuvx_mask_e32xm2_uint32xm2 (e32xm2_t merge, uint32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `e32xm4_t vmsltuvx_mask_e32xm4_uint32xm4 (e32xm4_t merge, uint32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `e32xm8_t vmsltuvx_mask_e32xm8_uint32xm8 (e32xm8_t merge, uint32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `e64xm1_t vmsltuvx_mask_e64xm1_uint64xm1 (e64xm1_t merge, uint64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`
- `e64xm2_t vmsltuvx_mask_e64xm2_uint64xm2 (e64xm2_t merge, uint64xm2_t a, unsigned long b, e64xm2_t mask, unsigned int gvl)`
- `e64xm4_t vmsltuvx_mask_e64xm4_uint64xm4 (e64xm4_t merge, uint64xm4_t a, unsigned long b, e64xm4_t mask, unsigned int gvl)`
- `e64xm8_t vmsltuvx_mask_e64xm8_uint64xm8 (e64xm8_t merge, uint64xm8_t a, unsigned long b, e64xm8_t mask, unsigned int gvl)`
- `e8xm1_t vmsltuvx_mask_e8xm1_uint8xm1 (e8xm1_t merge, uint8xm1_t a, unsigned char b, e8xm1_t mask, unsigned int gvl)`
- `e8xm2_t vmsltuvx_mask_e8xm2_uint8xm2 (e8xm2_t merge, uint8xm2_t a, unsigned char b, e8xm2_t mask, unsigned int gvl)`
- `e8xm4_t vmsltuvx_mask_e8xm4_uint8xm4 (e8xm4_t merge, uint8xm4_t a, unsigned char b, e8xm4_t mask, unsigned int gvl)`
- `e8xm8_t vmsltuvx_mask_e8xm8_uint8xm8 (e8xm8_t merge, uint8xm8_t a, unsigned char b, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] < b
      else
```

(continues on next page)

(continued from previous page)

```
result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.8.20 Compare elementwise vector-immediate for inequality

**Instruction:** [`vmsne.vi`']

**Prototypes:**

- *e16xm1\_t vmsnevi\_e16xm1\_int16xm1* (*int16xm1\_t a*, const short *b*, unsigned int *gvl*)
- *e16xm1\_t vmsnevi\_e16xm1\_uint16xm1* (*uint16xm1\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *e16xm2\_t vmsnevi\_e16xm2\_int16xm2* (*int16xm2\_t a*, const short *b*, unsigned int *gvl*)
- *e16xm2\_t vmsnevi\_e16xm2\_uint16xm2* (*uint16xm2\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *e16xm4\_t vmsnevi\_e16xm4\_int16xm4* (*int16xm4\_t a*, const short *b*, unsigned int *gvl*)
- *e16xm4\_t vmsnevi\_e16xm4\_uint16xm4* (*uint16xm4\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *e16xm8\_t vmsnevi\_e16xm8\_int16xm8* (*int16xm8\_t a*, const short *b*, unsigned int *gvl*)
- *e16xm8\_t vmsnevi\_e16xm8\_uint16xm8* (*uint16xm8\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *e32xm1\_t vmsnevi\_e32xm1\_int32xm1* (*int32xm1\_t a*, const int *b*, unsigned int *gvl*)
- *e32xm1\_t vmsnevi\_e32xm1\_uint32xm1* (*uint32xm1\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *e32xm2\_t vmsnevi\_e32xm2\_int32xm2* (*int32xm2\_t a*, const int *b*, unsigned int *gvl*)
- *e32xm2\_t vmsnevi\_e32xm2\_uint32xm2* (*uint32xm2\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *e32xm4\_t vmsnevi\_e32xm4\_int32xm4* (*int32xm4\_t a*, const int *b*, unsigned int *gvl*)
- *e32xm4\_t vmsnevi\_e32xm4\_uint32xm4* (*uint32xm4\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *e32xm8\_t vmsnevi\_e32xm8\_int32xm8* (*int32xm8\_t a*, const int *b*, unsigned int *gvl*)
- *e32xm8\_t vmsnevi\_e32xm8\_uint32xm8* (*uint32xm8\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *e64xm1\_t vmsnevi\_e64xm1\_int64xm1* (*int64xm1\_t a*, const long *b*, unsigned int *gvl*)
- *e64xm1\_t vmsnevi\_e64xm1\_uint64xm1* (*uint64xm1\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *e64xm2\_t vmsnevi\_e64xm2\_int64xm2* (*int64xm2\_t a*, const long *b*, unsigned int *gvl*)
- *e64xm2\_t vmsnevi\_e64xm2\_uint64xm2* (*uint64xm2\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *e64xm4\_t vmsnevi\_e64xm4\_int64xm4* (*int64xm4\_t a*, const long *b*, unsigned int *gvl*)
- *e64xm4\_t vmsnevi\_e64xm4\_uint64xm4* (*uint64xm4\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *e64xm8\_t vmsnevi\_e64xm8\_int64xm8* (*int64xm8\_t a*, const long *b*, unsigned int *gvl*)
- *e64xm8\_t vmsnevi\_e64xm8\_uint64xm8* (*uint64xm8\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *e8xm1\_t vmsnevi\_e8xm1\_int8xm1* (*int8xm1\_t a*, const signed char *b*, unsigned int *gvl*)
- *e8xm1\_t vmsnevi\_e8xm1\_uint8xm1* (*uint8xm1\_t a*, const unsigned char *b*, unsigned int *gvl*)
- *e8xm2\_t vmsnevi\_e8xm2\_int8xm2* (*int8xm2\_t a*, const signed char *b*, unsigned int *gvl*)
- *e8xm2\_t vmsnevi\_e8xm2\_uint8xm2* (*uint8xm2\_t a*, const unsigned char *b*, unsigned int *gvl*)
- *e8xm4\_t vmsnevi\_e8xm4\_int8xm4* (*int8xm4\_t a*, const signed char *b*, unsigned int *gvl*)

- *e8xm4\_t vmsnevi\_e8xm4\_uint8xm4* (*uint8xm4\_t a*, const unsigned char *b*, unsigned int *gvl*)
- *e8xm8\_t vmsnevi\_e8xm8\_int8xm8* (*int8xm8\_t a*, const signed char *b*, unsigned int *gvl*)
- *e8xm8\_t vmsnevi\_e8xm8\_uint8xm8* (*uint8xm8\_t a*, const unsigned char *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] != b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmsnevi\_mask\_e16xm1\_int16xm1* (*e16xm1\_t merge*, *int16xm1\_t a*, const short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm1\_t vmsnevi\_mask\_e16xm1\_uint16xm1* (*e16xm1\_t merge*, *uint16xm1\_t a*, const unsigned short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmsnevi\_mask\_e16xm2\_int16xm2* (*e16xm2\_t merge*, *int16xm2\_t a*, const short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmsnevi\_mask\_e16xm2\_uint16xm2* (*e16xm2\_t merge*, *uint16xm2\_t a*, const unsigned short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmsnevi\_mask\_e16xm4\_int16xm4* (*e16xm4\_t merge*, *int16xm4\_t a*, const short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmsnevi\_mask\_e16xm4\_uint16xm4* (*e16xm4\_t merge*, *uint16xm4\_t a*, const unsigned short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmsnevi\_mask\_e16xm8\_int16xm8* (*e16xm8\_t merge*, *int16xm8\_t a*, const short *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmsnevi\_mask\_e16xm8\_uint16xm8* (*e16xm8\_t merge*, *uint16xm8\_t a*, const unsigned short *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmsnevi\_mask\_e32xm1\_int32xm1* (*e32xm1\_t merge*, *int32xm1\_t a*, const int *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmsnevi\_mask\_e32xm1\_uint32xm1* (*e32xm1\_t merge*, *uint32xm1\_t a*, const unsigned int *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm2\_t vmsnevi\_mask\_e32xm2\_int32xm2* (*e32xm2\_t merge*, *int32xm2\_t a*, const int *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *e32xm2\_t vmsnevi\_mask\_e32xm2\_uint32xm2* (*e32xm2\_t merge*, *uint32xm2\_t a*, const unsigned int *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *e32xm4\_t vmsnevi\_mask\_e32xm4\_int32xm4* (*e32xm4\_t merge*, *int32xm4\_t a*, const int *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *e32xm4\_t vmsnevi\_mask\_e32xm4\_uint32xm4* (*e32xm4\_t merge*, *uint32xm4\_t a*, const unsigned int *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *e32xm8\_t vmsnevi\_mask\_e32xm8\_int32xm8* (*e32xm8\_t merge*, *int32xm8\_t a*, const int *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *e32xm8\_t vmsnevi\_mask\_e32xm8\_uint32xm8* (*e32xm8\_t merge*, *uint32xm8\_t a*, const unsigned int *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *e64xm1\_t vmsnevi\_mask\_e64xm1\_int64xm1* (*e64xm1\_t merge*, *int64xm1\_t a*, const long *b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *e64xm1\_t vmsnevi\_mask\_e64xm1\_uint64xm1* (*e64xm1\_t merge*, *uint64xm1\_t a*, const unsigned long *b*, *e64xm1\_t mask*, unsigned int *gvl*)



- `e64xm2_t vmsnevi_mask_e64xm2_int64xm2` (`e64xm2_t` merge, `int64xm2_t` *a*, const long *b*, `e64xm2_t` mask, unsigned int *gvl*)
- `e64xm2_t vmsnevi_mask_e64xm2_uint64xm2` (`e64xm2_t` merge, `uint64xm2_t` *a*, const unsigned long *b*, `e64xm2_t` mask, unsigned int *gvl*)
- `e64xm4_t vmsnevi_mask_e64xm4_int64xm4` (`e64xm4_t` merge, `int64xm4_t` *a*, const long *b*, `e64xm4_t` mask, unsigned int *gvl*)
- `e64xm4_t vmsnevi_mask_e64xm4_uint64xm4` (`e64xm4_t` merge, `uint64xm4_t` *a*, const unsigned long *b*, `e64xm4_t` mask, unsigned int *gvl*)
- `e64xm8_t vmsnevi_mask_e64xm8_int64xm8` (`e64xm8_t` merge, `int64xm8_t` *a*, const long *b*, `e64xm8_t` mask, unsigned int *gvl*)
- `e64xm8_t vmsnevi_mask_e64xm8_uint64xm8` (`e64xm8_t` merge, `uint64xm8_t` *a*, const unsigned long *b*, `e64xm8_t` mask, unsigned int *gvl*)
- `e8xm1_t vmsnevi_mask_e8xm1_int8xm1` (`e8xm1_t` merge, `int8xm1_t` *a*, const signed char *b*, `e8xm1_t` mask, unsigned int *gvl*)
- `e8xm1_t vmsnevi_mask_e8xm1_uint8xm1` (`e8xm1_t` merge, `uint8xm1_t` *a*, const unsigned char *b*, `e8xm1_t` mask, unsigned int *gvl*)
- `e8xm2_t vmsnevi_mask_e8xm2_int8xm2` (`e8xm2_t` merge, `int8xm2_t` *a*, const signed char *b*, `e8xm2_t` mask, unsigned int *gvl*)
- `e8xm2_t vmsnevi_mask_e8xm2_uint8xm2` (`e8xm2_t` merge, `uint8xm2_t` *a*, const unsigned char *b*, `e8xm2_t` mask, unsigned int *gvl*)
- `e8xm4_t vmsnevi_mask_e8xm4_int8xm4` (`e8xm4_t` merge, `int8xm4_t` *a*, const signed char *b*, `e8xm4_t` mask, unsigned int *gvl*)
- `e8xm4_t vmsnevi_mask_e8xm4_uint8xm4` (`e8xm4_t` merge, `uint8xm4_t` *a*, const unsigned char *b*, `e8xm4_t` mask, unsigned int *gvl*)
- `e8xm8_t vmsnevi_mask_e8xm8_int8xm8` (`e8xm8_t` merge, `int8xm8_t` *a*, const signed char *b*, `e8xm8_t` mask, unsigned int *gvl*)
- `e8xm8_t vmsnevi_mask_e8xm8_uint8xm8` (`e8xm8_t` merge, `uint8xm8_t` *a*, const unsigned char *b*, `e8xm8_t` mask, unsigned int *gvl*)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] != b
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.8.21 Compare elementwise vector-vector for inequality

**Instruction:** [`'vmsne.vv'`]

**Prototypes:**

- `e16xm1_t vmsnevv_e16xm1_int16xm1` (`int16xm1_t` *a*, `int16xm1_t` *b*, unsigned int *gvl*)
- `e16xm1_t vmsnevv_e16xm1_uint16xm1` (`uint16xm1_t` *a*, `uint16xm1_t` *b*, unsigned int *gvl*)
- `e16xm2_t vmsnevv_e16xm2_int16xm2` (`int16xm2_t` *a*, `int16xm2_t` *b*, unsigned int *gvl*)
- `e16xm2_t vmsnevv_e16xm2_uint16xm2` (`uint16xm2_t` *a*, `uint16xm2_t` *b*, unsigned int *gvl*)
- `e16xm4_t vmsnevv_e16xm4_int16xm4` (`int16xm4_t` *a*, `int16xm4_t` *b*, unsigned int *gvl*)



- *e16xm4\_t vmsnevv\_e16xm4\_uint16xm4* (*uint16xm4\_t a, uint16xm4\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmsnevv\_e16xm8\_int16xm8* (*int16xm8\_t a, int16xm8\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmsnevv\_e16xm8\_uint16xm8* (*uint16xm8\_t a, uint16xm8\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmsnevv\_e32xm1\_int32xm1* (*int32xm1\_t a, int32xm1\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmsnevv\_e32xm1\_uint32xm1* (*uint32xm1\_t a, uint32xm1\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmsnevv\_e32xm2\_int32xm2* (*int32xm2\_t a, int32xm2\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmsnevv\_e32xm2\_uint32xm2* (*uint32xm2\_t a, uint32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmsnevv\_e32xm4\_int32xm4* (*int32xm4\_t a, int32xm4\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmsnevv\_e32xm4\_uint32xm4* (*uint32xm4\_t a, uint32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmsnevv\_e32xm8\_int32xm8* (*int32xm8\_t a, int32xm8\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmsnevv\_e32xm8\_uint32xm8* (*uint32xm8\_t a, uint32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmsnevv\_e64xm1\_int64xm1* (*int64xm1\_t a, int64xm1\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmsnevv\_e64xm1\_uint64xm1* (*uint64xm1\_t a, uint64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmsnevv\_e64xm2\_int64xm2* (*int64xm2\_t a, int64xm2\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmsnevv\_e64xm2\_uint64xm2* (*uint64xm2\_t a, uint64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmsnevv\_e64xm4\_int64xm4* (*int64xm4\_t a, int64xm4\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmsnevv\_e64xm4\_uint64xm4* (*uint64xm4\_t a, uint64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmsnevv\_e64xm8\_int64xm8* (*int64xm8\_t a, int64xm8\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmsnevv\_e64xm8\_uint64xm8* (*uint64xm8\_t a, uint64xm8\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmsnevv\_e8xm1\_int8xm1* (*int8xm1\_t a, int8xm1\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmsnevv\_e8xm1\_uint8xm1* (*uint8xm1\_t a, uint8xm1\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmsnevv\_e8xm2\_int8xm2* (*int8xm2\_t a, int8xm2\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmsnevv\_e8xm2\_uint8xm2* (*uint8xm2\_t a, uint8xm2\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmsnevv\_e8xm4\_int8xm4* (*int8xm4\_t a, int8xm4\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmsnevv\_e8xm4\_uint8xm4* (*uint8xm4\_t a, uint8xm4\_t b*, unsigned int *gvl*)
- *e8xm8\_t vmsnevv\_e8xm8\_int8xm8* (*int8xm8\_t a, int8xm8\_t b*, unsigned int *gvl*)
- *e8xm8\_t vmsnevv\_e8xm8\_uint8xm8* (*uint8xm8\_t a, uint8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] != b[element]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmsnevv\_mask\_e16xm1\_int16xm1* (*e16xm1\_t merge, int16xm1\_t a, int16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm1\_t vmsnevv\_mask\_e16xm1\_uint16xm1* (*e16xm1\_t merge, uint16xm1\_t a, uint16xm1\_t b, e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmsnevv\_mask\_e16xm2\_int16xm2* (*e16xm2\_t merge, int16xm2\_t a, int16xm2\_t b, e16xm2\_t mask*, unsigned int *gvl*)

- *e16xm2\_t vmsnevv\_mask\_e16xm2\_uint16xm2* (*e16xm2\_t* merge, *uint16xm2\_t* a, *uint16xm2\_t* b, *e16xm2\_t* mask, unsigned int gvl)
- *e16xm4\_t vmsnevv\_mask\_e16xm4\_int16xm4* (*e16xm4\_t* merge, *int16xm4\_t* a, *int16xm4\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *e16xm4\_t vmsnevv\_mask\_e16xm4\_uint16xm4* (*e16xm4\_t* merge, *uint16xm4\_t* a, *uint16xm4\_t* b, *e16xm4\_t* mask, unsigned int gvl)
- *e16xm8\_t vmsnevv\_mask\_e16xm8\_int16xm8* (*e16xm8\_t* merge, *int16xm8\_t* a, *int16xm8\_t* b, *e16xm8\_t* mask, unsigned int gvl)
- *e16xm8\_t vmsnevv\_mask\_e16xm8\_uint16xm8* (*e16xm8\_t* merge, *uint16xm8\_t* a, *uint16xm8\_t* b, *e16xm8\_t* mask, unsigned int gvl)
- *e32xm1\_t vmsnevv\_mask\_e32xm1\_int32xm1* (*e32xm1\_t* merge, *int32xm1\_t* a, *int32xm1\_t* b, *e32xm1\_t* mask, unsigned int gvl)
- *e32xm1\_t vmsnevv\_mask\_e32xm1\_uint32xm1* (*e32xm1\_t* merge, *uint32xm1\_t* a, *uint32xm1\_t* b, *e32xm1\_t* mask, unsigned int gvl)
- *e32xm2\_t vmsnevv\_mask\_e32xm2\_int32xm2* (*e32xm2\_t* merge, *int32xm2\_t* a, *int32xm2\_t* b, *e32xm2\_t* mask, unsigned int gvl)
- *e32xm2\_t vmsnevv\_mask\_e32xm2\_uint32xm2* (*e32xm2\_t* merge, *uint32xm2\_t* a, *uint32xm2\_t* b, *e32xm2\_t* mask, unsigned int gvl)
- *e32xm4\_t vmsnevv\_mask\_e32xm4\_int32xm4* (*e32xm4\_t* merge, *int32xm4\_t* a, *int32xm4\_t* b, *e32xm4\_t* mask, unsigned int gvl)
- *e32xm4\_t vmsnevv\_mask\_e32xm4\_uint32xm4* (*e32xm4\_t* merge, *uint32xm4\_t* a, *uint32xm4\_t* b, *e32xm4\_t* mask, unsigned int gvl)
- *e32xm8\_t vmsnevv\_mask\_e32xm8\_int32xm8* (*e32xm8\_t* merge, *int32xm8\_t* a, *int32xm8\_t* b, *e32xm8\_t* mask, unsigned int gvl)
- *e32xm8\_t vmsnevv\_mask\_e32xm8\_uint32xm8* (*e32xm8\_t* merge, *uint32xm8\_t* a, *uint32xm8\_t* b, *e32xm8\_t* mask, unsigned int gvl)
- *e64xm1\_t vmsnevv\_mask\_e64xm1\_int64xm1* (*e64xm1\_t* merge, *int64xm1\_t* a, *int64xm1\_t* b, *e64xm1\_t* mask, unsigned int gvl)
- *e64xm1\_t vmsnevv\_mask\_e64xm1\_uint64xm1* (*e64xm1\_t* merge, *uint64xm1\_t* a, *uint64xm1\_t* b, *e64xm1\_t* mask, unsigned int gvl)
- *e64xm2\_t vmsnevv\_mask\_e64xm2\_int64xm2* (*e64xm2\_t* merge, *int64xm2\_t* a, *int64xm2\_t* b, *e64xm2\_t* mask, unsigned int gvl)
- *e64xm2\_t vmsnevv\_mask\_e64xm2\_uint64xm2* (*e64xm2\_t* merge, *uint64xm2\_t* a, *uint64xm2\_t* b, *e64xm2\_t* mask, unsigned int gvl)
- *e64xm4\_t vmsnevv\_mask\_e64xm4\_int64xm4* (*e64xm4\_t* merge, *int64xm4\_t* a, *int64xm4\_t* b, *e64xm4\_t* mask, unsigned int gvl)
- *e64xm4\_t vmsnevv\_mask\_e64xm4\_uint64xm4* (*e64xm4\_t* merge, *uint64xm4\_t* a, *uint64xm4\_t* b, *e64xm4\_t* mask, unsigned int gvl)
- *e64xm8\_t vmsnevv\_mask\_e64xm8\_int64xm8* (*e64xm8\_t* merge, *int64xm8\_t* a, *int64xm8\_t* b, *e64xm8\_t* mask, unsigned int gvl)
- *e64xm8\_t vmsnevv\_mask\_e64xm8\_uint64xm8* (*e64xm8\_t* merge, *uint64xm8\_t* a, *uint64xm8\_t* b, *e64xm8\_t* mask, unsigned int gvl)
- *e8xm1\_t vmsnevv\_mask\_e8xm1\_int8xm1* (*e8xm1\_t* merge, *int8xm1\_t* a, *int8xm1\_t* b, *e8xm1\_t* mask, unsigned int gvl)
- *e8xm1\_t vmsnevv\_mask\_e8xm1\_uint8xm1* (*e8xm1\_t* merge, *uint8xm1\_t* a, *uint8xm1\_t* b, *e8xm1\_t* mask, unsigned int gvl)

- `e8xm2_t vmsnevv_mask_e8xm2_int8xm2` (`e8xm2_t merge`, `int8xm2_t a`, `int8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `e8xm2_t vmsnevv_mask_e8xm2_uint8xm2` (`e8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int `gvl`)
- `e8xm4_t vmsnevv_mask_e8xm4_int8xm4` (`e8xm4_t merge`, `int8xm4_t a`, `int8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `e8xm4_t vmsnevv_mask_e8xm4_uint8xm4` (`e8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int `gvl`)
- `e8xm8_t vmsnevv_mask_e8xm8_int8xm8` (`e8xm8_t merge`, `int8xm8_t a`, `int8xm8_t b`, `e8xm8_t mask`, unsigned int `gvl`)
- `e8xm8_t vmsnevv_mask_e8xm8_uint8xm8` (`e8xm8_t merge`, `uint8xm8_t a`, `uint8xm8_t b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = a[element] != b[element]
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.8.22 Compare elementwise vector-scalar for inequality****Instruction:** [`'vmsne.vx'`]**Prototypes:**

- `e16xm1_t vmsnevx_e16xm1_int16xm1` (`int16xm1_t a`, short `b`, unsigned int `gvl`)
- `e16xm1_t vmsnevx_e16xm1_uint16xm1` (`uint16xm1_t a`, unsigned short `b`, unsigned int `gvl`)
- `e16xm2_t vmsnevx_e16xm2_int16xm2` (`int16xm2_t a`, short `b`, unsigned int `gvl`)
- `e16xm2_t vmsnevx_e16xm2_uint16xm2` (`uint16xm2_t a`, unsigned short `b`, unsigned int `gvl`)
- `e16xm4_t vmsnevx_e16xm4_int16xm4` (`int16xm4_t a`, short `b`, unsigned int `gvl`)
- `e16xm4_t vmsnevx_e16xm4_uint16xm4` (`uint16xm4_t a`, unsigned short `b`, unsigned int `gvl`)
- `e16xm8_t vmsnevx_e16xm8_int16xm8` (`int16xm8_t a`, short `b`, unsigned int `gvl`)
- `e16xm8_t vmsnevx_e16xm8_uint16xm8` (`uint16xm8_t a`, unsigned short `b`, unsigned int `gvl`)
- `e32xm1_t vmsnevx_e32xm1_int32xm1` (`int32xm1_t a`, int `b`, unsigned int `gvl`)
- `e32xm1_t vmsnevx_e32xm1_uint32xm1` (`uint32xm1_t a`, unsigned int `b`, unsigned int `gvl`)
- `e32xm2_t vmsnevx_e32xm2_int32xm2` (`int32xm2_t a`, int `b`, unsigned int `gvl`)
- `e32xm2_t vmsnevx_e32xm2_uint32xm2` (`uint32xm2_t a`, unsigned int `b`, unsigned int `gvl`)
- `e32xm4_t vmsnevx_e32xm4_int32xm4` (`int32xm4_t a`, int `b`, unsigned int `gvl`)
- `e32xm4_t vmsnevx_e32xm4_uint32xm4` (`uint32xm4_t a`, unsigned int `b`, unsigned int `gvl`)
- `e32xm8_t vmsnevx_e32xm8_int32xm8` (`int32xm8_t a`, int `b`, unsigned int `gvl`)
- `e32xm8_t vmsnevx_e32xm8_uint32xm8` (`uint32xm8_t a`, unsigned int `b`, unsigned int `gvl`)
- `e64xm1_t vmsnevx_e64xm1_int64xm1` (`int64xm1_t a`, long `b`, unsigned int `gvl`)

- `e64xm1_t vmsnevx_e64xm1_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `e64xm2_t vmsnevx_e64xm2_int64xm2 (int64xm2_t a, long b, unsigned int gvl)`
- `e64xm2_t vmsnevx_e64xm2_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `e64xm4_t vmsnevx_e64xm4_int64xm4 (int64xm4_t a, long b, unsigned int gvl)`
- `e64xm4_t vmsnevx_e64xm4_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `e64xm8_t vmsnevx_e64xm8_int64xm8 (int64xm8_t a, long b, unsigned int gvl)`
- `e64xm8_t vmsnevx_e64xm8_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `e8xm1_t vmsnevx_e8xm1_int8xm1 (int8xm1_t a, signed char b, unsigned int gvl)`
- `e8xm1_t vmsnevx_e8xm1_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `e8xm2_t vmsnevx_e8xm2_int8xm2 (int8xm2_t a, signed char b, unsigned int gvl)`
- `e8xm2_t vmsnevx_e8xm2_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `e8xm4_t vmsnevx_e8xm4_int8xm4 (int8xm4_t a, signed char b, unsigned int gvl)`
- `e8xm4_t vmsnevx_e8xm4_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `e8xm8_t vmsnevx_e8xm8_int8xm8 (int8xm8_t a, signed char b, unsigned int gvl)`
- `e8xm8_t vmsnevx_e8xm8_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element] != b
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `e16xm1_t vmsnevx_mask_e16xm1_int16xm1 (e16xm1_t merge, int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `e16xm1_t vmsnevx_mask_e16xm1_uint16xm1 (e16xm1_t merge, uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `e16xm2_t vmsnevx_mask_e16xm2_int16xm2 (e16xm2_t merge, int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `e16xm2_t vmsnevx_mask_e16xm2_uint16xm2 (e16xm2_t merge, uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `e16xm4_t vmsnevx_mask_e16xm4_int16xm4 (e16xm4_t merge, int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `e16xm4_t vmsnevx_mask_e16xm4_uint16xm4 (e16xm4_t merge, uint16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `e16xm8_t vmsnevx_mask_e16xm8_int16xm8 (e16xm8_t merge, int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `e16xm8_t vmsnevx_mask_e16xm8_uint16xm8 (e16xm8_t merge, uint16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `e32xm1_t vmsnevx_mask_e32xm1_int32xm1 (e32xm1_t merge, int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `e32xm1_t vmsnevx_mask_e32xm1_uint32xm1 (e32xm1_t merge, uint32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`

- `e32xm2_t vmsnevx_mask_e32xm2_int32xm2` (`e32xm2_t` merge, `int32xm2_t` *a*, `int` *b*, `e32xm2_t` mask, unsigned int gvl)
- `e32xm2_t vmsnevx_mask_e32xm2_uint32xm2` (`e32xm2_t` merge, `uint32xm2_t` *a*, unsigned `int` *b*, `e32xm2_t` mask, unsigned int gvl)
- `e32xm4_t vmsnevx_mask_e32xm4_int32xm4` (`e32xm4_t` merge, `int32xm4_t` *a*, `int` *b*, `e32xm4_t` mask, unsigned int gvl)
- `e32xm4_t vmsnevx_mask_e32xm4_uint32xm4` (`e32xm4_t` merge, `uint32xm4_t` *a*, unsigned `int` *b*, `e32xm4_t` mask, unsigned int gvl)
- `e32xm8_t vmsnevx_mask_e32xm8_int32xm8` (`e32xm8_t` merge, `int32xm8_t` *a*, `int` *b*, `e32xm8_t` mask, unsigned int gvl)
- `e32xm8_t vmsnevx_mask_e32xm8_uint32xm8` (`e32xm8_t` merge, `uint32xm8_t` *a*, unsigned `int` *b*, `e32xm8_t` mask, unsigned int gvl)
- `e64xm1_t vmsnevx_mask_e64xm1_int64xm1` (`e64xm1_t` merge, `int64xm1_t` *a*, `long` *b*, `e64xm1_t` mask, unsigned int gvl)
- `e64xm1_t vmsnevx_mask_e64xm1_uint64xm1` (`e64xm1_t` merge, `uint64xm1_t` *a*, unsigned `long` *b*, `e64xm1_t` mask, unsigned int gvl)
- `e64xm2_t vmsnevx_mask_e64xm2_int64xm2` (`e64xm2_t` merge, `int64xm2_t` *a*, `long` *b*, `e64xm2_t` mask, unsigned int gvl)
- `e64xm2_t vmsnevx_mask_e64xm2_uint64xm2` (`e64xm2_t` merge, `uint64xm2_t` *a*, unsigned `long` *b*, `e64xm2_t` mask, unsigned int gvl)
- `e64xm4_t vmsnevx_mask_e64xm4_int64xm4` (`e64xm4_t` merge, `int64xm4_t` *a*, `long` *b*, `e64xm4_t` mask, unsigned int gvl)
- `e64xm4_t vmsnevx_mask_e64xm4_uint64xm4` (`e64xm4_t` merge, `uint64xm4_t` *a*, unsigned `long` *b*, `e64xm4_t` mask, unsigned int gvl)
- `e64xm8_t vmsnevx_mask_e64xm8_int64xm8` (`e64xm8_t` merge, `int64xm8_t` *a*, `long` *b*, `e64xm8_t` mask, unsigned int gvl)
- `e64xm8_t vmsnevx_mask_e64xm8_uint64xm8` (`e64xm8_t` merge, `uint64xm8_t` *a*, unsigned `long` *b*, `e64xm8_t` mask, unsigned int gvl)
- `e8xm1_t vmsnevx_mask_e8xm1_int8xm1` (`e8xm1_t` merge, `int8xm1_t` *a*, `signed char` *b*, `e8xm1_t` mask, unsigned int gvl)
- `e8xm1_t vmsnevx_mask_e8xm1_uint8xm1` (`e8xm1_t` merge, `uint8xm1_t` *a*, unsigned `char` *b*, `e8xm1_t` mask, unsigned int gvl)
- `e8xm2_t vmsnevx_mask_e8xm2_int8xm2` (`e8xm2_t` merge, `int8xm2_t` *a*, `signed char` *b*, `e8xm2_t` mask, unsigned int gvl)
- `e8xm2_t vmsnevx_mask_e8xm2_uint8xm2` (`e8xm2_t` merge, `uint8xm2_t` *a*, unsigned `char` *b*, `e8xm2_t` mask, unsigned int gvl)
- `e8xm4_t vmsnevx_mask_e8xm4_int8xm4` (`e8xm4_t` merge, `int8xm4_t` *a*, `signed char` *b*, `e8xm4_t` mask, unsigned int gvl)
- `e8xm4_t vmsnevx_mask_e8xm4_uint8xm4` (`e8xm4_t` merge, `uint8xm4_t` *a*, unsigned `char` *b*, `e8xm4_t` mask, unsigned int gvl)
- `e8xm8_t vmsnevx_mask_e8xm8_int8xm8` (`e8xm8_t` merge, `int8xm8_t` *a*, `signed char` *b*, `e8xm8_t` mask, unsigned int gvl)
- `e8xm8_t vmsnevx_mask_e8xm8_uint8xm8` (`e8xm8_t` merge, `uint8xm8_t` *a*, unsigned `char` *b*, `e8xm8_t` mask, unsigned int gvl)

Masked operation:



```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = a[element] != b
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.9 Memory accesses

### 2.9.1 Load 8b signed in memory to vector

**Instruction:** [`vlb.v`']

**Prototypes:**

- `int16xm1_t vlbv_int16xm1` (const short \*address, unsigned int gvl)
- `int16xm2_t vlbv_int16xm2` (const short \*address, unsigned int gvl)
- `int16xm4_t vlbv_int16xm4` (const short \*address, unsigned int gvl)
- `int16xm8_t vlbv_int16xm8` (const short \*address, unsigned int gvl)
- `int32xm1_t vlbv_int32xm1` (const int \*address, unsigned int gvl)
- `int32xm2_t vlbv_int32xm2` (const int \*address, unsigned int gvl)
- `int32xm4_t vlbv_int32xm4` (const int \*address, unsigned int gvl)
- `int32xm8_t vlbv_int32xm8` (const int \*address, unsigned int gvl)
- `int64xm1_t vlbv_int64xm1` (const long \*address, unsigned int gvl)
- `int64xm2_t vlbv_int64xm2` (const long \*address, unsigned int gvl)
- `int64xm4_t vlbv_int64xm4` (const long \*address, unsigned int gvl)
- `int64xm8_t vlbv_int64xm8` (const long \*address, unsigned int gvl)
- `int8xm1_t vlbv_int8xm1` (const signed char \*address, unsigned int gvl)
- `int8xm2_t vlbv_int8xm2` (const signed char \*address, unsigned int gvl)
- `int8xm4_t vlbv_int8xm4` (const signed char \*address, unsigned int gvl)
- `int8xm8_t vlbv_int8xm8` (const signed char \*address, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = load_element(address)
    address = address + 1
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vlbv_mask_int16xm1` (`int16xm1_t merge`, const short \*address, `e16xm1_t mask`, unsigned int gvl)
- `int16xm2_t vlbv_mask_int16xm2` (`int16xm2_t merge`, const short \*address, `e16xm2_t mask`, unsigned int gvl)



- `int16xm4_t vlbv_mask_int16xm4` (`int16xm4_t` merge, const short `*address`, `e16xm4_t` mask, unsigned int gvl)
- `int16xm8_t vlbv_mask_int16xm8` (`int16xm8_t` merge, const short `*address`, `e16xm8_t` mask, unsigned int gvl)
- `int32xm1_t vlbv_mask_int32xm1` (`int32xm1_t` merge, const int `*address`, `e32xm1_t` mask, unsigned int gvl)
- `int32xm2_t vlbv_mask_int32xm2` (`int32xm2_t` merge, const int `*address`, `e32xm2_t` mask, unsigned int gvl)
- `int32xm4_t vlbv_mask_int32xm4` (`int32xm4_t` merge, const int `*address`, `e32xm4_t` mask, unsigned int gvl)
- `int32xm8_t vlbv_mask_int32xm8` (`int32xm8_t` merge, const int `*address`, `e32xm8_t` mask, unsigned int gvl)
- `int64xm1_t vlbv_mask_int64xm1` (`int64xm1_t` merge, const long `*address`, `e64xm1_t` mask, unsigned int gvl)
- `int64xm2_t vlbv_mask_int64xm2` (`int64xm2_t` merge, const long `*address`, `e64xm2_t` mask, unsigned int gvl)
- `int64xm4_t vlbv_mask_int64xm4` (`int64xm4_t` merge, const long `*address`, `e64xm4_t` mask, unsigned int gvl)
- `int64xm8_t vlbv_mask_int64xm8` (`int64xm8_t` merge, const long `*address`, `e64xm8_t` mask, unsigned int gvl)
- `int8xm1_t vlbv_mask_int8xm1` (`int8xm1_t` merge, const signed char `*address`, `e8xm1_t` mask, unsigned int gvl)
- `int8xm2_t vlbv_mask_int8xm2` (`int8xm2_t` merge, const signed char `*address`, `e8xm2_t` mask, unsigned int gvl)
- `int8xm4_t vlbv_mask_int8xm4` (`int8xm4_t` merge, const signed char `*address`, `e8xm4_t` mask, unsigned int gvl)
- `int8xm8_t vlbv_mask_int8xm8` (`int8xm8_t` merge, const signed char `*address`, `e8xm8_t` mask, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = load_element(address)
        address = address + 1
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.9.2 Load 8b unsigned in memory to vector

**Instruction:** [`vlbu.v`']**Prototypes:**

- `uint16xm1_t vlbu_v_uint16xm1` (const unsigned short `*address`, unsigned int gvl)
- `uint16xm2_t vlbu_v_uint16xm2` (const unsigned short `*address`, unsigned int gvl)
- `uint16xm4_t vlbu_v_uint16xm4` (const unsigned short `*address`, unsigned int gvl)
- `uint16xm8_t vlbu_v_uint16xm8` (const unsigned short `*address`, unsigned int gvl)

- `uint32xm1_t vlbuv_uint32xm1` (const unsigned int \*address, unsigned int gvl)
- `uint32xm2_t vlbuv_uint32xm2` (const unsigned int \*address, unsigned int gvl)
- `uint32xm4_t vlbuv_uint32xm4` (const unsigned int \*address, unsigned int gvl)
- `uint32xm8_t vlbuv_uint32xm8` (const unsigned int \*address, unsigned int gvl)
- `uint64xm1_t vlbuv_uint64xm1` (const unsigned long \*address, unsigned int gvl)
- `uint64xm2_t vlbuv_uint64xm2` (const unsigned long \*address, unsigned int gvl)
- `uint64xm4_t vlbuv_uint64xm4` (const unsigned long \*address, unsigned int gvl)
- `uint64xm8_t vlbuv_uint64xm8` (const unsigned long \*address, unsigned int gvl)
- `uint8xm1_t vlbuv_uint8xm1` (const unsigned char \*address, unsigned int gvl)
- `uint8xm2_t vlbuv_uint8xm2` (const unsigned char \*address, unsigned int gvl)
- `uint8xm4_t vlbuv_uint8xm4` (const unsigned char \*address, unsigned int gvl)
- `uint8xm8_t vlbuv_uint8xm8` (const unsigned char \*address, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = load_element(address)
      address = address + 1
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vlbuv_mask_uint16xm1` (`uint16xm1_t merge`, const unsigned short \*address, `e16xm1_t mask`, unsigned int gvl)
- `uint16xm2_t vlbuv_mask_uint16xm2` (`uint16xm2_t merge`, const unsigned short \*address, `e16xm2_t mask`, unsigned int gvl)
- `uint16xm4_t vlbuv_mask_uint16xm4` (`uint16xm4_t merge`, const unsigned short \*address, `e16xm4_t mask`, unsigned int gvl)
- `uint16xm8_t vlbuv_mask_uint16xm8` (`uint16xm8_t merge`, const unsigned short \*address, `e16xm8_t mask`, unsigned int gvl)
- `uint32xm1_t vlbuv_mask_uint32xm1` (`uint32xm1_t merge`, const unsigned int \*address, `e32xm1_t mask`, unsigned int gvl)
- `uint32xm2_t vlbuv_mask_uint32xm2` (`uint32xm2_t merge`, const unsigned int \*address, `e32xm2_t mask`, unsigned int gvl)
- `uint32xm4_t vlbuv_mask_uint32xm4` (`uint32xm4_t merge`, const unsigned int \*address, `e32xm4_t mask`, unsigned int gvl)
- `uint32xm8_t vlbuv_mask_uint32xm8` (`uint32xm8_t merge`, const unsigned int \*address, `e32xm8_t mask`, unsigned int gvl)
- `uint64xm1_t vlbuv_mask_uint64xm1` (`uint64xm1_t merge`, const unsigned long \*address, `e64xm1_t mask`, unsigned int gvl)
- `uint64xm2_t vlbuv_mask_uint64xm2` (`uint64xm2_t merge`, const unsigned long \*address, `e64xm2_t mask`, unsigned int gvl)
- `uint64xm4_t vlbuv_mask_uint64xm4` (`uint64xm4_t merge`, const unsigned long \*address, `e64xm4_t mask`, unsigned int gvl)
- `uint64xm8_t vlbuv_mask_uint64xm8` (`uint64xm8_t merge`, const unsigned long \*address, `e64xm8_t mask`, unsigned int gvl)

- `uint8xm1_t vlbuv_mask_uint8xm1 (uint8xm1_t merge, const unsigned char *address, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vlbuv_mask_uint8xm2 (uint8xm2_t merge, const unsigned char *address, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vlbuv_mask_uint8xm4 (uint8xm4_t merge, const unsigned char *address, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vlbuv_mask_uint8xm8 (uint8xm8_t merge, const unsigned char *address, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = load_element (address)
        address = address + 1
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.9.3 Load elements in memory to vector

**Instruction:** [`'vle.v'`]**Prototypes:**

- `float16xm1_t vlev_float16xm1 (const float16_t *address, unsigned int gvl)`
- `float16xm2_t vlev_float16xm2 (const float16_t *address, unsigned int gvl)`
- `float16xm4_t vlev_float16xm4 (const float16_t *address, unsigned int gvl)`
- `float16xm8_t vlev_float16xm8 (const float16_t *address, unsigned int gvl)`
- `float32xm1_t vlev_float32xm1 (const float *address, unsigned int gvl)`
- `float32xm2_t vlev_float32xm2 (const float *address, unsigned int gvl)`
- `float32xm4_t vlev_float32xm4 (const float *address, unsigned int gvl)`
- `float32xm8_t vlev_float32xm8 (const float *address, unsigned int gvl)`
- `float64xm1_t vlev_float64xm1 (const double *address, unsigned int gvl)`
- `float64xm2_t vlev_float64xm2 (const double *address, unsigned int gvl)`
- `float64xm4_t vlev_float64xm4 (const double *address, unsigned int gvl)`
- `float64xm8_t vlev_float64xm8 (const double *address, unsigned int gvl)`
- `int16xm1_t vlev_int16xm1 (const short *address, unsigned int gvl)`
- `int16xm2_t vlev_int16xm2 (const short *address, unsigned int gvl)`
- `int16xm4_t vlev_int16xm4 (const short *address, unsigned int gvl)`
- `int16xm8_t vlev_int16xm8 (const short *address, unsigned int gvl)`
- `int32xm1_t vlev_int32xm1 (const int *address, unsigned int gvl)`
- `int32xm2_t vlev_int32xm2 (const int *address, unsigned int gvl)`
- `int32xm4_t vlev_int32xm4 (const int *address, unsigned int gvl)`

- `int32xm8_t vlev_int32xm8` (const int \*address, unsigned int gvl)
- `int64xm1_t vlev_int64xm1` (const long \*address, unsigned int gvl)
- `int64xm2_t vlev_int64xm2` (const long \*address, unsigned int gvl)
- `int64xm4_t vlev_int64xm4` (const long \*address, unsigned int gvl)
- `int64xm8_t vlev_int64xm8` (const long \*address, unsigned int gvl)
- `int8xm1_t vlev_int8xm1` (const signed char \*address, unsigned int gvl)
- `int8xm2_t vlev_int8xm2` (const signed char \*address, unsigned int gvl)
- `int8xm4_t vlev_int8xm4` (const signed char \*address, unsigned int gvl)
- `int8xm8_t vlev_int8xm8` (const signed char \*address, unsigned int gvl)
- `uint16xm1_t vlev_uint16xm1` (const unsigned short \*address, unsigned int gvl)
- `uint16xm2_t vlev_uint16xm2` (const unsigned short \*address, unsigned int gvl)
- `uint16xm4_t vlev_uint16xm4` (const unsigned short \*address, unsigned int gvl)
- `uint16xm8_t vlev_uint16xm8` (const unsigned short \*address, unsigned int gvl)
- `uint32xm1_t vlev_uint32xm1` (const unsigned int \*address, unsigned int gvl)
- `uint32xm2_t vlev_uint32xm2` (const unsigned int \*address, unsigned int gvl)
- `uint32xm4_t vlev_uint32xm4` (const unsigned int \*address, unsigned int gvl)
- `uint32xm8_t vlev_uint32xm8` (const unsigned int \*address, unsigned int gvl)
- `uint64xm1_t vlev_uint64xm1` (const unsigned long \*address, unsigned int gvl)
- `uint64xm2_t vlev_uint64xm2` (const unsigned long \*address, unsigned int gvl)
- `uint64xm4_t vlev_uint64xm4` (const unsigned long \*address, unsigned int gvl)
- `uint64xm8_t vlev_uint64xm8` (const unsigned long \*address, unsigned int gvl)
- `uint8xm1_t vlev_uint8xm1` (const unsigned char \*address, unsigned int gvl)
- `uint8xm2_t vlev_uint8xm2` (const unsigned char \*address, unsigned int gvl)
- `uint8xm4_t vlev_uint8xm4` (const unsigned char \*address, unsigned int gvl)
- `uint8xm8_t vlev_uint8xm8` (const unsigned char \*address, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = load_element(address)
    address = address + SEW / 8
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vlev_mask_float16xm1` (`float16xm1_t merge`, const `float16_t` \*address, `e16xm1_t mask`, unsigned int gvl)
- `float16xm2_t vlev_mask_float16xm2` (`float16xm2_t merge`, const `float16_t` \*address, `e16xm2_t mask`, unsigned int gvl)
- `float16xm4_t vlev_mask_float16xm4` (`float16xm4_t merge`, const `float16_t` \*address, `e16xm4_t mask`, unsigned int gvl)
- `float16xm8_t vlev_mask_float16xm8` (`float16xm8_t merge`, const `float16_t` \*address, `e16xm8_t mask`, unsigned int gvl)

- `float32xm1_t vlev_mask_float32xm1` (`float32xm1_t` merge, const float `*address`, `e32xm1_t` mask, unsigned int gvl)
- `float32xm2_t vlev_mask_float32xm2` (`float32xm2_t` merge, const float `*address`, `e32xm2_t` mask, unsigned int gvl)
- `float32xm4_t vlev_mask_float32xm4` (`float32xm4_t` merge, const float `*address`, `e32xm4_t` mask, unsigned int gvl)
- `float32xm8_t vlev_mask_float32xm8` (`float32xm8_t` merge, const float `*address`, `e32xm8_t` mask, unsigned int gvl)
- `float64xm1_t vlev_mask_float64xm1` (`float64xm1_t` merge, const double `*address`, `e64xm1_t` mask, unsigned int gvl)
- `float64xm2_t vlev_mask_float64xm2` (`float64xm2_t` merge, const double `*address`, `e64xm2_t` mask, unsigned int gvl)
- `float64xm4_t vlev_mask_float64xm4` (`float64xm4_t` merge, const double `*address`, `e64xm4_t` mask, unsigned int gvl)
- `float64xm8_t vlev_mask_float64xm8` (`float64xm8_t` merge, const double `*address`, `e64xm8_t` mask, unsigned int gvl)
- `int16xm1_t vlev_mask_int16xm1` (`int16xm1_t` merge, const short `*address`, `e16xm1_t` mask, unsigned int gvl)
- `int16xm2_t vlev_mask_int16xm2` (`int16xm2_t` merge, const short `*address`, `e16xm2_t` mask, unsigned int gvl)
- `int16xm4_t vlev_mask_int16xm4` (`int16xm4_t` merge, const short `*address`, `e16xm4_t` mask, unsigned int gvl)
- `int16xm8_t vlev_mask_int16xm8` (`int16xm8_t` merge, const short `*address`, `e16xm8_t` mask, unsigned int gvl)
- `int32xm1_t vlev_mask_int32xm1` (`int32xm1_t` merge, const int `*address`, `e32xm1_t` mask, unsigned int gvl)
- `int32xm2_t vlev_mask_int32xm2` (`int32xm2_t` merge, const int `*address`, `e32xm2_t` mask, unsigned int gvl)
- `int32xm4_t vlev_mask_int32xm4` (`int32xm4_t` merge, const int `*address`, `e32xm4_t` mask, unsigned int gvl)
- `int32xm8_t vlev_mask_int32xm8` (`int32xm8_t` merge, const int `*address`, `e32xm8_t` mask, unsigned int gvl)
- `int64xm1_t vlev_mask_int64xm1` (`int64xm1_t` merge, const long `*address`, `e64xm1_t` mask, unsigned int gvl)
- `int64xm2_t vlev_mask_int64xm2` (`int64xm2_t` merge, const long `*address`, `e64xm2_t` mask, unsigned int gvl)
- `int64xm4_t vlev_mask_int64xm4` (`int64xm4_t` merge, const long `*address`, `e64xm4_t` mask, unsigned int gvl)
- `int64xm8_t vlev_mask_int64xm8` (`int64xm8_t` merge, const long `*address`, `e64xm8_t` mask, unsigned int gvl)
- `int8xm1_t vlev_mask_int8xm1` (`int8xm1_t` merge, const signed char `*address`, `e8xm1_t` mask, unsigned int gvl)
- `int8xm2_t vlev_mask_int8xm2` (`int8xm2_t` merge, const signed char `*address`, `e8xm2_t` mask, unsigned int gvl)
- `int8xm4_t vlev_mask_int8xm4` (`int8xm4_t` merge, const signed char `*address`, `e8xm4_t` mask, unsigned int gvl)

- `int8xm8_t vlev_mask_int8xm8 (int8xm8_t merge, const signed char *address, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vlev_mask_uint16xm1 (uint16xm1_t merge, const unsigned short *address, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vlev_mask_uint16xm2 (uint16xm2_t merge, const unsigned short *address, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vlev_mask_uint16xm4 (uint16xm4_t merge, const unsigned short *address, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vlev_mask_uint16xm8 (uint16xm8_t merge, const unsigned short *address, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vlev_mask_uint32xm1 (uint32xm1_t merge, const unsigned int *address, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vlev_mask_uint32xm2 (uint32xm2_t merge, const unsigned int *address, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vlev_mask_uint32xm4 (uint32xm4_t merge, const unsigned int *address, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vlev_mask_uint32xm8 (uint32xm8_t merge, const unsigned int *address, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vlev_mask_uint64xm1 (uint64xm1_t merge, const unsigned long *address, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vlev_mask_uint64xm2 (uint64xm2_t merge, const unsigned long *address, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vlev_mask_uint64xm4 (uint64xm4_t merge, const unsigned long *address, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vlev_mask_uint64xm8 (uint64xm8_t merge, const unsigned long *address, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vlev_mask_uint8xm1 (uint8xm1_t merge, const unsigned char *address, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vlev_mask_uint8xm2 (uint8xm2_t merge, const unsigned char *address, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vlev_mask_uint8xm4 (uint8xm4_t merge, const unsigned char *address, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vlev_mask_uint8xm8 (uint8xm8_t merge, const unsigned char *address, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = load_element(address)
        address = address + SEW / 8
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.9.4 Load 16b signed in memory to vector

**Instruction:** [`vlh.v`']



**Prototypes:**

- *int16xm1\_t* **vlhv\_int16xm1** (const short \**address*, unsigned int *gvl*)
- *int16xm2\_t* **vlhv\_int16xm2** (const short \**address*, unsigned int *gvl*)
- *int16xm4\_t* **vlhv\_int16xm4** (const short \**address*, unsigned int *gvl*)
- *int16xm8\_t* **vlhv\_int16xm8** (const short \**address*, unsigned int *gvl*)
- *int32xm1\_t* **vlhv\_int32xm1** (const int \**address*, unsigned int *gvl*)
- *int32xm2\_t* **vlhv\_int32xm2** (const int \**address*, unsigned int *gvl*)
- *int32xm4\_t* **vlhv\_int32xm4** (const int \**address*, unsigned int *gvl*)
- *int32xm8\_t* **vlhv\_int32xm8** (const int \**address*, unsigned int *gvl*)
- *int64xm1\_t* **vlhv\_int64xm1** (const long \**address*, unsigned int *gvl*)
- *int64xm2\_t* **vlhv\_int64xm2** (const long \**address*, unsigned int *gvl*)
- *int64xm4\_t* **vlhv\_int64xm4** (const long \**address*, unsigned int *gvl*)
- *int64xm8\_t* **vlhv\_int64xm8** (const long \**address*, unsigned int *gvl*)
- *int8xm1\_t* **vlhv\_int8xm1** (const signed char \**address*, unsigned int *gvl*)
- *int8xm2\_t* **vlhv\_int8xm2** (const signed char \**address*, unsigned int *gvl*)
- *int8xm4\_t* **vlhv\_int8xm4** (const signed char \**address*, unsigned int *gvl*)
- *int8xm8\_t* **vlhv\_int8xm8** (const signed char \**address*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = load_element(address)
    address = address + 1
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t* **vlhv\_mask\_int16xm1** (*int16xm1\_t* merge, const short \**address*, *e16xm1\_t* mask, unsigned int *gvl*)
- *int16xm2\_t* **vlhv\_mask\_int16xm2** (*int16xm2\_t* merge, const short \**address*, *e16xm2\_t* mask, unsigned int *gvl*)
- *int16xm4\_t* **vlhv\_mask\_int16xm4** (*int16xm4\_t* merge, const short \**address*, *e16xm4\_t* mask, unsigned int *gvl*)
- *int16xm8\_t* **vlhv\_mask\_int16xm8** (*int16xm8\_t* merge, const short \**address*, *e16xm8\_t* mask, unsigned int *gvl*)
- *int32xm1\_t* **vlhv\_mask\_int32xm1** (*int32xm1\_t* merge, const int \**address*, *e32xm1\_t* mask, unsigned int *gvl*)
- *int32xm2\_t* **vlhv\_mask\_int32xm2** (*int32xm2\_t* merge, const int \**address*, *e32xm2\_t* mask, unsigned int *gvl*)
- *int32xm4\_t* **vlhv\_mask\_int32xm4** (*int32xm4\_t* merge, const int \**address*, *e32xm4\_t* mask, unsigned int *gvl*)
- *int32xm8\_t* **vlhv\_mask\_int32xm8** (*int32xm8\_t* merge, const int \**address*, *e32xm8\_t* mask, unsigned int *gvl*)
- *int64xm1\_t* **vlhv\_mask\_int64xm1** (*int64xm1\_t* merge, const long \**address*, *e64xm1\_t* mask, unsigned int *gvl*)

- `int64xm2_t vlhv_mask_int64xm2` (`int64xm2_t merge`, `const long *address`, `e64xm2_t mask`, `unsigned int gvl`)
- `int64xm4_t vlhv_mask_int64xm4` (`int64xm4_t merge`, `const long *address`, `e64xm4_t mask`, `unsigned int gvl`)
- `int64xm8_t vlhv_mask_int64xm8` (`int64xm8_t merge`, `const long *address`, `e64xm8_t mask`, `unsigned int gvl`)
- `int8xm1_t vlhv_mask_int8xm1` (`int8xm1_t merge`, `const signed char *address`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8xm2_t vlhv_mask_int8xm2` (`int8xm2_t merge`, `const signed char *address`, `e8xm2_t mask`, `unsigned int gvl`)
- `int8xm4_t vlhv_mask_int8xm4` (`int8xm4_t merge`, `const signed char *address`, `e8xm4_t mask`, `unsigned int gvl`)
- `int8xm8_t vlhv_mask_int8xm8` (`int8xm8_t merge`, `const signed char *address`, `e8xm8_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = load_element(address)
        address = address + 1
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.9.5 Load 16b unsigned in memory to vector

**Instruction:** [`'vlhu.v'`]

**Prototypes:**

- `uint16xm1_t vlhuv_uint16xm1` (`const unsigned short *address`, `unsigned int gvl`)
- `uint16xm2_t vlhuv_uint16xm2` (`const unsigned short *address`, `unsigned int gvl`)
- `uint16xm4_t vlhuv_uint16xm4` (`const unsigned short *address`, `unsigned int gvl`)
- `uint16xm8_t vlhuv_uint16xm8` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32xm1_t vlhuv_uint32xm1` (`const unsigned int *address`, `unsigned int gvl`)
- `uint32xm2_t vlhuv_uint32xm2` (`const unsigned int *address`, `unsigned int gvl`)
- `uint32xm4_t vlhuv_uint32xm4` (`const unsigned int *address`, `unsigned int gvl`)
- `uint32xm8_t vlhuv_uint32xm8` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64xm1_t vlhuv_uint64xm1` (`const unsigned long *address`, `unsigned int gvl`)
- `uint64xm2_t vlhuv_uint64xm2` (`const unsigned long *address`, `unsigned int gvl`)
- `uint64xm4_t vlhuv_uint64xm4` (`const unsigned long *address`, `unsigned int gvl`)
- `uint64xm8_t vlhuv_uint64xm8` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8xm1_t vlhuv_uint8xm1` (`const unsigned char *address`, `unsigned int gvl`)
- `uint8xm2_t vlhuv_uint8xm2` (`const unsigned char *address`, `unsigned int gvl`)
- `uint8xm4_t vlhuv_uint8xm4` (`const unsigned char *address`, `unsigned int gvl`)

- `uint8xm8_t vlhuv_uint8xm8` (const unsigned char \*address, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = load_element(address)
      address = address + 1
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vlhuv_mask_uint16xm1` (`uint16xm1_t merge`, const unsigned short \*address, `e16xm1_t mask`, unsigned int gvl)
- `uint16xm2_t vlhuv_mask_uint16xm2` (`uint16xm2_t merge`, const unsigned short \*address, `e16xm2_t mask`, unsigned int gvl)
- `uint16xm4_t vlhuv_mask_uint16xm4` (`uint16xm4_t merge`, const unsigned short \*address, `e16xm4_t mask`, unsigned int gvl)
- `uint16xm8_t vlhuv_mask_uint16xm8` (`uint16xm8_t merge`, const unsigned short \*address, `e16xm8_t mask`, unsigned int gvl)
- `uint32xm1_t vlhuv_mask_uint32xm1` (`uint32xm1_t merge`, const unsigned int \*address, `e32xm1_t mask`, unsigned int gvl)
- `uint32xm2_t vlhuv_mask_uint32xm2` (`uint32xm2_t merge`, const unsigned int \*address, `e32xm2_t mask`, unsigned int gvl)
- `uint32xm4_t vlhuv_mask_uint32xm4` (`uint32xm4_t merge`, const unsigned int \*address, `e32xm4_t mask`, unsigned int gvl)
- `uint32xm8_t vlhuv_mask_uint32xm8` (`uint32xm8_t merge`, const unsigned int \*address, `e32xm8_t mask`, unsigned int gvl)
- `uint64xm1_t vlhuv_mask_uint64xm1` (`uint64xm1_t merge`, const unsigned long \*address, `e64xm1_t mask`, unsigned int gvl)
- `uint64xm2_t vlhuv_mask_uint64xm2` (`uint64xm2_t merge`, const unsigned long \*address, `e64xm2_t mask`, unsigned int gvl)
- `uint64xm4_t vlhuv_mask_uint64xm4` (`uint64xm4_t merge`, const unsigned long \*address, `e64xm4_t mask`, unsigned int gvl)
- `uint64xm8_t vlhuv_mask_uint64xm8` (`uint64xm8_t merge`, const unsigned long \*address, `e64xm8_t mask`, unsigned int gvl)
- `uint8xm1_t vlhuv_mask_uint8xm1` (`uint8xm1_t merge`, const unsigned char \*address, `e8xm1_t mask`, unsigned int gvl)
- `uint8xm2_t vlhuv_mask_uint8xm2` (`uint8xm2_t merge`, const unsigned char \*address, `e8xm2_t mask`, unsigned int gvl)
- `uint8xm4_t vlhuv_mask_uint8xm4` (`uint8xm4_t merge`, const unsigned char \*address, `e8xm4_t mask`, unsigned int gvl)
- `uint8xm8_t vlhuv_mask_uint8xm8` (`uint8xm8_t merge`, const unsigned char \*address, `e8xm8_t mask`, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = load_element(address)
        address = address + 1
      else
```

(continues on next page)

(continued from previous page)

```
result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.9.6 Load strided 8b signed in memory to vector

**Instruction:** [*vlsb.v*']

**Prototypes:**

- *int16xm1\_t vlsbv\_int16xm1* (const short *\*address*, long *stride*, unsigned int *gvl*)
- *int16xm2\_t vlsbv\_int16xm2* (const short *\*address*, long *stride*, unsigned int *gvl*)
- *int16xm4\_t vlsbv\_int16xm4* (const short *\*address*, long *stride*, unsigned int *gvl*)
- *int16xm8\_t vlsbv\_int16xm8* (const short *\*address*, long *stride*, unsigned int *gvl*)
- *int32xm1\_t vlsbv\_int32xm1* (const int *\*address*, long *stride*, unsigned int *gvl*)
- *int32xm2\_t vlsbv\_int32xm2* (const int *\*address*, long *stride*, unsigned int *gvl*)
- *int32xm4\_t vlsbv\_int32xm4* (const int *\*address*, long *stride*, unsigned int *gvl*)
- *int32xm8\_t vlsbv\_int32xm8* (const int *\*address*, long *stride*, unsigned int *gvl*)
- *int64xm1\_t vlsbv\_int64xm1* (const long *\*address*, long *stride*, unsigned int *gvl*)
- *int64xm2\_t vlsbv\_int64xm2* (const long *\*address*, long *stride*, unsigned int *gvl*)
- *int64xm4\_t vlsbv\_int64xm4* (const long *\*address*, long *stride*, unsigned int *gvl*)
- *int64xm8\_t vlsbv\_int64xm8* (const long *\*address*, long *stride*, unsigned int *gvl*)
- *int8xm1\_t vlsbv\_int8xm1* (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- *int8xm2\_t vlsbv\_int8xm2* (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- *int8xm4\_t vlsbv\_int8xm4* (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- *int8xm8\_t vlsbv\_int8xm8* (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = load_element(address)
    address = address + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vlsbv\_mask\_int16xm1* (*int16xm1\_t merge*, const short *\*address*, long *stride*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vlsbv\_mask\_int16xm2* (*int16xm2\_t merge*, const short *\*address*, long *stride*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vlsbv\_mask\_int16xm4* (*int16xm4\_t merge*, const short *\*address*, long *stride*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vlsbv\_mask\_int16xm8* (*int16xm8\_t merge*, const short *\*address*, long *stride*, *e16xm8\_t mask*, unsigned int *gvl*)
- *int32xm1\_t vlsbv\_mask\_int32xm1* (*int32xm1\_t merge*, const int *\*address*, long *stride*, *e32xm1\_t mask*, unsigned int *gvl*)

- `int32xm2_t vlsbv_mask_int32xm2(int32xm2_t merge, const int *address, long stride, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vlsbv_mask_int32xm4(int32xm4_t merge, const int *address, long stride, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vlsbv_mask_int32xm8(int32xm8_t merge, const int *address, long stride, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vlsbv_mask_int64xm1(int64xm1_t merge, const long *address, long stride, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vlsbv_mask_int64xm2(int64xm2_t merge, const long *address, long stride, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vlsbv_mask_int64xm4(int64xm4_t merge, const long *address, long stride, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vlsbv_mask_int64xm8(int64xm8_t merge, const long *address, long stride, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vlsbv_mask_int8xm1(int8xm1_t merge, const signed char *address, long stride, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vlsbv_mask_int8xm2(int8xm2_t merge, const signed char *address, long stride, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vlsbv_mask_int8xm4(int8xm4_t merge, const signed char *address, long stride, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vlsbv_mask_int8xm8(int8xm8_t merge, const signed char *address, long stride, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = load_element(address)
      else
        result[element] = merge[element]
      address = address + stride
result[gvl : VLMAX] = 0
```

**2.9.7 Load strided 8b unsigned in memory to vector****Instruction:** [`vlsbu.v'`]**Prototypes:**

- `uint16xm1_t vlsbu_v_uint16xm1(const unsigned short *address, long stride, unsigned int gvl)`
- `uint16xm2_t vlsbu_v_uint16xm2(const unsigned short *address, long stride, unsigned int gvl)`
- `uint16xm4_t vlsbu_v_uint16xm4(const unsigned short *address, long stride, unsigned int gvl)`
- `uint16xm8_t vlsbu_v_uint16xm8(const unsigned short *address, long stride, unsigned int gvl)`
- `uint32xm1_t vlsbu_v_uint32xm1(const unsigned int *address, long stride, unsigned int gvl)`
- `uint32xm2_t vlsbu_v_uint32xm2(const unsigned int *address, long stride, unsigned int gvl)`
- `uint32xm4_t vlsbu_v_uint32xm4(const unsigned int *address, long stride, unsigned int gvl)`
- `uint32xm8_t vlsbu_v_uint32xm8(const unsigned int *address, long stride, unsigned int gvl)`
- `uint64xm1_t vlsbu_v_uint64xm1(const unsigned long *address, long stride, unsigned int gvl)`

- `uint64xm2_t vlsbuu_uint64xm2` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64xm4_t vlsbuu_uint64xm4` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64xm8_t vlsbuu_uint64xm8` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8xm1_t vlsbuu_uint8xm1` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8xm2_t vlsbuu_uint8xm2` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8xm4_t vlsbuu_uint8xm4` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8xm8_t vlsbuu_uint8xm8` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = load_element(address)
      address = address + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vlsbuu_mask_uint16xm1` (`uint16xm1_t merge`, const unsigned short *\*address*, long *stride*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16xm2_t vlsbuu_mask_uint16xm2` (`uint16xm2_t merge`, const unsigned short *\*address*, long *stride*, `e16xm2_t mask`, unsigned int *gvl*)
- `uint16xm4_t vlsbuu_mask_uint16xm4` (`uint16xm4_t merge`, const unsigned short *\*address*, long *stride*, `e16xm4_t mask`, unsigned int *gvl*)
- `uint16xm8_t vlsbuu_mask_uint16xm8` (`uint16xm8_t merge`, const unsigned short *\*address*, long *stride*, `e16xm8_t mask`, unsigned int *gvl*)
- `uint32xm1_t vlsbuu_mask_uint32xm1` (`uint32xm1_t merge`, const unsigned int *\*address*, long *stride*, `e32xm1_t mask`, unsigned int *gvl*)
- `uint32xm2_t vlsbuu_mask_uint32xm2` (`uint32xm2_t merge`, const unsigned int *\*address*, long *stride*, `e32xm2_t mask`, unsigned int *gvl*)
- `uint32xm4_t vlsbuu_mask_uint32xm4` (`uint32xm4_t merge`, const unsigned int *\*address*, long *stride*, `e32xm4_t mask`, unsigned int *gvl*)
- `uint32xm8_t vlsbuu_mask_uint32xm8` (`uint32xm8_t merge`, const unsigned int *\*address*, long *stride*, `e32xm8_t mask`, unsigned int *gvl*)
- `uint64xm1_t vlsbuu_mask_uint64xm1` (`uint64xm1_t merge`, const unsigned long *\*address*, long *stride*, `e64xm1_t mask`, unsigned int *gvl*)
- `uint64xm2_t vlsbuu_mask_uint64xm2` (`uint64xm2_t merge`, const unsigned long *\*address*, long *stride*, `e64xm2_t mask`, unsigned int *gvl*)
- `uint64xm4_t vlsbuu_mask_uint64xm4` (`uint64xm4_t merge`, const unsigned long *\*address*, long *stride*, `e64xm4_t mask`, unsigned int *gvl*)
- `uint64xm8_t vlsbuu_mask_uint64xm8` (`uint64xm8_t merge`, const unsigned long *\*address*, long *stride*, `e64xm8_t mask`, unsigned int *gvl*)
- `uint8xm1_t vlsbuu_mask_uint8xm1` (`uint8xm1_t merge`, const unsigned char *\*address*, long *stride*, `e8xm1_t mask`, unsigned int *gvl*)
- `uint8xm2_t vlsbuu_mask_uint8xm2` (`uint8xm2_t merge`, const unsigned char *\*address*, long *stride*, `e8xm2_t mask`, unsigned int *gvl*)
- `uint8xm4_t vlsbuu_mask_uint8xm4` (`uint8xm4_t merge`, const unsigned char *\*address*, long *stride*, `e8xm4_t mask`, unsigned int *gvl*)



- `uint8xm8_t vlsbuvmask_uint8xm8 (uint8xm8_t merge, const unsigned char *address, long stride, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = load_element(address)
    else
        result[element] = merge[element]
    address = address + stride
result[gvl : VLMAX] = 0
```

## 2.9.8 Load strided elements in memory to vector

**Instruction:** [`'vlse.v'`]

**Prototypes:**

- `float16xm1_t vlsev_float16xm1 (const float16_t *address, long stride, unsigned int gvl)`
- `float16xm2_t vlsev_float16xm2 (const float16_t *address, long stride, unsigned int gvl)`
- `float16xm4_t vlsev_float16xm4 (const float16_t *address, long stride, unsigned int gvl)`
- `float16xm8_t vlsev_float16xm8 (const float16_t *address, long stride, unsigned int gvl)`
- `float32xm1_t vlsev_float32xm1 (const float *address, long stride, unsigned int gvl)`
- `float32xm2_t vlsev_float32xm2 (const float *address, long stride, unsigned int gvl)`
- `float32xm4_t vlsev_float32xm4 (const float *address, long stride, unsigned int gvl)`
- `float32xm8_t vlsev_float32xm8 (const float *address, long stride, unsigned int gvl)`
- `float64xm1_t vlsev_float64xm1 (const double *address, long stride, unsigned int gvl)`
- `float64xm2_t vlsev_float64xm2 (const double *address, long stride, unsigned int gvl)`
- `float64xm4_t vlsev_float64xm4 (const double *address, long stride, unsigned int gvl)`
- `float64xm8_t vlsev_float64xm8 (const double *address, long stride, unsigned int gvl)`
- `int16xm1_t vlsev_int16xm1 (const short *address, long stride, unsigned int gvl)`
- `int16xm2_t vlsev_int16xm2 (const short *address, long stride, unsigned int gvl)`
- `int16xm4_t vlsev_int16xm4 (const short *address, long stride, unsigned int gvl)`
- `int16xm8_t vlsev_int16xm8 (const short *address, long stride, unsigned int gvl)`
- `int32xm1_t vlsev_int32xm1 (const int *address, long stride, unsigned int gvl)`
- `int32xm2_t vlsev_int32xm2 (const int *address, long stride, unsigned int gvl)`
- `int32xm4_t vlsev_int32xm4 (const int *address, long stride, unsigned int gvl)`
- `int32xm8_t vlsev_int32xm8 (const int *address, long stride, unsigned int gvl)`
- `int64xm1_t vlsev_int64xm1 (const long *address, long stride, unsigned int gvl)`
- `int64xm2_t vlsev_int64xm2 (const long *address, long stride, unsigned int gvl)`
- `int64xm4_t vlsev_int64xm4 (const long *address, long stride, unsigned int gvl)`
- `int64xm8_t vlsev_int64xm8 (const long *address, long stride, unsigned int gvl)`

- `int8xm1_t vlsev_int8xm1` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8xm2_t vlsev_int8xm2` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8xm4_t vlsev_int8xm4` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8xm8_t vlsev_int8xm8` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `uint16xm1_t vlsev_uint16xm1` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint16xm2_t vlsev_uint16xm2` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint16xm4_t vlsev_uint16xm4` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint16xm8_t vlsev_uint16xm8` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32xm1_t vlsev_uint32xm1` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32xm2_t vlsev_uint32xm2` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32xm4_t vlsev_uint32xm4` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32xm8_t vlsev_uint32xm8` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64xm1_t vlsev_uint64xm1` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64xm2_t vlsev_uint64xm2` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64xm4_t vlsev_uint64xm4` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64xm8_t vlsev_uint64xm8` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8xm1_t vlsev_uint8xm1` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8xm2_t vlsev_uint8xm2` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8xm4_t vlsev_uint8xm4` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8xm8_t vlsev_uint8xm8` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = load_element(address)
    address = address + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vlsev_mask_float16xm1` (`float16xm1_t merge`, const float16\_t *\*address*, long *stride*, `e16xm1_t mask`, unsigned int *gvl*)
- `float16xm2_t vlsev_mask_float16xm2` (`float16xm2_t merge`, const float16\_t *\*address*, long *stride*, `e16xm2_t mask`, unsigned int *gvl*)
- `float16xm4_t vlsev_mask_float16xm4` (`float16xm4_t merge`, const float16\_t *\*address*, long *stride*, `e16xm4_t mask`, unsigned int *gvl*)
- `float16xm8_t vlsev_mask_float16xm8` (`float16xm8_t merge`, const float16\_t *\*address*, long *stride*, `e16xm8_t mask`, unsigned int *gvl*)
- `float32xm1_t vlsev_mask_float32xm1` (`float32xm1_t merge`, const float *\*address*, long *stride*, `e32xm1_t mask`, unsigned int *gvl*)
- `float32xm2_t vlsev_mask_float32xm2` (`float32xm2_t merge`, const float *\*address*, long *stride*, `e32xm2_t mask`, unsigned int *gvl*)
- `float32xm4_t vlsev_mask_float32xm4` (`float32xm4_t merge`, const float *\*address*, long *stride*, `e32xm4_t mask`, unsigned int *gvl*)

- *float32xm8\_t vlsev\_mask\_float32xm8* (*float32xm8\_t* merge, const float \**address*, long *stride*, *e32xm8\_t* mask, unsigned int *gvl*)
- *float64xm1\_t vlsev\_mask\_float64xm1* (*float64xm1\_t* merge, const double \**address*, long *stride*, *e64xm1\_t* mask, unsigned int *gvl*)
- *float64xm2\_t vlsev\_mask\_float64xm2* (*float64xm2\_t* merge, const double \**address*, long *stride*, *e64xm2\_t* mask, unsigned int *gvl*)
- *float64xm4\_t vlsev\_mask\_float64xm4* (*float64xm4\_t* merge, const double \**address*, long *stride*, *e64xm4\_t* mask, unsigned int *gvl*)
- *float64xm8\_t vlsev\_mask\_float64xm8* (*float64xm8\_t* merge, const double \**address*, long *stride*, *e64xm8\_t* mask, unsigned int *gvl*)
- *int16xm1\_t vlsev\_mask\_int16xm1* (*int16xm1\_t* merge, const short \**address*, long *stride*, *e16xm1\_t* mask, unsigned int *gvl*)
- *int16xm2\_t vlsev\_mask\_int16xm2* (*int16xm2\_t* merge, const short \**address*, long *stride*, *e16xm2\_t* mask, unsigned int *gvl*)
- *int16xm4\_t vlsev\_mask\_int16xm4* (*int16xm4\_t* merge, const short \**address*, long *stride*, *e16xm4\_t* mask, unsigned int *gvl*)
- *int16xm8\_t vlsev\_mask\_int16xm8* (*int16xm8\_t* merge, const short \**address*, long *stride*, *e16xm8\_t* mask, unsigned int *gvl*)
- *int32xm1\_t vlsev\_mask\_int32xm1* (*int32xm1\_t* merge, const int \**address*, long *stride*, *e32xm1\_t* mask, unsigned int *gvl*)
- *int32xm2\_t vlsev\_mask\_int32xm2* (*int32xm2\_t* merge, const int \**address*, long *stride*, *e32xm2\_t* mask, unsigned int *gvl*)
- *int32xm4\_t vlsev\_mask\_int32xm4* (*int32xm4\_t* merge, const int \**address*, long *stride*, *e32xm4\_t* mask, unsigned int *gvl*)
- *int32xm8\_t vlsev\_mask\_int32xm8* (*int32xm8\_t* merge, const int \**address*, long *stride*, *e32xm8\_t* mask, unsigned int *gvl*)
- *int64xm1\_t vlsev\_mask\_int64xm1* (*int64xm1\_t* merge, const long \**address*, long *stride*, *e64xm1\_t* mask, unsigned int *gvl*)
- *int64xm2\_t vlsev\_mask\_int64xm2* (*int64xm2\_t* merge, const long \**address*, long *stride*, *e64xm2\_t* mask, unsigned int *gvl*)
- *int64xm4\_t vlsev\_mask\_int64xm4* (*int64xm4\_t* merge, const long \**address*, long *stride*, *e64xm4\_t* mask, unsigned int *gvl*)
- *int64xm8\_t vlsev\_mask\_int64xm8* (*int64xm8\_t* merge, const long \**address*, long *stride*, *e64xm8\_t* mask, unsigned int *gvl*)
- *int8xm1\_t vlsev\_mask\_int8xm1* (*int8xm1\_t* merge, const signed char \**address*, long *stride*, *e8xm1\_t* mask, unsigned int *gvl*)
- *int8xm2\_t vlsev\_mask\_int8xm2* (*int8xm2\_t* merge, const signed char \**address*, long *stride*, *e8xm2\_t* mask, unsigned int *gvl*)
- *int8xm4\_t vlsev\_mask\_int8xm4* (*int8xm4\_t* merge, const signed char \**address*, long *stride*, *e8xm4\_t* mask, unsigned int *gvl*)
- *int8xm8\_t vlsev\_mask\_int8xm8* (*int8xm8\_t* merge, const signed char \**address*, long *stride*, *e8xm8\_t* mask, unsigned int *gvl*)
- *uint16xm1\_t vlsev\_mask\_uint16xm1* (*uint16xm1\_t* merge, const unsigned short \**address*, long *stride*, *e16xm1\_t* mask, unsigned int *gvl*)
- *uint16xm2\_t vlsev\_mask\_uint16xm2* (*uint16xm2\_t* merge, const unsigned short \**address*, long *stride*, *e16xm2\_t* mask, unsigned int *gvl*)

- `uint16xm4_t vlsev_mask_uint16xm4` (`uint16xm4_t merge`, const unsigned short `*address`, long `stride`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vlsev_mask_uint16xm8` (`uint16xm8_t merge`, const unsigned short `*address`, long `stride`, `e16xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vlsev_mask_uint32xm1` (`uint32xm1_t merge`, const unsigned int `*address`, long `stride`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vlsev_mask_uint32xm2` (`uint32xm2_t merge`, const unsigned int `*address`, long `stride`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vlsev_mask_uint32xm4` (`uint32xm4_t merge`, const unsigned int `*address`, long `stride`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint32xm8_t vlsev_mask_uint32xm8` (`uint32xm8_t merge`, const unsigned int `*address`, long `stride`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint64xm1_t vlsev_mask_uint64xm1` (`uint64xm1_t merge`, const unsigned long `*address`, long `stride`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vlsev_mask_uint64xm2` (`uint64xm2_t merge`, const unsigned long `*address`, long `stride`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vlsev_mask_uint64xm4` (`uint64xm4_t merge`, const unsigned long `*address`, long `stride`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vlsev_mask_uint64xm8` (`uint64xm8_t merge`, const unsigned long `*address`, long `stride`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vlsev_mask_uint8xm1` (`uint8xm1_t merge`, const unsigned char `*address`, long `stride`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vlsev_mask_uint8xm2` (`uint8xm2_t merge`, const unsigned char `*address`, long `stride`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vlsev_mask_uint8xm4` (`uint8xm4_t merge`, const unsigned char `*address`, long `stride`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vlsev_mask_uint8xm8` (`uint8xm8_t merge`, const unsigned char `*address`, long `stride`, `e8xm8_t mask`, unsigned int `gvl`)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = load_element(address)
      else
        result[element] = merge[element]
      address = address + stride
result[gvl : VLMAX] = 0
```

### 2.9.9 Load strided 16b signed in memory to vector

**Instruction:** [`'vlsh.v'`]

#### Prototypes:

- `int16xm1_t vlshv_int16xm1` (const short `*address`, long `stride`, unsigned int `gvl`)
- `int16xm2_t vlshv_int16xm2` (const short `*address`, long `stride`, unsigned int `gvl`)
- `int16xm4_t vlshv_int16xm4` (const short `*address`, long `stride`, unsigned int `gvl`)
- `int16xm8_t vlshv_int16xm8` (const short `*address`, long `stride`, unsigned int `gvl`)

- `int32xm1_t vlshv_int32xm1` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32xm2_t vlshv_int32xm2` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32xm4_t vlshv_int32xm4` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32xm8_t vlshv_int32xm8` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64xm1_t vlshv_int64xm1` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64xm2_t vlshv_int64xm2` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64xm4_t vlshv_int64xm4` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64xm8_t vlshv_int64xm8` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8xm1_t vlshv_int8xm1` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8xm2_t vlshv_int8xm2` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8xm4_t vlshv_int8xm4` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8xm8_t vlshv_int8xm8` (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = load_element(address)
      address = address + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vlshv_mask_int16xm1` (*int16xm1\_t merge*, const short *\*address*, long *stride*, *e16xm1\_t mask*, unsigned int *gvl*)
- `int16xm2_t vlshv_mask_int16xm2` (*int16xm2\_t merge*, const short *\*address*, long *stride*, *e16xm2\_t mask*, unsigned int *gvl*)
- `int16xm4_t vlshv_mask_int16xm4` (*int16xm4\_t merge*, const short *\*address*, long *stride*, *e16xm4\_t mask*, unsigned int *gvl*)
- `int16xm8_t vlshv_mask_int16xm8` (*int16xm8\_t merge*, const short *\*address*, long *stride*, *e16xm8\_t mask*, unsigned int *gvl*)
- `int32xm1_t vlshv_mask_int32xm1` (*int32xm1\_t merge*, const int *\*address*, long *stride*, *e32xm1\_t mask*, unsigned int *gvl*)
- `int32xm2_t vlshv_mask_int32xm2` (*int32xm2\_t merge*, const int *\*address*, long *stride*, *e32xm2\_t mask*, unsigned int *gvl*)
- `int32xm4_t vlshv_mask_int32xm4` (*int32xm4\_t merge*, const int *\*address*, long *stride*, *e32xm4\_t mask*, unsigned int *gvl*)
- `int32xm8_t vlshv_mask_int32xm8` (*int32xm8\_t merge*, const int *\*address*, long *stride*, *e32xm8\_t mask*, unsigned int *gvl*)
- `int64xm1_t vlshv_mask_int64xm1` (*int64xm1\_t merge*, const long *\*address*, long *stride*, *e64xm1\_t mask*, unsigned int *gvl*)
- `int64xm2_t vlshv_mask_int64xm2` (*int64xm2\_t merge*, const long *\*address*, long *stride*, *e64xm2\_t mask*, unsigned int *gvl*)
- `int64xm4_t vlshv_mask_int64xm4` (*int64xm4\_t merge*, const long *\*address*, long *stride*, *e64xm4\_t mask*, unsigned int *gvl*)
- `int64xm8_t vlshv_mask_int64xm8` (*int64xm8\_t merge*, const long *\*address*, long *stride*, *e64xm8\_t mask*, unsigned int *gvl*)



- `int8xm1_t vlshv_mask_int8xm1` (`int8xm1_t merge`, `const signed char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8xm2_t vlshv_mask_int8xm2` (`int8xm2_t merge`, `const signed char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)
- `int8xm4_t vlshv_mask_int8xm4` (`int8xm4_t merge`, `const signed char *address`, `long stride`, `e8xm4_t mask`, `unsigned int gvl`)
- `int8xm8_t vlshv_mask_int8xm8` (`int8xm8_t merge`, `const signed char *address`, `long stride`, `e8xm8_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = load_element(address)
      else
        result[element] = merge[element]
      address = address + stride
result[gvl : VLMAX] = 0
```

## 2.9.10 Load strided 16b unsigned in memory to vector

**Instruction:** [`'vlshu.v'`]

**Prototypes:**

- `uint16xm1_t vlshuv_uint16xm1` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint16xm2_t vlshuv_uint16xm2` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint16xm4_t vlshuv_uint16xm4` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint16xm8_t vlshuv_uint16xm8` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint32xm1_t vlshuv_uint32xm1` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint32xm2_t vlshuv_uint32xm2` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint32xm4_t vlshuv_uint32xm4` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint32xm8_t vlshuv_uint32xm8` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint64xm1_t vlshuv_uint64xm1` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint64xm2_t vlshuv_uint64xm2` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint64xm4_t vlshuv_uint64xm4` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint64xm8_t vlshuv_uint64xm8` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint8xm1_t vlshuv_uint8xm1` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)
- `uint8xm2_t vlshuv_uint8xm2` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)
- `uint8xm4_t vlshuv_uint8xm4` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)
- `uint8xm8_t vlshuv_uint8xm8` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)

**Operation:**



```
>>> for element = 0 to gvl - 1
      result[element] = load_element(address)
      address = address + stride
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *uint16xm1\_t* **vlshuv\_mask\_uint16xm1** (*uint16xm1\_t* merge, const unsigned short \*address, long stride, *e16xm1\_t* mask, unsigned int gvl)
- *uint16xm2\_t* **vlshuv\_mask\_uint16xm2** (*uint16xm2\_t* merge, const unsigned short \*address, long stride, *e16xm2\_t* mask, unsigned int gvl)
- *uint16xm4\_t* **vlshuv\_mask\_uint16xm4** (*uint16xm4\_t* merge, const unsigned short \*address, long stride, *e16xm4\_t* mask, unsigned int gvl)
- *uint16xm8\_t* **vlshuv\_mask\_uint16xm8** (*uint16xm8\_t* merge, const unsigned short \*address, long stride, *e16xm8\_t* mask, unsigned int gvl)
- *uint32xm1\_t* **vlshuv\_mask\_uint32xm1** (*uint32xm1\_t* merge, const unsigned int \*address, long stride, *e32xm1\_t* mask, unsigned int gvl)
- *uint32xm2\_t* **vlshuv\_mask\_uint32xm2** (*uint32xm2\_t* merge, const unsigned int \*address, long stride, *e32xm2\_t* mask, unsigned int gvl)
- *uint32xm4\_t* **vlshuv\_mask\_uint32xm4** (*uint32xm4\_t* merge, const unsigned int \*address, long stride, *e32xm4\_t* mask, unsigned int gvl)
- *uint32xm8\_t* **vlshuv\_mask\_uint32xm8** (*uint32xm8\_t* merge, const unsigned int \*address, long stride, *e32xm8\_t* mask, unsigned int gvl)
- *uint64xm1\_t* **vlshuv\_mask\_uint64xm1** (*uint64xm1\_t* merge, const unsigned long \*address, long stride, *e64xm1\_t* mask, unsigned int gvl)
- *uint64xm2\_t* **vlshuv\_mask\_uint64xm2** (*uint64xm2\_t* merge, const unsigned long \*address, long stride, *e64xm2\_t* mask, unsigned int gvl)
- *uint64xm4\_t* **vlshuv\_mask\_uint64xm4** (*uint64xm4\_t* merge, const unsigned long \*address, long stride, *e64xm4\_t* mask, unsigned int gvl)
- *uint64xm8\_t* **vlshuv\_mask\_uint64xm8** (*uint64xm8\_t* merge, const unsigned long \*address, long stride, *e64xm8\_t* mask, unsigned int gvl)
- *uint8xm1\_t* **vlshuv\_mask\_uint8xm1** (*uint8xm1\_t* merge, const unsigned char \*address, long stride, *e8xm1\_t* mask, unsigned int gvl)
- *uint8xm2\_t* **vlshuv\_mask\_uint8xm2** (*uint8xm2\_t* merge, const unsigned char \*address, long stride, *e8xm2\_t* mask, unsigned int gvl)
- *uint8xm4\_t* **vlshuv\_mask\_uint8xm4** (*uint8xm4\_t* merge, const unsigned char \*address, long stride, *e8xm4\_t* mask, unsigned int gvl)
- *uint8xm8\_t* **vlshuv\_mask\_uint8xm8** (*uint8xm8\_t* merge, const unsigned char \*address, long stride, *e8xm8\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = load_element(address)
      else
        result[element] = merge[element]
      address = address + stride
      result[gvl : VLMAX] = 0
```

## 2.9.11 Load strided 32b signed in memory to vector

**Instruction:** [`vlsww.v`']

**Prototypes:**

- `int16xm1_t vlsww_int16xm1` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16xm2_t vlsww_int16xm2` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16xm4_t vlsww_int16xm4` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16xm8_t vlsww_int16xm8` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int32xm1_t vlsww_int32xm1` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32xm2_t vlsww_int32xm2` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32xm4_t vlsww_int32xm4` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32xm8_t vlsww_int32xm8` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64xm1_t vlsww_int64xm1` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64xm2_t vlsww_int64xm2` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64xm4_t vlsww_int64xm4` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64xm8_t vlsww_int64xm8` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8xm1_t vlsww_int8xm1` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8xm2_t vlsww_int8xm2` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8xm4_t vlsww_int8xm4` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8xm8_t vlsww_int8xm8` (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = load_element(address)
      address = address + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vlsww_mask_int16xm1` (*int16xm1\_t merge*, const short *\*address*, long *stride*, *e16xm1\_t mask*, unsigned int *gvl*)
- `int16xm2_t vlsww_mask_int16xm2` (*int16xm2\_t merge*, const short *\*address*, long *stride*, *e16xm2\_t mask*, unsigned int *gvl*)
- `int16xm4_t vlsww_mask_int16xm4` (*int16xm4\_t merge*, const short *\*address*, long *stride*, *e16xm4\_t mask*, unsigned int *gvl*)
- `int16xm8_t vlsww_mask_int16xm8` (*int16xm8\_t merge*, const short *\*address*, long *stride*, *e16xm8\_t mask*, unsigned int *gvl*)
- `int32xm1_t vlsww_mask_int32xm1` (*int32xm1\_t merge*, const int *\*address*, long *stride*, *e32xm1\_t mask*, unsigned int *gvl*)
- `int32xm2_t vlsww_mask_int32xm2` (*int32xm2\_t merge*, const int *\*address*, long *stride*, *e32xm2\_t mask*, unsigned int *gvl*)
- `int32xm4_t vlsww_mask_int32xm4` (*int32xm4\_t merge*, const int *\*address*, long *stride*, *e32xm4\_t mask*, unsigned int *gvl*)

- `int32xm8_t vlsvw_mask_int32xm8(int32xm8_t merge, const int *address, long stride, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vlsvw_mask_int64xm1(int64xm1_t merge, const long *address, long stride, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vlsvw_mask_int64xm2(int64xm2_t merge, const long *address, long stride, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vlsvw_mask_int64xm4(int64xm4_t merge, const long *address, long stride, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vlsvw_mask_int64xm8(int64xm8_t merge, const long *address, long stride, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vlsvw_mask_int8xm1(int8xm1_t merge, const signed char *address, long stride, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vlsvw_mask_int8xm2(int8xm2_t merge, const signed char *address, long stride, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vlsvw_mask_int8xm4(int8xm4_t merge, const signed char *address, long stride, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vlsvw_mask_int8xm8(int8xm8_t merge, const signed char *address, long stride, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = load_element(address)
      else
        result[element] = merge[element]
      address = address + stride
    result[gvl : VLMAX] = 0
```

**2.9.12 Load strided 32b unsigned in memory to vector****Instruction:** [`vlsuw.v`']**Prototypes:**

- `uint16xm1_t vlsuwv_uint16xm1(const unsigned short *address, long stride, unsigned int gvl)`
- `uint16xm2_t vlsuwv_uint16xm2(const unsigned short *address, long stride, unsigned int gvl)`
- `uint16xm4_t vlsuwv_uint16xm4(const unsigned short *address, long stride, unsigned int gvl)`
- `uint16xm8_t vlsuwv_uint16xm8(const unsigned short *address, long stride, unsigned int gvl)`
- `uint32xm1_t vlsuwv_uint32xm1(const unsigned int *address, long stride, unsigned int gvl)`
- `uint32xm2_t vlsuwv_uint32xm2(const unsigned int *address, long stride, unsigned int gvl)`
- `uint32xm4_t vlsuwv_uint32xm4(const unsigned int *address, long stride, unsigned int gvl)`
- `uint32xm8_t vlsuwv_uint32xm8(const unsigned int *address, long stride, unsigned int gvl)`
- `uint64xm1_t vlsuwv_uint64xm1(const unsigned long *address, long stride, unsigned int gvl)`
- `uint64xm2_t vlsuwv_uint64xm2(const unsigned long *address, long stride, unsigned int gvl)`
- `uint64xm4_t vlsuwv_uint64xm4(const unsigned long *address, long stride, unsigned int gvl)`
- `uint64xm8_t vlsuwv_uint64xm8(const unsigned long *address, long stride, unsigned int gvl)`

- `uint8xm1_t vlsuv_uint8xm1` (const unsigned char \**address*, long *stride*, unsigned int *gvl*)
- `uint8xm2_t vlsuv_uint8xm2` (const unsigned char \**address*, long *stride*, unsigned int *gvl*)
- `uint8xm4_t vlsuv_uint8xm4` (const unsigned char \**address*, long *stride*, unsigned int *gvl*)
- `uint8xm8_t vlsuv_uint8xm8` (const unsigned char \**address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = load_element(address)
      address = address + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vlsuv_mask_uint16xm1` (`uint16xm1_t` *merge*, const unsigned short \**address*, long *stride*, `e16xm1_t` *mask*, unsigned int *gvl*)
- `uint16xm2_t vlsuv_mask_uint16xm2` (`uint16xm2_t` *merge*, const unsigned short \**address*, long *stride*, `e16xm2_t` *mask*, unsigned int *gvl*)
- `uint16xm4_t vlsuv_mask_uint16xm4` (`uint16xm4_t` *merge*, const unsigned short \**address*, long *stride*, `e16xm4_t` *mask*, unsigned int *gvl*)
- `uint16xm8_t vlsuv_mask_uint16xm8` (`uint16xm8_t` *merge*, const unsigned short \**address*, long *stride*, `e16xm8_t` *mask*, unsigned int *gvl*)
- `uint32xm1_t vlsuv_mask_uint32xm1` (`uint32xm1_t` *merge*, const unsigned int \**address*, long *stride*, `e32xm1_t` *mask*, unsigned int *gvl*)
- `uint32xm2_t vlsuv_mask_uint32xm2` (`uint32xm2_t` *merge*, const unsigned int \**address*, long *stride*, `e32xm2_t` *mask*, unsigned int *gvl*)
- `uint32xm4_t vlsuv_mask_uint32xm4` (`uint32xm4_t` *merge*, const unsigned int \**address*, long *stride*, `e32xm4_t` *mask*, unsigned int *gvl*)
- `uint32xm8_t vlsuv_mask_uint32xm8` (`uint32xm8_t` *merge*, const unsigned int \**address*, long *stride*, `e32xm8_t` *mask*, unsigned int *gvl*)
- `uint64xm1_t vlsuv_mask_uint64xm1` (`uint64xm1_t` *merge*, const unsigned long \**address*, long *stride*, `e64xm1_t` *mask*, unsigned int *gvl*)
- `uint64xm2_t vlsuv_mask_uint64xm2` (`uint64xm2_t` *merge*, const unsigned long \**address*, long *stride*, `e64xm2_t` *mask*, unsigned int *gvl*)
- `uint64xm4_t vlsuv_mask_uint64xm4` (`uint64xm4_t` *merge*, const unsigned long \**address*, long *stride*, `e64xm4_t` *mask*, unsigned int *gvl*)
- `uint64xm8_t vlsuv_mask_uint64xm8` (`uint64xm8_t` *merge*, const unsigned long \**address*, long *stride*, `e64xm8_t` *mask*, unsigned int *gvl*)
- `uint8xm1_t vlsuv_mask_uint8xm1` (`uint8xm1_t` *merge*, const unsigned char \**address*, long *stride*, `e8xm1_t` *mask*, unsigned int *gvl*)
- `uint8xm2_t vlsuv_mask_uint8xm2` (`uint8xm2_t` *merge*, const unsigned char \**address*, long *stride*, `e8xm2_t` *mask*, unsigned int *gvl*)
- `uint8xm4_t vlsuv_mask_uint8xm4` (`uint8xm4_t` *merge*, const unsigned char \**address*, long *stride*, `e8xm4_t` *mask*, unsigned int *gvl*)
- `uint8xm8_t vlsuv_mask_uint8xm8` (`uint8xm8_t` *merge*, const unsigned char \**address*, long *stride*, `e8xm8_t` *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = load_element(address)
    else
        result[element] = merge[element]
    address = address + stride
result[gvl : VLMAX] = 0
```

### 2.9.13 Load 32b signed in memory to vector

**Instruction:** [`vlw.v'`]

**Prototypes:**

- `int16xm1_t vlwv_int16xm1` (const short \**address*, unsigned int *gvl*)
- `int16xm2_t vlwv_int16xm2` (const short \**address*, unsigned int *gvl*)
- `int16xm4_t vlwv_int16xm4` (const short \**address*, unsigned int *gvl*)
- `int16xm8_t vlwv_int16xm8` (const short \**address*, unsigned int *gvl*)
- `int32xm1_t vlwv_int32xm1` (const int \**address*, unsigned int *gvl*)
- `int32xm2_t vlwv_int32xm2` (const int \**address*, unsigned int *gvl*)
- `int32xm4_t vlwv_int32xm4` (const int \**address*, unsigned int *gvl*)
- `int32xm8_t vlwv_int32xm8` (const int \**address*, unsigned int *gvl*)
- `int64xm1_t vlwv_int64xm1` (const long \**address*, unsigned int *gvl*)
- `int64xm2_t vlwv_int64xm2` (const long \**address*, unsigned int *gvl*)
- `int64xm4_t vlwv_int64xm4` (const long \**address*, unsigned int *gvl*)
- `int64xm8_t vlwv_int64xm8` (const long \**address*, unsigned int *gvl*)
- `int8xm1_t vlwv_int8xm1` (const signed char \**address*, unsigned int *gvl*)
- `int8xm2_t vlwv_int8xm2` (const signed char \**address*, unsigned int *gvl*)
- `int8xm4_t vlwv_int8xm4` (const signed char \**address*, unsigned int *gvl*)
- `int8xm8_t vlwv_int8xm8` (const signed char \**address*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = load_element(address)
    address = address + 1
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vlwv_mask_int16xm1` (`int16xm1_t` *merge*, const short \**address*, `e16xm1_t` *mask*, unsigned int *gvl*)
- `int16xm2_t vlwv_mask_int16xm2` (`int16xm2_t` *merge*, const short \**address*, `e16xm2_t` *mask*, unsigned int *gvl*)
- `int16xm4_t vlwv_mask_int16xm4` (`int16xm4_t` *merge*, const short \**address*, `e16xm4_t` *mask*, unsigned int *gvl*)

- `int16xm8_t vlwv_mask_int16xm8` (`int16xm8_t merge`, const short `*address`, `e16xm8_t mask`, unsigned int `gvl`)
- `int32xm1_t vlwv_mask_int32xm1` (`int32xm1_t merge`, const int `*address`, `e32xm1_t mask`, unsigned int `gvl`)
- `int32xm2_t vlwv_mask_int32xm2` (`int32xm2_t merge`, const int `*address`, `e32xm2_t mask`, unsigned int `gvl`)
- `int32xm4_t vlwv_mask_int32xm4` (`int32xm4_t merge`, const int `*address`, `e32xm4_t mask`, unsigned int `gvl`)
- `int32xm8_t vlwv_mask_int32xm8` (`int32xm8_t merge`, const int `*address`, `e32xm8_t mask`, unsigned int `gvl`)
- `int64xm1_t vlwv_mask_int64xm1` (`int64xm1_t merge`, const long `*address`, `e64xm1_t mask`, unsigned int `gvl`)
- `int64xm2_t vlwv_mask_int64xm2` (`int64xm2_t merge`, const long `*address`, `e64xm2_t mask`, unsigned int `gvl`)
- `int64xm4_t vlwv_mask_int64xm4` (`int64xm4_t merge`, const long `*address`, `e64xm4_t mask`, unsigned int `gvl`)
- `int64xm8_t vlwv_mask_int64xm8` (`int64xm8_t merge`, const long `*address`, `e64xm8_t mask`, unsigned int `gvl`)
- `int8xm1_t vlwv_mask_int8xm1` (`int8xm1_t merge`, const signed char `*address`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8xm2_t vlwv_mask_int8xm2` (`int8xm2_t merge`, const signed char `*address`, `e8xm2_t mask`, unsigned int `gvl`)
- `int8xm4_t vlwv_mask_int8xm4` (`int8xm4_t merge`, const signed char `*address`, `e8xm4_t mask`, unsigned int `gvl`)
- `int8xm8_t vlwv_mask_int8xm8` (`int8xm8_t merge`, const signed char `*address`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = load_element(address)
        address = address + 1
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.9.14 Load 32b unsigned in memory to vector

Instruction: [`'vlwu.v'`]

Prototypes:

- `uint16xm1_t vlwuv_uint16xm1` (const unsigned short `*address`, unsigned int `gvl`)
- `uint16xm2_t vlwuv_uint16xm2` (const unsigned short `*address`, unsigned int `gvl`)
- `uint16xm4_t vlwuv_uint16xm4` (const unsigned short `*address`, unsigned int `gvl`)
- `uint16xm8_t vlwuv_uint16xm8` (const unsigned short `*address`, unsigned int `gvl`)
- `uint32xm1_t vlwuv_uint32xm1` (const unsigned int `*address`, unsigned int `gvl`)
- `uint32xm2_t vlwuv_uint32xm2` (const unsigned int `*address`, unsigned int `gvl`)



- `uint32xm4_t vlwuv_uint32xm4` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint32xm8_t vlwuv_uint32xm8` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64xm1_t vlwuv_uint64xm1` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint64xm2_t vlwuv_uint64xm2` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint64xm4_t vlwuv_uint64xm4` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint64xm8_t vlwuv_uint64xm8` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8xm1_t vlwuv_uint8xm1` (const unsigned char *\*address*, unsigned int *gvl*)
- `uint8xm2_t vlwuv_uint8xm2` (const unsigned char *\*address*, unsigned int *gvl*)
- `uint8xm4_t vlwuv_uint8xm4` (const unsigned char *\*address*, unsigned int *gvl*)
- `uint8xm8_t vlwuv_uint8xm8` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = load_element(address)
      address = address + 1
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vlwuv_mask_uint16xm1` (*uint16xm1\_t merge*, const unsigned short *\*address*, *e16xm1\_t mask*, unsigned int *gvl*)
- `uint16xm2_t vlwuv_mask_uint16xm2` (*uint16xm2\_t merge*, const unsigned short *\*address*, *e16xm2\_t mask*, unsigned int *gvl*)
- `uint16xm4_t vlwuv_mask_uint16xm4` (*uint16xm4\_t merge*, const unsigned short *\*address*, *e16xm4\_t mask*, unsigned int *gvl*)
- `uint16xm8_t vlwuv_mask_uint16xm8` (*uint16xm8\_t merge*, const unsigned short *\*address*, *e16xm8\_t mask*, unsigned int *gvl*)
- `uint32xm1_t vlwuv_mask_uint32xm1` (*uint32xm1\_t merge*, const unsigned int *\*address*, *e32xm1\_t mask*, unsigned int *gvl*)
- `uint32xm2_t vlwuv_mask_uint32xm2` (*uint32xm2\_t merge*, const unsigned int *\*address*, *e32xm2\_t mask*, unsigned int *gvl*)
- `uint32xm4_t vlwuv_mask_uint32xm4` (*uint32xm4\_t merge*, const unsigned int *\*address*, *e32xm4\_t mask*, unsigned int *gvl*)
- `uint32xm8_t vlwuv_mask_uint32xm8` (*uint32xm8\_t merge*, const unsigned int *\*address*, *e32xm8\_t mask*, unsigned int *gvl*)
- `uint64xm1_t vlwuv_mask_uint64xm1` (*uint64xm1\_t merge*, const unsigned long *\*address*, *e64xm1\_t mask*, unsigned int *gvl*)
- `uint64xm2_t vlwuv_mask_uint64xm2` (*uint64xm2\_t merge*, const unsigned long *\*address*, *e64xm2\_t mask*, unsigned int *gvl*)
- `uint64xm4_t vlwuv_mask_uint64xm4` (*uint64xm4\_t merge*, const unsigned long *\*address*, *e64xm4\_t mask*, unsigned int *gvl*)
- `uint64xm8_t vlwuv_mask_uint64xm8` (*uint64xm8\_t merge*, const unsigned long *\*address*, *e64xm8\_t mask*, unsigned int *gvl*)
- `uint8xm1_t vlwuv_mask_uint8xm1` (*uint8xm1\_t merge*, const unsigned char *\*address*, *e8xm1\_t mask*, unsigned int *gvl*)

- `uint8xm2_t vluuv_mask_uint8xm2 (uint8xm2_t merge, const unsigned char *address, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vluuv_mask_uint8xm4 (uint8xm4_t merge, const unsigned char *address, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vluuv_mask_uint8xm8 (uint8xm8_t merge, const unsigned char *address, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        result[element] = load_element(address)
        address = address + 1
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

**2.9.15 Load indexed 8b signed in memory to vector****Instruction:** [`'vlxb.v'`]**Prototypes:**

- `int16xm1_t vlxbv_int16xm1 (const short *address, int16xm1_t index, unsigned int gvl)`
- `int16xm2_t vlxbv_int16xm2 (const short *address, int16xm2_t index, unsigned int gvl)`
- `int16xm4_t vlxbv_int16xm4 (const short *address, int16xm4_t index, unsigned int gvl)`
- `int16xm8_t vlxbv_int16xm8 (const short *address, int16xm8_t index, unsigned int gvl)`
- `int32xm1_t vlxbv_int32xm1 (const int *address, int32xm1_t index, unsigned int gvl)`
- `int32xm2_t vlxbv_int32xm2 (const int *address, int32xm2_t index, unsigned int gvl)`
- `int32xm4_t vlxbv_int32xm4 (const int *address, int32xm4_t index, unsigned int gvl)`
- `int32xm8_t vlxbv_int32xm8 (const int *address, int32xm8_t index, unsigned int gvl)`
- `int64xm1_t vlxbv_int64xm1 (const long *address, int64xm1_t index, unsigned int gvl)`
- `int64xm2_t vlxbv_int64xm2 (const long *address, int64xm2_t index, unsigned int gvl)`
- `int64xm4_t vlxbv_int64xm4 (const long *address, int64xm4_t index, unsigned int gvl)`
- `int64xm8_t vlxbv_int64xm8 (const long *address, int64xm8_t index, unsigned int gvl)`
- `int8xm1_t vlxbv_int8xm1 (const signed char *address, int8xm1_t index, unsigned int gvl)`
- `int8xm2_t vlxbv_int8xm2 (const signed char *address, int8xm2_t index, unsigned int gvl)`
- `int8xm4_t vlxbv_int8xm4 (const signed char *address, int8xm4_t index, unsigned int gvl)`
- `int8xm8_t vlxbv_int8xm8 (const signed char *address, int8xm8_t index, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = load_element(address + index[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16xm1_t vlxbv_mask_int16xm1(int16xm1_t merge, const short *address, int16xm1_t index, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vlxbv_mask_int16xm2(int16xm2_t merge, const short *address, int16xm2_t index, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vlxbv_mask_int16xm4(int16xm4_t merge, const short *address, int16xm4_t index, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vlxbv_mask_int16xm8(int16xm8_t merge, const short *address, int16xm8_t index, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vlxbv_mask_int32xm1(int32xm1_t merge, const int *address, int32xm1_t index, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vlxbv_mask_int32xm2(int32xm2_t merge, const int *address, int32xm2_t index, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vlxbv_mask_int32xm4(int32xm4_t merge, const int *address, int32xm4_t index, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vlxbv_mask_int32xm8(int32xm8_t merge, const int *address, int32xm8_t index, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vlxbv_mask_int64xm1(int64xm1_t merge, const long *address, int64xm1_t index, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vlxbv_mask_int64xm2(int64xm2_t merge, const long *address, int64xm2_t index, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vlxbv_mask_int64xm4(int64xm4_t merge, const long *address, int64xm4_t index, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vlxbv_mask_int64xm8(int64xm8_t merge, const long *address, int64xm8_t index, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vlxbv_mask_int8xm1(int8xm1_t merge, const signed char *address, int8xm1_t index, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vlxbv_mask_int8xm2(int8xm2_t merge, const signed char *address, int8xm2_t index, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vlxbv_mask_int8xm4(int8xm4_t merge, const signed char *address, int8xm4_t index, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vlxbv_mask_int8xm8(int8xm8_t merge, const signed char *address, int8xm8_t index, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
    result[gvl : VLMAX] = 0
    for element = 0 to gvl - 1
        if mask[element] then
            result[element] = load_element(address + index[element])
        else
            result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

### 2.9.16 Load indexed 8b unsigned in memory to vector

**Instruction:** [`vlxbu.v`']

**Prototypes:**

- `uint16xm1_t vlxbuf_uint16xm1` (const unsigned short \*address, `uint16xm1_t` index, unsigned int gvl)
- `uint16xm2_t vlxbuf_uint16xm2` (const unsigned short \*address, `uint16xm2_t` index, unsigned int gvl)
- `uint16xm4_t vlxbuf_uint16xm4` (const unsigned short \*address, `uint16xm4_t` index, unsigned int gvl)
- `uint16xm8_t vlxbuf_uint16xm8` (const unsigned short \*address, `uint16xm8_t` index, unsigned int gvl)
- `uint32xm1_t vlxbuf_uint32xm1` (const unsigned int \*address, `uint32xm1_t` index, unsigned int gvl)
- `uint32xm2_t vlxbuf_uint32xm2` (const unsigned int \*address, `uint32xm2_t` index, unsigned int gvl)
- `uint32xm4_t vlxbuf_uint32xm4` (const unsigned int \*address, `uint32xm4_t` index, unsigned int gvl)
- `uint32xm8_t vlxbuf_uint32xm8` (const unsigned int \*address, `uint32xm8_t` index, unsigned int gvl)
- `uint64xm1_t vlxbuf_uint64xm1` (const unsigned long \*address, `uint64xm1_t` index, unsigned int gvl)
- `uint64xm2_t vlxbuf_uint64xm2` (const unsigned long \*address, `uint64xm2_t` index, unsigned int gvl)
- `uint64xm4_t vlxbuf_uint64xm4` (const unsigned long \*address, `uint64xm4_t` index, unsigned int gvl)
- `uint64xm8_t vlxbuf_uint64xm8` (const unsigned long \*address, `uint64xm8_t` index, unsigned int gvl)
- `uint8xm1_t vlxbuf_uint8xm1` (const unsigned char \*address, `uint8xm1_t` index, unsigned int gvl)
- `uint8xm2_t vlxbuf_uint8xm2` (const unsigned char \*address, `uint8xm2_t` index, unsigned int gvl)
- `uint8xm4_t vlxbuf_uint8xm4` (const unsigned char \*address, `uint8xm4_t` index, unsigned int gvl)
- `uint8xm8_t vlxbuf_uint8xm8` (const unsigned char \*address, `uint8xm8_t` index, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = load_element(address + index[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vlxbuf_mask_uint16xm1` (`uint16xm1_t` merge, const unsigned short \*address, `uint16xm1_t` index, `e16xm1_t` mask, unsigned int gvl)
- `uint16xm2_t vlxbuf_mask_uint16xm2` (`uint16xm2_t` merge, const unsigned short \*address, `uint16xm2_t` index, `e16xm2_t` mask, unsigned int gvl)
- `uint16xm4_t vlxbuf_mask_uint16xm4` (`uint16xm4_t` merge, const unsigned short \*address, `uint16xm4_t` index, `e16xm4_t` mask, unsigned int gvl)
- `uint16xm8_t vlxbuf_mask_uint16xm8` (`uint16xm8_t` merge, const unsigned short \*address, `uint16xm8_t` index, `e16xm8_t` mask, unsigned int gvl)
- `uint32xm1_t vlxbuf_mask_uint32xm1` (`uint32xm1_t` merge, const unsigned int \*address, `uint32xm1_t` index, `e32xm1_t` mask, unsigned int gvl)
- `uint32xm2_t vlxbuf_mask_uint32xm2` (`uint32xm2_t` merge, const unsigned int \*address, `uint32xm2_t` index, `e32xm2_t` mask, unsigned int gvl)
- `uint32xm4_t vlxbuf_mask_uint32xm4` (`uint32xm4_t` merge, const unsigned int \*address, `uint32xm4_t` index, `e32xm4_t` mask, unsigned int gvl)
- `uint32xm8_t vlxbuf_mask_uint32xm8` (`uint32xm8_t` merge, const unsigned int \*address, `uint32xm8_t` index, `e32xm8_t` mask, unsigned int gvl)

- `uint64xm1_t vlxbuv_mask_uint64xm1` (`uint64xm1_t merge`, `const unsigned long *address`, `uint64xm1_t index`, `e64xm1_t mask`, `unsigned int gvl`)
- `uint64xm2_t vlxbuv_mask_uint64xm2` (`uint64xm2_t merge`, `const unsigned long *address`, `uint64xm2_t index`, `e64xm2_t mask`, `unsigned int gvl`)
- `uint64xm4_t vlxbuv_mask_uint64xm4` (`uint64xm4_t merge`, `const unsigned long *address`, `uint64xm4_t index`, `e64xm4_t mask`, `unsigned int gvl`)
- `uint64xm8_t vlxbuv_mask_uint64xm8` (`uint64xm8_t merge`, `const unsigned long *address`, `uint64xm8_t index`, `e64xm8_t mask`, `unsigned int gvl`)
- `uint8xm1_t vlxbuv_mask_uint8xm1` (`uint8xm1_t merge`, `const unsigned char *address`, `uint8xm1_t index`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8xm2_t vlxbuv_mask_uint8xm2` (`uint8xm2_t merge`, `const unsigned char *address`, `uint8xm2_t index`, `e8xm2_t mask`, `unsigned int gvl`)
- `uint8xm4_t vlxbuv_mask_uint8xm4` (`uint8xm4_t merge`, `const unsigned char *address`, `uint8xm4_t index`, `e8xm4_t mask`, `unsigned int gvl`)
- `uint8xm8_t vlxbuv_mask_uint8xm8` (`uint8xm8_t merge`, `const unsigned char *address`, `uint8xm8_t index`, `e8xm8_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    result[gvl : VLMAX] = 0
    for element = 0 to gvl - 1
        if mask[element] then
            result[element] = load_element(address + index[element])
        else
            result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.9.17 Load indexed element in memory to vector****Instruction:** [`'vlxe.v'`]**Prototypes:**

- `float16xm1_t vlxe_v_float16xm1` (`const float16_t *address`, `int16xm1_t index`, `unsigned int gvl`)
- `float16xm2_t vlxe_v_float16xm2` (`const float16_t *address`, `int16xm2_t index`, `unsigned int gvl`)
- `float16xm4_t vlxe_v_float16xm4` (`const float16_t *address`, `int16xm4_t index`, `unsigned int gvl`)
- `float16xm8_t vlxe_v_float16xm8` (`const float16_t *address`, `int16xm8_t index`, `unsigned int gvl`)
- `float32xm1_t vlxe_v_float32xm1` (`const float *address`, `int32xm1_t index`, `unsigned int gvl`)
- `float32xm2_t vlxe_v_float32xm2` (`const float *address`, `int32xm2_t index`, `unsigned int gvl`)
- `float32xm4_t vlxe_v_float32xm4` (`const float *address`, `int32xm4_t index`, `unsigned int gvl`)
- `float32xm8_t vlxe_v_float32xm8` (`const float *address`, `int32xm8_t index`, `unsigned int gvl`)
- `float64xm1_t vlxe_v_float64xm1` (`const double *address`, `int64xm1_t index`, `unsigned int gvl`)
- `float64xm2_t vlxe_v_float64xm2` (`const double *address`, `int64xm2_t index`, `unsigned int gvl`)
- `float64xm4_t vlxe_v_float64xm4` (`const double *address`, `int64xm4_t index`, `unsigned int gvl`)
- `float64xm8_t vlxe_v_float64xm8` (`const double *address`, `int64xm8_t index`, `unsigned int gvl`)
- `int16xm1_t vlxe_v_int16xm1` (`const short *address`, `int16xm1_t index`, `unsigned int gvl`)



- `int16xm2_t vlxeve_int16xm2` (const short *\*address*, *int16xm2\_t* index, unsigned int *gvl*)
- `int16xm4_t vlxeve_int16xm4` (const short *\*address*, *int16xm4\_t* index, unsigned int *gvl*)
- `int16xm8_t vlxeve_int16xm8` (const short *\*address*, *int16xm8\_t* index, unsigned int *gvl*)
- `int32xm1_t vlxeve_int32xm1` (const int *\*address*, *int32xm1\_t* index, unsigned int *gvl*)
- `int32xm2_t vlxeve_int32xm2` (const int *\*address*, *int32xm2\_t* index, unsigned int *gvl*)
- `int32xm4_t vlxeve_int32xm4` (const int *\*address*, *int32xm4\_t* index, unsigned int *gvl*)
- `int32xm8_t vlxeve_int32xm8` (const int *\*address*, *int32xm8\_t* index, unsigned int *gvl*)
- `int64xm1_t vlxeve_int64xm1` (const long *\*address*, *int64xm1\_t* index, unsigned int *gvl*)
- `int64xm2_t vlxeve_int64xm2` (const long *\*address*, *int64xm2\_t* index, unsigned int *gvl*)
- `int64xm4_t vlxeve_int64xm4` (const long *\*address*, *int64xm4\_t* index, unsigned int *gvl*)
- `int64xm8_t vlxeve_int64xm8` (const long *\*address*, *int64xm8\_t* index, unsigned int *gvl*)
- `int8xm1_t vlxeve_int8xm1` (const signed char *\*address*, *int8xm1\_t* index, unsigned int *gvl*)
- `int8xm2_t vlxeve_int8xm2` (const signed char *\*address*, *int8xm2\_t* index, unsigned int *gvl*)
- `int8xm4_t vlxeve_int8xm4` (const signed char *\*address*, *int8xm4\_t* index, unsigned int *gvl*)
- `int8xm8_t vlxeve_int8xm8` (const signed char *\*address*, *int8xm8\_t* index, unsigned int *gvl*)
- `uint16xm1_t vlxeve_uint16xm1` (const unsigned short *\*address*, *uint16xm1\_t* index, unsigned int *gvl*)
- `uint16xm2_t vlxeve_uint16xm2` (const unsigned short *\*address*, *uint16xm2\_t* index, unsigned int *gvl*)
- `uint16xm4_t vlxeve_uint16xm4` (const unsigned short *\*address*, *uint16xm4\_t* index, unsigned int *gvl*)
- `uint16xm8_t vlxeve_uint16xm8` (const unsigned short *\*address*, *uint16xm8\_t* index, unsigned int *gvl*)
- `uint32xm1_t vlxeve_uint32xm1` (const unsigned int *\*address*, *uint32xm1\_t* index, unsigned int *gvl*)
- `uint32xm2_t vlxeve_uint32xm2` (const unsigned int *\*address*, *uint32xm2\_t* index, unsigned int *gvl*)
- `uint32xm4_t vlxeve_uint32xm4` (const unsigned int *\*address*, *uint32xm4\_t* index, unsigned int *gvl*)
- `uint32xm8_t vlxeve_uint32xm8` (const unsigned int *\*address*, *uint32xm8\_t* index, unsigned int *gvl*)
- `uint64xm1_t vlxeve_uint64xm1` (const unsigned long *\*address*, *uint64xm1\_t* index, unsigned int *gvl*)
- `uint64xm2_t vlxeve_uint64xm2` (const unsigned long *\*address*, *uint64xm2\_t* index, unsigned int *gvl*)
- `uint64xm4_t vlxeve_uint64xm4` (const unsigned long *\*address*, *uint64xm4\_t* index, unsigned int *gvl*)
- `uint64xm8_t vlxeve_uint64xm8` (const unsigned long *\*address*, *uint64xm8\_t* index, unsigned int *gvl*)
- `uint8xm1_t vlxeve_uint8xm1` (const unsigned char *\*address*, *uint8xm1\_t* index, unsigned int *gvl*)
- `uint8xm2_t vlxeve_uint8xm2` (const unsigned char *\*address*, *uint8xm2\_t* index, unsigned int *gvl*)
- `uint8xm4_t vlxeve_uint8xm4` (const unsigned char *\*address*, *uint8xm4\_t* index, unsigned int *gvl*)
- `uint8xm8_t vlxeve_uint8xm8` (const unsigned char *\*address*, *uint8xm8\_t* index, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = load_element(address + index[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**



- *float16xm1\_t vlxev\_mask\_float16xm1* (*float16xm1\_t* merge, const *float16\_t* \**address*, *int16xm1\_t* index, *e16xm1\_t* mask, unsigned int gvl)
- *float16xm2\_t vlxev\_mask\_float16xm2* (*float16xm2\_t* merge, const *float16\_t* \**address*, *int16xm2\_t* index, *e16xm2\_t* mask, unsigned int gvl)
- *float16xm4\_t vlxev\_mask\_float16xm4* (*float16xm4\_t* merge, const *float16\_t* \**address*, *int16xm4\_t* index, *e16xm4\_t* mask, unsigned int gvl)
- *float16xm8\_t vlxev\_mask\_float16xm8* (*float16xm8\_t* merge, const *float16\_t* \**address*, *int16xm8\_t* index, *e16xm8\_t* mask, unsigned int gvl)
- *float32xm1\_t vlxev\_mask\_float32xm1* (*float32xm1\_t* merge, const *float* \**address*, *int32xm1\_t* index, *e32xm1\_t* mask, unsigned int gvl)
- *float32xm2\_t vlxev\_mask\_float32xm2* (*float32xm2\_t* merge, const *float* \**address*, *int32xm2\_t* index, *e32xm2\_t* mask, unsigned int gvl)
- *float32xm4\_t vlxev\_mask\_float32xm4* (*float32xm4\_t* merge, const *float* \**address*, *int32xm4\_t* index, *e32xm4\_t* mask, unsigned int gvl)
- *float32xm8\_t vlxev\_mask\_float32xm8* (*float32xm8\_t* merge, const *float* \**address*, *int32xm8\_t* index, *e32xm8\_t* mask, unsigned int gvl)
- *float64xm1\_t vlxev\_mask\_float64xm1* (*float64xm1\_t* merge, const *double* \**address*, *int64xm1\_t* index, *e64xm1\_t* mask, unsigned int gvl)
- *float64xm2\_t vlxev\_mask\_float64xm2* (*float64xm2\_t* merge, const *double* \**address*, *int64xm2\_t* index, *e64xm2\_t* mask, unsigned int gvl)
- *float64xm4\_t vlxev\_mask\_float64xm4* (*float64xm4\_t* merge, const *double* \**address*, *int64xm4\_t* index, *e64xm4\_t* mask, unsigned int gvl)
- *float64xm8\_t vlxev\_mask\_float64xm8* (*float64xm8\_t* merge, const *double* \**address*, *int64xm8\_t* index, *e64xm8\_t* mask, unsigned int gvl)
- *int16xm1\_t vlxev\_mask\_int16xm1* (*int16xm1\_t* merge, const *short* \**address*, *int16xm1\_t* index, *e16xm1\_t* mask, unsigned int gvl)
- *int16xm2\_t vlxev\_mask\_int16xm2* (*int16xm2\_t* merge, const *short* \**address*, *int16xm2\_t* index, *e16xm2\_t* mask, unsigned int gvl)
- *int16xm4\_t vlxev\_mask\_int16xm4* (*int16xm4\_t* merge, const *short* \**address*, *int16xm4\_t* index, *e16xm4\_t* mask, unsigned int gvl)
- *int16xm8\_t vlxev\_mask\_int16xm8* (*int16xm8\_t* merge, const *short* \**address*, *int16xm8\_t* index, *e16xm8\_t* mask, unsigned int gvl)
- *int32xm1\_t vlxev\_mask\_int32xm1* (*int32xm1\_t* merge, const *int* \**address*, *int32xm1\_t* index, *e32xm1\_t* mask, unsigned int gvl)
- *int32xm2\_t vlxev\_mask\_int32xm2* (*int32xm2\_t* merge, const *int* \**address*, *int32xm2\_t* index, *e32xm2\_t* mask, unsigned int gvl)
- *int32xm4\_t vlxev\_mask\_int32xm4* (*int32xm4\_t* merge, const *int* \**address*, *int32xm4\_t* index, *e32xm4\_t* mask, unsigned int gvl)
- *int32xm8\_t vlxev\_mask\_int32xm8* (*int32xm8\_t* merge, const *int* \**address*, *int32xm8\_t* index, *e32xm8\_t* mask, unsigned int gvl)
- *int64xm1\_t vlxev\_mask\_int64xm1* (*int64xm1\_t* merge, const *long* \**address*, *int64xm1\_t* index, *e64xm1\_t* mask, unsigned int gvl)
- *int64xm2\_t vlxev\_mask\_int64xm2* (*int64xm2\_t* merge, const *long* \**address*, *int64xm2\_t* index, *e64xm2\_t* mask, unsigned int gvl)
- *int64xm4\_t vlxev\_mask\_int64xm4* (*int64xm4\_t* merge, const *long* \**address*, *int64xm4\_t* index, *e64xm4\_t* mask, unsigned int gvl)

- `int64xm8_t vlxeve_mask_int64xm8(int64xm8_t merge, const long *address, int64xm8_t index, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vlxeve_mask_int8xm1(int8xm1_t merge, const signed char *address, int8xm1_t index, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vlxeve_mask_int8xm2(int8xm2_t merge, const signed char *address, int8xm2_t index, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vlxeve_mask_int8xm4(int8xm4_t merge, const signed char *address, int8xm4_t index, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vlxeve_mask_int8xm8(int8xm8_t merge, const signed char *address, int8xm8_t index, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vlxeve_mask_uint16xm1(uint16xm1_t merge, const unsigned short *address, uint16xm1_t index, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vlxeve_mask_uint16xm2(uint16xm2_t merge, const unsigned short *address, uint16xm2_t index, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vlxeve_mask_uint16xm4(uint16xm4_t merge, const unsigned short *address, uint16xm4_t index, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vlxeve_mask_uint16xm8(uint16xm8_t merge, const unsigned short *address, uint16xm8_t index, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vlxeve_mask_uint32xm1(uint32xm1_t merge, const unsigned int *address, uint32xm1_t index, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vlxeve_mask_uint32xm2(uint32xm2_t merge, const unsigned int *address, uint32xm2_t index, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vlxeve_mask_uint32xm4(uint32xm4_t merge, const unsigned int *address, uint32xm4_t index, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vlxeve_mask_uint32xm8(uint32xm8_t merge, const unsigned int *address, uint32xm8_t index, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vlxeve_mask_uint64xm1(uint64xm1_t merge, const unsigned long *address, uint64xm1_t index, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vlxeve_mask_uint64xm2(uint64xm2_t merge, const unsigned long *address, uint64xm2_t index, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vlxeve_mask_uint64xm4(uint64xm4_t merge, const unsigned long *address, uint64xm4_t index, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vlxeve_mask_uint64xm8(uint64xm8_t merge, const unsigned long *address, uint64xm8_t index, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vlxeve_mask_uint8xm1(uint8xm1_t merge, const unsigned char *address, uint8xm1_t index, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vlxeve_mask_uint8xm2(uint8xm2_t merge, const unsigned char *address, uint8xm2_t index, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vlxeve_mask_uint8xm4(uint8xm4_t merge, const unsigned char *address, uint8xm4_t index, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vlxeve_mask_uint8xm8(uint8xm8_t merge, const unsigned char *address, uint8xm8_t index, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      result[gvl : VLMAX] = 0
      for element = 0 to gvl - 1
```

(continues on next page)

(continued from previous page)

```

    if mask[element] then
        result[element] = load_element(address + index[element])
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0

```

## 2.9.18 Load indexed 16b signed in memory to vector

**Instruction:** [`vlxh.v`']

**Prototypes:**

- `int16xm1_t vlxhv_int16xm1` (const short *\*address*, `int16xm1_t` *index*, unsigned int *gvl*)
- `int16xm2_t vlxhv_int16xm2` (const short *\*address*, `int16xm2_t` *index*, unsigned int *gvl*)
- `int16xm4_t vlxhv_int16xm4` (const short *\*address*, `int16xm4_t` *index*, unsigned int *gvl*)
- `int16xm8_t vlxhv_int16xm8` (const short *\*address*, `int16xm8_t` *index*, unsigned int *gvl*)
- `int32xm1_t vlxhv_int32xm1` (const int *\*address*, `int32xm1_t` *index*, unsigned int *gvl*)
- `int32xm2_t vlxhv_int32xm2` (const int *\*address*, `int32xm2_t` *index*, unsigned int *gvl*)
- `int32xm4_t vlxhv_int32xm4` (const int *\*address*, `int32xm4_t` *index*, unsigned int *gvl*)
- `int32xm8_t vlxhv_int32xm8` (const int *\*address*, `int32xm8_t` *index*, unsigned int *gvl*)
- `int64xm1_t vlxhv_int64xm1` (const long *\*address*, `int64xm1_t` *index*, unsigned int *gvl*)
- `int64xm2_t vlxhv_int64xm2` (const long *\*address*, `int64xm2_t` *index*, unsigned int *gvl*)
- `int64xm4_t vlxhv_int64xm4` (const long *\*address*, `int64xm4_t` *index*, unsigned int *gvl*)
- `int64xm8_t vlxhv_int64xm8` (const long *\*address*, `int64xm8_t` *index*, unsigned int *gvl*)
- `int8xm1_t vlxhv_int8xm1` (const signed char *\*address*, `int8xm1_t` *index*, unsigned int *gvl*)
- `int8xm2_t vlxhv_int8xm2` (const signed char *\*address*, `int8xm2_t` *index*, unsigned int *gvl*)
- `int8xm4_t vlxhv_int8xm4` (const signed char *\*address*, `int8xm4_t` *index*, unsigned int *gvl*)
- `int8xm8_t vlxhv_int8xm8` (const signed char *\*address*, `int8xm8_t` *index*, unsigned int *gvl*)

**Operation:**

```

>>> for element = 0 to gvl - 1
    result[element] = load_element(address + index[element])
result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- `int16xm1_t vlxhv_mask_int16xm1` (`int16xm1_t` *merge*, const short *\*address*, `int16xm1_t` *index*, `e16xm1_t` *mask*, unsigned int *gvl*)
- `int16xm2_t vlxhv_mask_int16xm2` (`int16xm2_t` *merge*, const short *\*address*, `int16xm2_t` *index*, `e16xm2_t` *mask*, unsigned int *gvl*)
- `int16xm4_t vlxhv_mask_int16xm4` (`int16xm4_t` *merge*, const short *\*address*, `int16xm4_t` *index*, `e16xm4_t` *mask*, unsigned int *gvl*)
- `int16xm8_t vlxhv_mask_int16xm8` (`int16xm8_t` *merge*, const short *\*address*, `int16xm8_t` *index*, `e16xm8_t` *mask*, unsigned int *gvl*)

- `int32xm1_t vlxhv_mask_int32xm1` (`int32xm1_t merge`, `const int *address`, `int32xm1_t index`, `e32xm1_t mask`, `unsigned int gvl`)
- `int32xm2_t vlxhv_mask_int32xm2` (`int32xm2_t merge`, `const int *address`, `int32xm2_t index`, `e32xm2_t mask`, `unsigned int gvl`)
- `int32xm4_t vlxhv_mask_int32xm4` (`int32xm4_t merge`, `const int *address`, `int32xm4_t index`, `e32xm4_t mask`, `unsigned int gvl`)
- `int32xm8_t vlxhv_mask_int32xm8` (`int32xm8_t merge`, `const int *address`, `int32xm8_t index`, `e32xm8_t mask`, `unsigned int gvl`)
- `int64xm1_t vlxhv_mask_int64xm1` (`int64xm1_t merge`, `const long *address`, `int64xm1_t index`, `e64xm1_t mask`, `unsigned int gvl`)
- `int64xm2_t vlxhv_mask_int64xm2` (`int64xm2_t merge`, `const long *address`, `int64xm2_t index`, `e64xm2_t mask`, `unsigned int gvl`)
- `int64xm4_t vlxhv_mask_int64xm4` (`int64xm4_t merge`, `const long *address`, `int64xm4_t index`, `e64xm4_t mask`, `unsigned int gvl`)
- `int64xm8_t vlxhv_mask_int64xm8` (`int64xm8_t merge`, `const long *address`, `int64xm8_t index`, `e64xm8_t mask`, `unsigned int gvl`)
- `int8xm1_t vlxhv_mask_int8xm1` (`int8xm1_t merge`, `const signed char *address`, `int8xm1_t index`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8xm2_t vlxhv_mask_int8xm2` (`int8xm2_t merge`, `const signed char *address`, `int8xm2_t index`, `e8xm2_t mask`, `unsigned int gvl`)
- `int8xm4_t vlxhv_mask_int8xm4` (`int8xm4_t merge`, `const signed char *address`, `int8xm4_t index`, `e8xm4_t mask`, `unsigned int gvl`)
- `int8xm8_t vlxhv_mask_int8xm8` (`int8xm8_t merge`, `const signed char *address`, `int8xm8_t index`, `e8xm8_t mask`, `unsigned int gvl`)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
    result[gvl : VLMAX] = 0
    for element = 0 to gvl - 1
        if mask[element] then
            result[element] = load_element(address + index[element])
        else
            result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

### 2.9.19 Load indexed 16b unsigned in memory to vector

**Instruction:** [`'vlxhu.v'`]

#### Prototypes:

- `uint16xm1_t vlxhuv_uint16xm1` (`const unsigned short *address`, `uint16xm1_t index`, `unsigned int gvl`)
- `uint16xm2_t vlxhuv_uint16xm2` (`const unsigned short *address`, `uint16xm2_t index`, `unsigned int gvl`)
- `uint16xm4_t vlxhuv_uint16xm4` (`const unsigned short *address`, `uint16xm4_t index`, `unsigned int gvl`)
- `uint16xm8_t vlxhuv_uint16xm8` (`const unsigned short *address`, `uint16xm8_t index`, `unsigned int gvl`)

- `uint32xm1_t vlxhuv_uint32xm1` (const unsigned int \*address, `uint32xm1_t` index, unsigned int gvl)
- `uint32xm2_t vlxhuv_uint32xm2` (const unsigned int \*address, `uint32xm2_t` index, unsigned int gvl)
- `uint32xm4_t vlxhuv_uint32xm4` (const unsigned int \*address, `uint32xm4_t` index, unsigned int gvl)
- `uint32xm8_t vlxhuv_uint32xm8` (const unsigned int \*address, `uint32xm8_t` index, unsigned int gvl)
- `uint64xm1_t vlxhuv_uint64xm1` (const unsigned long \*address, `uint64xm1_t` index, unsigned int gvl)
- `uint64xm2_t vlxhuv_uint64xm2` (const unsigned long \*address, `uint64xm2_t` index, unsigned int gvl)
- `uint64xm4_t vlxhuv_uint64xm4` (const unsigned long \*address, `uint64xm4_t` index, unsigned int gvl)
- `uint64xm8_t vlxhuv_uint64xm8` (const unsigned long \*address, `uint64xm8_t` index, unsigned int gvl)
- `uint8xm1_t vlxhuv_uint8xm1` (const unsigned char \*address, `uint8xm1_t` index, unsigned int gvl)
- `uint8xm2_t vlxhuv_uint8xm2` (const unsigned char \*address, `uint8xm2_t` index, unsigned int gvl)
- `uint8xm4_t vlxhuv_uint8xm4` (const unsigned char \*address, `uint8xm4_t` index, unsigned int gvl)
- `uint8xm8_t vlxhuv_uint8xm8` (const unsigned char \*address, `uint8xm8_t` index, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = load_element(address + index[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vlxhuv_mask_uint16xm1` (`uint16xm1_t` merge, const unsigned short \*address, `uint16xm1_t` index, `e16xm1_t` mask, unsigned int gvl)
- `uint16xm2_t vlxhuv_mask_uint16xm2` (`uint16xm2_t` merge, const unsigned short \*address, `uint16xm2_t` index, `e16xm2_t` mask, unsigned int gvl)
- `uint16xm4_t vlxhuv_mask_uint16xm4` (`uint16xm4_t` merge, const unsigned short \*address, `uint16xm4_t` index, `e16xm4_t` mask, unsigned int gvl)
- `uint16xm8_t vlxhuv_mask_uint16xm8` (`uint16xm8_t` merge, const unsigned short \*address, `uint16xm8_t` index, `e16xm8_t` mask, unsigned int gvl)
- `uint32xm1_t vlxhuv_mask_uint32xm1` (`uint32xm1_t` merge, const unsigned int \*address, `uint32xm1_t` index, `e32xm1_t` mask, unsigned int gvl)
- `uint32xm2_t vlxhuv_mask_uint32xm2` (`uint32xm2_t` merge, const unsigned int \*address, `uint32xm2_t` index, `e32xm2_t` mask, unsigned int gvl)
- `uint32xm4_t vlxhuv_mask_uint32xm4` (`uint32xm4_t` merge, const unsigned int \*address, `uint32xm4_t` index, `e32xm4_t` mask, unsigned int gvl)
- `uint32xm8_t vlxhuv_mask_uint32xm8` (`uint32xm8_t` merge, const unsigned int \*address, `uint32xm8_t` index, `e32xm8_t` mask, unsigned int gvl)
- `uint64xm1_t vlxhuv_mask_uint64xm1` (`uint64xm1_t` merge, const unsigned long \*address, `uint64xm1_t` index, `e64xm1_t` mask, unsigned int gvl)
- `uint64xm2_t vlxhuv_mask_uint64xm2` (`uint64xm2_t` merge, const unsigned long \*address, `uint64xm2_t` index, `e64xm2_t` mask, unsigned int gvl)
- `uint64xm4_t vlxhuv_mask_uint64xm4` (`uint64xm4_t` merge, const unsigned long \*address, `uint64xm4_t` index, `e64xm4_t` mask, unsigned int gvl)
- `uint64xm8_t vlxhuv_mask_uint64xm8` (`uint64xm8_t` merge, const unsigned long \*address, `uint64xm8_t` index, `e64xm8_t` mask, unsigned int gvl)



- `uint8xm1_t vlxhuv_mask_uint8xm1 (uint8xm1_t merge, const unsigned char *address, uint8xm1_t index, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vlxhuv_mask_uint8xm2 (uint8xm2_t merge, const unsigned char *address, uint8xm2_t index, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vlxhuv_mask_uint8xm4 (uint8xm4_t merge, const unsigned char *address, uint8xm4_t index, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vlxhuv_mask_uint8xm8 (uint8xm8_t merge, const unsigned char *address, uint8xm8_t index, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      result[gvl : VLMAX] = 0
      for element = 0 to gvl - 1
        if mask[element] then
          result[element] = load_element(address + index[element])
        else
          result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.9.20 Load indexed 32b signed in memory to vector

**Instruction:** [`vlxw.v`']

**Prototypes:**

- `int16xm1_t vlxwv_int16xm1 (const short *address, int16xm1_t index, unsigned int gvl)`
- `int16xm2_t vlxwv_int16xm2 (const short *address, int16xm2_t index, unsigned int gvl)`
- `int16xm4_t vlxwv_int16xm4 (const short *address, int16xm4_t index, unsigned int gvl)`
- `int16xm8_t vlxwv_int16xm8 (const short *address, int16xm8_t index, unsigned int gvl)`
- `int32xm1_t vlxwv_int32xm1 (const int *address, int32xm1_t index, unsigned int gvl)`
- `int32xm2_t vlxwv_int32xm2 (const int *address, int32xm2_t index, unsigned int gvl)`
- `int32xm4_t vlxwv_int32xm4 (const int *address, int32xm4_t index, unsigned int gvl)`
- `int32xm8_t vlxwv_int32xm8 (const int *address, int32xm8_t index, unsigned int gvl)`
- `int64xm1_t vlxwv_int64xm1 (const long *address, int64xm1_t index, unsigned int gvl)`
- `int64xm2_t vlxwv_int64xm2 (const long *address, int64xm2_t index, unsigned int gvl)`
- `int64xm4_t vlxwv_int64xm4 (const long *address, int64xm4_t index, unsigned int gvl)`
- `int64xm8_t vlxwv_int64xm8 (const long *address, int64xm8_t index, unsigned int gvl)`
- `int8xm1_t vlxwv_int8xm1 (const signed char *address, int8xm1_t index, unsigned int gvl)`
- `int8xm2_t vlxwv_int8xm2 (const signed char *address, int8xm2_t index, unsigned int gvl)`
- `int8xm4_t vlxwv_int8xm4 (const signed char *address, int8xm4_t index, unsigned int gvl)`
- `int8xm8_t vlxwv_int8xm8 (const signed char *address, int8xm8_t index, unsigned int gvl)`

**Operation:**



```
>>> for element = 0 to gvl - 1
      result[element] = load_element(address + index[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *int16xm1\_t vlxwv\_mask\_int16xm1* (*int16xm1\_t* merge, const short \**address*, *int16xm1\_t* index, *e16xm1\_t* mask, unsigned int *gvl*)
- *int16xm2\_t vlxwv\_mask\_int16xm2* (*int16xm2\_t* merge, const short \**address*, *int16xm2\_t* index, *e16xm2\_t* mask, unsigned int *gvl*)
- *int16xm4\_t vlxwv\_mask\_int16xm4* (*int16xm4\_t* merge, const short \**address*, *int16xm4\_t* index, *e16xm4\_t* mask, unsigned int *gvl*)
- *int16xm8\_t vlxwv\_mask\_int16xm8* (*int16xm8\_t* merge, const short \**address*, *int16xm8\_t* index, *e16xm8\_t* mask, unsigned int *gvl*)
- *int32xm1\_t vlxwv\_mask\_int32xm1* (*int32xm1\_t* merge, const int \**address*, *int32xm1\_t* index, *e32xm1\_t* mask, unsigned int *gvl*)
- *int32xm2\_t vlxwv\_mask\_int32xm2* (*int32xm2\_t* merge, const int \**address*, *int32xm2\_t* index, *e32xm2\_t* mask, unsigned int *gvl*)
- *int32xm4\_t vlxwv\_mask\_int32xm4* (*int32xm4\_t* merge, const int \**address*, *int32xm4\_t* index, *e32xm4\_t* mask, unsigned int *gvl*)
- *int32xm8\_t vlxwv\_mask\_int32xm8* (*int32xm8\_t* merge, const int \**address*, *int32xm8\_t* index, *e32xm8\_t* mask, unsigned int *gvl*)
- *int64xm1\_t vlxwv\_mask\_int64xm1* (*int64xm1\_t* merge, const long \**address*, *int64xm1\_t* index, *e64xm1\_t* mask, unsigned int *gvl*)
- *int64xm2\_t vlxwv\_mask\_int64xm2* (*int64xm2\_t* merge, const long \**address*, *int64xm2\_t* index, *e64xm2\_t* mask, unsigned int *gvl*)
- *int64xm4\_t vlxwv\_mask\_int64xm4* (*int64xm4\_t* merge, const long \**address*, *int64xm4\_t* index, *e64xm4\_t* mask, unsigned int *gvl*)
- *int64xm8\_t vlxwv\_mask\_int64xm8* (*int64xm8\_t* merge, const long \**address*, *int64xm8\_t* index, *e64xm8\_t* mask, unsigned int *gvl*)
- *int8xm1\_t vlxwv\_mask\_int8xm1* (*int8xm1\_t* merge, const signed char \**address*, *int8xm1\_t* index, *e8xm1\_t* mask, unsigned int *gvl*)
- *int8xm2\_t vlxwv\_mask\_int8xm2* (*int8xm2\_t* merge, const signed char \**address*, *int8xm2\_t* index, *e8xm2\_t* mask, unsigned int *gvl*)
- *int8xm4\_t vlxwv\_mask\_int8xm4* (*int8xm4\_t* merge, const signed char \**address*, *int8xm4\_t* index, *e8xm4\_t* mask, unsigned int *gvl*)
- *int8xm8\_t vlxwv\_mask\_int8xm8* (*int8xm8\_t* merge, const signed char \**address*, *int8xm8\_t* index, *e8xm8\_t* mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      result[gvl : VLMAX] = 0
      for element = 0 to gvl - 1
          if mask[element] then
              result[element] = load_element(address + index[element])
          else
              result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

## 2.9.21 Load indexed 32b unsigned in memory to vector

**Instruction:** [`'vlxwu.v'`]

**Prototypes:**

- `uint16xm1_t vlxwuv_uint16xm1` (const unsigned short \*address, `uint16xm1_t` index, unsigned int gvl)
- `uint16xm2_t vlxwuv_uint16xm2` (const unsigned short \*address, `uint16xm2_t` index, unsigned int gvl)
- `uint16xm4_t vlxwuv_uint16xm4` (const unsigned short \*address, `uint16xm4_t` index, unsigned int gvl)
- `uint16xm8_t vlxwuv_uint16xm8` (const unsigned short \*address, `uint16xm8_t` index, unsigned int gvl)
- `uint32xm1_t vlxwuv_uint32xm1` (const unsigned int \*address, `uint32xm1_t` index, unsigned int gvl)
- `uint32xm2_t vlxwuv_uint32xm2` (const unsigned int \*address, `uint32xm2_t` index, unsigned int gvl)
- `uint32xm4_t vlxwuv_uint32xm4` (const unsigned int \*address, `uint32xm4_t` index, unsigned int gvl)
- `uint32xm8_t vlxwuv_uint32xm8` (const unsigned int \*address, `uint32xm8_t` index, unsigned int gvl)
- `uint64xm1_t vlxwuv_uint64xm1` (const unsigned long \*address, `uint64xm1_t` index, unsigned int gvl)
- `uint64xm2_t vlxwuv_uint64xm2` (const unsigned long \*address, `uint64xm2_t` index, unsigned int gvl)
- `uint64xm4_t vlxwuv_uint64xm4` (const unsigned long \*address, `uint64xm4_t` index, unsigned int gvl)
- `uint64xm8_t vlxwuv_uint64xm8` (const unsigned long \*address, `uint64xm8_t` index, unsigned int gvl)
- `uint8xm1_t vlxwuv_uint8xm1` (const unsigned char \*address, `uint8xm1_t` index, unsigned int gvl)
- `uint8xm2_t vlxwuv_uint8xm2` (const unsigned char \*address, `uint8xm2_t` index, unsigned int gvl)
- `uint8xm4_t vlxwuv_uint8xm4` (const unsigned char \*address, `uint8xm4_t` index, unsigned int gvl)
- `uint8xm8_t vlxwuv_uint8xm8` (const unsigned char \*address, `uint8xm8_t` index, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = load_element(address + index[element])
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vlxwuv_mask_uint16xm1` (`uint16xm1_t` merge, const unsigned short \*address, `uint16xm1_t` index, `e16xm1_t` mask, unsigned int gvl)
- `uint16xm2_t vlxwuv_mask_uint16xm2` (`uint16xm2_t` merge, const unsigned short \*address, `uint16xm2_t` index, `e16xm2_t` mask, unsigned int gvl)
- `uint16xm4_t vlxwuv_mask_uint16xm4` (`uint16xm4_t` merge, const unsigned short \*address, `uint16xm4_t` index, `e16xm4_t` mask, unsigned int gvl)
- `uint16xm8_t vlxwuv_mask_uint16xm8` (`uint16xm8_t` merge, const unsigned short \*address, `uint16xm8_t` index, `e16xm8_t` mask, unsigned int gvl)
- `uint32xm1_t vlxwuv_mask_uint32xm1` (`uint32xm1_t` merge, const unsigned int \*address, `uint32xm1_t` index, `e32xm1_t` mask, unsigned int gvl)
- `uint32xm2_t vlxwuv_mask_uint32xm2` (`uint32xm2_t` merge, const unsigned int \*address, `uint32xm2_t` index, `e32xm2_t` mask, unsigned int gvl)

- `uint32xm4_t vlxwuv_mask_uint32xm4 (uint32xm4_t merge, const unsigned int *address, uint32xm4_t index, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vlxwuv_mask_uint32xm8 (uint32xm8_t merge, const unsigned int *address, uint32xm8_t index, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vlxwuv_mask_uint64xm1 (uint64xm1_t merge, const unsigned long *address, uint64xm1_t index, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vlxwuv_mask_uint64xm2 (uint64xm2_t merge, const unsigned long *address, uint64xm2_t index, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vlxwuv_mask_uint64xm4 (uint64xm4_t merge, const unsigned long *address, uint64xm4_t index, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vlxwuv_mask_uint64xm8 (uint64xm8_t merge, const unsigned long *address, uint64xm8_t index, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vlxwuv_mask_uint8xm1 (uint8xm1_t merge, const unsigned char *address, uint8xm1_t index, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vlxwuv_mask_uint8xm2 (uint8xm2_t merge, const unsigned char *address, uint8xm2_t index, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vlxwuv_mask_uint8xm4 (uint8xm4_t merge, const unsigned char *address, uint8xm4_t index, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vlxwuv_mask_uint8xm8 (uint8xm8_t merge, const unsigned char *address, uint8xm8_t index, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
    result[gvl : VLMAX] = 0
    for element = 0 to gvl - 1
        if mask[element] then
            result[element] = load_element(address + index[element])
        else
            result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

## 2.9.22 Store 8b in memory from vector

**Instruction:** [`*vsb.v'`]

#### Prototypes:

- void `vsbv_int16xm1` (short \*address, `int16xm1_t` a, unsigned int gvl)
- void `vsbv_int16xm2` (short \*address, `int16xm2_t` a, unsigned int gvl)
- void `vsbv_int16xm4` (short \*address, `int16xm4_t` a, unsigned int gvl)
- void `vsbv_int16xm8` (short \*address, `int16xm8_t` a, unsigned int gvl)
- void `vsbv_int32xm1` (int \*address, `int32xm1_t` a, unsigned int gvl)
- void `vsbv_int32xm2` (int \*address, `int32xm2_t` a, unsigned int gvl)
- void `vsbv_int32xm4` (int \*address, `int32xm4_t` a, unsigned int gvl)
- void `vsbv_int32xm8` (int \*address, `int32xm8_t` a, unsigned int gvl)
- void `vsbv_int64xm1` (long \*address, `int64xm1_t` a, unsigned int gvl)
- void `vsbv_int64xm2` (long \*address, `int64xm2_t` a, unsigned int gvl)

- void **vsbv\_int64xm4** (long \*address, *int64xm4\_t* a, unsigned int gvl)
- void **vsbv\_int64xm8** (long \*address, *int64xm8\_t* a, unsigned int gvl)
- void **vsbv\_int8xm1** (signed char \*address, *int8xm1\_t* a, unsigned int gvl)
- void **vsbv\_int8xm2** (signed char \*address, *int8xm2\_t* a, unsigned int gvl)
- void **vsbv\_int8xm4** (signed char \*address, *int8xm4\_t* a, unsigned int gvl)
- void **vsbv\_int8xm8** (signed char \*address, *int8xm8\_t* a, unsigned int gvl)
- void **vsbv\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* a, unsigned int gvl)
- void **vsbv\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* a, unsigned int gvl)
- void **vsbv\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* a, unsigned int gvl)
- void **vsbv\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* a, unsigned int gvl)
- void **vsbv\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* a, unsigned int gvl)
- void **vsbv\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* a, unsigned int gvl)
- void **vsbv\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* a, unsigned int gvl)
- void **vsbv\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* a, unsigned int gvl)
- void **vsbv\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* a, unsigned int gvl)
- void **vsbv\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* a, unsigned int gvl)
- void **vsbv\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* a, unsigned int gvl)
- void **vsbv\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* a, unsigned int gvl)
- void **vsbv\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* a, unsigned int gvl)
- void **vsbv\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* a, unsigned int gvl)
- void **vsbv\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* a, unsigned int gvl)
- void **vsbv\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* a, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      store_element(address, a[element])
      address = address + 1
```

**Masked prototypes:**

- void **vsbv\_mask\_int16xm1** (short \*address, *int16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsbv\_mask\_int16xm2** (short \*address, *int16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsbv\_mask\_int16xm4** (short \*address, *int16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsbv\_mask\_int16xm8** (short \*address, *int16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsbv\_mask\_int32xm1** (int \*address, *int32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsbv\_mask\_int32xm2** (int \*address, *int32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsbv\_mask\_int32xm4** (int \*address, *int32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsbv\_mask\_int32xm8** (int \*address, *int32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsbv\_mask\_int64xm1** (long \*address, *int64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)

- void **vsbv\_mask\_int64xm2** (long \*address, *int64xm2\_t a*, *e64xm2\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_int64xm4** (long \*address, *int64xm4\_t a*, *e64xm4\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_int64xm8** (long \*address, *int64xm8\_t a*, *e64xm8\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_int8xm1** (signed char \*address, *int8xm1\_t a*, *e8xm1\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_int8xm2** (signed char \*address, *int8xm2\_t a*, *e8xm2\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_int8xm4** (signed char \*address, *int8xm4\_t a*, *e8xm4\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_int8xm8** (signed char \*address, *int8xm8\_t a*, *e8xm8\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint16xm1** (unsigned short \*address, *uint16xm1\_t a*, *e16xm1\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint16xm2** (unsigned short \*address, *uint16xm2\_t a*, *e16xm2\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint16xm4** (unsigned short \*address, *uint16xm4\_t a*, *e16xm4\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint16xm8** (unsigned short \*address, *uint16xm8\_t a*, *e16xm8\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint32xm1** (unsigned int \*address, *uint32xm1\_t a*, *e32xm1\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint32xm2** (unsigned int \*address, *uint32xm2\_t a*, *e32xm2\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint32xm4** (unsigned int \*address, *uint32xm4\_t a*, *e32xm4\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint32xm8** (unsigned int \*address, *uint32xm8\_t a*, *e32xm8\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint64xm1** (unsigned long \*address, *uint64xm1\_t a*, *e64xm1\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint64xm2** (unsigned long \*address, *uint64xm2\_t a*, *e64xm2\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint64xm4** (unsigned long \*address, *uint64xm4\_t a*, *e64xm4\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint64xm8** (unsigned long \*address, *uint64xm8\_t a*, *e64xm8\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint8xm1** (unsigned char \*address, *uint8xm1\_t a*, *e8xm1\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint8xm2** (unsigned char \*address, *uint8xm2\_t a*, *e8xm2\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint8xm4** (unsigned char \*address, *uint8xm4\_t a*, *e8xm4\_t mask*, unsigned int gvl)
- void **vsbv\_mask\_uint8xm8** (unsigned char \*address, *uint8xm8\_t a*, *e8xm8\_t mask*, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address, a[element])
        address = address + 1
```

## 2.9.23 Store elements in memory from vector

**Instruction:** [`'vse.v'`]

**Prototypes:**

- void **vsev\_float16xm1** (float16\_t \*address, *float16xm1\_t* a, unsigned int gvl)
- void **vsev\_float16xm2** (float16\_t \*address, *float16xm2\_t* a, unsigned int gvl)
- void **vsev\_float16xm4** (float16\_t \*address, *float16xm4\_t* a, unsigned int gvl)
- void **vsev\_float16xm8** (float16\_t \*address, *float16xm8\_t* a, unsigned int gvl)
- void **vsev\_float32xm1** (float \*address, *float32xm1\_t* a, unsigned int gvl)
- void **vsev\_float32xm2** (float \*address, *float32xm2\_t* a, unsigned int gvl)
- void **vsev\_float32xm4** (float \*address, *float32xm4\_t* a, unsigned int gvl)
- void **vsev\_float32xm8** (float \*address, *float32xm8\_t* a, unsigned int gvl)
- void **vsev\_float64xm1** (double \*address, *float64xm1\_t* a, unsigned int gvl)
- void **vsev\_float64xm2** (double \*address, *float64xm2\_t* a, unsigned int gvl)
- void **vsev\_float64xm4** (double \*address, *float64xm4\_t* a, unsigned int gvl)
- void **vsev\_float64xm8** (double \*address, *float64xm8\_t* a, unsigned int gvl)
- void **vsev\_int16xm1** (short \*address, *int16xm1\_t* a, unsigned int gvl)
- void **vsev\_int16xm2** (short \*address, *int16xm2\_t* a, unsigned int gvl)
- void **vsev\_int16xm4** (short \*address, *int16xm4\_t* a, unsigned int gvl)
- void **vsev\_int16xm8** (short \*address, *int16xm8\_t* a, unsigned int gvl)
- void **vsev\_int32xm1** (int \*address, *int32xm1\_t* a, unsigned int gvl)
- void **vsev\_int32xm2** (int \*address, *int32xm2\_t* a, unsigned int gvl)
- void **vsev\_int32xm4** (int \*address, *int32xm4\_t* a, unsigned int gvl)
- void **vsev\_int32xm8** (int \*address, *int32xm8\_t* a, unsigned int gvl)
- void **vsev\_int64xm1** (long \*address, *int64xm1\_t* a, unsigned int gvl)
- void **vsev\_int64xm2** (long \*address, *int64xm2\_t* a, unsigned int gvl)
- void **vsev\_int64xm4** (long \*address, *int64xm4\_t* a, unsigned int gvl)
- void **vsev\_int64xm8** (long \*address, *int64xm8\_t* a, unsigned int gvl)
- void **vsev\_int8xm1** (signed char \*address, *int8xm1\_t* a, unsigned int gvl)
- void **vsev\_int8xm2** (signed char \*address, *int8xm2\_t* a, unsigned int gvl)
- void **vsev\_int8xm4** (signed char \*address, *int8xm4\_t* a, unsigned int gvl)
- void **vsev\_int8xm8** (signed char \*address, *int8xm8\_t* a, unsigned int gvl)
- void **vsev\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* a, unsigned int gvl)
- void **vsev\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* a, unsigned int gvl)
- void **vsev\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* a, unsigned int gvl)
- void **vsev\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* a, unsigned int gvl)
- void **vsev\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* a, unsigned int gvl)



- void **vsev\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* a, unsigned int gvl)
- void **vsev\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* a, unsigned int gvl)
- void **vsev\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* a, unsigned int gvl)
- void **vsev\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* a, unsigned int gvl)
- void **vsev\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* a, unsigned int gvl)
- void **vsev\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* a, unsigned int gvl)
- void **vsev\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* a, unsigned int gvl)
- void **vsev\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* a, unsigned int gvl)
- void **vsev\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* a, unsigned int gvl)
- void **vsev\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* a, unsigned int gvl)
- void **vsev\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* a, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
    store_element(address, a[element])
    address = address + SEW / 8
```

**Masked prototypes:**

- void **vsev\_mask\_float16xm1** (float16\_t \*address, *float16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsev\_mask\_float16xm2** (float16\_t \*address, *float16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsev\_mask\_float16xm4** (float16\_t \*address, *float16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsev\_mask\_float16xm8** (float16\_t \*address, *float16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsev\_mask\_float32xm1** (float \*address, *float32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsev\_mask\_float32xm2** (float \*address, *float32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsev\_mask\_float32xm4** (float \*address, *float32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsev\_mask\_float32xm8** (float \*address, *float32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsev\_mask\_float64xm1** (double \*address, *float64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsev\_mask\_float64xm2** (double \*address, *float64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsev\_mask\_float64xm4** (double \*address, *float64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsev\_mask\_float64xm8** (double \*address, *float64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int16xm1** (short \*address, *int16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int16xm2** (short \*address, *int16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int16xm4** (short \*address, *int16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int16xm8** (short \*address, *int16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int32xm1** (int \*address, *int32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int32xm2** (int \*address, *int32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int32xm4** (int \*address, *int32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int32xm8** (int \*address, *int32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)

- void **vsev\_mask\_int64xm1** (long \*address, *int64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int64xm2** (long \*address, *int64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int64xm4** (long \*address, *int64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int64xm8** (long \*address, *int64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int8xm1** (signed char \*address, *int8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int8xm2** (signed char \*address, *int8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int8xm4** (signed char \*address, *int8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsev\_mask\_int8xm8** (signed char \*address, *int8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsev\_mask\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address, a[element])
        address = address + SEW / 8
```

## 2.9.24 Store 16b in memory from vector

**Instruction:** [`vsh.v`']

**Prototypes:**

- void **vshv\_int16xm1** (short *\*address*, *int16xm1\_t a*, unsigned int *gvl*)
- void **vshv\_int16xm2** (short *\*address*, *int16xm2\_t a*, unsigned int *gvl*)
- void **vshv\_int16xm4** (short *\*address*, *int16xm4\_t a*, unsigned int *gvl*)
- void **vshv\_int16xm8** (short *\*address*, *int16xm8\_t a*, unsigned int *gvl*)
- void **vshv\_int32xm1** (int *\*address*, *int32xm1\_t a*, unsigned int *gvl*)
- void **vshv\_int32xm2** (int *\*address*, *int32xm2\_t a*, unsigned int *gvl*)
- void **vshv\_int32xm4** (int *\*address*, *int32xm4\_t a*, unsigned int *gvl*)
- void **vshv\_int32xm8** (int *\*address*, *int32xm8\_t a*, unsigned int *gvl*)
- void **vshv\_int64xm1** (long *\*address*, *int64xm1\_t a*, unsigned int *gvl*)
- void **vshv\_int64xm2** (long *\*address*, *int64xm2\_t a*, unsigned int *gvl*)
- void **vshv\_int64xm4** (long *\*address*, *int64xm4\_t a*, unsigned int *gvl*)
- void **vshv\_int64xm8** (long *\*address*, *int64xm8\_t a*, unsigned int *gvl*)
- void **vshv\_int8xm1** (signed char *\*address*, *int8xm1\_t a*, unsigned int *gvl*)
- void **vshv\_int8xm2** (signed char *\*address*, *int8xm2\_t a*, unsigned int *gvl*)
- void **vshv\_int8xm4** (signed char *\*address*, *int8xm4\_t a*, unsigned int *gvl*)
- void **vshv\_int8xm8** (signed char *\*address*, *int8xm8\_t a*, unsigned int *gvl*)
- void **vshv\_uint16xm1** (unsigned short *\*address*, *uint16xm1\_t a*, unsigned int *gvl*)
- void **vshv\_uint16xm2** (unsigned short *\*address*, *uint16xm2\_t a*, unsigned int *gvl*)
- void **vshv\_uint16xm4** (unsigned short *\*address*, *uint16xm4\_t a*, unsigned int *gvl*)
- void **vshv\_uint16xm8** (unsigned short *\*address*, *uint16xm8\_t a*, unsigned int *gvl*)
- void **vshv\_uint32xm1** (unsigned int *\*address*, *uint32xm1\_t a*, unsigned int *gvl*)
- void **vshv\_uint32xm2** (unsigned int *\*address*, *uint32xm2\_t a*, unsigned int *gvl*)
- void **vshv\_uint32xm4** (unsigned int *\*address*, *uint32xm4\_t a*, unsigned int *gvl*)
- void **vshv\_uint32xm8** (unsigned int *\*address*, *uint32xm8\_t a*, unsigned int *gvl*)
- void **vshv\_uint64xm1** (unsigned long *\*address*, *uint64xm1\_t a*, unsigned int *gvl*)
- void **vshv\_uint64xm2** (unsigned long *\*address*, *uint64xm2\_t a*, unsigned int *gvl*)
- void **vshv\_uint64xm4** (unsigned long *\*address*, *uint64xm4\_t a*, unsigned int *gvl*)
- void **vshv\_uint64xm8** (unsigned long *\*address*, *uint64xm8\_t a*, unsigned int *gvl*)
- void **vshv\_uint8xm1** (unsigned char *\*address*, *uint8xm1\_t a*, unsigned int *gvl*)
- void **vshv\_uint8xm2** (unsigned char *\*address*, *uint8xm2\_t a*, unsigned int *gvl*)
- void **vshv\_uint8xm4** (unsigned char *\*address*, *uint8xm4\_t a*, unsigned int *gvl*)
- void **vshv\_uint8xm8** (unsigned char *\*address*, *uint8xm8\_t a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    store_element(address, a[element])
    address = address + 1
```

### Masked prototypes:

- void **vshv\_mask\_int16xm1** (short \*address, *int16xm1\_t a*, *e16xm1\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int16xm2** (short \*address, *int16xm2\_t a*, *e16xm2\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int16xm4** (short \*address, *int16xm4\_t a*, *e16xm4\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int16xm8** (short \*address, *int16xm8\_t a*, *e16xm8\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int32xm1** (int \*address, *int32xm1\_t a*, *e32xm1\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int32xm2** (int \*address, *int32xm2\_t a*, *e32xm2\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int32xm4** (int \*address, *int32xm4\_t a*, *e32xm4\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int32xm8** (int \*address, *int32xm8\_t a*, *e32xm8\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int64xm1** (long \*address, *int64xm1\_t a*, *e64xm1\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int64xm2** (long \*address, *int64xm2\_t a*, *e64xm2\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int64xm4** (long \*address, *int64xm4\_t a*, *e64xm4\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int64xm8** (long \*address, *int64xm8\_t a*, *e64xm8\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int8xm1** (signed char \*address, *int8xm1\_t a*, *e8xm1\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int8xm2** (signed char \*address, *int8xm2\_t a*, *e8xm2\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int8xm4** (signed char \*address, *int8xm4\_t a*, *e8xm4\_t mask*, unsigned int gvl)
- void **vshv\_mask\_int8xm8** (signed char \*address, *int8xm8\_t a*, *e8xm8\_t mask*, unsigned int gvl)
- void **vshv\_mask\_uint16xm1** (unsigned short \*address, *uint16xm1\_t a*, *e16xm1\_t mask*, unsigned int gvl)
- void **vshv\_mask\_uint16xm2** (unsigned short \*address, *uint16xm2\_t a*, *e16xm2\_t mask*, unsigned int gvl)
- void **vshv\_mask\_uint16xm4** (unsigned short \*address, *uint16xm4\_t a*, *e16xm4\_t mask*, unsigned int gvl)
- void **vshv\_mask\_uint16xm8** (unsigned short \*address, *uint16xm8\_t a*, *e16xm8\_t mask*, unsigned int gvl)
- void **vshv\_mask\_uint32xm1** (unsigned int \*address, *uint32xm1\_t a*, *e32xm1\_t mask*, unsigned int gvl)
- void **vshv\_mask\_uint32xm2** (unsigned int \*address, *uint32xm2\_t a*, *e32xm2\_t mask*, unsigned int gvl)
- void **vshv\_mask\_uint32xm4** (unsigned int \*address, *uint32xm4\_t a*, *e32xm4\_t mask*, unsigned int gvl)
- void **vshv\_mask\_uint32xm8** (unsigned int \*address, *uint32xm8\_t a*, *e32xm8\_t mask*, unsigned int gvl)
- void **vshv\_mask\_uint64xm1** (unsigned long \*address, *uint64xm1\_t a*, *e64xm1\_t mask*, unsigned int gvl)
- void **vshv\_mask\_uint64xm2** (unsigned long \*address, *uint64xm2\_t a*, *e64xm2\_t mask*, unsigned int gvl)

- void **vshv\_mask\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vshv\_mask\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vshv\_mask\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vshv\_mask\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vshv\_mask\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vshv\_mask\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address, a[element])
        address = address + 1
```

### 2.9.25 Store 8b in strided memory from vector

**Instruction:** [*vssb.v*']

#### Prototypes:

- void **vssbv\_int16xm1** (short \*address, long stride, *int16xm1\_t* a, unsigned int gvl)
- void **vssbv\_int16xm2** (short \*address, long stride, *int16xm2\_t* a, unsigned int gvl)
- void **vssbv\_int16xm4** (short \*address, long stride, *int16xm4\_t* a, unsigned int gvl)
- void **vssbv\_int16xm8** (short \*address, long stride, *int16xm8\_t* a, unsigned int gvl)
- void **vssbv\_int32xm1** (int \*address, long stride, *int32xm1\_t* a, unsigned int gvl)
- void **vssbv\_int32xm2** (int \*address, long stride, *int32xm2\_t* a, unsigned int gvl)
- void **vssbv\_int32xm4** (int \*address, long stride, *int32xm4\_t* a, unsigned int gvl)
- void **vssbv\_int32xm8** (int \*address, long stride, *int32xm8\_t* a, unsigned int gvl)
- void **vssbv\_int64xm1** (long \*address, long stride, *int64xm1\_t* a, unsigned int gvl)
- void **vssbv\_int64xm2** (long \*address, long stride, *int64xm2\_t* a, unsigned int gvl)
- void **vssbv\_int64xm4** (long \*address, long stride, *int64xm4\_t* a, unsigned int gvl)
- void **vssbv\_int64xm8** (long \*address, long stride, *int64xm8\_t* a, unsigned int gvl)
- void **vssbv\_int8xm1** (signed char \*address, long stride, *int8xm1\_t* a, unsigned int gvl)
- void **vssbv\_int8xm2** (signed char \*address, long stride, *int8xm2\_t* a, unsigned int gvl)
- void **vssbv\_int8xm4** (signed char \*address, long stride, *int8xm4\_t* a, unsigned int gvl)
- void **vssbv\_int8xm8** (signed char \*address, long stride, *int8xm8\_t* a, unsigned int gvl)
- void **vssbv\_uint16xm1** (unsigned short \*address, long stride, *uint16xm1\_t* a, unsigned int gvl)
- void **vssbv\_uint16xm2** (unsigned short \*address, long stride, *uint16xm2\_t* a, unsigned int gvl)
- void **vssbv\_uint16xm4** (unsigned short \*address, long stride, *uint16xm4\_t* a, unsigned int gvl)
- void **vssbv\_uint16xm8** (unsigned short \*address, long stride, *uint16xm8\_t* a, unsigned int gvl)

- void **vssbv\_uint32xm1** (unsigned int \*address, long stride, *uint32xm1\_t* a, unsigned int gvl)
- void **vssbv\_uint32xm2** (unsigned int \*address, long stride, *uint32xm2\_t* a, unsigned int gvl)
- void **vssbv\_uint32xm4** (unsigned int \*address, long stride, *uint32xm4\_t* a, unsigned int gvl)
- void **vssbv\_uint32xm8** (unsigned int \*address, long stride, *uint32xm8\_t* a, unsigned int gvl)
- void **vssbv\_uint64xm1** (unsigned long \*address, long stride, *uint64xm1\_t* a, unsigned int gvl)
- void **vssbv\_uint64xm2** (unsigned long \*address, long stride, *uint64xm2\_t* a, unsigned int gvl)
- void **vssbv\_uint64xm4** (unsigned long \*address, long stride, *uint64xm4\_t* a, unsigned int gvl)
- void **vssbv\_uint64xm8** (unsigned long \*address, long stride, *uint64xm8\_t* a, unsigned int gvl)
- void **vssbv\_uint8xm1** (unsigned char \*address, long stride, *uint8xm1\_t* a, unsigned int gvl)
- void **vssbv\_uint8xm2** (unsigned char \*address, long stride, *uint8xm2\_t* a, unsigned int gvl)
- void **vssbv\_uint8xm4** (unsigned char \*address, long stride, *uint8xm4\_t* a, unsigned int gvl)
- void **vssbv\_uint8xm8** (unsigned char \*address, long stride, *uint8xm8\_t* a, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      store_element(address, a[element])
      address = address + stride
```

**Masked prototypes:**

- void **vssbv\_mask\_int16xm1** (short \*address, long stride, *int16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int16xm2** (short \*address, long stride, *int16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int16xm4** (short \*address, long stride, *int16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int16xm8** (short \*address, long stride, *int16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int32xm1** (int \*address, long stride, *int32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int32xm2** (int \*address, long stride, *int32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int32xm4** (int \*address, long stride, *int32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int32xm8** (int \*address, long stride, *int32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int64xm1** (long \*address, long stride, *int64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int64xm2** (long \*address, long stride, *int64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int64xm4** (long \*address, long stride, *int64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int64xm8** (long \*address, long stride, *int64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)



- void **vssbv\_mask\_int8xm1** (signed char \*address, long stride, *int8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int8xm2** (signed char \*address, long stride, *int8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int8xm4** (signed char \*address, long stride, *int8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_int8xm8** (signed char \*address, long stride, *int8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint16xm1** (unsigned short \*address, long stride, *uint16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint16xm2** (unsigned short \*address, long stride, *uint16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint16xm4** (unsigned short \*address, long stride, *uint16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint16xm8** (unsigned short \*address, long stride, *uint16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint32xm1** (unsigned int \*address, long stride, *uint32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint32xm2** (unsigned int \*address, long stride, *uint32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint32xm4** (unsigned int \*address, long stride, *uint32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint32xm8** (unsigned int \*address, long stride, *uint32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint64xm1** (unsigned long \*address, long stride, *uint64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint64xm2** (unsigned long \*address, long stride, *uint64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint64xm4** (unsigned long \*address, long stride, *uint64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint64xm8** (unsigned long \*address, long stride, *uint64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint8xm1** (unsigned char \*address, long stride, *uint8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint8xm2** (unsigned char \*address, long stride, *uint8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint8xm4** (unsigned char \*address, long stride, *uint8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vssbv\_mask\_uint8xm8** (unsigned char \*address, long stride, *uint8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address, a[element])
        address = address + stride
```

## 2.9.26 Store elements in strided memory from vector

**Instruction:** [`'vsse.v'`]

**Prototypes:**

- void **vssev\_float16xm1** (float16\_t \*address, long stride, float16xm1\_t a, unsigned int gvl)
- void **vssev\_float16xm2** (float16\_t \*address, long stride, float16xm2\_t a, unsigned int gvl)
- void **vssev\_float16xm4** (float16\_t \*address, long stride, float16xm4\_t a, unsigned int gvl)
- void **vssev\_float16xm8** (float16\_t \*address, long stride, float16xm8\_t a, unsigned int gvl)
- void **vssev\_float32xm1** (float \*address, long stride, float32xm1\_t a, unsigned int gvl)
- void **vssev\_float32xm2** (float \*address, long stride, float32xm2\_t a, unsigned int gvl)
- void **vssev\_float32xm4** (float \*address, long stride, float32xm4\_t a, unsigned int gvl)
- void **vssev\_float32xm8** (float \*address, long stride, float32xm8\_t a, unsigned int gvl)
- void **vssev\_float64xm1** (double \*address, long stride, float64xm1\_t a, unsigned int gvl)
- void **vssev\_float64xm2** (double \*address, long stride, float64xm2\_t a, unsigned int gvl)
- void **vssev\_float64xm4** (double \*address, long stride, float64xm4\_t a, unsigned int gvl)
- void **vssev\_float64xm8** (double \*address, long stride, float64xm8\_t a, unsigned int gvl)
- void **vssev\_int16xm1** (short \*address, long stride, int16xm1\_t a, unsigned int gvl)
- void **vssev\_int16xm2** (short \*address, long stride, int16xm2\_t a, unsigned int gvl)
- void **vssev\_int16xm4** (short \*address, long stride, int16xm4\_t a, unsigned int gvl)
- void **vssev\_int16xm8** (short \*address, long stride, int16xm8\_t a, unsigned int gvl)
- void **vssev\_int32xm1** (int \*address, long stride, int32xm1\_t a, unsigned int gvl)
- void **vssev\_int32xm2** (int \*address, long stride, int32xm2\_t a, unsigned int gvl)
- void **vssev\_int32xm4** (int \*address, long stride, int32xm4\_t a, unsigned int gvl)
- void **vssev\_int32xm8** (int \*address, long stride, int32xm8\_t a, unsigned int gvl)
- void **vssev\_int64xm1** (long \*address, long stride, int64xm1\_t a, unsigned int gvl)
- void **vssev\_int64xm2** (long \*address, long stride, int64xm2\_t a, unsigned int gvl)
- void **vssev\_int64xm4** (long \*address, long stride, int64xm4\_t a, unsigned int gvl)
- void **vssev\_int64xm8** (long \*address, long stride, int64xm8\_t a, unsigned int gvl)
- void **vssev\_int8xm1** (signed char \*address, long stride, int8xm1\_t a, unsigned int gvl)
- void **vssev\_int8xm2** (signed char \*address, long stride, int8xm2\_t a, unsigned int gvl)
- void **vssev\_int8xm4** (signed char \*address, long stride, int8xm4\_t a, unsigned int gvl)
- void **vssev\_int8xm8** (signed char \*address, long stride, int8xm8\_t a, unsigned int gvl)
- void **vssev\_uint16xm1** (unsigned short \*address, long stride, uint16xm1\_t a, unsigned int gvl)
- void **vssev\_uint16xm2** (unsigned short \*address, long stride, uint16xm2\_t a, unsigned int gvl)
- void **vssev\_uint16xm4** (unsigned short \*address, long stride, uint16xm4\_t a, unsigned int gvl)
- void **vssev\_uint16xm8** (unsigned short \*address, long stride, uint16xm8\_t a, unsigned int gvl)
- void **vssev\_uint32xm1** (unsigned int \*address, long stride, uint32xm1\_t a, unsigned int gvl)

- void **vssev\_uint32xm2** (unsigned int \**address*, long *stride*, *uint32xm2\_t* *a*, unsigned int *gvl*)
- void **vssev\_uint32xm4** (unsigned int \**address*, long *stride*, *uint32xm4\_t* *a*, unsigned int *gvl*)
- void **vssev\_uint32xm8** (unsigned int \**address*, long *stride*, *uint32xm8\_t* *a*, unsigned int *gvl*)
- void **vssev\_uint64xm1** (unsigned long \**address*, long *stride*, *uint64xm1\_t* *a*, unsigned int *gvl*)
- void **vssev\_uint64xm2** (unsigned long \**address*, long *stride*, *uint64xm2\_t* *a*, unsigned int *gvl*)
- void **vssev\_uint64xm4** (unsigned long \**address*, long *stride*, *uint64xm4\_t* *a*, unsigned int *gvl*)
- void **vssev\_uint64xm8** (unsigned long \**address*, long *stride*, *uint64xm8\_t* *a*, unsigned int *gvl*)
- void **vssev\_uint8xm1** (unsigned char \**address*, long *stride*, *uint8xm1\_t* *a*, unsigned int *gvl*)
- void **vssev\_uint8xm2** (unsigned char \**address*, long *stride*, *uint8xm2\_t* *a*, unsigned int *gvl*)
- void **vssev\_uint8xm4** (unsigned char \**address*, long *stride*, *uint8xm4\_t* *a*, unsigned int *gvl*)
- void **vssev\_uint8xm8** (unsigned char \**address*, long *stride*, *uint8xm8\_t* *a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    store_element(address, a[element])
    address = address + stride
```

**Masked prototypes:**

- void **vssev\_mask\_float16xm1** (float16\_t \**address*, long *stride*, *float16xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssev\_mask\_float16xm2** (float16\_t \**address*, long *stride*, *float16xm2\_t* *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssev\_mask\_float16xm4** (float16\_t \**address*, long *stride*, *float16xm4\_t* *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vssev\_mask\_float16xm8** (float16\_t \**address*, long *stride*, *float16xm8\_t* *a*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- void **vssev\_mask\_float32xm1** (float \**address*, long *stride*, *float32xm1\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssev\_mask\_float32xm2** (float \**address*, long *stride*, *float32xm2\_t* *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssev\_mask\_float32xm4** (float \**address*, long *stride*, *float32xm4\_t* *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vssev\_mask\_float32xm8** (float \**address*, long *stride*, *float32xm8\_t* *a*, *e32xm8\_t* *mask*, unsigned int *gvl*)
- void **vssev\_mask\_float64xm1** (double \**address*, long *stride*, *float64xm1\_t* *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssev\_mask\_float64xm2** (double \**address*, long *stride*, *float64xm2\_t* *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssev\_mask\_float64xm4** (double \**address*, long *stride*, *float64xm4\_t* *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- void **vssev\_mask\_float64xm8** (double \**address*, long *stride*, *float64xm8\_t* *a*, *e64xm8\_t* *mask*, unsigned int *gvl*)
- void **vssev\_mask\_int16xm1** (short \**address*, long *stride*, *int16xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)

- void **vssev\_mask\_int16xm2** (short *\*address*, long *stride*, *int16xm2\_t a*, *e16xm2\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int16xm4** (short *\*address*, long *stride*, *int16xm4\_t a*, *e16xm4\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int16xm8** (short *\*address*, long *stride*, *int16xm8\_t a*, *e16xm8\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int32xm1** (int *\*address*, long *stride*, *int32xm1\_t a*, *e32xm1\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int32xm2** (int *\*address*, long *stride*, *int32xm2\_t a*, *e32xm2\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int32xm4** (int *\*address*, long *stride*, *int32xm4\_t a*, *e32xm4\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int32xm8** (int *\*address*, long *stride*, *int32xm8\_t a*, *e32xm8\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int64xm1** (long *\*address*, long *stride*, *int64xm1\_t a*, *e64xm1\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int64xm2** (long *\*address*, long *stride*, *int64xm2\_t a*, *e64xm2\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int64xm4** (long *\*address*, long *stride*, *int64xm4\_t a*, *e64xm4\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int64xm8** (long *\*address*, long *stride*, *int64xm8\_t a*, *e64xm8\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int8xm1** (signed char *\*address*, long *stride*, *int8xm1\_t a*, *e8xm1\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int8xm2** (signed char *\*address*, long *stride*, *int8xm2\_t a*, *e8xm2\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int8xm4** (signed char *\*address*, long *stride*, *int8xm4\_t a*, *e8xm4\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_int8xm8** (signed char *\*address*, long *stride*, *int8xm8\_t a*, *e8xm8\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint16xm1** (unsigned short *\*address*, long *stride*, *uint16xm1\_t a*, *e16xm1\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint16xm2** (unsigned short *\*address*, long *stride*, *uint16xm2\_t a*, *e16xm2\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint16xm4** (unsigned short *\*address*, long *stride*, *uint16xm4\_t a*, *e16xm4\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint16xm8** (unsigned short *\*address*, long *stride*, *uint16xm8\_t a*, *e16xm8\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint32xm1** (unsigned int *\*address*, long *stride*, *uint32xm1\_t a*, *e32xm1\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint32xm2** (unsigned int *\*address*, long *stride*, *uint32xm2\_t a*, *e32xm2\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint32xm4** (unsigned int *\*address*, long *stride*, *uint32xm4\_t a*, *e32xm4\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint32xm8** (unsigned int *\*address*, long *stride*, *uint32xm8\_t a*, *e32xm8\_t mask*, unsigned int *gvl*)

- void **vssev\_mask\_uint64xm1** (unsigned long *\*address*, long *stride*, *uint64xm1\_t a*, *e64xm1\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint64xm2** (unsigned long *\*address*, long *stride*, *uint64xm2\_t a*, *e64xm2\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint64xm4** (unsigned long *\*address*, long *stride*, *uint64xm4\_t a*, *e64xm4\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint64xm8** (unsigned long *\*address*, long *stride*, *uint64xm8\_t a*, *e64xm8\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint8xm1** (unsigned char *\*address*, long *stride*, *uint8xm1\_t a*, *e8xm1\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint8xm2** (unsigned char *\*address*, long *stride*, *uint8xm2\_t a*, *e8xm2\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint8xm4** (unsigned char *\*address*, long *stride*, *uint8xm4\_t a*, *e8xm4\_t mask*, unsigned int *gvl*)
- void **vssev\_mask\_uint8xm8** (unsigned char *\*address*, long *stride*, *uint8xm8\_t a*, *e8xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address, a[element])
        address = address + stride
```

## 2.9.27 Store 16b in strided memory from vector

**Instruction:** [*'vssh.v'*]

**Prototypes:**

- void **vsshv\_int16xm1** (short *\*address*, long *stride*, *int16xm1\_t a*, unsigned int *gvl*)
- void **vsshv\_int16xm2** (short *\*address*, long *stride*, *int16xm2\_t a*, unsigned int *gvl*)
- void **vsshv\_int16xm4** (short *\*address*, long *stride*, *int16xm4\_t a*, unsigned int *gvl*)
- void **vsshv\_int16xm8** (short *\*address*, long *stride*, *int16xm8\_t a*, unsigned int *gvl*)
- void **vsshv\_int32xm1** (int *\*address*, long *stride*, *int32xm1\_t a*, unsigned int *gvl*)
- void **vsshv\_int32xm2** (int *\*address*, long *stride*, *int32xm2\_t a*, unsigned int *gvl*)
- void **vsshv\_int32xm4** (int *\*address*, long *stride*, *int32xm4\_t a*, unsigned int *gvl*)
- void **vsshv\_int32xm8** (int *\*address*, long *stride*, *int32xm8\_t a*, unsigned int *gvl*)
- void **vsshv\_int64xm1** (long *\*address*, long *stride*, *int64xm1\_t a*, unsigned int *gvl*)
- void **vsshv\_int64xm2** (long *\*address*, long *stride*, *int64xm2\_t a*, unsigned int *gvl*)
- void **vsshv\_int64xm4** (long *\*address*, long *stride*, *int64xm4\_t a*, unsigned int *gvl*)
- void **vsshv\_int64xm8** (long *\*address*, long *stride*, *int64xm8\_t a*, unsigned int *gvl*)
- void **vsshv\_int8xm1** (signed char *\*address*, long *stride*, *int8xm1\_t a*, unsigned int *gvl*)
- void **vsshv\_int8xm2** (signed char *\*address*, long *stride*, *int8xm2\_t a*, unsigned int *gvl*)
- void **vsshv\_int8xm4** (signed char *\*address*, long *stride*, *int8xm4\_t a*, unsigned int *gvl*)



- void **vsshv\_int8xm8** (signed char \*address, long stride, *int8xm8\_t a*, unsigned int gvl)
- void **vsshv\_uint16xm1** (unsigned short \*address, long stride, *uint16xm1\_t a*, unsigned int gvl)
- void **vsshv\_uint16xm2** (unsigned short \*address, long stride, *uint16xm2\_t a*, unsigned int gvl)
- void **vsshv\_uint16xm4** (unsigned short \*address, long stride, *uint16xm4\_t a*, unsigned int gvl)
- void **vsshv\_uint16xm8** (unsigned short \*address, long stride, *uint16xm8\_t a*, unsigned int gvl)
- void **vsshv\_uint32xm1** (unsigned int \*address, long stride, *uint32xm1\_t a*, unsigned int gvl)
- void **vsshv\_uint32xm2** (unsigned int \*address, long stride, *uint32xm2\_t a*, unsigned int gvl)
- void **vsshv\_uint32xm4** (unsigned int \*address, long stride, *uint32xm4\_t a*, unsigned int gvl)
- void **vsshv\_uint32xm8** (unsigned int \*address, long stride, *uint32xm8\_t a*, unsigned int gvl)
- void **vsshv\_uint64xm1** (unsigned long \*address, long stride, *uint64xm1\_t a*, unsigned int gvl)
- void **vsshv\_uint64xm2** (unsigned long \*address, long stride, *uint64xm2\_t a*, unsigned int gvl)
- void **vsshv\_uint64xm4** (unsigned long \*address, long stride, *uint64xm4\_t a*, unsigned int gvl)
- void **vsshv\_uint64xm8** (unsigned long \*address, long stride, *uint64xm8\_t a*, unsigned int gvl)
- void **vsshv\_uint8xm1** (unsigned char \*address, long stride, *uint8xm1\_t a*, unsigned int gvl)
- void **vsshv\_uint8xm2** (unsigned char \*address, long stride, *uint8xm2\_t a*, unsigned int gvl)
- void **vsshv\_uint8xm4** (unsigned char \*address, long stride, *uint8xm4\_t a*, unsigned int gvl)
- void **vsshv\_uint8xm8** (unsigned char \*address, long stride, *uint8xm8\_t a*, unsigned int gvl)

#### Operation:

```
>>> for element = 0 to gvl - 1
    store_element(address, a[element])
    address = address + stride
```

#### Masked prototypes:

- void **vsshv\_mask\_int16xm1** (short \*address, long stride, *int16xm1\_t a*, *e16xm1\_t mask*, unsigned int gvl)
- void **vsshv\_mask\_int16xm2** (short \*address, long stride, *int16xm2\_t a*, *e16xm2\_t mask*, unsigned int gvl)
- void **vsshv\_mask\_int16xm4** (short \*address, long stride, *int16xm4\_t a*, *e16xm4\_t mask*, unsigned int gvl)
- void **vsshv\_mask\_int16xm8** (short \*address, long stride, *int16xm8\_t a*, *e16xm8\_t mask*, unsigned int gvl)
- void **vsshv\_mask\_int32xm1** (int \*address, long stride, *int32xm1\_t a*, *e32xm1\_t mask*, unsigned int gvl)
- void **vsshv\_mask\_int32xm2** (int \*address, long stride, *int32xm2\_t a*, *e32xm2\_t mask*, unsigned int gvl)
- void **vsshv\_mask\_int32xm4** (int \*address, long stride, *int32xm4\_t a*, *e32xm4\_t mask*, unsigned int gvl)
- void **vsshv\_mask\_int32xm8** (int \*address, long stride, *int32xm8\_t a*, *e32xm8\_t mask*, unsigned int gvl)
- void **vsshv\_mask\_int64xm1** (long \*address, long stride, *int64xm1\_t a*, *e64xm1\_t mask*, unsigned int gvl)



- void **vsshv\_mask\_int64xm2** (long *\*address*, long *stride*, *int64xm2\_t a*, *e64xm2\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_int64xm4** (long *\*address*, long *stride*, *int64xm4\_t a*, *e64xm4\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_int64xm8** (long *\*address*, long *stride*, *int64xm8\_t a*, *e64xm8\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_int8xm1** (signed char *\*address*, long *stride*, *int8xm1\_t a*, *e8xm1\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_int8xm2** (signed char *\*address*, long *stride*, *int8xm2\_t a*, *e8xm2\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_int8xm4** (signed char *\*address*, long *stride*, *int8xm4\_t a*, *e8xm4\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_int8xm8** (signed char *\*address*, long *stride*, *int8xm8\_t a*, *e8xm8\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint16xm1** (unsigned short *\*address*, long *stride*, *uint16xm1\_t a*, *e16xm1\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint16xm2** (unsigned short *\*address*, long *stride*, *uint16xm2\_t a*, *e16xm2\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint16xm4** (unsigned short *\*address*, long *stride*, *uint16xm4\_t a*, *e16xm4\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint16xm8** (unsigned short *\*address*, long *stride*, *uint16xm8\_t a*, *e16xm8\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint32xm1** (unsigned int *\*address*, long *stride*, *uint32xm1\_t a*, *e32xm1\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint32xm2** (unsigned int *\*address*, long *stride*, *uint32xm2\_t a*, *e32xm2\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint32xm4** (unsigned int *\*address*, long *stride*, *uint32xm4\_t a*, *e32xm4\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint32xm8** (unsigned int *\*address*, long *stride*, *uint32xm8\_t a*, *e32xm8\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint64xm1** (unsigned long *\*address*, long *stride*, *uint64xm1\_t a*, *e64xm1\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint64xm2** (unsigned long *\*address*, long *stride*, *uint64xm2\_t a*, *e64xm2\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint64xm4** (unsigned long *\*address*, long *stride*, *uint64xm4\_t a*, *e64xm4\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint64xm8** (unsigned long *\*address*, long *stride*, *uint64xm8\_t a*, *e64xm8\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint8xm1** (unsigned char *\*address*, long *stride*, *uint8xm1\_t a*, *e8xm1\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint8xm2** (unsigned char *\*address*, long *stride*, *uint8xm2\_t a*, *e8xm2\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint8xm4** (unsigned char *\*address*, long *stride*, *uint8xm4\_t a*, *e8xm4\_t mask*, unsigned int *gvl*)
- void **vsshv\_mask\_uint8xm8** (unsigned char *\*address*, long *stride*, *uint8xm8\_t a*, *e8xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address, a[element])
        address = address + stride
```

**2.9.28 Store 32b in strided memory from vector****Instruction:** [`'vssw.v'`]**Prototypes:**

- void **vsswv\_int16xm1** (short \*address, long stride, *int16xm1\_t* a, unsigned int gvl)
- void **vsswv\_int16xm2** (short \*address, long stride, *int16xm2\_t* a, unsigned int gvl)
- void **vsswv\_int16xm4** (short \*address, long stride, *int16xm4\_t* a, unsigned int gvl)
- void **vsswv\_int16xm8** (short \*address, long stride, *int16xm8\_t* a, unsigned int gvl)
- void **vsswv\_int32xm1** (int \*address, long stride, *int32xm1\_t* a, unsigned int gvl)
- void **vsswv\_int32xm2** (int \*address, long stride, *int32xm2\_t* a, unsigned int gvl)
- void **vsswv\_int32xm4** (int \*address, long stride, *int32xm4\_t* a, unsigned int gvl)
- void **vsswv\_int32xm8** (int \*address, long stride, *int32xm8\_t* a, unsigned int gvl)
- void **vsswv\_int64xm1** (long \*address, long stride, *int64xm1\_t* a, unsigned int gvl)
- void **vsswv\_int64xm2** (long \*address, long stride, *int64xm2\_t* a, unsigned int gvl)
- void **vsswv\_int64xm4** (long \*address, long stride, *int64xm4\_t* a, unsigned int gvl)
- void **vsswv\_int64xm8** (long \*address, long stride, *int64xm8\_t* a, unsigned int gvl)
- void **vsswv\_int8xm1** (signed char \*address, long stride, *int8xm1\_t* a, unsigned int gvl)
- void **vsswv\_int8xm2** (signed char \*address, long stride, *int8xm2\_t* a, unsigned int gvl)
- void **vsswv\_int8xm4** (signed char \*address, long stride, *int8xm4\_t* a, unsigned int gvl)
- void **vsswv\_int8xm8** (signed char \*address, long stride, *int8xm8\_t* a, unsigned int gvl)
- void **vsswv\_uint16xm1** (unsigned short \*address, long stride, *uint16xm1\_t* a, unsigned int gvl)
- void **vsswv\_uint16xm2** (unsigned short \*address, long stride, *uint16xm2\_t* a, unsigned int gvl)
- void **vsswv\_uint16xm4** (unsigned short \*address, long stride, *uint16xm4\_t* a, unsigned int gvl)
- void **vsswv\_uint16xm8** (unsigned short \*address, long stride, *uint16xm8\_t* a, unsigned int gvl)
- void **vsswv\_uint32xm1** (unsigned int \*address, long stride, *uint32xm1\_t* a, unsigned int gvl)
- void **vsswv\_uint32xm2** (unsigned int \*address, long stride, *uint32xm2\_t* a, unsigned int gvl)
- void **vsswv\_uint32xm4** (unsigned int \*address, long stride, *uint32xm4\_t* a, unsigned int gvl)
- void **vsswv\_uint32xm8** (unsigned int \*address, long stride, *uint32xm8\_t* a, unsigned int gvl)
- void **vsswv\_uint64xm1** (unsigned long \*address, long stride, *uint64xm1\_t* a, unsigned int gvl)
- void **vsswv\_uint64xm2** (unsigned long \*address, long stride, *uint64xm2\_t* a, unsigned int gvl)
- void **vsswv\_uint64xm4** (unsigned long \*address, long stride, *uint64xm4\_t* a, unsigned int gvl)
- void **vsswv\_uint64xm8** (unsigned long \*address, long stride, *uint64xm8\_t* a, unsigned int gvl)

- void **vsswv\_uint8xm1** (unsigned char \**address*, long *stride*, *uint8xm1\_t* *a*, unsigned int *gvl*)
- void **vsswv\_uint8xm2** (unsigned char \**address*, long *stride*, *uint8xm2\_t* *a*, unsigned int *gvl*)
- void **vsswv\_uint8xm4** (unsigned char \**address*, long *stride*, *uint8xm4\_t* *a*, unsigned int *gvl*)
- void **vsswv\_uint8xm8** (unsigned char \**address*, long *stride*, *uint8xm8\_t* *a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    store_element(address, a[element])
    address = address + stride
```

**Masked prototypes:**

- void **vsswv\_mask\_int16xm1** (short \**address*, long *stride*, *int16xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int16xm2** (short \**address*, long *stride*, *int16xm2\_t* *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int16xm4** (short \**address*, long *stride*, *int16xm4\_t* *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int16xm8** (short \**address*, long *stride*, *int16xm8\_t* *a*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int32xm1** (int \**address*, long *stride*, *int32xm1\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int32xm2** (int \**address*, long *stride*, *int32xm2\_t* *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int32xm4** (int \**address*, long *stride*, *int32xm4\_t* *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int32xm8** (int \**address*, long *stride*, *int32xm8\_t* *a*, *e32xm8\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int64xm1** (long \**address*, long *stride*, *int64xm1\_t* *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int64xm2** (long \**address*, long *stride*, *int64xm2\_t* *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int64xm4** (long \**address*, long *stride*, *int64xm4\_t* *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int64xm8** (long \**address*, long *stride*, *int64xm8\_t* *a*, *e64xm8\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int8xm1** (signed char \**address*, long *stride*, *int8xm1\_t* *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int8xm2** (signed char \**address*, long *stride*, *int8xm2\_t* *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int8xm4** (signed char \**address*, long *stride*, *int8xm4\_t* *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_int8xm8** (signed char \**address*, long *stride*, *int8xm8\_t* *a*, *e8xm8\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint16xm1** (unsigned short \**address*, long *stride*, *uint16xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint16xm2** (unsigned short \**address*, long *stride*, *uint16xm2\_t* *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)

- void **vsswv\_mask\_uint16xm4** (unsigned short *\*address*, long *stride*, *uint16xm4\_t a*, *e16xm4\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint16xm8** (unsigned short *\*address*, long *stride*, *uint16xm8\_t a*, *e16xm8\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint32xm1** (unsigned int *\*address*, long *stride*, *uint32xm1\_t a*, *e32xm1\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint32xm2** (unsigned int *\*address*, long *stride*, *uint32xm2\_t a*, *e32xm2\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint32xm4** (unsigned int *\*address*, long *stride*, *uint32xm4\_t a*, *e32xm4\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint32xm8** (unsigned int *\*address*, long *stride*, *uint32xm8\_t a*, *e32xm8\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint64xm1** (unsigned long *\*address*, long *stride*, *uint64xm1\_t a*, *e64xm1\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint64xm2** (unsigned long *\*address*, long *stride*, *uint64xm2\_t a*, *e64xm2\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint64xm4** (unsigned long *\*address*, long *stride*, *uint64xm4\_t a*, *e64xm4\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint64xm8** (unsigned long *\*address*, long *stride*, *uint64xm8\_t a*, *e64xm8\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint8xm1** (unsigned char *\*address*, long *stride*, *uint8xm1\_t a*, *e8xm1\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint8xm2** (unsigned char *\*address*, long *stride*, *uint8xm2\_t a*, *e8xm2\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint8xm4** (unsigned char *\*address*, long *stride*, *uint8xm4\_t a*, *e8xm4\_t mask*, unsigned int *gvl*)
- void **vsswv\_mask\_uint8xm8** (unsigned char *\*address*, long *stride*, *uint8xm8\_t a*, *e8xm8\_t mask*, unsigned int *gvl*)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address, a[element])
        address = address + stride
```

### 2.9.29 Store 8b in unordered-indexed memory from vector

**Instruction:** [*'vsuxb.v'*]

#### Prototypes:

- void **vsuxbv\_int16xm1** (short *\*address*, *int16xm1\_t index*, *int16xm1\_t a*, unsigned int *gvl*)
- void **vsuxbv\_int16xm2** (short *\*address*, *int16xm2\_t index*, *int16xm2\_t a*, unsigned int *gvl*)
- void **vsuxbv\_int16xm4** (short *\*address*, *int16xm4\_t index*, *int16xm4\_t a*, unsigned int *gvl*)
- void **vsuxbv\_int16xm8** (short *\*address*, *int16xm8\_t index*, *int16xm8\_t a*, unsigned int *gvl*)
- void **vsuxbv\_int32xm1** (int *\*address*, *int32xm1\_t index*, *int32xm1\_t a*, unsigned int *gvl*)
- void **vsuxbv\_int32xm2** (int *\*address*, *int32xm2\_t index*, *int32xm2\_t a*, unsigned int *gvl*)

- void **vsuxbv\_int32xm4** (int \*address, *int32xm4\_t* index, *int32xm4\_t* a, unsigned int gvl)
- void **vsuxbv\_int32xm8** (int \*address, *int32xm8\_t* index, *int32xm8\_t* a, unsigned int gvl)
- void **vsuxbv\_int64xm1** (long \*address, *int64xm1\_t* index, *int64xm1\_t* a, unsigned int gvl)
- void **vsuxbv\_int64xm2** (long \*address, *int64xm2\_t* index, *int64xm2\_t* a, unsigned int gvl)
- void **vsuxbv\_int64xm4** (long \*address, *int64xm4\_t* index, *int64xm4\_t* a, unsigned int gvl)
- void **vsuxbv\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, unsigned int gvl)
- void **vsuxbv\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, unsigned int gvl)
- void **vsuxbv\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, unsigned int gvl)
- void **vsuxbv\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, unsigned int gvl)
- void **vsuxbv\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, unsigned int gvl)
- void **vsuxbv\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, unsigned int gvl)
- void **vsuxbv\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, unsigned int gvl)
- void **vsuxbv\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, unsigned int gvl)
- void **vsuxbv\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, unsigned int gvl)
- void **vsuxbv\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, unsigned int gvl)
- void **vsuxbv\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, unsigned int gvl)
- void **vsuxbv\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, unsigned int gvl)
- void **vsuxbv\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, unsigned int gvl)
- void **vsuxbv\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, unsigned int gvl)
- void **vsuxbv\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, unsigned int gvl)
- void **vsuxbv\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, unsigned int gvl)
- void **vsuxbv\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, unsigned int gvl)
- void **vsuxbv\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, unsigned int gvl)
- void **vsuxbv\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, unsigned int gvl)
- void **vsuxbv\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, unsigned int gvl)
- void **vsuxbv\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      store_element(address + index[element], a[element])
```

**Masked prototypes:**

- void **vsuxbv\_mask\_int16xm1** (short \*address, *int16xm1\_t* index, *int16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)



- void **vsuxbv\_mask\_int16xm2** (short \*address, *int16xm2\_t* index, *int16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int16xm4** (short \*address, *int16xm4\_t* index, *int16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int16xm8** (short \*address, *int16xm8\_t* index, *int16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int32xm1** (int \*address, *int32xm1\_t* index, *int32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int32xm2** (int \*address, *int32xm2\_t* index, *int32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int32xm4** (int \*address, *int32xm4\_t* index, *int32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int32xm8** (int \*address, *int32xm8\_t* index, *int32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int64xm1** (long \*address, *int64xm1\_t* index, *int64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int64xm2** (long \*address, *int64xm2\_t* index, *int64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int64xm4** (long \*address, *int64xm4\_t* index, *int64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)



- void **vsuxbv\_mask\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsuxbv\_mask\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address + index[element], a[element])
```

### 2.9.30 Store element in unordered-indexed memory from vector

Instruction: ['vsuxe.v']

Prototypes:

- void **vsuxev\_float16xm1** (float16\_t \*address, *float16xm1\_t* index, *float16xm1\_t* a, unsigned int gvl)
- void **vsuxev\_float16xm2** (float16\_t \*address, *float16xm2\_t* index, *float16xm2\_t* a, unsigned int gvl)
- void **vsuxev\_float16xm4** (float16\_t \*address, *float16xm4\_t* index, *float16xm4\_t* a, unsigned int gvl)
- void **vsuxev\_float16xm8** (float16\_t \*address, *float16xm8\_t* index, *float16xm8\_t* a, unsigned int gvl)
- void **vsuxev\_float32xm1** (float \*address, *float32xm1\_t* index, *float32xm1\_t* a, unsigned int gvl)
- void **vsuxev\_float32xm2** (float \*address, *float32xm2\_t* index, *float32xm2\_t* a, unsigned int gvl)
- void **vsuxev\_float32xm4** (float \*address, *float32xm4\_t* index, *float32xm4\_t* a, unsigned int gvl)
- void **vsuxev\_float32xm8** (float \*address, *float32xm8\_t* index, *float32xm8\_t* a, unsigned int gvl)
- void **vsuxev\_float64xm1** (double \*address, *float64xm1\_t* index, *float64xm1\_t* a, unsigned int gvl)
- void **vsuxev\_float64xm2** (double \*address, *float64xm2\_t* index, *float64xm2\_t* a, unsigned int gvl)
- void **vsuxev\_float64xm4** (double \*address, *float64xm4\_t* index, *float64xm4\_t* a, unsigned int gvl)
- void **vsuxev\_float64xm8** (double \*address, *float64xm8\_t* index, *float64xm8\_t* a, unsigned int gvl)
- void **vsuxev\_int16xm1** (short \*address, *int16xm1\_t* index, *int16xm1\_t* a, unsigned int gvl)
- void **vsuxev\_int16xm2** (short \*address, *int16xm2\_t* index, *int16xm2\_t* a, unsigned int gvl)
- void **vsuxev\_int16xm4** (short \*address, *int16xm4\_t* index, *int16xm4\_t* a, unsigned int gvl)
- void **vsuxev\_int16xm8** (short \*address, *int16xm8\_t* index, *int16xm8\_t* a, unsigned int gvl)

- void **vsuxev\_int32xm1** (int \*address, *int32xm1\_t* index, *int32xm1\_t* a, unsigned int gvl)
- void **vsuxev\_int32xm2** (int \*address, *int32xm2\_t* index, *int32xm2\_t* a, unsigned int gvl)
- void **vsuxev\_int32xm4** (int \*address, *int32xm4\_t* index, *int32xm4\_t* a, unsigned int gvl)
- void **vsuxev\_int32xm8** (int \*address, *int32xm8\_t* index, *int32xm8\_t* a, unsigned int gvl)
- void **vsuxev\_int64xm1** (long \*address, *int64xm1\_t* index, *int64xm1\_t* a, unsigned int gvl)
- void **vsuxev\_int64xm2** (long \*address, *int64xm2\_t* index, *int64xm2\_t* a, unsigned int gvl)
- void **vsuxev\_int64xm4** (long \*address, *int64xm4\_t* index, *int64xm4\_t* a, unsigned int gvl)
- void **vsuxev\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, unsigned int gvl)
- void **vsuxev\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, unsigned int gvl)
- void **vsuxev\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, unsigned int gvl)
- void **vsuxev\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, unsigned int gvl)
- void **vsuxev\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, unsigned int gvl)
- void **vsuxev\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, unsigned int gvl)
- void **vsuxev\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, unsigned int gvl)
- void **vsuxev\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, unsigned int gvl)
- void **vsuxev\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, unsigned int gvl)
- void **vsuxev\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, unsigned int gvl)
- void **vsuxev\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, unsigned int gvl)
- void **vsuxev\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, unsigned int gvl)
- void **vsuxev\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, unsigned int gvl)
- void **vsuxev\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, unsigned int gvl)
- void **vsuxev\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, unsigned int gvl)
- void **vsuxev\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, unsigned int gvl)
- void **vsuxev\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, unsigned int gvl)
- void **vsuxev\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, unsigned int gvl)
- void **vsuxev\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, unsigned int gvl)
- void **vsuxev\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, unsigned int gvl)
- void **vsuxev\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      store_element(address + index[element], a[element])
```

**Masked prototypes:**

- void **vsuxev\_mask\_float16xm1** (float16\_t \*address, float16xm1\_t index, float16xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_float16xm2** (float16\_t \*address, float16xm2\_t index, float16xm2\_t a, e16xm2\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_float16xm4** (float16\_t \*address, float16xm4\_t index, float16xm4\_t a, e16xm4\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_float16xm8** (float16\_t \*address, float16xm8\_t index, float16xm8\_t a, e16xm8\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_float32xm1** (float \*address, float32xm1\_t index, float32xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_float32xm2** (float \*address, float32xm2\_t index, float32xm2\_t a, e32xm2\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_float32xm4** (float \*address, float32xm4\_t index, float32xm4\_t a, e32xm4\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_float32xm8** (float \*address, float32xm8\_t index, float32xm8\_t a, e32xm8\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_float64xm1** (double \*address, float64xm1\_t index, float64xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_float64xm2** (double \*address, float64xm2\_t index, float64xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_float64xm4** (double \*address, float64xm4\_t index, float64xm4\_t a, e64xm4\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_float64xm8** (double \*address, float64xm8\_t index, float64xm8\_t a, e64xm8\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_int16xm1** (short \*address, int16xm1\_t index, int16xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_int16xm2** (short \*address, int16xm2\_t index, int16xm2\_t a, e16xm2\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_int16xm4** (short \*address, int16xm4\_t index, int16xm4\_t a, e16xm4\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_int16xm8** (short \*address, int16xm8\_t index, int16xm8\_t a, e16xm8\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_int32xm1** (int \*address, int32xm1\_t index, int32xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_int32xm2** (int \*address, int32xm2\_t index, int32xm2\_t a, e32xm2\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_int32xm4** (int \*address, int32xm4\_t index, int32xm4\_t a, e32xm4\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_int32xm8** (int \*address, int32xm8\_t index, int32xm8\_t a, e32xm8\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_int64xm1** (long \*address, int64xm1\_t index, int64xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_int64xm2** (long \*address, int64xm2\_t index, int64xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsuxev\_mask\_int64xm4** (long \*address, int64xm4\_t index, int64xm4\_t a, e64xm4\_t mask, unsigned int gvl)

- void **vsuxev\_mask\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsuxev\_mask\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address + index[element], a[element])
```

## 2.9.31 Store 16b in unordered-indexed memory from vector

**Instruction:** [`'vsuxh.v'`]

**Prototypes:**

- void **vsuxhv\_int16xm1** (short *\*address*, *int16xm1\_t* index, *int16xm1\_t* a, unsigned int gvl)
- void **vsuxhv\_int16xm2** (short *\*address*, *int16xm2\_t* index, *int16xm2\_t* a, unsigned int gvl)
- void **vsuxhv\_int16xm4** (short *\*address*, *int16xm4\_t* index, *int16xm4\_t* a, unsigned int gvl)
- void **vsuxhv\_int16xm8** (short *\*address*, *int16xm8\_t* index, *int16xm8\_t* a, unsigned int gvl)
- void **vsuxhv\_int32xm1** (int *\*address*, *int32xm1\_t* index, *int32xm1\_t* a, unsigned int gvl)
- void **vsuxhv\_int32xm2** (int *\*address*, *int32xm2\_t* index, *int32xm2\_t* a, unsigned int gvl)
- void **vsuxhv\_int32xm4** (int *\*address*, *int32xm4\_t* index, *int32xm4\_t* a, unsigned int gvl)
- void **vsuxhv\_int32xm8** (int *\*address*, *int32xm8\_t* index, *int32xm8\_t* a, unsigned int gvl)
- void **vsuxhv\_int64xm1** (long *\*address*, *int64xm1\_t* index, *int64xm1\_t* a, unsigned int gvl)
- void **vsuxhv\_int64xm2** (long *\*address*, *int64xm2\_t* index, *int64xm2\_t* a, unsigned int gvl)
- void **vsuxhv\_int64xm4** (long *\*address*, *int64xm4\_t* index, *int64xm4\_t* a, unsigned int gvl)
- void **vsuxhv\_int64xm8** (long *\*address*, *int64xm8\_t* index, *int64xm8\_t* a, unsigned int gvl)
- void **vsuxhv\_int8xm1** (signed char *\*address*, *int8xm1\_t* index, *int8xm1\_t* a, unsigned int gvl)
- void **vsuxhv\_int8xm2** (signed char *\*address*, *int8xm2\_t* index, *int8xm2\_t* a, unsigned int gvl)
- void **vsuxhv\_int8xm4** (signed char *\*address*, *int8xm4\_t* index, *int8xm4\_t* a, unsigned int gvl)
- void **vsuxhv\_int8xm8** (signed char *\*address*, *int8xm8\_t* index, *int8xm8\_t* a, unsigned int gvl)
- void **vsuxhv\_uint16xm1** (unsigned short *\*address*, *uint16xm1\_t* index, *uint16xm1\_t* a, unsigned int gvl)
- void **vsuxhv\_uint16xm2** (unsigned short *\*address*, *uint16xm2\_t* index, *uint16xm2\_t* a, unsigned int gvl)
- void **vsuxhv\_uint16xm4** (unsigned short *\*address*, *uint16xm4\_t* index, *uint16xm4\_t* a, unsigned int gvl)
- void **vsuxhv\_uint16xm8** (unsigned short *\*address*, *uint16xm8\_t* index, *uint16xm8\_t* a, unsigned int gvl)
- void **vsuxhv\_uint32xm1** (unsigned int *\*address*, *uint32xm1\_t* index, *uint32xm1\_t* a, unsigned int gvl)
- void **vsuxhv\_uint32xm2** (unsigned int *\*address*, *uint32xm2\_t* index, *uint32xm2\_t* a, unsigned int gvl)
- void **vsuxhv\_uint32xm4** (unsigned int *\*address*, *uint32xm4\_t* index, *uint32xm4\_t* a, unsigned int gvl)
- void **vsuxhv\_uint32xm8** (unsigned int *\*address*, *uint32xm8\_t* index, *uint32xm8\_t* a, unsigned int gvl)
- void **vsuxhv\_uint64xm1** (unsigned long *\*address*, *uint64xm1\_t* index, *uint64xm1\_t* a, unsigned int gvl)
- void **vsuxhv\_uint64xm2** (unsigned long *\*address*, *uint64xm2\_t* index, *uint64xm2\_t* a, unsigned int gvl)
- void **vsuxhv\_uint64xm4** (unsigned long *\*address*, *uint64xm4\_t* index, *uint64xm4\_t* a, unsigned int gvl)
- void **vsuxhv\_uint64xm8** (unsigned long *\*address*, *uint64xm8\_t* index, *uint64xm8\_t* a, unsigned int gvl)



- void **vsuxhv\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, unsigned int gvl)
- void **vsuxhv\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, unsigned int gvl)
- void **vsuxhv\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, unsigned int gvl)
- void **vsuxhv\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      store_element(address + index[element], a[element])
```

**Masked prototypes:**

- void **vsuxhv\_mask\_int16xm1** (short \*address, *int16xm1\_t* index, *int16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int16xm2** (short \*address, *int16xm2\_t* index, *int16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int16xm4** (short \*address, *int16xm4\_t* index, *int16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int16xm8** (short \*address, *int16xm8\_t* index, *int16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int32xm1** (int \*address, *int32xm1\_t* index, *int32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int32xm2** (int \*address, *int32xm2\_t* index, *int32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int32xm4** (int \*address, *int32xm4\_t* index, *int32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int32xm8** (int \*address, *int32xm8\_t* index, *int32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int64xm1** (long \*address, *int64xm1\_t* index, *int64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int64xm2** (long \*address, *int64xm2\_t* index, *int64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int64xm4** (long \*address, *int64xm4\_t* index, *int64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)



- void **vsuxhv\_mask\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsuxhv\_mask\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address + index[element], a[element])
```

**2.9.32 Store 32b in unordered-indexed memory from vector****Instruction:** [*'vsuxw.v'*]**Prototypes:**

- void **vsuxwv\_int16xm1** (short \*address, *int16xm1\_t* index, *int16xm1\_t* a, unsigned int gvl)
- void **vsuxwv\_int16xm2** (short \*address, *int16xm2\_t* index, *int16xm2\_t* a, unsigned int gvl)
- void **vsuxwv\_int16xm4** (short \*address, *int16xm4\_t* index, *int16xm4\_t* a, unsigned int gvl)
- void **vsuxwv\_int16xm8** (short \*address, *int16xm8\_t* index, *int16xm8\_t* a, unsigned int gvl)
- void **vsuxwv\_int32xm1** (int \*address, *int32xm1\_t* index, *int32xm1\_t* a, unsigned int gvl)
- void **vsuxwv\_int32xm2** (int \*address, *int32xm2\_t* index, *int32xm2\_t* a, unsigned int gvl)

- void **vsuxwv\_int32xm4** (int \*address, *int32xm4\_t* index, *int32xm4\_t* a, unsigned int gvl)
- void **vsuxwv\_int32xm8** (int \*address, *int32xm8\_t* index, *int32xm8\_t* a, unsigned int gvl)
- void **vsuxwv\_int64xm1** (long \*address, *int64xm1\_t* index, *int64xm1\_t* a, unsigned int gvl)
- void **vsuxwv\_int64xm2** (long \*address, *int64xm2\_t* index, *int64xm2\_t* a, unsigned int gvl)
- void **vsuxwv\_int64xm4** (long \*address, *int64xm4\_t* index, *int64xm4\_t* a, unsigned int gvl)
- void **vsuxwv\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, unsigned int gvl)
- void **vsuxwv\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, unsigned int gvl)
- void **vsuxwv\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, unsigned int gvl)
- void **vsuxwv\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, unsigned int gvl)
- void **vsuxwv\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, unsigned int gvl)
- void **vsuxwv\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, unsigned int gvl)
- void **vsuxwv\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, unsigned int gvl)
- void **vsuxwv\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, unsigned int gvl)
- void **vsuxwv\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, unsigned int gvl)
- void **vsuxwv\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, unsigned int gvl)
- void **vsuxwv\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, unsigned int gvl)
- void **vsuxwv\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, unsigned int gvl)
- void **vsuxwv\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, unsigned int gvl)
- void **vsuxwv\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, unsigned int gvl)
- void **vsuxwv\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, unsigned int gvl)
- void **vsuxwv\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, unsigned int gvl)
- void **vsuxwv\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, unsigned int gvl)
- void **vsuxwv\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, unsigned int gvl)
- void **vsuxwv\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, unsigned int gvl)
- void **vsuxwv\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, unsigned int gvl)
- void **vsuxwv\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      store_element(address + index[element], a[element])
```

**Masked prototypes:**

- void **vsuxwv\_mask\_int16xm1** (short \*address, *int16xm1\_t* index, *int16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)

- void **vsuxwv\_mask\_int16xm2** (short \*address, *int16xm2\_t* index, *int16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int16xm4** (short \*address, *int16xm4\_t* index, *int16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int16xm8** (short \*address, *int16xm8\_t* index, *int16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int32xm1** (int \*address, *int32xm1\_t* index, *int32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int32xm2** (int \*address, *int32xm2\_t* index, *int32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int32xm4** (int \*address, *int32xm4\_t* index, *int32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int32xm8** (int \*address, *int32xm8\_t* index, *int32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int64xm1** (long \*address, *int64xm1\_t* index, *int64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int64xm2** (long \*address, *int64xm2\_t* index, *int64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int64xm4** (long \*address, *int64xm4\_t* index, *int64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)

- void **vsuxwv\_mask\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsuxwv\_mask\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address + index[element], a[element])
```

### 2.9.33 Store 32b in memory from vector

**Instruction:** [*vsw.v*']

#### Prototypes:

- void **vswv\_int16xm1** (short \*address, *int16xm1\_t* a, unsigned int gvl)
- void **vswv\_int16xm2** (short \*address, *int16xm2\_t* a, unsigned int gvl)
- void **vswv\_int16xm4** (short \*address, *int16xm4\_t* a, unsigned int gvl)
- void **vswv\_int16xm8** (short \*address, *int16xm8\_t* a, unsigned int gvl)
- void **vswv\_int32xm1** (int \*address, *int32xm1\_t* a, unsigned int gvl)
- void **vswv\_int32xm2** (int \*address, *int32xm2\_t* a, unsigned int gvl)
- void **vswv\_int32xm4** (int \*address, *int32xm4\_t* a, unsigned int gvl)
- void **vswv\_int32xm8** (int \*address, *int32xm8\_t* a, unsigned int gvl)
- void **vswv\_int64xm1** (long \*address, *int64xm1\_t* a, unsigned int gvl)
- void **vswv\_int64xm2** (long \*address, *int64xm2\_t* a, unsigned int gvl)
- void **vswv\_int64xm4** (long \*address, *int64xm4\_t* a, unsigned int gvl)
- void **vswv\_int64xm8** (long \*address, *int64xm8\_t* a, unsigned int gvl)
- void **vswv\_int8xm1** (signed char \*address, *int8xm1\_t* a, unsigned int gvl)
- void **vswv\_int8xm2** (signed char \*address, *int8xm2\_t* a, unsigned int gvl)
- void **vswv\_int8xm4** (signed char \*address, *int8xm4\_t* a, unsigned int gvl)
- void **vswv\_int8xm8** (signed char \*address, *int8xm8\_t* a, unsigned int gvl)

- void **vswv\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* a, unsigned int gvl)
- void **vswv\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* a, unsigned int gvl)
- void **vswv\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* a, unsigned int gvl)
- void **vswv\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* a, unsigned int gvl)
- void **vswv\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* a, unsigned int gvl)
- void **vswv\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* a, unsigned int gvl)
- void **vswv\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* a, unsigned int gvl)
- void **vswv\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* a, unsigned int gvl)
- void **vswv\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* a, unsigned int gvl)
- void **vswv\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* a, unsigned int gvl)
- void **vswv\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* a, unsigned int gvl)
- void **vswv\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* a, unsigned int gvl)
- void **vswv\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* a, unsigned int gvl)
- void **vswv\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* a, unsigned int gvl)
- void **vswv\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* a, unsigned int gvl)
- void **vswv\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* a, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      store_element(address, a[element])
      address = address + 1
```

**Masked prototypes:**

- void **vswv\_mask\_int16xm1** (short \*address, *int16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int16xm2** (short \*address, *int16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int16xm4** (short \*address, *int16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int16xm8** (short \*address, *int16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int32xm1** (int \*address, *int32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int32xm2** (int \*address, *int32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int32xm4** (int \*address, *int32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int32xm8** (int \*address, *int32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int64xm1** (long \*address, *int64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int64xm2** (long \*address, *int64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int64xm4** (long \*address, *int64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int64xm8** (long \*address, *int64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int8xm1** (signed char \*address, *int8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int8xm2** (signed char \*address, *int8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vswv\_mask\_int8xm4** (signed char \*address, *int8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)



- void **vswv\_mask\_int8xm8** (signed char \*address, *int8xm8\_t a*, *e8xm8\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint16xm1** (unsigned short \*address, *uint16xm1\_t a*, *e16xm1\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint16xm2** (unsigned short \*address, *uint16xm2\_t a*, *e16xm2\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint16xm4** (unsigned short \*address, *uint16xm4\_t a*, *e16xm4\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint16xm8** (unsigned short \*address, *uint16xm8\_t a*, *e16xm8\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint32xm1** (unsigned int \*address, *uint32xm1\_t a*, *e32xm1\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint32xm2** (unsigned int \*address, *uint32xm2\_t a*, *e32xm2\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint32xm4** (unsigned int \*address, *uint32xm4\_t a*, *e32xm4\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint32xm8** (unsigned int \*address, *uint32xm8\_t a*, *e32xm8\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint64xm1** (unsigned long \*address, *uint64xm1\_t a*, *e64xm1\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint64xm2** (unsigned long \*address, *uint64xm2\_t a*, *e64xm2\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint64xm4** (unsigned long \*address, *uint64xm4\_t a*, *e64xm4\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint64xm8** (unsigned long \*address, *uint64xm8\_t a*, *e64xm8\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint8xm1** (unsigned char \*address, *uint8xm1\_t a*, *e8xm1\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint8xm2** (unsigned char \*address, *uint8xm2\_t a*, *e8xm2\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint8xm4** (unsigned char \*address, *uint8xm4\_t a*, *e8xm4\_t mask*, unsigned int gvl)
- void **vswv\_mask\_uint8xm8** (unsigned char \*address, *uint8xm8\_t a*, *e8xm8\_t mask*, unsigned int gvl)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address, a[element])
        address = address + 1
```

### 2.9.34 Store 8b in ordered-indexed memory from vector

**Instruction:** [*'vsxb.v'*]

#### Prototypes:

- void **vsxbv\_int16xm1** (short \*address, *int16xm1\_t index*, *int16xm1\_t a*, unsigned int gvl)
- void **vsxbv\_int16xm2** (short \*address, *int16xm2\_t index*, *int16xm2\_t a*, unsigned int gvl)
- void **vsxbv\_int16xm4** (short \*address, *int16xm4\_t index*, *int16xm4\_t a*, unsigned int gvl)
- void **vsxbv\_int16xm8** (short \*address, *int16xm8\_t index*, *int16xm8\_t a*, unsigned int gvl)



- void **vsxbv\_int32xm1** (int \*address, *int32xm1\_t* index, *int32xm1\_t* a, unsigned int gvl)
- void **vsxbv\_int32xm2** (int \*address, *int32xm2\_t* index, *int32xm2\_t* a, unsigned int gvl)
- void **vsxbv\_int32xm4** (int \*address, *int32xm4\_t* index, *int32xm4\_t* a, unsigned int gvl)
- void **vsxbv\_int32xm8** (int \*address, *int32xm8\_t* index, *int32xm8\_t* a, unsigned int gvl)
- void **vsxbv\_int64xm1** (long \*address, *int64xm1\_t* index, *int64xm1\_t* a, unsigned int gvl)
- void **vsxbv\_int64xm2** (long \*address, *int64xm2\_t* index, *int64xm2\_t* a, unsigned int gvl)
- void **vsxbv\_int64xm4** (long \*address, *int64xm4\_t* index, *int64xm4\_t* a, unsigned int gvl)
- void **vsxbv\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, unsigned int gvl)
- void **vsxbv\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, unsigned int gvl)
- void **vsxbv\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, unsigned int gvl)
- void **vsxbv\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, unsigned int gvl)
- void **vsxbv\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, unsigned int gvl)
- void **vsxbv\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, unsigned int gvl)
- void **vsxbv\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, unsigned int gvl)
- void **vsxbv\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, unsigned int gvl)
- void **vsxbv\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, unsigned int gvl)
- void **vsxbv\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, unsigned int gvl)
- void **vsxbv\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, unsigned int gvl)
- void **vsxbv\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, unsigned int gvl)
- void **vsxbv\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, unsigned int gvl)
- void **vsxbv\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, unsigned int gvl)
- void **vsxbv\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, unsigned int gvl)
- void **vsxbv\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, unsigned int gvl)
- void **vsxbv\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, unsigned int gvl)
- void **vsxbv\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, unsigned int gvl)
- void **vsxbv\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, unsigned int gvl)
- void **vsxbv\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, unsigned int gvl)
- void **vsxbv\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      store_element(address + index[element], a[element])
```

**Masked prototypes:**

- void **vsxbv\_mask\_int16xm1** (short \*address, *int16xm1\_t* index, *int16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)

- void **vsxbv\_mask\_int16xm2** (short \*address, *int16xm2\_t* index, *int16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int16xm4** (short \*address, *int16xm4\_t* index, *int16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int16xm8** (short \*address, *int16xm8\_t* index, *int16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int32xm1** (int \*address, *int32xm1\_t* index, *int32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int32xm2** (int \*address, *int32xm2\_t* index, *int32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int32xm4** (int \*address, *int32xm4\_t* index, *int32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int32xm8** (int \*address, *int32xm8\_t* index, *int32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int64xm1** (long \*address, *int64xm1\_t* index, *int64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int64xm2** (long \*address, *int64xm2\_t* index, *int64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int64xm4** (long \*address, *int64xm4\_t* index, *int64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)

- void **vsxbv\_mask\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsxbv\_mask\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address + index[element], a[element])
```

## 2.9.35 Store element in ordered-indexed memory from vector

Instruction: [*vsxe.v*]

Prototypes:

- void **vsxev\_float16xm1** (float16\_t \*address, *float16xm1\_t* index, *float16xm1\_t* a, unsigned int gvl)
- void **vsxev\_float16xm2** (float16\_t \*address, *float16xm2\_t* index, *float16xm2\_t* a, unsigned int gvl)
- void **vsxev\_float16xm4** (float16\_t \*address, *float16xm4\_t* index, *float16xm4\_t* a, unsigned int gvl)
- void **vsxev\_float16xm8** (float16\_t \*address, *float16xm8\_t* index, *float16xm8\_t* a, unsigned int gvl)
- void **vsxev\_float32xm1** (float \*address, *float32xm1\_t* index, *float32xm1\_t* a, unsigned int gvl)
- void **vsxev\_float32xm2** (float \*address, *float32xm2\_t* index, *float32xm2\_t* a, unsigned int gvl)
- void **vsxev\_float32xm4** (float \*address, *float32xm4\_t* index, *float32xm4\_t* a, unsigned int gvl)
- void **vsxev\_float32xm8** (float \*address, *float32xm8\_t* index, *float32xm8\_t* a, unsigned int gvl)
- void **vsxev\_float64xm1** (double \*address, *float64xm1\_t* index, *float64xm1\_t* a, unsigned int gvl)
- void **vsxev\_float64xm2** (double \*address, *float64xm2\_t* index, *float64xm2\_t* a, unsigned int gvl)
- void **vsxev\_float64xm4** (double \*address, *float64xm4\_t* index, *float64xm4\_t* a, unsigned int gvl)
- void **vsxev\_float64xm8** (double \*address, *float64xm8\_t* index, *float64xm8\_t* a, unsigned int gvl)
- void **vsxev\_int16xm1** (short \*address, *int16xm1\_t* index, *int16xm1\_t* a, unsigned int gvl)
- void **vsxev\_int16xm2** (short \*address, *int16xm2\_t* index, *int16xm2\_t* a, unsigned int gvl)
- void **vsxev\_int16xm4** (short \*address, *int16xm4\_t* index, *int16xm4\_t* a, unsigned int gvl)
- void **vsxev\_int16xm8** (short \*address, *int16xm8\_t* index, *int16xm8\_t* a, unsigned int gvl)

- void **vsxev\_int32xm1** (int \*address, *int32xm1\_t* index, *int32xm1\_t* a, unsigned int gvl)
- void **vsxev\_int32xm2** (int \*address, *int32xm2\_t* index, *int32xm2\_t* a, unsigned int gvl)
- void **vsxev\_int32xm4** (int \*address, *int32xm4\_t* index, *int32xm4\_t* a, unsigned int gvl)
- void **vsxev\_int32xm8** (int \*address, *int32xm8\_t* index, *int32xm8\_t* a, unsigned int gvl)
- void **vsxev\_int64xm1** (long \*address, *int64xm1\_t* index, *int64xm1\_t* a, unsigned int gvl)
- void **vsxev\_int64xm2** (long \*address, *int64xm2\_t* index, *int64xm2\_t* a, unsigned int gvl)
- void **vsxev\_int64xm4** (long \*address, *int64xm4\_t* index, *int64xm4\_t* a, unsigned int gvl)
- void **vsxev\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, unsigned int gvl)
- void **vsxev\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, unsigned int gvl)
- void **vsxev\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, unsigned int gvl)
- void **vsxev\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, unsigned int gvl)
- void **vsxev\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, unsigned int gvl)
- void **vsxev\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, unsigned int gvl)
- void **vsxev\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, unsigned int gvl)
- void **vsxev\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, unsigned int gvl)
- void **vsxev\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, unsigned int gvl)
- void **vsxev\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, unsigned int gvl)
- void **vsxev\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, unsigned int gvl)
- void **vsxev\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, unsigned int gvl)
- void **vsxev\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, unsigned int gvl)
- void **vsxev\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, unsigned int gvl)
- void **vsxev\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, unsigned int gvl)
- void **vsxev\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, unsigned int gvl)
- void **vsxev\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, unsigned int gvl)
- void **vsxev\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, unsigned int gvl)
- void **vsxev\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, unsigned int gvl)
- void **vsxev\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, unsigned int gvl)
- void **vsxev\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, unsigned int gvl)

#### Operation:

```
>>> for element = 0 to gvl - 1
      store_element(address + index[element], a[element])
```

#### Masked prototypes:

- void **vsxev\_mask\_float16xm1** (float16\_t \*address, *float16xm1\_t* index, *float16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)

- void **vsxev\_mask\_float16xm2** (float16\_t \*address, float16xm2\_t index, float16xm2\_t a, e16xm2\_t mask, unsigned int gvl)
- void **vsxev\_mask\_float16xm4** (float16\_t \*address, float16xm4\_t index, float16xm4\_t a, e16xm4\_t mask, unsigned int gvl)
- void **vsxev\_mask\_float16xm8** (float16\_t \*address, float16xm8\_t index, float16xm8\_t a, e16xm8\_t mask, unsigned int gvl)
- void **vsxev\_mask\_float32xm1** (float \*address, float32xm1\_t index, float32xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsxev\_mask\_float32xm2** (float \*address, float32xm2\_t index, float32xm2\_t a, e32xm2\_t mask, unsigned int gvl)
- void **vsxev\_mask\_float32xm4** (float \*address, float32xm4\_t index, float32xm4\_t a, e32xm4\_t mask, unsigned int gvl)
- void **vsxev\_mask\_float32xm8** (float \*address, float32xm8\_t index, float32xm8\_t a, e32xm8\_t mask, unsigned int gvl)
- void **vsxev\_mask\_float64xm1** (double \*address, float64xm1\_t index, float64xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxev\_mask\_float64xm2** (double \*address, float64xm2\_t index, float64xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsxev\_mask\_float64xm4** (double \*address, float64xm4\_t index, float64xm4\_t a, e64xm4\_t mask, unsigned int gvl)
- void **vsxev\_mask\_float64xm8** (double \*address, float64xm8\_t index, float64xm8\_t a, e64xm8\_t mask, unsigned int gvl)
- void **vsxev\_mask\_int16xm1** (short \*address, int16xm1\_t index, int16xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsxev\_mask\_int16xm2** (short \*address, int16xm2\_t index, int16xm2\_t a, e16xm2\_t mask, unsigned int gvl)
- void **vsxev\_mask\_int16xm4** (short \*address, int16xm4\_t index, int16xm4\_t a, e16xm4\_t mask, unsigned int gvl)
- void **vsxev\_mask\_int16xm8** (short \*address, int16xm8\_t index, int16xm8\_t a, e16xm8\_t mask, unsigned int gvl)
- void **vsxev\_mask\_int32xm1** (int \*address, int32xm1\_t index, int32xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsxev\_mask\_int32xm2** (int \*address, int32xm2\_t index, int32xm2\_t a, e32xm2\_t mask, unsigned int gvl)
- void **vsxev\_mask\_int32xm4** (int \*address, int32xm4\_t index, int32xm4\_t a, e32xm4\_t mask, unsigned int gvl)
- void **vsxev\_mask\_int32xm8** (int \*address, int32xm8\_t index, int32xm8\_t a, e32xm8\_t mask, unsigned int gvl)
- void **vsxev\_mask\_int64xm1** (long \*address, int64xm1\_t index, int64xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxev\_mask\_int64xm2** (long \*address, int64xm2\_t index, int64xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsxev\_mask\_int64xm4** (long \*address, int64xm4\_t index, int64xm4\_t a, e64xm4\_t mask, unsigned int gvl)
- void **vsxev\_mask\_int64xm8** (long \*address, int64xm8\_t index, int64xm8\_t a, e64xm8\_t mask, unsigned int gvl)



- void **vsxev\_mask\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsxev\_mask\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address + index[element], a[element])
```



## 2.9.36 Store 16b in ordered-indexed memory from vector

**Instruction:** [`'vsxh.v'`]

**Prototypes:**

- void **vsxhv\_int16xm1** (short *\*address*, *int16xm1\_t* index, *int16xm1\_t* a, unsigned int gvl)
- void **vsxhv\_int16xm2** (short *\*address*, *int16xm2\_t* index, *int16xm2\_t* a, unsigned int gvl)
- void **vsxhv\_int16xm4** (short *\*address*, *int16xm4\_t* index, *int16xm4\_t* a, unsigned int gvl)
- void **vsxhv\_int16xm8** (short *\*address*, *int16xm8\_t* index, *int16xm8\_t* a, unsigned int gvl)
- void **vsxhv\_int32xm1** (int *\*address*, *int32xm1\_t* index, *int32xm1\_t* a, unsigned int gvl)
- void **vsxhv\_int32xm2** (int *\*address*, *int32xm2\_t* index, *int32xm2\_t* a, unsigned int gvl)
- void **vsxhv\_int32xm4** (int *\*address*, *int32xm4\_t* index, *int32xm4\_t* a, unsigned int gvl)
- void **vsxhv\_int32xm8** (int *\*address*, *int32xm8\_t* index, *int32xm8\_t* a, unsigned int gvl)
- void **vsxhv\_int64xm1** (long *\*address*, *int64xm1\_t* index, *int64xm1\_t* a, unsigned int gvl)
- void **vsxhv\_int64xm2** (long *\*address*, *int64xm2\_t* index, *int64xm2\_t* a, unsigned int gvl)
- void **vsxhv\_int64xm4** (long *\*address*, *int64xm4\_t* index, *int64xm4\_t* a, unsigned int gvl)
- void **vsxhv\_int64xm8** (long *\*address*, *int64xm8\_t* index, *int64xm8\_t* a, unsigned int gvl)
- void **vsxhv\_int8xm1** (signed char *\*address*, *int8xm1\_t* index, *int8xm1\_t* a, unsigned int gvl)
- void **vsxhv\_int8xm2** (signed char *\*address*, *int8xm2\_t* index, *int8xm2\_t* a, unsigned int gvl)
- void **vsxhv\_int8xm4** (signed char *\*address*, *int8xm4\_t* index, *int8xm4\_t* a, unsigned int gvl)
- void **vsxhv\_int8xm8** (signed char *\*address*, *int8xm8\_t* index, *int8xm8\_t* a, unsigned int gvl)
- void **vsxhv\_uint16xm1** (unsigned short *\*address*, *uint16xm1\_t* index, *uint16xm1\_t* a, unsigned int gvl)
- void **vsxhv\_uint16xm2** (unsigned short *\*address*, *uint16xm2\_t* index, *uint16xm2\_t* a, unsigned int gvl)
- void **vsxhv\_uint16xm4** (unsigned short *\*address*, *uint16xm4\_t* index, *uint16xm4\_t* a, unsigned int gvl)
- void **vsxhv\_uint16xm8** (unsigned short *\*address*, *uint16xm8\_t* index, *uint16xm8\_t* a, unsigned int gvl)
- void **vsxhv\_uint32xm1** (unsigned int *\*address*, *uint32xm1\_t* index, *uint32xm1\_t* a, unsigned int gvl)
- void **vsxhv\_uint32xm2** (unsigned int *\*address*, *uint32xm2\_t* index, *uint32xm2\_t* a, unsigned int gvl)
- void **vsxhv\_uint32xm4** (unsigned int *\*address*, *uint32xm4\_t* index, *uint32xm4\_t* a, unsigned int gvl)
- void **vsxhv\_uint32xm8** (unsigned int *\*address*, *uint32xm8\_t* index, *uint32xm8\_t* a, unsigned int gvl)
- void **vsxhv\_uint64xm1** (unsigned long *\*address*, *uint64xm1\_t* index, *uint64xm1\_t* a, unsigned int gvl)
- void **vsxhv\_uint64xm2** (unsigned long *\*address*, *uint64xm2\_t* index, *uint64xm2\_t* a, unsigned int gvl)
- void **vsxhv\_uint64xm4** (unsigned long *\*address*, *uint64xm4\_t* index, *uint64xm4\_t* a, unsigned int gvl)
- void **vsxhv\_uint64xm8** (unsigned long *\*address*, *uint64xm8\_t* index, *uint64xm8\_t* a, unsigned int gvl)
- void **vsxhv\_uint8xm1** (unsigned char *\*address*, *uint8xm1\_t* index, *uint8xm1\_t* a, unsigned int gvl)
- void **vsxhv\_uint8xm2** (unsigned char *\*address*, *uint8xm2\_t* index, *uint8xm2\_t* a, unsigned int gvl)
- void **vsxhv\_uint8xm4** (unsigned char *\*address*, *uint8xm4\_t* index, *uint8xm4\_t* a, unsigned int gvl)

- void **vsxhv\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, unsigned int gvl)

#### Operation:

```
>>> for element = 0 to gvl - 1
      store_element(address + index[element], a[element])
```

#### Masked prototypes:

- void **vsxhv\_mask\_int16xm1** (short \*address, *int16xm1\_t* index, *int16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int16xm2** (short \*address, *int16xm2\_t* index, *int16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int16xm4** (short \*address, *int16xm4\_t* index, *int16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int16xm8** (short \*address, *int16xm8\_t* index, *int16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int32xm1** (int \*address, *int32xm1\_t* index, *int32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int32xm2** (int \*address, *int32xm2\_t* index, *int32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int32xm4** (int \*address, *int32xm4\_t* index, *int32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int32xm8** (int \*address, *int32xm8\_t* index, *int32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int64xm1** (long \*address, *int64xm1\_t* index, *int64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int64xm2** (long \*address, *int64xm2\_t* index, *int64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int64xm4** (long \*address, *int64xm4\_t* index, *int64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)

- void **vsxhv\_mask\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsxhv\_mask\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address + index[element], a[element])
```

## 2.9.37 Store 32b in ordered-indexed memory from vector

Instruction: [*'vsxw.v'*]

Prototypes:

- void **vsxwv\_int16xm1** (short \*address, *int16xm1\_t* index, *int16xm1\_t* a, unsigned int gvl)
- void **vsxwv\_int16xm2** (short \*address, *int16xm2\_t* index, *int16xm2\_t* a, unsigned int gvl)
- void **vsxwv\_int16xm4** (short \*address, *int16xm4\_t* index, *int16xm4\_t* a, unsigned int gvl)
- void **vsxwv\_int16xm8** (short \*address, *int16xm8\_t* index, *int16xm8\_t* a, unsigned int gvl)
- void **vsxwv\_int32xm1** (int \*address, *int32xm1\_t* index, *int32xm1\_t* a, unsigned int gvl)
- void **vsxwv\_int32xm2** (int \*address, *int32xm2\_t* index, *int32xm2\_t* a, unsigned int gvl)
- void **vsxwv\_int32xm4** (int \*address, *int32xm4\_t* index, *int32xm4\_t* a, unsigned int gvl)
- void **vsxwv\_int32xm8** (int \*address, *int32xm8\_t* index, *int32xm8\_t* a, unsigned int gvl)
- void **vsxwv\_int64xm1** (long \*address, *int64xm1\_t* index, *int64xm1\_t* a, unsigned int gvl)
- void **vsxwv\_int64xm2** (long \*address, *int64xm2\_t* index, *int64xm2\_t* a, unsigned int gvl)

- void **vsxwv\_int64xm4** (long \*address, *int64xm4\_t* index, *int64xm4\_t* a, unsigned int gvl)
- void **vsxwv\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, unsigned int gvl)
- void **vsxwv\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, unsigned int gvl)
- void **vsxwv\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, unsigned int gvl)
- void **vsxwv\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, unsigned int gvl)
- void **vsxwv\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, unsigned int gvl)
- void **vsxwv\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, unsigned int gvl)
- void **vsxwv\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, unsigned int gvl)
- void **vsxwv\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, unsigned int gvl)
- void **vsxwv\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, unsigned int gvl)
- void **vsxwv\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, unsigned int gvl)
- void **vsxwv\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, unsigned int gvl)
- void **vsxwv\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, unsigned int gvl)
- void **vsxwv\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, unsigned int gvl)
- void **vsxwv\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, unsigned int gvl)
- void **vsxwv\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, unsigned int gvl)
- void **vsxwv\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, unsigned int gvl)
- void **vsxwv\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, unsigned int gvl)
- void **vsxwv\_uint8xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, unsigned int gvl)
- void **vsxwv\_uint8xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, unsigned int gvl)
- void **vsxwv\_uint8xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, unsigned int gvl)
- void **vsxwv\_uint8xm8** (unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      store_element(address + index[element], a[element])
```

**Masked prototypes:**

- void **vsxwv\_mask\_int16xm1** (short \*address, *int16xm1\_t* index, *int16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int16xm2** (short \*address, *int16xm2\_t* index, *int16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int16xm4** (short \*address, *int16xm4\_t* index, *int16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int16xm8** (short \*address, *int16xm8\_t* index, *int16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int32xm1** (int \*address, *int32xm1\_t* index, *int32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)

- void **vsxwv\_mask\_int32xm2** (int \*address, *int32xm2\_t* index, *int32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int32xm4** (int \*address, *int32xm4\_t* index, *int32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int32xm8** (int \*address, *int32xm8\_t* index, *int32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int64xm1** (long \*address, *int64xm1\_t* index, *int64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int64xm2** (long \*address, *int64xm2\_t* index, *int64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int64xm4** (long \*address, *int64xm4\_t* index, *int64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int64xm8** (long \*address, *int64xm8\_t* index, *int64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int8xm1** (signed char \*address, *int8xm1\_t* index, *int8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int8xm2** (signed char \*address, *int8xm2\_t* index, *int8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int8xm4** (signed char \*address, *int8xm4\_t* index, *int8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_int8xm8** (signed char \*address, *int8xm8\_t* index, *int8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint16xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint16xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint16xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint16xm8** (unsigned short \*address, *uint16xm8\_t* index, *uint16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint32xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint32xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint32xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint32xm8** (unsigned int \*address, *uint32xm8\_t* index, *uint32xm8\_t* a, *e32xm8\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint64xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint64xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint64xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint64xm8** (unsigned long \*address, *uint64xm8\_t* index, *uint64xm8\_t* a, *e64xm8\_t* mask, unsigned int gvl)



- void **vsxwv\_mask\_uint8xm1**(unsigned char \*address, *uint8xm1\_t* index, *uint8xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint8xm2**(unsigned char \*address, *uint8xm2\_t* index, *uint8xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint8xm4**(unsigned char \*address, *uint8xm4\_t* index, *uint8xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsxwv\_mask\_uint8xm8**(unsigned char \*address, *uint8xm8\_t* index, *uint8xm8\_t* a, *e8xm8\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        store_element(address + index[element], a[element])
```

## 2.10 Memory accesses(segment)

### 2.10.1 Load 2 contiguous 8b signed fields in memory to consecutively numbered vector registers

Instruction: ['vlseg2b.v']

Prototypes:

- *int16x2xm1\_t* **vlseg2bv\_int16x2xm1**(const short \*address, unsigned int gvl)
- *int16x2xm2\_t* **vlseg2bv\_int16x2xm2**(const short \*address, unsigned int gvl)
- *int16x2xm4\_t* **vlseg2bv\_int16x2xm4**(const short \*address, unsigned int gvl)
- *int32x2xm1\_t* **vlseg2bv\_int32x2xm1**(const int \*address, unsigned int gvl)
- *int32x2xm2\_t* **vlseg2bv\_int32x2xm2**(const int \*address, unsigned int gvl)
- *int32x2xm4\_t* **vlseg2bv\_int32x2xm4**(const int \*address, unsigned int gvl)
- *int64x2xm1\_t* **vlseg2bv\_int64x2xm1**(const long \*address, unsigned int gvl)
- *int64x2xm2\_t* **vlseg2bv\_int64x2xm2**(const long \*address, unsigned int gvl)
- *int64x2xm4\_t* **vlseg2bv\_int64x2xm4**(const long \*address, unsigned int gvl)
- *int8x2xm1\_t* **vlseg2bv\_int8x2xm1**(const signed char \*address, unsigned int gvl)
- *int8x2xm2\_t* **vlseg2bv\_int8x2xm2**(const signed char \*address, unsigned int gvl)
- *int8x2xm4\_t* **vlseg2bv\_int8x2xm4**(const signed char \*address, unsigned int gvl)

Operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
      result[segment] = load_segment(address)
      address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

Masked prototypes:

- *int16x2xm1\_t* **vlseg2bv\_mask\_int16x2xm1**(*int16x2xm1\_t* merge, const short \*address, *e16xm1\_t* mask, unsigned int gvl)



- `int16x2xm2_t vlseg2bv_mask_int16x2xm2` (`int16x2xm2_t merge`, `const short *address`, `e16xm2_t mask`, unsigned int `gvl`)
- `int16x2xm4_t vlseg2bv_mask_int16x2xm4` (`int16x2xm4_t merge`, `const short *address`, `e16xm4_t mask`, unsigned int `gvl`)
- `int32x2xm1_t vlseg2bv_mask_int32x2xm1` (`int32x2xm1_t merge`, `const int *address`, `e32xm1_t mask`, unsigned int `gvl`)
- `int32x2xm2_t vlseg2bv_mask_int32x2xm2` (`int32x2xm2_t merge`, `const int *address`, `e32xm2_t mask`, unsigned int `gvl`)
- `int32x2xm4_t vlseg2bv_mask_int32x2xm4` (`int32x2xm4_t merge`, `const int *address`, `e32xm4_t mask`, unsigned int `gvl`)
- `int64x2xm1_t vlseg2bv_mask_int64x2xm1` (`int64x2xm1_t merge`, `const long *address`, `e64xm1_t mask`, unsigned int `gvl`)
- `int64x2xm2_t vlseg2bv_mask_int64x2xm2` (`int64x2xm2_t merge`, `const long *address`, `e64xm2_t mask`, unsigned int `gvl`)
- `int64x2xm4_t vlseg2bv_mask_int64x2xm4` (`int64x2xm4_t merge`, `const long *address`, `e64xm4_t mask`, unsigned int `gvl`)
- `int8x2xm1_t vlseg2bv_mask_int8x2xm1` (`int8x2xm1_t merge`, `const signed char *address`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8x2xm2_t vlseg2bv_mask_int8x2xm2` (`int8x2xm2_t merge`, `const signed char *address`, `e8xm2_t mask`, unsigned int `gvl`)
- `int8x2xm4_t vlseg2bv_mask_int8x2xm4` (`int8x2xm4_t merge`, `const signed char *address`, `e8xm4_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
      if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
      else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.2 Load 2 contiguous 8b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg2bu.v'`]

**Prototypes:**

- `uint16x2xm1_t vlseg2bu_v_uint16x2xm1` (`const unsigned short *address`, unsigned int `gvl`)
- `uint16x2xm2_t vlseg2bu_v_uint16x2xm2` (`const unsigned short *address`, unsigned int `gvl`)
- `uint16x2xm4_t vlseg2bu_v_uint16x2xm4` (`const unsigned short *address`, unsigned int `gvl`)
- `uint32x2xm1_t vlseg2bu_v_uint32x2xm1` (`const unsigned int *address`, unsigned int `gvl`)
- `uint32x2xm2_t vlseg2bu_v_uint32x2xm2` (`const unsigned int *address`, unsigned int `gvl`)
- `uint32x2xm4_t vlseg2bu_v_uint32x2xm4` (`const unsigned int *address`, unsigned int `gvl`)
- `uint64x2xm1_t vlseg2bu_v_uint64x2xm1` (`const unsigned long *address`, unsigned int `gvl`)
- `uint64x2xm2_t vlseg2bu_v_uint64x2xm2` (`const unsigned long *address`, unsigned int `gvl`)

- `uint64x2xm4_t vlseg2buv_uint64x2xm4` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x2xm1_t vlseg2buv_uint8x2xm1` (const unsigned char *\*address*, unsigned int *gvl*)
- `uint8x2xm2_t vlseg2buv_uint8x2xm2` (const unsigned char *\*address*, unsigned int *gvl*)
- `uint8x2xm4_t vlseg2buv_uint8x2xm4` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x2xm1_t vlseg2buv_mask_uint16x2xm1` (`uint16x2xm1_t merge`, const unsigned short *\*address*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16x2xm2_t vlseg2buv_mask_uint16x2xm2` (`uint16x2xm2_t merge`, const unsigned short *\*address*, `e16xm2_t mask`, unsigned int *gvl*)
- `uint16x2xm4_t vlseg2buv_mask_uint16x2xm4` (`uint16x2xm4_t merge`, const unsigned short *\*address*, `e16xm4_t mask`, unsigned int *gvl*)
- `uint32x2xm1_t vlseg2buv_mask_uint32x2xm1` (`uint32x2xm1_t merge`, const unsigned int *\*address*, `e32xm1_t mask`, unsigned int *gvl*)
- `uint32x2xm2_t vlseg2buv_mask_uint32x2xm2` (`uint32x2xm2_t merge`, const unsigned int *\*address*, `e32xm2_t mask`, unsigned int *gvl*)
- `uint32x2xm4_t vlseg2buv_mask_uint32x2xm4` (`uint32x2xm4_t merge`, const unsigned int *\*address*, `e32xm4_t mask`, unsigned int *gvl*)
- `uint64x2xm1_t vlseg2buv_mask_uint64x2xm1` (`uint64x2xm1_t merge`, const unsigned long *\*address*, `e64xm1_t mask`, unsigned int *gvl*)
- `uint64x2xm2_t vlseg2buv_mask_uint64x2xm2` (`uint64x2xm2_t merge`, const unsigned long *\*address*, `e64xm2_t mask`, unsigned int *gvl*)
- `uint64x2xm4_t vlseg2buv_mask_uint64x2xm4` (`uint64x2xm4_t merge`, const unsigned long *\*address*, `e64xm4_t mask`, unsigned int *gvl*)
- `uint8x2xm1_t vlseg2buv_mask_uint8x2xm1` (`uint8x2xm1_t merge`, const unsigned char *\*address*, `e8xm1_t mask`, unsigned int *gvl*)
- `uint8x2xm2_t vlseg2buv_mask_uint8x2xm2` (`uint8x2xm2_t merge`, const unsigned char *\*address*, `e8xm2_t mask`, unsigned int *gvl*)
- `uint8x2xm4_t vlseg2buv_mask_uint8x2xm4` (`uint8x2xm4_t merge`, const unsigned char *\*address*, `e8xm4_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.3 Load 2 contiguous element fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg2e.v'`]

**Prototypes:**

- `float16x2xm1_t vlseg2ev_float16x2xm1` (const float16\_t \**address*, unsigned int *gvl*)
- `float16x2xm2_t vlseg2ev_float16x2xm2` (const float16\_t \**address*, unsigned int *gvl*)
- `float16x2xm4_t vlseg2ev_float16x2xm4` (const float16\_t \**address*, unsigned int *gvl*)
- `float32x2xm1_t vlseg2ev_float32x2xm1` (const float \**address*, unsigned int *gvl*)
- `float32x2xm2_t vlseg2ev_float32x2xm2` (const float \**address*, unsigned int *gvl*)
- `float32x2xm4_t vlseg2ev_float32x2xm4` (const float \**address*, unsigned int *gvl*)
- `float64x2xm1_t vlseg2ev_float64x2xm1` (const double \**address*, unsigned int *gvl*)
- `float64x2xm2_t vlseg2ev_float64x2xm2` (const double \**address*, unsigned int *gvl*)
- `float64x2xm4_t vlseg2ev_float64x2xm4` (const double \**address*, unsigned int *gvl*)
- `int16x2xm1_t vlseg2ev_int16x2xm1` (const short \**address*, unsigned int *gvl*)
- `int16x2xm2_t vlseg2ev_int16x2xm2` (const short \**address*, unsigned int *gvl*)
- `int16x2xm4_t vlseg2ev_int16x2xm4` (const short \**address*, unsigned int *gvl*)
- `int32x2xm1_t vlseg2ev_int32x2xm1` (const int \**address*, unsigned int *gvl*)
- `int32x2xm2_t vlseg2ev_int32x2xm2` (const int \**address*, unsigned int *gvl*)
- `int32x2xm4_t vlseg2ev_int32x2xm4` (const int \**address*, unsigned int *gvl*)
- `int64x2xm1_t vlseg2ev_int64x2xm1` (const long \**address*, unsigned int *gvl*)
- `int64x2xm2_t vlseg2ev_int64x2xm2` (const long \**address*, unsigned int *gvl*)
- `int64x2xm4_t vlseg2ev_int64x2xm4` (const long \**address*, unsigned int *gvl*)
- `int8x2xm1_t vlseg2ev_int8x2xm1` (const signed char \**address*, unsigned int *gvl*)
- `int8x2xm2_t vlseg2ev_int8x2xm2` (const signed char \**address*, unsigned int *gvl*)
- `int8x2xm4_t vlseg2ev_int8x2xm4` (const signed char \**address*, unsigned int *gvl*)
- `uint16x2xm1_t vlseg2ev_uint16x2xm1` (const unsigned short \**address*, unsigned int *gvl*)
- `uint16x2xm2_t vlseg2ev_uint16x2xm2` (const unsigned short \**address*, unsigned int *gvl*)
- `uint16x2xm4_t vlseg2ev_uint16x2xm4` (const unsigned short \**address*, unsigned int *gvl*)
- `uint32x2xm1_t vlseg2ev_uint32x2xm1` (const unsigned int \**address*, unsigned int *gvl*)
- `uint32x2xm2_t vlseg2ev_uint32x2xm2` (const unsigned int \**address*, unsigned int *gvl*)
- `uint32x2xm4_t vlseg2ev_uint32x2xm4` (const unsigned int \**address*, unsigned int *gvl*)
- `uint64x2xm1_t vlseg2ev_uint64x2xm1` (const unsigned long \**address*, unsigned int *gvl*)
- `uint64x2xm2_t vlseg2ev_uint64x2xm2` (const unsigned long \**address*, unsigned int *gvl*)
- `uint64x2xm4_t vlseg2ev_uint64x2xm4` (const unsigned long \**address*, unsigned int *gvl*)
- `uint8x2xm1_t vlseg2ev_uint8x2xm1` (const unsigned char \**address*, unsigned int *gvl*)
- `uint8x2xm2_t vlseg2ev_uint8x2xm2` (const unsigned char \**address*, unsigned int *gvl*)

- `uint8x2xm4_t vlseg2ev_uint8x2xm4` (const unsigned char \*address, unsigned int gvl)

#### Operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- `float16x2xm1_t vlseg2ev_mask_float16x2xm1` (float16x2xm1\_t merge, const float16\_t \*address, *e16xm1\_t* mask, unsigned int gvl)
- `float16x2xm2_t vlseg2ev_mask_float16x2xm2` (float16x2xm2\_t merge, const float16\_t \*address, *e16xm2\_t* mask, unsigned int gvl)
- `float16x2xm4_t vlseg2ev_mask_float16x2xm4` (float16x2xm4\_t merge, const float16\_t \*address, *e16xm4\_t* mask, unsigned int gvl)
- `float32x2xm1_t vlseg2ev_mask_float32x2xm1` (float32x2xm1\_t merge, const float \*address, *e32xm1\_t* mask, unsigned int gvl)
- `float32x2xm2_t vlseg2ev_mask_float32x2xm2` (float32x2xm2\_t merge, const float \*address, *e32xm2\_t* mask, unsigned int gvl)
- `float32x2xm4_t vlseg2ev_mask_float32x2xm4` (float32x2xm4\_t merge, const float \*address, *e32xm4\_t* mask, unsigned int gvl)
- `float64x2xm1_t vlseg2ev_mask_float64x2xm1` (float64x2xm1\_t merge, const double \*address, *e64xm1\_t* mask, unsigned int gvl)
- `float64x2xm2_t vlseg2ev_mask_float64x2xm2` (float64x2xm2\_t merge, const double \*address, *e64xm2\_t* mask, unsigned int gvl)
- `float64x2xm4_t vlseg2ev_mask_float64x2xm4` (float64x2xm4\_t merge, const double \*address, *e64xm4\_t* mask, unsigned int gvl)
- `int16x2xm1_t vlseg2ev_mask_int16x2xm1` (int16x2xm1\_t merge, const short \*address, *e16xm1\_t* mask, unsigned int gvl)
- `int16x2xm2_t vlseg2ev_mask_int16x2xm2` (int16x2xm2\_t merge, const short \*address, *e16xm2\_t* mask, unsigned int gvl)
- `int16x2xm4_t vlseg2ev_mask_int16x2xm4` (int16x2xm4\_t merge, const short \*address, *e16xm4\_t* mask, unsigned int gvl)
- `int32x2xm1_t vlseg2ev_mask_int32x2xm1` (int32x2xm1\_t merge, const int \*address, *e32xm1\_t* mask, unsigned int gvl)
- `int32x2xm2_t vlseg2ev_mask_int32x2xm2` (int32x2xm2\_t merge, const int \*address, *e32xm2\_t* mask, unsigned int gvl)
- `int32x2xm4_t vlseg2ev_mask_int32x2xm4` (int32x2xm4\_t merge, const int \*address, *e32xm4\_t* mask, unsigned int gvl)
- `int64x2xm1_t vlseg2ev_mask_int64x2xm1` (int64x2xm1\_t merge, const long \*address, *e64xm1\_t* mask, unsigned int gvl)
- `int64x2xm2_t vlseg2ev_mask_int64x2xm2` (int64x2xm2\_t merge, const long \*address, *e64xm2\_t* mask, unsigned int gvl)
- `int64x2xm4_t vlseg2ev_mask_int64x2xm4` (int64x2xm4\_t merge, const long \*address, *e64xm4\_t* mask, unsigned int gvl)
- `int8x2xm1_t vlseg2ev_mask_int8x2xm1` (int8x2xm1\_t merge, const signed char \*address, *e8xm1\_t* mask, unsigned int gvl)

- `int8x2xm2_t vlseg2ev_mask_int8x2xm2` (`int8x2xm2_t merge`, `const signed char *address`, `e8xm2_t mask`, `unsigned int gvl`)
- `int8x2xm4_t vlseg2ev_mask_int8x2xm4` (`int8x2xm4_t merge`, `const signed char *address`, `e8xm4_t mask`, `unsigned int gvl`)
- `uint16x2xm1_t vlseg2ev_mask_uint16x2xm1` (`uint16x2xm1_t merge`, `const unsigned short *address`, `e16xm1_t mask`, `unsigned int gvl`)
- `uint16x2xm2_t vlseg2ev_mask_uint16x2xm2` (`uint16x2xm2_t merge`, `const unsigned short *address`, `e16xm2_t mask`, `unsigned int gvl`)
- `uint16x2xm4_t vlseg2ev_mask_uint16x2xm4` (`uint16x2xm4_t merge`, `const unsigned short *address`, `e16xm4_t mask`, `unsigned int gvl`)
- `uint32x2xm1_t vlseg2ev_mask_uint32x2xm1` (`uint32x2xm1_t merge`, `const unsigned int *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `uint32x2xm2_t vlseg2ev_mask_uint32x2xm2` (`uint32x2xm2_t merge`, `const unsigned int *address`, `e32xm2_t mask`, `unsigned int gvl`)
- `uint32x2xm4_t vlseg2ev_mask_uint32x2xm4` (`uint32x2xm4_t merge`, `const unsigned int *address`, `e32xm4_t mask`, `unsigned int gvl`)
- `uint64x2xm1_t vlseg2ev_mask_uint64x2xm1` (`uint64x2xm1_t merge`, `const unsigned long *address`, `e64xm1_t mask`, `unsigned int gvl`)
- `uint64x2xm2_t vlseg2ev_mask_uint64x2xm2` (`uint64x2xm2_t merge`, `const unsigned long *address`, `e64xm2_t mask`, `unsigned int gvl`)
- `uint64x2xm4_t vlseg2ev_mask_uint64x2xm4` (`uint64x2xm4_t merge`, `const unsigned long *address`, `e64xm4_t mask`, `unsigned int gvl`)
- `uint8x2xm1_t vlseg2ev_mask_uint8x2xm1` (`uint8x2xm1_t merge`, `const unsigned char *address`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8x2xm2_t vlseg2ev_mask_uint8x2xm2` (`uint8x2xm2_t merge`, `const unsigned char *address`, `e8xm2_t mask`, `unsigned int gvl`)
- `uint8x2xm4_t vlseg2ev_mask_uint8x2xm4` (`uint8x2xm4_t merge`, `const unsigned char *address`, `e8xm4_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.4 Load 2 contiguous 16b signed fields in memory to consecutively numbered vector registers****Instruction:** [`'vlseg2h.v'`]**Prototypes:**

- `int16x2xm1_t vlseg2hv_int16x2xm1` (`const short *address`, `unsigned int gvl`)
- `int16x2xm2_t vlseg2hv_int16x2xm2` (`const short *address`, `unsigned int gvl`)
- `int16x2xm4_t vlseg2hv_int16x2xm4` (`const short *address`, `unsigned int gvl`)

- `int32x2xm1_t vlseg2hv_int32x2xm1` (const int \*address, unsigned int gvl)
- `int32x2xm2_t vlseg2hv_int32x2xm2` (const int \*address, unsigned int gvl)
- `int32x2xm4_t vlseg2hv_int32x2xm4` (const int \*address, unsigned int gvl)
- `int64x2xm1_t vlseg2hv_int64x2xm1` (const long \*address, unsigned int gvl)
- `int64x2xm2_t vlseg2hv_int64x2xm2` (const long \*address, unsigned int gvl)
- `int64x2xm4_t vlseg2hv_int64x2xm4` (const long \*address, unsigned int gvl)
- `int8x2xm1_t vlseg2hv_int8x2xm1` (const signed char \*address, unsigned int gvl)
- `int8x2xm2_t vlseg2hv_int8x2xm2` (const signed char \*address, unsigned int gvl)
- `int8x2xm4_t vlseg2hv_int8x2xm4` (const signed char \*address, unsigned int gvl)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
      result[segment] = load_segment(address)
      address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x2xm1_t vlseg2hv_mask_int16x2xm1` (int16x2xm1\_t merge, const short \*address, *e16xm1\_t mask*, unsigned int gvl)
- `int16x2xm2_t vlseg2hv_mask_int16x2xm2` (int16x2xm2\_t merge, const short \*address, *e16xm2\_t mask*, unsigned int gvl)
- `int16x2xm4_t vlseg2hv_mask_int16x2xm4` (int16x2xm4\_t merge, const short \*address, *e16xm4\_t mask*, unsigned int gvl)
- `int32x2xm1_t vlseg2hv_mask_int32x2xm1` (int32x2xm1\_t merge, const int \*address, *e32xm1\_t mask*, unsigned int gvl)
- `int32x2xm2_t vlseg2hv_mask_int32x2xm2` (int32x2xm2\_t merge, const int \*address, *e32xm2\_t mask*, unsigned int gvl)
- `int32x2xm4_t vlseg2hv_mask_int32x2xm4` (int32x2xm4\_t merge, const int \*address, *e32xm4\_t mask*, unsigned int gvl)
- `int64x2xm1_t vlseg2hv_mask_int64x2xm1` (int64x2xm1\_t merge, const long \*address, *e64xm1\_t mask*, unsigned int gvl)
- `int64x2xm2_t vlseg2hv_mask_int64x2xm2` (int64x2xm2\_t merge, const long \*address, *e64xm2\_t mask*, unsigned int gvl)
- `int64x2xm4_t vlseg2hv_mask_int64x2xm4` (int64x2xm4\_t merge, const long \*address, *e64xm4\_t mask*, unsigned int gvl)
- `int8x2xm1_t vlseg2hv_mask_int8x2xm1` (int8x2xm1\_t merge, const signed char \*address, *e8xm1\_t mask*, unsigned int gvl)
- `int8x2xm2_t vlseg2hv_mask_int8x2xm2` (int8x2xm2\_t merge, const signed char \*address, *e8xm2\_t mask*, unsigned int gvl)
- `int8x2xm4_t vlseg2hv_mask_int8x2xm4` (int8x2xm4\_t merge, const signed char \*address, *e8xm4\_t mask*, unsigned int gvl)

**Masked operation:**



```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.5 Load 2 contiguous 16b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg2hu.v`]

**Prototypes:**

- `uint16x2xm1_t vlseg2huv_uint16x2xm1` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint16x2xm2_t vlseg2huv_uint16x2xm2` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint16x2xm4_t vlseg2huv_uint16x2xm4` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint32x2xm1_t vlseg2huv_uint32x2xm1` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint32x2xm2_t vlseg2huv_uint32x2xm2` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint32x2xm4_t vlseg2huv_uint32x2xm4` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64x2xm1_t vlseg2huv_uint64x2xm1` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint64x2xm2_t vlseg2huv_uint64x2xm2` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint64x2xm4_t vlseg2huv_uint64x2xm4` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x2xm1_t vlseg2huv_uint8x2xm1` (const unsigned char *\*address*, unsigned int *gvl*)
- `uint8x2xm2_t vlseg2huv_uint8x2xm2` (const unsigned char *\*address*, unsigned int *gvl*)
- `uint8x2xm4_t vlseg2huv_uint8x2xm4` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x2xm1_t vlseg2huv_mask_uint16x2xm1` (`uint16x2xm1_t merge`, const unsigned short *\*address*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16x2xm2_t vlseg2huv_mask_uint16x2xm2` (`uint16x2xm2_t merge`, const unsigned short *\*address*, `e16xm2_t mask`, unsigned int *gvl*)
- `uint16x2xm4_t vlseg2huv_mask_uint16x2xm4` (`uint16x2xm4_t merge`, const unsigned short *\*address*, `e16xm4_t mask`, unsigned int *gvl*)
- `uint32x2xm1_t vlseg2huv_mask_uint32x2xm1` (`uint32x2xm1_t merge`, const unsigned int *\*address*, `e32xm1_t mask`, unsigned int *gvl*)
- `uint32x2xm2_t vlseg2huv_mask_uint32x2xm2` (`uint32x2xm2_t merge`, const unsigned int *\*address*, `e32xm2_t mask`, unsigned int *gvl*)

- `uint32x2xm4_t vlseg2huv_mask_uint32x2xm4` (`uint32x2xm4_t merge`, const unsigned int *\*address*, `e32xm4_t mask`, unsigned int *gvl*)
- `uint64x2xm1_t vlseg2huv_mask_uint64x2xm1` (`uint64x2xm1_t merge`, const unsigned long *\*address*, `e64xm1_t mask`, unsigned int *gvl*)
- `uint64x2xm2_t vlseg2huv_mask_uint64x2xm2` (`uint64x2xm2_t merge`, const unsigned long *\*address*, `e64xm2_t mask`, unsigned int *gvl*)
- `uint64x2xm4_t vlseg2huv_mask_uint64x2xm4` (`uint64x2xm4_t merge`, const unsigned long *\*address*, `e64xm4_t mask`, unsigned int *gvl*)
- `uint8x2xm1_t vlseg2huv_mask_uint8x2xm1` (`uint8x2xm1_t merge`, const unsigned char *\*address*, `e8xm1_t mask`, unsigned int *gvl*)
- `uint8x2xm2_t vlseg2huv_mask_uint8x2xm2` (`uint8x2xm2_t merge`, const unsigned char *\*address*, `e8xm2_t mask`, unsigned int *gvl*)
- `uint8x2xm4_t vlseg2huv_mask_uint8x2xm4` (`uint8x2xm4_t merge`, const unsigned char *\*address*, `e8xm4_t mask`, unsigned int *gvl*)

Masked operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.6 Load 2 contiguous 32b signed fields in memory to consecutively numbered vector registers

Instruction: [`vlseg2w.v`]

Prototypes:

- `int16x2xm1_t vlseg2wv_int16x2xm1` (const short *\*address*, unsigned int *gvl*)
- `int16x2xm2_t vlseg2wv_int16x2xm2` (const short *\*address*, unsigned int *gvl*)
- `int16x2xm4_t vlseg2wv_int16x2xm4` (const short *\*address*, unsigned int *gvl*)
- `int32x2xm1_t vlseg2wv_int32x2xm1` (const int *\*address*, unsigned int *gvl*)
- `int32x2xm2_t vlseg2wv_int32x2xm2` (const int *\*address*, unsigned int *gvl*)
- `int32x2xm4_t vlseg2wv_int32x2xm4` (const int *\*address*, unsigned int *gvl*)
- `int64x2xm1_t vlseg2wv_int64x2xm1` (const long *\*address*, unsigned int *gvl*)
- `int64x2xm2_t vlseg2wv_int64x2xm2` (const long *\*address*, unsigned int *gvl*)
- `int64x2xm4_t vlseg2wv_int64x2xm4` (const long *\*address*, unsigned int *gvl*)
- `int8x2xm1_t vlseg2wv_int8x2xm1` (const signed char *\*address*, unsigned int *gvl*)
- `int8x2xm2_t vlseg2wv_int8x2xm2` (const signed char *\*address*, unsigned int *gvl*)
- `int8x2xm4_t vlseg2wv_int8x2xm4` (const signed char *\*address*, unsigned int *gvl*)

Operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x2xm1_t vlseg2wv_mask_int16x2xm1` (`int16x2xm1_t merge`, `const short *address`, `e16xm1_t mask`, `unsigned int gvl`)
- `int16x2xm2_t vlseg2wv_mask_int16x2xm2` (`int16x2xm2_t merge`, `const short *address`, `e16xm2_t mask`, `unsigned int gvl`)
- `int16x2xm4_t vlseg2wv_mask_int16x2xm4` (`int16x2xm4_t merge`, `const short *address`, `e16xm4_t mask`, `unsigned int gvl`)
- `int32x2xm1_t vlseg2wv_mask_int32x2xm1` (`int32x2xm1_t merge`, `const int *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `int32x2xm2_t vlseg2wv_mask_int32x2xm2` (`int32x2xm2_t merge`, `const int *address`, `e32xm2_t mask`, `unsigned int gvl`)
- `int32x2xm4_t vlseg2wv_mask_int32x2xm4` (`int32x2xm4_t merge`, `const int *address`, `e32xm4_t mask`, `unsigned int gvl`)
- `int64x2xm1_t vlseg2wv_mask_int64x2xm1` (`int64x2xm1_t merge`, `const long *address`, `e64xm1_t mask`, `unsigned int gvl`)
- `int64x2xm2_t vlseg2wv_mask_int64x2xm2` (`int64x2xm2_t merge`, `const long *address`, `e64xm2_t mask`, `unsigned int gvl`)
- `int64x2xm4_t vlseg2wv_mask_int64x2xm4` (`int64x2xm4_t merge`, `const long *address`, `e64xm4_t mask`, `unsigned int gvl`)
- `int8x2xm1_t vlseg2wv_mask_int8x2xm1` (`int8x2xm1_t merge`, `const signed char *address`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8x2xm2_t vlseg2wv_mask_int8x2xm2` (`int8x2xm2_t merge`, `const signed char *address`, `e8xm2_t mask`, `unsigned int gvl`)
- `int8x2xm4_t vlseg2wv_mask_int8x2xm4` (`int8x2xm4_t merge`, `const signed char *address`, `e8xm4_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.7 Load 2 contiguous 32b unsigned fields in memory to consecutively numbered vector registers****Instruction:** [`vlseg2wv.v`']**Prototypes:**

- `uint16x2xm1_t vlseg2wuv_uint16x2xm1` (`const unsigned short *address`, `unsigned int gvl`)
- `uint16x2xm2_t vlseg2wuv_uint16x2xm2` (`const unsigned short *address`, `unsigned int gvl`)

- `uint16x2xm4_t vlseg2wuv_uint16x2xm4` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint32x2xm1_t vlseg2wuv_uint32x2xm1` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint32x2xm2_t vlseg2wuv_uint32x2xm2` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint32x2xm4_t vlseg2wuv_uint32x2xm4` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64x2xm1_t vlseg2wuv_uint64x2xm1` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint64x2xm2_t vlseg2wuv_uint64x2xm2` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint64x2xm4_t vlseg2wuv_uint64x2xm4` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x2xm1_t vlseg2wuv_uint8x2xm1` (const unsigned char *\*address*, unsigned int *gvl*)
- `uint8x2xm2_t vlseg2wuv_uint8x2xm2` (const unsigned char *\*address*, unsigned int *gvl*)
- `uint8x2xm4_t vlseg2wuv_uint8x2xm4` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x2xm1_t vlseg2wuv_mask_uint16x2xm1` (`uint16x2xm1_t merge`, const unsigned short *\*address*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16x2xm2_t vlseg2wuv_mask_uint16x2xm2` (`uint16x2xm2_t merge`, const unsigned short *\*address*, `e16xm2_t mask`, unsigned int *gvl*)
- `uint16x2xm4_t vlseg2wuv_mask_uint16x2xm4` (`uint16x2xm4_t merge`, const unsigned short *\*address*, `e16xm4_t mask`, unsigned int *gvl*)
- `uint32x2xm1_t vlseg2wuv_mask_uint32x2xm1` (`uint32x2xm1_t merge`, const unsigned int *\*address*, `e32xm1_t mask`, unsigned int *gvl*)
- `uint32x2xm2_t vlseg2wuv_mask_uint32x2xm2` (`uint32x2xm2_t merge`, const unsigned int *\*address*, `e32xm2_t mask`, unsigned int *gvl*)
- `uint32x2xm4_t vlseg2wuv_mask_uint32x2xm4` (`uint32x2xm4_t merge`, const unsigned int *\*address*, `e32xm4_t mask`, unsigned int *gvl*)
- `uint64x2xm1_t vlseg2wuv_mask_uint64x2xm1` (`uint64x2xm1_t merge`, const unsigned long *\*address*, `e64xm1_t mask`, unsigned int *gvl*)
- `uint64x2xm2_t vlseg2wuv_mask_uint64x2xm2` (`uint64x2xm2_t merge`, const unsigned long *\*address*, `e64xm2_t mask`, unsigned int *gvl*)
- `uint64x2xm4_t vlseg2wuv_mask_uint64x2xm4` (`uint64x2xm4_t merge`, const unsigned long *\*address*, `e64xm4_t mask`, unsigned int *gvl*)
- `uint8x2xm1_t vlseg2wuv_mask_uint8x2xm1` (`uint8x2xm1_t merge`, const unsigned char *\*address*, `e8xm1_t mask`, unsigned int *gvl*)
- `uint8x2xm2_t vlseg2wuv_mask_uint8x2xm2` (`uint8x2xm2_t merge`, const unsigned char *\*address*, `e8xm2_t mask`, unsigned int *gvl*)
- `uint8x2xm4_t vlseg2wuv_mask_uint8x2xm4` (`uint8x2xm4_t merge`, const unsigned char *\*address*, `e8xm4_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.8 Load 3 contiguous 8b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg3b.v`']

**Prototypes:**

- `int16x3xm1_t vlseg3bv_int16x3xm1` (const short \*address, unsigned int gvl)
- `int16x3xm2_t vlseg3bv_int16x3xm2` (const short \*address, unsigned int gvl)
- `int32x3xm1_t vlseg3bv_int32x3xm1` (const int \*address, unsigned int gvl)
- `int32x3xm2_t vlseg3bv_int32x3xm2` (const int \*address, unsigned int gvl)
- `int64x3xm1_t vlseg3bv_int64x3xm1` (const long \*address, unsigned int gvl)
- `int64x3xm2_t vlseg3bv_int64x3xm2` (const long \*address, unsigned int gvl)
- `int8x3xm1_t vlseg3bv_int8x3xm1` (const signed char \*address, unsigned int gvl)
- `int8x3xm2_t vlseg3bv_int8x3xm2` (const signed char \*address, unsigned int gvl)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x3xm1_t vlseg3bv_mask_int16x3xm1` (int16x3xm1\_t merge, const short \*address, *e16xm1\_t mask*, unsigned int gvl)
- `int16x3xm2_t vlseg3bv_mask_int16x3xm2` (int16x3xm2\_t merge, const short \*address, *e16xm2\_t mask*, unsigned int gvl)
- `int32x3xm1_t vlseg3bv_mask_int32x3xm1` (int32x3xm1\_t merge, const int \*address, *e32xm1\_t mask*, unsigned int gvl)
- `int32x3xm2_t vlseg3bv_mask_int32x3xm2` (int32x3xm2\_t merge, const int \*address, *e32xm2\_t mask*, unsigned int gvl)
- `int64x3xm1_t vlseg3bv_mask_int64x3xm1` (int64x3xm1\_t merge, const long \*address, *e64xm1\_t mask*, unsigned int gvl)
- `int64x3xm2_t vlseg3bv_mask_int64x3xm2` (int64x3xm2\_t merge, const long \*address, *e64xm2\_t mask*, unsigned int gvl)
- `int8x3xm1_t vlseg3bv_mask_int8x3xm1` (int8x3xm1\_t merge, const signed char \*address, *e8xm1\_t mask*, unsigned int gvl)
- `int8x3xm2_t vlseg3bv_mask_int8x3xm2` (int8x3xm2\_t merge, const signed char \*address, *e8xm2\_t mask*, unsigned int gvl)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.9 Load 3 contiguous 8b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg3bu.v`]

**Prototypes:**

- `uint16x3xm1_t vlseg3bu_v_uint16x3xm1` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint16x3xm2_t vlseg3bu_v_uint16x3xm2` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint32x3xm1_t vlseg3bu_v_uint32x3xm1` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint32x3xm2_t vlseg3bu_v_uint32x3xm2` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64x3xm1_t vlseg3bu_v_uint64x3xm1` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint64x3xm2_t vlseg3bu_v_uint64x3xm2` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x3xm1_t vlseg3bu_v_uint8x3xm1` (const unsigned char *\*address*, unsigned int *gvl*)
- `uint8x3xm2_t vlseg3bu_v_uint8x3xm2` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x3xm1_t vlseg3bu_mask_uint16x3xm1` (`uint16x3xm1_t merge`, const unsigned short *\*address*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16x3xm2_t vlseg3bu_mask_uint16x3xm2` (`uint16x3xm2_t merge`, const unsigned short *\*address*, `e16xm2_t mask`, unsigned int *gvl*)
- `uint32x3xm1_t vlseg3bu_mask_uint32x3xm1` (`uint32x3xm1_t merge`, const unsigned int *\*address*, `e32xm1_t mask`, unsigned int *gvl*)
- `uint32x3xm2_t vlseg3bu_mask_uint32x3xm2` (`uint32x3xm2_t merge`, const unsigned int *\*address*, `e32xm2_t mask`, unsigned int *gvl*)
- `uint64x3xm1_t vlseg3bu_mask_uint64x3xm1` (`uint64x3xm1_t merge`, const unsigned long *\*address*, `e64xm1_t mask`, unsigned int *gvl*)
- `uint64x3xm2_t vlseg3bu_mask_uint64x3xm2` (`uint64x3xm2_t merge`, const unsigned long *\*address*, `e64xm2_t mask`, unsigned int *gvl*)
- `uint8x3xm1_t vlseg3bu_mask_uint8x3xm1` (`uint8x3xm1_t merge`, const unsigned char *\*address*, `e8xm1_t mask`, unsigned int *gvl*)



- `uint8x3xm2_t vlseg3buvmask_uint8x3xm2` (`uint8x3xm2_t merge`, `const unsigned char *address`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.10 Load 3 contiguous element fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg3e.v`']

**Prototypes:**

- `float16x3xm1_t vlseg3ev_float16x3xm1` (`const float16_t *address`, `unsigned int gvl`)
- `float16x3xm2_t vlseg3ev_float16x3xm2` (`const float16_t *address`, `unsigned int gvl`)
- `float32x3xm1_t vlseg3ev_float32x3xm1` (`const float *address`, `unsigned int gvl`)
- `float32x3xm2_t vlseg3ev_float32x3xm2` (`const float *address`, `unsigned int gvl`)
- `float64x3xm1_t vlseg3ev_float64x3xm1` (`const double *address`, `unsigned int gvl`)
- `float64x3xm2_t vlseg3ev_float64x3xm2` (`const double *address`, `unsigned int gvl`)
- `int16x3xm1_t vlseg3ev_int16x3xm1` (`const short *address`, `unsigned int gvl`)
- `int16x3xm2_t vlseg3ev_int16x3xm2` (`const short *address`, `unsigned int gvl`)
- `int32x3xm1_t vlseg3ev_int32x3xm1` (`const int *address`, `unsigned int gvl`)
- `int32x3xm2_t vlseg3ev_int32x3xm2` (`const int *address`, `unsigned int gvl`)
- `int64x3xm1_t vlseg3ev_int64x3xm1` (`const long *address`, `unsigned int gvl`)
- `int64x3xm2_t vlseg3ev_int64x3xm2` (`const long *address`, `unsigned int gvl`)
- `int8x3xm1_t vlseg3ev_int8x3xm1` (`const signed char *address`, `unsigned int gvl`)
- `int8x3xm2_t vlseg3ev_int8x3xm2` (`const signed char *address`, `unsigned int gvl`)
- `uint16x3xm1_t vlseg3ev_uint16x3xm1` (`const unsigned short *address`, `unsigned int gvl`)
- `uint16x3xm2_t vlseg3ev_uint16x3xm2` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32x3xm1_t vlseg3ev_uint32x3xm1` (`const unsigned int *address`, `unsigned int gvl`)
- `uint32x3xm2_t vlseg3ev_uint32x3xm2` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64x3xm1_t vlseg3ev_uint64x3xm1` (`const unsigned long *address`, `unsigned int gvl`)
- `uint64x3xm2_t vlseg3ev_uint64x3xm2` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8x3xm1_t vlseg3ev_uint8x3xm1` (`const unsigned char *address`, `unsigned int gvl`)
- `uint8x3xm2_t vlseg3ev_uint8x3xm2` (`const unsigned char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
      result[segment] = load_segment(address)
      address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

### Masked prototypes:

- float16x3xm1\_t vlseg3ev\_mask\_float16x3xm1 (float16x3xm1\_t merge, const float16\_t \*address, e16xm1\_t mask, unsigned int gvl)
- float16x3xm2\_t vlseg3ev\_mask\_float16x3xm2 (float16x3xm2\_t merge, const float16\_t \*address, e16xm2\_t mask, unsigned int gvl)
- float32x3xm1\_t vlseg3ev\_mask\_float32x3xm1 (float32x3xm1\_t merge, const float \*address, e32xm1\_t mask, unsigned int gvl)
- float32x3xm2\_t vlseg3ev\_mask\_float32x3xm2 (float32x3xm2\_t merge, const float \*address, e32xm2\_t mask, unsigned int gvl)
- float64x3xm1\_t vlseg3ev\_mask\_float64x3xm1 (float64x3xm1\_t merge, const double \*address, e64xm1\_t mask, unsigned int gvl)
- float64x3xm2\_t vlseg3ev\_mask\_float64x3xm2 (float64x3xm2\_t merge, const double \*address, e64xm2\_t mask, unsigned int gvl)
- int16x3xm1\_t vlseg3ev\_mask\_int16x3xm1 (int16x3xm1\_t merge, const short \*address, e16xm1\_t mask, unsigned int gvl)
- int16x3xm2\_t vlseg3ev\_mask\_int16x3xm2 (int16x3xm2\_t merge, const short \*address, e16xm2\_t mask, unsigned int gvl)
- int32x3xm1\_t vlseg3ev\_mask\_int32x3xm1 (int32x3xm1\_t merge, const int \*address, e32xm1\_t mask, unsigned int gvl)
- int32x3xm2\_t vlseg3ev\_mask\_int32x3xm2 (int32x3xm2\_t merge, const int \*address, e32xm2\_t mask, unsigned int gvl)
- int64x3xm1\_t vlseg3ev\_mask\_int64x3xm1 (int64x3xm1\_t merge, const long \*address, e64xm1\_t mask, unsigned int gvl)
- int64x3xm2\_t vlseg3ev\_mask\_int64x3xm2 (int64x3xm2\_t merge, const long \*address, e64xm2\_t mask, unsigned int gvl)
- int8x3xm1\_t vlseg3ev\_mask\_int8x3xm1 (int8x3xm1\_t merge, const signed char \*address, e8xm1\_t mask, unsigned int gvl)
- int8x3xm2\_t vlseg3ev\_mask\_int8x3xm2 (int8x3xm2\_t merge, const signed char \*address, e8xm2\_t mask, unsigned int gvl)
- uint16x3xm1\_t vlseg3ev\_mask\_uint16x3xm1 (uint16x3xm1\_t merge, const unsigned short \*address, e16xm1\_t mask, unsigned int gvl)
- uint16x3xm2\_t vlseg3ev\_mask\_uint16x3xm2 (uint16x3xm2\_t merge, const unsigned short \*address, e16xm2\_t mask, unsigned int gvl)
- uint32x3xm1\_t vlseg3ev\_mask\_uint32x3xm1 (uint32x3xm1\_t merge, const unsigned int \*address, e32xm1\_t mask, unsigned int gvl)
- uint32x3xm2\_t vlseg3ev\_mask\_uint32x3xm2 (uint32x3xm2\_t merge, const unsigned int \*address, e32xm2\_t mask, unsigned int gvl)
- uint64x3xm1\_t vlseg3ev\_mask\_uint64x3xm1 (uint64x3xm1\_t merge, const unsigned long \*address, e64xm1\_t mask, unsigned int gvl)
- uint64x3xm2\_t vlseg3ev\_mask\_uint64x3xm2 (uint64x3xm2\_t merge, const unsigned long \*address, e64xm2\_t mask, unsigned int gvl)

- `uint8x3xm1_t vlseg3ev_mask_uint8x3xm1` (`uint8x3xm1_t merge`, `const unsigned char *address`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8x3xm2_t vlseg3ev_mask_uint8x3xm2` (`uint8x3xm2_t merge`, `const unsigned char *address`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.11 Load 3 contiguous 16b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg3h.v`']

**Prototypes:**

- `int16x3xm1_t vlseg3hv_int16x3xm1` (`const short *address`, `unsigned int gvl`)
- `int16x3xm2_t vlseg3hv_int16x3xm2` (`const short *address`, `unsigned int gvl`)
- `int32x3xm1_t vlseg3hv_int32x3xm1` (`const int *address`, `unsigned int gvl`)
- `int32x3xm2_t vlseg3hv_int32x3xm2` (`const int *address`, `unsigned int gvl`)
- `int64x3xm1_t vlseg3hv_int64x3xm1` (`const long *address`, `unsigned int gvl`)
- `int64x3xm2_t vlseg3hv_int64x3xm2` (`const long *address`, `unsigned int gvl`)
- `int8x3xm1_t vlseg3hv_int8x3xm1` (`const signed char *address`, `unsigned int gvl`)
- `int8x3xm2_t vlseg3hv_int8x3xm2` (`const signed char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x3xm1_t vlseg3hv_mask_int16x3xm1` (`int16x3xm1_t merge`, `const short *address`, `e16xm1_t mask`, `unsigned int gvl`)
- `int16x3xm2_t vlseg3hv_mask_int16x3xm2` (`int16x3xm2_t merge`, `const short *address`, `e16xm2_t mask`, `unsigned int gvl`)
- `int32x3xm1_t vlseg3hv_mask_int32x3xm1` (`int32x3xm1_t merge`, `const int *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `int32x3xm2_t vlseg3hv_mask_int32x3xm2` (`int32x3xm2_t merge`, `const int *address`, `e32xm2_t mask`, `unsigned int gvl`)
- `int64x3xm1_t vlseg3hv_mask_int64x3xm1` (`int64x3xm1_t merge`, `const long *address`, `e64xm1_t mask`, `unsigned int gvl`)

- `int64x3xm2_t vlseg3hv_mask_int64x3xm2` (`int64x3xm2_t merge`, `const long *address`, `e64xm2_t mask`, `unsigned int gvl`)
- `int8x3xm1_t vlseg3hv_mask_int8x3xm1` (`int8x3xm1_t merge`, `const signed char *address`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8x3xm2_t vlseg3hv_mask_int8x3xm2` (`int8x3xm2_t merge`, `const signed char *address`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.12 Load 3 contiguous 16b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg3hu.v`]

**Prototypes:**

- `uint16x3xm1_t vlseg3huv_uint16x3xm1` (`const unsigned short *address`, `unsigned int gvl`)
- `uint16x3xm2_t vlseg3huv_uint16x3xm2` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32x3xm1_t vlseg3huv_uint32x3xm1` (`const unsigned int *address`, `unsigned int gvl`)
- `uint32x3xm2_t vlseg3huv_uint32x3xm2` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64x3xm1_t vlseg3huv_uint64x3xm1` (`const unsigned long *address`, `unsigned int gvl`)
- `uint64x3xm2_t vlseg3huv_uint64x3xm2` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8x3xm1_t vlseg3huv_uint8x3xm1` (`const unsigned char *address`, `unsigned int gvl`)
- `uint8x3xm2_t vlseg3huv_uint8x3xm2` (`const unsigned char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x3xm1_t vlseg3huv_mask_uint16x3xm1` (`uint16x3xm1_t merge`, `const unsigned short *address`, `e16xm1_t mask`, `unsigned int gvl`)
- `uint16x3xm2_t vlseg3huv_mask_uint16x3xm2` (`uint16x3xm2_t merge`, `const unsigned short *address`, `e16xm2_t mask`, `unsigned int gvl`)
- `uint32x3xm1_t vlseg3huv_mask_uint32x3xm1` (`uint32x3xm1_t merge`, `const unsigned int *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `uint32x3xm2_t vlseg3huv_mask_uint32x3xm2` (`uint32x3xm2_t merge`, `const unsigned int *address`, `e32xm2_t mask`, `unsigned int gvl`)

- `uint64x3xm1_t vlseg3huv_mask_uint64x3xm1` (`uint64x3xm1_t merge`, const unsigned long *\*address*, *e64xm1\_t mask*, unsigned int *gvl*)
- `uint64x3xm2_t vlseg3huv_mask_uint64x3xm2` (`uint64x3xm2_t merge`, const unsigned long *\*address*, *e64xm2\_t mask*, unsigned int *gvl*)
- `uint8x3xm1_t vlseg3huv_mask_uint8x3xm1` (`uint8x3xm1_t merge`, const unsigned char *\*address*, *e8xm1\_t mask*, unsigned int *gvl*)
- `uint8x3xm2_t vlseg3huv_mask_uint8x3xm2` (`uint8x3xm2_t merge`, const unsigned char *\*address*, *e8xm2\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.13 Load 3 contiguous 32b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg3w.v'`]

**Prototypes:**

- `int16x3xm1_t vlseg3wv_int16x3xm1` (const short *\*address*, unsigned int *gvl*)
- `int16x3xm2_t vlseg3wv_int16x3xm2` (const short *\*address*, unsigned int *gvl*)
- `int32x3xm1_t vlseg3wv_int32x3xm1` (const int *\*address*, unsigned int *gvl*)
- `int32x3xm2_t vlseg3wv_int32x3xm2` (const int *\*address*, unsigned int *gvl*)
- `int64x3xm1_t vlseg3wv_int64x3xm1` (const long *\*address*, unsigned int *gvl*)
- `int64x3xm2_t vlseg3wv_int64x3xm2` (const long *\*address*, unsigned int *gvl*)
- `int8x3xm1_t vlseg3wv_int8x3xm1` (const signed char *\*address*, unsigned int *gvl*)
- `int8x3xm2_t vlseg3wv_int8x3xm2` (const signed char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x3xm1_t vlseg3wv_mask_int16x3xm1` (`int16x3xm1_t merge`, const short *\*address*, *e16xm1\_t mask*, unsigned int *gvl*)
- `int16x3xm2_t vlseg3wv_mask_int16x3xm2` (`int16x3xm2_t merge`, const short *\*address*, *e16xm2\_t mask*, unsigned int *gvl*)
- `int32x3xm1_t vlseg3wv_mask_int32x3xm1` (`int32x3xm1_t merge`, const int *\*address*, *e32xm1\_t mask*, unsigned int *gvl*)

- `int32x3xm2_t vlseg3wv_mask_int32x3xm2` (`int32x3xm2_t merge`, `const int *address`, `e32xm2_t mask`, `unsigned int gvl`)
- `int64x3xm1_t vlseg3wv_mask_int64x3xm1` (`int64x3xm1_t merge`, `const long *address`, `e64xm1_t mask`, `unsigned int gvl`)
- `int64x3xm2_t vlseg3wv_mask_int64x3xm2` (`int64x3xm2_t merge`, `const long *address`, `e64xm2_t mask`, `unsigned int gvl`)
- `int8x3xm1_t vlseg3wv_mask_int8x3xm1` (`int8x3xm1_t merge`, `const signed char *address`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8x3xm2_t vlseg3wv_mask_int8x3xm2` (`int8x3xm2_t merge`, `const signed char *address`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.14 Load 3 contiguous 32b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg3wu.v`']

**Prototypes:**

- `uint16x3xm1_t vlseg3wuv_uint16x3xm1` (`const unsigned short *address`, `unsigned int gvl`)
- `uint16x3xm2_t vlseg3wuv_uint16x3xm2` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32x3xm1_t vlseg3wuv_uint32x3xm1` (`const unsigned int *address`, `unsigned int gvl`)
- `uint32x3xm2_t vlseg3wuv_uint32x3xm2` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64x3xm1_t vlseg3wuv_uint64x3xm1` (`const unsigned long *address`, `unsigned int gvl`)
- `uint64x3xm2_t vlseg3wuv_uint64x3xm2` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8x3xm1_t vlseg3wuv_uint8x3xm1` (`const unsigned char *address`, `unsigned int gvl`)
- `uint8x3xm2_t vlseg3wuv_uint8x3xm2` (`const unsigned char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x3xm1_t vlseg3wuv_mask_uint16x3xm1` (`uint16x3xm1_t merge`, `const unsigned short *address`, `e16xm1_t mask`, `unsigned int gvl`)
- `uint16x3xm2_t vlseg3wuv_mask_uint16x3xm2` (`uint16x3xm2_t merge`, `const unsigned short *address`, `e16xm2_t mask`, `unsigned int gvl`)



- `uint32x3xm1_t vlseg3wuv_mask_uint32x3xm1` (`uint32x3xm1_t merge`, `const unsigned int *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `uint32x3xm2_t vlseg3wuv_mask_uint32x3xm2` (`uint32x3xm2_t merge`, `const unsigned int *address`, `e32xm2_t mask`, `unsigned int gvl`)
- `uint64x3xm1_t vlseg3wuv_mask_uint64x3xm1` (`uint64x3xm1_t merge`, `const unsigned long *address`, `e64xm1_t mask`, `unsigned int gvl`)
- `uint64x3xm2_t vlseg3wuv_mask_uint64x3xm2` (`uint64x3xm2_t merge`, `const unsigned long *address`, `e64xm2_t mask`, `unsigned int gvl`)
- `uint8x3xm1_t vlseg3wuv_mask_uint8x3xm1` (`uint8x3xm1_t merge`, `const unsigned char *address`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8x3xm2_t vlseg3wuv_mask_uint8x3xm2` (`uint8x3xm2_t merge`, `const unsigned char *address`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.15 Load 4 contiguous 8b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg4b.v`]

**Prototypes:**

- `int16x4xm1_t vlseg4bv_int16x4xm1` (`const short *address`, `unsigned int gvl`)
- `int16x4xm2_t vlseg4bv_int16x4xm2` (`const short *address`, `unsigned int gvl`)
- `int32x4xm1_t vlseg4bv_int32x4xm1` (`const int *address`, `unsigned int gvl`)
- `int32x4xm2_t vlseg4bv_int32x4xm2` (`const int *address`, `unsigned int gvl`)
- `int64x4xm1_t vlseg4bv_int64x4xm1` (`const long *address`, `unsigned int gvl`)
- `int64x4xm2_t vlseg4bv_int64x4xm2` (`const long *address`, `unsigned int gvl`)
- `int8x4xm1_t vlseg4bv_int8x4xm1` (`const signed char *address`, `unsigned int gvl`)
- `int8x4xm2_t vlseg4bv_int8x4xm2` (`const signed char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x4xm1_t vlseg4bv_mask_int16x4xm1` (`int16x4xm1_t merge`, `const short *address`, `e16xm1_t mask`, `unsigned int gvl`)

- `int16x4xm2_t vlseg4bv_mask_int16x4xm2` (`int16x4xm2_t merge`, `const short *address`, `e16xm2_t mask`, `unsigned int gvl`)
- `int32x4xm1_t vlseg4bv_mask_int32x4xm1` (`int32x4xm1_t merge`, `const int *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `int32x4xm2_t vlseg4bv_mask_int32x4xm2` (`int32x4xm2_t merge`, `const int *address`, `e32xm2_t mask`, `unsigned int gvl`)
- `int64x4xm1_t vlseg4bv_mask_int64x4xm1` (`int64x4xm1_t merge`, `const long *address`, `e64xm1_t mask`, `unsigned int gvl`)
- `int64x4xm2_t vlseg4bv_mask_int64x4xm2` (`int64x4xm2_t merge`, `const long *address`, `e64xm2_t mask`, `unsigned int gvl`)
- `int8x4xm1_t vlseg4bv_mask_int8x4xm1` (`int8x4xm1_t merge`, `const signed char *address`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8x4xm2_t vlseg4bv_mask_int8x4xm2` (`int8x4xm2_t merge`, `const signed char *address`, `e8xm2_t mask`, `unsigned int gvl`)

Masked operation:

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.16 Load 4 contiguous 8b unsigned fields in memory to consecutively numbered vector registers

Instruction: [`vlseg4bu.v`']

Prototypes:

- `uint16x4xm1_t vlseg4bu_v_uint16x4xm1` (`const unsigned short *address`, `unsigned int gvl`)
- `uint16x4xm2_t vlseg4bu_v_uint16x4xm2` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32x4xm1_t vlseg4bu_v_uint32x4xm1` (`const unsigned int *address`, `unsigned int gvl`)
- `uint32x4xm2_t vlseg4bu_v_uint32x4xm2` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64x4xm1_t vlseg4bu_v_uint64x4xm1` (`const unsigned long *address`, `unsigned int gvl`)
- `uint64x4xm2_t vlseg4bu_v_uint64x4xm2` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8x4xm1_t vlseg4bu_v_uint8x4xm1` (`const unsigned char *address`, `unsigned int gvl`)
- `uint8x4xm2_t vlseg4bu_v_uint8x4xm2` (`const unsigned char *address`, `unsigned int gvl`)

Operation:

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

Masked prototypes:

- `uint16x4xm1_t vlseg4buv_mask_uint16x4xm1` (`uint16x4xm1_t merge`, const unsigned short *\*address*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16x4xm2_t vlseg4buv_mask_uint16x4xm2` (`uint16x4xm2_t merge`, const unsigned short *\*address*, `e16xm2_t mask`, unsigned int *gvl*)
- `uint32x4xm1_t vlseg4buv_mask_uint32x4xm1` (`uint32x4xm1_t merge`, const unsigned int *\*address*, `e32xm1_t mask`, unsigned int *gvl*)
- `uint32x4xm2_t vlseg4buv_mask_uint32x4xm2` (`uint32x4xm2_t merge`, const unsigned int *\*address*, `e32xm2_t mask`, unsigned int *gvl*)
- `uint64x4xm1_t vlseg4buv_mask_uint64x4xm1` (`uint64x4xm1_t merge`, const unsigned long *\*address*, `e64xm1_t mask`, unsigned int *gvl*)
- `uint64x4xm2_t vlseg4buv_mask_uint64x4xm2` (`uint64x4xm2_t merge`, const unsigned long *\*address*, `e64xm2_t mask`, unsigned int *gvl*)
- `uint8x4xm1_t vlseg4buv_mask_uint8x4xm1` (`uint8x4xm1_t merge`, const unsigned char *\*address*, `e8xm1_t mask`, unsigned int *gvl*)
- `uint8x4xm2_t vlseg4buv_mask_uint8x4xm2` (`uint8x4xm2_t merge`, const unsigned char *\*address*, `e8xm2_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.17 Load 4 contiguous element fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg4e.v`']

**Prototypes:**

- `float16x4xm1_t vlseg4ev_float16x4xm1` (const float16\_t *\*address*, unsigned int *gvl*)
- `float16x4xm2_t vlseg4ev_float16x4xm2` (const float16\_t *\*address*, unsigned int *gvl*)
- `float32x4xm1_t vlseg4ev_float32x4xm1` (const float *\*address*, unsigned int *gvl*)
- `float32x4xm2_t vlseg4ev_float32x4xm2` (const float *\*address*, unsigned int *gvl*)
- `float64x4xm1_t vlseg4ev_float64x4xm1` (const double *\*address*, unsigned int *gvl*)
- `float64x4xm2_t vlseg4ev_float64x4xm2` (const double *\*address*, unsigned int *gvl*)
- `int16x4xm1_t vlseg4ev_int16x4xm1` (const short *\*address*, unsigned int *gvl*)
- `int16x4xm2_t vlseg4ev_int16x4xm2` (const short *\*address*, unsigned int *gvl*)
- `int32x4xm1_t vlseg4ev_int32x4xm1` (const int *\*address*, unsigned int *gvl*)
- `int32x4xm2_t vlseg4ev_int32x4xm2` (const int *\*address*, unsigned int *gvl*)
- `int64x4xm1_t vlseg4ev_int64x4xm1` (const long *\*address*, unsigned int *gvl*)
- `int64x4xm2_t vlseg4ev_int64x4xm2` (const long *\*address*, unsigned int *gvl*)
- `int8x4xm1_t vlseg4ev_int8x4xm1` (const signed char *\*address*, unsigned int *gvl*)

- `int8x4xm2_t vlseg4ev_int8x4xm2` (const signed char *\*address*, unsigned int *gvl*)
- `uint16x4xm1_t vlseg4ev_uint16x4xm1` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint16x4xm2_t vlseg4ev_uint16x4xm2` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint32x4xm1_t vlseg4ev_uint32x4xm1` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint32x4xm2_t vlseg4ev_uint32x4xm2` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64x4xm1_t vlseg4ev_uint64x4xm1` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint64x4xm2_t vlseg4ev_uint64x4xm2` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x4xm1_t vlseg4ev_uint8x4xm1` (const unsigned char *\*address*, unsigned int *gvl*)
- `uint8x4xm2_t vlseg4ev_uint8x4xm2` (const unsigned char *\*address*, unsigned int *gvl*)

#### Operation:

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- `float16x4xm1_t vlseg4ev_mask_float16x4xm1` (*float16x4xm1\_t merge*, const float16\_t *\*address*, *e16xm1\_t mask*, unsigned int *gvl*)
- `float16x4xm2_t vlseg4ev_mask_float16x4xm2` (*float16x4xm2\_t merge*, const float16\_t *\*address*, *e16xm2\_t mask*, unsigned int *gvl*)
- `float32x4xm1_t vlseg4ev_mask_float32x4xm1` (*float32x4xm1\_t merge*, const float *\*address*, *e32xm1\_t mask*, unsigned int *gvl*)
- `float32x4xm2_t vlseg4ev_mask_float32x4xm2` (*float32x4xm2\_t merge*, const float *\*address*, *e32xm2\_t mask*, unsigned int *gvl*)
- `float64x4xm1_t vlseg4ev_mask_float64x4xm1` (*float64x4xm1\_t merge*, const double *\*address*, *e64xm1\_t mask*, unsigned int *gvl*)
- `float64x4xm2_t vlseg4ev_mask_float64x4xm2` (*float64x4xm2\_t merge*, const double *\*address*, *e64xm2\_t mask*, unsigned int *gvl*)
- `int16x4xm1_t vlseg4ev_mask_int16x4xm1` (*int16x4xm1\_t merge*, const short *\*address*, *e16xm1\_t mask*, unsigned int *gvl*)
- `int16x4xm2_t vlseg4ev_mask_int16x4xm2` (*int16x4xm2\_t merge*, const short *\*address*, *e16xm2\_t mask*, unsigned int *gvl*)
- `int32x4xm1_t vlseg4ev_mask_int32x4xm1` (*int32x4xm1\_t merge*, const int *\*address*, *e32xm1\_t mask*, unsigned int *gvl*)
- `int32x4xm2_t vlseg4ev_mask_int32x4xm2` (*int32x4xm2\_t merge*, const int *\*address*, *e32xm2\_t mask*, unsigned int *gvl*)
- `int64x4xm1_t vlseg4ev_mask_int64x4xm1` (*int64x4xm1\_t merge*, const long *\*address*, *e64xm1\_t mask*, unsigned int *gvl*)
- `int64x4xm2_t vlseg4ev_mask_int64x4xm2` (*int64x4xm2\_t merge*, const long *\*address*, *e64xm2\_t mask*, unsigned int *gvl*)
- `int8x4xm1_t vlseg4ev_mask_int8x4xm1` (*int8x4xm1\_t merge*, const signed char *\*address*, *e8xm1\_t mask*, unsigned int *gvl*)
- `int8x4xm2_t vlseg4ev_mask_int8x4xm2` (*int8x4xm2\_t merge*, const signed char *\*address*, *e8xm2\_t mask*, unsigned int *gvl*)

- `uint16x4xm1_t vlseg4ev_mask_uint16x4xm1` (`uint16x4xm1_t merge`, const unsigned short *\*address*, *e16xm1\_t* mask, unsigned int *gvl*)
- `uint16x4xm2_t vlseg4ev_mask_uint16x4xm2` (`uint16x4xm2_t merge`, const unsigned short *\*address*, *e16xm2\_t* mask, unsigned int *gvl*)
- `uint32x4xm1_t vlseg4ev_mask_uint32x4xm1` (`uint32x4xm1_t merge`, const unsigned int *\*address*, *e32xm1\_t* mask, unsigned int *gvl*)
- `uint32x4xm2_t vlseg4ev_mask_uint32x4xm2` (`uint32x4xm2_t merge`, const unsigned int *\*address*, *e32xm2\_t* mask, unsigned int *gvl*)
- `uint64x4xm1_t vlseg4ev_mask_uint64x4xm1` (`uint64x4xm1_t merge`, const unsigned long *\*address*, *e64xm1\_t* mask, unsigned int *gvl*)
- `uint64x4xm2_t vlseg4ev_mask_uint64x4xm2` (`uint64x4xm2_t merge`, const unsigned long *\*address*, *e64xm2\_t* mask, unsigned int *gvl*)
- `uint8x4xm1_t vlseg4ev_mask_uint8x4xm1` (`uint8x4xm1_t merge`, const unsigned char *\*address*, *e8xm1\_t* mask, unsigned int *gvl*)
- `uint8x4xm2_t vlseg4ev_mask_uint8x4xm2` (`uint8x4xm2_t merge`, const unsigned char *\*address*, *e8xm2\_t* mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.18 Load 4 contiguous 16b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg4h.v`']

**Prototypes:**

- `int16x4xm1_t vlseg4hv_int16x4xm1` (const short *\*address*, unsigned int *gvl*)
- `int16x4xm2_t vlseg4hv_int16x4xm2` (const short *\*address*, unsigned int *gvl*)
- `int32x4xm1_t vlseg4hv_int32x4xm1` (const int *\*address*, unsigned int *gvl*)
- `int32x4xm2_t vlseg4hv_int32x4xm2` (const int *\*address*, unsigned int *gvl*)
- `int64x4xm1_t vlseg4hv_int64x4xm1` (const long *\*address*, unsigned int *gvl*)
- `int64x4xm2_t vlseg4hv_int64x4xm2` (const long *\*address*, unsigned int *gvl*)
- `int8x4xm1_t vlseg4hv_int8x4xm1` (const signed char *\*address*, unsigned int *gvl*)
- `int8x4xm2_t vlseg4hv_int8x4xm2` (const signed char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x4xm1_t vlseg4hv_mask_int16x4xm1` (`int16x4xm1_t merge`, `const short *address`, `e16xm1_t mask`, `unsigned int gvl`)
- `int16x4xm2_t vlseg4hv_mask_int16x4xm2` (`int16x4xm2_t merge`, `const short *address`, `e16xm2_t mask`, `unsigned int gvl`)
- `int32x4xm1_t vlseg4hv_mask_int32x4xm1` (`int32x4xm1_t merge`, `const int *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `int32x4xm2_t vlseg4hv_mask_int32x4xm2` (`int32x4xm2_t merge`, `const int *address`, `e32xm2_t mask`, `unsigned int gvl`)
- `int64x4xm1_t vlseg4hv_mask_int64x4xm1` (`int64x4xm1_t merge`, `const long *address`, `e64xm1_t mask`, `unsigned int gvl`)
- `int64x4xm2_t vlseg4hv_mask_int64x4xm2` (`int64x4xm2_t merge`, `const long *address`, `e64xm2_t mask`, `unsigned int gvl`)
- `int8x4xm1_t vlseg4hv_mask_int8x4xm1` (`int8x4xm1_t merge`, `const signed char *address`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8x4xm2_t vlseg4hv_mask_int8x4xm2` (`int8x4xm2_t merge`, `const signed char *address`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.19 Load 4 contiguous 16b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg4hu.v`']

**Prototypes:**

- `uint16x4xm1_t vlseg4huv_uint16x4xm1` (`const unsigned short *address`, `unsigned int gvl`)
- `uint16x4xm2_t vlseg4huv_uint16x4xm2` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32x4xm1_t vlseg4huv_uint32x4xm1` (`const unsigned int *address`, `unsigned int gvl`)
- `uint32x4xm2_t vlseg4huv_uint32x4xm2` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64x4xm1_t vlseg4huv_uint64x4xm1` (`const unsigned long *address`, `unsigned int gvl`)
- `uint64x4xm2_t vlseg4huv_uint64x4xm2` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8x4xm1_t vlseg4huv_uint8x4xm1` (`const unsigned char *address`, `unsigned int gvl`)
- `uint8x4xm2_t vlseg4huv_uint8x4xm2` (`const unsigned char *address`, `unsigned int gvl`)

**Operation:**



```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x4xm1_t vlseg4huv_mask_uint16x4xm1` (`uint16x4xm1_t merge`, const unsigned short *\*address*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16x4xm2_t vlseg4huv_mask_uint16x4xm2` (`uint16x4xm2_t merge`, const unsigned short *\*address*, `e16xm2_t mask`, unsigned int *gvl*)
- `uint32x4xm1_t vlseg4huv_mask_uint32x4xm1` (`uint32x4xm1_t merge`, const unsigned int *\*address*, `e32xm1_t mask`, unsigned int *gvl*)
- `uint32x4xm2_t vlseg4huv_mask_uint32x4xm2` (`uint32x4xm2_t merge`, const unsigned int *\*address*, `e32xm2_t mask`, unsigned int *gvl*)
- `uint64x4xm1_t vlseg4huv_mask_uint64x4xm1` (`uint64x4xm1_t merge`, const unsigned long *\*address*, `e64xm1_t mask`, unsigned int *gvl*)
- `uint64x4xm2_t vlseg4huv_mask_uint64x4xm2` (`uint64x4xm2_t merge`, const unsigned long *\*address*, `e64xm2_t mask`, unsigned int *gvl*)
- `uint8x4xm1_t vlseg4huv_mask_uint8x4xm1` (`uint8x4xm1_t merge`, const unsigned char *\*address*, `e8xm1_t mask`, unsigned int *gvl*)
- `uint8x4xm2_t vlseg4huv_mask_uint8x4xm2` (`uint8x4xm2_t merge`, const unsigned char *\*address*, `e8xm2_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.20 Load 4 contiguous 32b signed fields in memory to consecutively numbered vector registers****Instruction:** [`'vlseg4w.v'`]**Prototypes:**

- `int16x4xm1_t vlseg4wv_int16x4xm1` (const short *\*address*, unsigned int *gvl*)
- `int16x4xm2_t vlseg4wv_int16x4xm2` (const short *\*address*, unsigned int *gvl*)
- `int32x4xm1_t vlseg4wv_int32x4xm1` (const int *\*address*, unsigned int *gvl*)
- `int32x4xm2_t vlseg4wv_int32x4xm2` (const int *\*address*, unsigned int *gvl*)
- `int64x4xm1_t vlseg4wv_int64x4xm1` (const long *\*address*, unsigned int *gvl*)
- `int64x4xm2_t vlseg4wv_int64x4xm2` (const long *\*address*, unsigned int *gvl*)
- `int8x4xm1_t vlseg4wv_int8x4xm1` (const signed char *\*address*, unsigned int *gvl*)
- `int8x4xm2_t vlseg4wv_int8x4xm2` (const signed char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      result[segment] = load_segment(address)
      address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x4xm1_t vlseg4wv_mask_int16x4xm1` (`int16x4xm1_t merge`, `const short *address`, `e16xm1_t mask`, `unsigned int gvl`)
- `int16x4xm2_t vlseg4wv_mask_int16x4xm2` (`int16x4xm2_t merge`, `const short *address`, `e16xm2_t mask`, `unsigned int gvl`)
- `int32x4xm1_t vlseg4wv_mask_int32x4xm1` (`int32x4xm1_t merge`, `const int *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `int32x4xm2_t vlseg4wv_mask_int32x4xm2` (`int32x4xm2_t merge`, `const int *address`, `e32xm2_t mask`, `unsigned int gvl`)
- `int64x4xm1_t vlseg4wv_mask_int64x4xm1` (`int64x4xm1_t merge`, `const long *address`, `e64xm1_t mask`, `unsigned int gvl`)
- `int64x4xm2_t vlseg4wv_mask_int64x4xm2` (`int64x4xm2_t merge`, `const long *address`, `e64xm2_t mask`, `unsigned int gvl`)
- `int8x4xm1_t vlseg4wv_mask_int8x4xm1` (`int8x4xm1_t merge`, `const signed char *address`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8x4xm2_t vlseg4wv_mask_int8x4xm2` (`int8x4xm2_t merge`, `const signed char *address`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
      else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.21 Load 4 contiguous 32b unsigned fields in memory to consecutively numbered vector registers****Instruction:** [`'vlseg4wuv.v'`]**Prototypes:**

- `uint16x4xm1_t vlseg4wuv_uint16x4xm1` (`const unsigned short *address`, `unsigned int gvl`)
- `uint16x4xm2_t vlseg4wuv_uint16x4xm2` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32x4xm1_t vlseg4wuv_uint32x4xm1` (`const unsigned int *address`, `unsigned int gvl`)
- `uint32x4xm2_t vlseg4wuv_uint32x4xm2` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64x4xm1_t vlseg4wuv_uint64x4xm1` (`const unsigned long *address`, `unsigned int gvl`)
- `uint64x4xm2_t vlseg4wuv_uint64x4xm2` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8x4xm1_t vlseg4wuv_uint8x4xm1` (`const unsigned char *address`, `unsigned int gvl`)

- `uint8x4xm2_t vlseg4wuv_uint8x4xm2` (const unsigned char \**address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x4xm1_t vlseg4wuv_mask_uint16x4xm1` (`uint16x4xm1_t merge`, const unsigned short \**address*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16x4xm2_t vlseg4wuv_mask_uint16x4xm2` (`uint16x4xm2_t merge`, const unsigned short \**address*, `e16xm2_t mask`, unsigned int *gvl*)
- `uint32x4xm1_t vlseg4wuv_mask_uint32x4xm1` (`uint32x4xm1_t merge`, const unsigned int \**address*, `e32xm1_t mask`, unsigned int *gvl*)
- `uint32x4xm2_t vlseg4wuv_mask_uint32x4xm2` (`uint32x4xm2_t merge`, const unsigned int \**address*, `e32xm2_t mask`, unsigned int *gvl*)
- `uint64x4xm1_t vlseg4wuv_mask_uint64x4xm1` (`uint64x4xm1_t merge`, const unsigned long \**address*, `e64xm1_t mask`, unsigned int *gvl*)
- `uint64x4xm2_t vlseg4wuv_mask_uint64x4xm2` (`uint64x4xm2_t merge`, const unsigned long \**address*, `e64xm2_t mask`, unsigned int *gvl*)
- `uint8x4xm1_t vlseg4wuv_mask_uint8x4xm1` (`uint8x4xm1_t merge`, const unsigned char \**address*, `e8xm1_t mask`, unsigned int *gvl*)
- `uint8x4xm2_t vlseg4wuv_mask_uint8x4xm2` (`uint8x4xm2_t merge`, const unsigned char \**address*, `e8xm2_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.22 Load 5 contiguous 8b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg5b.v`]

**Prototypes:**

- `int16x5xm1_t vlseg5bv_int16x5xm1` (const short \**address*, unsigned int *gvl*)
- `int32x5xm1_t vlseg5bv_int32x5xm1` (const int \**address*, unsigned int *gvl*)
- `int64x5xm1_t vlseg5bv_int64x5xm1` (const long \**address*, unsigned int *gvl*)
- `int8x5xm1_t vlseg5bv_int8x5xm1` (const signed char \**address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x5xml_t vlseg5bv_mask_int16x5xml` (`int16x5xml_t merge`, `const short *address`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x5xml_t vlseg5bv_mask_int32x5xml` (`int32x5xml_t merge`, `const int *address`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x5xml_t vlseg5bv_mask_int64x5xml` (`int64x5xml_t merge`, `const long *address`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x5xml_t vlseg5bv_mask_int8x5xml` (`int8x5xml_t merge`, `const signed char *address`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.23 Load 5 contiguous 8b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg5bu.v'`]

**Prototypes:**

- `uint16x5xml_t vlseg5buv_uint16x5xml` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32x5xml_t vlseg5buv_uint32x5xml` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64x5xml_t vlseg5buv_uint64x5xml` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8x5xml_t vlseg5buv_uint8x5xml` (`const unsigned char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x5xml_t vlseg5buv_mask_uint16x5xml` (`uint16x5xml_t merge`, `const unsigned short *address`, `e16xml_t mask`, `unsigned int gvl`)
- `uint32x5xml_t vlseg5buv_mask_uint32x5xml` (`uint32x5xml_t merge`, `const unsigned int *address`, `e32xml_t mask`, `unsigned int gvl`)
- `uint64x5xml_t vlseg5buv_mask_uint64x5xml` (`uint64x5xml_t merge`, `const unsigned long *address`, `e64xml_t mask`, `unsigned int gvl`)

- `uint8x5xm1_t vlseg5buvmask_uint8x5xm1` (`uint8x5xm1_t merge`, `const unsigned char *address`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.24 Load 5 contiguous element fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg5e.v`]

**Prototypes:**

- `float16x5xm1_t vlseg5ev_float16x5xm1` (`const float16_t *address`, `unsigned int gvl`)
- `float32x5xm1_t vlseg5ev_float32x5xm1` (`const float *address`, `unsigned int gvl`)
- `float64x5xm1_t vlseg5ev_float64x5xm1` (`const double *address`, `unsigned int gvl`)
- `int16x5xm1_t vlseg5ev_int16x5xm1` (`const short *address`, `unsigned int gvl`)
- `int32x5xm1_t vlseg5ev_int32x5xm1` (`const int *address`, `unsigned int gvl`)
- `int64x5xm1_t vlseg5ev_int64x5xm1` (`const long *address`, `unsigned int gvl`)
- `int8x5xm1_t vlseg5ev_int8x5xm1` (`const signed char *address`, `unsigned int gvl`)
- `uint16x5xm1_t vlseg5ev_uint16x5xm1` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32x5xm1_t vlseg5ev_uint32x5xm1` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64x5xm1_t vlseg5ev_uint64x5xm1` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8x5xm1_t vlseg5ev_uint8x5xm1` (`const unsigned char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x5xm1_t vlseg5ev_mask_float16x5xm1` (`float16x5xm1_t merge`, `const float16_t *address`, `e16xm1_t mask`, `unsigned int gvl`)
- `float32x5xm1_t vlseg5ev_mask_float32x5xm1` (`float32x5xm1_t merge`, `const float *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `float64x5xm1_t vlseg5ev_mask_float64x5xm1` (`float64x5xm1_t merge`, `const double *address`, `e64xm1_t mask`, `unsigned int gvl`)
- `int16x5xm1_t vlseg5ev_mask_int16x5xm1` (`int16x5xm1_t merge`, `const short *address`, `e16xm1_t mask`, `unsigned int gvl`)

- `int32x5xml_t vlseg5ev_mask_int32x5xml` (`int32x5xml_t merge`, `const int *address`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x5xml_t vlseg5ev_mask_int64x5xml` (`int64x5xml_t merge`, `const long *address`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x5xml_t vlseg5ev_mask_int8x5xml` (`int8x5xml_t merge`, `const signed char *address`, `e8xml_t mask`, `unsigned int gvl`)
- `uint16x5xml_t vlseg5ev_mask_uint16x5xml` (`uint16x5xml_t merge`, `const unsigned short *address`, `e16xml_t mask`, `unsigned int gvl`)
- `uint32x5xml_t vlseg5ev_mask_uint32x5xml` (`uint32x5xml_t merge`, `const unsigned int *address`, `e32xml_t mask`, `unsigned int gvl`)
- `uint64x5xml_t vlseg5ev_mask_uint64x5xml` (`uint64x5xml_t merge`, `const unsigned long *address`, `e64xml_t mask`, `unsigned int gvl`)
- `uint8x5xml_t vlseg5ev_mask_uint8x5xml` (`uint8x5xml_t merge`, `const unsigned char *address`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.25 Load 5 contiguous 16b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg5h.v`]

**Prototypes:**

- `int16x5xml_t vlseg5hv_int16x5xml` (`const short *address`, `unsigned int gvl`)
- `int32x5xml_t vlseg5hv_int32x5xml` (`const int *address`, `unsigned int gvl`)
- `int64x5xml_t vlseg5hv_int64x5xml` (`const long *address`, `unsigned int gvl`)
- `int8x5xml_t vlseg5hv_int8x5xml` (`const signed char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x5xml_t vlseg5hv_mask_int16x5xml` (`int16x5xml_t merge`, `const short *address`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x5xml_t vlseg5hv_mask_int32x5xml` (`int32x5xml_t merge`, `const int *address`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x5xml_t vlseg5hv_mask_int64x5xml` (`int64x5xml_t merge`, `const long *address`, `e64xml_t mask`, `unsigned int gvl`)



- `int8x5xm1_t vlseg5hv_mask_int8x5xm1` (`int8x5xm1_t merge`, `const signed char *address`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.26 Load 5 contiguous 16b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg5hu.v`']

**Prototypes:**

- `uint16x5xm1_t vlseg5huv_uint16x5xm1` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32x5xm1_t vlseg5huv_uint32x5xm1` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64x5xm1_t vlseg5huv_uint64x5xm1` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8x5xm1_t vlseg5huv_uint8x5xm1` (`const unsigned char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x5xm1_t vlseg5huv_mask_uint16x5xm1` (`uint16x5xm1_t merge`, `const unsigned short *address`, `e16xm1_t mask`, `unsigned int gvl`)
- `uint32x5xm1_t vlseg5huv_mask_uint32x5xm1` (`uint32x5xm1_t merge`, `const unsigned int *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `uint64x5xm1_t vlseg5huv_mask_uint64x5xm1` (`uint64x5xm1_t merge`, `const unsigned long *address`, `e64xm1_t mask`, `unsigned int gvl`)
- `uint8x5xm1_t vlseg5huv_mask_uint8x5xm1` (`uint8x5xm1_t merge`, `const unsigned char *address`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.27 Load 5 contiguous 32b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg5w.v'`]

**Prototypes:**

- `int16x5xml_t vlseg5wv_int16x5xml` (const short *\*address*, unsigned int *gvl*)
- `int32x5xml_t vlseg5wv_int32x5xml` (const int *\*address*, unsigned int *gvl*)
- `int64x5xml_t vlseg5wv_int64x5xml` (const long *\*address*, unsigned int *gvl*)
- `int8x5xml_t vlseg5wv_int8x5xml` (const signed char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x5xml_t vlseg5wv_mask_int16x5xml` (`int16x5xml_t merge`, const short *\*address*, `e16xml_t mask`, unsigned int *gvl*)
- `int32x5xml_t vlseg5wv_mask_int32x5xml` (`int32x5xml_t merge`, const int *\*address*, `e32xml_t mask`, unsigned int *gvl*)
- `int64x5xml_t vlseg5wv_mask_int64x5xml` (`int64x5xml_t merge`, const long *\*address*, `e64xml_t mask`, unsigned int *gvl*)
- `int8x5xml_t vlseg5wv_mask_int8x5xml` (`int8x5xml_t merge`, const signed char *\*address*, `e8xml_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.28 Load 5 contiguous 32b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg5wu.v'`]

**Prototypes:**

- `uint16x5xml_t vlseg5wuv_uint16x5xml` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint32x5xml_t vlseg5wuv_uint32x5xml` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64x5xml_t vlseg5wuv_uint64x5xml` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x5xml_t vlseg5wuv_uint8x5xml` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x5xml_t vlseg5wuv_mask_uint16x5xml` (`uint16x5xml_t merge`, const unsigned short *\*address*, *e16xml\_t mask*, unsigned int *gvl*)
- `uint32x5xml_t vlseg5wuv_mask_uint32x5xml` (`uint32x5xml_t merge`, const unsigned int *\*address*, *e32xml\_t mask*, unsigned int *gvl*)
- `uint64x5xml_t vlseg5wuv_mask_uint64x5xml` (`uint64x5xml_t merge`, const unsigned long *\*address*, *e64xml\_t mask*, unsigned int *gvl*)
- `uint8x5xml_t vlseg5wuv_mask_uint8x5xml` (`uint8x5xml_t merge`, const unsigned char *\*address*, *e8xml\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.29 Load 6 contiguous 8b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg6b.v`]**Prototypes:**

- `int16x6xml_t vlseg6bv_int16x6xml` (const short *\*address*, unsigned int *gvl*)
- `int32x6xml_t vlseg6bv_int32x6xml` (const int *\*address*, unsigned int *gvl*)
- `int64x6xml_t vlseg6bv_int64x6xml` (const long *\*address*, unsigned int *gvl*)
- `int8x6xml_t vlseg6bv_int8x6xml` (const signed char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x6xml_t vlseg6bv_mask_int16x6xml` (`int16x6xml_t merge`, const short *\*address*, *e16xml\_t mask*, unsigned int *gvl*)
- `int32x6xml_t vlseg6bv_mask_int32x6xml` (`int32x6xml_t merge`, const int *\*address*, *e32xml\_t mask*, unsigned int *gvl*)
- `int64x6xml_t vlseg6bv_mask_int64x6xml` (`int64x6xml_t merge`, const long *\*address*, *e64xml\_t mask*, unsigned int *gvl*)

- `int8x6xm1_t vlseg6bv_mask_int8x6xm1` (`int8x6xm1_t merge`, `const signed char *address`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.30 Load 6 contiguous 8b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg6bu.v`']

**Prototypes:**

- `uint16x6xm1_t vlseg6buv_uint16x6xm1` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32x6xm1_t vlseg6buv_uint32x6xm1` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64x6xm1_t vlseg6buv_uint64x6xm1` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8x6xm1_t vlseg6buv_uint8x6xm1` (`const unsigned char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x6xm1_t vlseg6buv_mask_uint16x6xm1` (`uint16x6xm1_t merge`, `const unsigned short *address`, `e16xm1_t mask`, `unsigned int gvl`)
- `uint32x6xm1_t vlseg6buv_mask_uint32x6xm1` (`uint32x6xm1_t merge`, `const unsigned int *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `uint64x6xm1_t vlseg6buv_mask_uint64x6xm1` (`uint64x6xm1_t merge`, `const unsigned long *address`, `e64xm1_t mask`, `unsigned int gvl`)
- `uint8x6xm1_t vlseg6buv_mask_uint8x6xm1` (`uint8x6xm1_t merge`, `const unsigned char *address`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.31 Load 6 contiguous element fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg6e.v'`]

**Prototypes:**

- `float16x6xm1_t vlseg6ev_float16x6xm1` (const `float16_t *address`, unsigned int `gvl`)
- `float32x6xm1_t vlseg6ev_float32x6xm1` (const `float *address`, unsigned int `gvl`)
- `float64x6xm1_t vlseg6ev_float64x6xm1` (const `double *address`, unsigned int `gvl`)
- `int16x6xm1_t vlseg6ev_int16x6xm1` (const `short *address`, unsigned int `gvl`)
- `int32x6xm1_t vlseg6ev_int32x6xm1` (const `int *address`, unsigned int `gvl`)
- `int64x6xm1_t vlseg6ev_int64x6xm1` (const `long *address`, unsigned int `gvl`)
- `int8x6xm1_t vlseg6ev_int8x6xm1` (const `signed char *address`, unsigned int `gvl`)
- `uint16x6xm1_t vlseg6ev_uint16x6xm1` (const `unsigned short *address`, unsigned int `gvl`)
- `uint32x6xm1_t vlseg6ev_uint32x6xm1` (const `unsigned int *address`, unsigned int `gvl`)
- `uint64x6xm1_t vlseg6ev_uint64x6xm1` (const `unsigned long *address`, unsigned int `gvl`)
- `uint8x6xm1_t vlseg6ev_uint8x6xm1` (const `unsigned char *address`, unsigned int `gvl`)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x6xm1_t vlseg6ev_mask_float16x6xm1` (`float16x6xm1_t merge`, const `float16_t *address`, `e16xm1_t mask`, unsigned int `gvl`)
- `float32x6xm1_t vlseg6ev_mask_float32x6xm1` (`float32x6xm1_t merge`, const `float *address`, `e32xm1_t mask`, unsigned int `gvl`)
- `float64x6xm1_t vlseg6ev_mask_float64x6xm1` (`float64x6xm1_t merge`, const `double *address`, `e64xm1_t mask`, unsigned int `gvl`)
- `int16x6xm1_t vlseg6ev_mask_int16x6xm1` (`int16x6xm1_t merge`, const `short *address`, `e16xm1_t mask`, unsigned int `gvl`)
- `int32x6xm1_t vlseg6ev_mask_int32x6xm1` (`int32x6xm1_t merge`, const `int *address`, `e32xm1_t mask`, unsigned int `gvl`)
- `int64x6xm1_t vlseg6ev_mask_int64x6xm1` (`int64x6xm1_t merge`, const `long *address`, `e64xm1_t mask`, unsigned int `gvl`)
- `int8x6xm1_t vlseg6ev_mask_int8x6xm1` (`int8x6xm1_t merge`, const `signed char *address`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint16x6xm1_t vlseg6ev_mask_uint16x6xm1` (`uint16x6xm1_t merge`, const `unsigned short *address`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint32x6xm1_t vlseg6ev_mask_uint32x6xm1` (`uint32x6xm1_t merge`, const `unsigned int *address`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint64x6xm1_t vlseg6ev_mask_uint64x6xm1` (`uint64x6xm1_t merge`, const `unsigned long *address`, `e64xm1_t mask`, unsigned int `gvl`)

- `uint8x6xm1_t vlseg6ev_mask_uint8x6xm1` (`uint8x6xm1_t merge`, `const unsigned char *address`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.32 Load 6 contiguous 16b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg6h.v`']

**Prototypes:**

- `int16x6xm1_t vlseg6hv_int16x6xm1` (`const short *address`, `unsigned int gvl`)
- `int32x6xm1_t vlseg6hv_int32x6xm1` (`const int *address`, `unsigned int gvl`)
- `int64x6xm1_t vlseg6hv_int64x6xm1` (`const long *address`, `unsigned int gvl`)
- `int8x6xm1_t vlseg6hv_int8x6xm1` (`const signed char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x6xm1_t vlseg6hv_mask_int16x6xm1` (`int16x6xm1_t merge`, `const short *address`, `e16xm1_t mask`, `unsigned int gvl`)
- `int32x6xm1_t vlseg6hv_mask_int32x6xm1` (`int32x6xm1_t merge`, `const int *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `int64x6xm1_t vlseg6hv_mask_int64x6xm1` (`int64x6xm1_t merge`, `const long *address`, `e64xm1_t mask`, `unsigned int gvl`)
- `int8x6xm1_t vlseg6hv_mask_int8x6xm1` (`int8x6xm1_t merge`, `const signed char *address`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```



## 2.10.33 Load 6 contiguous 16b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg6hu.v'`]

**Prototypes:**

- `uint16x6xml_t vlseg6huv_uint16x6xml` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint32x6xml_t vlseg6huv_uint32x6xml` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64x6xml_t vlseg6huv_uint64x6xml` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x6xml_t vlseg6huv_uint8x6xml` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x6xml_t vlseg6huv_mask_uint16x6xml` (uint16x6xml\_t *merge*, const unsigned short *\*address*, *e16xml\_t* mask, unsigned int *gvl*)
- `uint32x6xml_t vlseg6huv_mask_uint32x6xml` (uint32x6xml\_t *merge*, const unsigned int *\*address*, *e32xml\_t* mask, unsigned int *gvl*)
- `uint64x6xml_t vlseg6huv_mask_uint64x6xml` (uint64x6xml\_t *merge*, const unsigned long *\*address*, *e64xml\_t* mask, unsigned int *gvl*)
- `uint8x6xml_t vlseg6huv_mask_uint8x6xml` (uint8x6xml\_t *merge*, const unsigned char *\*address*, *e8xml\_t* mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.34 Load 6 contiguous 32b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg6w.v'`]

**Prototypes:**

- `int16x6xml_t vlseg6wv_int16x6xml` (const short *\*address*, unsigned int *gvl*)
- `int32x6xml_t vlseg6wv_int32x6xml` (const int *\*address*, unsigned int *gvl*)
- `int64x6xml_t vlseg6wv_int64x6xml` (const long *\*address*, unsigned int *gvl*)
- `int8x6xml_t vlseg6wv_int8x6xml` (const signed char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x6xml_t vlseg6wv_mask_int16x6xml` (`int16x6xml_t merge`, `const short *address`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x6xml_t vlseg6wv_mask_int32x6xml` (`int32x6xml_t merge`, `const int *address`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x6xml_t vlseg6wv_mask_int64x6xml` (`int64x6xml_t merge`, `const long *address`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x6xml_t vlseg6wv_mask_int8x6xml` (`int8x6xml_t merge`, `const signed char *address`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.35 Load 6 contiguous 32b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg6wu.v`']

**Prototypes:**

- `uint16x6xml_t vlseg6wuv_uint16x6xml` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32x6xml_t vlseg6wuv_uint32x6xml` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64x6xml_t vlseg6wuv_uint64x6xml` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8x6xml_t vlseg6wuv_uint8x6xml` (`const unsigned char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x6xml_t vlseg6wuv_mask_uint16x6xml` (`uint16x6xml_t merge`, `const unsigned short *address`, `e16xml_t mask`, `unsigned int gvl`)
- `uint32x6xml_t vlseg6wuv_mask_uint32x6xml` (`uint32x6xml_t merge`, `const unsigned int *address`, `e32xml_t mask`, `unsigned int gvl`)
- `uint64x6xml_t vlseg6wuv_mask_uint64x6xml` (`uint64x6xml_t merge`, `const unsigned long *address`, `e64xml_t mask`, `unsigned int gvl`)

- `uint8x6xm1_t vlseg6wuv_mask_uint8x6xm1` (`uint8x6xm1_t merge`, `const unsigned char *address`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.36 Load 7 contiguous 8b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg7b.v`]

**Prototypes:**

- `int16x7xm1_t vlseg7bv_int16x7xm1` (`const short *address`, `unsigned int gvl`)
- `int32x7xm1_t vlseg7bv_int32x7xm1` (`const int *address`, `unsigned int gvl`)
- `int64x7xm1_t vlseg7bv_int64x7xm1` (`const long *address`, `unsigned int gvl`)
- `int8x7xm1_t vlseg7bv_int8x7xm1` (`const signed char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x7xm1_t vlseg7bv_mask_int16x7xm1` (`int16x7xm1_t merge`, `const short *address`, `e16xm1_t mask`, `unsigned int gvl`)
- `int32x7xm1_t vlseg7bv_mask_int32x7xm1` (`int32x7xm1_t merge`, `const int *address`, `e32xm1_t mask`, `unsigned int gvl`)
- `int64x7xm1_t vlseg7bv_mask_int64x7xm1` (`int64x7xm1_t merge`, `const long *address`, `e64xm1_t mask`, `unsigned int gvl`)
- `int8x7xm1_t vlseg7bv_mask_int8x7xm1` (`int8x7xm1_t merge`, `const signed char *address`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.37 Load 7 contiguous 8b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg7bu.v'`]

**Prototypes:**

- `uint16x7xml_t vlseg7bu_v_uint16x7xml` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint32x7xml_t vlseg7bu_v_uint32x7xml` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64x7xml_t vlseg7bu_v_uint64x7xml` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x7xml_t vlseg7bu_v_uint8x7xml` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x7xml_t vlseg7bu_v_mask_uint16x7xml` (uint16x7xml\_t *merge*, const unsigned short *\*address*, *e16xml\_t* mask, unsigned int *gvl*)
- `uint32x7xml_t vlseg7bu_v_mask_uint32x7xml` (uint32x7xml\_t *merge*, const unsigned int *\*address*, *e32xml\_t* mask, unsigned int *gvl*)
- `uint64x7xml_t vlseg7bu_v_mask_uint64x7xml` (uint64x7xml\_t *merge*, const unsigned long *\*address*, *e64xml\_t* mask, unsigned int *gvl*)
- `uint8x7xml_t vlseg7bu_v_mask_uint8x7xml` (uint8x7xml\_t *merge*, const unsigned char *\*address*, *e8xml\_t* mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.38 Load 7 contiguous element fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg7e.v'`]

**Prototypes:**

- `float16x7xml_t vlseg7ev_float16x7xml` (const float16\_t *\*address*, unsigned int *gvl*)
- `float32x7xml_t vlseg7ev_float32x7xml` (const float *\*address*, unsigned int *gvl*)
- `float64x7xml_t vlseg7ev_float64x7xml` (const double *\*address*, unsigned int *gvl*)
- `int16x7xml_t vlseg7ev_int16x7xml` (const short *\*address*, unsigned int *gvl*)
- `int32x7xml_t vlseg7ev_int32x7xml` (const int *\*address*, unsigned int *gvl*)

- `int64x7xml_t vlseg7ev_int64x7xml` (const long *\*address*, unsigned int *gvl*)
- `int8x7xml_t vlseg7ev_int8x7xml` (const signed char *\*address*, unsigned int *gvl*)
- `uint16x7xml_t vlseg7ev_uint16x7xml` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint32x7xml_t vlseg7ev_uint32x7xml` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64x7xml_t vlseg7ev_uint64x7xml` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x7xml_t vlseg7ev_uint8x7xml` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x7xml_t vlseg7ev_mask_float16x7xml` (float16x7xml\_t *merge*, const float16\_t *\*address*, *e16xml\_t mask*, unsigned int *gvl*)
- `float32x7xml_t vlseg7ev_mask_float32x7xml` (float32x7xml\_t *merge*, const float *\*address*, *e32xml\_t mask*, unsigned int *gvl*)
- `float64x7xml_t vlseg7ev_mask_float64x7xml` (float64x7xml\_t *merge*, const double *\*address*, *e64xml\_t mask*, unsigned int *gvl*)
- `int16x7xml_t vlseg7ev_mask_int16x7xml` (int16x7xml\_t *merge*, const short *\*address*, *e16xml\_t mask*, unsigned int *gvl*)
- `int32x7xml_t vlseg7ev_mask_int32x7xml` (int32x7xml\_t *merge*, const int *\*address*, *e32xml\_t mask*, unsigned int *gvl*)
- `int64x7xml_t vlseg7ev_mask_int64x7xml` (int64x7xml\_t *merge*, const long *\*address*, *e64xml\_t mask*, unsigned int *gvl*)
- `int8x7xml_t vlseg7ev_mask_int8x7xml` (int8x7xml\_t *merge*, const signed char *\*address*, *e8xml\_t mask*, unsigned int *gvl*)
- `uint16x7xml_t vlseg7ev_mask_uint16x7xml` (uint16x7xml\_t *merge*, const unsigned short *\*address*, *e16xml\_t mask*, unsigned int *gvl*)
- `uint32x7xml_t vlseg7ev_mask_uint32x7xml` (uint32x7xml\_t *merge*, const unsigned int *\*address*, *e32xml\_t mask*, unsigned int *gvl*)
- `uint64x7xml_t vlseg7ev_mask_uint64x7xml` (uint64x7xml\_t *merge*, const unsigned long *\*address*, *e64xml\_t mask*, unsigned int *gvl*)
- `uint8x7xml_t vlseg7ev_mask_uint8x7xml` (uint8x7xml\_t *merge*, const unsigned char *\*address*, *e8xml\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.39 Load 7 contiguous 16b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg7h.v'`]

**Prototypes:**

- `int16x7xml_t vlseg7hv_int16x7xml` (const short *\*address*, unsigned int *gvl*)
- `int32x7xml_t vlseg7hv_int32x7xml` (const int *\*address*, unsigned int *gvl*)
- `int64x7xml_t vlseg7hv_int64x7xml` (const long *\*address*, unsigned int *gvl*)
- `int8x7xml_t vlseg7hv_int8x7xml` (const signed char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x7xml_t vlseg7hv_mask_int16x7xml` (`int16x7xml_t merge`, const short *\*address*, `e16xml_t mask`, unsigned int *gvl*)
- `int32x7xml_t vlseg7hv_mask_int32x7xml` (`int32x7xml_t merge`, const int *\*address*, `e32xml_t mask`, unsigned int *gvl*)
- `int64x7xml_t vlseg7hv_mask_int64x7xml` (`int64x7xml_t merge`, const long *\*address*, `e64xml_t mask`, unsigned int *gvl*)
- `int8x7xml_t vlseg7hv_mask_int8x7xml` (`int8x7xml_t merge`, const signed char *\*address*, `e8xml_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.40 Load 7 contiguous 16b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg7hu.v'`]

**Prototypes:**

- `uint16x7xml_t vlseg7huv_uint16x7xml` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint32x7xml_t vlseg7huv_uint32x7xml` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64x7xml_t vlseg7huv_uint64x7xml` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x7xml_t vlseg7huv_uint8x7xml` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**



```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x7xml_t vlseg7huv_mask_uint16x7xml` (`uint16x7xml_t merge`, const unsigned short *\*address*, *e16xml\_t mask*, unsigned int *gvl*)
- `uint32x7xml_t vlseg7huv_mask_uint32x7xml` (`uint32x7xml_t merge`, const unsigned int *\*address*, *e32xml\_t mask*, unsigned int *gvl*)
- `uint64x7xml_t vlseg7huv_mask_uint64x7xml` (`uint64x7xml_t merge`, const unsigned long *\*address*, *e64xml\_t mask*, unsigned int *gvl*)
- `uint8x7xml_t vlseg7huv_mask_uint8x7xml` (`uint8x7xml_t merge`, const unsigned char *\*address*, *e8xml\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.41 Load 7 contiguous 32b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg7w.v`]

**Prototypes:**

- `int16x7xml_t vlseg7wv_int16x7xml` (const short *\*address*, unsigned int *gvl*)
- `int32x7xml_t vlseg7wv_int32x7xml` (const int *\*address*, unsigned int *gvl*)
- `int64x7xml_t vlseg7wv_int64x7xml` (const long *\*address*, unsigned int *gvl*)
- `int8x7xml_t vlseg7wv_int8x7xml` (const signed char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x7xml_t vlseg7wv_mask_int16x7xml` (`int16x7xml_t merge`, const short *\*address*, *e16xml\_t mask*, unsigned int *gvl*)
- `int32x7xml_t vlseg7wv_mask_int32x7xml` (`int32x7xml_t merge`, const int *\*address*, *e32xml\_t mask*, unsigned int *gvl*)
- `int64x7xml_t vlseg7wv_mask_int64x7xml` (`int64x7xml_t merge`, const long *\*address*, *e64xml\_t mask*, unsigned int *gvl*)

- `int8x7xml_t vlseg7wv_mask_int8x7xml` (`int8x7xml_t merge`, `const signed char *address`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.42 Load 7 contiguous 32b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg7wu.v`']

**Prototypes:**

- `uint16x7xml_t vlseg7wuv_uint16x7xml` (`const unsigned short *address`, `unsigned int gvl`)
- `uint32x7xml_t vlseg7wuv_uint32x7xml` (`const unsigned int *address`, `unsigned int gvl`)
- `uint64x7xml_t vlseg7wuv_uint64x7xml` (`const unsigned long *address`, `unsigned int gvl`)
- `uint8x7xml_t vlseg7wuv_uint8x7xml` (`const unsigned char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x7xml_t vlseg7wuv_mask_uint16x7xml` (`uint16x7xml_t merge`, `const unsigned short *address`, `e16xml_t mask`, `unsigned int gvl`)
- `uint32x7xml_t vlseg7wuv_mask_uint32x7xml` (`uint32x7xml_t merge`, `const unsigned int *address`, `e32xml_t mask`, `unsigned int gvl`)
- `uint64x7xml_t vlseg7wuv_mask_uint64x7xml` (`uint64x7xml_t merge`, `const unsigned long *address`, `e64xml_t mask`, `unsigned int gvl`)
- `uint8x7xml_t vlseg7wuv_mask_uint8x7xml` (`uint8x7xml_t merge`, `const unsigned char *address`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.43 Load 8 contiguous 8b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg8b.v'`]

**Prototypes:**

- `int16x8xml_t vlseg8bv_int16x8xml` (const short *\*address*, unsigned int *gvl*)
- `int32x8xml_t vlseg8bv_int32x8xml` (const int *\*address*, unsigned int *gvl*)
- `int64x8xml_t vlseg8bv_int64x8xml` (const long *\*address*, unsigned int *gvl*)
- `int8x8xml_t vlseg8bv_int8x8xml` (const signed char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x8xml_t vlseg8bv_mask_int16x8xml` (`int16x8xml_t merge`, const short *\*address*, `e16xml_t mask`, unsigned int *gvl*)
- `int32x8xml_t vlseg8bv_mask_int32x8xml` (`int32x8xml_t merge`, const int *\*address*, `e32xml_t mask`, unsigned int *gvl*)
- `int64x8xml_t vlseg8bv_mask_int64x8xml` (`int64x8xml_t merge`, const long *\*address*, `e64xml_t mask`, unsigned int *gvl*)
- `int8x8xml_t vlseg8bv_mask_int8x8xml` (`int8x8xml_t merge`, const signed char *\*address*, `e8xml_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.44 Load 8 contiguous 8b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`'vlseg8bu.v'`]

**Prototypes:**

- `uint16x8xml_t vlseg8buv_uint16x8xml` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint32x8xml_t vlseg8buv_uint32x8xml` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64x8xml_t vlseg8buv_uint64x8xml` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x8xml_t vlseg8buv_uint8x8xml` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x8xml_t vlseg8buv_mask_uint16x8xml` (`uint16x8xml_t merge`, const unsigned short *\*address*, `e16xml_t mask`, unsigned int *gvl*)
- `uint32x8xml_t vlseg8buv_mask_uint32x8xml` (`uint32x8xml_t merge`, const unsigned int *\*address*, `e32xml_t mask`, unsigned int *gvl*)
- `uint64x8xml_t vlseg8buv_mask_uint64x8xml` (`uint64x8xml_t merge`, const unsigned long *\*address*, `e64xml_t mask`, unsigned int *gvl*)
- `uint8x8xml_t vlseg8buv_mask_uint8x8xml` (`uint8x8xml_t merge`, const unsigned char *\*address*, `e8xml_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.45 Load 8 contiguous element fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg8e.v`]

**Prototypes:**

- `float16x8xml_t vlseg8ev_float16x8xml` (const float16\_t *\*address*, unsigned int *gvl*)
- `float32x8xml_t vlseg8ev_float32x8xml` (const float *\*address*, unsigned int *gvl*)
- `float64x8xml_t vlseg8ev_float64x8xml` (const double *\*address*, unsigned int *gvl*)
- `int16x8xml_t vlseg8ev_int16x8xml` (const short *\*address*, unsigned int *gvl*)
- `int32x8xml_t vlseg8ev_int32x8xml` (const int *\*address*, unsigned int *gvl*)
- `int64x8xml_t vlseg8ev_int64x8xml` (const long *\*address*, unsigned int *gvl*)
- `int8x8xml_t vlseg8ev_int8x8xml` (const signed char *\*address*, unsigned int *gvl*)
- `uint16x8xml_t vlseg8ev_uint16x8xml` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint32x8xml_t vlseg8ev_uint32x8xml` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64x8xml_t vlseg8ev_uint64x8xml` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x8xml_t vlseg8ev_uint8x8xml` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- float16x8xml\_t **vlseg8ev\_mask\_float16x8xml** (float16x8xml\_t merge, const float16\_t \*address, e16xml\_t mask, unsigned int gvl)
- float32x8xml\_t **vlseg8ev\_mask\_float32x8xml** (float32x8xml\_t merge, const float \*address, e32xml\_t mask, unsigned int gvl)
- float64x8xml\_t **vlseg8ev\_mask\_float64x8xml** (float64x8xml\_t merge, const double \*address, e64xml\_t mask, unsigned int gvl)
- int16x8xml\_t **vlseg8ev\_mask\_int16x8xml** (int16x8xml\_t merge, const short \*address, e16xml\_t mask, unsigned int gvl)
- int32x8xml\_t **vlseg8ev\_mask\_int32x8xml** (int32x8xml\_t merge, const int \*address, e32xml\_t mask, unsigned int gvl)
- int64x8xml\_t **vlseg8ev\_mask\_int64x8xml** (int64x8xml\_t merge, const long \*address, e64xml\_t mask, unsigned int gvl)
- int8x8xml\_t **vlseg8ev\_mask\_int8x8xml** (int8x8xml\_t merge, const signed char \*address, e8xml\_t mask, unsigned int gvl)
- uint16x8xml\_t **vlseg8ev\_mask\_uint16x8xml** (uint16x8xml\_t merge, const unsigned short \*address, e16xml\_t mask, unsigned int gvl)
- uint32x8xml\_t **vlseg8ev\_mask\_uint32x8xml** (uint32x8xml\_t merge, const unsigned int \*address, e32xml\_t mask, unsigned int gvl)
- uint64x8xml\_t **vlseg8ev\_mask\_uint64x8xml** (uint64x8xml\_t merge, const unsigned long \*address, e64xml\_t mask, unsigned int gvl)
- uint8x8xml\_t **vlseg8ev\_mask\_uint8x8xml** (uint8x8xml\_t merge, const unsigned char \*address, e8xml\_t mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.46 Load 8 contiguous 16b signed fields in memory to consecutively numbered vector registers****Instruction:** [‘vlseg8h.v’]**Prototypes:**

- int16x8xml\_t **vlseg8hv\_int16x8xml** (const short \*address, unsigned int gvl)
- int32x8xml\_t **vlseg8hv\_int32x8xml** (const int \*address, unsigned int gvl)
- int64x8xml\_t **vlseg8hv\_int64x8xml** (const long \*address, unsigned int gvl)

- `int8x8xml_t vlseg8hv_int8x8xml` (const signed char \**address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x8xml_t vlseg8hv_mask_int16x8xml` (int16x8xml\_t *merge*, const short \**address*, *e16xml\_t mask*, unsigned int *gvl*)
- `int32x8xml_t vlseg8hv_mask_int32x8xml` (int32x8xml\_t *merge*, const int \**address*, *e32xml\_t mask*, unsigned int *gvl*)
- `int64x8xml_t vlseg8hv_mask_int64x8xml` (int64x8xml\_t *merge*, const long \**address*, *e64xml\_t mask*, unsigned int *gvl*)
- `int8x8xml_t vlseg8hv_mask_int8x8xml` (int8x8xml\_t *merge*, const signed char \**address*, *e8xml\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.47 Load 8 contiguous 16b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg8hu.v`']

**Prototypes:**

- `uint16x8xml_t vlseg8huv_uint16x8xml` (const unsigned short \**address*, unsigned int *gvl*)
- `uint32x8xml_t vlseg8huv_uint32x8xml` (const unsigned int \**address*, unsigned int *gvl*)
- `uint64x8xml_t vlseg8huv_uint64x8xml` (const unsigned long \**address*, unsigned int *gvl*)
- `uint8x8xml_t vlseg8huv_uint8x8xml` (const unsigned char \**address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x8xml_t vlseg8huv_mask_uint16x8xml` (uint16x8xml\_t *merge*, const unsigned short \**address*, *e16xml\_t mask*, unsigned int *gvl*)
- `uint32x8xml_t vlseg8huv_mask_uint32x8xml` (uint32x8xml\_t *merge*, const unsigned int \**address*, *e32xml\_t mask*, unsigned int *gvl*)



- `uint64x8xml_t vlseg8huv_mask_uint64x8xml` (`uint64x8xml_t merge`, `const unsigned long *address`, `e64xml_t mask`, `unsigned int gvl`)
- `uint8x8xml_t vlseg8huv_mask_uint8x8xml` (`uint8x8xml_t merge`, `const unsigned char *address`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.48 Load 8 contiguous 32b signed fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg8w.v`']

**Prototypes:**

- `int16x8xml_t vlseg8wv_int16x8xml` (`const short *address`, `unsigned int gvl`)
- `int32x8xml_t vlseg8wv_int32x8xml` (`const int *address`, `unsigned int gvl`)
- `int64x8xml_t vlseg8wv_int64x8xml` (`const long *address`, `unsigned int gvl`)
- `int8x8xml_t vlseg8wv_int8x8xml` (`const signed char *address`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x8xml_t vlseg8wv_mask_int16x8xml` (`int16x8xml_t merge`, `const short *address`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x8xml_t vlseg8wv_mask_int32x8xml` (`int32x8xml_t merge`, `const int *address`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x8xml_t vlseg8wv_mask_int64x8xml` (`int64x8xml_t merge`, `const long *address`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x8xml_t vlseg8wv_mask_int8x8xml` (`int8x8xml_t merge`, `const signed char *address`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.49 Load 8 contiguous 32b unsigned fields in memory to consecutively numbered vector registers

**Instruction:** [`vlseg8wu.v`]

**Prototypes:**

- `uint16x8xml_t vlseg8wuv_uint16x8xml` (const unsigned short *\*address*, unsigned int *gvl*)
- `uint32x8xml_t vlseg8wuv_uint32x8xml` (const unsigned int *\*address*, unsigned int *gvl*)
- `uint64x8xml_t vlseg8wuv_uint64x8xml` (const unsigned long *\*address*, unsigned int *gvl*)
- `uint8x8xml_t vlseg8wuv_uint8x8xml` (const unsigned char *\*address*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment)
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x8xml_t vlseg8wuv_mask_uint16x8xml` (uint16x8xml\_t *merge*, const unsigned short *\*address*, *e16xml\_t* *mask*, unsigned int *gvl*)
- `uint32x8xml_t vlseg8wuv_mask_uint32x8xml` (uint32x8xml\_t *merge*, const unsigned int *\*address*, *e32xml\_t* *mask*, unsigned int *gvl*)
- `uint64x8xml_t vlseg8wuv_mask_uint64x8xml` (uint64x8xml\_t *merge*, const unsigned long *\*address*, *e64xml\_t* *mask*, unsigned int *gvl*)
- `uint8x8xml_t vlseg8wuv_mask_uint8x8xml` (uint8x8xml\_t *merge*, const unsigned char *\*address*, *e8xml\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment)
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.50 Load 2 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg2b.v`]

**Prototypes:**

- `int16x2xml_t vlsseg2bv_int16x2xml` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16x2xml2_t vlsseg2bv_int16x2xml2` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16x2xml4_t vlsseg2bv_int16x2xml4` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int32x2xml_t vlsseg2bv_int32x2xml` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32x2xml2_t vlsseg2bv_int32x2xml2` (const int *\*address*, long *stride*, unsigned int *gvl*)

- `int32x2xm4_t vlsseg2bv_int32x2xm4` (const int \*address, long stride, unsigned int gvl)
- `int64x2xm1_t vlsseg2bv_int64x2xm1` (const long \*address, long stride, unsigned int gvl)
- `int64x2xm2_t vlsseg2bv_int64x2xm2` (const long \*address, long stride, unsigned int gvl)
- `int64x2xm4_t vlsseg2bv_int64x2xm4` (const long \*address, long stride, unsigned int gvl)
- `int8x2xm1_t vlsseg2bv_int8x2xm1` (const signed char \*address, long stride, unsigned int gvl)
- `int8x2xm2_t vlsseg2bv_int8x2xm2` (const signed char \*address, long stride, unsigned int gvl)
- `int8x2xm4_t vlsseg2bv_int8x2xm4` (const signed char \*address, long stride, unsigned int gvl)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x2xm1_t vlsseg2bv_mask_int16x2xm1` (int16x2xm1\_t merge, const short \*address, long stride, *e16xm1\_t* mask, unsigned int gvl)
- `int16x2xm2_t vlsseg2bv_mask_int16x2xm2` (int16x2xm2\_t merge, const short \*address, long stride, *e16xm2\_t* mask, unsigned int gvl)
- `int16x2xm4_t vlsseg2bv_mask_int16x2xm4` (int16x2xm4\_t merge, const short \*address, long stride, *e16xm4\_t* mask, unsigned int gvl)
- `int32x2xm1_t vlsseg2bv_mask_int32x2xm1` (int32x2xm1\_t merge, const int \*address, long stride, *e32xm1\_t* mask, unsigned int gvl)
- `int32x2xm2_t vlsseg2bv_mask_int32x2xm2` (int32x2xm2\_t merge, const int \*address, long stride, *e32xm2\_t* mask, unsigned int gvl)
- `int32x2xm4_t vlsseg2bv_mask_int32x2xm4` (int32x2xm4\_t merge, const int \*address, long stride, *e32xm4\_t* mask, unsigned int gvl)
- `int64x2xm1_t vlsseg2bv_mask_int64x2xm1` (int64x2xm1\_t merge, const long \*address, long stride, *e64xm1\_t* mask, unsigned int gvl)
- `int64x2xm2_t vlsseg2bv_mask_int64x2xm2` (int64x2xm2\_t merge, const long \*address, long stride, *e64xm2\_t* mask, unsigned int gvl)
- `int64x2xm4_t vlsseg2bv_mask_int64x2xm4` (int64x2xm4\_t merge, const long \*address, long stride, *e64xm4\_t* mask, unsigned int gvl)
- `int8x2xm1_t vlsseg2bv_mask_int8x2xm1` (int8x2xm1\_t merge, const signed char \*address, long stride, *e8xm1\_t* mask, unsigned int gvl)
- `int8x2xm2_t vlsseg2bv_mask_int8x2xm2` (int8x2xm2\_t merge, const signed char \*address, long stride, *e8xm2\_t* mask, unsigned int gvl)
- `int8x2xm4_t vlsseg2bv_mask_int8x2xm4` (int8x2xm4\_t merge, const signed char \*address, long stride, *e8xm4\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
```

(continues on next page)

(continued from previous page)

```
result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.51 Load 2 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg2bu.v'`]

**Prototypes:**

- `uint16x2xm1_t vlsseg2bu_v_uint16x2xm1` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint16x2xm2_t vlsseg2bu_v_uint16x2xm2` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint16x2xm4_t vlsseg2bu_v_uint16x2xm4` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x2xm1_t vlsseg2bu_v_uint32x2xm1` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x2xm2_t vlsseg2bu_v_uint32x2xm2` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x2xm4_t vlsseg2bu_v_uint32x2xm4` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x2xm1_t vlsseg2bu_v_uint64x2xm1` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x2xm2_t vlsseg2bu_v_uint64x2xm2` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x2xm4_t vlsseg2bu_v_uint64x2xm4` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x2xm1_t vlsseg2bu_v_uint8x2xm1` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x2xm2_t vlsseg2bu_v_uint8x2xm2` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x2xm4_t vlsseg2bu_v_uint8x2xm4` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x2xm1_t vlsseg2bu_mask_uint16x2xm1` (uint16x2xm1\_t *merge*, const unsigned short *\*address*, long *stride*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `uint16x2xm2_t vlsseg2bu_mask_uint16x2xm2` (uint16x2xm2\_t *merge*, const unsigned short *\*address*, long *stride*, *e16xm2\_t* *mask*, unsigned int *gvl*)

- `uint16x2xm4_t vlsseg2buvmask_uint16x2xm4` (`uint16x2xm4_t merge`, `const unsigned short *address`, `long stride`, `e16xm4_t mask`, `unsigned int gvl`)
- `uint32x2xm1_t vlsseg2buvmask_uint32x2xm1` (`uint32x2xm1_t merge`, `const unsigned int *address`, `long stride`, `e32xm1_t mask`, `unsigned int gvl`)
- `uint32x2xm2_t vlsseg2buvmask_uint32x2xm2` (`uint32x2xm2_t merge`, `const unsigned int *address`, `long stride`, `e32xm2_t mask`, `unsigned int gvl`)
- `uint32x2xm4_t vlsseg2buvmask_uint32x2xm4` (`uint32x2xm4_t merge`, `const unsigned int *address`, `long stride`, `e32xm4_t mask`, `unsigned int gvl`)
- `uint64x2xm1_t vlsseg2buvmask_uint64x2xm1` (`uint64x2xm1_t merge`, `const unsigned long *address`, `long stride`, `e64xm1_t mask`, `unsigned int gvl`)
- `uint64x2xm2_t vlsseg2buvmask_uint64x2xm2` (`uint64x2xm2_t merge`, `const unsigned long *address`, `long stride`, `e64xm2_t mask`, `unsigned int gvl`)
- `uint64x2xm4_t vlsseg2buvmask_uint64x2xm4` (`uint64x2xm4_t merge`, `const unsigned long *address`, `long stride`, `e64xm4_t mask`, `unsigned int gvl`)
- `uint8x2xm1_t vlsseg2buvmask_uint8x2xm1` (`uint8x2xm1_t merge`, `const unsigned char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8x2xm2_t vlsseg2buvmask_uint8x2xm2` (`uint8x2xm2_t merge`, `const unsigned char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)
- `uint8x2xm4_t vlsseg2buvmask_uint8x2xm4` (`uint8x2xm4_t merge`, `const unsigned char *address`, `long stride`, `e8xm4_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.52 Load 2 contiguous element fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg2e.v'`]

**Prototypes:**

- `float16x2xm1_t vlsseg2ev_float16x2xm1` (`const float16_t *address`, `long stride`, `unsigned int gvl`)
- `float16x2xm2_t vlsseg2ev_float16x2xm2` (`const float16_t *address`, `long stride`, `unsigned int gvl`)
- `float16x2xm4_t vlsseg2ev_float16x2xm4` (`const float16_t *address`, `long stride`, `unsigned int gvl`)
- `float32x2xm1_t vlsseg2ev_float32x2xm1` (`const float *address`, `long stride`, `unsigned int gvl`)
- `float32x2xm2_t vlsseg2ev_float32x2xm2` (`const float *address`, `long stride`, `unsigned int gvl`)

- `float32x2xm4_t vlsseg2ev_float32x2xm4` (const float *\*address*, long *stride*, unsigned int *gvl*)
- `float64x2xm1_t vlsseg2ev_float64x2xm1` (const double *\*address*, long *stride*, unsigned int *gvl*)
- `float64x2xm2_t vlsseg2ev_float64x2xm2` (const double *\*address*, long *stride*, unsigned int *gvl*)
- `float64x2xm4_t vlsseg2ev_float64x2xm4` (const double *\*address*, long *stride*, unsigned int *gvl*)
- `int16x2xm1_t vlsseg2ev_int16x2xm1` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16x2xm2_t vlsseg2ev_int16x2xm2` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16x2xm4_t vlsseg2ev_int16x2xm4` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int32x2xm1_t vlsseg2ev_int32x2xm1` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32x2xm2_t vlsseg2ev_int32x2xm2` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32x2xm4_t vlsseg2ev_int32x2xm4` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64x2xm1_t vlsseg2ev_int64x2xm1` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64x2xm2_t vlsseg2ev_int64x2xm2` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64x2xm4_t vlsseg2ev_int64x2xm4` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8x2xm1_t vlsseg2ev_int8x2xm1` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8x2xm2_t vlsseg2ev_int8x2xm2` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8x2xm4_t vlsseg2ev_int8x2xm4` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `uint16x2xm1_t vlsseg2ev_uint16x2xm1` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint16x2xm2_t vlsseg2ev_uint16x2xm2` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint16x2xm4_t vlsseg2ev_uint16x2xm4` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x2xm1_t vlsseg2ev_uint32x2xm1` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x2xm2_t vlsseg2ev_uint32x2xm2` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x2xm4_t vlsseg2ev_uint32x2xm4` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x2xm1_t vlsseg2ev_uint64x2xm1` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x2xm2_t vlsseg2ev_uint64x2xm2` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x2xm4_t vlsseg2ev_uint64x2xm4` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x2xm1_t vlsseg2ev_uint8x2xm1` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x2xm2_t vlsseg2ev_uint8x2xm2` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x2xm4_t vlsseg2ev_uint8x2xm4` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**



```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

### Masked prototypes:

- `float16x2xm1_t vlsseg2ev_mask_float16x2xm1` (`float16x2xm1_t merge`, `const float16_t *address`, `long stride`, `e16xm1_t mask`, `unsigned int gvl`)
- `float16x2xm2_t vlsseg2ev_mask_float16x2xm2` (`float16x2xm2_t merge`, `const float16_t *address`, `long stride`, `e16xm2_t mask`, `unsigned int gvl`)
- `float16x2xm4_t vlsseg2ev_mask_float16x2xm4` (`float16x2xm4_t merge`, `const float16_t *address`, `long stride`, `e16xm4_t mask`, `unsigned int gvl`)
- `float32x2xm1_t vlsseg2ev_mask_float32x2xm1` (`float32x2xm1_t merge`, `const float *address`, `long stride`, `e32xm1_t mask`, `unsigned int gvl`)
- `float32x2xm2_t vlsseg2ev_mask_float32x2xm2` (`float32x2xm2_t merge`, `const float *address`, `long stride`, `e32xm2_t mask`, `unsigned int gvl`)
- `float32x2xm4_t vlsseg2ev_mask_float32x2xm4` (`float32x2xm4_t merge`, `const float *address`, `long stride`, `e32xm4_t mask`, `unsigned int gvl`)
- `float64x2xm1_t vlsseg2ev_mask_float64x2xm1` (`float64x2xm1_t merge`, `const double *address`, `long stride`, `e64xm1_t mask`, `unsigned int gvl`)
- `float64x2xm2_t vlsseg2ev_mask_float64x2xm2` (`float64x2xm2_t merge`, `const double *address`, `long stride`, `e64xm2_t mask`, `unsigned int gvl`)
- `float64x2xm4_t vlsseg2ev_mask_float64x2xm4` (`float64x2xm4_t merge`, `const double *address`, `long stride`, `e64xm4_t mask`, `unsigned int gvl`)
- `int16x2xm1_t vlsseg2ev_mask_int16x2xm1` (`int16x2xm1_t merge`, `const short *address`, `long stride`, `e16xm1_t mask`, `unsigned int gvl`)
- `int16x2xm2_t vlsseg2ev_mask_int16x2xm2` (`int16x2xm2_t merge`, `const short *address`, `long stride`, `e16xm2_t mask`, `unsigned int gvl`)
- `int16x2xm4_t vlsseg2ev_mask_int16x2xm4` (`int16x2xm4_t merge`, `const short *address`, `long stride`, `e16xm4_t mask`, `unsigned int gvl`)
- `int32x2xm1_t vlsseg2ev_mask_int32x2xm1` (`int32x2xm1_t merge`, `const int *address`, `long stride`, `e32xm1_t mask`, `unsigned int gvl`)
- `int32x2xm2_t vlsseg2ev_mask_int32x2xm2` (`int32x2xm2_t merge`, `const int *address`, `long stride`, `e32xm2_t mask`, `unsigned int gvl`)
- `int32x2xm4_t vlsseg2ev_mask_int32x2xm4` (`int32x2xm4_t merge`, `const int *address`, `long stride`, `e32xm4_t mask`, `unsigned int gvl`)
- `int64x2xm1_t vlsseg2ev_mask_int64x2xm1` (`int64x2xm1_t merge`, `const long *address`, `long stride`, `e64xm1_t mask`, `unsigned int gvl`)
- `int64x2xm2_t vlsseg2ev_mask_int64x2xm2` (`int64x2xm2_t merge`, `const long *address`, `long stride`, `e64xm2_t mask`, `unsigned int gvl`)
- `int64x2xm4_t vlsseg2ev_mask_int64x2xm4` (`int64x2xm4_t merge`, `const long *address`, `long stride`, `e64xm4_t mask`, `unsigned int gvl`)
- `int8x2xm1_t vlsseg2ev_mask_int8x2xm1` (`int8x2xm1_t merge`, `const signed char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)

- `int8x2xm2_t vlsseg2ev_mask_int8x2xm2` (`int8x2xm2_t merge`, `const signed char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)
- `int8x2xm4_t vlsseg2ev_mask_int8x2xm4` (`int8x2xm4_t merge`, `const signed char *address`, `long stride`, `e8xm4_t mask`, `unsigned int gvl`)
- `uint16x2xm1_t vlsseg2ev_mask_uint16x2xm1` (`uint16x2xm1_t merge`, `const unsigned short *address`, `long stride`, `e16xm1_t mask`, `unsigned int gvl`)
- `uint16x2xm2_t vlsseg2ev_mask_uint16x2xm2` (`uint16x2xm2_t merge`, `const unsigned short *address`, `long stride`, `e16xm2_t mask`, `unsigned int gvl`)
- `uint16x2xm4_t vlsseg2ev_mask_uint16x2xm4` (`uint16x2xm4_t merge`, `const unsigned short *address`, `long stride`, `e16xm4_t mask`, `unsigned int gvl`)
- `uint32x2xm1_t vlsseg2ev_mask_uint32x2xm1` (`uint32x2xm1_t merge`, `const unsigned int *address`, `long stride`, `e32xm1_t mask`, `unsigned int gvl`)
- `uint32x2xm2_t vlsseg2ev_mask_uint32x2xm2` (`uint32x2xm2_t merge`, `const unsigned int *address`, `long stride`, `e32xm2_t mask`, `unsigned int gvl`)
- `uint32x2xm4_t vlsseg2ev_mask_uint32x2xm4` (`uint32x2xm4_t merge`, `const unsigned int *address`, `long stride`, `e32xm4_t mask`, `unsigned int gvl`)
- `uint64x2xm1_t vlsseg2ev_mask_uint64x2xm1` (`uint64x2xm1_t merge`, `const unsigned long *address`, `long stride`, `e64xm1_t mask`, `unsigned int gvl`)
- `uint64x2xm2_t vlsseg2ev_mask_uint64x2xm2` (`uint64x2xm2_t merge`, `const unsigned long *address`, `long stride`, `e64xm2_t mask`, `unsigned int gvl`)
- `uint64x2xm4_t vlsseg2ev_mask_uint64x2xm4` (`uint64x2xm4_t merge`, `const unsigned long *address`, `long stride`, `e64xm4_t mask`, `unsigned int gvl`)
- `uint8x2xm1_t vlsseg2ev_mask_uint8x2xm1` (`uint8x2xm1_t merge`, `const unsigned char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8x2xm2_t vlsseg2ev_mask_uint8x2xm2` (`uint8x2xm2_t merge`, `const unsigned char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)
- `uint8x2xm4_t vlsseg2ev_mask_uint8x2xm4` (`uint8x2xm4_t merge`, `const unsigned char *address`, `long stride`, `e8xm4_t mask`, `unsigned int gvl`)

#### Masked operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.53 Load 2 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg2h.v'`]

**Prototypes:**

- `int16x2xm1_t vlsseg2hv_int16x2xm1` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16x2xm2_t vlsseg2hv_int16x2xm2` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16x2xm4_t vlsseg2hv_int16x2xm4` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int32x2xm1_t vlsseg2hv_int32x2xm1` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32x2xm2_t vlsseg2hv_int32x2xm2` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32x2xm4_t vlsseg2hv_int32x2xm4` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64x2xm1_t vlsseg2hv_int64x2xm1` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64x2xm2_t vlsseg2hv_int64x2xm2` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64x2xm4_t vlsseg2hv_int64x2xm4` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8x2xm1_t vlsseg2hv_int8x2xm1` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8x2xm2_t vlsseg2hv_int8x2xm2` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8x2xm4_t vlsseg2hv_int8x2xm4` (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x2xm1_t vlsseg2hv_mask_int16x2xm1` (`int16x2xm1_t merge`, const short *\*address*, long *stride*, `e16xm1_t mask`, unsigned int *gvl*)
- `int16x2xm2_t vlsseg2hv_mask_int16x2xm2` (`int16x2xm2_t merge`, const short *\*address*, long *stride*, `e16xm2_t mask`, unsigned int *gvl*)
- `int16x2xm4_t vlsseg2hv_mask_int16x2xm4` (`int16x2xm4_t merge`, const short *\*address*, long *stride*, `e16xm4_t mask`, unsigned int *gvl*)
- `int32x2xm1_t vlsseg2hv_mask_int32x2xm1` (`int32x2xm1_t merge`, const int *\*address*, long *stride*, `e32xm1_t mask`, unsigned int *gvl*)
- `int32x2xm2_t vlsseg2hv_mask_int32x2xm2` (`int32x2xm2_t merge`, const int *\*address*, long *stride*, `e32xm2_t mask`, unsigned int *gvl*)
- `int32x2xm4_t vlsseg2hv_mask_int32x2xm4` (`int32x2xm4_t merge`, const int *\*address*, long *stride*, `e32xm4_t mask`, unsigned int *gvl*)
- `int64x2xm1_t vlsseg2hv_mask_int64x2xm1` (`int64x2xm1_t merge`, const long *\*address*, long *stride*, `e64xm1_t mask`, unsigned int *gvl*)
- `int64x2xm2_t vlsseg2hv_mask_int64x2xm2` (`int64x2xm2_t merge`, const long *\*address*, long *stride*, `e64xm2_t mask`, unsigned int *gvl*)
- `int64x2xm4_t vlsseg2hv_mask_int64x2xm4` (`int64x2xm4_t merge`, const long *\*address*, long *stride*, `e64xm4_t mask`, unsigned int *gvl*)

- `int8x2xm1_t vlsseg2hv_mask_int8x2xm1` (`int8x2xm1_t merge`, `const signed char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8x2xm2_t vlsseg2hv_mask_int8x2xm2` (`int8x2xm2_t merge`, `const signed char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)
- `int8x2xm4_t vlsseg2hv_mask_int8x2xm4` (`int8x2xm4_t merge`, `const signed char *address`, `long stride`, `e8xm4_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
      if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
      else
        result[segment] = merge[segment]
    result[gvl : VLMAX] = 0
```

**2.10.54 Load 2 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers****Instruction:** [`'vlsseg2hu.v'`]**Prototypes:**

- `uint16x2xm1_t vlsseg2huv_uint16x2xm1` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint16x2xm2_t vlsseg2huv_uint16x2xm2` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint16x2xm4_t vlsseg2huv_uint16x2xm4` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint32x2xm1_t vlsseg2huv_uint32x2xm1` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint32x2xm2_t vlsseg2huv_uint32x2xm2` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint32x2xm4_t vlsseg2huv_uint32x2xm4` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint64x2xm1_t vlsseg2huv_uint64x2xm1` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint64x2xm2_t vlsseg2huv_uint64x2xm2` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint64x2xm4_t vlsseg2huv_uint64x2xm4` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint8x2xm1_t vlsseg2huv_uint8x2xm1` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)
- `uint8x2xm2_t vlsseg2huv_uint8x2xm2` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)
- `uint8x2xm4_t vlsseg2huv_uint8x2xm4` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x2xm1_t vlsseg2huv_mask_uint16x2xm1` (`uint16x2xm1_t merge`, `const unsigned short *address`, `long stride`, `e16xm1_t mask`, `unsigned int gvl`)
- `uint16x2xm2_t vlsseg2huv_mask_uint16x2xm2` (`uint16x2xm2_t merge`, `const unsigned short *address`, `long stride`, `e16xm2_t mask`, `unsigned int gvl`)
- `uint16x2xm4_t vlsseg2huv_mask_uint16x2xm4` (`uint16x2xm4_t merge`, `const unsigned short *address`, `long stride`, `e16xm4_t mask`, `unsigned int gvl`)
- `uint32x2xm1_t vlsseg2huv_mask_uint32x2xm1` (`uint32x2xm1_t merge`, `const unsigned int *address`, `long stride`, `e32xm1_t mask`, `unsigned int gvl`)
- `uint32x2xm2_t vlsseg2huv_mask_uint32x2xm2` (`uint32x2xm2_t merge`, `const unsigned int *address`, `long stride`, `e32xm2_t mask`, `unsigned int gvl`)
- `uint32x2xm4_t vlsseg2huv_mask_uint32x2xm4` (`uint32x2xm4_t merge`, `const unsigned int *address`, `long stride`, `e32xm4_t mask`, `unsigned int gvl`)
- `uint64x2xm1_t vlsseg2huv_mask_uint64x2xm1` (`uint64x2xm1_t merge`, `const unsigned long *address`, `long stride`, `e64xm1_t mask`, `unsigned int gvl`)
- `uint64x2xm2_t vlsseg2huv_mask_uint64x2xm2` (`uint64x2xm2_t merge`, `const unsigned long *address`, `long stride`, `e64xm2_t mask`, `unsigned int gvl`)
- `uint64x2xm4_t vlsseg2huv_mask_uint64x2xm4` (`uint64x2xm4_t merge`, `const unsigned long *address`, `long stride`, `e64xm4_t mask`, `unsigned int gvl`)
- `uint8x2xm1_t vlsseg2huv_mask_uint8x2xm1` (`uint8x2xm1_t merge`, `const unsigned char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8x2xm2_t vlsseg2huv_mask_uint8x2xm2` (`uint8x2xm2_t merge`, `const unsigned char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)
- `uint8x2xm4_t vlsseg2huv_mask_uint8x2xm4` (`uint8x2xm4_t merge`, `const unsigned char *address`, `long stride`, `e8xm4_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.55 Load 2 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** ['vlsseg2w.v']

**Prototypes:**

- `int16x2xm1_t vlsseg2wv_int16x2xm1` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16x2xm2_t vlsseg2wv_int16x2xm2` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16x2xm4_t vlsseg2wv_int16x2xm4` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int32x2xm1_t vlsseg2wv_int32x2xm1` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32x2xm2_t vlsseg2wv_int32x2xm2` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32x2xm4_t vlsseg2wv_int32x2xm4` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64x2xm1_t vlsseg2wv_int64x2xm1` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64x2xm2_t vlsseg2wv_int64x2xm2` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64x2xm4_t vlsseg2wv_int64x2xm4` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8x2xm1_t vlsseg2wv_int8x2xm1` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8x2xm2_t vlsseg2wv_int8x2xm2` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8x2xm4_t vlsseg2wv_int8x2xm4` (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x2xm1_t vlsseg2wv_mask_int16x2xm1` (*int16x2xm1\_t merge*, const short *\*address*, long *stride*, *e16xm1\_t mask*, unsigned int *gvl*)
- `int16x2xm2_t vlsseg2wv_mask_int16x2xm2` (*int16x2xm2\_t merge*, const short *\*address*, long *stride*, *e16xm2\_t mask*, unsigned int *gvl*)
- `int16x2xm4_t vlsseg2wv_mask_int16x2xm4` (*int16x2xm4\_t merge*, const short *\*address*, long *stride*, *e16xm4\_t mask*, unsigned int *gvl*)
- `int32x2xm1_t vlsseg2wv_mask_int32x2xm1` (*int32x2xm1\_t merge*, const int *\*address*, long *stride*, *e32xm1\_t mask*, unsigned int *gvl*)
- `int32x2xm2_t vlsseg2wv_mask_int32x2xm2` (*int32x2xm2\_t merge*, const int *\*address*, long *stride*, *e32xm2\_t mask*, unsigned int *gvl*)
- `int32x2xm4_t vlsseg2wv_mask_int32x2xm4` (*int32x2xm4\_t merge*, const int *\*address*, long *stride*, *e32xm4\_t mask*, unsigned int *gvl*)
- `int64x2xm1_t vlsseg2wv_mask_int64x2xm1` (*int64x2xm1\_t merge*, const long *\*address*, long *stride*, *e64xm1\_t mask*, unsigned int *gvl*)
- `int64x2xm2_t vlsseg2wv_mask_int64x2xm2` (*int64x2xm2\_t merge*, const long *\*address*, long *stride*, *e64xm2\_t mask*, unsigned int *gvl*)
- `int64x2xm4_t vlsseg2wv_mask_int64x2xm4` (*int64x2xm4\_t merge*, const long *\*address*, long *stride*, *e64xm4\_t mask*, unsigned int *gvl*)



- `int8x2xm1_t vlsseg2wv_mask_int8x2xm1` (`int8x2xm1_t merge`, `const signed char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8x2xm2_t vlsseg2wv_mask_int8x2xm2` (`int8x2xm2_t merge`, `const signed char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)
- `int8x2xm4_t vlsseg2wv_mask_int8x2xm4` (`int8x2xm4_t merge`, `const signed char *address`, `long stride`, `e8xm4_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.56 Load 2 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg2wuv.v'`]

**Prototypes:**

- `uint16x2xm1_t vlsseg2wuv_uint16x2xm1` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint16x2xm2_t vlsseg2wuv_uint16x2xm2` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint16x2xm4_t vlsseg2wuv_uint16x2xm4` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint32x2xm1_t vlsseg2wuv_uint32x2xm1` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint32x2xm2_t vlsseg2wuv_uint32x2xm2` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint32x2xm4_t vlsseg2wuv_uint32x2xm4` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint64x2xm1_t vlsseg2wuv_uint64x2xm1` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint64x2xm2_t vlsseg2wuv_uint64x2xm2` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint64x2xm4_t vlsseg2wuv_uint64x2xm4` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint8x2xm1_t vlsseg2wuv_uint8x2xm1` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)
- `uint8x2xm2_t vlsseg2wuv_uint8x2xm2` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)
- `uint8x2xm4_t vlsseg2wuv_uint8x2xm4` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- `uint16x2xm1_t vlsseg2wuv_mask_uint16x2xm1` (`uint16x2xm1_t merge`, `const unsigned short *address`, `long stride`, `e16xm1_t mask`, `unsigned int gvl`)
- `uint16x2xm2_t vlsseg2wuv_mask_uint16x2xm2` (`uint16x2xm2_t merge`, `const unsigned short *address`, `long stride`, `e16xm2_t mask`, `unsigned int gvl`)
- `uint16x2xm4_t vlsseg2wuv_mask_uint16x2xm4` (`uint16x2xm4_t merge`, `const unsigned short *address`, `long stride`, `e16xm4_t mask`, `unsigned int gvl`)
- `uint32x2xm1_t vlsseg2wuv_mask_uint32x2xm1` (`uint32x2xm1_t merge`, `const unsigned int *address`, `long stride`, `e32xm1_t mask`, `unsigned int gvl`)
- `uint32x2xm2_t vlsseg2wuv_mask_uint32x2xm2` (`uint32x2xm2_t merge`, `const unsigned int *address`, `long stride`, `e32xm2_t mask`, `unsigned int gvl`)
- `uint32x2xm4_t vlsseg2wuv_mask_uint32x2xm4` (`uint32x2xm4_t merge`, `const unsigned int *address`, `long stride`, `e32xm4_t mask`, `unsigned int gvl`)
- `uint64x2xm1_t vlsseg2wuv_mask_uint64x2xm1` (`uint64x2xm1_t merge`, `const unsigned long *address`, `long stride`, `e64xm1_t mask`, `unsigned int gvl`)
- `uint64x2xm2_t vlsseg2wuv_mask_uint64x2xm2` (`uint64x2xm2_t merge`, `const unsigned long *address`, `long stride`, `e64xm2_t mask`, `unsigned int gvl`)
- `uint64x2xm4_t vlsseg2wuv_mask_uint64x2xm4` (`uint64x2xm4_t merge`, `const unsigned long *address`, `long stride`, `e64xm4_t mask`, `unsigned int gvl`)
- `uint8x2xm1_t vlsseg2wuv_mask_uint8x2xm1` (`uint8x2xm1_t merge`, `const unsigned char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8x2xm2_t vlsseg2wuv_mask_uint8x2xm2` (`uint8x2xm2_t merge`, `const unsigned char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)
- `uint8x2xm4_t vlsseg2wuv_mask_uint8x2xm4` (`uint8x2xm4_t merge`, `const unsigned char *address`, `long stride`, `e8xm4_t mask`, `unsigned int gvl`)

#### Masked operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.57 Load 3 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg3b.v'`]

**Prototypes:**

- `int16x3xm1_t vlsseg3bv_int16x3xm1` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16x3xm2_t vlsseg3bv_int16x3xm2` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int32x3xm1_t vlsseg3bv_int32x3xm1` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32x3xm2_t vlsseg3bv_int32x3xm2` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64x3xm1_t vlsseg3bv_int64x3xm1` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64x3xm2_t vlsseg3bv_int64x3xm2` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8x3xm1_t vlsseg3bv_int8x3xm1` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8x3xm2_t vlsseg3bv_int8x3xm2` (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x3xm1_t vlsseg3bv_mask_int16x3xm1` (*int16x3xm1\_t merge*, const short *\*address*, long *stride*, *e16xm1\_t mask*, unsigned int *gvl*)
- `int16x3xm2_t vlsseg3bv_mask_int16x3xm2` (*int16x3xm2\_t merge*, const short *\*address*, long *stride*, *e16xm2\_t mask*, unsigned int *gvl*)
- `int32x3xm1_t vlsseg3bv_mask_int32x3xm1` (*int32x3xm1\_t merge*, const int *\*address*, long *stride*, *e32xm1\_t mask*, unsigned int *gvl*)
- `int32x3xm2_t vlsseg3bv_mask_int32x3xm2` (*int32x3xm2\_t merge*, const int *\*address*, long *stride*, *e32xm2\_t mask*, unsigned int *gvl*)
- `int64x3xm1_t vlsseg3bv_mask_int64x3xm1` (*int64x3xm1\_t merge*, const long *\*address*, long *stride*, *e64xm1\_t mask*, unsigned int *gvl*)
- `int64x3xm2_t vlsseg3bv_mask_int64x3xm2` (*int64x3xm2\_t merge*, const long *\*address*, long *stride*, *e64xm2\_t mask*, unsigned int *gvl*)
- `int8x3xm1_t vlsseg3bv_mask_int8x3xm1` (*int8x3xm1\_t merge*, const signed char *\*address*, long *stride*, *e8xm1\_t mask*, unsigned int *gvl*)
- `int8x3xm2_t vlsseg3bv_mask_int8x3xm2` (*int8x3xm2\_t merge*, const signed char *\*address*, long *stride*, *e8xm2\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.58 Load 3 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg3bu.v`]

**Prototypes:**

- `uint16x3xm1_t vlsseg3bu_v_uint16x3xm1` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint16x3xm2_t vlsseg3bu_v_uint16x3xm2` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x3xm1_t vlsseg3bu_v_uint32x3xm1` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x3xm2_t vlsseg3bu_v_uint32x3xm2` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x3xm1_t vlsseg3bu_v_uint64x3xm1` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x3xm2_t vlsseg3bu_v_uint64x3xm2` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x3xm1_t vlsseg3bu_v_uint8x3xm1` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x3xm2_t vlsseg3bu_v_uint8x3xm2` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x3xm1_t vlsseg3bu_mask_uint16x3xm1` (`uint16x3xm1_t merge`, const unsigned short *\*address*, long *stride*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16x3xm2_t vlsseg3bu_mask_uint16x3xm2` (`uint16x3xm2_t merge`, const unsigned short *\*address*, long *stride*, `e16xm2_t mask`, unsigned int *gvl*)
- `uint32x3xm1_t vlsseg3bu_mask_uint32x3xm1` (`uint32x3xm1_t merge`, const unsigned int *\*address*, long *stride*, `e32xm1_t mask`, unsigned int *gvl*)
- `uint32x3xm2_t vlsseg3bu_mask_uint32x3xm2` (`uint32x3xm2_t merge`, const unsigned int *\*address*, long *stride*, `e32xm2_t mask`, unsigned int *gvl*)
- `uint64x3xm1_t vlsseg3bu_mask_uint64x3xm1` (`uint64x3xm1_t merge`, const unsigned long *\*address*, long *stride*, `e64xm1_t mask`, unsigned int *gvl*)
- `uint64x3xm2_t vlsseg3bu_mask_uint64x3xm2` (`uint64x3xm2_t merge`, const unsigned long *\*address*, long *stride*, `e64xm2_t mask`, unsigned int *gvl*)

- `uint8x3xm1_t vlsseg3buvmask_uint8x3xm1` (`uint8x3xm1_t merge`, `const unsigned char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8x3xm2_t vlsseg3buvmask_uint8x3xm2` (`uint8x3xm2_t merge`, `const unsigned char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.59 Load 3 contiguous element fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg3e.v`]

**Prototypes:**

- `float16x3xm1_t vlsseg3ev_float16x3xm1` (`const float16_t *address`, `long stride`, `unsigned int gvl`)
- `float16x3xm2_t vlsseg3ev_float16x3xm2` (`const float16_t *address`, `long stride`, `unsigned int gvl`)
- `float32x3xm1_t vlsseg3ev_float32x3xm1` (`const float *address`, `long stride`, `unsigned int gvl`)
- `float32x3xm2_t vlsseg3ev_float32x3xm2` (`const float *address`, `long stride`, `unsigned int gvl`)
- `float64x3xm1_t vlsseg3ev_float64x3xm1` (`const double *address`, `long stride`, `unsigned int gvl`)
- `float64x3xm2_t vlsseg3ev_float64x3xm2` (`const double *address`, `long stride`, `unsigned int gvl`)
- `int16x3xm1_t vlsseg3ev_int16x3xm1` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int16x3xm2_t vlsseg3ev_int16x3xm2` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int32x3xm1_t vlsseg3ev_int32x3xm1` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int32x3xm2_t vlsseg3ev_int32x3xm2` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int64x3xm1_t vlsseg3ev_int64x3xm1` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int64x3xm2_t vlsseg3ev_int64x3xm2` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int8x3xm1_t vlsseg3ev_int8x3xm1` (`const signed char *address`, `long stride`, `unsigned int gvl`)
- `int8x3xm2_t vlsseg3ev_int8x3xm2` (`const signed char *address`, `long stride`, `unsigned int gvl`)
- `uint16x3xm1_t vlsseg3ev_uint16x3xm1` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint16x3xm2_t vlsseg3ev_uint16x3xm2` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint32x3xm1_t vlsseg3ev_uint32x3xm1` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint32x3xm2_t vlsseg3ev_uint32x3xm2` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint64x3xm1_t vlsseg3ev_uint64x3xm1` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)

- `uint64x3xm2_t vlsseg3ev_uint64x3xm2` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x3xm1_t vlsseg3ev_uint8x3xm1` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x3xm2_t vlsseg3ev_uint8x3xm2` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x3xm1_t vlsseg3ev_mask_float16x3xm1` (`float16x3xm1_t merge`, const `float16_t *address`, long *stride*, `e16xm1_t mask`, unsigned int *gvl*)
- `float16x3xm2_t vlsseg3ev_mask_float16x3xm2` (`float16x3xm2_t merge`, const `float16_t *address`, long *stride*, `e16xm2_t mask`, unsigned int *gvl*)
- `float32x3xm1_t vlsseg3ev_mask_float32x3xm1` (`float32x3xm1_t merge`, const `float *address`, long *stride*, `e32xm1_t mask`, unsigned int *gvl*)
- `float32x3xm2_t vlsseg3ev_mask_float32x3xm2` (`float32x3xm2_t merge`, const `float *address`, long *stride*, `e32xm2_t mask`, unsigned int *gvl*)
- `float64x3xm1_t vlsseg3ev_mask_float64x3xm1` (`float64x3xm1_t merge`, const `double *address`, long *stride*, `e64xm1_t mask`, unsigned int *gvl*)
- `float64x3xm2_t vlsseg3ev_mask_float64x3xm2` (`float64x3xm2_t merge`, const `double *address`, long *stride*, `e64xm2_t mask`, unsigned int *gvl*)
- `int16x3xm1_t vlsseg3ev_mask_int16x3xm1` (`int16x3xm1_t merge`, const `short *address`, long *stride*, `e16xm1_t mask`, unsigned int *gvl*)
- `int16x3xm2_t vlsseg3ev_mask_int16x3xm2` (`int16x3xm2_t merge`, const `short *address`, long *stride*, `e16xm2_t mask`, unsigned int *gvl*)
- `int32x3xm1_t vlsseg3ev_mask_int32x3xm1` (`int32x3xm1_t merge`, const `int *address`, long *stride*, `e32xm1_t mask`, unsigned int *gvl*)
- `int32x3xm2_t vlsseg3ev_mask_int32x3xm2` (`int32x3xm2_t merge`, const `int *address`, long *stride*, `e32xm2_t mask`, unsigned int *gvl*)
- `int64x3xm1_t vlsseg3ev_mask_int64x3xm1` (`int64x3xm1_t merge`, const `long *address`, long *stride*, `e64xm1_t mask`, unsigned int *gvl*)
- `int64x3xm2_t vlsseg3ev_mask_int64x3xm2` (`int64x3xm2_t merge`, const `long *address`, long *stride*, `e64xm2_t mask`, unsigned int *gvl*)
- `int8x3xm1_t vlsseg3ev_mask_int8x3xm1` (`int8x3xm1_t merge`, const `signed char *address`, long *stride*, `e8xm1_t mask`, unsigned int *gvl*)
- `int8x3xm2_t vlsseg3ev_mask_int8x3xm2` (`int8x3xm2_t merge`, const `signed char *address`, long *stride*, `e8xm2_t mask`, unsigned int *gvl*)
- `uint16x3xm1_t vlsseg3ev_mask_uint16x3xm1` (`uint16x3xm1_t merge`, const unsigned short *\*address*, long *stride*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16x3xm2_t vlsseg3ev_mask_uint16x3xm2` (`uint16x3xm2_t merge`, const unsigned short *\*address*, long *stride*, `e16xm2_t mask`, unsigned int *gvl*)



- `uint32x3xm1_t vlsseg3ev_mask_uint32x3xm1` (`uint32x3xm1_t merge`, const unsigned int *\*address*, long *stride*, *e32xm1\_t mask*, unsigned int *gvl*)
- `uint32x3xm2_t vlsseg3ev_mask_uint32x3xm2` (`uint32x3xm2_t merge`, const unsigned int *\*address*, long *stride*, *e32xm2\_t mask*, unsigned int *gvl*)
- `uint64x3xm1_t vlsseg3ev_mask_uint64x3xm1` (`uint64x3xm1_t merge`, const unsigned long *\*address*, long *stride*, *e64xm1\_t mask*, unsigned int *gvl*)
- `uint64x3xm2_t vlsseg3ev_mask_uint64x3xm2` (`uint64x3xm2_t merge`, const unsigned long *\*address*, long *stride*, *e64xm2\_t mask*, unsigned int *gvl*)
- `uint8x3xm1_t vlsseg3ev_mask_uint8x3xm1` (`uint8x3xm1_t merge`, const unsigned char *\*address*, long *stride*, *e8xm1\_t mask*, unsigned int *gvl*)
- `uint8x3xm2_t vlsseg3ev_mask_uint8x3xm2` (`uint8x3xm2_t merge`, const unsigned char *\*address*, long *stride*, *e8xm2\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.60 Load 3 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg3h.v`]

**Prototypes:**

- `int16x3xm1_t vlsseg3hv_int16x3xm1` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16x3xm2_t vlsseg3hv_int16x3xm2` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int32x3xm1_t vlsseg3hv_int32x3xm1` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32x3xm2_t vlsseg3hv_int32x3xm2` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64x3xm1_t vlsseg3hv_int64x3xm1` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64x3xm2_t vlsseg3hv_int64x3xm2` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8x3xm1_t vlsseg3hv_int8x3xm1` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8x3xm2_t vlsseg3hv_int8x3xm2` (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x3xm1_t vlsseg3hv_mask_int16x3xm1` (`int16x3xm1_t merge`, `const short *address`, `long stride`, `e16xm1_t mask`, `unsigned int gvl`)
- `int16x3xm2_t vlsseg3hv_mask_int16x3xm2` (`int16x3xm2_t merge`, `const short *address`, `long stride`, `e16xm2_t mask`, `unsigned int gvl`)
- `int32x3xm1_t vlsseg3hv_mask_int32x3xm1` (`int32x3xm1_t merge`, `const int *address`, `long stride`, `e32xm1_t mask`, `unsigned int gvl`)
- `int32x3xm2_t vlsseg3hv_mask_int32x3xm2` (`int32x3xm2_t merge`, `const int *address`, `long stride`, `e32xm2_t mask`, `unsigned int gvl`)
- `int64x3xm1_t vlsseg3hv_mask_int64x3xm1` (`int64x3xm1_t merge`, `const long *address`, `long stride`, `e64xm1_t mask`, `unsigned int gvl`)
- `int64x3xm2_t vlsseg3hv_mask_int64x3xm2` (`int64x3xm2_t merge`, `const long *address`, `long stride`, `e64xm2_t mask`, `unsigned int gvl`)
- `int8x3xm1_t vlsseg3hv_mask_int8x3xm1` (`int8x3xm1_t merge`, `const signed char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8x3xm2_t vlsseg3hv_mask_int8x3xm2` (`int8x3xm2_t merge`, `const signed char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.61 Load 3 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg3hu.v`]

**Prototypes:**

- `uint16x3xm1_t vlsseg3huv_uint16x3xm1` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint16x3xm2_t vlsseg3huv_uint16x3xm2` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint32x3xm1_t vlsseg3huv_uint32x3xm1` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint32x3xm2_t vlsseg3huv_uint32x3xm2` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint64x3xm1_t vlsseg3huv_uint64x3xm1` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint64x3xm2_t vlsseg3huv_uint64x3xm2` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint8x3xm1_t vlsseg3huv_uint8x3xm1` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)
- `uint8x3xm2_t vlsseg3huv_uint8x3xm2` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x3xm1_t vlsseg3huv_mask_uint16x3xm1` (`uint16x3xm1_t merge`, `const unsigned short *address`, `long stride`, `e16xm1_t mask`, `unsigned int gvl`)
- `uint16x3xm2_t vlsseg3huv_mask_uint16x3xm2` (`uint16x3xm2_t merge`, `const unsigned short *address`, `long stride`, `e16xm2_t mask`, `unsigned int gvl`)
- `uint32x3xm1_t vlsseg3huv_mask_uint32x3xm1` (`uint32x3xm1_t merge`, `const unsigned int *address`, `long stride`, `e32xm1_t mask`, `unsigned int gvl`)
- `uint32x3xm2_t vlsseg3huv_mask_uint32x3xm2` (`uint32x3xm2_t merge`, `const unsigned int *address`, `long stride`, `e32xm2_t mask`, `unsigned int gvl`)
- `uint64x3xm1_t vlsseg3huv_mask_uint64x3xm1` (`uint64x3xm1_t merge`, `const unsigned long *address`, `long stride`, `e64xm1_t mask`, `unsigned int gvl`)
- `uint64x3xm2_t vlsseg3huv_mask_uint64x3xm2` (`uint64x3xm2_t merge`, `const unsigned long *address`, `long stride`, `e64xm2_t mask`, `unsigned int gvl`)
- `uint8x3xm1_t vlsseg3huv_mask_uint8x3xm1` (`uint8x3xm1_t merge`, `const unsigned char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8x3xm2_t vlsseg3huv_mask_uint8x3xm2` (`uint8x3xm2_t merge`, `const unsigned char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.62 Load 3 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg3w.v'`]

**Prototypes:**

- `int16x3xm1_t vlsseg3wv_int16x3xm1` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int16x3xm2_t vlsseg3wv_int16x3xm2` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int32x3xm1_t vlsseg3wv_int32x3xm1` (`const int *address`, `long stride`, `unsigned int gvl`)

- `int32x3xm2_t vlsseg3wv_int32x3xm2` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64x3xm1_t vlsseg3wv_int64x3xm1` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64x3xm2_t vlsseg3wv_int64x3xm2` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8x3xm1_t vlsseg3wv_int8x3xm1` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8x3xm2_t vlsseg3wv_int8x3xm2` (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x3xm1_t vlsseg3wv_mask_int16x3xm1` (int16x3xm1\_t *merge*, const short *\*address*, long *stride*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `int16x3xm2_t vlsseg3wv_mask_int16x3xm2` (int16x3xm2\_t *merge*, const short *\*address*, long *stride*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- `int32x3xm1_t vlsseg3wv_mask_int32x3xm1` (int32x3xm1\_t *merge*, const int *\*address*, long *stride*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- `int32x3xm2_t vlsseg3wv_mask_int32x3xm2` (int32x3xm2\_t *merge*, const int *\*address*, long *stride*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- `int64x3xm1_t vlsseg3wv_mask_int64x3xm1` (int64x3xm1\_t *merge*, const long *\*address*, long *stride*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- `int64x3xm2_t vlsseg3wv_mask_int64x3xm2` (int64x3xm2\_t *merge*, const long *\*address*, long *stride*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- `int8x3xm1_t vlsseg3wv_mask_int8x3xm1` (int8x3xm1\_t *merge*, const signed char *\*address*, long *stride*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- `int8x3xm2_t vlsseg3wv_mask_int8x3xm2` (int8x3xm2\_t *merge*, const signed char *\*address*, long *stride*, *e8xm2\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.63 Load 3 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg3wu.v`]

**Prototypes:**

- `uint16x3xm1_t vlsseg3wuv_uint16x3xm1` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)

- `uint16x3xm2_t vlsseg3wuv_uint16x3xm2` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x3xm1_t vlsseg3wuv_uint32x3xm1` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x3xm2_t vlsseg3wuv_uint32x3xm2` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x3xm1_t vlsseg3wuv_uint64x3xm1` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x3xm2_t vlsseg3wuv_uint64x3xm2` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x3xm1_t vlsseg3wuv_uint8x3xm1` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x3xm2_t vlsseg3wuv_uint8x3xm2` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x3xm1_t vlsseg3wuv_mask_uint16x3xm1` (`uint16x3xm1_t merge`, const unsigned short *\*address*, long *stride*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16x3xm2_t vlsseg3wuv_mask_uint16x3xm2` (`uint16x3xm2_t merge`, const unsigned short *\*address*, long *stride*, `e16xm2_t mask`, unsigned int *gvl*)
- `uint32x3xm1_t vlsseg3wuv_mask_uint32x3xm1` (`uint32x3xm1_t merge`, const unsigned int *\*address*, long *stride*, `e32xm1_t mask`, unsigned int *gvl*)
- `uint32x3xm2_t vlsseg3wuv_mask_uint32x3xm2` (`uint32x3xm2_t merge`, const unsigned int *\*address*, long *stride*, `e32xm2_t mask`, unsigned int *gvl*)
- `uint64x3xm1_t vlsseg3wuv_mask_uint64x3xm1` (`uint64x3xm1_t merge`, const unsigned long *\*address*, long *stride*, `e64xm1_t mask`, unsigned int *gvl*)
- `uint64x3xm2_t vlsseg3wuv_mask_uint64x3xm2` (`uint64x3xm2_t merge`, const unsigned long *\*address*, long *stride*, `e64xm2_t mask`, unsigned int *gvl*)
- `uint8x3xm1_t vlsseg3wuv_mask_uint8x3xm1` (`uint8x3xm1_t merge`, const unsigned char *\*address*, long *stride*, `e8xm1_t mask`, unsigned int *gvl*)
- `uint8x3xm2_t vlsseg3wuv_mask_uint8x3xm2` (`uint8x3xm2_t merge`, const unsigned char *\*address*, long *stride*, `e8xm2_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
```

(continues on next page)

(continued from previous page)

```

    address = address + sizeof(segment) + stride
else
    result[segment] = merge[segment]
result[gvl : VLMAX] = 0

```

## 2.10.64 Load 4 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** ['vlsseg4b.v']

**Prototypes:**

- `int16x4xm1_t vlsseg4bv_int16x4xm1` (const short \**address*, long *stride*, unsigned int *gvl*)
- `int16x4xm2_t vlsseg4bv_int16x4xm2` (const short \**address*, long *stride*, unsigned int *gvl*)
- `int32x4xm1_t vlsseg4bv_int32x4xm1` (const int \**address*, long *stride*, unsigned int *gvl*)
- `int32x4xm2_t vlsseg4bv_int32x4xm2` (const int \**address*, long *stride*, unsigned int *gvl*)
- `int64x4xm1_t vlsseg4bv_int64x4xm1` (const long \**address*, long *stride*, unsigned int *gvl*)
- `int64x4xm2_t vlsseg4bv_int64x4xm2` (const long \**address*, long *stride*, unsigned int *gvl*)
- `int8x4xm1_t vlsseg4bv_int8x4xm1` (const signed char \**address*, long *stride*, unsigned int *gvl*)
- `int8x4xm2_t vlsseg4bv_int8x4xm2` (const signed char \**address*, long *stride*, unsigned int *gvl*)

**Operation:**

```

>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- `int16x4xm1_t vlsseg4bv_mask_int16x4xm1` (int16x4xm1\_t *merge*, const short \**address*, long *stride*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `int16x4xm2_t vlsseg4bv_mask_int16x4xm2` (int16x4xm2\_t *merge*, const short \**address*, long *stride*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- `int32x4xm1_t vlsseg4bv_mask_int32x4xm1` (int32x4xm1\_t *merge*, const int \**address*, long *stride*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- `int32x4xm2_t vlsseg4bv_mask_int32x4xm2` (int32x4xm2\_t *merge*, const int \**address*, long *stride*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- `int64x4xm1_t vlsseg4bv_mask_int64x4xm1` (int64x4xm1\_t *merge*, const long \**address*, long *stride*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- `int64x4xm2_t vlsseg4bv_mask_int64x4xm2` (int64x4xm2\_t *merge*, const long \**address*, long *stride*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- `int8x4xm1_t vlsseg4bv_mask_int8x4xm1` (int8x4xm1\_t *merge*, const signed char \**address*, long *stride*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- `int8x4xm2_t vlsseg4bv_mask_int8x4xm2` (int8x4xm2\_t *merge*, const signed char \**address*, long *stride*, *e8xm2\_t* *mask*, unsigned int *gvl*)

**Masked operation:**



```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.65 Load 4 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg4bu.v'`]

**Prototypes:**

- `uint16x4xm1_t vlsseg4buv_uint16x4xm1` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint16x4xm2_t vlsseg4buv_uint16x4xm2` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x4xm1_t vlsseg4buv_uint32x4xm1` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x4xm2_t vlsseg4buv_uint32x4xm2` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x4xm1_t vlsseg4buv_uint64x4xm1` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x4xm2_t vlsseg4buv_uint64x4xm2` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x4xm1_t vlsseg4buv_uint8x4xm1` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x4xm2_t vlsseg4buv_uint8x4xm2` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x4xm1_t vlsseg4buv_mask_uint16x4xm1` (uint16x4xm1\_t *merge*, const unsigned short *\*address*, long *stride*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `uint16x4xm2_t vlsseg4buv_mask_uint16x4xm2` (uint16x4xm2\_t *merge*, const unsigned short *\*address*, long *stride*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- `uint32x4xm1_t vlsseg4buv_mask_uint32x4xm1` (uint32x4xm1\_t *merge*, const unsigned int *\*address*, long *stride*, *e32xm1\_t* *mask*, unsigned int *gvl*)

- `uint32x4xm2_t vlsseg4buv_mask_uint32x4xm2` (`uint32x4xm2_t merge`, `const unsigned int *address`, `long stride`, `e32xm2_t mask`, `unsigned int gvl`)
- `uint64x4xm1_t vlsseg4buv_mask_uint64x4xm1` (`uint64x4xm1_t merge`, `const unsigned long *address`, `long stride`, `e64xm1_t mask`, `unsigned int gvl`)
- `uint64x4xm2_t vlsseg4buv_mask_uint64x4xm2` (`uint64x4xm2_t merge`, `const unsigned long *address`, `long stride`, `e64xm2_t mask`, `unsigned int gvl`)
- `uint8x4xm1_t vlsseg4buv_mask_uint8x4xm1` (`uint8x4xm1_t merge`, `const unsigned char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8x4xm2_t vlsseg4buv_mask_uint8x4xm2` (`uint8x4xm2_t merge`, `const unsigned char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)

#### Masked operation:

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.66 Load 4 contiguous element fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg4e.v`]

#### Prototypes:

- `float16x4xm1_t vlsseg4ev_float16x4xm1` (`const float16_t *address`, `long stride`, `unsigned int gvl`)
- `float16x4xm2_t vlsseg4ev_float16x4xm2` (`const float16_t *address`, `long stride`, `unsigned int gvl`)
- `float32x4xm1_t vlsseg4ev_float32x4xm1` (`const float *address`, `long stride`, `unsigned int gvl`)
- `float32x4xm2_t vlsseg4ev_float32x4xm2` (`const float *address`, `long stride`, `unsigned int gvl`)
- `float64x4xm1_t vlsseg4ev_float64x4xm1` (`const double *address`, `long stride`, `unsigned int gvl`)
- `float64x4xm2_t vlsseg4ev_float64x4xm2` (`const double *address`, `long stride`, `unsigned int gvl`)
- `int16x4xm1_t vlsseg4ev_int16x4xm1` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int16x4xm2_t vlsseg4ev_int16x4xm2` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int32x4xm1_t vlsseg4ev_int32x4xm1` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int32x4xm2_t vlsseg4ev_int32x4xm2` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int64x4xm1_t vlsseg4ev_int64x4xm1` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int64x4xm2_t vlsseg4ev_int64x4xm2` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int8x4xm1_t vlsseg4ev_int8x4xm1` (`const signed char *address`, `long stride`, `unsigned int gvl`)
- `int8x4xm2_t vlsseg4ev_int8x4xm2` (`const signed char *address`, `long stride`, `unsigned int gvl`)
- `uint16x4xm1_t vlsseg4ev_uint16x4xm1` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)

- `uint16x4xm2_t vlsseg4ev_uint16x4xm2` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x4xm1_t vlsseg4ev_uint32x4xm1` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x4xm2_t vlsseg4ev_uint32x4xm2` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x4xm1_t vlsseg4ev_uint64x4xm1` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x4xm2_t vlsseg4ev_uint64x4xm2` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x4xm1_t vlsseg4ev_uint8x4xm1` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x4xm2_t vlsseg4ev_uint8x4xm2` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x4xm1_t vlsseg4ev_mask_float16x4xm1` (float16x4xm1\_t *merge*, const float16\_t *\*address*, long *stride*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `float16x4xm2_t vlsseg4ev_mask_float16x4xm2` (float16x4xm2\_t *merge*, const float16\_t *\*address*, long *stride*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- `float32x4xm1_t vlsseg4ev_mask_float32x4xm1` (float32x4xm1\_t *merge*, const float *\*address*, long *stride*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- `float32x4xm2_t vlsseg4ev_mask_float32x4xm2` (float32x4xm2\_t *merge*, const float *\*address*, long *stride*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- `float64x4xm1_t vlsseg4ev_mask_float64x4xm1` (float64x4xm1\_t *merge*, const double *\*address*, long *stride*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- `float64x4xm2_t vlsseg4ev_mask_float64x4xm2` (float64x4xm2\_t *merge*, const double *\*address*, long *stride*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- `int16x4xm1_t vlsseg4ev_mask_int16x4xm1` (int16x4xm1\_t *merge*, const short *\*address*, long *stride*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `int16x4xm2_t vlsseg4ev_mask_int16x4xm2` (int16x4xm2\_t *merge*, const short *\*address*, long *stride*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- `int32x4xm1_t vlsseg4ev_mask_int32x4xm1` (int32x4xm1\_t *merge*, const int *\*address*, long *stride*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- `int32x4xm2_t vlsseg4ev_mask_int32x4xm2` (int32x4xm2\_t *merge*, const int *\*address*, long *stride*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- `int64x4xm1_t vlsseg4ev_mask_int64x4xm1` (int64x4xm1\_t *merge*, const long *\*address*, long *stride*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- `int64x4xm2_t vlsseg4ev_mask_int64x4xm2` (int64x4xm2\_t *merge*, const long *\*address*, long *stride*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- `int8x4xm1_t vlsseg4ev_mask_int8x4xm1` (int8x4xm1\_t *merge*, const signed char *\*address*, long *stride*, *e8xm1\_t* *mask*, unsigned int *gvl*)

- `int8x4xm2_t vlsseg4ev_mask_int8x4xm2` (`int8x4xm2_t merge`, `const signed char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)
- `uint16x4xm1_t vlsseg4ev_mask_uint16x4xm1` (`uint16x4xm1_t merge`, `const unsigned short *address`, `long stride`, `e16xm1_t mask`, `unsigned int gvl`)
- `uint16x4xm2_t vlsseg4ev_mask_uint16x4xm2` (`uint16x4xm2_t merge`, `const unsigned short *address`, `long stride`, `e16xm2_t mask`, `unsigned int gvl`)
- `uint32x4xm1_t vlsseg4ev_mask_uint32x4xm1` (`uint32x4xm1_t merge`, `const unsigned int *address`, `long stride`, `e32xm1_t mask`, `unsigned int gvl`)
- `uint32x4xm2_t vlsseg4ev_mask_uint32x4xm2` (`uint32x4xm2_t merge`, `const unsigned int *address`, `long stride`, `e32xm2_t mask`, `unsigned int gvl`)
- `uint64x4xm1_t vlsseg4ev_mask_uint64x4xm1` (`uint64x4xm1_t merge`, `const unsigned long *address`, `long stride`, `e64xm1_t mask`, `unsigned int gvl`)
- `uint64x4xm2_t vlsseg4ev_mask_uint64x4xm2` (`uint64x4xm2_t merge`, `const unsigned long *address`, `long stride`, `e64xm2_t mask`, `unsigned int gvl`)
- `uint8x4xm1_t vlsseg4ev_mask_uint8x4xm1` (`uint8x4xm1_t merge`, `const unsigned char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8x4xm2_t vlsseg4ev_mask_uint8x4xm2` (`uint8x4xm2_t merge`, `const unsigned char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.67 Load 4 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers****Instruction:** [`'vlsseg4h.v'`]**Prototypes:**

- `int16x4xm1_t vlsseg4hv_int16x4xm1` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int16x4xm2_t vlsseg4hv_int16x4xm2` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int32x4xm1_t vlsseg4hv_int32x4xm1` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int32x4xm2_t vlsseg4hv_int32x4xm2` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int64x4xm1_t vlsseg4hv_int64x4xm1` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int64x4xm2_t vlsseg4hv_int64x4xm2` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int8x4xm1_t vlsseg4hv_int8x4xm1` (`const signed char *address`, `long stride`, `unsigned int gvl`)

- `int8x4xm2_t vlsseg4hv_int8x4xm2` (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x4xm1_t vlsseg4hv_mask_int16x4xm1` (int16x4xm1\_t *merge*, const short *\*address*, long *stride*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `int16x4xm2_t vlsseg4hv_mask_int16x4xm2` (int16x4xm2\_t *merge*, const short *\*address*, long *stride*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- `int32x4xm1_t vlsseg4hv_mask_int32x4xm1` (int32x4xm1\_t *merge*, const int *\*address*, long *stride*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- `int32x4xm2_t vlsseg4hv_mask_int32x4xm2` (int32x4xm2\_t *merge*, const int *\*address*, long *stride*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- `int64x4xm1_t vlsseg4hv_mask_int64x4xm1` (int64x4xm1\_t *merge*, const long *\*address*, long *stride*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- `int64x4xm2_t vlsseg4hv_mask_int64x4xm2` (int64x4xm2\_t *merge*, const long *\*address*, long *stride*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- `int8x4xm1_t vlsseg4hv_mask_int8x4xm1` (int8x4xm1\_t *merge*, const signed char *\*address*, long *stride*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- `int8x4xm2_t vlsseg4hv_mask_int8x4xm2` (int8x4xm2\_t *merge*, const signed char *\*address*, long *stride*, *e8xm2\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.68 Load 4 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg4hu.v'`]

**Prototypes:**

- `uint16x4xm1_t vlsseg4huv_uint16x4xm1` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint16x4xm2_t vlsseg4huv_uint16x4xm2` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x4xm1_t vlsseg4huv_uint32x4xm1` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x4xm2_t vlsseg4huv_uint32x4xm2` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)

- `uint64x4xm1_t vlsseg4huv_uint64x4xm1` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x4xm2_t vlsseg4huv_uint64x4xm2` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x4xm1_t vlsseg4huv_uint8x4xm1` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x4xm2_t vlsseg4huv_uint8x4xm2` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x4xm1_t vlsseg4huv_mask_uint16x4xm1` (`uint16x4xm1_t merge`, const unsigned short *\*address*, long *stride*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16x4xm2_t vlsseg4huv_mask_uint16x4xm2` (`uint16x4xm2_t merge`, const unsigned short *\*address*, long *stride*, `e16xm2_t mask`, unsigned int *gvl*)
- `uint32x4xm1_t vlsseg4huv_mask_uint32x4xm1` (`uint32x4xm1_t merge`, const unsigned int *\*address*, long *stride*, `e32xm1_t mask`, unsigned int *gvl*)
- `uint32x4xm2_t vlsseg4huv_mask_uint32x4xm2` (`uint32x4xm2_t merge`, const unsigned int *\*address*, long *stride*, `e32xm2_t mask`, unsigned int *gvl*)
- `uint64x4xm1_t vlsseg4huv_mask_uint64x4xm1` (`uint64x4xm1_t merge`, const unsigned long *\*address*, long *stride*, `e64xm1_t mask`, unsigned int *gvl*)
- `uint64x4xm2_t vlsseg4huv_mask_uint64x4xm2` (`uint64x4xm2_t merge`, const unsigned long *\*address*, long *stride*, `e64xm2_t mask`, unsigned int *gvl*)
- `uint8x4xm1_t vlsseg4huv_mask_uint8x4xm1` (`uint8x4xm1_t merge`, const unsigned char *\*address*, long *stride*, `e8xm1_t mask`, unsigned int *gvl*)
- `uint8x4xm2_t vlsseg4huv_mask_uint8x4xm2` (`uint8x4xm2_t merge`, const unsigned char *\*address*, long *stride*, `e8xm2_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```



## 2.10.69 Load 4 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** ['vlsseg4w.v']

**Prototypes:**

- `int16x4xm1_t vlsseg4wv_int16x4xm1` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int16x4xm2_t vlsseg4wv_int16x4xm2` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int32x4xm1_t vlsseg4wv_int32x4xm1` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int32x4xm2_t vlsseg4wv_int32x4xm2` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64x4xm1_t vlsseg4wv_int64x4xm1` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int64x4xm2_t vlsseg4wv_int64x4xm2` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8x4xm1_t vlsseg4wv_int8x4xm1` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `int8x4xm2_t vlsseg4wv_int8x4xm2` (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x4xm1_t vlsseg4wv_mask_int16x4xm1` (*int16x4xm1\_t merge*, const short *\*address*, long *stride*, *e16xm1\_t mask*, unsigned int *gvl*)
- `int16x4xm2_t vlsseg4wv_mask_int16x4xm2` (*int16x4xm2\_t merge*, const short *\*address*, long *stride*, *e16xm2\_t mask*, unsigned int *gvl*)
- `int32x4xm1_t vlsseg4wv_mask_int32x4xm1` (*int32x4xm1\_t merge*, const int *\*address*, long *stride*, *e32xm1\_t mask*, unsigned int *gvl*)
- `int32x4xm2_t vlsseg4wv_mask_int32x4xm2` (*int32x4xm2\_t merge*, const int *\*address*, long *stride*, *e32xm2\_t mask*, unsigned int *gvl*)
- `int64x4xm1_t vlsseg4wv_mask_int64x4xm1` (*int64x4xm1\_t merge*, const long *\*address*, long *stride*, *e64xm1\_t mask*, unsigned int *gvl*)
- `int64x4xm2_t vlsseg4wv_mask_int64x4xm2` (*int64x4xm2\_t merge*, const long *\*address*, long *stride*, *e64xm2\_t mask*, unsigned int *gvl*)
- `int8x4xm1_t vlsseg4wv_mask_int8x4xm1` (*int8x4xm1\_t merge*, const signed char *\*address*, long *stride*, *e8xm1\_t mask*, unsigned int *gvl*)
- `int8x4xm2_t vlsseg4wv_mask_int8x4xm2` (*int8x4xm2\_t merge*, const signed char *\*address*, long *stride*, *e8xm2\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.70 Load 4 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg4wuv.v'`]

**Prototypes:**

- `uint16x4xm1_t vlsseg4wuv_uint16x4xm1` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint16x4xm2_t vlsseg4wuv_uint16x4xm2` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x4xm1_t vlsseg4wuv_uint32x4xm1` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x4xm2_t vlsseg4wuv_uint32x4xm2` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x4xm1_t vlsseg4wuv_uint64x4xm1` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x4xm2_t vlsseg4wuv_uint64x4xm2` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x4xm1_t vlsseg4wuv_uint8x4xm1` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x4xm2_t vlsseg4wuv_uint8x4xm2` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x4xm1_t vlsseg4wuv_mask_uint16x4xm1` (uint16x4xm1\_t *merge*, const unsigned short *\*address*, long *stride*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `uint16x4xm2_t vlsseg4wuv_mask_uint16x4xm2` (uint16x4xm2\_t *merge*, const unsigned short *\*address*, long *stride*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- `uint32x4xm1_t vlsseg4wuv_mask_uint32x4xm1` (uint32x4xm1\_t *merge*, const unsigned int *\*address*, long *stride*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- `uint32x4xm2_t vlsseg4wuv_mask_uint32x4xm2` (uint32x4xm2\_t *merge*, const unsigned int *\*address*, long *stride*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- `uint64x4xm1_t vlsseg4wuv_mask_uint64x4xm1` (uint64x4xm1\_t *merge*, const unsigned long *\*address*, long *stride*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- `uint64x4xm2_t vlsseg4wuv_mask_uint64x4xm2` (uint64x4xm2\_t *merge*, const unsigned long *\*address*, long *stride*, *e64xm2\_t* *mask*, unsigned int *gvl*)

- `uint8x4xm1_t vlsseg4wuv_mask_uint8x4xm1` (`uint8x4xm1_t merge`, `const unsigned char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)
- `uint8x4xm2_t vlsseg4wuv_mask_uint8x4xm2` (`uint8x4xm2_t merge`, `const unsigned char *address`, `long stride`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.71 Load 5 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg5b.v`]

**Prototypes:**

- `int16x5xm1_t vlsseg5bv_int16x5xm1` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int32x5xm1_t vlsseg5bv_int32x5xm1` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int64x5xm1_t vlsseg5bv_int64x5xm1` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int8x5xm1_t vlsseg5bv_int8x5xm1` (`const signed char *address`, `long stride`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x5xm1_t vlsseg5bv_mask_int16x5xm1` (`int16x5xm1_t merge`, `const short *address`, `long stride`, `e16xm1_t mask`, `unsigned int gvl`)
- `int32x5xm1_t vlsseg5bv_mask_int32x5xm1` (`int32x5xm1_t merge`, `const int *address`, `long stride`, `e32xm1_t mask`, `unsigned int gvl`)
- `int64x5xm1_t vlsseg5bv_mask_int64x5xm1` (`int64x5xm1_t merge`, `const long *address`, `long stride`, `e64xm1_t mask`, `unsigned int gvl`)
- `int8x5xm1_t vlsseg5bv_mask_int8x5xm1` (`int8x5xm1_t merge`, `const signed char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.72 Load 5 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg5bu.v'`]

**Prototypes:**

- `uint16x5xml_t vlsseg5bu_v_uint16x5xml` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x5xml_t vlsseg5bu_v_uint32x5xml` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x5xml_t vlsseg5bu_v_uint64x5xml` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x5xml_t vlsseg5bu_v_uint8x5xml` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x5xml_t vlsseg5bu_mask_uint16x5xml` (uint16x5xml\_t *merge*, const unsigned short *\*address*, long *stride*, *e16xml\_t* *mask*, unsigned int *gvl*)
- `uint32x5xml_t vlsseg5bu_mask_uint32x5xml` (uint32x5xml\_t *merge*, const unsigned int *\*address*, long *stride*, *e32xml\_t* *mask*, unsigned int *gvl*)
- `uint64x5xml_t vlsseg5bu_mask_uint64x5xml` (uint64x5xml\_t *merge*, const unsigned long *\*address*, long *stride*, *e64xml\_t* *mask*, unsigned int *gvl*)
- `uint8x5xml_t vlsseg5bu_mask_uint8x5xml` (uint8x5xml\_t *merge*, const unsigned char *\*address*, long *stride*, *e8xml\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.73 Load 5 contiguous element fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg5e.v'`]

**Prototypes:**

- `float16x5xml_t vlsseg5ev_float16x5xml` (const float16\_t *\*address*, long *stride*, unsigned int *gvl*)

- `float32x5xm1_t vlsseg5ev_float32x5xm1` (const float *\*address*, long *stride*, unsigned int *gvl*)
- `float64x5xm1_t vlsseg5ev_float64x5xm1` (const double *\*address*, long *stride*, unsigned int *gvl*)
- `int16x5xm1_t vlsseg5ev_int16x5xm1` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int32x5xm1_t vlsseg5ev_int32x5xm1` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64x5xm1_t vlsseg5ev_int64x5xm1` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8x5xm1_t vlsseg5ev_int8x5xm1` (const signed char *\*address*, long *stride*, unsigned int *gvl*)
- `uint16x5xm1_t vlsseg5ev_uint16x5xm1` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x5xm1_t vlsseg5ev_uint32x5xm1` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x5xm1_t vlsseg5ev_uint64x5xm1` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x5xm1_t vlsseg5ev_uint8x5xm1` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x5xm1_t vlsseg5ev_mask_float16x5xm1` (`float16x5xm1_t merge`, const float *\*address*, long *stride*, `e16xm1_t mask`, unsigned int *gvl*)
- `float32x5xm1_t vlsseg5ev_mask_float32x5xm1` (`float32x5xm1_t merge`, const float *\*address*, long *stride*, `e32xm1_t mask`, unsigned int *gvl*)
- `float64x5xm1_t vlsseg5ev_mask_float64x5xm1` (`float64x5xm1_t merge`, const double *\*address*, long *stride*, `e64xm1_t mask`, unsigned int *gvl*)
- `int16x5xm1_t vlsseg5ev_mask_int16x5xm1` (`int16x5xm1_t merge`, const short *\*address*, long *stride*, `e16xm1_t mask`, unsigned int *gvl*)
- `int32x5xm1_t vlsseg5ev_mask_int32x5xm1` (`int32x5xm1_t merge`, const int *\*address*, long *stride*, `e32xm1_t mask`, unsigned int *gvl*)
- `int64x5xm1_t vlsseg5ev_mask_int64x5xm1` (`int64x5xm1_t merge`, const long *\*address*, long *stride*, `e64xm1_t mask`, unsigned int *gvl*)
- `int8x5xm1_t vlsseg5ev_mask_int8x5xm1` (`int8x5xm1_t merge`, const signed char *\*address*, long *stride*, `e8xm1_t mask`, unsigned int *gvl*)
- `uint16x5xm1_t vlsseg5ev_mask_uint16x5xm1` (`uint16x5xm1_t merge`, const unsigned short *\*address*, long *stride*, `e16xm1_t mask`, unsigned int *gvl*)
- `uint32x5xm1_t vlsseg5ev_mask_uint32x5xm1` (`uint32x5xm1_t merge`, const unsigned int *\*address*, long *stride*, `e32xm1_t mask`, unsigned int *gvl*)
- `uint64x5xm1_t vlsseg5ev_mask_uint64x5xm1` (`uint64x5xm1_t merge`, const unsigned long *\*address*, long *stride*, `e64xm1_t mask`, unsigned int *gvl*)

- `uint8x5xm1_t vlsseg5ev_mask_uint8x5xm1` (`uint8x5xm1_t merge`, `const unsigned char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.74 Load 5 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg5h.v`]

**Prototypes:**

- `int16x5xm1_t vlsseg5hv_int16x5xm1` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int32x5xm1_t vlsseg5hv_int32x5xm1` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int64x5xm1_t vlsseg5hv_int64x5xm1` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int8x5xm1_t vlsseg5hv_int8x5xm1` (`const signed char *address`, `long stride`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x5xm1_t vlsseg5hv_mask_int16x5xm1` (`int16x5xm1_t merge`, `const short *address`, `long stride`, `e16xm1_t mask`, `unsigned int gvl`)
- `int32x5xm1_t vlsseg5hv_mask_int32x5xm1` (`int32x5xm1_t merge`, `const int *address`, `long stride`, `e32xm1_t mask`, `unsigned int gvl`)
- `int64x5xm1_t vlsseg5hv_mask_int64x5xm1` (`int64x5xm1_t merge`, `const long *address`, `long stride`, `e64xm1_t mask`, `unsigned int gvl`)
- `int8x5xm1_t vlsseg5hv_mask_int8x5xm1` (`int8x5xm1_t merge`, `const signed char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```



## 2.10.75 Load 5 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg5hu.v'`]

**Prototypes:**

- `uint16x5xml_t vlsseg5huv_uint16x5xml` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x5xml_t vlsseg5huv_uint32x5xml` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x5xml_t vlsseg5huv_uint64x5xml` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x5xml_t vlsseg5huv_uint8x5xml` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x5xml_t vlsseg5huv_mask_uint16x5xml` (uint16x5xml\_t *merge*, const unsigned short *\*address*, long *stride*, *e16xml\_t* *mask*, unsigned int *gvl*)
- `uint32x5xml_t vlsseg5huv_mask_uint32x5xml` (uint32x5xml\_t *merge*, const unsigned int *\*address*, long *stride*, *e32xml\_t* *mask*, unsigned int *gvl*)
- `uint64x5xml_t vlsseg5huv_mask_uint64x5xml` (uint64x5xml\_t *merge*, const unsigned long *\*address*, long *stride*, *e64xml\_t* *mask*, unsigned int *gvl*)
- `uint8x5xml_t vlsseg5huv_mask_uint8x5xml` (uint8x5xml\_t *merge*, const unsigned char *\*address*, long *stride*, *e8xml\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.76 Load 5 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg5w.v'`]

**Prototypes:**

- `int16x5xml_t vlsseg5wv_int16x5xml` (const short *\*address*, long *stride*, unsigned int *gvl*)

- `int32x5xml_t vlsseg5wv_int32x5xml` (const int \**address*, long *stride*, unsigned int *gvl*)
- `int64x5xml_t vlsseg5wv_int64x5xml` (const long \**address*, long *stride*, unsigned int *gvl*)
- `int8x5xml_t vlsseg5wv_int8x5xml` (const signed char \**address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x5xml_t vlsseg5wv_mask_int16x5xml` (int16x5xml\_t *merge*, const short \**address*, long *stride*, `e16xml_t` *mask*, unsigned int *gvl*)
- `int32x5xml_t vlsseg5wv_mask_int32x5xml` (int32x5xml\_t *merge*, const int \**address*, long *stride*, `e32xml_t` *mask*, unsigned int *gvl*)
- `int64x5xml_t vlsseg5wv_mask_int64x5xml` (int64x5xml\_t *merge*, const long \**address*, long *stride*, `e64xml_t` *mask*, unsigned int *gvl*)
- `int8x5xml_t vlsseg5wv_mask_int8x5xml` (int8x5xml\_t *merge*, const signed char \**address*, long *stride*, `e8xml_t` *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.77 Load 5 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg5wu.v`]

**Prototypes:**

- `uint16x5xml_t vlsseg5wuv_uint16x5xml` (const unsigned short \**address*, long *stride*, unsigned int *gvl*)
- `uint32x5xml_t vlsseg5wuv_uint32x5xml` (const unsigned int \**address*, long *stride*, unsigned int *gvl*)
- `uint64x5xml_t vlsseg5wuv_uint64x5xml` (const unsigned long \**address*, long *stride*, unsigned int *gvl*)
- `uint8x5xml_t vlsseg5wuv_uint8x5xml` (const unsigned char \**address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x5xml_t vlsseg5wuv_mask_uint16x5xml` (`uint16x5xml_t merge`, `const unsigned short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)
- `uint32x5xml_t vlsseg5wuv_mask_uint32x5xml` (`uint32x5xml_t merge`, `const unsigned int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `uint64x5xml_t vlsseg5wuv_mask_uint64x5xml` (`uint64x5xml_t merge`, `const unsigned long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `uint8x5xml_t vlsseg5wuv_mask_uint8x5xml` (`uint8x5xml_t merge`, `const unsigned char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.78 Load 6 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers****Instruction:** [`vlsseg6b.v`]**Prototypes:**

- `int16x6xml_t vlsseg6bv_int16x6xml` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int32x6xml_t vlsseg6bv_int32x6xml` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int64x6xml_t vlsseg6bv_int64x6xml` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int8x6xml_t vlsseg6bv_int8x6xml` (`const signed char *address`, `long stride`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x6xml_t vlsseg6bv_mask_int16x6xml` (`int16x6xml_t merge`, `const short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x6xml_t vlsseg6bv_mask_int32x6xml` (`int32x6xml_t merge`, `const int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x6xml_t vlsseg6bv_mask_int64x6xml` (`int64x6xml_t merge`, `const long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x6xml_t vlsseg6bv_mask_int8x6xml` (`int8x6xml_t merge`, `const signed char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.79 Load 6 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg6bu.v'`]

**Prototypes:**

- `uint16x6xml_t vlsseg6bu_uint16x6xml` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x6xml_t vlsseg6bu_uint32x6xml` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x6xml_t vlsseg6bu_uint64x6xml` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x6xml_t vlsseg6bu_uint8x6xml` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x6xml_t vlsseg6bu_mask_uint16x6xml` (uint16x6xml\_t *merge*, const unsigned short *\*address*, long *stride*, *e16xml\_t* *mask*, unsigned int *gvl*)
- `uint32x6xml_t vlsseg6bu_mask_uint32x6xml` (uint32x6xml\_t *merge*, const unsigned int *\*address*, long *stride*, *e32xml\_t* *mask*, unsigned int *gvl*)
- `uint64x6xml_t vlsseg6bu_mask_uint64x6xml` (uint64x6xml\_t *merge*, const unsigned long *\*address*, long *stride*, *e64xml\_t* *mask*, unsigned int *gvl*)
- `uint8x6xml_t vlsseg6bu_mask_uint8x6xml` (uint8x6xml\_t *merge*, const unsigned char *\*address*, long *stride*, *e8xml\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
```

(continues on next page)

(continued from previous page)

```
result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.80 Load 6 contiguous element fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg6e.v`]

**Prototypes:**

- `float16x6xml_t vlsseg6ev_float16x6xml` (const float16\_t \*address, long stride, unsigned int gvl)
- `float32x6xml_t vlsseg6ev_float32x6xml` (const float \*address, long stride, unsigned int gvl)
- `float64x6xml_t vlsseg6ev_float64x6xml` (const double \*address, long stride, unsigned int gvl)
- `int16x6xml_t vlsseg6ev_int16x6xml` (const short \*address, long stride, unsigned int gvl)
- `int32x6xml_t vlsseg6ev_int32x6xml` (const int \*address, long stride, unsigned int gvl)
- `int64x6xml_t vlsseg6ev_int64x6xml` (const long \*address, long stride, unsigned int gvl)
- `int8x6xml_t vlsseg6ev_int8x6xml` (const signed char \*address, long stride, unsigned int gvl)
- `uint16x6xml_t vlsseg6ev_uint16x6xml` (const unsigned short \*address, long stride, unsigned int gvl)
- `uint32x6xml_t vlsseg6ev_uint32x6xml` (const unsigned int \*address, long stride, unsigned int gvl)
- `uint64x6xml_t vlsseg6ev_uint64x6xml` (const unsigned long \*address, long stride, unsigned int gvl)
- `uint8x6xml_t vlsseg6ev_uint8x6xml` (const unsigned char \*address, long stride, unsigned int gvl)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x6xml_t vlsseg6ev_mask_float16x6xml` (float16x6xml\_t merge, const float16\_t \*address, long stride, e16xml\_t mask, unsigned int gvl)
- `float32x6xml_t vlsseg6ev_mask_float32x6xml` (float32x6xml\_t merge, const float \*address, long stride, e32xml\_t mask, unsigned int gvl)
- `float64x6xml_t vlsseg6ev_mask_float64x6xml` (float64x6xml\_t merge, const double \*address, long stride, e64xml\_t mask, unsigned int gvl)
- `int16x6xml_t vlsseg6ev_mask_int16x6xml` (int16x6xml\_t merge, const short \*address, long stride, e16xml\_t mask, unsigned int gvl)
- `int32x6xml_t vlsseg6ev_mask_int32x6xml` (int32x6xml\_t merge, const int \*address, long stride, e32xml\_t mask, unsigned int gvl)
- `int64x6xml_t vlsseg6ev_mask_int64x6xml` (int64x6xml\_t merge, const long \*address, long stride, e64xml\_t mask, unsigned int gvl)

- `int8x6xml_t vlsseg6ev_mask_int8x6xml` (`int8x6xml_t merge`, `const signed char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)
- `uint16x6xml_t vlsseg6ev_mask_uint16x6xml` (`uint16x6xml_t merge`, `const unsigned short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)
- `uint32x6xml_t vlsseg6ev_mask_uint32x6xml` (`uint32x6xml_t merge`, `const unsigned int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `uint64x6xml_t vlsseg6ev_mask_uint64x6xml` (`uint64x6xml_t merge`, `const unsigned long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `uint8x6xml_t vlsseg6ev_mask_uint8x6xml` (`uint8x6xml_t merge`, `const unsigned char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.81 Load 6 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg6h.v'`]

**Prototypes:**

- `int16x6xml_t vlsseg6hv_int16x6xml` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int32x6xml_t vlsseg6hv_int32x6xml` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int64x6xml_t vlsseg6hv_int64x6xml` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int8x6xml_t vlsseg6hv_int8x6xml` (`const signed char *address`, `long stride`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x6xml_t vlsseg6hv_mask_int16x6xml` (`int16x6xml_t merge`, `const short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x6xml_t vlsseg6hv_mask_int32x6xml` (`int32x6xml_t merge`, `const int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x6xml_t vlsseg6hv_mask_int64x6xml` (`int64x6xml_t merge`, `const long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x6xml_t vlsseg6hv_mask_int8x6xml` (`int8x6xml_t merge`, `const signed char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)



**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.82 Load 6 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg6hu.v'`]

**Prototypes:**

- `uint16x6xml_t vlsseg6huv_uint16x6xml` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x6xml_t vlsseg6huv_uint32x6xml` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x6xml_t vlsseg6huv_uint64x6xml` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x6xml_t vlsseg6huv_uint8x6xml` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x6xml_t vlsseg6huv_mask_uint16x6xml` (`uint16x6xml_t merge`, const unsigned short *\*address*, long *stride*, `e16xml_t mask`, unsigned int *gvl*)
- `uint32x6xml_t vlsseg6huv_mask_uint32x6xml` (`uint32x6xml_t merge`, const unsigned int *\*address*, long *stride*, `e32xml_t mask`, unsigned int *gvl*)
- `uint64x6xml_t vlsseg6huv_mask_uint64x6xml` (`uint64x6xml_t merge`, const unsigned long *\*address*, long *stride*, `e64xml_t mask`, unsigned int *gvl*)
- `uint8x6xml_t vlsseg6huv_mask_uint8x6xml` (`uint8x6xml_t merge`, const unsigned char *\*address*, long *stride*, `e8xml_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
```

(continues on next page)

(continued from previous page)

```
result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.83 Load 6 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** ['vlsseg6w.v']

**Prototypes:**

- `int16x6xml_t vlsseg6wv_int16x6xml` (const short \*address, long stride, unsigned int gvl)
- `int32x6xml_t vlsseg6wv_int32x6xml` (const int \*address, long stride, unsigned int gvl)
- `int64x6xml_t vlsseg6wv_int64x6xml` (const long \*address, long stride, unsigned int gvl)
- `int8x6xml_t vlsseg6wv_int8x6xml` (const signed char \*address, long stride, unsigned int gvl)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x6xml_t vlsseg6wv_mask_int16x6xml` (int16x6xml\_t merge, const short \*address, long stride, e16xml\_t mask, unsigned int gvl)
- `int32x6xml_t vlsseg6wv_mask_int32x6xml` (int32x6xml\_t merge, const int \*address, long stride, e32xml\_t mask, unsigned int gvl)
- `int64x6xml_t vlsseg6wv_mask_int64x6xml` (int64x6xml\_t merge, const long \*address, long stride, e64xml\_t mask, unsigned int gvl)
- `int8x6xml_t vlsseg6wv_mask_int8x6xml` (int8x6xml\_t merge, const signed char \*address, long stride, e8xml\_t mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.84 Load 6 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** ['vlsseg6wu.v']

**Prototypes:**

- `uint16x6xml_t vlsseg6wuv_uint16x6xml` (const unsigned short \*address, long stride, unsigned int gvl)

- `uint32x6xml_t vlsseg6wuv_uint32x6xml` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x6xml_t vlsseg6wuv_uint64x6xml` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x6xml_t vlsseg6wuv_uint8x6xml` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x6xml_t vlsseg6wuv_mask_uint16x6xml` (uint16x6xml\_t *merge*, const unsigned short *\*address*, long *stride*, *e16xml\_t* *mask*, unsigned int *gvl*)
- `uint32x6xml_t vlsseg6wuv_mask_uint32x6xml` (uint32x6xml\_t *merge*, const unsigned int *\*address*, long *stride*, *e32xml\_t* *mask*, unsigned int *gvl*)
- `uint64x6xml_t vlsseg6wuv_mask_uint64x6xml` (uint64x6xml\_t *merge*, const unsigned long *\*address*, long *stride*, *e64xml\_t* *mask*, unsigned int *gvl*)
- `uint8x6xml_t vlsseg6wuv_mask_uint8x6xml` (uint8x6xml\_t *merge*, const unsigned char *\*address*, long *stride*, *e8xml\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.85 Load 7 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg7b.v`]

**Prototypes:**

- `int16x7xml_t vlsseg7bv_int16x7xml` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int32x7xml_t vlsseg7bv_int32x7xml` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64x7xml_t vlsseg7bv_int64x7xml` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8x7xml_t vlsseg7bv_int8x7xml` (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x7xml_t vlsseg7bv_mask_int16x7xml` (`int16x7xml_t merge`, `const short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x7xml_t vlsseg7bv_mask_int32x7xml` (`int32x7xml_t merge`, `const int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x7xml_t vlsseg7bv_mask_int64x7xml` (`int64x7xml_t merge`, `const long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x7xml_t vlsseg7bv_mask_int8x7xml` (`int8x7xml_t merge`, `const signed char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.86 Load 7 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg7bu.v'`]

**Prototypes:**

- `uint16x7xml_t vlsseg7buv_uint16x7xml` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint32x7xml_t vlsseg7buv_uint32x7xml` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint64x7xml_t vlsseg7buv_uint64x7xml` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint8x7xml_t vlsseg7buv_uint8x7xml` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x7xml_t vlsseg7buv_mask_uint16x7xml` (`uint16x7xml_t merge`, `const unsigned short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)

- `uint32x7xml_t vlsseg7buv_mask_uint32x7xml` (`uint32x7xml_t merge`, `const unsigned int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `uint64x7xml_t vlsseg7buv_mask_uint64x7xml` (`uint64x7xml_t merge`, `const unsigned long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `uint8x7xml_t vlsseg7buv_mask_uint8x7xml` (`uint8x7xml_t merge`, `const unsigned char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.87 Load 7 contiguous element fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg7e.v`]

**Prototypes:**

- `float16x7xml_t vlsseg7ev_float16x7xml` (`const float16_t *address`, `long stride`, `unsigned int gvl`)
- `float32x7xml_t vlsseg7ev_float32x7xml` (`const float *address`, `long stride`, `unsigned int gvl`)
- `float64x7xml_t vlsseg7ev_float64x7xml` (`const double *address`, `long stride`, `unsigned int gvl`)
- `int16x7xml_t vlsseg7ev_int16x7xml` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int32x7xml_t vlsseg7ev_int32x7xml` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int64x7xml_t vlsseg7ev_int64x7xml` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int8x7xml_t vlsseg7ev_int8x7xml` (`const signed char *address`, `long stride`, `unsigned int gvl`)
- `uint16x7xml_t vlsseg7ev_uint16x7xml` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint32x7xml_t vlsseg7ev_uint32x7xml` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint64x7xml_t vlsseg7ev_uint64x7xml` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint8x7xml_t vlsseg7ev_uint8x7xml` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x7xml_t vlsseg7ev_mask_float16x7xml` (`float16x7xml_t merge`, `const float16_t *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)
- `float32x7xml_t vlsseg7ev_mask_float32x7xml` (`float32x7xml_t merge`, `const float *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `float64x7xml_t vlsseg7ev_mask_float64x7xml` (`float64x7xml_t merge`, `const double *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `int16x7xml_t vlsseg7ev_mask_int16x7xml` (`int16x7xml_t merge`, `const short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x7xml_t vlsseg7ev_mask_int32x7xml` (`int32x7xml_t merge`, `const int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x7xml_t vlsseg7ev_mask_int64x7xml` (`int64x7xml_t merge`, `const long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x7xml_t vlsseg7ev_mask_int8x7xml` (`int8x7xml_t merge`, `const signed char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)
- `uint16x7xml_t vlsseg7ev_mask_uint16x7xml` (`uint16x7xml_t merge`, `const unsigned short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)
- `uint32x7xml_t vlsseg7ev_mask_uint32x7xml` (`uint32x7xml_t merge`, `const unsigned int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `uint64x7xml_t vlsseg7ev_mask_uint64x7xml` (`uint64x7xml_t merge`, `const unsigned long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `uint8x7xml_t vlsseg7ev_mask_uint8x7xml` (`uint8x7xml_t merge`, `const unsigned char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.88 Load 7 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers****Instruction:** [`'vlsseg7h.v'`]**Prototypes:**

- `int16x7xml_t vlsseg7hv_int16x7xml` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int32x7xml_t vlsseg7hv_int32x7xml` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int64x7xml_t vlsseg7hv_int64x7xml` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int8x7xml_t vlsseg7hv_int8x7xml` (`const signed char *address`, `long stride`, `unsigned int gvl`)

**Operation:**



```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x7xml_t vlsseg7hv_mask_int16x7xml` (`int16x7xml_t merge`, `const short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x7xml_t vlsseg7hv_mask_int32x7xml` (`int32x7xml_t merge`, `const int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x7xml_t vlsseg7hv_mask_int64x7xml` (`int64x7xml_t merge`, `const long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x7xml_t vlsseg7hv_mask_int8x7xml` (`int8x7xml_t merge`, `const signed char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.89 Load 7 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg7hu.v'`]

**Prototypes:**

- `uint16x7xml_t vlsseg7huv_uint16x7xml` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint32x7xml_t vlsseg7huv_uint32x7xml` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint64x7xml_t vlsseg7huv_uint64x7xml` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint8x7xml_t vlsseg7huv_uint8x7xml` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x7xml_t vlsseg7huv_mask_uint16x7xml` (`uint16x7xml_t merge`, `const unsigned short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)

- `uint32x7xml_t vlsseg7huv_mask_uint32x7xml` (`uint32x7xml_t merge`, `const unsigned int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `uint64x7xml_t vlsseg7huv_mask_uint64x7xml` (`uint64x7xml_t merge`, `const unsigned long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `uint8x7xml_t vlsseg7huv_mask_uint8x7xml` (`uint8x7xml_t merge`, `const unsigned char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.90 Load 7 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg7w.v'`]

**Prototypes:**

- `int16x7xml_t vlsseg7wv_int16x7xml` (`const short *address`, `long stride`, `unsigned int gvl`)
- `int32x7xml_t vlsseg7wv_int32x7xml` (`const int *address`, `long stride`, `unsigned int gvl`)
- `int64x7xml_t vlsseg7wv_int64x7xml` (`const long *address`, `long stride`, `unsigned int gvl`)
- `int8x7xml_t vlsseg7wv_int8x7xml` (`const signed char *address`, `long stride`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x7xml_t vlsseg7wv_mask_int16x7xml` (`int16x7xml_t merge`, `const short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x7xml_t vlsseg7wv_mask_int32x7xml` (`int32x7xml_t merge`, `const int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x7xml_t vlsseg7wv_mask_int64x7xml` (`int64x7xml_t merge`, `const long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x7xml_t vlsseg7wv_mask_int8x7xml` (`int8x7xml_t merge`, `const signed char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
```

(continues on next page)

(continued from previous page)

```

    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
else
    result[segment] = merge[segment]
result[gvl : VLMAX] = 0

```

## 2.10.91 Load 7 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`vlsseg7wuv.v`']

**Prototypes:**

- `uint16x7xml_t vlsseg7wuv_uint16x7xml` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x7xml_t vlsseg7wuv_uint32x7xml` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x7xml_t vlsseg7wuv_uint64x7xml` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x7xml_t vlsseg7wuv_uint8x7xml` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```

>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- `uint16x7xml_t vlsseg7wuv_mask_uint16x7xml` (uint16x7xml\_t *merge*, const unsigned short *\*address*, long *stride*, *e16xml\_t* *mask*, unsigned int *gvl*)
- `uint32x7xml_t vlsseg7wuv_mask_uint32x7xml` (uint32x7xml\_t *merge*, const unsigned int *\*address*, long *stride*, *e32xml\_t* *mask*, unsigned int *gvl*)
- `uint64x7xml_t vlsseg7wuv_mask_uint64x7xml` (uint64x7xml\_t *merge*, const unsigned long *\*address*, long *stride*, *e64xml\_t* *mask*, unsigned int *gvl*)
- `uint8x7xml_t vlsseg7wuv_mask_uint8x7xml` (uint8x7xml\_t *merge*, const unsigned char *\*address*, long *stride*, *e8xml\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```

>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0

```

## 2.10.92 Load 8 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg8b.v'`]

**Prototypes:**

- `int16x8xml_t vlsseg8bv_int16x8xml` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int32x8xml_t vlsseg8bv_int32x8xml` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64x8xml_t vlsseg8bv_int64x8xml` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8x8xml_t vlsseg8bv_int8x8xml` (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x8xml_t vlsseg8bv_mask_int16x8xml` (int16x8xml\_t *merge*, const short *\*address*, long *stride*, *e16xml\_t* *mask*, unsigned int *gvl*)
- `int32x8xml_t vlsseg8bv_mask_int32x8xml` (int32x8xml\_t *merge*, const int *\*address*, long *stride*, *e32xml\_t* *mask*, unsigned int *gvl*)
- `int64x8xml_t vlsseg8bv_mask_int64x8xml` (int64x8xml\_t *merge*, const long *\*address*, long *stride*, *e64xml\_t* *mask*, unsigned int *gvl*)
- `int8x8xml_t vlsseg8bv_mask_int8x8xml` (int8x8xml\_t *merge*, const signed char *\*address*, long *stride*, *e8xml\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.93 Load 8 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg8bu.v'`]

**Prototypes:**

- `uint16x8xml_t vlsseg8buv_uint16x8xml` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x8xml_t vlsseg8buv_uint32x8xml` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x8xml_t vlsseg8buv_uint64x8xml` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)

- `uint8x8xml_t vlsseg8buu_uint8x8xml` (const unsigned char \**address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x8xml_t vlsseg8buu_mask_uint16x8xml` (uint16x8xml\_t *merge*, const unsigned short \**address*, long *stride*, *e16xml\_t* *mask*, unsigned int *gvl*)
- `uint32x8xml_t vlsseg8buu_mask_uint32x8xml` (uint32x8xml\_t *merge*, const unsigned int \**address*, long *stride*, *e32xml\_t* *mask*, unsigned int *gvl*)
- `uint64x8xml_t vlsseg8buu_mask_uint64x8xml` (uint64x8xml\_t *merge*, const unsigned long \**address*, long *stride*, *e64xml\_t* *mask*, unsigned int *gvl*)
- `uint8x8xml_t vlsseg8buu_mask_uint8x8xml` (uint8x8xml\_t *merge*, const unsigned char \**address*, long *stride*, *e8xml\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.94 Load 8 contiguous element fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`*vlsseg8e.v`']

**Prototypes:**

- `float16x8xml_t vlsseg8ev_float16x8xml` (const float16\_t \**address*, long *stride*, unsigned int *gvl*)
- `float32x8xml_t vlsseg8ev_float32x8xml` (const float \**address*, long *stride*, unsigned int *gvl*)
- `float64x8xml_t vlsseg8ev_float64x8xml` (const double \**address*, long *stride*, unsigned int *gvl*)
- `int16x8xml_t vlsseg8ev_int16x8xml` (const short \**address*, long *stride*, unsigned int *gvl*)
- `int32x8xml_t vlsseg8ev_int32x8xml` (const int \**address*, long *stride*, unsigned int *gvl*)
- `int64x8xml_t vlsseg8ev_int64x8xml` (const long \**address*, long *stride*, unsigned int *gvl*)
- `int8x8xml_t vlsseg8ev_int8x8xml` (const signed char \**address*, long *stride*, unsigned int *gvl*)
- `uint16x8xml_t vlsseg8ev_uint16x8xml` (const unsigned short \**address*, long *stride*, unsigned int *gvl*)
- `uint32x8xml_t vlsseg8ev_uint32x8xml` (const unsigned int \**address*, long *stride*, unsigned int *gvl*)

- `uint64x8xml_t vlsseg8ev_uint64x8xml` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)
- `uint8x8xml_t vlsseg8ev_uint8x8xml` (const unsigned char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x8xml_t vlsseg8ev_mask_float16x8xml` (`float16x8xml_t merge`, const `float16_t *address`, long *stride*, `e16xml_t mask`, unsigned int *gvl*)
- `float32x8xml_t vlsseg8ev_mask_float32x8xml` (`float32x8xml_t merge`, const `float *address`, long *stride*, `e32xml_t mask`, unsigned int *gvl*)
- `float64x8xml_t vlsseg8ev_mask_float64x8xml` (`float64x8xml_t merge`, const `double *address`, long *stride*, `e64xml_t mask`, unsigned int *gvl*)
- `int16x8xml_t vlsseg8ev_mask_int16x8xml` (`int16x8xml_t merge`, const `short *address`, long *stride*, `e16xml_t mask`, unsigned int *gvl*)
- `int32x8xml_t vlsseg8ev_mask_int32x8xml` (`int32x8xml_t merge`, const `int *address`, long *stride*, `e32xml_t mask`, unsigned int *gvl*)
- `int64x8xml_t vlsseg8ev_mask_int64x8xml` (`int64x8xml_t merge`, const `long *address`, long *stride*, `e64xml_t mask`, unsigned int *gvl*)
- `int8x8xml_t vlsseg8ev_mask_int8x8xml` (`int8x8xml_t merge`, const `signed char *address`, long *stride*, `e8xml_t mask`, unsigned int *gvl*)
- `uint16x8xml_t vlsseg8ev_mask_uint16x8xml` (`uint16x8xml_t merge`, const unsigned `short *address`, long *stride*, `e16xml_t mask`, unsigned int *gvl*)
- `uint32x8xml_t vlsseg8ev_mask_uint32x8xml` (`uint32x8xml_t merge`, const unsigned `int *address`, long *stride*, `e32xml_t mask`, unsigned int *gvl*)
- `uint64x8xml_t vlsseg8ev_mask_uint64x8xml` (`uint64x8xml_t merge`, const unsigned `long *address`, long *stride*, `e64xml_t mask`, unsigned int *gvl*)
- `uint8x8xml_t vlsseg8ev_mask_uint8x8xml` (`uint8x8xml_t merge`, const unsigned `char *address`, long *stride*, `e8xml_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```



## 2.10.95 Load 8 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg8h.v'`]

**Prototypes:**

- `int16x8xml_t vlsseg8hv_int16x8xml` (const short *\*address*, long *stride*, unsigned int *gvl*)
- `int32x8xml_t vlsseg8hv_int32x8xml` (const int *\*address*, long *stride*, unsigned int *gvl*)
- `int64x8xml_t vlsseg8hv_int64x8xml` (const long *\*address*, long *stride*, unsigned int *gvl*)
- `int8x8xml_t vlsseg8hv_int8x8xml` (const signed char *\*address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x8xml_t vlsseg8hv_mask_int16x8xml` (int16x8xml\_t *merge*, const short *\*address*, long *stride*, *e16xml\_t* *mask*, unsigned int *gvl*)
- `int32x8xml_t vlsseg8hv_mask_int32x8xml` (int32x8xml\_t *merge*, const int *\*address*, long *stride*, *e32xml\_t* *mask*, unsigned int *gvl*)
- `int64x8xml_t vlsseg8hv_mask_int64x8xml` (int64x8xml\_t *merge*, const long *\*address*, long *stride*, *e64xml\_t* *mask*, unsigned int *gvl*)
- `int8x8xml_t vlsseg8hv_mask_int8x8xml` (int8x8xml\_t *merge*, const signed char *\*address*, long *stride*, *e8xml\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.96 Load 8 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg8hu.v'`]

**Prototypes:**

- `uint16x8xml_t vlsseg8huv_uint16x8xml` (const unsigned short *\*address*, long *stride*, unsigned int *gvl*)
- `uint32x8xml_t vlsseg8huv_uint32x8xml` (const unsigned int *\*address*, long *stride*, unsigned int *gvl*)
- `uint64x8xml_t vlsseg8huv_uint64x8xml` (const unsigned long *\*address*, long *stride*, unsigned int *gvl*)

- `uint8x8xml_t vlsseg8huv_uint8x8xml` (const unsigned char \**address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x8xml_t vlsseg8huv_mask_uint16x8xml` (uint16x8xml\_t *merge*, const unsigned short \**address*, long *stride*, `e16xml_t` *mask*, unsigned int *gvl*)
- `uint32x8xml_t vlsseg8huv_mask_uint32x8xml` (uint32x8xml\_t *merge*, const unsigned int \**address*, long *stride*, `e32xml_t` *mask*, unsigned int *gvl*)
- `uint64x8xml_t vlsseg8huv_mask_uint64x8xml` (uint64x8xml\_t *merge*, const unsigned long \**address*, long *stride*, `e64xml_t` *mask*, unsigned int *gvl*)
- `uint8x8xml_t vlsseg8huv_mask_uint8x8xml` (uint8x8xml\_t *merge*, const unsigned char \**address*, long *stride*, `e8xml_t` *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.97 Load 8 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`*vlsseg8w.v`']

**Prototypes:**

- `int16x8xml_t vlsseg8wv_int16x8xml` (const short \**address*, long *stride*, unsigned int *gvl*)
- `int32x8xml_t vlsseg8wv_int32x8xml` (const int \**address*, long *stride*, unsigned int *gvl*)
- `int64x8xml_t vlsseg8wv_int64x8xml` (const long \**address*, long *stride*, unsigned int *gvl*)
- `int8x8xml_t vlsseg8wv_int8x8xml` (const signed char \**address*, long *stride*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + stride
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x8xml_t vlsseg8wv_mask_int16x8xml` (`int16x8xml_t merge`, `const short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x8xml_t vlsseg8wv_mask_int32x8xml` (`int32x8xml_t merge`, `const int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x8xml_t vlsseg8wv_mask_int64x8xml` (`int64x8xml_t merge`, `const long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x8xml_t vlsseg8wv_mask_int8x8xml` (`int8x8xml_t merge`, `const signed char *address`, `long stride`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
      if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
      else
        result[segment] = merge[segment]
    result[gvl : VLMAX] = 0
```

## 2.10.98 Load 8 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers

**Instruction:** [`'vlsseg8wuv.v'`]

**Prototypes:**

- `uint16x8xml_t vlsseg8wuv_uint16x8xml` (`const unsigned short *address`, `long stride`, `unsigned int gvl`)
- `uint32x8xml_t vlsseg8wuv_uint32x8xml` (`const unsigned int *address`, `long stride`, `unsigned int gvl`)
- `uint64x8xml_t vlsseg8wuv_uint64x8xml` (`const unsigned long *address`, `long stride`, `unsigned int gvl`)
- `uint8x8xml_t vlsseg8wuv_uint8x8xml` (`const unsigned char *address`, `long stride`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
      result[segment] = load_segment(address)
      address = address + sizeof(segment) + stride
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x8xml_t vlsseg8wuv_mask_uint16x8xml` (`uint16x8xml_t merge`, `const unsigned short *address`, `long stride`, `e16xml_t mask`, `unsigned int gvl`)
- `uint32x8xml_t vlsseg8wuv_mask_uint32x8xml` (`uint32x8xml_t merge`, `const unsigned int *address`, `long stride`, `e32xml_t mask`, `unsigned int gvl`)
- `uint64x8xml_t vlsseg8wuv_mask_uint64x8xml` (`uint64x8xml_t merge`, `const unsigned long *address`, `long stride`, `e64xml_t mask`, `unsigned int gvl`)

- `uint8x8xm1_t vlsseg8wuv_mask_uint8x8xm1` (`uint8x8xm1_t merge`, `const unsigned char *address`, `long stride`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + stride
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.99 Load 2 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlsseg2b.v`']

**Prototypes:**

- `int16x2xm1_t vlsseg2bv_int16x2xm1_int16xm1` (`const short *address`, `int16xm1_t index`, `unsigned int gvl`)
- `int16x2xm2_t vlsseg2bv_int16x2xm2_int16xm2` (`const short *address`, `int16xm2_t index`, `unsigned int gvl`)
- `int16x2xm4_t vlsseg2bv_int16x2xm4_int16xm4` (`const short *address`, `int16xm4_t index`, `unsigned int gvl`)
- `int32x2xm1_t vlsseg2bv_int32x2xm1_int32xm1` (`const int *address`, `int32xm1_t index`, `unsigned int gvl`)
- `int32x2xm2_t vlsseg2bv_int32x2xm2_int32xm2` (`const int *address`, `int32xm2_t index`, `unsigned int gvl`)
- `int32x2xm4_t vlsseg2bv_int32x2xm4_int32xm4` (`const int *address`, `int32xm4_t index`, `unsigned int gvl`)
- `int64x2xm1_t vlsseg2bv_int64x2xm1_int64xm1` (`const long *address`, `int64xm1_t index`, `unsigned int gvl`)
- `int64x2xm2_t vlsseg2bv_int64x2xm2_int64xm2` (`const long *address`, `int64xm2_t index`, `unsigned int gvl`)
- `int64x2xm4_t vlsseg2bv_int64x2xm4_int64xm4` (`const long *address`, `int64xm4_t index`, `unsigned int gvl`)
- `int8x2xm1_t vlsseg2bv_int8x2xm1_int8xm1` (`const signed char *address`, `int8xm1_t index`, `unsigned int gvl`)
- `int8x2xm2_t vlsseg2bv_int8x2xm2_int8xm2` (`const signed char *address`, `int8xm2_t index`, `unsigned int gvl`)
- `int8x2xm4_t vlsseg2bv_int8x2xm4_int8xm4` (`const signed char *address`, `int8xm4_t index`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x2xm1_t vlxseg2bv_mask_int16x2xm1_int16xm1` (`int16x2xm1_t merge`, `const short *address`, `int16xm1_t index`, `e16xm1_t mask`, `unsigned int gvl`)
- `int16x2xm2_t vlxseg2bv_mask_int16x2xm2_int16xm2` (`int16x2xm2_t merge`, `const short *address`, `int16xm2_t index`, `e16xm2_t mask`, `unsigned int gvl`)
- `int16x2xm4_t vlxseg2bv_mask_int16x2xm4_int16xm4` (`int16x2xm4_t merge`, `const short *address`, `int16xm4_t index`, `e16xm4_t mask`, `unsigned int gvl`)
- `int32x2xm1_t vlxseg2bv_mask_int32x2xm1_int32xm1` (`int32x2xm1_t merge`, `const int *address`, `int32xm1_t index`, `e32xm1_t mask`, `unsigned int gvl`)
- `int32x2xm2_t vlxseg2bv_mask_int32x2xm2_int32xm2` (`int32x2xm2_t merge`, `const int *address`, `int32xm2_t index`, `e32xm2_t mask`, `unsigned int gvl`)
- `int32x2xm4_t vlxseg2bv_mask_int32x2xm4_int32xm4` (`int32x2xm4_t merge`, `const int *address`, `int32xm4_t index`, `e32xm4_t mask`, `unsigned int gvl`)
- `int64x2xm1_t vlxseg2bv_mask_int64x2xm1_int64xm1` (`int64x2xm1_t merge`, `const long *address`, `int64xm1_t index`, `e64xm1_t mask`, `unsigned int gvl`)
- `int64x2xm2_t vlxseg2bv_mask_int64x2xm2_int64xm2` (`int64x2xm2_t merge`, `const long *address`, `int64xm2_t index`, `e64xm2_t mask`, `unsigned int gvl`)
- `int64x2xm4_t vlxseg2bv_mask_int64x2xm4_int64xm4` (`int64x2xm4_t merge`, `const long *address`, `int64xm4_t index`, `e64xm4_t mask`, `unsigned int gvl`)
- `int8x2xm1_t vlxseg2bv_mask_int8x2xm1_int8xm1` (`int8x2xm1_t merge`, `const signed char *address`, `int8xm1_t index`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8x2xm2_t vlxseg2bv_mask_int8x2xm2_int8xm2` (`int8x2xm2_t merge`, `const signed char *address`, `int8xm2_t index`, `e8xm2_t mask`, `unsigned int gvl`)
- `int8x2xm4_t vlxseg2bv_mask_int8x2xm4_int8xm4` (`int8x2xm4_t merge`, `const signed char *address`, `int8xm4_t index`, `e8xm4_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.100 Load 2 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlsxseg2bu.v'`]

**Prototypes:**

- `uint16x2xm1_t vlsxseg2bu_v_uint16x2xm1_uint16xm1` (const unsigned short *\*address*, *uint16xm1\_t* *index*, unsigned int *gvl*)
- `uint16x2xm2_t vlsxseg2bu_v_uint16x2xm2_uint16xm2` (const unsigned short *\*address*, *uint16xm2\_t* *index*, unsigned int *gvl*)
- `uint16x2xm4_t vlsxseg2bu_v_uint16x2xm4_uint16xm4` (const unsigned short *\*address*, *uint16xm4\_t* *index*, unsigned int *gvl*)
- `uint32x2xm1_t vlsxseg2bu_v_uint32x2xm1_uint32xm1` (const unsigned int *\*address*, *uint32xm1\_t* *index*, unsigned int *gvl*)
- `uint32x2xm2_t vlsxseg2bu_v_uint32x2xm2_uint32xm2` (const unsigned int *\*address*, *uint32xm2\_t* *index*, unsigned int *gvl*)
- `uint32x2xm4_t vlsxseg2bu_v_uint32x2xm4_uint32xm4` (const unsigned int *\*address*, *uint32xm4\_t* *index*, unsigned int *gvl*)
- `uint64x2xm1_t vlsxseg2bu_v_uint64x2xm1_uint64xm1` (const unsigned long *\*address*, *uint64xm1\_t* *index*, unsigned int *gvl*)
- `uint64x2xm2_t vlsxseg2bu_v_uint64x2xm2_uint64xm2` (const unsigned long *\*address*, *uint64xm2\_t* *index*, unsigned int *gvl*)
- `uint64x2xm4_t vlsxseg2bu_v_uint64x2xm4_uint64xm4` (const unsigned long *\*address*, *uint64xm4\_t* *index*, unsigned int *gvl*)
- `uint8x2xm1_t vlsxseg2bu_v_uint8x2xm1_uint8xm1` (const unsigned char *\*address*, *uint8xm1\_t* *index*, unsigned int *gvl*)
- `uint8x2xm2_t vlsxseg2bu_v_uint8x2xm2_uint8xm2` (const unsigned char *\*address*, *uint8xm2\_t* *index*, unsigned int *gvl*)
- `uint8x2xm4_t vlsxseg2bu_v_uint8x2xm4_uint8xm4` (const unsigned char *\*address*, *uint8xm4\_t* *index*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x2xm1_t vlsxseg2bu_mask_uint16x2xm1_uint16xm1` (*uint16x2xm1\_t* *merge*, const unsigned short *\*address*, *uint16xm1\_t* *index*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `uint16x2xm2_t vlsxseg2bu_mask_uint16x2xm2_uint16xm2` (*uint16x2xm2\_t* *merge*, const unsigned short *\*address*, *uint16xm2\_t* *index*, *e16xm2\_t* *mask*, unsigned int *gvl*)



- `uint16x2xm4_t vlxseg2buvmask_uint16x2xm4_uint16xm4` (`uint16x2xm4_t merge, const unsigned short *address, uint16xm4_t index, e16xm4_t mask, unsigned int gvl`)
- `uint32x2xm1_t vlxseg2buvmask_uint32x2xm1_uint32xm1` (`uint32x2xm1_t merge, const unsigned int *address, uint32xm1_t index, e32xm1_t mask, unsigned int gvl`)
- `uint32x2xm2_t vlxseg2buvmask_uint32x2xm2_uint32xm2` (`uint32x2xm2_t merge, const unsigned int *address, uint32xm2_t index, e32xm2_t mask, unsigned int gvl`)
- `uint32x2xm4_t vlxseg2buvmask_uint32x2xm4_uint32xm4` (`uint32x2xm4_t merge, const unsigned int *address, uint32xm4_t index, e32xm4_t mask, unsigned int gvl`)
- `uint64x2xm1_t vlxseg2buvmask_uint64x2xm1_uint64xm1` (`uint64x2xm1_t merge, const unsigned long *address, uint64xm1_t index, e64xm1_t mask, unsigned int gvl`)
- `uint64x2xm2_t vlxseg2buvmask_uint64x2xm2_uint64xm2` (`uint64x2xm2_t merge, const unsigned long *address, uint64xm2_t index, e64xm2_t mask, unsigned int gvl`)
- `uint64x2xm4_t vlxseg2buvmask_uint64x2xm4_uint64xm4` (`uint64x2xm4_t merge, const unsigned long *address, uint64xm4_t index, e64xm4_t mask, unsigned int gvl`)
- `uint8x2xm1_t vlxseg2buvmask_uint8x2xm1_uint8xm1` (`uint8x2xm1_t merge, const unsigned char *address, uint8xm1_t index, e8xm1_t mask, unsigned int gvl`)
- `uint8x2xm2_t vlxseg2buvmask_uint8x2xm2_uint8xm2` (`uint8x2xm2_t merge, const unsigned char *address, uint8xm2_t index, e8xm2_t mask, unsigned int gvl`)
- `uint8x2xm4_t vlxseg2buvmask_uint8x2xm4_uint8xm4` (`uint8x2xm4_t merge, const unsigned char *address, uint8xm4_t index, e8xm4_t mask, unsigned int gvl`)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
      if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
```

(continues on next page)

(continued from previous page)

```

else
    result[segment] = merge[segment]
result[gvl : VLMAX] = 0

```

### 2.10.101 Load 2 contiguous element fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** ['vlxseg2e.v']

**Prototypes:**

- float16x2xm1\_t **vlxseg2ev\_float16x2xm1\_float16xm1** (const float16\_t \*address, float16xm1\_t index, unsigned int gvl)
- float16x2xm2\_t **vlxseg2ev\_float16x2xm2\_float16xm2** (const float16\_t \*address, float16xm2\_t index, unsigned int gvl)
- float16x2xm4\_t **vlxseg2ev\_float16x2xm4\_float16xm4** (const float16\_t \*address, float16xm4\_t index, unsigned int gvl)
- float32x2xm1\_t **vlxseg2ev\_float32x2xm1\_float32xm1** (const float \*address, float32xm1\_t index, unsigned int gvl)
- float32x2xm2\_t **vlxseg2ev\_float32x2xm2\_float32xm2** (const float \*address, float32xm2\_t index, unsigned int gvl)
- float32x2xm4\_t **vlxseg2ev\_float32x2xm4\_float32xm4** (const float \*address, float32xm4\_t index, unsigned int gvl)
- float64x2xm1\_t **vlxseg2ev\_float64x2xm1\_float64xm1** (const double \*address, float64xm1\_t index, unsigned int gvl)
- float64x2xm2\_t **vlxseg2ev\_float64x2xm2\_float64xm2** (const double \*address, float64xm2\_t index, unsigned int gvl)
- float64x2xm4\_t **vlxseg2ev\_float64x2xm4\_float64xm4** (const double \*address, float64xm4\_t index, unsigned int gvl)
- int16x2xm1\_t **vlxseg2ev\_int16x2xm1\_int16xm1** (const short \*address, int16xm1\_t index, unsigned int gvl)
- int16x2xm2\_t **vlxseg2ev\_int16x2xm2\_int16xm2** (const short \*address, int16xm2\_t index, unsigned int gvl)
- int16x2xm4\_t **vlxseg2ev\_int16x2xm4\_int16xm4** (const short \*address, int16xm4\_t index, unsigned int gvl)
- int32x2xm1\_t **vlxseg2ev\_int32x2xm1\_int32xm1** (const int \*address, int32xm1\_t index, unsigned int gvl)
- int32x2xm2\_t **vlxseg2ev\_int32x2xm2\_int32xm2** (const int \*address, int32xm2\_t index, unsigned int gvl)
- int32x2xm4\_t **vlxseg2ev\_int32x2xm4\_int32xm4** (const int \*address, int32xm4\_t index, unsigned int gvl)
- int64x2xm1\_t **vlxseg2ev\_int64x2xm1\_int64xm1** (const long \*address, int64xm1\_t index, unsigned int gvl)
- int64x2xm2\_t **vlxseg2ev\_int64x2xm2\_int64xm2** (const long \*address, int64xm2\_t index, unsigned int gvl)
- int64x2xm4\_t **vlxseg2ev\_int64x2xm4\_int64xm4** (const long \*address, int64xm4\_t index, unsigned int gvl)

- `int8x2xm1_t vlxseg2ev_int8x2xm1_int8xm1` (const signed char *\*address*, *int8xm1\_t* *index*, unsigned int *gvl*)
- `int8x2xm2_t vlxseg2ev_int8x2xm2_int8xm2` (const signed char *\*address*, *int8xm2\_t* *index*, unsigned int *gvl*)
- `int8x2xm4_t vlxseg2ev_int8x2xm4_int8xm4` (const signed char *\*address*, *int8xm4\_t* *index*, unsigned int *gvl*)
- `uint16x2xm1_t vlxseg2ev_uint16x2xm1_uint16xm1` (const unsigned short *\*address*, *uint16xm1\_t* *index*, unsigned int *gvl*)
- `uint16x2xm2_t vlxseg2ev_uint16x2xm2_uint16xm2` (const unsigned short *\*address*, *uint16xm2\_t* *index*, unsigned int *gvl*)
- `uint16x2xm4_t vlxseg2ev_uint16x2xm4_uint16xm4` (const unsigned short *\*address*, *uint16xm4\_t* *index*, unsigned int *gvl*)
- `uint32x2xm1_t vlxseg2ev_uint32x2xm1_uint32xm1` (const unsigned int *\*address*, *uint32xm1\_t* *index*, unsigned int *gvl*)
- `uint32x2xm2_t vlxseg2ev_uint32x2xm2_uint32xm2` (const unsigned int *\*address*, *uint32xm2\_t* *index*, unsigned int *gvl*)
- `uint32x2xm4_t vlxseg2ev_uint32x2xm4_uint32xm4` (const unsigned int *\*address*, *uint32xm4\_t* *index*, unsigned int *gvl*)
- `uint64x2xm1_t vlxseg2ev_uint64x2xm1_uint64xm1` (const unsigned long *\*address*, *uint64xm1\_t* *index*, unsigned int *gvl*)
- `uint64x2xm2_t vlxseg2ev_uint64x2xm2_uint64xm2` (const unsigned long *\*address*, *uint64xm2\_t* *index*, unsigned int *gvl*)
- `uint64x2xm4_t vlxseg2ev_uint64x2xm4_uint64xm4` (const unsigned long *\*address*, *uint64xm4\_t* *index*, unsigned int *gvl*)
- `uint8x2xm1_t vlxseg2ev_uint8x2xm1_uint8xm1` (const unsigned char *\*address*, *uint8xm1\_t* *index*, unsigned int *gvl*)
- `uint8x2xm2_t vlxseg2ev_uint8x2xm2_uint8xm2` (const unsigned char *\*address*, *uint8xm2\_t* *index*, unsigned int *gvl*)
- `uint8x2xm4_t vlxseg2ev_uint8x2xm4_uint8xm4` (const unsigned char *\*address*, *uint8xm4\_t* *index*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x2xm1_t vlxseg2ev_mask_float16x2xm1_float16xm1` (float16x2xm1\_t *merge*, const float16\_t *\*address*, *float16xm1\_t* *index*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `float16x2xm2_t vlxseg2ev_mask_float16x2xm2_float16xm2` (float16x2xm2\_t *merge*, const float16\_t *\*address*, *float16xm2\_t* *index*, *e16xm2\_t* *mask*, unsigned int *gvl*)

- `float16x2xm4_t vlxseg2ev_mask_float16x2xm4_float16xm4` (`float16x2xm4_t` *merge*,  
const `float16_t` *\*address*, `float16xm4_t` *index*,  
`e16xm4_t` *mask*, unsigned  
int gvl)
- `float32x2xm1_t vlxseg2ev_mask_float32x2xm1_float32xm1` (`float32x2xm1_t` *merge*,  
const `float` *\*address*,  
`float32xm1_t` *index*,  
`e32xm1_t` *mask*, unsigned  
int gvl)
- `float32x2xm2_t vlxseg2ev_mask_float32x2xm2_float32xm2` (`float32x2xm2_t` *merge*,  
const `float` *\*address*,  
`float32xm2_t` *index*,  
`e32xm2_t` *mask*, unsigned  
int gvl)
- `float32x2xm4_t vlxseg2ev_mask_float32x2xm4_float32xm4` (`float32x2xm4_t` *merge*,  
const `float` *\*address*,  
`float32xm4_t` *index*,  
`e32xm4_t` *mask*, unsigned  
int gvl)
- `float64x2xm1_t vlxseg2ev_mask_float64x2xm1_float64xm1` (`float64x2xm1_t` *merge*,  
const `double` *\*address*,  
`float64xm1_t` *index*,  
`e64xm1_t` *mask*, unsigned  
int gvl)
- `float64x2xm2_t vlxseg2ev_mask_float64x2xm2_float64xm2` (`float64x2xm2_t` *merge*,  
const `double` *\*address*,  
`float64xm2_t` *index*,  
`e64xm2_t` *mask*, unsigned  
int gvl)
- `float64x2xm4_t vlxseg2ev_mask_float64x2xm4_float64xm4` (`float64x2xm4_t` *merge*,  
const `double` *\*address*,  
`float64xm4_t` *index*,  
`e64xm4_t` *mask*, unsigned  
int gvl)
- `int16x2xm1_t vlxseg2ev_mask_int16x2xm1_int16xm1` (`int16x2xm1_t` *merge*, const  
short *\*address*, `int16xm1_t` *index*,  
`e16xm1_t` *mask*, unsigned int gvl)
- `int16x2xm2_t vlxseg2ev_mask_int16x2xm2_int16xm2` (`int16x2xm2_t` *merge*, const  
short *\*address*, `int16xm2_t` *index*,  
`e16xm2_t` *mask*, unsigned int gvl)
- `int16x2xm4_t vlxseg2ev_mask_int16x2xm4_int16xm4` (`int16x2xm4_t` *merge*, const  
short *\*address*, `int16xm4_t` *index*,  
`e16xm4_t` *mask*, unsigned int gvl)
- `int32x2xm1_t vlxseg2ev_mask_int32x2xm1_int32xm1` (`int32x2xm1_t` *merge*, const  
int *\*address*, `int32xm1_t` *index*,  
`e32xm1_t` *mask*, unsigned int gvl)
- `int32x2xm2_t vlxseg2ev_mask_int32x2xm2_int32xm2` (`int32x2xm2_t` *merge*, const  
int *\*address*, `int32xm2_t` *index*,  
`e32xm2_t` *mask*, unsigned int gvl)

- `int32x2xm4_t vlxseg2ev_mask_int32x2xm4_int32xm4` (`int32x2xm4_t merge, const int *address, int32xm4_t index, e32xm4_t mask, unsigned int gvl`)
- `int64x2xm1_t vlxseg2ev_mask_int64x2xm1_int64xm1` (`int64x2xm1_t merge, const long *address, int64xm1_t index, e64xm1_t mask, unsigned int gvl`)
- `int64x2xm2_t vlxseg2ev_mask_int64x2xm2_int64xm2` (`int64x2xm2_t merge, const long *address, int64xm2_t index, e64xm2_t mask, unsigned int gvl`)
- `int64x2xm4_t vlxseg2ev_mask_int64x2xm4_int64xm4` (`int64x2xm4_t merge, const long *address, int64xm4_t index, e64xm4_t mask, unsigned int gvl`)
- `int8x2xm1_t vlxseg2ev_mask_int8x2xm1_int8xm1` (`int8x2xm1_t merge, const signed char *address, int8xm1_t index, e8xm1_t mask, unsigned int gvl`)
- `int8x2xm2_t vlxseg2ev_mask_int8x2xm2_int8xm2` (`int8x2xm2_t merge, const signed char *address, int8xm2_t index, e8xm2_t mask, unsigned int gvl`)
- `int8x2xm4_t vlxseg2ev_mask_int8x2xm4_int8xm4` (`int8x2xm4_t merge, const signed char *address, int8xm4_t index, e8xm4_t mask, unsigned int gvl`)
- `uint16x2xm1_t vlxseg2ev_mask_uint16x2xm1_uint16xm1` (`uint16x2xm1_t merge, const unsigned short *address, uint16xm1_t index, e16xm1_t mask, unsigned int gvl`)
- `uint16x2xm2_t vlxseg2ev_mask_uint16x2xm2_uint16xm2` (`uint16x2xm2_t merge, const unsigned short *address, uint16xm2_t index, e16xm2_t mask, unsigned int gvl`)
- `uint16x2xm4_t vlxseg2ev_mask_uint16x2xm4_uint16xm4` (`uint16x2xm4_t merge, const unsigned short *address, uint16xm4_t index, e16xm4_t mask, unsigned int gvl`)
- `uint32x2xm1_t vlxseg2ev_mask_uint32x2xm1_uint32xm1` (`uint32x2xm1_t merge, const unsigned int *address, uint32xm1_t index, e32xm1_t mask, unsigned int gvl`)
- `uint32x2xm2_t vlxseg2ev_mask_uint32x2xm2_uint32xm2` (`uint32x2xm2_t merge, const unsigned int *address, uint32xm2_t index, e32xm2_t mask, unsigned int gvl`)
- `uint32x2xm4_t vlxseg2ev_mask_uint32x2xm4_uint32xm4` (`uint32x2xm4_t merge, const unsigned int *address, uint32xm4_t index, e32xm4_t mask, unsigned int gvl`)
- `uint64x2xm1_t vlxseg2ev_mask_uint64x2xm1_uint64xm1` (`uint64x2xm1_t merge, const unsigned long *address, uint64xm1_t index, e64xm1_t mask, unsigned int gvl`)

- `uint64x2xm2_t vlxseg2ev_mask_uint64x2xm2_uint64xm2` (`uint64x2xm2_t` *merge*, `const unsigned long *address`, `uint64xm2_t` *index*, `e64xm2_t` *mask*, `unsigned int gvl`)
- `uint64x2xm4_t vlxseg2ev_mask_uint64x2xm4_uint64xm4` (`uint64x2xm4_t` *merge*, `const unsigned long *address`, `uint64xm4_t` *index*, `e64xm4_t` *mask*, `unsigned int gvl`)
- `uint8x2xm1_t vlxseg2ev_mask_uint8x2xm1_uint8xm1` (`uint8x2xm1_t` *merge*, `const unsigned char *address`, `uint8xm1_t` *index*, `e8xm1_t` *mask*, `unsigned int gvl`)
- `uint8x2xm2_t vlxseg2ev_mask_uint8x2xm2_uint8xm2` (`uint8x2xm2_t` *merge*, `const unsigned char *address`, `uint8xm2_t` *index*, `e8xm2_t` *mask*, `unsigned int gvl`)
- `uint8x2xm4_t vlxseg2ev_mask_uint8x2xm4_uint8xm4` (`uint8x2xm4_t` *merge*, `const unsigned char *address`, `uint8xm4_t` *index*, `e8xm4_t` *mask*, `unsigned int gvl`)

#### Masked operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.102 Load 2 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlxseg2h.v'`]

#### Prototypes:

- `int16x2xm1_t vlxseg2hv_int16x2xm1_int16xm1` (`const short *address`, `int16xm1_t` *index*, `unsigned int gvl`)
- `int16x2xm2_t vlxseg2hv_int16x2xm2_int16xm2` (`const short *address`, `int16xm2_t` *index*, `unsigned int gvl`)
- `int16x2xm4_t vlxseg2hv_int16x2xm4_int16xm4` (`const short *address`, `int16xm4_t` *index*, `unsigned int gvl`)
- `int32x2xm1_t vlxseg2hv_int32x2xm1_int32xm1` (`const int *address`, `int32xm1_t` *index*, `unsigned int gvl`)
- `int32x2xm2_t vlxseg2hv_int32x2xm2_int32xm2` (`const int *address`, `int32xm2_t` *index*, `unsigned int gvl`)
- `int32x2xm4_t vlxseg2hv_int32x2xm4_int32xm4` (`const int *address`, `int32xm4_t` *index*, `unsigned int gvl`)
- `int64x2xm1_t vlxseg2hv_int64x2xm1_int64xm1` (`const long *address`, `int64xm1_t` *index*, `unsigned int gvl`)
- `int64x2xm2_t vlxseg2hv_int64x2xm2_int64xm2` (`const long *address`, `int64xm2_t` *index*, `unsigned int gvl`)



- `int64x2xm4_t vlxseg2hv_int64x2xm4_int64xm4` (const long \*address, *int64xm4\_t* index, unsigned int gvl)
- `int8x2xm1_t vlxseg2hv_int8x2xm1_int8xm1` (const signed char \*address, *int8xm1\_t* index, unsigned int gvl)
- `int8x2xm2_t vlxseg2hv_int8x2xm2_int8xm2` (const signed char \*address, *int8xm2\_t* index, unsigned int gvl)
- `int8x2xm4_t vlxseg2hv_int8x2xm4_int8xm4` (const signed char \*address, *int8xm4\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x2xm1_t vlxseg2hv_mask_int16x2xm1_int16xm1` (int16x2xm1\_t merge, const short \*address, *int16xm1\_t* index, *e16xm1\_t* mask, unsigned int gvl)
- `int16x2xm2_t vlxseg2hv_mask_int16x2xm2_int16xm2` (int16x2xm2\_t merge, const short \*address, *int16xm2\_t* index, *e16xm2\_t* mask, unsigned int gvl)
- `int16x2xm4_t vlxseg2hv_mask_int16x2xm4_int16xm4` (int16x2xm4\_t merge, const short \*address, *int16xm4\_t* index, *e16xm4\_t* mask, unsigned int gvl)
- `int32x2xm1_t vlxseg2hv_mask_int32x2xm1_int32xm1` (int32x2xm1\_t merge, const int \*address, *int32xm1\_t* index, *e32xm1\_t* mask, unsigned int gvl)
- `int32x2xm2_t vlxseg2hv_mask_int32x2xm2_int32xm2` (int32x2xm2\_t merge, const int \*address, *int32xm2\_t* index, *e32xm2\_t* mask, unsigned int gvl)
- `int32x2xm4_t vlxseg2hv_mask_int32x2xm4_int32xm4` (int32x2xm4\_t merge, const int \*address, *int32xm4\_t* index, *e32xm4\_t* mask, unsigned int gvl)
- `int64x2xm1_t vlxseg2hv_mask_int64x2xm1_int64xm1` (int64x2xm1\_t merge, const long \*address, *int64xm1\_t* index, *e64xm1\_t* mask, unsigned int gvl)
- `int64x2xm2_t vlxseg2hv_mask_int64x2xm2_int64xm2` (int64x2xm2\_t merge, const long \*address, *int64xm2\_t* index, *e64xm2\_t* mask, unsigned int gvl)
- `int64x2xm4_t vlxseg2hv_mask_int64x2xm4_int64xm4` (int64x2xm4\_t merge, const long \*address, *int64xm4\_t* index, *e64xm4\_t* mask, unsigned int gvl)
- `int8x2xm1_t vlxseg2hv_mask_int8x2xm1_int8xm1` (int8x2xm1\_t merge, const signed char \*address, *int8xm1\_t* index, *e8xm1\_t* mask, unsigned int gvl)
- `int8x2xm2_t vlxseg2hv_mask_int8x2xm2_int8xm2` (int8x2xm2\_t merge, const signed char \*address, *int8xm2\_t* index, *e8xm2\_t* mask, unsigned int gvl)

- `int8x2xm4_t vlxseg2hv_mask_int8x2xm4_int8xm4` (`int8x2xm4_t merge`, `const signed char *address`, `int8xm4_t index`, `e8xm4_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.103 Load 2 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg2hu.v`]

**Prototypes:**

- `uint16x2xm1_t vlxseg2huv_uint16x2xm1_uint16xm1` (`const unsigned short *address`, `uint16xm1_t index`, `unsigned int gvl`)
- `uint16x2xm2_t vlxseg2huv_uint16x2xm2_uint16xm2` (`const unsigned short *address`, `uint16xm2_t index`, `unsigned int gvl`)
- `uint16x2xm4_t vlxseg2huv_uint16x2xm4_uint16xm4` (`const unsigned short *address`, `uint16xm4_t index`, `unsigned int gvl`)
- `uint32x2xm1_t vlxseg2huv_uint32x2xm1_uint32xm1` (`const unsigned int *address`, `uint32xm1_t index`, `unsigned int gvl`)
- `uint32x2xm2_t vlxseg2huv_uint32x2xm2_uint32xm2` (`const unsigned int *address`, `uint32xm2_t index`, `unsigned int gvl`)
- `uint32x2xm4_t vlxseg2huv_uint32x2xm4_uint32xm4` (`const unsigned int *address`, `uint32xm4_t index`, `unsigned int gvl`)
- `uint64x2xm1_t vlxseg2huv_uint64x2xm1_uint64xm1` (`const unsigned long *address`, `uint64xm1_t index`, `unsigned int gvl`)
- `uint64x2xm2_t vlxseg2huv_uint64x2xm2_uint64xm2` (`const unsigned long *address`, `uint64xm2_t index`, `unsigned int gvl`)
- `uint64x2xm4_t vlxseg2huv_uint64x2xm4_uint64xm4` (`const unsigned long *address`, `uint64xm4_t index`, `unsigned int gvl`)
- `uint8x2xm1_t vlxseg2huv_uint8x2xm1_uint8xm1` (`const unsigned char *address`, `uint8xm1_t index`, `unsigned int gvl`)
- `uint8x2xm2_t vlxseg2huv_uint8x2xm2_uint8xm2` (`const unsigned char *address`, `uint8xm2_t index`, `unsigned int gvl`)
- `uint8x2xm4_t vlxseg2huv_uint8x2xm4_uint8xm4` (`const unsigned char *address`, `uint8xm4_t index`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x2xm1_t vlxseg2huv_mask_uint16x2xm1_uint16xm1` (`uint16x2xm1_t` *merge*,  
const unsigned short *\*address*, `uint16xm1_t` *index*,  
`e16xm1_t` *mask*, unsigned  
int *gvl*)
- `uint16x2xm2_t vlxseg2huv_mask_uint16x2xm2_uint16xm2` (`uint16x2xm2_t` *merge*,  
const unsigned short *\*address*, `uint16xm2_t` *index*,  
`e16xm2_t` *mask*, unsigned  
int *gvl*)
- `uint16x2xm4_t vlxseg2huv_mask_uint16x2xm4_uint16xm4` (`uint16x2xm4_t` *merge*,  
const unsigned short *\*address*, `uint16xm4_t` *index*,  
`e16xm4_t` *mask*, unsigned  
int *gvl*)
- `uint32x2xm1_t vlxseg2huv_mask_uint32x2xm1_uint32xm1` (`uint32x2xm1_t` *merge*,  
const unsigned int *\*address*, `uint32xm1_t` *index*,  
`e32xm1_t` *mask*, unsigned  
int *gvl*)
- `uint32x2xm2_t vlxseg2huv_mask_uint32x2xm2_uint32xm2` (`uint32x2xm2_t` *merge*,  
const unsigned int *\*address*, `uint32xm2_t` *index*,  
`e32xm2_t` *mask*, unsigned  
int *gvl*)
- `uint32x2xm4_t vlxseg2huv_mask_uint32x2xm4_uint32xm4` (`uint32x2xm4_t` *merge*,  
const unsigned int *\*address*, `uint32xm4_t` *index*,  
`e32xm4_t` *mask*, unsigned  
int *gvl*)
- `uint64x2xm1_t vlxseg2huv_mask_uint64x2xm1_uint64xm1` (`uint64x2xm1_t` *merge*,  
const unsigned long *\*address*, `uint64xm1_t` *index*,  
`e64xm1_t` *mask*, unsigned  
int *gvl*)
- `uint64x2xm2_t vlxseg2huv_mask_uint64x2xm2_uint64xm2` (`uint64x2xm2_t` *merge*,  
const unsigned long *\*address*, `uint64xm2_t` *index*,  
`e64xm2_t` *mask*, unsigned  
int *gvl*)
- `uint64x2xm4_t vlxseg2huv_mask_uint64x2xm4_uint64xm4` (`uint64x2xm4_t` *merge*,  
const unsigned long *\*address*, `uint64xm4_t` *index*,  
`e64xm4_t` *mask*, unsigned  
int *gvl*)
- `uint8x2xm1_t vlxseg2huv_mask_uint8x2xm1_uint8xm1` (`uint8x2xm1_t` *merge*, const unsigned  
char *\*address*, `uint8xm1_t` *index*,  
`e8xm1_t` *mask*, unsigned int *gvl*)

- `uint8x2xm2_t vlxseg2huv_mask_uint8x2xm2_uint8xm2` (`uint8x2xm2_t merge`, const unsigned char *\*address*, `uint8xm2_t index`, `e8xm2_t mask`, unsigned int *gvl*)
- `uint8x2xm4_t vlxseg2huv_mask_uint8x2xm4_uint8xm4` (`uint8x2xm4_t merge`, const unsigned char *\*address*, `uint8xm4_t index`, `e8xm4_t mask`, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.104 Load 2 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg2w.v`']

**Prototypes:**

- `int16x2xm1_t vlxseg2wv_int16x2xm1_int16xm1` (const short *\*address*, `int16xm1_t index`, unsigned int *gvl*)
- `int16x2xm2_t vlxseg2wv_int16x2xm2_int16xm2` (const short *\*address*, `int16xm2_t index`, unsigned int *gvl*)
- `int16x2xm4_t vlxseg2wv_int16x2xm4_int16xm4` (const short *\*address*, `int16xm4_t index`, unsigned int *gvl*)
- `int32x2xm1_t vlxseg2wv_int32x2xm1_int32xm1` (const int *\*address*, `int32xm1_t index`, unsigned int *gvl*)
- `int32x2xm2_t vlxseg2wv_int32x2xm2_int32xm2` (const int *\*address*, `int32xm2_t index`, unsigned int *gvl*)
- `int32x2xm4_t vlxseg2wv_int32x2xm4_int32xm4` (const int *\*address*, `int32xm4_t index`, unsigned int *gvl*)
- `int64x2xm1_t vlxseg2wv_int64x2xm1_int64xm1` (const long *\*address*, `int64xm1_t index`, unsigned int *gvl*)
- `int64x2xm2_t vlxseg2wv_int64x2xm2_int64xm2` (const long *\*address*, `int64xm2_t index`, unsigned int *gvl*)
- `int64x2xm4_t vlxseg2wv_int64x2xm4_int64xm4` (const long *\*address*, `int64xm4_t index`, unsigned int *gvl*)
- `int8x2xm1_t vlxseg2wv_int8x2xm1_int8xm1` (const signed char *\*address*, `int8xm1_t index`, unsigned int *gvl*)
- `int8x2xm2_t vlxseg2wv_int8x2xm2_int8xm2` (const signed char *\*address*, `int8xm2_t index`, unsigned int *gvl*)
- `int8x2xm4_t vlxseg2wv_int8x2xm4_int8xm4` (const signed char *\*address*, `int8xm4_t index`, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

### Masked prototypes:

- `int16x2xm1_t vlxseg2wv_mask_int16x2xm1_int16xm1` (`int16x2xm1_t merge`, `const short *address`, `int16xm1_t index`, `e16xm1_t mask`, unsigned int `gvl`)
- `int16x2xm2_t vlxseg2wv_mask_int16x2xm2_int16xm2` (`int16x2xm2_t merge`, `const short *address`, `int16xm2_t index`, `e16xm2_t mask`, unsigned int `gvl`)
- `int16x2xm4_t vlxseg2wv_mask_int16x2xm4_int16xm4` (`int16x2xm4_t merge`, `const short *address`, `int16xm4_t index`, `e16xm4_t mask`, unsigned int `gvl`)
- `int32x2xm1_t vlxseg2wv_mask_int32x2xm1_int32xm1` (`int32x2xm1_t merge`, `const int *address`, `int32xm1_t index`, `e32xm1_t mask`, unsigned int `gvl`)
- `int32x2xm2_t vlxseg2wv_mask_int32x2xm2_int32xm2` (`int32x2xm2_t merge`, `const int *address`, `int32xm2_t index`, `e32xm2_t mask`, unsigned int `gvl`)
- `int32x2xm4_t vlxseg2wv_mask_int32x2xm4_int32xm4` (`int32x2xm4_t merge`, `const int *address`, `int32xm4_t index`, `e32xm4_t mask`, unsigned int `gvl`)
- `int64x2xm1_t vlxseg2wv_mask_int64x2xm1_int64xm1` (`int64x2xm1_t merge`, `const long *address`, `int64xm1_t index`, `e64xm1_t mask`, unsigned int `gvl`)
- `int64x2xm2_t vlxseg2wv_mask_int64x2xm2_int64xm2` (`int64x2xm2_t merge`, `const long *address`, `int64xm2_t index`, `e64xm2_t mask`, unsigned int `gvl`)
- `int64x2xm4_t vlxseg2wv_mask_int64x2xm4_int64xm4` (`int64x2xm4_t merge`, `const long *address`, `int64xm4_t index`, `e64xm4_t mask`, unsigned int `gvl`)
- `int8x2xm1_t vlxseg2wv_mask_int8x2xm1_int8xm1` (`int8x2xm1_t merge`, `const signed char *address`, `int8xm1_t index`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8x2xm2_t vlxseg2wv_mask_int8x2xm2_int8xm2` (`int8x2xm2_t merge`, `const signed char *address`, `int8xm2_t index`, `e8xm2_t mask`, unsigned int `gvl`)
- `int8x2xm4_t vlxseg2wv_mask_int8x2xm4_int8xm4` (`int8x2xm4_t merge`, `const signed char *address`, `int8xm4_t index`, `e8xm4_t mask`, unsigned int `gvl`)

### Masked operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
```

(continues on next page)

(continued from previous page)

```
result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.105 Load 2 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlxseg2wuv.v'`]

**Prototypes:**

- `uint16x2xm1_t vlxseg2wuv_uint16x2xm1_uint16xm1` (const unsigned short \*address, *uint16xm1\_t* index, unsigned int gvl)
- `uint16x2xm2_t vlxseg2wuv_uint16x2xm2_uint16xm2` (const unsigned short \*address, *uint16xm2\_t* index, unsigned int gvl)
- `uint16x2xm4_t vlxseg2wuv_uint16x2xm4_uint16xm4` (const unsigned short \*address, *uint16xm4\_t* index, unsigned int gvl)
- `uint32x2xm1_t vlxseg2wuv_uint32x2xm1_uint32xm1` (const unsigned int \*address, *uint32xm1\_t* index, unsigned int gvl)
- `uint32x2xm2_t vlxseg2wuv_uint32x2xm2_uint32xm2` (const unsigned int \*address, *uint32xm2\_t* index, unsigned int gvl)
- `uint32x2xm4_t vlxseg2wuv_uint32x2xm4_uint32xm4` (const unsigned int \*address, *uint32xm4\_t* index, unsigned int gvl)
- `uint64x2xm1_t vlxseg2wuv_uint64x2xm1_uint64xm1` (const unsigned long \*address, *uint64xm1\_t* index, unsigned int gvl)
- `uint64x2xm2_t vlxseg2wuv_uint64x2xm2_uint64xm2` (const unsigned long \*address, *uint64xm2\_t* index, unsigned int gvl)
- `uint64x2xm4_t vlxseg2wuv_uint64x2xm4_uint64xm4` (const unsigned long \*address, *uint64xm4\_t* index, unsigned int gvl)
- `uint8x2xm1_t vlxseg2wuv_uint8x2xm1_uint8xm1` (const unsigned char \*address, *uint8xm1\_t* index, unsigned int gvl)
- `uint8x2xm2_t vlxseg2wuv_uint8x2xm2_uint8xm2` (const unsigned char \*address, *uint8xm2\_t* index, unsigned int gvl)
- `uint8x2xm4_t vlxseg2wuv_uint8x2xm4_uint8xm4` (const unsigned char \*address, *uint8xm4\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x2xm1_t vlxseg2wuv_mask_uint16x2xm1_uint16xm1` (uint16x2xm1\_t merge, const unsigned short \*address, *uint16xm1\_t* index, *el16xm1\_t* mask, unsigned int gvl)



- `uint16x2xm2_t vlxseg2wuv_mask_uint16x2xm2_uint16xm2` (`uint16x2xm2_t` *merge*,  
const unsigned short *\*ad-*  
*dress*, `uint16xm2_t` *index*,  
`e16xm2_t` *mask*, unsigned  
int *gvl*)
- `uint16x2xm4_t vlxseg2wuv_mask_uint16x2xm4_uint16xm4` (`uint16x2xm4_t` *merge*,  
const unsigned short *\*ad-*  
*dress*, `uint16xm4_t` *index*,  
`e16xm4_t` *mask*, unsigned  
int *gvl*)
- `uint32x2xm1_t vlxseg2wuv_mask_uint32x2xm1_uint32xm1` (`uint32x2xm1_t` *merge*,  
const unsigned int *\*ad-*  
*dress*, `uint32xm1_t` *index*,  
`e32xm1_t` *mask*, unsigned  
int *gvl*)
- `uint32x2xm2_t vlxseg2wuv_mask_uint32x2xm2_uint32xm2` (`uint32x2xm2_t` *merge*,  
const unsigned int *\*ad-*  
*dress*, `uint32xm2_t` *index*,  
`e32xm2_t` *mask*, unsigned  
int *gvl*)
- `uint32x2xm4_t vlxseg2wuv_mask_uint32x2xm4_uint32xm4` (`uint32x2xm4_t` *merge*,  
const unsigned int *\*ad-*  
*dress*, `uint32xm4_t` *index*,  
`e32xm4_t` *mask*, unsigned  
int *gvl*)
- `uint64x2xm1_t vlxseg2wuv_mask_uint64x2xm1_uint64xm1` (`uint64x2xm1_t` *merge*,  
const unsigned long *\*ad-*  
*dress*, `uint64xm1_t` *index*,  
`e64xm1_t` *mask*, unsigned  
int *gvl*)
- `uint64x2xm2_t vlxseg2wuv_mask_uint64x2xm2_uint64xm2` (`uint64x2xm2_t` *merge*,  
const unsigned long *\*ad-*  
*dress*, `uint64xm2_t` *index*,  
`e64xm2_t` *mask*, unsigned  
int *gvl*)
- `uint64x2xm4_t vlxseg2wuv_mask_uint64x2xm4_uint64xm4` (`uint64x2xm4_t` *merge*,  
const unsigned long *\*ad-*  
*dress*, `uint64xm4_t` *index*,  
`e64xm4_t` *mask*, unsigned  
int *gvl*)
- `uint8x2xm1_t vlxseg2wuv_mask_uint8x2xm1_uint8xm1` (`uint8x2xm1_t` *merge*, const unsigned  
char *\*address*, `uint8xm1_t` *index*,  
`e8xm1_t` *mask*, unsigned int *gvl*)
- `uint8x2xm2_t vlxseg2wuv_mask_uint8x2xm2_uint8xm2` (`uint8x2xm2_t` *merge*, const unsigned  
char *\*address*, `uint8xm2_t` *index*,  
`e8xm2_t` *mask*, unsigned int *gvl*)
- `uint8x2xm4_t vlxseg2wuv_mask_uint8x2xm4_uint8xm4` (`uint8x2xm4_t` *merge*, const unsigned  
char *\*address*, `uint8xm4_t` *index*,  
`e8xm4_t` *mask*, unsigned int *gvl*)

Masked operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.106 Load 3 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** ['vlxseg3b.v']

**Prototypes:**

- `int16x3xm1_t vlxseg3bv_int16x3xm1_int16xm1` (const short \*address, *int16xm1\_t* index, unsigned int gvl)
- `int16x3xm2_t vlxseg3bv_int16x3xm2_int16xm2` (const short \*address, *int16xm2\_t* index, unsigned int gvl)
- `int32x3xm1_t vlxseg3bv_int32x3xm1_int32xm1` (const int \*address, *int32xm1\_t* index, unsigned int gvl)
- `int32x3xm2_t vlxseg3bv_int32x3xm2_int32xm2` (const int \*address, *int32xm2\_t* index, unsigned int gvl)
- `int64x3xm1_t vlxseg3bv_int64x3xm1_int64xm1` (const long \*address, *int64xm1\_t* index, unsigned int gvl)
- `int64x3xm2_t vlxseg3bv_int64x3xm2_int64xm2` (const long \*address, *int64xm2\_t* index, unsigned int gvl)
- `int8x3xm1_t vlxseg3bv_int8x3xm1_int8xm1` (const signed char \*address, *int8xm1\_t* index, unsigned int gvl)
- `int8x3xm2_t vlxseg3bv_int8x3xm2_int8xm2` (const signed char \*address, *int8xm2\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x3xm1_t vlxseg3bv_mask_int16x3xm1_int16xm1` (int16x3xm1\_t merge, const short \*address, *int16xm1\_t* index, *e16xm1\_t* mask, unsigned int gvl)
- `int16x3xm2_t vlxseg3bv_mask_int16x3xm2_int16xm2` (int16x3xm2\_t merge, const short \*address, *int16xm2\_t* index, *e16xm2\_t* mask, unsigned int gvl)
- `int32x3xm1_t vlxseg3bv_mask_int32x3xm1_int32xm1` (int32x3xm1\_t merge, const int \*address, *int32xm1\_t* index, *e32xm1\_t* mask, unsigned int gvl)

- `int32x3xm2_t vlxseg3bv_mask_int32x3xm2_int32xm2` (`int32x3xm2_t merge`, `const int *address`, `int32xm2_t index`, `e32xm2_t mask`, `unsigned int gvl`)
- `int64x3xm1_t vlxseg3bv_mask_int64x3xm1_int64xm1` (`int64x3xm1_t merge`, `const long *address`, `int64xm1_t index`, `e64xm1_t mask`, `unsigned int gvl`)
- `int64x3xm2_t vlxseg3bv_mask_int64x3xm2_int64xm2` (`int64x3xm2_t merge`, `const long *address`, `int64xm2_t index`, `e64xm2_t mask`, `unsigned int gvl`)
- `int8x3xm1_t vlxseg3bv_mask_int8x3xm1_int8xm1` (`int8x3xm1_t merge`, `const signed char *address`, `int8xm1_t index`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8x3xm2_t vlxseg3bv_mask_int8x3xm2_int8xm2` (`int8x3xm2_t merge`, `const signed char *address`, `int8xm2_t index`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.107 Load 3 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg3bu.v`]

**Prototypes:**

- `uint16x3xm1_t vlxseg3bu_v_uint16x3xm1_uint16xm1` (`const unsigned short *address`, `uint16xm1_t index`, `unsigned int gvl`)
- `uint16x3xm2_t vlxseg3bu_v_uint16x3xm2_uint16xm2` (`const unsigned short *address`, `uint16xm2_t index`, `unsigned int gvl`)
- `uint32x3xm1_t vlxseg3bu_v_uint32x3xm1_uint32xm1` (`const unsigned int *address`, `uint32xm1_t index`, `unsigned int gvl`)
- `uint32x3xm2_t vlxseg3bu_v_uint32x3xm2_uint32xm2` (`const unsigned int *address`, `uint32xm2_t index`, `unsigned int gvl`)
- `uint64x3xm1_t vlxseg3bu_v_uint64x3xm1_uint64xm1` (`const unsigned long *address`, `uint64xm1_t index`, `unsigned int gvl`)
- `uint64x3xm2_t vlxseg3bu_v_uint64x3xm2_uint64xm2` (`const unsigned long *address`, `uint64xm2_t index`, `unsigned int gvl`)
- `uint8x3xm1_t vlxseg3bu_v_uint8x3xm1_uint8xm1` (`const unsigned char *address`, `uint8xm1_t index`, `unsigned int gvl`)
- `uint8x3xm2_t vlxseg3bu_v_uint8x3xm2_uint8xm2` (`const unsigned char *address`, `uint8xm2_t index`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- `uint16x3xm1_t vlxseg3buv_mask_uint16x3xm1_uint16xm1` (`uint16x3xm1_t` *merge*,  
const unsigned short *\*address*, `uint16xm1_t` *index*,  
`e16xm1_t` *mask*, unsigned int *gvl*)
- `uint16x3xm2_t vlxseg3buv_mask_uint16x3xm2_uint16xm2` (`uint16x3xm2_t` *merge*,  
const unsigned short *\*address*, `uint16xm2_t` *index*,  
`e16xm2_t` *mask*, unsigned int *gvl*)
- `uint32x3xm1_t vlxseg3buv_mask_uint32x3xm1_uint32xm1` (`uint32x3xm1_t` *merge*,  
const unsigned int *\*address*, `uint32xm1_t` *index*,  
`e32xm1_t` *mask*, unsigned int *gvl*)
- `uint32x3xm2_t vlxseg3buv_mask_uint32x3xm2_uint32xm2` (`uint32x3xm2_t` *merge*,  
const unsigned int *\*address*, `uint32xm2_t` *index*,  
`e32xm2_t` *mask*, unsigned int *gvl*)
- `uint64x3xm1_t vlxseg3buv_mask_uint64x3xm1_uint64xm1` (`uint64x3xm1_t` *merge*,  
const unsigned long *\*address*, `uint64xm1_t` *index*,  
`e64xm1_t` *mask*, unsigned int *gvl*)
- `uint64x3xm2_t vlxseg3buv_mask_uint64x3xm2_uint64xm2` (`uint64x3xm2_t` *merge*,  
const unsigned long *\*address*, `uint64xm2_t` *index*,  
`e64xm2_t` *mask*, unsigned int *gvl*)
- `uint8x3xm1_t vlxseg3buv_mask_uint8x3xm1_uint8xm1` (`uint8x3xm1_t` *merge*, const unsigned  
char *\*address*, `uint8xm1_t` *index*,  
`e8xm1_t` *mask*, unsigned int *gvl*)
- `uint8x3xm2_t vlxseg3buv_mask_uint8x3xm2_uint8xm2` (`uint8x3xm2_t` *merge*, const unsigned  
char *\*address*, `uint8xm2_t` *index*,  
`e8xm2_t` *mask*, unsigned int *gvl*)

#### Masked operation:

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.108 Load 3 contiguous element fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlsxseg3e.v'`]

**Prototypes:**

- `float16x3xm1_t vlsxseg3ev_float16x3xm1_float16xm1` (const float16\_t \*address, float16xm1\_t index, unsigned int gvl)
- `float16x3xm2_t vlsxseg3ev_float16x3xm2_float16xm2` (const float16\_t \*address, float16xm2\_t index, unsigned int gvl)
- `float32x3xm1_t vlsxseg3ev_float32x3xm1_float32xm1` (const float \*address, float32xm1\_t index, unsigned int gvl)
- `float32x3xm2_t vlsxseg3ev_float32x3xm2_float32xm2` (const float \*address, float32xm2\_t index, unsigned int gvl)
- `float64x3xm1_t vlsxseg3ev_float64x3xm1_float64xm1` (const double \*address, float64xm1\_t index, unsigned int gvl)
- `float64x3xm2_t vlsxseg3ev_float64x3xm2_float64xm2` (const double \*address, float64xm2\_t index, unsigned int gvl)
- `int16x3xm1_t vlsxseg3ev_int16x3xm1_int16xm1` (const short \*address, int16xm1\_t index, unsigned int gvl)
- `int16x3xm2_t vlsxseg3ev_int16x3xm2_int16xm2` (const short \*address, int16xm2\_t index, unsigned int gvl)
- `int32x3xm1_t vlsxseg3ev_int32x3xm1_int32xm1` (const int \*address, int32xm1\_t index, unsigned int gvl)
- `int32x3xm2_t vlsxseg3ev_int32x3xm2_int32xm2` (const int \*address, int32xm2\_t index, unsigned int gvl)
- `int64x3xm1_t vlsxseg3ev_int64x3xm1_int64xm1` (const long \*address, int64xm1\_t index, unsigned int gvl)
- `int64x3xm2_t vlsxseg3ev_int64x3xm2_int64xm2` (const long \*address, int64xm2\_t index, unsigned int gvl)
- `int8x3xm1_t vlsxseg3ev_int8x3xm1_int8xm1` (const signed char \*address, int8xm1\_t index, unsigned int gvl)
- `int8x3xm2_t vlsxseg3ev_int8x3xm2_int8xm2` (const signed char \*address, int8xm2\_t index, unsigned int gvl)
- `uint16x3xm1_t vlsxseg3ev_uint16x3xm1_uint16xm1` (const unsigned short \*address, uint16xm1\_t index, unsigned int gvl)
- `uint16x3xm2_t vlsxseg3ev_uint16x3xm2_uint16xm2` (const unsigned short \*address, uint16xm2\_t index, unsigned int gvl)
- `uint32x3xm1_t vlsxseg3ev_uint32x3xm1_uint32xm1` (const unsigned int \*address, uint32xm1\_t index, unsigned int gvl)
- `uint32x3xm2_t vlsxseg3ev_uint32x3xm2_uint32xm2` (const unsigned int \*address, uint32xm2\_t index, unsigned int gvl)
- `uint64x3xm1_t vlsxseg3ev_uint64x3xm1_uint64xm1` (const unsigned long \*address, uint64xm1\_t index, unsigned int gvl)
- `uint64x3xm2_t vlsxseg3ev_uint64x3xm2_uint64xm2` (const unsigned long \*address, uint64xm2\_t index, unsigned int gvl)

- `uint8x3xm1_t vlxseg3ev_uint8x3xm1_uint8xm1` (const unsigned char \*address, *uint8xm1\_t* index, unsigned int gvl)
- `uint8x3xm2_t vlxseg3ev_uint8x3xm2_uint8xm2` (const unsigned char \*address, *uint8xm2\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x3xm1_t vlxseg3ev_mask_float16x3xm1_float16xm1` (float16x3xm1\_t merge, const float16\_t \*address, float16xm1\_t index, e16xm1\_t mask, unsigned int gvl)
- `float16x3xm2_t vlxseg3ev_mask_float16x3xm2_float16xm2` (float16x3xm2\_t merge, const float16\_t \*address, float16xm2\_t index, e16xm2\_t mask, unsigned int gvl)
- `float32x3xm1_t vlxseg3ev_mask_float32x3xm1_float32xm1` (float32x3xm1\_t merge, const float \*address, float32xm1\_t index, e32xm1\_t mask, unsigned int gvl)
- `float32x3xm2_t vlxseg3ev_mask_float32x3xm2_float32xm2` (float32x3xm2\_t merge, const float \*address, float32xm2\_t index, e32xm2\_t mask, unsigned int gvl)
- `float64x3xm1_t vlxseg3ev_mask_float64x3xm1_float64xm1` (float64x3xm1\_t merge, const double \*address, float64xm1\_t index, e64xm1\_t mask, unsigned int gvl)
- `float64x3xm2_t vlxseg3ev_mask_float64x3xm2_float64xm2` (float64x3xm2\_t merge, const double \*address, float64xm2\_t index, e64xm2\_t mask, unsigned int gvl)
- `int16x3xm1_t vlxseg3ev_mask_int16x3xm1_int16xm1` (int16x3xm1\_t merge, const short \*address, int16xm1\_t index, e16xm1\_t mask, unsigned int gvl)
- `int16x3xm2_t vlxseg3ev_mask_int16x3xm2_int16xm2` (int16x3xm2\_t merge, const short \*address, int16xm2\_t index, e16xm2\_t mask, unsigned int gvl)
- `int32x3xm1_t vlxseg3ev_mask_int32x3xm1_int32xm1` (int32x3xm1\_t merge, const int \*address, int32xm1\_t index, e32xm1\_t mask, unsigned int gvl)



- `int32x3xm2_t vlxseg3ev_mask_int32x3xm2_int32xm2` (`int32x3xm2_t merge`, `const int *address`, `int32xm2_t index`, `e32xm2_t mask`, unsigned int `gvl`)
- `int64x3xm1_t vlxseg3ev_mask_int64x3xm1_int64xm1` (`int64x3xm1_t merge`, `const long *address`, `int64xm1_t index`, `e64xm1_t mask`, unsigned int `gvl`)
- `int64x3xm2_t vlxseg3ev_mask_int64x3xm2_int64xm2` (`int64x3xm2_t merge`, `const long *address`, `int64xm2_t index`, `e64xm2_t mask`, unsigned int `gvl`)
- `int8x3xm1_t vlxseg3ev_mask_int8x3xm1_int8xm1` (`int8x3xm1_t merge`, `const signed char *address`, `int8xm1_t index`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8x3xm2_t vlxseg3ev_mask_int8x3xm2_int8xm2` (`int8x3xm2_t merge`, `const signed char *address`, `int8xm2_t index`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint16x3xm1_t vlxseg3ev_mask_uint16x3xm1_uint16xm1` (`uint16x3xm1_t merge`, `const unsigned short *address`, `uint16xm1_t index`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint16x3xm2_t vlxseg3ev_mask_uint16x3xm2_uint16xm2` (`uint16x3xm2_t merge`, `const unsigned short *address`, `uint16xm2_t index`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint32x3xm1_t vlxseg3ev_mask_uint32x3xm1_uint32xm1` (`uint32x3xm1_t merge`, `const unsigned int *address`, `uint32xm1_t index`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32x3xm2_t vlxseg3ev_mask_uint32x3xm2_uint32xm2` (`uint32x3xm2_t merge`, `const unsigned int *address`, `uint32xm2_t index`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint64x3xm1_t vlxseg3ev_mask_uint64x3xm1_uint64xm1` (`uint64x3xm1_t merge`, `const unsigned long *address`, `uint64xm1_t index`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64x3xm2_t vlxseg3ev_mask_uint64x3xm2_uint64xm2` (`uint64x3xm2_t merge`, `const unsigned long *address`, `uint64xm2_t index`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint8x3xm1_t vlxseg3ev_mask_uint8x3xm1_uint8xm1` (`uint8x3xm1_t merge`, `const unsigned char *address`, `uint8xm1_t index`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8x3xm2_t vlxseg3ev_mask_uint8x3xm2_uint8xm2` (`uint8x3xm2_t merge`, `const unsigned char *address`, `uint8xm2_t index`, `e8xm2_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
      if mask[segment] then
```

(continues on next page)

(continued from previous page)

```

    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
else
    result[segment] = merge[segment]
result[gvl : VLMAX] = 0

```

### 2.10.109 Load 3 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlxseg3h.v'`]

**Prototypes:**

- `int16x3xm1_t vlxseg3hv_int16x3xm1_int16xm1` (const short *\*address*, *int16xm1\_t* index, unsigned int *gvl*)
- `int16x3xm2_t vlxseg3hv_int16x3xm2_int16xm2` (const short *\*address*, *int16xm2\_t* index, unsigned int *gvl*)
- `int32x3xm1_t vlxseg3hv_int32x3xm1_int32xm1` (const int *\*address*, *int32xm1\_t* index, unsigned int *gvl*)
- `int32x3xm2_t vlxseg3hv_int32x3xm2_int32xm2` (const int *\*address*, *int32xm2\_t* index, unsigned int *gvl*)
- `int64x3xm1_t vlxseg3hv_int64x3xm1_int64xm1` (const long *\*address*, *int64xm1\_t* index, unsigned int *gvl*)
- `int64x3xm2_t vlxseg3hv_int64x3xm2_int64xm2` (const long *\*address*, *int64xm2\_t* index, unsigned int *gvl*)
- `int8x3xm1_t vlxseg3hv_int8x3xm1_int8xm1` (const signed char *\*address*, *int8xm1\_t* index, unsigned int *gvl*)
- `int8x3xm2_t vlxseg3hv_int8x3xm2_int8xm2` (const signed char *\*address*, *int8xm2\_t* index, unsigned int *gvl*)

**Operation:**

```

>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- `int16x3xm1_t vlxseg3hv_mask_int16x3xm1_int16xm1` (*int16x3xm1\_t* merge, const short *\*address*, *int16xm1\_t* index, *e16xm1\_t* mask, unsigned int *gvl*)
- `int16x3xm2_t vlxseg3hv_mask_int16x3xm2_int16xm2` (*int16x3xm2\_t* merge, const short *\*address*, *int16xm2\_t* index, *e16xm2\_t* mask, unsigned int *gvl*)
- `int32x3xm1_t vlxseg3hv_mask_int32x3xm1_int32xm1` (*int32x3xm1\_t* merge, const int *\*address*, *int32xm1\_t* index, *e32xm1\_t* mask, unsigned int *gvl*)
- `int32x3xm2_t vlxseg3hv_mask_int32x3xm2_int32xm2` (*int32x3xm2\_t* merge, const int *\*address*, *int32xm2\_t* index, *e32xm2\_t* mask, unsigned int *gvl*)

- `int64x3xm1_t vlxseg3hv_mask_int64x3xm1_int64xm1` (`int64x3xm1_t merge`, `const long *address`, `int64xm1_t index`, `e64xm1_t mask`, `unsigned int gvl`)
- `int64x3xm2_t vlxseg3hv_mask_int64x3xm2_int64xm2` (`int64x3xm2_t merge`, `const long *address`, `int64xm2_t index`, `e64xm2_t mask`, `unsigned int gvl`)
- `int8x3xm1_t vlxseg3hv_mask_int8x3xm1_int8xm1` (`int8x3xm1_t merge`, `const signed char *address`, `int8xm1_t index`, `e8xm1_t mask`, `unsigned int gvl`)
- `int8x3xm2_t vlxseg3hv_mask_int8x3xm2_int8xm2` (`int8x3xm2_t merge`, `const signed char *address`, `int8xm2_t index`, `e8xm2_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.110 Load 3 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`vlxseg3hu.v`']**Prototypes:**

- `uint16x3xm1_t vlxseg3huv_uint16x3xm1_uint16xm1` (`const unsigned short *address`, `uint16xm1_t index`, `unsigned int gvl`)
- `uint16x3xm2_t vlxseg3huv_uint16x3xm2_uint16xm2` (`const unsigned short *address`, `uint16xm2_t index`, `unsigned int gvl`)
- `uint32x3xm1_t vlxseg3huv_uint32x3xm1_uint32xm1` (`const unsigned int *address`, `uint32xm1_t index`, `unsigned int gvl`)
- `uint32x3xm2_t vlxseg3huv_uint32x3xm2_uint32xm2` (`const unsigned int *address`, `uint32xm2_t index`, `unsigned int gvl`)
- `uint64x3xm1_t vlxseg3huv_uint64x3xm1_uint64xm1` (`const unsigned long *address`, `uint64xm1_t index`, `unsigned int gvl`)
- `uint64x3xm2_t vlxseg3huv_uint64x3xm2_uint64xm2` (`const unsigned long *address`, `uint64xm2_t index`, `unsigned int gvl`)
- `uint8x3xm1_t vlxseg3huv_uint8x3xm1_uint8xm1` (`const unsigned char *address`, `uint8xm1_t index`, `unsigned int gvl`)
- `uint8x3xm2_t vlxseg3huv_uint8x3xm2_uint8xm2` (`const unsigned char *address`, `uint8xm2_t index`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
```

(continues on next page)

(continued from previous page)

```

address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- `uint16x3xm1_t vlxseg3huv_mask_uint16x3xm1_uint16xm1` (`uint16x3xm1_t merge, const unsigned short *address, uint16xm1_t index, e16xm1_t mask, unsigned int gvl`)
- `uint16x3xm2_t vlxseg3huv_mask_uint16x3xm2_uint16xm2` (`uint16x3xm2_t merge, const unsigned short *address, uint16xm2_t index, e16xm2_t mask, unsigned int gvl`)
- `uint32x3xm1_t vlxseg3huv_mask_uint32x3xm1_uint32xm1` (`uint32x3xm1_t merge, const unsigned int *address, uint32xm1_t index, e32xm1_t mask, unsigned int gvl`)
- `uint32x3xm2_t vlxseg3huv_mask_uint32x3xm2_uint32xm2` (`uint32x3xm2_t merge, const unsigned int *address, uint32xm2_t index, e32xm2_t mask, unsigned int gvl`)
- `uint64x3xm1_t vlxseg3huv_mask_uint64x3xm1_uint64xm1` (`uint64x3xm1_t merge, const unsigned long *address, uint64xm1_t index, e64xm1_t mask, unsigned int gvl`)
- `uint64x3xm2_t vlxseg3huv_mask_uint64x3xm2_uint64xm2` (`uint64x3xm2_t merge, const unsigned long *address, uint64xm2_t index, e64xm2_t mask, unsigned int gvl`)
- `uint8x3xm1_t vlxseg3huv_mask_uint8x3xm1_uint8xm1` (`uint8x3xm1_t merge, const unsigned char *address, uint8xm1_t index, e8xm1_t mask, unsigned int gvl`)
- `uint8x3xm2_t vlxseg3huv_mask_uint8x3xm2_uint8xm2` (`uint8x3xm2_t merge, const unsigned char *address, uint8xm2_t index, e8xm2_t mask, unsigned int gvl`)

**Masked operation:**

```

>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0

```

## 2.10.111 Load 3 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlxseg3w.v'`]

**Prototypes:**

- `int16x3xm1_t vlxseg3wv_int16x3xm1_int16xm1` (const short *\*address*, *int16xm1\_t* *index*, unsigned int *gvl*)
- `int16x3xm2_t vlxseg3wv_int16x3xm2_int16xm2` (const short *\*address*, *int16xm2\_t* *index*, unsigned int *gvl*)
- `int32x3xm1_t vlxseg3wv_int32x3xm1_int32xm1` (const int *\*address*, *int32xm1\_t* *index*, unsigned int *gvl*)
- `int32x3xm2_t vlxseg3wv_int32x3xm2_int32xm2` (const int *\*address*, *int32xm2\_t* *index*, unsigned int *gvl*)
- `int64x3xm1_t vlxseg3wv_int64x3xm1_int64xm1` (const long *\*address*, *int64xm1\_t* *index*, unsigned int *gvl*)
- `int64x3xm2_t vlxseg3wv_int64x3xm2_int64xm2` (const long *\*address*, *int64xm2\_t* *index*, unsigned int *gvl*)
- `int8x3xm1_t vlxseg3wv_int8x3xm1_int8xm1` (const signed char *\*address*, *int8xm1\_t* *index*, unsigned int *gvl*)
- `int8x3xm2_t vlxseg3wv_int8x3xm2_int8xm2` (const signed char *\*address*, *int8xm2\_t* *index*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x3xm1_t vlxseg3wv_mask_int16x3xm1_int16xm1` (*int16x3xm1\_t merge*, const short *\*address*, *int16xm1\_t* *index*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `int16x3xm2_t vlxseg3wv_mask_int16x3xm2_int16xm2` (*int16x3xm2\_t merge*, const short *\*address*, *int16xm2\_t* *index*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- `int32x3xm1_t vlxseg3wv_mask_int32x3xm1_int32xm1` (*int32x3xm1\_t merge*, const int *\*address*, *int32xm1\_t* *index*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- `int32x3xm2_t vlxseg3wv_mask_int32x3xm2_int32xm2` (*int32x3xm2\_t merge*, const int *\*address*, *int32xm2\_t* *index*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- `int64x3xm1_t vlxseg3wv_mask_int64x3xm1_int64xm1` (*int64x3xm1\_t merge*, const long *\*address*, *int64xm1\_t* *index*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- `int64x3xm2_t vlxseg3wv_mask_int64x3xm2_int64xm2` (*int64x3xm2\_t merge*, const long *\*address*, *int64xm2\_t* *index*, *e64xm2\_t* *mask*, unsigned int *gvl*)

- `int8x3xm1_t vlxseg3wv_mask_int8x3xm1_int8xm1` (`int8x3xm1_t merge`, `const signed char *address`, `int8xm1_t index`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8x3xm2_t vlxseg3wv_mask_int8x3xm2_int8xm2` (`int8x3xm2_t merge`, `const signed char *address`, `int8xm2_t index`, `e8xm2_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.112 Load 3 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`'vlxseg3wu.v'`]**Prototypes:**

- `uint16x3xm1_t vlxseg3wuv_uint16x3xm1_uint16xm1` (`const unsigned short *address`, `uint16xm1_t index`, unsigned int `gvl`)
- `uint16x3xm2_t vlxseg3wuv_uint16x3xm2_uint16xm2` (`const unsigned short *address`, `uint16xm2_t index`, unsigned int `gvl`)
- `uint32x3xm1_t vlxseg3wuv_uint32x3xm1_uint32xm1` (`const unsigned int *address`, `uint32xm1_t index`, unsigned int `gvl`)
- `uint32x3xm2_t vlxseg3wuv_uint32x3xm2_uint32xm2` (`const unsigned int *address`, `uint32xm2_t index`, unsigned int `gvl`)
- `uint64x3xm1_t vlxseg3wuv_uint64x3xm1_uint64xm1` (`const unsigned long *address`, `uint64xm1_t index`, unsigned int `gvl`)
- `uint64x3xm2_t vlxseg3wuv_uint64x3xm2_uint64xm2` (`const unsigned long *address`, `uint64xm2_t index`, unsigned int `gvl`)
- `uint8x3xm1_t vlxseg3wuv_uint8x3xm1_uint8xm1` (`const unsigned char *address`, `uint8xm1_t index`, unsigned int `gvl`)
- `uint8x3xm2_t vlxseg3wuv_uint8x3xm2_uint8xm2` (`const unsigned char *address`, `uint8xm2_t index`, unsigned int `gvl`)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**



- `uint16x3xm1_t vlxseg3wuv_mask_uint16x3xm1_uint16xm1` (`uint16x3xm1_t merge, const unsigned short *address, uint16xm1_t index, e16xm1_t mask, unsigned int gvl`)
- `uint16x3xm2_t vlxseg3wuv_mask_uint16x3xm2_uint16xm2` (`uint16x3xm2_t merge, const unsigned short *address, uint16xm2_t index, e16xm2_t mask, unsigned int gvl`)
- `uint32x3xm1_t vlxseg3wuv_mask_uint32x3xm1_uint32xm1` (`uint32x3xm1_t merge, const unsigned int *address, uint32xm1_t index, e32xm1_t mask, unsigned int gvl`)
- `uint32x3xm2_t vlxseg3wuv_mask_uint32x3xm2_uint32xm2` (`uint32x3xm2_t merge, const unsigned int *address, uint32xm2_t index, e32xm2_t mask, unsigned int gvl`)
- `uint64x3xm1_t vlxseg3wuv_mask_uint64x3xm1_uint64xm1` (`uint64x3xm1_t merge, const unsigned long *address, uint64xm1_t index, e64xm1_t mask, unsigned int gvl`)
- `uint64x3xm2_t vlxseg3wuv_mask_uint64x3xm2_uint64xm2` (`uint64x3xm2_t merge, const unsigned long *address, uint64xm2_t index, e64xm2_t mask, unsigned int gvl`)
- `uint8x3xm1_t vlxseg3wuv_mask_uint8x3xm1_uint8xm1` (`uint8x3xm1_t merge, const unsigned char *address, uint8xm1_t index, e8xm1_t mask, unsigned int gvl`)
- `uint8x3xm2_t vlxseg3wuv_mask_uint8x3xm2_uint8xm2` (`uint8x3xm2_t merge, const unsigned char *address, uint8xm2_t index, e8xm2_t mask, unsigned int gvl`)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.113 Load 4 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`'vlxseg4b.v'`]

**Prototypes:**

- `int16x4xm1_t vlxseg4bv_int16x4xm1_int16xm1` (const short \**address*, *int16xm1\_t* *index*, unsigned int *gvl*)
- `int16x4xm2_t vlxseg4bv_int16x4xm2_int16xm2` (const short \**address*, *int16xm2\_t* *index*, unsigned int *gvl*)
- `int32x4xm1_t vlxseg4bv_int32x4xm1_int32xm1` (const int \**address*, *int32xm1\_t* *index*, unsigned int *gvl*)
- `int32x4xm2_t vlxseg4bv_int32x4xm2_int32xm2` (const int \**address*, *int32xm2\_t* *index*, unsigned int *gvl*)
- `int64x4xm1_t vlxseg4bv_int64x4xm1_int64xm1` (const long \**address*, *int64xm1\_t* *index*, unsigned int *gvl*)
- `int64x4xm2_t vlxseg4bv_int64x4xm2_int64xm2` (const long \**address*, *int64xm2\_t* *index*, unsigned int *gvl*)
- `int8x4xm1_t vlxseg4bv_int8x4xm1_int8xm1` (const signed char \**address*, *int8xm1\_t* *index*, unsigned int *gvl*)
- `int8x4xm2_t vlxseg4bv_int8x4xm2_int8xm2` (const signed char \**address*, *int8xm2\_t* *index*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x4xm1_t vlxseg4bv_mask_int16x4xm1_int16xm1` (*int16x4xm1\_t merge*, const short \**address*, *int16xm1\_t* *index*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `int16x4xm2_t vlxseg4bv_mask_int16x4xm2_int16xm2` (*int16x4xm2\_t merge*, const short \**address*, *int16xm2\_t* *index*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- `int32x4xm1_t vlxseg4bv_mask_int32x4xm1_int32xm1` (*int32x4xm1\_t merge*, const int \**address*, *int32xm1\_t* *index*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- `int32x4xm2_t vlxseg4bv_mask_int32x4xm2_int32xm2` (*int32x4xm2\_t merge*, const int \**address*, *int32xm2\_t* *index*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- `int64x4xm1_t vlxseg4bv_mask_int64x4xm1_int64xm1` (*int64x4xm1\_t merge*, const long \**address*, *int64xm1\_t* *index*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- `int64x4xm2_t vlxseg4bv_mask_int64x4xm2_int64xm2` (*int64x4xm2\_t merge*, const long \**address*, *int64xm2\_t* *index*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- `int8x4xm1_t vlxseg4bv_mask_int8x4xm1_int8xm1` (*int8x4xm1\_t merge*, const signed char \**address*, *int8xm1\_t* *index*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- `int8x4xm2_t vlxseg4bv_mask_int8x4xm2_int8xm2` (*int8x4xm2\_t merge*, const signed char \**address*, *int8xm2\_t* *index*, *e8xm2\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.114 Load 4 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`vlxseg4bu.v`']**Prototypes:**

- `uint16x4xm1_t vlxseg4bu_v_uint16x4xm1_uint16xm1` (const unsigned short \*address, *uint16xm1\_t* index, unsigned int gvl)
- `uint16x4xm2_t vlxseg4bu_v_uint16x4xm2_uint16xm2` (const unsigned short \*address, *uint16xm2\_t* index, unsigned int gvl)
- `uint32x4xm1_t vlxseg4bu_v_uint32x4xm1_uint32xm1` (const unsigned int \*address, *uint32xm1\_t* index, unsigned int gvl)
- `uint32x4xm2_t vlxseg4bu_v_uint32x4xm2_uint32xm2` (const unsigned int \*address, *uint32xm2\_t* index, unsigned int gvl)
- `uint64x4xm1_t vlxseg4bu_v_uint64x4xm1_uint64xm1` (const unsigned long \*address, *uint64xm1\_t* index, unsigned int gvl)
- `uint64x4xm2_t vlxseg4bu_v_uint64x4xm2_uint64xm2` (const unsigned long \*address, *uint64xm2\_t* index, unsigned int gvl)
- `uint8x4xm1_t vlxseg4bu_v_uint8x4xm1_uint8xm1` (const unsigned char \*address, *uint8xm1\_t* index, unsigned int gvl)
- `uint8x4xm2_t vlxseg4bu_v_uint8x4xm2_uint8xm2` (const unsigned char \*address, *uint8xm2\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x4xm1_t vlxseg4bu_v_mask_uint16x4xm1_uint16xm1` (*uint16x4xm1\_t* merge, const unsigned short \*address, *uint16xm1\_t* index, *e16xm1\_t* mask, unsigned int gvl)
- `uint16x4xm2_t vlxseg4bu_v_mask_uint16x4xm2_uint16xm2` (*uint16x4xm2\_t* merge, const unsigned short \*address, *uint16xm2\_t* index, *e16xm2\_t* mask, unsigned int gvl)

- `uint32x4xm1_t vlxseg4buvmask_uint32x4xm1_uint32xm1` (`uint32x4xm1_t merge, const unsigned int *address, uint32xm1_t index, e32xm1_t mask, unsigned int gvl`)
- `uint32x4xm2_t vlxseg4buvmask_uint32x4xm2_uint32xm2` (`uint32x4xm2_t merge, const unsigned int *address, uint32xm2_t index, e32xm2_t mask, unsigned int gvl`)
- `uint64x4xm1_t vlxseg4buvmask_uint64x4xm1_uint64xm1` (`uint64x4xm1_t merge, const unsigned long *address, uint64xm1_t index, e64xm1_t mask, unsigned int gvl`)
- `uint64x4xm2_t vlxseg4buvmask_uint64x4xm2_uint64xm2` (`uint64x4xm2_t merge, const unsigned long *address, uint64xm2_t index, e64xm2_t mask, unsigned int gvl`)
- `uint8x4xm1_t vlxseg4buvmask_uint8x4xm1_uint8xm1` (`uint8x4xm1_t merge, const unsigned char *address, uint8xm1_t index, e8xm1_t mask, unsigned int gvl`)
- `uint8x4xm2_t vlxseg4buvmask_uint8x4xm2_uint8xm2` (`uint8x4xm2_t merge, const unsigned char *address, uint8xm2_t index, e8xm2_t mask, unsigned int gvl`)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.115 Load 4 contiguous element fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`'vlxseg4e.v'`]**Prototypes:**

- `float16x4xm1_t vlxseg4ev_float16x4xm1_float16xm1` (`const float16_t *address, float16xm1_t index, unsigned int gvl`)
- `float16x4xm2_t vlxseg4ev_float16x4xm2_float16xm2` (`const float16_t *address, float16xm2_t index, unsigned int gvl`)
- `float32x4xm1_t vlxseg4ev_float32x4xm1_float32xm1` (`const float *address, float32xm1_t index, unsigned int gvl`)
- `float32x4xm2_t vlxseg4ev_float32x4xm2_float32xm2` (`const float *address, float32xm2_t index, unsigned int gvl`)

- `float64x4xm1_t vlxseg4ev_float64x4xm1_float64xm1` (const double *\*address*, *float64xm1\_t* index, unsigned int gvl)
- `float64x4xm2_t vlxseg4ev_float64x4xm2_float64xm2` (const double *\*address*, *float64xm2\_t* index, unsigned int gvl)
- `int16x4xm1_t vlxseg4ev_int16x4xm1_int16xm1` (const short *\*address*, *int16xm1\_t* index, unsigned int gvl)
- `int16x4xm2_t vlxseg4ev_int16x4xm2_int16xm2` (const short *\*address*, *int16xm2\_t* index, unsigned int gvl)
- `int32x4xm1_t vlxseg4ev_int32x4xm1_int32xm1` (const int *\*address*, *int32xm1\_t* index, unsigned int gvl)
- `int32x4xm2_t vlxseg4ev_int32x4xm2_int32xm2` (const int *\*address*, *int32xm2\_t* index, unsigned int gvl)
- `int64x4xm1_t vlxseg4ev_int64x4xm1_int64xm1` (const long *\*address*, *int64xm1\_t* index, unsigned int gvl)
- `int64x4xm2_t vlxseg4ev_int64x4xm2_int64xm2` (const long *\*address*, *int64xm2\_t* index, unsigned int gvl)
- `int8x4xm1_t vlxseg4ev_int8x4xm1_int8xm1` (const signed char *\*address*, *int8xm1\_t* index, unsigned int gvl)
- `int8x4xm2_t vlxseg4ev_int8x4xm2_int8xm2` (const signed char *\*address*, *int8xm2\_t* index, unsigned int gvl)
- `uint16x4xm1_t vlxseg4ev_uint16x4xm1_uint16xm1` (const unsigned short *\*address*, *uint16xm1\_t* index, unsigned int gvl)
- `uint16x4xm2_t vlxseg4ev_uint16x4xm2_uint16xm2` (const unsigned short *\*address*, *uint16xm2\_t* index, unsigned int gvl)
- `uint32x4xm1_t vlxseg4ev_uint32x4xm1_uint32xm1` (const unsigned int *\*address*, *uint32xm1\_t* index, unsigned int gvl)
- `uint32x4xm2_t vlxseg4ev_uint32x4xm2_uint32xm2` (const unsigned int *\*address*, *uint32xm2\_t* index, unsigned int gvl)
- `uint64x4xm1_t vlxseg4ev_uint64x4xm1_uint64xm1` (const unsigned long *\*address*, *uint64xm1\_t* index, unsigned int gvl)
- `uint64x4xm2_t vlxseg4ev_uint64x4xm2_uint64xm2` (const unsigned long *\*address*, *uint64xm2\_t* index, unsigned int gvl)
- `uint8x4xm1_t vlxseg4ev_uint8x4xm1_uint8xm1` (const unsigned char *\*address*, *uint8xm1\_t* index, unsigned int gvl)
- `uint8x4xm2_t vlxseg4ev_uint8x4xm2_uint8xm2` (const unsigned char *\*address*, *uint8xm2\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      result[segment] = load_segment(address)
      address = address + sizeof(segment) + index[segment]
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x4xm1_t vlxseg4ev_mask_float16x4xm1_float16xm1` (`float16x4xm1_t` *merge*,  
const `float16_t` *\*address*, `float16xm1_t` *index*,  
`e16xm1_t` *mask*, unsigned  
int gvl)
- `float16x4xm2_t vlxseg4ev_mask_float16x4xm2_float16xm2` (`float16x4xm2_t` *merge*,  
const `float16_t` *\*address*, `float16xm2_t` *index*,  
`e16xm2_t` *mask*, unsigned  
int gvl)
- `float32x4xm1_t vlxseg4ev_mask_float32x4xm1_float32xm1` (`float32x4xm1_t` *merge*,  
const `float` *\*address*,  
`float32xm1_t` *index*,  
`e32xm1_t` *mask*, unsigned  
int gvl)
- `float32x4xm2_t vlxseg4ev_mask_float32x4xm2_float32xm2` (`float32x4xm2_t` *merge*,  
const `float` *\*address*,  
`float32xm2_t` *index*,  
`e32xm2_t` *mask*, unsigned  
int gvl)
- `float64x4xm1_t vlxseg4ev_mask_float64x4xm1_float64xm1` (`float64x4xm1_t` *merge*,  
const `double` *\*address*,  
`float64xm1_t` *index*,  
`e64xm1_t` *mask*, unsigned  
int gvl)
- `float64x4xm2_t vlxseg4ev_mask_float64x4xm2_float64xm2` (`float64x4xm2_t` *merge*,  
const `double` *\*address*,  
`float64xm2_t` *index*,  
`e64xm2_t` *mask*, unsigned  
int gvl)
- `int16x4xm1_t vlxseg4ev_mask_int16x4xm1_int16xm1` (`int16x4xm1_t` *merge*, const  
short *\*address*, `int16xm1_t` *index*,  
`e16xm1_t` *mask*, unsigned int gvl)
- `int16x4xm2_t vlxseg4ev_mask_int16x4xm2_int16xm2` (`int16x4xm2_t` *merge*, const  
short *\*address*, `int16xm2_t` *index*,  
`e16xm2_t` *mask*, unsigned int gvl)
- `int32x4xm1_t vlxseg4ev_mask_int32x4xm1_int32xm1` (`int32x4xm1_t` *merge*, const  
int *\*address*, `int32xm1_t` *index*,  
`e32xm1_t` *mask*, unsigned int gvl)
- `int32x4xm2_t vlxseg4ev_mask_int32x4xm2_int32xm2` (`int32x4xm2_t` *merge*, const  
int *\*address*, `int32xm2_t` *index*,  
`e32xm2_t` *mask*, unsigned int gvl)
- `int64x4xm1_t vlxseg4ev_mask_int64x4xm1_int64xm1` (`int64x4xm1_t` *merge*, const  
long *\*address*, `int64xm1_t` *index*,  
`e64xm1_t` *mask*, unsigned int gvl)
- `int64x4xm2_t vlxseg4ev_mask_int64x4xm2_int64xm2` (`int64x4xm2_t` *merge*, const  
long *\*address*, `int64xm2_t` *index*,  
`e64xm2_t` *mask*, unsigned int gvl)



- `int8x4xm1_t vlxseg4ev_mask_int8x4xm1_int8xm1` (`int8x4xm1_t merge`, const signed char *\*address*, `int8xm1_t` *index*, `e8xm1_t` *mask*, unsigned int *gvl*)
- `int8x4xm2_t vlxseg4ev_mask_int8x4xm2_int8xm2` (`int8x4xm2_t merge`, const signed char *\*address*, `int8xm2_t` *index*, `e8xm2_t` *mask*, unsigned int *gvl*)
- `uint16x4xm1_t vlxseg4ev_mask_uint16x4xm1_uint16xm1` (`uint16x4xm1_t merge`, const unsigned short *\*address*, `uint16xm1_t` *index*, `e16xm1_t` *mask*, unsigned int *gvl*)
- `uint16x4xm2_t vlxseg4ev_mask_uint16x4xm2_uint16xm2` (`uint16x4xm2_t merge`, const unsigned short *\*address*, `uint16xm2_t` *index*, `e16xm2_t` *mask*, unsigned int *gvl*)
- `uint32x4xm1_t vlxseg4ev_mask_uint32x4xm1_uint32xm1` (`uint32x4xm1_t merge`, const unsigned int *\*address*, `uint32xm1_t` *index*, `e32xm1_t` *mask*, unsigned int *gvl*)
- `uint32x4xm2_t vlxseg4ev_mask_uint32x4xm2_uint32xm2` (`uint32x4xm2_t merge`, const unsigned int *\*address*, `uint32xm2_t` *index*, `e32xm2_t` *mask*, unsigned int *gvl*)
- `uint64x4xm1_t vlxseg4ev_mask_uint64x4xm1_uint64xm1` (`uint64x4xm1_t merge`, const unsigned long *\*address*, `uint64xm1_t` *index*, `e64xm1_t` *mask*, unsigned int *gvl*)
- `uint64x4xm2_t vlxseg4ev_mask_uint64x4xm2_uint64xm2` (`uint64x4xm2_t merge`, const unsigned long *\*address*, `uint64xm2_t` *index*, `e64xm2_t` *mask*, unsigned int *gvl*)
- `uint8x4xm1_t vlxseg4ev_mask_uint8x4xm1_uint8xm1` (`uint8x4xm1_t merge`, const unsigned char *\*address*, `uint8xm1_t` *index*, `e8xm1_t` *mask*, unsigned int *gvl*)
- `uint8x4xm2_t vlxseg4ev_mask_uint8x4xm2_uint8xm2` (`uint8x4xm2_t merge`, const unsigned char *\*address*, `uint8xm2_t` *index*, `e8xm2_t` *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.116 Load 4 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`'vlxseg4h.v'`]

**Prototypes:**

- `int16x4xm1_t vlxseg4hv_int16x4xm1_int16xm1` (const short \**address*, *int16xm1\_t* *index*, unsigned int *gvl*)
- `int16x4xm2_t vlxseg4hv_int16x4xm2_int16xm2` (const short \**address*, *int16xm2\_t* *index*, unsigned int *gvl*)
- `int32x4xm1_t vlxseg4hv_int32x4xm1_int32xm1` (const int \**address*, *int32xm1\_t* *index*, unsigned int *gvl*)
- `int32x4xm2_t vlxseg4hv_int32x4xm2_int32xm2` (const int \**address*, *int32xm2\_t* *index*, unsigned int *gvl*)
- `int64x4xm1_t vlxseg4hv_int64x4xm1_int64xm1` (const long \**address*, *int64xm1\_t* *index*, unsigned int *gvl*)
- `int64x4xm2_t vlxseg4hv_int64x4xm2_int64xm2` (const long \**address*, *int64xm2\_t* *index*, unsigned int *gvl*)
- `int8x4xm1_t vlxseg4hv_int8x4xm1_int8xm1` (const signed char \**address*, *int8xm1\_t* *index*, unsigned int *gvl*)
- `int8x4xm2_t vlxseg4hv_int8x4xm2_int8xm2` (const signed char \**address*, *int8xm2\_t* *index*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x4xm1_t vlxseg4hv_mask_int16x4xm1_int16xm1` (*int16x4xm1\_t merge*, const short \**address*, *int16xm1\_t* *index*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- `int16x4xm2_t vlxseg4hv_mask_int16x4xm2_int16xm2` (*int16x4xm2\_t merge*, const short \**address*, *int16xm2\_t* *index*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- `int32x4xm1_t vlxseg4hv_mask_int32x4xm1_int32xm1` (*int32x4xm1\_t merge*, const int \**address*, *int32xm1\_t* *index*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- `int32x4xm2_t vlxseg4hv_mask_int32x4xm2_int32xm2` (*int32x4xm2\_t merge*, const int \**address*, *int32xm2\_t* *index*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- `int64x4xm1_t vlxseg4hv_mask_int64x4xm1_int64xm1` (*int64x4xm1\_t merge*, const long \**address*, *int64xm1\_t* *index*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- `int64x4xm2_t vlxseg4hv_mask_int64x4xm2_int64xm2` (*int64x4xm2\_t merge*, const long \**address*, *int64xm2\_t* *index*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- `int8x4xm1_t vlxseg4hv_mask_int8x4xm1_int8xm1` (*int8x4xm1\_t merge*, const signed char \**address*, *int8xm1\_t* *index*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- `int8x4xm2_t vlxseg4hv_mask_int8x4xm2_int8xm2` (*int8x4xm2\_t merge*, const signed char \**address*, *int8xm2\_t* *index*, *e8xm2\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.117 Load 4 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`vlxseg4hu.v`']**Prototypes:**

- `uint16x4xm1_t vlxseg4huv_uint16x4xm1_uint16xm1` (const unsigned short \*address, *uint16xm1\_t* index, unsigned int gvl)
- `uint16x4xm2_t vlxseg4huv_uint16x4xm2_uint16xm2` (const unsigned short \*address, *uint16xm2\_t* index, unsigned int gvl)
- `uint32x4xm1_t vlxseg4huv_uint32x4xm1_uint32xm1` (const unsigned int \*address, *uint32xm1\_t* index, unsigned int gvl)
- `uint32x4xm2_t vlxseg4huv_uint32x4xm2_uint32xm2` (const unsigned int \*address, *uint32xm2\_t* index, unsigned int gvl)
- `uint64x4xm1_t vlxseg4huv_uint64x4xm1_uint64xm1` (const unsigned long \*address, *uint64xm1\_t* index, unsigned int gvl)
- `uint64x4xm2_t vlxseg4huv_uint64x4xm2_uint64xm2` (const unsigned long \*address, *uint64xm2\_t* index, unsigned int gvl)
- `uint8x4xm1_t vlxseg4huv_uint8x4xm1_uint8xm1` (const unsigned char \*address, *uint8xm1\_t* index, unsigned int gvl)
- `uint8x4xm2_t vlxseg4huv_uint8x4xm2_uint8xm2` (const unsigned char \*address, *uint8xm2\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x4xm1_t vlxseg4huv_mask_uint16x4xm1_uint16xm1` (*uint16x4xm1\_t* merge, const unsigned short \*address, *uint16xm1\_t* index, *e16xm1\_t* mask, unsigned int gvl)
- `uint16x4xm2_t vlxseg4huv_mask_uint16x4xm2_uint16xm2` (*uint16x4xm2\_t* merge, const unsigned short \*address, *uint16xm2\_t* index, *e16xm2\_t* mask, unsigned int gvl)

- `uint32x4xm1_t vlxseg4huv_mask_uint32x4xm1_uint32xm1` (`uint32x4xm1_t merge, const unsigned int *address, uint32xm1_t index, e32xm1_t mask, unsigned int gvl`)
- `uint32x4xm2_t vlxseg4huv_mask_uint32x4xm2_uint32xm2` (`uint32x4xm2_t merge, const unsigned int *address, uint32xm2_t index, e32xm2_t mask, unsigned int gvl`)
- `uint64x4xm1_t vlxseg4huv_mask_uint64x4xm1_uint64xm1` (`uint64x4xm1_t merge, const unsigned long *address, uint64xm1_t index, e64xm1_t mask, unsigned int gvl`)
- `uint64x4xm2_t vlxseg4huv_mask_uint64x4xm2_uint64xm2` (`uint64x4xm2_t merge, const unsigned long *address, uint64xm2_t index, e64xm2_t mask, unsigned int gvl`)
- `uint8x4xm1_t vlxseg4huv_mask_uint8x4xm1_uint8xm1` (`uint8x4xm1_t merge, const unsigned char *address, uint8xm1_t index, e8xm1_t mask, unsigned int gvl`)
- `uint8x4xm2_t vlxseg4huv_mask_uint8x4xm2_uint8xm2` (`uint8x4xm2_t merge, const unsigned char *address, uint8xm2_t index, e8xm2_t mask, unsigned int gvl`)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.118 Load 4 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg4w.v`]

**Prototypes:**

- `int16x4xm1_t vlxseg4wv_int16x4xm1_int16xm1` (`const short *address, int16xm1_t index, unsigned int gvl`)
- `int16x4xm2_t vlxseg4wv_int16x4xm2_int16xm2` (`const short *address, int16xm2_t index, unsigned int gvl`)
- `int32x4xm1_t vlxseg4wv_int32x4xm1_int32xm1` (`const int *address, int32xm1_t index, unsigned int gvl`)
- `int32x4xm2_t vlxseg4wv_int32x4xm2_int32xm2` (`const int *address, int32xm2_t index, unsigned int gvl`)

- `int64x4xm1_t vlxseg4wv_int64x4xm1_int64xm1` (const long \*address, *int64xm1\_t* index, unsigned int gvl)
- `int64x4xm2_t vlxseg4wv_int64x4xm2_int64xm2` (const long \*address, *int64xm2\_t* index, unsigned int gvl)
- `int8x4xm1_t vlxseg4wv_int8x4xm1_int8xm1` (const signed char \*address, *int8xm1\_t* index, unsigned int gvl)
- `int8x4xm2_t vlxseg4wv_int8x4xm2_int8xm2` (const signed char \*address, *int8xm2\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x4xm1_t vlxseg4wv_mask_int16x4xm1_int16xm1` (int16x4xm1\_t merge, const short \*address, *int16xm1\_t* index, *e16xm1\_t* mask, unsigned int gvl)
- `int16x4xm2_t vlxseg4wv_mask_int16x4xm2_int16xm2` (int16x4xm2\_t merge, const short \*address, *int16xm2\_t* index, *e16xm2\_t* mask, unsigned int gvl)
- `int32x4xm1_t vlxseg4wv_mask_int32x4xm1_int32xm1` (int32x4xm1\_t merge, const int \*address, *int32xm1\_t* index, *e32xm1\_t* mask, unsigned int gvl)
- `int32x4xm2_t vlxseg4wv_mask_int32x4xm2_int32xm2` (int32x4xm2\_t merge, const int \*address, *int32xm2\_t* index, *e32xm2\_t* mask, unsigned int gvl)
- `int64x4xm1_t vlxseg4wv_mask_int64x4xm1_int64xm1` (int64x4xm1\_t merge, const long \*address, *int64xm1\_t* index, *e64xm1\_t* mask, unsigned int gvl)
- `int64x4xm2_t vlxseg4wv_mask_int64x4xm2_int64xm2` (int64x4xm2\_t merge, const long \*address, *int64xm2\_t* index, *e64xm2\_t* mask, unsigned int gvl)
- `int8x4xm1_t vlxseg4wv_mask_int8x4xm1_int8xm1` (int8x4xm1\_t merge, const signed char \*address, *int8xm1\_t* index, *e8xm1\_t* mask, unsigned int gvl)
- `int8x4xm2_t vlxseg4wv_mask_int8x4xm2_int8xm2` (int8x4xm2\_t merge, const signed char \*address, *int8xm2\_t* index, *e8xm2\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.119 Load 4 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg4wuv.v`]

**Prototypes:**

- `uint16x4xm1_t vlxseg4wuv_uint16x4xm1_uint16xm1` (const unsigned short \*address, *uint16xm1\_t* index, unsigned int gvl)
- `uint16x4xm2_t vlxseg4wuv_uint16x4xm2_uint16xm2` (const unsigned short \*address, *uint16xm2\_t* index, unsigned int gvl)
- `uint32x4xm1_t vlxseg4wuv_uint32x4xm1_uint32xm1` (const unsigned int \*address, *uint32xm1\_t* index, unsigned int gvl)
- `uint32x4xm2_t vlxseg4wuv_uint32x4xm2_uint32xm2` (const unsigned int \*address, *uint32xm2\_t* index, unsigned int gvl)
- `uint64x4xm1_t vlxseg4wuv_uint64x4xm1_uint64xm1` (const unsigned long \*address, *uint64xm1\_t* index, unsigned int gvl)
- `uint64x4xm2_t vlxseg4wuv_uint64x4xm2_uint64xm2` (const unsigned long \*address, *uint64xm2\_t* index, unsigned int gvl)
- `uint8x4xm1_t vlxseg4wuv_uint8x4xm1_uint8xm1` (const unsigned char \*address, *uint8xm1\_t* index, unsigned int gvl)
- `uint8x4xm2_t vlxseg4wuv_uint8x4xm2_uint8xm2` (const unsigned char \*address, *uint8xm2\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x4xm1_t vlxseg4wuv_mask_uint16x4xm1_uint16xm1` (*uint16x4xm1\_t* merge, const unsigned short \*address, *uint16xm1\_t* index, *e16xm1\_t* mask, unsigned int gvl)
- `uint16x4xm2_t vlxseg4wuv_mask_uint16x4xm2_uint16xm2` (*uint16x4xm2\_t* merge, const unsigned short \*address, *uint16xm2\_t* index, *e16xm2\_t* mask, unsigned int gvl)
- `uint32x4xm1_t vlxseg4wuv_mask_uint32x4xm1_uint32xm1` (*uint32x4xm1\_t* merge, const unsigned int \*address, *uint32xm1\_t* index, *e32xm1\_t* mask, unsigned int gvl)
- `uint32x4xm2_t vlxseg4wuv_mask_uint32x4xm2_uint32xm2` (*uint32x4xm2\_t* merge, const unsigned int \*address, *uint32xm2\_t* index, *e32xm2\_t* mask, unsigned int gvl)



- `uint64x4xm1_t vlxseg4wuv_mask_uint64x4xm1_uint64xm1` (`uint64x4xm1_t merge, const unsigned long *address, uint64xm1_t index, e64xm1_t mask, unsigned int gvl`)
- `uint64x4xm2_t vlxseg4wuv_mask_uint64x4xm2_uint64xm2` (`uint64x4xm2_t merge, const unsigned long *address, uint64xm2_t index, e64xm2_t mask, unsigned int gvl`)
- `uint8x4xm1_t vlxseg4wuv_mask_uint8x4xm1_uint8xm1` (`uint8x4xm1_t merge, const unsigned char *address, uint8xm1_t index, e8xm1_t mask, unsigned int gvl`)
- `uint8x4xm2_t vlxseg4wuv_mask_uint8x4xm2_uint8xm2` (`uint8x4xm2_t merge, const unsigned char *address, uint8xm2_t index, e8xm2_t mask, unsigned int gvl`)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.120 Load 5 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg5b.v`']

**Prototypes:**

- `int16x5xm1_t vlxseg5bv_int16x5xm1_int16xm1` (`const short *address, int16xm1_t index, unsigned int gvl`)
- `int32x5xm1_t vlxseg5bv_int32x5xm1_int32xm1` (`const int *address, int32xm1_t index, unsigned int gvl`)
- `int64x5xm1_t vlxseg5bv_int64x5xm1_int64xm1` (`const long *address, int64xm1_t index, unsigned int gvl`)
- `int8x5xm1_t vlxseg5bv_int8x5xm1_int8xm1` (`const signed char *address, int8xm1_t index, unsigned int gvl`)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x5xm1_t vlxseg5bv_mask_int16x5xm1_int16xm1` (`int16x5xm1_t merge, const short *address, int16xm1_t index, e16xm1_t mask, unsigned int gvl`)

- `int32x5xm1_t vlxseg5bv_mask_int32x5xm1_int32xm1` (`int32x5xm1_t merge`, `const int *address`, `int32xm1_t index`, `e32xm1_t mask`, `unsigned int gvl`)
- `int64x5xm1_t vlxseg5bv_mask_int64x5xm1_int64xm1` (`int64x5xm1_t merge`, `const long *address`, `int64xm1_t index`, `e64xm1_t mask`, `unsigned int gvl`)
- `int8x5xm1_t vlxseg5bv_mask_int8x5xm1_int8xm1` (`int8x5xm1_t merge`, `const signed char *address`, `int8xm1_t index`, `e8xm1_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.121 Load 5 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`vlxseg5bu.v`]**Prototypes:**

- `uint16x5xm1_t vlxseg5buv_uint16x5xm1_uint16xm1` (`const unsigned short *address`, `uint16xm1_t index`, `unsigned int gvl`)
- `uint32x5xm1_t vlxseg5buv_uint32x5xm1_uint32xm1` (`const unsigned int *address`, `uint32xm1_t index`, `unsigned int gvl`)
- `uint64x5xm1_t vlxseg5buv_uint64x5xm1_uint64xm1` (`const unsigned long *address`, `uint64xm1_t index`, `unsigned int gvl`)
- `uint8x5xm1_t vlxseg5buv_uint8x5xm1_uint8xm1` (`const unsigned char *address`, `uint8xm1_t index`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x5xm1_t vlxseg5buv_mask_uint16x5xm1_uint16xm1` (`uint16x5xm1_t merge`, `const unsigned short *address`, `uint16xm1_t index`, `e16xm1_t mask`, `unsigned int gvl`)
- `uint32x5xm1_t vlxseg5buv_mask_uint32x5xm1_uint32xm1` (`uint32x5xm1_t merge`, `const unsigned int *address`, `uint32xm1_t index`, `e32xm1_t mask`, `unsigned int gvl`)

- `uint64x5xml_t vlxseg5buvmask_uint64x5xml_uint64xml` (`uint64x5xml_t merge`, `const unsigned long *address`, `uint64xml_t index`, `e64xml_t mask`, `unsigned int gvl`)
- `uint8x5xml_t vlxseg5buvmask_uint8x5xml_uint8xml` (`uint8x5xml_t merge`, `const unsigned char *address`, `uint8xml_t index`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.122 Load 5 contiguous element fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg5e.v`]

**Prototypes:**

- `float16x5xml_t vlxseg5ev_float16x5xml_float16xml` (`const float16_t *address`, `float16xml_t index`, `unsigned int gvl`)
- `float32x5xml_t vlxseg5ev_float32x5xml_float32xml` (`const float *address`, `float32xml_t index`, `unsigned int gvl`)
- `float64x5xml_t vlxseg5ev_float64x5xml_float64xml` (`const double *address`, `float64xml_t index`, `unsigned int gvl`)
- `int16x5xml_t vlxseg5ev_int16x5xml_int16xml` (`const short *address`, `int16xml_t index`, `unsigned int gvl`)
- `int32x5xml_t vlxseg5ev_int32x5xml_int32xml` (`const int *address`, `int32xml_t index`, `unsigned int gvl`)
- `int64x5xml_t vlxseg5ev_int64x5xml_int64xml` (`const long *address`, `int64xml_t index`, `unsigned int gvl`)
- `int8x5xml_t vlxseg5ev_int8x5xml_int8xml` (`const signed char *address`, `int8xml_t index`, `unsigned int gvl`)
- `uint16x5xml_t vlxseg5ev_uint16x5xml_uint16xml` (`const unsigned short *address`, `uint16xml_t index`, `unsigned int gvl`)
- `uint32x5xml_t vlxseg5ev_uint32x5xml_uint32xml` (`const unsigned int *address`, `uint32xml_t index`, `unsigned int gvl`)
- `uint64x5xml_t vlxseg5ev_uint64x5xml_uint64xml` (`const unsigned long *address`, `uint64xml_t index`, `unsigned int gvl`)
- `uint8x5xml_t vlxseg5ev_uint8x5xml_uint8xml` (`const unsigned char *address`, `uint8xml_t index`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

### Masked prototypes:

- `float16x5xml_t vlxseg5ev_mask_float16x5xml_float16xml` (`float16x5xml_t` *merge*,  
const `float16_t` *\*address*, `float16xml_t` *index*,  
`e16xml_t` *mask*, unsigned int *gvl*)
- `float32x5xml_t vlxseg5ev_mask_float32x5xml_float32xml` (`float32x5xml_t` *merge*,  
const `float` *\*address*, `float32xml_t` *index*,  
`e32xml_t` *mask*, unsigned int *gvl*)
- `float64x5xml_t vlxseg5ev_mask_float64x5xml_float64xml` (`float64x5xml_t` *merge*,  
const `double` *\*address*, `float64xml_t` *index*,  
`e64xml_t` *mask*, unsigned int *gvl*)
- `int16x5xml_t vlxseg5ev_mask_int16x5xml_int16xml` (`int16x5xml_t` *merge*, const  
short *\*address*, `int16xml_t` *index*,  
`e16xml_t` *mask*, unsigned int *gvl*)
- `int32x5xml_t vlxseg5ev_mask_int32x5xml_int32xml` (`int32x5xml_t` *merge*, const  
int *\*address*, `int32xml_t` *index*,  
`e32xml_t` *mask*, unsigned int *gvl*)
- `int64x5xml_t vlxseg5ev_mask_int64x5xml_int64xml` (`int64x5xml_t` *merge*, const  
long *\*address*, `int64xml_t` *index*,  
`e64xml_t` *mask*, unsigned int *gvl*)
- `int8x5xml_t vlxseg5ev_mask_int8x5xml_int8xml` (`int8x5xml_t` *merge*, const signed char *\*ad-*  
*dress*, `int8xml_t` *index*, `e8xml_t` *mask*, un-  
signed int *gvl*)
- `uint16x5xml_t vlxseg5ev_mask_uint16x5xml_uint16xml` (`uint16x5xml_t` *merge*,  
const unsigned short *\*ad-*  
*dress*, `uint16xml_t` *index*,  
`e16xml_t` *mask*, unsigned int *gvl*)
- `uint32x5xml_t vlxseg5ev_mask_uint32x5xml_uint32xml` (`uint32x5xml_t` *merge*,  
const unsigned int *\*ad-*  
*dress*, `uint32xml_t` *index*,  
`e32xml_t` *mask*, unsigned int *gvl*)
- `uint64x5xml_t vlxseg5ev_mask_uint64x5xml_uint64xml` (`uint64x5xml_t` *merge*,  
const unsigned long *\*ad-*  
*dress*, `uint64xml_t` *index*,  
`e64xml_t` *mask*, unsigned int *gvl*)
- `uint8x5xml_t vlxseg5ev_mask_uint8x5xml_uint8xml` (`uint8x5xml_t` *merge*, const unsigned  
char *\*address*, `uint8xml_t` *index*,  
`e8xml_t` *mask*, unsigned int *gvl*)

### Masked operation:

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.123 Load 5 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** ['vlxseg5h.v']

**Prototypes:**

- `int16x5xml_t vlxseg5hv_int16x5xml_int16xml` (const short \*address, *int16xml\_t* index, unsigned int gvl)
- `int32x5xml_t vlxseg5hv_int32x5xml_int32xml` (const int \*address, *int32xml\_t* index, unsigned int gvl)
- `int64x5xml_t vlxseg5hv_int64x5xml_int64xml` (const long \*address, *int64xml\_t* index, unsigned int gvl)
- `int8x5xml_t vlxseg5hv_int8x5xml_int8xml` (const signed char \*address, *int8xml\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x5xml_t vlxseg5hv_mask_int16x5xml_int16xml` (int16x5xml\_t merge, const short \*address, *int16xml\_t* index, *e16xml\_t* mask, unsigned int gvl)
- `int32x5xml_t vlxseg5hv_mask_int32x5xml_int32xml` (int32x5xml\_t merge, const int \*address, *int32xml\_t* index, *e32xml\_t* mask, unsigned int gvl)
- `int64x5xml_t vlxseg5hv_mask_int64x5xml_int64xml` (int64x5xml\_t merge, const long \*address, *int64xml\_t* index, *e64xml\_t* mask, unsigned int gvl)
- `int8x5xml_t vlxseg5hv_mask_int8x5xml_int8xml` (int8x5xml\_t merge, const signed char \*address, *int8xml\_t* index, *e8xml\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
```

(continues on next page)

(continued from previous page)

```
result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.124 Load 5 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg5hu.v`]

**Prototypes:**

- `uint16x5xml_t vlxseg5huv_uint16x5xml_uint16xml` (const unsigned short \*address, *uint16xml\_t* index, unsigned int gvl)
- `uint32x5xml_t vlxseg5huv_uint32x5xml_uint32xml` (const unsigned int \*address, *uint32xml\_t* index, unsigned int gvl)
- `uint64x5xml_t vlxseg5huv_uint64x5xml_uint64xml` (const unsigned long \*address, *uint64xml\_t* index, unsigned int gvl)
- `uint8x5xml_t vlxseg5huv_uint8x5xml_uint8xml` (const unsigned char \*address, *uint8xml\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x5xml_t vlxseg5huv_mask_uint16x5xml_uint16xml` (uint16x5xml\_t merge, const unsigned short \*address, *uint16xml\_t* index, *e16xml\_t* mask, unsigned int gvl)
- `uint32x5xml_t vlxseg5huv_mask_uint32x5xml_uint32xml` (uint32x5xml\_t merge, const unsigned int \*address, *uint32xml\_t* index, *e32xml\_t* mask, unsigned int gvl)
- `uint64x5xml_t vlxseg5huv_mask_uint64x5xml_uint64xml` (uint64x5xml\_t merge, const unsigned long \*address, *uint64xml\_t* index, *e64xml\_t* mask, unsigned int gvl)
- `uint8x5xml_t vlxseg5huv_mask_uint8x5xml_uint8xml` (uint8x5xml\_t merge, const unsigned char \*address, *uint8xml\_t* index, *e8xml\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
```

(continues on next page)



(continued from previous page)

```

    address = address + sizeof(segment) + index[segment]
else
    result[segment] = merge[segment]
result[gvl : VLMAX] = 0

```

## 2.10.125 Load 5 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg5w.v`]

**Prototypes:**

- `int16x5xml_t vlxseg5wv_int16x5xml_int16xml` (const short \*address, *int16xml\_t* index, unsigned int gvl)
- `int32x5xml_t vlxseg5wv_int32x5xml_int32xml` (const int \*address, *int32xml\_t* index, unsigned int gvl)
- `int64x5xml_t vlxseg5wv_int64x5xml_int64xml` (const long \*address, *int64xml\_t* index, unsigned int gvl)
- `int8x5xml_t vlxseg5wv_int8x5xml_int8xml` (const signed char \*address, *int8xml\_t* index, unsigned int gvl)

**Operation:**

```

>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- `int16x5xml_t vlxseg5wv_mask_int16x5xml_int16xml` (int16x5xml\_t merge, const short \*address, *int16xml\_t* index, *e16xml\_t* mask, unsigned int gvl)
- `int32x5xml_t vlxseg5wv_mask_int32x5xml_int32xml` (int32x5xml\_t merge, const int \*address, *int32xml\_t* index, *e32xml\_t* mask, unsigned int gvl)
- `int64x5xml_t vlxseg5wv_mask_int64x5xml_int64xml` (int64x5xml\_t merge, const long \*address, *int64xml\_t* index, *e64xml\_t* mask, unsigned int gvl)
- `int8x5xml_t vlxseg5wv_mask_int8x5xml_int8xml` (int8x5xml\_t merge, const signed char \*address, *int8xml\_t* index, *e8xml\_t* mask, unsigned int gvl)

**Masked operation:**

```

>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0

```

## 2.10.126 Load 5 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlxseg5wuv.v'`]

**Prototypes:**

- `uint16x5xm1_t vlxseg5wuv_uint16x5xm1_uint16xm1` (const unsigned short \*address, *uint16xm1\_t* index, unsigned int gvl)
- `uint32x5xm1_t vlxseg5wuv_uint32x5xm1_uint32xm1` (const unsigned int \*address, *uint32xm1\_t* index, unsigned int gvl)
- `uint64x5xm1_t vlxseg5wuv_uint64x5xm1_uint64xm1` (const unsigned long \*address, *uint64xm1\_t* index, unsigned int gvl)
- `uint8x5xm1_t vlxseg5wuv_uint8x5xm1_uint8xm1` (const unsigned char \*address, *uint8xm1\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x5xm1_t vlxseg5wuv_mask_uint16x5xm1_uint16xm1` (uint16x5xm1\_t *merge*, const unsigned short \*address, *uint16xm1\_t* index, *e16xm1\_t* mask, unsigned int gvl)
- `uint32x5xm1_t vlxseg5wuv_mask_uint32x5xm1_uint32xm1` (uint32x5xm1\_t *merge*, const unsigned int \*address, *uint32xm1\_t* index, *e32xm1\_t* mask, unsigned int gvl)
- `uint64x5xm1_t vlxseg5wuv_mask_uint64x5xm1_uint64xm1` (uint64x5xm1\_t *merge*, const unsigned long \*address, *uint64xm1\_t* index, *e64xm1\_t* mask, unsigned int gvl)
- `uint8x5xm1_t vlxseg5wuv_mask_uint8x5xm1_uint8xm1` (uint8x5xm1\_t *merge*, const unsigned char \*address, *uint8xm1\_t* index, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.127 Load 6 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlxseg6b.v'`]

**Prototypes:**

- `int16x6xml_t vlxseg6bv_int16x6xml_int16xml` (const short *\*address*, *int16xml\_t* *index*, unsigned int *gvl*)
- `int32x6xml_t vlxseg6bv_int32x6xml_int32xml` (const int *\*address*, *int32xml\_t* *index*, unsigned int *gvl*)
- `int64x6xml_t vlxseg6bv_int64x6xml_int64xml` (const long *\*address*, *int64xml\_t* *index*, unsigned int *gvl*)
- `int8x6xml_t vlxseg6bv_int8x6xml_int8xml` (const signed char *\*address*, *int8xml\_t* *index*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x6xml_t vlxseg6bv_mask_int16x6xml_int16xml` (*int16x6xml\_t merge*, const short *\*address*, *int16xml\_t* *index*, *e16xml\_t* *mask*, unsigned int *gvl*)
- `int32x6xml_t vlxseg6bv_mask_int32x6xml_int32xml` (*int32x6xml\_t merge*, const int *\*address*, *int32xml\_t* *index*, *e32xml\_t* *mask*, unsigned int *gvl*)
- `int64x6xml_t vlxseg6bv_mask_int64x6xml_int64xml` (*int64x6xml\_t merge*, const long *\*address*, *int64xml\_t* *index*, *e64xml\_t* *mask*, unsigned int *gvl*)
- `int8x6xml_t vlxseg6bv_mask_int8x6xml_int8xml` (*int8x6xml\_t merge*, const signed char *\*address*, *int8xml\_t* *index*, *e8xml\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.128 Load 6 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlxseg6bu.v'`]

**Prototypes:**

- `uint16x6xm1_t vlxseg6buu_uint16x6xm1_uint16xm1` (const unsigned short \*address, *uint16xm1\_t* index, unsigned int gvl)
- `uint32x6xm1_t vlxseg6buu_uint32x6xm1_uint32xm1` (const unsigned int \*address, *uint32xm1\_t* index, unsigned int gvl)
- `uint64x6xm1_t vlxseg6buu_uint64x6xm1_uint64xm1` (const unsigned long \*address, *uint64xm1\_t* index, unsigned int gvl)
- `uint8x6xm1_t vlxseg6buu_uint8x6xm1_uint8xm1` (const unsigned char \*address, *uint8xm1\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x6xm1_t vlxseg6buu_mask_uint16x6xm1_uint16xm1` (*uint16x6xm1\_t* merge, const unsigned short \*address, *uint16xm1\_t* index, *e16xm1\_t* mask, unsigned int gvl)
- `uint32x6xm1_t vlxseg6buu_mask_uint32x6xm1_uint32xm1` (*uint32x6xm1\_t* merge, const unsigned int \*address, *uint32xm1\_t* index, *e32xm1\_t* mask, unsigned int gvl)
- `uint64x6xm1_t vlxseg6buu_mask_uint64x6xm1_uint64xm1` (*uint64x6xm1\_t* merge, const unsigned long \*address, *uint64xm1\_t* index, *e64xm1\_t* mask, unsigned int gvl)
- `uint8x6xm1_t vlxseg6buu_mask_uint8x6xm1_uint8xm1` (*uint8x6xm1\_t* merge, const unsigned char \*address, *uint8xm1\_t* index, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.129 Load 6 contiguous element fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg6e.v`]

**Prototypes:**

- `float16x6xm1_t vlxseg6ev_float16x6xm1_float16xm1` (const float16\_t \*address, float16xm1\_t index, unsigned int gvl)
- `float32x6xm1_t vlxseg6ev_float32x6xm1_float32xm1` (const float \*address, float32xm1\_t index, unsigned int gvl)
- `float64x6xm1_t vlxseg6ev_float64x6xm1_float64xm1` (const double \*address, float64xm1\_t index, unsigned int gvl)
- `int16x6xm1_t vlxseg6ev_int16x6xm1_int16xm1` (const short \*address, int16xm1\_t index, unsigned int gvl)
- `int32x6xm1_t vlxseg6ev_int32x6xm1_int32xm1` (const int \*address, int32xm1\_t index, unsigned int gvl)
- `int64x6xm1_t vlxseg6ev_int64x6xm1_int64xm1` (const long \*address, int64xm1\_t index, unsigned int gvl)
- `int8x6xm1_t vlxseg6ev_int8x6xm1_int8xm1` (const signed char \*address, int8xm1\_t index, unsigned int gvl)
- `uint16x6xm1_t vlxseg6ev_uint16x6xm1_uint16xm1` (const unsigned short \*address, uint16xm1\_t index, unsigned int gvl)
- `uint32x6xm1_t vlxseg6ev_uint32x6xm1_uint32xm1` (const unsigned int \*address, uint32xm1\_t index, unsigned int gvl)
- `uint64x6xm1_t vlxseg6ev_uint64x6xm1_uint64xm1` (const unsigned long \*address, uint64xm1\_t index, unsigned int gvl)
- `uint8x6xm1_t vlxseg6ev_uint8x6xm1_uint8xm1` (const unsigned char \*address, uint8xm1\_t index, unsigned int gvl)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x6xm1_t vlxseg6ev_mask_float16x6xm1_float16xm1` (float16x6xm1\_t merge, const float16\_t \*address, float16xm1\_t index, e16xm1\_t mask, unsigned int gvl)
- `float32x6xm1_t vlxseg6ev_mask_float32x6xm1_float32xm1` (float32x6xm1\_t merge, const float \*address, float32xm1\_t index, e32xm1\_t mask, unsigned int gvl)
- `float64x6xm1_t vlxseg6ev_mask_float64x6xm1_float64xm1` (float64x6xm1\_t merge, const double \*address, float64xm1\_t index, e64xm1\_t mask, unsigned int gvl)
- `int16x6xm1_t vlxseg6ev_mask_int16x6xm1_int16xm1` (int16x6xm1\_t merge, const short \*address, int16xm1\_t index, e16xm1\_t mask, unsigned int gvl)

- `int32x6xml_t vlxseg6ev_mask_int32x6xml_int32xml` (`int32x6xml_t merge`, `const int *address`, `int32xml_t index`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x6xml_t vlxseg6ev_mask_int64x6xml_int64xml` (`int64x6xml_t merge`, `const long *address`, `int64xml_t index`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x6xml_t vlxseg6ev_mask_int8x6xml_int8xml` (`int8x6xml_t merge`, `const signed char *address`, `int8xml_t index`, `e8xml_t mask`, `unsigned int gvl`)
- `uint16x6xml_t vlxseg6ev_mask_uint16x6xml_uint16xml` (`uint16x6xml_t merge`, `const unsigned short *address`, `uint16xml_t index`, `e16xml_t mask`, `unsigned int gvl`)
- `uint32x6xml_t vlxseg6ev_mask_uint32x6xml_uint32xml` (`uint32x6xml_t merge`, `const unsigned int *address`, `uint32xml_t index`, `e32xml_t mask`, `unsigned int gvl`)
- `uint64x6xml_t vlxseg6ev_mask_uint64x6xml_uint64xml` (`uint64x6xml_t merge`, `const unsigned long *address`, `uint64xml_t index`, `e64xml_t mask`, `unsigned int gvl`)
- `uint8x6xml_t vlxseg6ev_mask_uint8x6xml_uint8xml` (`uint8x6xml_t merge`, `const unsigned char *address`, `uint8xml_t index`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.130 Load 6 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`'vlxseg6h.v'`]**Prototypes:**

- `int16x6xml_t vlxseg6hv_int16x6xml_int16xml` (`const short *address`, `int16xml_t index`, `unsigned int gvl`)
- `int32x6xml_t vlxseg6hv_int32x6xml_int32xml` (`const int *address`, `int32xml_t index`, `unsigned int gvl`)
- `int64x6xml_t vlxseg6hv_int64x6xml_int64xml` (`const long *address`, `int64xml_t index`, `unsigned int gvl`)
- `int8x6xml_t vlxseg6hv_int8x6xml_int8xml` (`const signed char *address`, `int8xml_t index`, `unsigned int gvl`)

**Operation:**



```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x6xml_t vlxseg6hv_mask_int16x6xml_int16xml` (`int16x6xml_t merge`, `const short *address`, `int16xml_t index`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x6xml_t vlxseg6hv_mask_int32x6xml_int32xml` (`int32x6xml_t merge`, `const int *address`, `int32xml_t index`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x6xml_t vlxseg6hv_mask_int64x6xml_int64xml` (`int64x6xml_t merge`, `const long *address`, `int64xml_t index`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x6xml_t vlxseg6hv_mask_int8x6xml_int8xml` (`int8x6xml_t merge`, `const signed char *address`, `int8xml_t index`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.131 Load 6 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlxseg6hu.v'`]

**Prototypes:**

- `uint16x6xml_t vlxseg6huv_uint16x6xml_uint16xml` (`const unsigned short *address`, `uint16xml_t index`, `unsigned int gvl`)
- `uint32x6xml_t vlxseg6huv_uint32x6xml_uint32xml` (`const unsigned int *address`, `uint32xml_t index`, `unsigned int gvl`)
- `uint64x6xml_t vlxseg6huv_uint64x6xml_uint64xml` (`const unsigned long *address`, `uint64xml_t index`, `unsigned int gvl`)
- `uint8x6xml_t vlxseg6huv_uint8x6xml_uint8xml` (`const unsigned char *address`, `uint8xml_t index`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x6xml_t vlxseg6huv_mask_uint16x6xml_uint16xml` (`uint16x6xml_t` *merge*,  
const unsigned short *\*address*, `uint16xml_t` *index*,  
`e16xml_t` *mask*, unsigned  
int *gvl*)
- `uint32x6xml_t vlxseg6huv_mask_uint32x6xml_uint32xml` (`uint32x6xml_t` *merge*,  
const unsigned int *\*address*, `uint32xml_t` *index*,  
`e32xml_t` *mask*, unsigned  
int *gvl*)
- `uint64x6xml_t vlxseg6huv_mask_uint64x6xml_uint64xml` (`uint64x6xml_t` *merge*,  
const unsigned long *\*address*, `uint64xml_t` *index*,  
`e64xml_t` *mask*, unsigned  
int *gvl*)
- `uint8x6xml_t vlxseg6huv_mask_uint8x6xml_uint8xml` (`uint8x6xml_t` *merge*, const unsigned  
char *\*address*, `uint8xml_t` *index*,  
`e8xml_t` *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.132 Load 6 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlxseg6w.v'`]**Prototypes:**

- `int16x6xml_t vlxseg6wv_int16x6xml_int16xml` (const short *\*address*, `int16xml_t` *index*, unsigned int *gvl*)
- `int32x6xml_t vlxseg6wv_int32x6xml_int32xml` (const int *\*address*, `int32xml_t` *index*, unsigned int *gvl*)
- `int64x6xml_t vlxseg6wv_int64x6xml_int64xml` (const long *\*address*, `int64xml_t` *index*, unsigned int *gvl*)
- `int8x6xml_t vlxseg6wv_int8x6xml_int8xml` (const signed char *\*address*, `int8xml_t` *index*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x6xml_t vlxseg6wv_mask_int16x6xml_int16xml` (`int16x6xml_t merge`, `const short *address`, `int16xml_t index`, `e16xml_t mask`, `unsigned int gvl`)
- `int32x6xml_t vlxseg6wv_mask_int32x6xml_int32xml` (`int32x6xml_t merge`, `const int *address`, `int32xml_t index`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x6xml_t vlxseg6wv_mask_int64x6xml_int64xml` (`int64x6xml_t merge`, `const long *address`, `int64xml_t index`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x6xml_t vlxseg6wv_mask_int8x6xml_int8xml` (`int8x6xml_t merge`, `const signed char *address`, `int8xml_t index`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.133 Load 6 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`vlxseg6wu.v`']**Prototypes:**

- `uint16x6xml_t vlxseg6wuv_uint16x6xml_uint16xml` (`const unsigned short *address`, `uint16xml_t index`, `unsigned int gvl`)
- `uint32x6xml_t vlxseg6wuv_uint32x6xml_uint32xml` (`const unsigned int *address`, `uint32xml_t index`, `unsigned int gvl`)
- `uint64x6xml_t vlxseg6wuv_uint64x6xml_uint64xml` (`const unsigned long *address`, `uint64xml_t index`, `unsigned int gvl`)
- `uint8x6xml_t vlxseg6wuv_uint8x6xml_uint8xml` (`const unsigned char *address`, `uint8xml_t index`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x6xml_t vlxseg6wuv_mask_uint16x6xml_uint16xml` (`uint16x6xml_t merge`, `const unsigned short *address`, `uint16xml_t index`, `e16xml_t mask`, `unsigned int gvl`)

- `uint32x6xml_t vlxseg6wuv_mask_uint32x6xml_uint32xml` (`uint32x6xml_t merge, const unsigned int *address, uint32xml_t index, e32xml_t mask, unsigned int gvl`)
- `uint64x6xml_t vlxseg6wuv_mask_uint64x6xml_uint64xml` (`uint64x6xml_t merge, const unsigned long *address, uint64xml_t index, e64xml_t mask, unsigned int gvl`)
- `uint8x6xml_t vlxseg6wuv_mask_uint8x6xml_uint8xml` (`uint8x6xml_t merge, const unsigned char *address, uint8xml_t index, e8xml_t mask, unsigned int gvl`)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.134 Load 7 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`'vlxseg7b.v'`]**Prototypes:**

- `int16x7xml_t vlxseg7bv_int16x7xml_int16xml` (`const short *address, int16xml_t index, unsigned int gvl`)
- `int32x7xml_t vlxseg7bv_int32x7xml_int32xml` (`const int *address, int32xml_t index, unsigned int gvl`)
- `int64x7xml_t vlxseg7bv_int64x7xml_int64xml` (`const long *address, int64xml_t index, unsigned int gvl`)
- `int8x7xml_t vlxseg7bv_int8x7xml_int8xml` (`const signed char *address, int8xml_t index, unsigned int gvl`)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x7xml_t vlxseg7bv_mask_int16x7xml_int16xml` (`int16x7xml_t merge, const short *address, int16xml_t index, e16xml_t mask, unsigned int gvl`)
- `int32x7xml_t vlxseg7bv_mask_int32x7xml_int32xml` (`int32x7xml_t merge, const int *address, int32xml_t index, e32xml_t mask, unsigned int gvl`)

- `int64x7xml_t vlxseg7bv_mask_int64x7xml_int64xml` (`int64x7xml_t merge`, `const long *address`, `int64xml_t index`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x7xml_t vlxseg7bv_mask_int8x7xml_int8xml` (`int8x7xml_t merge`, `const signed char *address`, `int8xml_t index`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.135 Load 7 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`'vlxseg7bu.v'`]**Prototypes:**

- `uint16x7xml_t vlxseg7buv_uint16x7xml_uint16xml` (`const unsigned short *address`, `uint16xml_t index`, `unsigned int gvl`)
- `uint32x7xml_t vlxseg7buv_uint32x7xml_uint32xml` (`const unsigned int *address`, `uint32xml_t index`, `unsigned int gvl`)
- `uint64x7xml_t vlxseg7buv_uint64x7xml_uint64xml` (`const unsigned long *address`, `uint64xml_t index`, `unsigned int gvl`)
- `uint8x7xml_t vlxseg7buv_uint8x7xml_uint8xml` (`const unsigned char *address`, `uint8xml_t index`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x7xml_t vlxseg7buv_mask_uint16x7xml_uint16xml` (`uint16x7xml_t merge`, `const unsigned short *address`, `uint16xml_t index`, `e16xml_t mask`, `unsigned int gvl`)
- `uint32x7xml_t vlxseg7buv_mask_uint32x7xml_uint32xml` (`uint32x7xml_t merge`, `const unsigned int *address`, `uint32xml_t index`, `e32xml_t mask`, `unsigned int gvl`)

- `uint64x7xml_t vlxseg7buvmask_uint64x7xml_uint64xml` (`uint64x7xml_t merge`, `const unsigned long *address`, `uint64xml_t index`, `e64xml_t mask`, `unsigned int gvl`)
- `uint8x7xml_t vlxseg7buvmask_uint8x7xml_uint8xml` (`uint8x7xml_t merge`, `const unsigned char *address`, `uint8xml_t index`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.136 Load 7 contiguous element fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg7e.v`]

**Prototypes:**

- `float16x7xml_t vlxseg7ev_float16x7xml_float16xml` (`const float16_t *address`, `float16xml_t index`, `unsigned int gvl`)
- `float32x7xml_t vlxseg7ev_float32x7xml_float32xml` (`const float *address`, `float32xml_t index`, `unsigned int gvl`)
- `float64x7xml_t vlxseg7ev_float64x7xml_float64xml` (`const double *address`, `float64xml_t index`, `unsigned int gvl`)
- `int16x7xml_t vlxseg7ev_int16x7xml_int16xml` (`const short *address`, `int16xml_t index`, `unsigned int gvl`)
- `int32x7xml_t vlxseg7ev_int32x7xml_int32xml` (`const int *address`, `int32xml_t index`, `unsigned int gvl`)
- `int64x7xml_t vlxseg7ev_int64x7xml_int64xml` (`const long *address`, `int64xml_t index`, `unsigned int gvl`)
- `int8x7xml_t vlxseg7ev_int8x7xml_int8xml` (`const signed char *address`, `int8xml_t index`, `unsigned int gvl`)
- `uint16x7xml_t vlxseg7ev_uint16x7xml_uint16xml` (`const unsigned short *address`, `uint16xml_t index`, `unsigned int gvl`)
- `uint32x7xml_t vlxseg7ev_uint32x7xml_uint32xml` (`const unsigned int *address`, `uint32xml_t index`, `unsigned int gvl`)
- `uint64x7xml_t vlxseg7ev_uint64x7xml_uint64xml` (`const unsigned long *address`, `uint64xml_t index`, `unsigned int gvl`)
- `uint8x7xml_t vlxseg7ev_uint8x7xml_uint8xml` (`const unsigned char *address`, `uint8xml_t index`, `unsigned int gvl`)

**Operation:**



```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x7xml_t vlxseg7ev_mask_float16x7xml_float16xml` (`float16x7xml_t merge, const float16_t *address, float16xml_t index, e16xml_t mask, unsigned int gvl`)
- `float32x7xml_t vlxseg7ev_mask_float32x7xml_float32xml` (`float32x7xml_t merge, const float *address, float32xml_t index, e32xml_t mask, unsigned int gvl`)
- `float64x7xml_t vlxseg7ev_mask_float64x7xml_float64xml` (`float64x7xml_t merge, const double *address, float64xml_t index, e64xml_t mask, unsigned int gvl`)
- `int16x7xml_t vlxseg7ev_mask_int16x7xml_int16xml` (`int16x7xml_t merge, const short *address, int16xml_t index, e16xml_t mask, unsigned int gvl`)
- `int32x7xml_t vlxseg7ev_mask_int32x7xml_int32xml` (`int32x7xml_t merge, const int *address, int32xml_t index, e32xml_t mask, unsigned int gvl`)
- `int64x7xml_t vlxseg7ev_mask_int64x7xml_int64xml` (`int64x7xml_t merge, const long *address, int64xml_t index, e64xml_t mask, unsigned int gvl`)
- `int8x7xml_t vlxseg7ev_mask_int8x7xml_int8xml` (`int8x7xml_t merge, const signed char *address, int8xml_t index, e8xml_t mask, unsigned int gvl`)
- `uint16x7xml_t vlxseg7ev_mask_uint16x7xml_uint16xml` (`uint16x7xml_t merge, const unsigned short *address, uint16xml_t index, e16xml_t mask, unsigned int gvl`)
- `uint32x7xml_t vlxseg7ev_mask_uint32x7xml_uint32xml` (`uint32x7xml_t merge, const unsigned int *address, uint32xml_t index, e32xml_t mask, unsigned int gvl`)
- `uint64x7xml_t vlxseg7ev_mask_uint64x7xml_uint64xml` (`uint64x7xml_t merge, const unsigned long *address, uint64xml_t index, e64xml_t mask, unsigned int gvl`)
- `uint8x7xml_t vlxseg7ev_mask_uint8x7xml_uint8xml` (`uint8x7xml_t merge, const unsigned char *address, uint8xml_t index, e8xml_t mask, unsigned int gvl`)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.137 Load 7 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** ['vlxseg7h.v']

**Prototypes:**

- `int16x7xml_t vlxseg7hv_int16x7xml_int16xml` (const short \*address, *int16xml\_t* index, unsigned int gvl)
- `int32x7xml_t vlxseg7hv_int32x7xml_int32xml` (const int \*address, *int32xml\_t* index, unsigned int gvl)
- `int64x7xml_t vlxseg7hv_int64x7xml_int64xml` (const long \*address, *int64xml\_t* index, unsigned int gvl)
- `int8x7xml_t vlxseg7hv_int8x7xml_int8xml` (const signed char \*address, *int8xml\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x7xml_t vlxseg7hv_mask_int16x7xml_int16xml` (int16x7xml\_t merge, const short \*address, *int16xml\_t* index, *e16xml\_t* mask, unsigned int gvl)
- `int32x7xml_t vlxseg7hv_mask_int32x7xml_int32xml` (int32x7xml\_t merge, const int \*address, *int32xml\_t* index, *e32xml\_t* mask, unsigned int gvl)
- `int64x7xml_t vlxseg7hv_mask_int64x7xml_int64xml` (int64x7xml\_t merge, const long \*address, *int64xml\_t* index, *e64xml\_t* mask, unsigned int gvl)
- `int8x7xml_t vlxseg7hv_mask_int8x7xml_int8xml` (int8x7xml\_t merge, const signed char \*address, *int8xml\_t* index, *e8xml\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
```

(continues on next page)

(continued from previous page)

```
result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

### 2.10.138 Load 7 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg7hu.v`']

**Prototypes:**

- `uint16x7xml_t vlxseg7huv_uint16x7xml_uint16xml` (const unsigned short \*address, *uint16xml\_t* index, unsigned int gvl)
- `uint32x7xml_t vlxseg7huv_uint32x7xml_uint32xml` (const unsigned int \*address, *uint32xml\_t* index, unsigned int gvl)
- `uint64x7xml_t vlxseg7huv_uint64x7xml_uint64xml` (const unsigned long \*address, *uint64xml\_t* index, unsigned int gvl)
- `uint8x7xml_t vlxseg7huv_uint8x7xml_uint8xml` (const unsigned char \*address, *uint8xml\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x7xml_t vlxseg7huv_mask_uint16x7xml_uint16xml` (uint16x7xml\_t merge, const unsigned short \*address, *uint16xml\_t* index, *e16xml\_t* mask, unsigned int gvl)
- `uint32x7xml_t vlxseg7huv_mask_uint32x7xml_uint32xml` (uint32x7xml\_t merge, const unsigned int \*address, *uint32xml\_t* index, *e32xml\_t* mask, unsigned int gvl)
- `uint64x7xml_t vlxseg7huv_mask_uint64x7xml_uint64xml` (uint64x7xml\_t merge, const unsigned long \*address, *uint64xml\_t* index, *e64xml\_t* mask, unsigned int gvl)
- `uint8x7xml_t vlxseg7huv_mask_uint8x7xml_uint8xml` (uint8x7xml\_t merge, const unsigned char \*address, *uint8xml\_t* index, *e8xml\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
```

(continues on next page)

(continued from previous page)

```

    address = address + sizeof(segment) + index[segment]
else
    result[segment] = merge[segment]
result[gvl : VLMAX] = 0

```

### 2.10.139 Load 7 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg7w.v`']

**Prototypes:**

- `int16x7xml_t vlxseg7wv_int16x7xml_int16xml` (const short \*address, *int16xml\_t* index, unsigned int gvl)
- `int32x7xml_t vlxseg7wv_int32x7xml_int32xml` (const int \*address, *int32xml\_t* index, unsigned int gvl)
- `int64x7xml_t vlxseg7wv_int64x7xml_int64xml` (const long \*address, *int64xml\_t* index, unsigned int gvl)
- `int8x7xml_t vlxseg7wv_int8x7xml_int8xml` (const signed char \*address, *int8xml\_t* index, unsigned int gvl)

**Operation:**

```

>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0

```

**Masked prototypes:**

- `int16x7xml_t vlxseg7wv_mask_int16x7xml_int16xml` (int16x7xml\_t merge, const short \*address, *int16xml\_t* index, *e16xml\_t* mask, unsigned int gvl)
- `int32x7xml_t vlxseg7wv_mask_int32x7xml_int32xml` (int32x7xml\_t merge, const int \*address, *int32xml\_t* index, *e32xml\_t* mask, unsigned int gvl)
- `int64x7xml_t vlxseg7wv_mask_int64x7xml_int64xml` (int64x7xml\_t merge, const long \*address, *int64xml\_t* index, *e64xml\_t* mask, unsigned int gvl)
- `int8x7xml_t vlxseg7wv_mask_int8x7xml_int8xml` (int8x7xml\_t merge, const signed char \*address, *int8xml\_t* index, *e8xml\_t* mask, unsigned int gvl)

**Masked operation:**

```

>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0

```

## 2.10.140 Load 7 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlxseg7wuv.v'`]

**Prototypes:**

- `uint16x7xml_t vlxseg7wuv_uint16x7xml_uint16xml` (const unsigned short \*address, *uint16xml\_t* index, unsigned int gvl)
- `uint32x7xml_t vlxseg7wuv_uint32x7xml_uint32xml` (const unsigned int \*address, *uint32xml\_t* index, unsigned int gvl)
- `uint64x7xml_t vlxseg7wuv_uint64x7xml_uint64xml` (const unsigned long \*address, *uint64xml\_t* index, unsigned int gvl)
- `uint8x7xml_t vlxseg7wuv_uint8x7xml_uint8xml` (const unsigned char \*address, *uint8xml\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x7xml_t vlxseg7wuv_mask_uint16x7xml_uint16xml` (uint16x7xml\_t *merge*, const unsigned short \*address, *uint16xml\_t* index, *e16xml\_t* mask, unsigned int gvl)
- `uint32x7xml_t vlxseg7wuv_mask_uint32x7xml_uint32xml` (uint32x7xml\_t *merge*, const unsigned int \*address, *uint32xml\_t* index, *e32xml\_t* mask, unsigned int gvl)
- `uint64x7xml_t vlxseg7wuv_mask_uint64x7xml_uint64xml` (uint64x7xml\_t *merge*, const unsigned long \*address, *uint64xml\_t* index, *e64xml\_t* mask, unsigned int gvl)
- `uint8x7xml_t vlxseg7wuv_mask_uint8x7xml_uint8xml` (uint8x7xml\_t *merge*, const unsigned char \*address, *uint8xml\_t* index, *e8xml\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.141 Load 8 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlsxseg8b.v'`]

**Prototypes:**

- `int16x8xml_t vlsxseg8bv_int16x8xml_int16xml` (const short *\*address*, *int16xml\_t* *index*, unsigned int *gvl*)
- `int32x8xml_t vlsxseg8bv_int32x8xml_int32xml` (const int *\*address*, *int32xml\_t* *index*, unsigned int *gvl*)
- `int64x8xml_t vlsxseg8bv_int64x8xml_int64xml` (const long *\*address*, *int64xml\_t* *index*, unsigned int *gvl*)
- `int8x8xml_t vlsxseg8bv_int8x8xml_int8xml` (const signed char *\*address*, *int8xml\_t* *index*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x8xml_t vlsxseg8bv_mask_int16x8xml_int16xml` (*int16x8xml\_t merge*, const short *\*address*, *int16xml\_t* *index*, *e16xml\_t* *mask*, unsigned int *gvl*)
- `int32x8xml_t vlsxseg8bv_mask_int32x8xml_int32xml` (*int32x8xml\_t merge*, const int *\*address*, *int32xml\_t* *index*, *e32xml\_t* *mask*, unsigned int *gvl*)
- `int64x8xml_t vlsxseg8bv_mask_int64x8xml_int64xml` (*int64x8xml\_t merge*, const long *\*address*, *int64xml\_t* *index*, *e64xml\_t* *mask*, unsigned int *gvl*)
- `int8x8xml_t vlsxseg8bv_mask_int8x8xml_int8xml` (*int8x8xml\_t merge*, const signed char *\*address*, *int8xml\_t* *index*, *e8xml\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.142 Load 8 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlsxseg8bu.v'`]

**Prototypes:**



- `uint16x8xm1_t vlxseg8buu_uint16x8xm1_uint16xm1` (const unsigned short \*address, *uint16xm1\_t* index, unsigned int gvl)
- `uint32x8xm1_t vlxseg8buu_uint32x8xm1_uint32xm1` (const unsigned int \*address, *uint32xm1\_t* index, unsigned int gvl)
- `uint64x8xm1_t vlxseg8buu_uint64x8xm1_uint64xm1` (const unsigned long \*address, *uint64xm1\_t* index, unsigned int gvl)
- `uint8x8xm1_t vlxseg8buu_uint8x8xm1_uint8xm1` (const unsigned char \*address, *uint8xm1\_t* index, unsigned int gvl)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x8xm1_t vlxseg8buu_mask_uint16x8xm1_uint16xm1` (uint16x8xm1\_t *merge*, const unsigned short \*address, *uint16xm1\_t* index, *e16xm1\_t* mask, unsigned int gvl)
- `uint32x8xm1_t vlxseg8buu_mask_uint32x8xm1_uint32xm1` (uint32x8xm1\_t *merge*, const unsigned int \*address, *uint32xm1\_t* index, *e32xm1\_t* mask, unsigned int gvl)
- `uint64x8xm1_t vlxseg8buu_mask_uint64x8xm1_uint64xm1` (uint64x8xm1\_t *merge*, const unsigned long \*address, *uint64xm1\_t* index, *e64xm1\_t* mask, unsigned int gvl)
- `uint8x8xm1_t vlxseg8buu_mask_uint8x8xm1_uint8xm1` (uint8x8xm1\_t *merge*, const unsigned char \*address, *uint8xm1\_t* index, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.143 Load 8 contiguous element fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg8e.v`]

**Prototypes:**

- `float16x8xml_t vlxseg8ev_float16x8xml_float16xml` (const float16\_t \*address, float16xml\_t index, unsigned int gvl)
- `float32x8xml_t vlxseg8ev_float32x8xml_float32xml` (const float \*address, float32xml\_t index, unsigned int gvl)
- `float64x8xml_t vlxseg8ev_float64x8xml_float64xml` (const double \*address, float64xml\_t index, unsigned int gvl)
- `int16x8xml_t vlxseg8ev_int16x8xml_int16xml` (const short \*address, int16xml\_t index, unsigned int gvl)
- `int32x8xml_t vlxseg8ev_int32x8xml_int32xml` (const int \*address, int32xml\_t index, unsigned int gvl)
- `int64x8xml_t vlxseg8ev_int64x8xml_int64xml` (const long \*address, int64xml\_t index, unsigned int gvl)
- `int8x8xml_t vlxseg8ev_int8x8xml_int8xml` (const signed char \*address, int8xml\_t index, unsigned int gvl)
- `uint16x8xml_t vlxseg8ev_uint16x8xml_uint16xml` (const unsigned short \*address, uint16xml\_t index, unsigned int gvl)
- `uint32x8xml_t vlxseg8ev_uint32x8xml_uint32xml` (const unsigned int \*address, uint32xml\_t index, unsigned int gvl)
- `uint64x8xml_t vlxseg8ev_uint64x8xml_uint64xml` (const unsigned long \*address, uint64xml\_t index, unsigned int gvl)
- `uint8x8xml_t vlxseg8ev_uint8x8xml_uint8xml` (const unsigned char \*address, uint8xml\_t index, unsigned int gvl)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16x8xml_t vlxseg8ev_mask_float16x8xml_float16xml` (float16x8xml\_t merge, const float16\_t \*address, float16xml\_t index, e16xml\_t mask, unsigned int gvl)
- `float32x8xml_t vlxseg8ev_mask_float32x8xml_float32xml` (float32x8xml\_t merge, const float \*address, float32xml\_t index, e32xml\_t mask, unsigned int gvl)
- `float64x8xml_t vlxseg8ev_mask_float64x8xml_float64xml` (float64x8xml\_t merge, const double \*address, float64xml\_t index, e64xml\_t mask, unsigned int gvl)
- `int16x8xml_t vlxseg8ev_mask_int16x8xml_int16xml` (int16x8xml\_t merge, const short \*address, int16xml\_t index, e16xml\_t mask, unsigned int gvl)

- `int32x8xml_t vlxseg8ev_mask_int32x8xml_int32xml` (`int32x8xml_t merge`, `const int *address`, `int32xml_t index`, `e32xml_t mask`, `unsigned int gvl`)
- `int64x8xml_t vlxseg8ev_mask_int64x8xml_int64xml` (`int64x8xml_t merge`, `const long *address`, `int64xml_t index`, `e64xml_t mask`, `unsigned int gvl`)
- `int8x8xml_t vlxseg8ev_mask_int8x8xml_int8xml` (`int8x8xml_t merge`, `const signed char *address`, `int8xml_t index`, `e8xml_t mask`, `unsigned int gvl`)
- `uint16x8xml_t vlxseg8ev_mask_uint16x8xml_uint16xml` (`uint16x8xml_t merge`, `const unsigned short *address`, `uint16xml_t index`, `e16xml_t mask`, `unsigned int gvl`)
- `uint32x8xml_t vlxseg8ev_mask_uint32x8xml_uint32xml` (`uint32x8xml_t merge`, `const unsigned int *address`, `uint32xml_t index`, `e32xml_t mask`, `unsigned int gvl`)
- `uint64x8xml_t vlxseg8ev_mask_uint64x8xml_uint64xml` (`uint64x8xml_t merge`, `const unsigned long *address`, `uint64xml_t index`, `e64xml_t mask`, `unsigned int gvl`)
- `uint8x8xml_t vlxseg8ev_mask_uint8x8xml_uint8xml` (`uint8x8xml_t merge`, `const unsigned char *address`, `uint8xml_t index`, `e8xml_t mask`, `unsigned int gvl`)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.144 Load 8 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`'vlxseg8h.v'`]**Prototypes:**

- `int16x8xml_t vlxseg8hv_int16x8xml_int16xml` (`const short *address`, `int16xml_t index`, `unsigned int gvl`)
- `int32x8xml_t vlxseg8hv_int32x8xml_int32xml` (`const int *address`, `int32xml_t index`, `unsigned int gvl`)
- `int64x8xml_t vlxseg8hv_int64x8xml_int64xml` (`const long *address`, `int64xml_t index`, `unsigned int gvl`)
- `int8x8xml_t vlxseg8hv_int8x8xml_int8xml` (`const signed char *address`, `int8xml_t index`, `unsigned int gvl`)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x8xml_t vlxseg8hv_mask_int16x8xml_int16xml` (`int16x8xml_t merge`, `const short *address`, `int16xml_t index`, `e16xml_t mask`, unsigned int `gvl`)
- `int32x8xml_t vlxseg8hv_mask_int32x8xml_int32xml` (`int32x8xml_t merge`, `const int *address`, `int32xml_t index`, `e32xml_t mask`, unsigned int `gvl`)
- `int64x8xml_t vlxseg8hv_mask_int64x8xml_int64xml` (`int64x8xml_t merge`, `const long *address`, `int64xml_t index`, `e64xml_t mask`, unsigned int `gvl`)
- `int8x8xml_t vlxseg8hv_mask_int8x8xml_int8xml` (`int8x8xml_t merge`, `const signed char *address`, `int8xml_t index`, `e8xml_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.145 Load 8 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers****Instruction:** [`'vlxseg8hu.v'`]**Prototypes:**

- `uint16x8xml_t vlxseg8huv_uint16x8xml_uint16xml` (`const unsigned short *address`, `uint16xml_t index`, unsigned int `gvl`)
- `uint32x8xml_t vlxseg8huv_uint32x8xml_uint32xml` (`const unsigned int *address`, `uint32xml_t index`, unsigned int `gvl`)
- `uint64x8xml_t vlxseg8huv_uint64x8xml_uint64xml` (`const unsigned long *address`, `uint64xml_t index`, unsigned int `gvl`)
- `uint8x8xml_t vlxseg8huv_uint8x8xml_uint8xml` (`const unsigned char *address`, `uint8xml_t index`, unsigned int `gvl`)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x8xml_t vlxseg8huv_mask_uint16x8xml_uint16xml` (`uint16x8xml_t merge, const unsigned short *address, uint16xml_t index, e16xml_t mask, unsigned int gvl`)
- `uint32x8xml_t vlxseg8huv_mask_uint32x8xml_uint32xml` (`uint32x8xml_t merge, const unsigned int *address, uint32xml_t index, e32xml_t mask, unsigned int gvl`)
- `uint64x8xml_t vlxseg8huv_mask_uint64x8xml_uint64xml` (`uint64x8xml_t merge, const unsigned long *address, uint64xml_t index, e64xml_t mask, unsigned int gvl`)
- `uint8x8xml_t vlxseg8huv_mask_uint8x8xml_uint8xml` (`uint8x8xml_t merge, const unsigned char *address, uint8xml_t index, e8xml_t mask, unsigned int gvl`)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.146 Load 8 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`'vlxseg8w.v'`]**Prototypes:**

- `int16x8xml_t vlxseg8wv_int16x8xml_int16xml` (`const short *address, int16xml_t index, unsigned int gvl`)
- `int32x8xml_t vlxseg8wv_int32x8xml_int32xml` (`const int *address, int32xml_t index, unsigned int gvl`)
- `int64x8xml_t vlxseg8wv_int64x8xml_int64xml` (`const long *address, int64xml_t index, unsigned int gvl`)
- `int8x8xml_t vlxseg8wv_int8x8xml_int8xml` (`const signed char *address, int8xml_t index, unsigned int gvl`)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `int16x8xml_t vlxseg8wv_mask_int16x8xml_int16xml` (`int16x8xml_t merge`, `const short *address`, `int16xml_t index`, `e16xml_t mask`, unsigned int `gvl`)
- `int32x8xml_t vlxseg8wv_mask_int32x8xml_int32xml` (`int32x8xml_t merge`, `const int *address`, `int32xml_t index`, `e32xml_t mask`, unsigned int `gvl`)
- `int64x8xml_t vlxseg8wv_mask_int64x8xml_int64xml` (`int64x8xml_t merge`, `const long *address`, `int64xml_t index`, `e64xml_t mask`, unsigned int `gvl`)
- `int8x8xml_t vlxseg8wv_mask_int8x8xml_int8xml` (`int8x8xml_t merge`, `const signed char *address`, `int8xml_t index`, `e8xml_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

## 2.10.147 Load 8 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers

**Instruction:** [`vlxseg8wu.v`]

**Prototypes:**

- `uint16x8xml_t vlxseg8wuv_uint16x8xml_uint16xml` (`const unsigned short *address`, `uint16xml_t index`, unsigned int `gvl`)
- `uint32x8xml_t vlxseg8wuv_uint32x8xml_uint32xml` (`const unsigned int *address`, `uint32xml_t index`, unsigned int `gvl`)
- `uint64x8xml_t vlxseg8wuv_uint64x8xml_uint64xml` (`const unsigned long *address`, `uint64xml_t index`, unsigned int `gvl`)
- `uint8x8xml_t vlxseg8wuv_uint8x8xml_uint8xml` (`const unsigned char *address`, `uint8xml_t index`, unsigned int `gvl`)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    result[segment] = load_segment(address)
    address = address + sizeof(segment) + index[segment]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16x8xml_t vlxseg8wuv_mask_uint16x8xml_uint16xml` (`uint16x8xml_t merge`, `const unsigned short *address`, `uint16xml_t index`, `e16xml_t mask`, unsigned int `gvl`)



- `uint32x8xm1_t vlxseg8wuv_mask_uint32x8xm1_uint32xm1` (`uint32x8xm1_t merge, const unsigned int *address, uint32xm1_t index, e32xm1_t mask, unsigned int gvl`)
- `uint64x8xm1_t vlxseg8wuv_mask_uint64x8xm1_uint64xm1` (`uint64x8xm1_t merge, const unsigned long *address, uint64xm1_t index, e64xm1_t mask, unsigned int gvl`)
- `uint8x8xm1_t vlxseg8wuv_mask_uint8x8xm1_uint8xm1` (`uint8x8xm1_t merge, const unsigned char *address, uint8xm1_t index, e8xm1_t mask, unsigned int gvl`)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        result[segment] = load_segment(address)
        address = address + sizeof(segment) + index[segment]
    else
        result[segment] = merge[segment]
result[gvl : VLMAX] = 0
```

**2.10.148 Store 2 contiguous 8b fields in memory from consecutively numbered vector registers****Instruction:** [`'vsseg2bv.v'`]**Prototypes:**

- `void vsseg2bv_int16x2xm1` (`short *address, int16x2xm1_t a, unsigned int gvl`)
- `void vsseg2bv_int16x2xm2` (`short *address, int16x2xm2_t a, unsigned int gvl`)
- `void vsseg2bv_int16x2xm4` (`short *address, int16x2xm4_t a, unsigned int gvl`)
- `void vsseg2bv_int32x2xm1` (`int *address, int32x2xm1_t a, unsigned int gvl`)
- `void vsseg2bv_int32x2xm2` (`int *address, int32x2xm2_t a, unsigned int gvl`)
- `void vsseg2bv_int32x2xm4` (`int *address, int32x2xm4_t a, unsigned int gvl`)
- `void vsseg2bv_int64x2xm1` (`long *address, int64x2xm1_t a, unsigned int gvl`)
- `void vsseg2bv_int64x2xm2` (`long *address, int64x2xm2_t a, unsigned int gvl`)
- `void vsseg2bv_int64x2xm4` (`long *address, int64x2xm4_t a, unsigned int gvl`)
- `void vsseg2bv_int8x2xm1` (`signed char *address, int8x2xm1_t a, unsigned int gvl`)
- `void vsseg2bv_int8x2xm2` (`signed char *address, int8x2xm2_t a, unsigned int gvl`)
- `void vsseg2bv_int8x2xm4` (`signed char *address, int8x2xm4_t a, unsigned int gvl`)
- `void vsseg2bv_uint16x2xm1` (`unsigned short *address, uint16x2xm1_t a, unsigned int gvl`)
- `void vsseg2bv_uint16x2xm2` (`unsigned short *address, uint16x2xm2_t a, unsigned int gvl`)
- `void vsseg2bv_uint16x2xm4` (`unsigned short *address, uint16x2xm4_t a, unsigned int gvl`)
- `void vsseg2bv_uint32x2xm1` (`unsigned int *address, uint32x2xm1_t a, unsigned int gvl`)

- void **vsseg2bv\_uint32x2xm2** (unsigned int \*address, uint32x2xm2\_t a, unsigned int gvl)
- void **vsseg2bv\_uint32x2xm4** (unsigned int \*address, uint32x2xm4\_t a, unsigned int gvl)
- void **vsseg2bv\_uint64x2xm1** (unsigned long \*address, uint64x2xm1\_t a, unsigned int gvl)
- void **vsseg2bv\_uint64x2xm2** (unsigned long \*address, uint64x2xm2\_t a, unsigned int gvl)
- void **vsseg2bv\_uint64x2xm4** (unsigned long \*address, uint64x2xm4\_t a, unsigned int gvl)
- void **vsseg2bv\_uint8x2xm1** (unsigned char \*address, uint8x2xm1\_t a, unsigned int gvl)
- void **vsseg2bv\_uint8x2xm2** (unsigned char \*address, uint8x2xm2\_t a, unsigned int gvl)
- void **vsseg2bv\_uint8x2xm4** (unsigned char \*address, uint8x2xm4\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg2bv\_mask\_int16x2xm1** (short \*address, int16x2xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_int16x2xm2** (short \*address, int16x2xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_int16x2xm4** (short \*address, int16x2xm4\_t a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_int32x2xm1** (int \*address, int32x2xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_int32x2xm2** (int \*address, int32x2xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_int32x2xm4** (int \*address, int32x2xm4\_t a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_int64x2xm1** (long \*address, int64x2xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_int64x2xm2** (long \*address, int64x2xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_int64x2xm4** (long \*address, int64x2xm4\_t a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_int8x2xm1** (signed char \*address, int8x2xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_int8x2xm2** (signed char \*address, int8x2xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_int8x2xm4** (signed char \*address, int8x2xm4\_t a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_uint16x2xm1** (unsigned short \*address, uint16x2xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_uint16x2xm2** (unsigned short \*address, uint16x2xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_uint16x2xm4** (unsigned short \*address, uint16x2xm4\_t a, *e16xm4\_t* mask, unsigned int gvl)

- void **vsseg2bv\_mask\_uint32x2xm1** (unsigned int \*address, uint32x2xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_uint32x2xm2** (unsigned int \*address, uint32x2xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_uint32x2xm4** (unsigned int \*address, uint32x2xm4\_t a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_uint64x2xm1** (unsigned long \*address, uint64x2xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_uint64x2xm2** (unsigned long \*address, uint64x2xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_uint64x2xm4** (unsigned long \*address, uint64x2xm4\_t a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_uint8x2xm1** (unsigned char \*address, uint8x2xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_uint8x2xm2** (unsigned char \*address, uint8x2xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsseg2bv\_mask\_uint8x2xm4** (unsigned char \*address, uint8x2xm4\_t a, *e8xm4\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.149 Store 2 contiguous element fields in memory from consecutively numbered vector registers

**Instruction:** [*'vsseg2e.v'*]

**Prototypes:**

- void **vsseg2ev\_float16x2xm1** (float16\_t \*address, float16x2xm1\_t a, unsigned int gvl)
- void **vsseg2ev\_float16x2xm2** (float16\_t \*address, float16x2xm2\_t a, unsigned int gvl)
- void **vsseg2ev\_float16x2xm4** (float16\_t \*address, float16x2xm4\_t a, unsigned int gvl)
- void **vsseg2ev\_float32x2xm1** (float \*address, float32x2xm1\_t a, unsigned int gvl)
- void **vsseg2ev\_float32x2xm2** (float \*address, float32x2xm2\_t a, unsigned int gvl)
- void **vsseg2ev\_float32x2xm4** (float \*address, float32x2xm4\_t a, unsigned int gvl)
- void **vsseg2ev\_float64x2xm1** (double \*address, float64x2xm1\_t a, unsigned int gvl)
- void **vsseg2ev\_float64x2xm2** (double \*address, float64x2xm2\_t a, unsigned int gvl)
- void **vsseg2ev\_float64x2xm4** (double \*address, float64x2xm4\_t a, unsigned int gvl)
- void **vsseg2ev\_int16x2xm1** (short \*address, int16x2xm1\_t a, unsigned int gvl)
- void **vsseg2ev\_int16x2xm2** (short \*address, int16x2xm2\_t a, unsigned int gvl)
- void **vsseg2ev\_int16x2xm4** (short \*address, int16x2xm4\_t a, unsigned int gvl)
- void **vsseg2ev\_int32x2xm1** (int \*address, int32x2xm1\_t a, unsigned int gvl)

- void **vsseg2ev\_int32x2xm2** (int \*address, int32x2xm2\_t a, unsigned int gvl)
- void **vsseg2ev\_int32x2xm4** (int \*address, int32x2xm4\_t a, unsigned int gvl)
- void **vsseg2ev\_int64x2xm1** (long \*address, int64x2xm1\_t a, unsigned int gvl)
- void **vsseg2ev\_int64x2xm2** (long \*address, int64x2xm2\_t a, unsigned int gvl)
- void **vsseg2ev\_int64x2xm4** (long \*address, int64x2xm4\_t a, unsigned int gvl)
- void **vsseg2ev\_int8x2xm1** (signed char \*address, int8x2xm1\_t a, unsigned int gvl)
- void **vsseg2ev\_int8x2xm2** (signed char \*address, int8x2xm2\_t a, unsigned int gvl)
- void **vsseg2ev\_int8x2xm4** (signed char \*address, int8x2xm4\_t a, unsigned int gvl)
- void **vsseg2ev\_uint16x2xm1** (unsigned short \*address, uint16x2xm1\_t a, unsigned int gvl)
- void **vsseg2ev\_uint16x2xm2** (unsigned short \*address, uint16x2xm2\_t a, unsigned int gvl)
- void **vsseg2ev\_uint16x2xm4** (unsigned short \*address, uint16x2xm4\_t a, unsigned int gvl)
- void **vsseg2ev\_uint32x2xm1** (unsigned int \*address, uint32x2xm1\_t a, unsigned int gvl)
- void **vsseg2ev\_uint32x2xm2** (unsigned int \*address, uint32x2xm2\_t a, unsigned int gvl)
- void **vsseg2ev\_uint32x2xm4** (unsigned int \*address, uint32x2xm4\_t a, unsigned int gvl)
- void **vsseg2ev\_uint64x2xm1** (unsigned long \*address, uint64x2xm1\_t a, unsigned int gvl)
- void **vsseg2ev\_uint64x2xm2** (unsigned long \*address, uint64x2xm2\_t a, unsigned int gvl)
- void **vsseg2ev\_uint64x2xm4** (unsigned long \*address, uint64x2xm4\_t a, unsigned int gvl)
- void **vsseg2ev\_uint8x2xm1** (unsigned char \*address, uint8x2xm1\_t a, unsigned int gvl)
- void **vsseg2ev\_uint8x2xm2** (unsigned char \*address, uint8x2xm2\_t a, unsigned int gvl)
- void **vsseg2ev\_uint8x2xm4** (unsigned char \*address, uint8x2xm4\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg2ev\_mask\_float16x2xm1** (float16\_t \*address, float16x2xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_float16x2xm2** (float16\_t \*address, float16x2xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_float16x2xm4** (float16\_t \*address, float16x2xm4\_t a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_float32x2xm1** (float \*address, float32x2xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_float32x2xm2** (float \*address, float32x2xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_float32x2xm4** (float \*address, float32x2xm4\_t a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_float64x2xm1** (double \*address, float64x2xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)

- void **vsseg2ev\_mask\_float64x2xm2** (double \*address, float64x2xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_float64x2xm4** (double \*address, float64x2xm4\_t a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_int16x2xm1** (short \*address, int16x2xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_int16x2xm2** (short \*address, int16x2xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_int16x2xm4** (short \*address, int16x2xm4\_t a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_int32x2xm1** (int \*address, int32x2xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_int32x2xm2** (int \*address, int32x2xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_int32x2xm4** (int \*address, int32x2xm4\_t a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_int64x2xm1** (long \*address, int64x2xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_int64x2xm2** (long \*address, int64x2xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_int64x2xm4** (long \*address, int64x2xm4\_t a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_int8x2xm1** (signed char \*address, int8x2xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_int8x2xm2** (signed char \*address, int8x2xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_int8x2xm4** (signed char \*address, int8x2xm4\_t a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_uint16x2xm1** (unsigned short \*address, uint16x2xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_uint16x2xm2** (unsigned short \*address, uint16x2xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_uint16x2xm4** (unsigned short \*address, uint16x2xm4\_t a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_uint32x2xm1** (unsigned int \*address, uint32x2xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_uint32x2xm2** (unsigned int \*address, uint32x2xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_uint32x2xm4** (unsigned int \*address, uint32x2xm4\_t a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_uint64x2xm1** (unsigned long \*address, uint64x2xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_uint64x2xm2** (unsigned long \*address, uint64x2xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg2ev\_mask\_uint64x2xm4** (unsigned long \*address, uint64x2xm4\_t a, *e64xm4\_t* mask, unsigned int gvl)

- void **vsseg2ev\_mask\_uint8x2xm1** (unsigned char \*address, uint8x2xm1\_t a, e8xm1\_t mask, unsigned int gvl)
- void **vsseg2ev\_mask\_uint8x2xm2** (unsigned char \*address, uint8x2xm2\_t a, e8xm2\_t mask, unsigned int gvl)
- void **vsseg2ev\_mask\_uint8x2xm4** (unsigned char \*address, uint8x2xm4\_t a, e8xm4\_t mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

### 2.10.150 Store 2 contiguous 16b fields in memory from consecutively numbered vector registers

**Instruction:** ['vsseg2hv.v']

**Prototypes:**

- void **vsseg2hv\_int16x2xm1** (short \*address, int16x2xm1\_t a, unsigned int gvl)
- void **vsseg2hv\_int16x2xm2** (short \*address, int16x2xm2\_t a, unsigned int gvl)
- void **vsseg2hv\_int16x2xm4** (short \*address, int16x2xm4\_t a, unsigned int gvl)
- void **vsseg2hv\_int32x2xm1** (int \*address, int32x2xm1\_t a, unsigned int gvl)
- void **vsseg2hv\_int32x2xm2** (int \*address, int32x2xm2\_t a, unsigned int gvl)
- void **vsseg2hv\_int32x2xm4** (int \*address, int32x2xm4\_t a, unsigned int gvl)
- void **vsseg2hv\_int64x2xm1** (long \*address, int64x2xm1\_t a, unsigned int gvl)
- void **vsseg2hv\_int64x2xm2** (long \*address, int64x2xm2\_t a, unsigned int gvl)
- void **vsseg2hv\_int64x2xm4** (long \*address, int64x2xm4\_t a, unsigned int gvl)
- void **vsseg2hv\_int8x2xm1** (signed char \*address, int8x2xm1\_t a, unsigned int gvl)
- void **vsseg2hv\_int8x2xm2** (signed char \*address, int8x2xm2\_t a, unsigned int gvl)
- void **vsseg2hv\_int8x2xm4** (signed char \*address, int8x2xm4\_t a, unsigned int gvl)
- void **vsseg2hv\_uint16x2xm1** (unsigned short \*address, uint16x2xm1\_t a, unsigned int gvl)
- void **vsseg2hv\_uint16x2xm2** (unsigned short \*address, uint16x2xm2\_t a, unsigned int gvl)
- void **vsseg2hv\_uint16x2xm4** (unsigned short \*address, uint16x2xm4\_t a, unsigned int gvl)
- void **vsseg2hv\_uint32x2xm1** (unsigned int \*address, uint32x2xm1\_t a, unsigned int gvl)
- void **vsseg2hv\_uint32x2xm2** (unsigned int \*address, uint32x2xm2\_t a, unsigned int gvl)
- void **vsseg2hv\_uint32x2xm4** (unsigned int \*address, uint32x2xm4\_t a, unsigned int gvl)
- void **vsseg2hv\_uint64x2xm1** (unsigned long \*address, uint64x2xm1\_t a, unsigned int gvl)
- void **vsseg2hv\_uint64x2xm2** (unsigned long \*address, uint64x2xm2\_t a, unsigned int gvl)
- void **vsseg2hv\_uint64x2xm4** (unsigned long \*address, uint64x2xm4\_t a, unsigned int gvl)
- void **vsseg2hv\_uint8x2xm1** (unsigned char \*address, uint8x2xm1\_t a, unsigned int gvl)



- void **vsseg2hv\_uint8x2xm2** (unsigned char \**address*, uint8x2xm2\_t *a*, unsigned int *gvl*)
- void **vsseg2hv\_uint8x2xm4** (unsigned char \**address*, uint8x2xm4\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg2hv\_mask\_int16x2xm1** (short \**address*, int16x2xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_int16x2xm2** (short \**address*, int16x2xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_int16x2xm4** (short \**address*, int16x2xm4\_t *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_int32x2xm1** (int \**address*, int32x2xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_int32x2xm2** (int \**address*, int32x2xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_int32x2xm4** (int \**address*, int32x2xm4\_t *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_int64x2xm1** (long \**address*, int64x2xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_int64x2xm2** (long \**address*, int64x2xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_int64x2xm4** (long \**address*, int64x2xm4\_t *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_int8x2xm1** (signed char \**address*, int8x2xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_int8x2xm2** (signed char \**address*, int8x2xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_int8x2xm4** (signed char \**address*, int8x2xm4\_t *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_uint16x2xm1** (unsigned short \**address*, uint16x2xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_uint16x2xm2** (unsigned short \**address*, uint16x2xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_uint16x2xm4** (unsigned short \**address*, uint16x2xm4\_t *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_uint32x2xm1** (unsigned int \**address*, uint32x2xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_uint32x2xm2** (unsigned int \**address*, uint32x2xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_uint32x2xm4** (unsigned int \**address*, uint32x2xm4\_t *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vsseg2hv\_mask\_uint64x2xm1** (unsigned long \**address*, uint64x2xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)

- void **vsseg2hv\_mask\_uint64x2xm2** (unsigned long \*address, uint64x2xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsseg2hv\_mask\_uint64x2xm4** (unsigned long \*address, uint64x2xm4\_t a, e64xm4\_t mask, unsigned int gvl)
- void **vsseg2hv\_mask\_uint8x2xm1** (unsigned char \*address, uint8x2xm1\_t a, e8xm1\_t mask, unsigned int gvl)
- void **vsseg2hv\_mask\_uint8x2xm2** (unsigned char \*address, uint8x2xm2\_t a, e8xm2\_t mask, unsigned int gvl)
- void **vsseg2hv\_mask\_uint8x2xm4** (unsigned char \*address, uint8x2xm4\_t a, e8xm4\_t mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

### 2.10.151 Store 2 contiguous 32b fields in memory from consecutively numbered vector registers

**Instruction:** ['vsseg2w.v']

**Prototypes:**

- void **vsseg2wv\_int16x2xm1** (short \*address, int16x2xm1\_t a, unsigned int gvl)
- void **vsseg2wv\_int16x2xm2** (short \*address, int16x2xm2\_t a, unsigned int gvl)
- void **vsseg2wv\_int16x2xm4** (short \*address, int16x2xm4\_t a, unsigned int gvl)
- void **vsseg2wv\_int32x2xm1** (int \*address, int32x2xm1\_t a, unsigned int gvl)
- void **vsseg2wv\_int32x2xm2** (int \*address, int32x2xm2\_t a, unsigned int gvl)
- void **vsseg2wv\_int32x2xm4** (int \*address, int32x2xm4\_t a, unsigned int gvl)
- void **vsseg2wv\_int64x2xm1** (long \*address, int64x2xm1\_t a, unsigned int gvl)
- void **vsseg2wv\_int64x2xm2** (long \*address, int64x2xm2\_t a, unsigned int gvl)
- void **vsseg2wv\_int64x2xm4** (long \*address, int64x2xm4\_t a, unsigned int gvl)
- void **vsseg2wv\_int8x2xm1** (signed char \*address, int8x2xm1\_t a, unsigned int gvl)
- void **vsseg2wv\_int8x2xm2** (signed char \*address, int8x2xm2\_t a, unsigned int gvl)
- void **vsseg2wv\_int8x2xm4** (signed char \*address, int8x2xm4\_t a, unsigned int gvl)
- void **vsseg2wv\_uint16x2xm1** (unsigned short \*address, uint16x2xm1\_t a, unsigned int gvl)
- void **vsseg2wv\_uint16x2xm2** (unsigned short \*address, uint16x2xm2\_t a, unsigned int gvl)
- void **vsseg2wv\_uint16x2xm4** (unsigned short \*address, uint16x2xm4\_t a, unsigned int gvl)
- void **vsseg2wv\_uint32x2xm1** (unsigned int \*address, uint32x2xm1\_t a, unsigned int gvl)
- void **vsseg2wv\_uint32x2xm2** (unsigned int \*address, uint32x2xm2\_t a, unsigned int gvl)
- void **vsseg2wv\_uint32x2xm4** (unsigned int \*address, uint32x2xm4\_t a, unsigned int gvl)
- void **vsseg2wv\_uint64x2xm1** (unsigned long \*address, uint64x2xm1\_t a, unsigned int gvl)

- void **vsseg2wv\_uint64x2xm2** (unsigned long \*address, uint64x2xm2\_t a, unsigned int gvl)
- void **vsseg2wv\_uint64x2xm4** (unsigned long \*address, uint64x2xm4\_t a, unsigned int gvl)
- void **vsseg2wv\_uint8x2xm1** (unsigned char \*address, uint8x2xm1\_t a, unsigned int gvl)
- void **vsseg2wv\_uint8x2xm2** (unsigned char \*address, uint8x2xm2\_t a, unsigned int gvl)
- void **vsseg2wv\_uint8x2xm4** (unsigned char \*address, uint8x2xm4\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg2wv\_mask\_int16x2xm1** (short \*address, int16x2xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_int16x2xm2** (short \*address, int16x2xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_int16x2xm4** (short \*address, int16x2xm4\_t a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_int32x2xm1** (int \*address, int32x2xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_int32x2xm2** (int \*address, int32x2xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_int32x2xm4** (int \*address, int32x2xm4\_t a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_int64x2xm1** (long \*address, int64x2xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_int64x2xm2** (long \*address, int64x2xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_int64x2xm4** (long \*address, int64x2xm4\_t a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_int8x2xm1** (signed char \*address, int8x2xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_int8x2xm2** (signed char \*address, int8x2xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_int8x2xm4** (signed char \*address, int8x2xm4\_t a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_uint16x2xm1** (unsigned short \*address, uint16x2xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_uint16x2xm2** (unsigned short \*address, uint16x2xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_uint16x2xm4** (unsigned short \*address, uint16x2xm4\_t a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_uint32x2xm1** (unsigned int \*address, uint32x2xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg2wv\_mask\_uint32x2xm2** (unsigned int \*address, uint32x2xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)

- void **vsseg2wv\_mask\_uint32x2xm4** (unsigned int \*address, uint32x2xm4\_t a, e32xm4\_t mask, unsigned int gvl)
- void **vsseg2wv\_mask\_uint64x2xm1** (unsigned long \*address, uint64x2xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsseg2wv\_mask\_uint64x2xm2** (unsigned long \*address, uint64x2xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsseg2wv\_mask\_uint64x2xm4** (unsigned long \*address, uint64x2xm4\_t a, e64xm4\_t mask, unsigned int gvl)
- void **vsseg2wv\_mask\_uint8x2xm1** (unsigned char \*address, uint8x2xm1\_t a, e8xm1\_t mask, unsigned int gvl)
- void **vsseg2wv\_mask\_uint8x2xm2** (unsigned char \*address, uint8x2xm2\_t a, e8xm2\_t mask, unsigned int gvl)
- void **vsseg2wv\_mask\_uint8x2xm4** (unsigned char \*address, uint8x2xm4\_t a, e8xm4\_t mask, unsigned int gvl)

Masked operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.152 Store 3 contiguous 8b fields in memory from consecutively numbered vector registers

Instruction: ['vsseg3bv.v']

Prototypes:

- void **vsseg3bv\_int16x3xm1** (short \*address, int16x3xm1\_t a, unsigned int gvl)
- void **vsseg3bv\_int16x3xm2** (short \*address, int16x3xm2\_t a, unsigned int gvl)
- void **vsseg3bv\_int32x3xm1** (int \*address, int32x3xm1\_t a, unsigned int gvl)
- void **vsseg3bv\_int32x3xm2** (int \*address, int32x3xm2\_t a, unsigned int gvl)
- void **vsseg3bv\_int64x3xm1** (long \*address, int64x3xm1\_t a, unsigned int gvl)
- void **vsseg3bv\_int64x3xm2** (long \*address, int64x3xm2\_t a, unsigned int gvl)
- void **vsseg3bv\_int8x3xm1** (signed char \*address, int8x3xm1\_t a, unsigned int gvl)
- void **vsseg3bv\_int8x3xm2** (signed char \*address, int8x3xm2\_t a, unsigned int gvl)
- void **vsseg3bv\_uint16x3xm1** (unsigned short \*address, uint16x3xm1\_t a, unsigned int gvl)
- void **vsseg3bv\_uint16x3xm2** (unsigned short \*address, uint16x3xm2\_t a, unsigned int gvl)
- void **vsseg3bv\_uint32x3xm1** (unsigned int \*address, uint32x3xm1\_t a, unsigned int gvl)
- void **vsseg3bv\_uint32x3xm2** (unsigned int \*address, uint32x3xm2\_t a, unsigned int gvl)
- void **vsseg3bv\_uint64x3xm1** (unsigned long \*address, uint64x3xm1\_t a, unsigned int gvl)
- void **vsseg3bv\_uint64x3xm2** (unsigned long \*address, uint64x3xm2\_t a, unsigned int gvl)
- void **vsseg3bv\_uint8x3xm1** (unsigned char \*address, uint8x3xm1\_t a, unsigned int gvl)
- void **vsseg3bv\_uint8x3xm2** (unsigned char \*address, uint8x3xm2\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg3bv\_mask\_int16x3xm1** (short \*address, int16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_int16x3xm2** (short \*address, int16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_int32x3xm1** (int \*address, int32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_int32x3xm2** (int \*address, int32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_int64x3xm1** (long \*address, int64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_int64x3xm2** (long \*address, int64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_int8x3xm1** (signed char \*address, int8x3xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_int8x3xm2** (signed char \*address, int8x3xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_uint16x3xm1** (unsigned short \*address, uint16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_uint16x3xm2** (unsigned short \*address, uint16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_uint32x3xm1** (unsigned int \*address, uint32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_uint32x3xm2** (unsigned int \*address, uint32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_uint64x3xm1** (unsigned long \*address, uint64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_uint64x3xm2** (unsigned long \*address, uint64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_uint8x3xm1** (unsigned char \*address, uint8x3xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg3bv\_mask\_uint8x3xm2** (unsigned char \*address, uint8x3xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.153 Store 3 contiguous element fields in memory from consecutively numbered vector registers

**Instruction:** [`'vsseg3e.v'`]

**Prototypes:**

- void **vsseg3ev\_float16x3xm1** (float16\_t \*address, float16x3xm1\_t a, unsigned int gvl)
- void **vsseg3ev\_float16x3xm2** (float16\_t \*address, float16x3xm2\_t a, unsigned int gvl)
- void **vsseg3ev\_float32x3xm1** (float \*address, float32x3xm1\_t a, unsigned int gvl)
- void **vsseg3ev\_float32x3xm2** (float \*address, float32x3xm2\_t a, unsigned int gvl)
- void **vsseg3ev\_float64x3xm1** (double \*address, float64x3xm1\_t a, unsigned int gvl)
- void **vsseg3ev\_float64x3xm2** (double \*address, float64x3xm2\_t a, unsigned int gvl)
- void **vsseg3ev\_int16x3xm1** (short \*address, int16x3xm1\_t a, unsigned int gvl)
- void **vsseg3ev\_int16x3xm2** (short \*address, int16x3xm2\_t a, unsigned int gvl)
- void **vsseg3ev\_int32x3xm1** (int \*address, int32x3xm1\_t a, unsigned int gvl)
- void **vsseg3ev\_int32x3xm2** (int \*address, int32x3xm2\_t a, unsigned int gvl)
- void **vsseg3ev\_int64x3xm1** (long \*address, int64x3xm1\_t a, unsigned int gvl)
- void **vsseg3ev\_int64x3xm2** (long \*address, int64x3xm2\_t a, unsigned int gvl)
- void **vsseg3ev\_int8x3xm1** (signed char \*address, int8x3xm1\_t a, unsigned int gvl)
- void **vsseg3ev\_int8x3xm2** (signed char \*address, int8x3xm2\_t a, unsigned int gvl)
- void **vsseg3ev\_uint16x3xm1** (unsigned short \*address, uint16x3xm1\_t a, unsigned int gvl)
- void **vsseg3ev\_uint16x3xm2** (unsigned short \*address, uint16x3xm2\_t a, unsigned int gvl)
- void **vsseg3ev\_uint32x3xm1** (unsigned int \*address, uint32x3xm1\_t a, unsigned int gvl)
- void **vsseg3ev\_uint32x3xm2** (unsigned int \*address, uint32x3xm2\_t a, unsigned int gvl)
- void **vsseg3ev\_uint64x3xm1** (unsigned long \*address, uint64x3xm1\_t a, unsigned int gvl)
- void **vsseg3ev\_uint64x3xm2** (unsigned long \*address, uint64x3xm2\_t a, unsigned int gvl)
- void **vsseg3ev\_uint8x3xm1** (unsigned char \*address, uint8x3xm1\_t a, unsigned int gvl)
- void **vsseg3ev\_uint8x3xm2** (unsigned char \*address, uint8x3xm2\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg3ev\_mask\_float16x3xm1** (float16\_t \*address, float16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_float16x3xm2** (float16\_t \*address, float16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_float32x3xm1** (float \*address, float32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)



- void **vsseg3ev\_mask\_float32x3xm2** (float \*address, float32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_float64x3xm1** (double \*address, float64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_float64x3xm2** (double \*address, float64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_int16x3xm1** (short \*address, int16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_int16x3xm2** (short \*address, int16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_int32x3xm1** (int \*address, int32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_int32x3xm2** (int \*address, int32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_int64x3xm1** (long \*address, int64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_int64x3xm2** (long \*address, int64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_int8x3xm1** (signed char \*address, int8x3xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_int8x3xm2** (signed char \*address, int8x3xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_uint16x3xm1** (unsigned short \*address, uint16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_uint16x3xm2** (unsigned short \*address, uint16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_uint32x3xm1** (unsigned int \*address, uint32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_uint32x3xm2** (unsigned int \*address, uint32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_uint64x3xm1** (unsigned long \*address, uint64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_uint64x3xm2** (unsigned long \*address, uint64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_uint8x3xm1** (unsigned char \*address, uint8x3xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg3ev\_mask\_uint8x3xm2** (unsigned char \*address, uint8x3xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.154 Store 3 contiguous 16b fields in memory from consecutively numbered vector registers

**Instruction:** [`'vsseg3h.v'`]

**Prototypes:**

- void **vsseg3hv\_int16x3xm1** (short *\*address*, int16x3xm1\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_int16x3xm2** (short *\*address*, int16x3xm2\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_int32x3xm1** (int *\*address*, int32x3xm1\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_int32x3xm2** (int *\*address*, int32x3xm2\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_int64x3xm1** (long *\*address*, int64x3xm1\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_int64x3xm2** (long *\*address*, int64x3xm2\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_int8x3xm1** (signed char *\*address*, int8x3xm1\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_int8x3xm2** (signed char *\*address*, int8x3xm2\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_uint16x3xm1** (unsigned short *\*address*, uint16x3xm1\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_uint16x3xm2** (unsigned short *\*address*, uint16x3xm2\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_uint32x3xm1** (unsigned int *\*address*, uint32x3xm1\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_uint32x3xm2** (unsigned int *\*address*, uint32x3xm2\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_uint64x3xm1** (unsigned long *\*address*, uint64x3xm1\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_uint64x3xm2** (unsigned long *\*address*, uint64x3xm2\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_uint8x3xm1** (unsigned char *\*address*, uint8x3xm1\_t *a*, unsigned int *gvl*)
- void **vsseg3hv\_uint8x3xm2** (unsigned char *\*address*, uint8x3xm2\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg3hv\_mask\_int16x3xm1** (short *\*address*, int16x3xm1\_t *a*, *e16xm1\_t mask*, unsigned int *gvl*)
- void **vsseg3hv\_mask\_int16x3xm2** (short *\*address*, int16x3xm2\_t *a*, *e16xm2\_t mask*, unsigned int *gvl*)
- void **vsseg3hv\_mask\_int32x3xm1** (int *\*address*, int32x3xm1\_t *a*, *e32xm1\_t mask*, unsigned int *gvl*)
- void **vsseg3hv\_mask\_int32x3xm2** (int *\*address*, int32x3xm2\_t *a*, *e32xm2\_t mask*, unsigned int *gvl*)
- void **vsseg3hv\_mask\_int64x3xm1** (long *\*address*, int64x3xm1\_t *a*, *e64xm1\_t mask*, unsigned int *gvl*)
- void **vsseg3hv\_mask\_int64x3xm2** (long *\*address*, int64x3xm2\_t *a*, *e64xm2\_t mask*, unsigned int *gvl*)
- void **vsseg3hv\_mask\_int8x3xm1** (signed char *\*address*, int8x3xm1\_t *a*, *e8xm1\_t mask*, unsigned int *gvl*)

- void **vsseg3hv\_mask\_int8x3xm2** (signed char \*address, int8x3xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsseg3hv\_mask\_uint16x3xm1** (unsigned short \*address, uint16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg3hv\_mask\_uint16x3xm2** (unsigned short \*address, uint16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg3hv\_mask\_uint32x3xm1** (unsigned int \*address, uint32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg3hv\_mask\_uint32x3xm2** (unsigned int \*address, uint32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg3hv\_mask\_uint64x3xm1** (unsigned long \*address, uint64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg3hv\_mask\_uint64x3xm2** (unsigned long \*address, uint64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg3hv\_mask\_uint8x3xm1** (unsigned char \*address, uint8x3xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg3hv\_mask\_uint8x3xm2** (unsigned char \*address, uint8x3xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.155 Store 3 contiguous 32b fields in memory from consecutively numbered vector registers

**Instruction:** [*'vsseg3w.v'*]

**Prototypes:**

- void **vsseg3wv\_int16x3xm1** (short \*address, int16x3xm1\_t a, unsigned int gvl)
- void **vsseg3wv\_int16x3xm2** (short \*address, int16x3xm2\_t a, unsigned int gvl)
- void **vsseg3wv\_int32x3xm1** (int \*address, int32x3xm1\_t a, unsigned int gvl)
- void **vsseg3wv\_int32x3xm2** (int \*address, int32x3xm2\_t a, unsigned int gvl)
- void **vsseg3wv\_int64x3xm1** (long \*address, int64x3xm1\_t a, unsigned int gvl)
- void **vsseg3wv\_int64x3xm2** (long \*address, int64x3xm2\_t a, unsigned int gvl)
- void **vsseg3wv\_int8x3xm1** (signed char \*address, int8x3xm1\_t a, unsigned int gvl)
- void **vsseg3wv\_int8x3xm2** (signed char \*address, int8x3xm2\_t a, unsigned int gvl)
- void **vsseg3wv\_uint16x3xm1** (unsigned short \*address, uint16x3xm1\_t a, unsigned int gvl)
- void **vsseg3wv\_uint16x3xm2** (unsigned short \*address, uint16x3xm2\_t a, unsigned int gvl)
- void **vsseg3wv\_uint32x3xm1** (unsigned int \*address, uint32x3xm1\_t a, unsigned int gvl)
- void **vsseg3wv\_uint32x3xm2** (unsigned int \*address, uint32x3xm2\_t a, unsigned int gvl)
- void **vsseg3wv\_uint64x3xm1** (unsigned long \*address, uint64x3xm1\_t a, unsigned int gvl)

- void **vsseg3wv\_uint64x3xm2** (unsigned long \*address, uint64x3xm2\_t a, unsigned int gvl)
- void **vsseg3wv\_uint8x3xm1** (unsigned char \*address, uint8x3xm1\_t a, unsigned int gvl)
- void **vsseg3wv\_uint8x3xm2** (unsigned char \*address, uint8x3xm2\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg3wv\_mask\_int16x3xm1** (short \*address, int16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_int16x3xm2** (short \*address, int16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_int32x3xm1** (int \*address, int32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_int32x3xm2** (int \*address, int32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_int64x3xm1** (long \*address, int64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_int64x3xm2** (long \*address, int64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_int8x3xm1** (signed char \*address, int8x3xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_int8x3xm2** (signed char \*address, int8x3xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_uint16x3xm1** (unsigned short \*address, uint16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_uint16x3xm2** (unsigned short \*address, uint16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_uint32x3xm1** (unsigned int \*address, uint32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_uint32x3xm2** (unsigned int \*address, uint32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_uint64x3xm1** (unsigned long \*address, uint64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_uint64x3xm2** (unsigned long \*address, uint64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_uint8x3xm1** (unsigned char \*address, uint8x3xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg3wv\_mask\_uint8x3xm2** (unsigned char \*address, uint8x3xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
      if mask[segment] then
```

(continues on next page)

(continued from previous page)

```
store_segment(address, a[segment])
address = address + sizeof(segment)
```

## 2.10.156 Store 4 contiguous 8b fields in memory from consecutively numbered vector registers

**Instruction:** [`vsseg4b.v`]

**Prototypes:**

- void **vsseg4bv\_int16x4xm1** (short *\*address*, int16x4xm1\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_int16x4xm2** (short *\*address*, int16x4xm2\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_int32x4xm1** (int *\*address*, int32x4xm1\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_int32x4xm2** (int *\*address*, int32x4xm2\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_int64x4xm1** (long *\*address*, int64x4xm1\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_int64x4xm2** (long *\*address*, int64x4xm2\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_int8x4xm1** (signed char *\*address*, int8x4xm1\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_int8x4xm2** (signed char *\*address*, int8x4xm2\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_uint16x4xm1** (unsigned short *\*address*, uint16x4xm1\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_uint16x4xm2** (unsigned short *\*address*, uint16x4xm2\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_uint32x4xm1** (unsigned int *\*address*, uint32x4xm1\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_uint32x4xm2** (unsigned int *\*address*, uint32x4xm2\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_uint64x4xm1** (unsigned long *\*address*, uint64x4xm1\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_uint64x4xm2** (unsigned long *\*address*, uint64x4xm2\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_uint8x4xm1** (unsigned char *\*address*, uint8x4xm1\_t *a*, unsigned int *gvl*)
- void **vsseg4bv\_uint8x4xm2** (unsigned char *\*address*, uint8x4xm2\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg4bv\_mask\_int16x4xm1** (short *\*address*, int16x4xm1\_t *a*, *e16xm1\_t mask*, unsigned int *gvl*)
- void **vsseg4bv\_mask\_int16x4xm2** (short *\*address*, int16x4xm2\_t *a*, *e16xm2\_t mask*, unsigned int *gvl*)
- void **vsseg4bv\_mask\_int32x4xm1** (int *\*address*, int32x4xm1\_t *a*, *e32xm1\_t mask*, unsigned int *gvl*)
- void **vsseg4bv\_mask\_int32x4xm2** (int *\*address*, int32x4xm2\_t *a*, *e32xm2\_t mask*, unsigned int *gvl*)
- void **vsseg4bv\_mask\_int64x4xm1** (long *\*address*, int64x4xm1\_t *a*, *e64xm1\_t mask*, unsigned int *gvl*)

- void **vsseg4bv\_mask\_int64x4xm2** (long \*address, int64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg4bv\_mask\_int8x4xm1** (signed char \*address, int8x4xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg4bv\_mask\_int8x4xm2** (signed char \*address, int8x4xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsseg4bv\_mask\_uint16x4xm1** (unsigned short \*address, uint16x4xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg4bv\_mask\_uint16x4xm2** (unsigned short \*address, uint16x4xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg4bv\_mask\_uint32x4xm1** (unsigned int \*address, uint32x4xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg4bv\_mask\_uint32x4xm2** (unsigned int \*address, uint32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg4bv\_mask\_uint64x4xm1** (unsigned long \*address, uint64x4xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg4bv\_mask\_uint64x4xm2** (unsigned long \*address, uint64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg4bv\_mask\_uint8x4xm1** (unsigned char \*address, uint8x4xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg4bv\_mask\_uint8x4xm2** (unsigned char \*address, uint8x4xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.157 Store 4 contiguous element fields in memory from consecutively numbered vector registers

**Instruction:** [*vsseg4e.v*]

**Prototypes:**

- void **vsseg4ev\_float16x4xm1** (float16\_t \*address, float16x4xm1\_t a, unsigned int gvl)
- void **vsseg4ev\_float16x4xm2** (float16\_t \*address, float16x4xm2\_t a, unsigned int gvl)
- void **vsseg4ev\_float32x4xm1** (float \*address, float32x4xm1\_t a, unsigned int gvl)
- void **vsseg4ev\_float32x4xm2** (float \*address, float32x4xm2\_t a, unsigned int gvl)
- void **vsseg4ev\_float64x4xm1** (double \*address, float64x4xm1\_t a, unsigned int gvl)
- void **vsseg4ev\_float64x4xm2** (double \*address, float64x4xm2\_t a, unsigned int gvl)
- void **vsseg4ev\_int16x4xm1** (short \*address, int16x4xm1\_t a, unsigned int gvl)
- void **vsseg4ev\_int16x4xm2** (short \*address, int16x4xm2\_t a, unsigned int gvl)
- void **vsseg4ev\_int32x4xm1** (int \*address, int32x4xm1\_t a, unsigned int gvl)
- void **vsseg4ev\_int32x4xm2** (int \*address, int32x4xm2\_t a, unsigned int gvl)



- void **vsseg4ev\_int64x4xm1** (long \*address, int64x4xm1\_t a, unsigned int gvl)
- void **vsseg4ev\_int64x4xm2** (long \*address, int64x4xm2\_t a, unsigned int gvl)
- void **vsseg4ev\_int8x4xm1** (signed char \*address, int8x4xm1\_t a, unsigned int gvl)
- void **vsseg4ev\_int8x4xm2** (signed char \*address, int8x4xm2\_t a, unsigned int gvl)
- void **vsseg4ev\_uint16x4xm1** (unsigned short \*address, uint16x4xm1\_t a, unsigned int gvl)
- void **vsseg4ev\_uint16x4xm2** (unsigned short \*address, uint16x4xm2\_t a, unsigned int gvl)
- void **vsseg4ev\_uint32x4xm1** (unsigned int \*address, uint32x4xm1\_t a, unsigned int gvl)
- void **vsseg4ev\_uint32x4xm2** (unsigned int \*address, uint32x4xm2\_t a, unsigned int gvl)
- void **vsseg4ev\_uint64x4xm1** (unsigned long \*address, uint64x4xm1\_t a, unsigned int gvl)
- void **vsseg4ev\_uint64x4xm2** (unsigned long \*address, uint64x4xm2\_t a, unsigned int gvl)
- void **vsseg4ev\_uint8x4xm1** (unsigned char \*address, uint8x4xm1\_t a, unsigned int gvl)
- void **vsseg4ev\_uint8x4xm2** (unsigned char \*address, uint8x4xm2\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg4ev\_mask\_float16x4xm1** (float16\_t \*address, float16x4xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_float16x4xm2** (float16\_t \*address, float16x4xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_float32x4xm1** (float \*address, float32x4xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_float32x4xm2** (float \*address, float32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_float64x4xm1** (double \*address, float64x4xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_float64x4xm2** (double \*address, float64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_int16x4xm1** (short \*address, int16x4xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_int16x4xm2** (short \*address, int16x4xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_int32x4xm1** (int \*address, int32x4xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_int32x4xm2** (int \*address, int32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_int64x4xm1** (long \*address, int64x4xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_int64x4xm2** (long \*address, int64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)

- void **vsseg4ev\_mask\_int8x4xm1** (signed char \*address, int8x4xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_int8x4xm2** (signed char \*address, int8x4xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_uint16x4xm1** (unsigned short \*address, uint16x4xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_uint16x4xm2** (unsigned short \*address, uint16x4xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_uint32x4xm1** (unsigned int \*address, uint32x4xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_uint32x4xm2** (unsigned int \*address, uint32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_uint64x4xm1** (unsigned long \*address, uint64x4xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_uint64x4xm2** (unsigned long \*address, uint64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_uint8x4xm1** (unsigned char \*address, uint8x4xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg4ev\_mask\_uint8x4xm2** (unsigned char \*address, uint8x4xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for segment(4 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.158 Store 4 contiguous 16b fields in memory from consecutively numbered vector registers

Instruction: ['vsseg4h.v']

Prototypes:

- void **vsseg4hv\_int16x4xm1** (short \*address, int16x4xm1\_t a, unsigned int gvl)
- void **vsseg4hv\_int16x4xm2** (short \*address, int16x4xm2\_t a, unsigned int gvl)
- void **vsseg4hv\_int32x4xm1** (int \*address, int32x4xm1\_t a, unsigned int gvl)
- void **vsseg4hv\_int32x4xm2** (int \*address, int32x4xm2\_t a, unsigned int gvl)
- void **vsseg4hv\_int64x4xm1** (long \*address, int64x4xm1\_t a, unsigned int gvl)
- void **vsseg4hv\_int64x4xm2** (long \*address, int64x4xm2\_t a, unsigned int gvl)
- void **vsseg4hv\_int8x4xm1** (signed char \*address, int8x4xm1\_t a, unsigned int gvl)
- void **vsseg4hv\_int8x4xm2** (signed char \*address, int8x4xm2\_t a, unsigned int gvl)
- void **vsseg4hv\_uint16x4xm1** (unsigned short \*address, uint16x4xm1\_t a, unsigned int gvl)
- void **vsseg4hv\_uint16x4xm2** (unsigned short \*address, uint16x4xm2\_t a, unsigned int gvl)
- void **vsseg4hv\_uint32x4xm1** (unsigned int \*address, uint32x4xm1\_t a, unsigned int gvl)

- void **vsseg4hv\_uint32x4xm2** (unsigned int \*address, uint32x4xm2\_t a, unsigned int gvl)
- void **vsseg4hv\_uint64x4xm1** (unsigned long \*address, uint64x4xm1\_t a, unsigned int gvl)
- void **vsseg4hv\_uint64x4xm2** (unsigned long \*address, uint64x4xm2\_t a, unsigned int gvl)
- void **vsseg4hv\_uint8x4xm1** (unsigned char \*address, uint8x4xm1\_t a, unsigned int gvl)
- void **vsseg4hv\_uint8x4xm2** (unsigned char \*address, uint8x4xm2\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg4hv\_mask\_int16x4xm1** (short \*address, int16x4xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_int16x4xm2** (short \*address, int16x4xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_int32x4xm1** (int \*address, int32x4xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_int32x4xm2** (int \*address, int32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_int64x4xm1** (long \*address, int64x4xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_int64x4xm2** (long \*address, int64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_int8x4xm1** (signed char \*address, int8x4xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_int8x4xm2** (signed char \*address, int8x4xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_uint16x4xm1** (unsigned short \*address, uint16x4xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_uint16x4xm2** (unsigned short \*address, uint16x4xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_uint32x4xm1** (unsigned int \*address, uint32x4xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_uint32x4xm2** (unsigned int \*address, uint32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_uint64x4xm1** (unsigned long \*address, uint64x4xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_uint64x4xm2** (unsigned long \*address, uint64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_uint8x4xm1** (unsigned char \*address, uint8x4xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg4hv\_mask\_uint8x4xm2** (unsigned char \*address, uint8x4xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

### 2.10.159 Store 4 contiguous 32b fields in memory from consecutively numbered vector registers

**Instruction:** ['vsseg4w.v']

**Prototypes:**

- void **vsseg4wv\_int16x4xm1** (short \*address, int16x4xm1\_t a, unsigned int gvl)
- void **vsseg4wv\_int16x4xm2** (short \*address, int16x4xm2\_t a, unsigned int gvl)
- void **vsseg4wv\_int32x4xm1** (int \*address, int32x4xm1\_t a, unsigned int gvl)
- void **vsseg4wv\_int32x4xm2** (int \*address, int32x4xm2\_t a, unsigned int gvl)
- void **vsseg4wv\_int64x4xm1** (long \*address, int64x4xm1\_t a, unsigned int gvl)
- void **vsseg4wv\_int64x4xm2** (long \*address, int64x4xm2\_t a, unsigned int gvl)
- void **vsseg4wv\_int8x4xm1** (signed char \*address, int8x4xm1\_t a, unsigned int gvl)
- void **vsseg4wv\_int8x4xm2** (signed char \*address, int8x4xm2\_t a, unsigned int gvl)
- void **vsseg4wv\_uint16x4xm1** (unsigned short \*address, uint16x4xm1\_t a, unsigned int gvl)
- void **vsseg4wv\_uint16x4xm2** (unsigned short \*address, uint16x4xm2\_t a, unsigned int gvl)
- void **vsseg4wv\_uint32x4xm1** (unsigned int \*address, uint32x4xm1\_t a, unsigned int gvl)
- void **vsseg4wv\_uint32x4xm2** (unsigned int \*address, uint32x4xm2\_t a, unsigned int gvl)
- void **vsseg4wv\_uint64x4xm1** (unsigned long \*address, uint64x4xm1\_t a, unsigned int gvl)
- void **vsseg4wv\_uint64x4xm2** (unsigned long \*address, uint64x4xm2\_t a, unsigned int gvl)
- void **vsseg4wv\_uint8x4xm1** (unsigned char \*address, uint8x4xm1\_t a, unsigned int gvl)
- void **vsseg4wv\_uint8x4xm2** (unsigned char \*address, uint8x4xm2\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg4wv\_mask\_int16x4xm1** (short \*address, int16x4xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg4wv\_mask\_int16x4xm2** (short \*address, int16x4xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsseg4wv\_mask\_int32x4xm1** (int \*address, int32x4xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg4wv\_mask\_int32x4xm2** (int \*address, int32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)

- void **vsseg4wv\_mask\_int64x4xm1** (long *\*address*, int64x4xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg4wv\_mask\_int64x4xm2** (long *\*address*, int64x4xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vsseg4wv\_mask\_int8x4xm1** (signed char *\*address*, int8x4xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg4wv\_mask\_int8x4xm2** (signed char *\*address*, int8x4xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vsseg4wv\_mask\_uint16x4xm1** (unsigned short *\*address*, uint16x4xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg4wv\_mask\_uint16x4xm2** (unsigned short *\*address*, uint16x4xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vsseg4wv\_mask\_uint32x4xm1** (unsigned int *\*address*, uint32x4xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg4wv\_mask\_uint32x4xm2** (unsigned int *\*address*, uint32x4xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vsseg4wv\_mask\_uint64x4xm1** (unsigned long *\*address*, uint64x4xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg4wv\_mask\_uint64x4xm2** (unsigned long *\*address*, uint64x4xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vsseg4wv\_mask\_uint8x4xm1** (unsigned char *\*address*, uint8x4xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg4wv\_mask\_uint8x4xm2** (unsigned char *\*address*, uint8x4xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.160 Store 5 contiguous 8b fields in memory from consecutively numbered vector registers

**Instruction:** [*'vsseg5b.v'*]

**Prototypes:**

- void **vsseg5bv\_int16x5xm1** (short *\*address*, int16x5xm1\_t *a*, unsigned int *gvl*)
- void **vsseg5bv\_int32x5xm1** (int *\*address*, int32x5xm1\_t *a*, unsigned int *gvl*)
- void **vsseg5bv\_int64x5xm1** (long *\*address*, int64x5xm1\_t *a*, unsigned int *gvl*)
- void **vsseg5bv\_int8x5xm1** (signed char *\*address*, int8x5xm1\_t *a*, unsigned int *gvl*)
- void **vsseg5bv\_uint16x5xm1** (unsigned short *\*address*, uint16x5xm1\_t *a*, unsigned int *gvl*)
- void **vsseg5bv\_uint32x5xm1** (unsigned int *\*address*, uint32x5xm1\_t *a*, unsigned int *gvl*)
- void **vsseg5bv\_uint64x5xm1** (unsigned long *\*address*, uint64x5xm1\_t *a*, unsigned int *gvl*)
- void **vsseg5bv\_uint8x5xm1** (unsigned char *\*address*, uint8x5xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg5bv\_mask\_int16x5xm1** (short \*address, int16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg5bv\_mask\_int32x5xm1** (int \*address, int32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg5bv\_mask\_int64x5xm1** (long \*address, int64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg5bv\_mask\_int8x5xm1** (signed char \*address, int8x5xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg5bv\_mask\_uint16x5xm1** (unsigned short \*address, uint16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg5bv\_mask\_uint32x5xm1** (unsigned int \*address, uint32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg5bv\_mask\_uint64x5xm1** (unsigned long \*address, uint64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg5bv\_mask\_uint8x5xm1** (unsigned char \*address, uint8x5xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.161 Store 5 contiguous element fields in memory from consecutively numbered vector registers

**Instruction:** [*vsseg5e.v*]

**Prototypes:**

- void **vsseg5ev\_float16x5xm1** (float16\_t \*address, float16x5xm1\_t a, unsigned int gvl)
- void **vsseg5ev\_float32x5xm1** (float \*address, float32x5xm1\_t a, unsigned int gvl)
- void **vsseg5ev\_float64x5xm1** (double \*address, float64x5xm1\_t a, unsigned int gvl)
- void **vsseg5ev\_int16x5xm1** (short \*address, int16x5xm1\_t a, unsigned int gvl)
- void **vsseg5ev\_int32x5xm1** (int \*address, int32x5xm1\_t a, unsigned int gvl)
- void **vsseg5ev\_int64x5xm1** (long \*address, int64x5xm1\_t a, unsigned int gvl)
- void **vsseg5ev\_int8x5xm1** (signed char \*address, int8x5xm1\_t a, unsigned int gvl)
- void **vsseg5ev\_uint16x5xm1** (unsigned short \*address, uint16x5xm1\_t a, unsigned int gvl)
- void **vsseg5ev\_uint32x5xm1** (unsigned int \*address, uint32x5xm1\_t a, unsigned int gvl)
- void **vsseg5ev\_uint64x5xm1** (unsigned long \*address, uint64x5xm1\_t a, unsigned int gvl)



- void **vsseg5ev\_uint8x5xm1** (unsigned char \**address*, uint8x5xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg5ev\_mask\_float16x5xm1** (float16\_t \**address*, float16x5xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg5ev\_mask\_float32x5xm1** (float \**address*, float32x5xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg5ev\_mask\_float64x5xm1** (double \**address*, float64x5xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg5ev\_mask\_int16x5xm1** (short \**address*, int16x5xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg5ev\_mask\_int32x5xm1** (int \**address*, int32x5xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg5ev\_mask\_int64x5xm1** (long \**address*, int64x5xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg5ev\_mask\_int8x5xm1** (signed char \**address*, int8x5xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg5ev\_mask\_uint16x5xm1** (unsigned short \**address*, uint16x5xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg5ev\_mask\_uint32x5xm1** (unsigned int \**address*, uint32x5xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg5ev\_mask\_uint64x5xm1** (unsigned long \**address*, uint64x5xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg5ev\_mask\_uint8x5xm1** (unsigned char \**address*, uint8x5xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.162 Store 5 contiguous 16b fields in memory from consecutively numbered vector registers

**Instruction:** [*'vsseg5h.v'*]

**Prototypes:**

- void **vsseg5hv\_int16x5xm1** (short \**address*, int16x5xm1\_t *a*, unsigned int *gvl*)
- void **vsseg5hv\_int32x5xm1** (int \**address*, int32x5xm1\_t *a*, unsigned int *gvl*)
- void **vsseg5hv\_int64x5xm1** (long \**address*, int64x5xm1\_t *a*, unsigned int *gvl*)
- void **vsseg5hv\_int8x5xm1** (signed char \**address*, int8x5xm1\_t *a*, unsigned int *gvl*)

- void **vsseg5hv\_uint16x5xm1** (unsigned short \*address, uint16x5xm1\_t a, unsigned int gvl)
- void **vsseg5hv\_uint32x5xm1** (unsigned int \*address, uint32x5xm1\_t a, unsigned int gvl)
- void **vsseg5hv\_uint64x5xm1** (unsigned long \*address, uint64x5xm1\_t a, unsigned int gvl)
- void **vsseg5hv\_uint8x5xm1** (unsigned char \*address, uint8x5xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg5hv\_mask\_int16x5xm1** (short \*address, int16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg5hv\_mask\_int32x5xm1** (int \*address, int32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg5hv\_mask\_int64x5xm1** (long \*address, int64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg5hv\_mask\_int8x5xm1** (signed char \*address, int8x5xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg5hv\_mask\_uint16x5xm1** (unsigned short \*address, uint16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg5hv\_mask\_uint32x5xm1** (unsigned int \*address, uint32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg5hv\_mask\_uint64x5xm1** (unsigned long \*address, uint64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg5hv\_mask\_uint8x5xm1** (unsigned char \*address, uint8x5xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

### 2.10.163 Store 5 contiguous 32b fields in memory from consecutively numbered vector registers

**Instruction:** ['vsseg5w.v']

**Prototypes:**

- void **vsseg5wv\_int16x5xm1** (short \*address, int16x5xm1\_t a, unsigned int gvl)
- void **vsseg5wv\_int32x5xm1** (int \*address, int32x5xm1\_t a, unsigned int gvl)
- void **vsseg5wv\_int64x5xm1** (long \*address, int64x5xm1\_t a, unsigned int gvl)
- void **vsseg5wv\_int8x5xm1** (signed char \*address, int8x5xm1\_t a, unsigned int gvl)
- void **vsseg5wv\_uint16x5xm1** (unsigned short \*address, uint16x5xm1\_t a, unsigned int gvl)
- void **vsseg5wv\_uint32x5xm1** (unsigned int \*address, uint32x5xm1\_t a, unsigned int gvl)

- void **vsseg5wv\_uint64x5xm1** (unsigned long \*address, uint64x5xm1\_t a, unsigned int gvl)
- void **vsseg5wv\_uint8x5xm1** (unsigned char \*address, uint8x5xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg5wv\_mask\_int16x5xm1** (short \*address, int16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg5wv\_mask\_int32x5xm1** (int \*address, int32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg5wv\_mask\_int64x5xm1** (long \*address, int64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg5wv\_mask\_int8x5xm1** (signed char \*address, int8x5xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg5wv\_mask\_uint16x5xm1** (unsigned short \*address, uint16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg5wv\_mask\_uint32x5xm1** (unsigned int \*address, uint32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg5wv\_mask\_uint64x5xm1** (unsigned long \*address, uint64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg5wv\_mask\_uint8x5xm1** (unsigned char \*address, uint8x5xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.164 Store 6 contiguous 8b fields in memory from consecutively numbered vector registers

**Instruction:** ['vsseg6b.v']

**Prototypes:**

- void **vsseg6bv\_int16x6xm1** (short \*address, int16x6xm1\_t a, unsigned int gvl)
- void **vsseg6bv\_int32x6xm1** (int \*address, int32x6xm1\_t a, unsigned int gvl)
- void **vsseg6bv\_int64x6xm1** (long \*address, int64x6xm1\_t a, unsigned int gvl)
- void **vsseg6bv\_int8x6xm1** (signed char \*address, int8x6xm1\_t a, unsigned int gvl)
- void **vsseg6bv\_uint16x6xm1** (unsigned short \*address, uint16x6xm1\_t a, unsigned int gvl)
- void **vsseg6bv\_uint32x6xm1** (unsigned int \*address, uint32x6xm1\_t a, unsigned int gvl)
- void **vsseg6bv\_uint64x6xm1** (unsigned long \*address, uint64x6xm1\_t a, unsigned int gvl)
- void **vsseg6bv\_uint8x6xm1** (unsigned char \*address, uint8x6xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg6bv\_mask\_int16x6xm1** (short \*address, int16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg6bv\_mask\_int32x6xm1** (int \*address, int32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg6bv\_mask\_int64x6xm1** (long \*address, int64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg6bv\_mask\_int8x6xm1** (signed char \*address, int8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg6bv\_mask\_uint16x6xm1** (unsigned short \*address, uint16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg6bv\_mask\_uint32x6xm1** (unsigned int \*address, uint32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg6bv\_mask\_uint64x6xm1** (unsigned long \*address, uint64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg6bv\_mask\_uint8x6xm1** (unsigned char \*address, uint8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.165 Store 6 contiguous element fields in memory from consecutively numbered vector registers

**Instruction:** [*vsseg6e.v*]

**Prototypes:**

- void **vsseg6ev\_float16x6xm1** (float16\_t \*address, float16x6xm1\_t a, unsigned int gvl)
- void **vsseg6ev\_float32x6xm1** (float \*address, float32x6xm1\_t a, unsigned int gvl)
- void **vsseg6ev\_float64x6xm1** (double \*address, float64x6xm1\_t a, unsigned int gvl)
- void **vsseg6ev\_int16x6xm1** (short \*address, int16x6xm1\_t a, unsigned int gvl)
- void **vsseg6ev\_int32x6xm1** (int \*address, int32x6xm1\_t a, unsigned int gvl)
- void **vsseg6ev\_int64x6xm1** (long \*address, int64x6xm1\_t a, unsigned int gvl)
- void **vsseg6ev\_int8x6xm1** (signed char \*address, int8x6xm1\_t a, unsigned int gvl)
- void **vsseg6ev\_uint16x6xm1** (unsigned short \*address, uint16x6xm1\_t a, unsigned int gvl)
- void **vsseg6ev\_uint32x6xm1** (unsigned int \*address, uint32x6xm1\_t a, unsigned int gvl)
- void **vsseg6ev\_uint64x6xm1** (unsigned long \*address, uint64x6xm1\_t a, unsigned int gvl)

- void **vsseg6ev\_uint8x6xm1** (unsigned char \**address*, uint8x6xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg6ev\_mask\_float16x6xm1** (float16\_t \**address*, float16x6xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg6ev\_mask\_float32x6xm1** (float \**address*, float32x6xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg6ev\_mask\_float64x6xm1** (double \**address*, float64x6xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg6ev\_mask\_int16x6xm1** (short \**address*, int16x6xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg6ev\_mask\_int32x6xm1** (int \**address*, int32x6xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg6ev\_mask\_int64x6xm1** (long \**address*, int64x6xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg6ev\_mask\_int8x6xm1** (signed char \**address*, int8x6xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg6ev\_mask\_uint16x6xm1** (unsigned short \**address*, uint16x6xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg6ev\_mask\_uint32x6xm1** (unsigned int \**address*, uint32x6xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg6ev\_mask\_uint64x6xm1** (unsigned long \**address*, uint64x6xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg6ev\_mask\_uint8x6xm1** (unsigned char \**address*, uint8x6xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.166 Store 6 contiguous 16b fields in memory from consecutively numbered vector registers

**Instruction:** ['vsseg6h.v']

**Prototypes:**

- void **vsseg6hv\_int16x6xm1** (short \**address*, int16x6xm1\_t *a*, unsigned int *gvl*)
- void **vsseg6hv\_int32x6xm1** (int \**address*, int32x6xm1\_t *a*, unsigned int *gvl*)
- void **vsseg6hv\_int64x6xm1** (long \**address*, int64x6xm1\_t *a*, unsigned int *gvl*)
- void **vsseg6hv\_int8x6xm1** (signed char \**address*, int8x6xm1\_t *a*, unsigned int *gvl*)

- void **vsseg6hv\_uint16x6xm1** (unsigned short \*address, uint16x6xm1\_t a, unsigned int gvl)
- void **vsseg6hv\_uint32x6xm1** (unsigned int \*address, uint32x6xm1\_t a, unsigned int gvl)
- void **vsseg6hv\_uint64x6xm1** (unsigned long \*address, uint64x6xm1\_t a, unsigned int gvl)
- void **vsseg6hv\_uint8x6xm1** (unsigned char \*address, uint8x6xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg6hv\_mask\_int16x6xm1** (short \*address, int16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg6hv\_mask\_int32x6xm1** (int \*address, int32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg6hv\_mask\_int64x6xm1** (long \*address, int64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg6hv\_mask\_int8x6xm1** (signed char \*address, int8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg6hv\_mask\_uint16x6xm1** (unsigned short \*address, uint16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg6hv\_mask\_uint32x6xm1** (unsigned int \*address, uint32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg6hv\_mask\_uint64x6xm1** (unsigned long \*address, uint64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg6hv\_mask\_uint8x6xm1** (unsigned char \*address, uint8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.167 Store 6 contiguous 32b fields in memory from consecutively numbered vector registers

**Instruction:** [*'vsseg6w.v'*]

**Prototypes:**

- void **vsseg6wv\_int16x6xm1** (short \*address, int16x6xm1\_t a, unsigned int gvl)
- void **vsseg6wv\_int32x6xm1** (int \*address, int32x6xm1\_t a, unsigned int gvl)
- void **vsseg6wv\_int64x6xm1** (long \*address, int64x6xm1\_t a, unsigned int gvl)
- void **vsseg6wv\_int8x6xm1** (signed char \*address, int8x6xm1\_t a, unsigned int gvl)
- void **vsseg6wv\_uint16x6xm1** (unsigned short \*address, uint16x6xm1\_t a, unsigned int gvl)
- void **vsseg6wv\_uint32x6xm1** (unsigned int \*address, uint32x6xm1\_t a, unsigned int gvl)



- void **vsseg6wv\_uint64x6xm1** (unsigned long \*address, uint64x6xm1\_t a, unsigned int gvl)
- void **vsseg6wv\_uint8x6xm1** (unsigned char \*address, uint8x6xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg6wv\_mask\_int16x6xm1** (short \*address, int16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg6wv\_mask\_int32x6xm1** (int \*address, int32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg6wv\_mask\_int64x6xm1** (long \*address, int64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg6wv\_mask\_int8x6xm1** (signed char \*address, int8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg6wv\_mask\_uint16x6xm1** (unsigned short \*address, uint16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg6wv\_mask\_uint32x6xm1** (unsigned int \*address, uint32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg6wv\_mask\_uint64x6xm1** (unsigned long \*address, uint64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg6wv\_mask\_uint8x6xm1** (unsigned char \*address, uint8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.168 Store 7 contiguous 8b fields in memory from consecutively numbered vector registers

**Instruction:** ['vsseg7bv.v']

**Prototypes:**

- void **vsseg7bv\_int16x7xm1** (short \*address, int16x7xm1\_t a, unsigned int gvl)
- void **vsseg7bv\_int32x7xm1** (int \*address, int32x7xm1\_t a, unsigned int gvl)
- void **vsseg7bv\_int64x7xm1** (long \*address, int64x7xm1\_t a, unsigned int gvl)
- void **vsseg7bv\_int8x7xm1** (signed char \*address, int8x7xm1\_t a, unsigned int gvl)
- void **vsseg7bv\_uint16x7xm1** (unsigned short \*address, uint16x7xm1\_t a, unsigned int gvl)
- void **vsseg7bv\_uint32x7xm1** (unsigned int \*address, uint32x7xm1\_t a, unsigned int gvl)
- void **vsseg7bv\_uint64x7xm1** (unsigned long \*address, uint64x7xm1\_t a, unsigned int gvl)
- void **vsseg7bv\_uint8x7xm1** (unsigned char \*address, uint8x7xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg7bv\_mask\_int16x7xm1** (short \*address, int16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg7bv\_mask\_int32x7xm1** (int \*address, int32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg7bv\_mask\_int64x7xm1** (long \*address, int64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg7bv\_mask\_int8x7xm1** (signed char \*address, int8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg7bv\_mask\_uint16x7xm1** (unsigned short \*address, uint16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg7bv\_mask\_uint32x7xm1** (unsigned int \*address, uint32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg7bv\_mask\_uint64x7xm1** (unsigned long \*address, uint64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg7bv\_mask\_uint8x7xm1** (unsigned char \*address, uint8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.169 Store 7 contiguous element fields in memory from consecutively numbered vector registers

**Instruction:** [*'vsseg7e.v'*]

**Prototypes:**

- void **vsseg7ev\_float16x7xm1** (float16\_t \*address, float16x7xm1\_t a, unsigned int gvl)
- void **vsseg7ev\_float32x7xm1** (float \*address, float32x7xm1\_t a, unsigned int gvl)
- void **vsseg7ev\_float64x7xm1** (double \*address, float64x7xm1\_t a, unsigned int gvl)
- void **vsseg7ev\_int16x7xm1** (short \*address, int16x7xm1\_t a, unsigned int gvl)
- void **vsseg7ev\_int32x7xm1** (int \*address, int32x7xm1\_t a, unsigned int gvl)
- void **vsseg7ev\_int64x7xm1** (long \*address, int64x7xm1\_t a, unsigned int gvl)
- void **vsseg7ev\_int8x7xm1** (signed char \*address, int8x7xm1\_t a, unsigned int gvl)
- void **vsseg7ev\_uint16x7xm1** (unsigned short \*address, uint16x7xm1\_t a, unsigned int gvl)
- void **vsseg7ev\_uint32x7xm1** (unsigned int \*address, uint32x7xm1\_t a, unsigned int gvl)
- void **vsseg7ev\_uint64x7xm1** (unsigned long \*address, uint64x7xm1\_t a, unsigned int gvl)

- void **vsseg7ev\_uint8x7xm1** (unsigned char \**address*, uint8x7xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg7ev\_mask\_float16x7xm1** (float16\_t \**address*, float16x7xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg7ev\_mask\_float32x7xm1** (float \**address*, float32x7xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg7ev\_mask\_float64x7xm1** (double \**address*, float64x7xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg7ev\_mask\_int16x7xm1** (short \**address*, int16x7xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg7ev\_mask\_int32x7xm1** (int \**address*, int32x7xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg7ev\_mask\_int64x7xm1** (long \**address*, int64x7xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg7ev\_mask\_int8x7xm1** (signed char \**address*, int8x7xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg7ev\_mask\_uint16x7xm1** (unsigned short \**address*, uint16x7xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg7ev\_mask\_uint32x7xm1** (unsigned int \**address*, uint32x7xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg7ev\_mask\_uint64x7xm1** (unsigned long \**address*, uint64x7xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg7ev\_mask\_uint8x7xm1** (unsigned char \**address*, uint8x7xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.170 Store 7 contiguous 16b fields in memory from consecutively numbered vector registers

**Instruction:** [*'vsseg7h.v'*]

**Prototypes:**

- void **vsseg7hv\_int16x7xm1** (short \**address*, int16x7xm1\_t *a*, unsigned int *gvl*)
- void **vsseg7hv\_int32x7xm1** (int \**address*, int32x7xm1\_t *a*, unsigned int *gvl*)
- void **vsseg7hv\_int64x7xm1** (long \**address*, int64x7xm1\_t *a*, unsigned int *gvl*)
- void **vsseg7hv\_int8x7xm1** (signed char \**address*, int8x7xm1\_t *a*, unsigned int *gvl*)

- void **vsseg7hv\_uint16x7xm1** (unsigned short \*address, uint16x7xm1\_t a, unsigned int gvl)
- void **vsseg7hv\_uint32x7xm1** (unsigned int \*address, uint32x7xm1\_t a, unsigned int gvl)
- void **vsseg7hv\_uint64x7xm1** (unsigned long \*address, uint64x7xm1\_t a, unsigned int gvl)
- void **vsseg7hv\_uint8x7xm1** (unsigned char \*address, uint8x7xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg7hv\_mask\_int16x7xm1** (short \*address, int16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg7hv\_mask\_int32x7xm1** (int \*address, int32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg7hv\_mask\_int64x7xm1** (long \*address, int64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg7hv\_mask\_int8x7xm1** (signed char \*address, int8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg7hv\_mask\_uint16x7xm1** (unsigned short \*address, uint16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg7hv\_mask\_uint32x7xm1** (unsigned int \*address, uint32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg7hv\_mask\_uint64x7xm1** (unsigned long \*address, uint64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg7hv\_mask\_uint8x7xm1** (unsigned char \*address, uint8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.171 Store 7 contiguous 32b fields in memory from consecutively numbered vector registers

**Instruction:** [**'vsseg7w.v'**]

**Prototypes:**

- void **vsseg7wv\_int16x7xm1** (short \*address, int16x7xm1\_t a, unsigned int gvl)
- void **vsseg7wv\_int32x7xm1** (int \*address, int32x7xm1\_t a, unsigned int gvl)
- void **vsseg7wv\_int64x7xm1** (long \*address, int64x7xm1\_t a, unsigned int gvl)
- void **vsseg7wv\_int8x7xm1** (signed char \*address, int8x7xm1\_t a, unsigned int gvl)
- void **vsseg7wv\_uint16x7xm1** (unsigned short \*address, uint16x7xm1\_t a, unsigned int gvl)
- void **vsseg7wv\_uint32x7xm1** (unsigned int \*address, uint32x7xm1\_t a, unsigned int gvl)

- void **vsseg7wv\_uint64x7xm1** (unsigned long \*address, uint64x7xm1\_t a, unsigned int gvl)
- void **vsseg7wv\_uint8x7xm1** (unsigned char \*address, uint8x7xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg7wv\_mask\_int16x7xm1** (short \*address, int16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg7wv\_mask\_int32x7xm1** (int \*address, int32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg7wv\_mask\_int64x7xm1** (long \*address, int64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg7wv\_mask\_int8x7xm1** (signed char \*address, int8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg7wv\_mask\_uint16x7xm1** (unsigned short \*address, uint16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg7wv\_mask\_uint32x7xm1** (unsigned int \*address, uint32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg7wv\_mask\_uint64x7xm1** (unsigned long \*address, uint64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg7wv\_mask\_uint8x7xm1** (unsigned char \*address, uint8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.172 Store 8 contiguous 8b fields in memory from consecutively numbered vector registers

**Instruction:** [*'vsseg8bv.v'*]

**Prototypes:**

- void **vsseg8bv\_int16x8xm1** (short \*address, int16x8xm1\_t a, unsigned int gvl)
- void **vsseg8bv\_int32x8xm1** (int \*address, int32x8xm1\_t a, unsigned int gvl)
- void **vsseg8bv\_int64x8xm1** (long \*address, int64x8xm1\_t a, unsigned int gvl)
- void **vsseg8bv\_int8x8xm1** (signed char \*address, int8x8xm1\_t a, unsigned int gvl)
- void **vsseg8bv\_uint16x8xm1** (unsigned short \*address, uint16x8xm1\_t a, unsigned int gvl)
- void **vsseg8bv\_uint32x8xm1** (unsigned int \*address, uint32x8xm1\_t a, unsigned int gvl)
- void **vsseg8bv\_uint64x8xm1** (unsigned long \*address, uint64x8xm1\_t a, unsigned int gvl)
- void **vsseg8bv\_uint8x8xm1** (unsigned char \*address, uint8x8xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg8bv\_mask\_int16x8xm1** (short \*address, int16x8xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg8bv\_mask\_int32x8xm1** (int \*address, int32x8xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg8bv\_mask\_int64x8xm1** (long \*address, int64x8xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg8bv\_mask\_int8x8xm1** (signed char \*address, int8x8xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg8bv\_mask\_uint16x8xm1** (unsigned short \*address, uint16x8xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg8bv\_mask\_uint32x8xm1** (unsigned int \*address, uint32x8xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg8bv\_mask\_uint64x8xm1** (unsigned long \*address, uint64x8xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg8bv\_mask\_uint8x8xm1** (unsigned char \*address, uint8x8xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.173 Store 8 contiguous element fields in memory from consecutively numbered vector registers

**Instruction:** [*vsseg8e.v*]

**Prototypes:**

- void **vsseg8ev\_float16x8xm1** (float16\_t \*address, float16x8xm1\_t a, unsigned int gvl)
- void **vsseg8ev\_float32x8xm1** (float \*address, float32x8xm1\_t a, unsigned int gvl)
- void **vsseg8ev\_float64x8xm1** (double \*address, float64x8xm1\_t a, unsigned int gvl)
- void **vsseg8ev\_int16x8xm1** (short \*address, int16x8xm1\_t a, unsigned int gvl)
- void **vsseg8ev\_int32x8xm1** (int \*address, int32x8xm1\_t a, unsigned int gvl)
- void **vsseg8ev\_int64x8xm1** (long \*address, int64x8xm1\_t a, unsigned int gvl)
- void **vsseg8ev\_int8x8xm1** (signed char \*address, int8x8xm1\_t a, unsigned int gvl)
- void **vsseg8ev\_uint16x8xm1** (unsigned short \*address, uint16x8xm1\_t a, unsigned int gvl)
- void **vsseg8ev\_uint32x8xm1** (unsigned int \*address, uint32x8xm1\_t a, unsigned int gvl)
- void **vsseg8ev\_uint64x8xm1** (unsigned long \*address, uint64x8xm1\_t a, unsigned int gvl)



- void **vsseg8ev\_uint8x8xm1** (unsigned char \**address*, uint8x8xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg8ev\_mask\_float16x8xm1** (float16\_t \**address*, float16x8xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg8ev\_mask\_float32x8xm1** (float \**address*, float32x8xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg8ev\_mask\_float64x8xm1** (double \**address*, float64x8xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg8ev\_mask\_int16x8xm1** (short \**address*, int16x8xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg8ev\_mask\_int32x8xm1** (int \**address*, int32x8xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg8ev\_mask\_int64x8xm1** (long \**address*, int64x8xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg8ev\_mask\_int8x8xm1** (signed char \**address*, int8x8xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg8ev\_mask\_uint16x8xm1** (unsigned short \**address*, uint16x8xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg8ev\_mask\_uint32x8xm1** (unsigned int \**address*, uint32x8xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg8ev\_mask\_uint64x8xm1** (unsigned long \**address*, uint64x8xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsseg8ev\_mask\_uint8x8xm1** (unsigned char \**address*, uint8x8xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.174 Store 8 contiguous 16b fields in memory from consecutively numbered vector registers

**Instruction:** ['vsseg8h.v']

**Prototypes:**

- void **vsseg8hv\_int16x8xm1** (short \**address*, int16x8xm1\_t *a*, unsigned int *gvl*)
- void **vsseg8hv\_int32x8xm1** (int \**address*, int32x8xm1\_t *a*, unsigned int *gvl*)
- void **vsseg8hv\_int64x8xm1** (long \**address*, int64x8xm1\_t *a*, unsigned int *gvl*)
- void **vsseg8hv\_int8x8xm1** (signed char \**address*, int8x8xm1\_t *a*, unsigned int *gvl*)

- void **vsseg8hv\_uint16x8xm1** (unsigned short \*address, uint16x8xm1\_t a, unsigned int gvl)
- void **vsseg8hv\_uint32x8xm1** (unsigned int \*address, uint32x8xm1\_t a, unsigned int gvl)
- void **vsseg8hv\_uint64x8xm1** (unsigned long \*address, uint64x8xm1\_t a, unsigned int gvl)
- void **vsseg8hv\_uint8x8xm1** (unsigned char \*address, uint8x8xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg8hv\_mask\_int16x8xm1** (short \*address, int16x8xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg8hv\_mask\_int32x8xm1** (int \*address, int32x8xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg8hv\_mask\_int64x8xm1** (long \*address, int64x8xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg8hv\_mask\_int8x8xm1** (signed char \*address, int8x8xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg8hv\_mask\_uint16x8xm1** (unsigned short \*address, uint16x8xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg8hv\_mask\_uint32x8xm1** (unsigned int \*address, uint32x8xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg8hv\_mask\_uint64x8xm1** (unsigned long \*address, uint64x8xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg8hv\_mask\_uint8x8xm1** (unsigned char \*address, uint8x8xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.175 Store 8 contiguous 32b fields in memory from consecutively numbered vector registers

**Instruction:** ['vsseg8w.v']

**Prototypes:**

- void **vsseg8wv\_int16x8xm1** (short \*address, int16x8xm1\_t a, unsigned int gvl)
- void **vsseg8wv\_int32x8xm1** (int \*address, int32x8xm1\_t a, unsigned int gvl)
- void **vsseg8wv\_int64x8xm1** (long \*address, int64x8xm1\_t a, unsigned int gvl)
- void **vsseg8wv\_int8x8xm1** (signed char \*address, int8x8xm1\_t a, unsigned int gvl)
- void **vsseg8wv\_uint16x8xm1** (unsigned short \*address, uint16x8xm1\_t a, unsigned int gvl)
- void **vsseg8wv\_uint32x8xm1** (unsigned int \*address, uint32x8xm1\_t a, unsigned int gvl)

- void **vsseg8wv\_uint64x8xm1** (unsigned long \*address, uint64x8xm1\_t a, unsigned int gvl)
- void **vsseg8wv\_uint8x8xm1** (unsigned char \*address, uint8x8xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment)
```

**Masked prototypes:**

- void **vsseg8wv\_mask\_int16x8xm1** (short \*address, int16x8xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg8wv\_mask\_int32x8xm1** (int \*address, int32x8xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg8wv\_mask\_int64x8xm1** (long \*address, int64x8xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg8wv\_mask\_int8x8xm1** (signed char \*address, int8x8xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsseg8wv\_mask\_uint16x8xm1** (unsigned short \*address, uint16x8xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsseg8wv\_mask\_uint32x8xm1** (unsigned int \*address, uint32x8xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsseg8wv\_mask\_uint64x8xm1** (unsigned long \*address, uint64x8xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsseg8wv\_mask\_uint8x8xm1** (unsigned char \*address, uint8x8xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment)
```

## 2.10.176 Store 2 contiguous 8b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** ['vssseg2bv.v']

**Prototypes:**

- void **vssseg2bv\_int16x2xm1** (short \*address, long stride, int16x2xm1\_t a, unsigned int gvl)
- void **vssseg2bv\_int16x2xm2** (short \*address, long stride, int16x2xm2\_t a, unsigned int gvl)
- void **vssseg2bv\_int16x2xm4** (short \*address, long stride, int16x2xm4\_t a, unsigned int gvl)
- void **vssseg2bv\_int32x2xm1** (int \*address, long stride, int32x2xm1\_t a, unsigned int gvl)
- void **vssseg2bv\_int32x2xm2** (int \*address, long stride, int32x2xm2\_t a, unsigned int gvl)
- void **vssseg2bv\_int32x2xm4** (int \*address, long stride, int32x2xm4\_t a, unsigned int gvl)
- void **vssseg2bv\_int64x2xm1** (long \*address, long stride, int64x2xm1\_t a, unsigned int gvl)
- void **vssseg2bv\_int64x2xm2** (long \*address, long stride, int64x2xm2\_t a, unsigned int gvl)

- void **vssseg2bv\_int64x2xm4** (long *\*address*, long *stride*, int64x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_int8x2xm1** (signed char *\*address*, long *stride*, int8x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_int8x2xm2** (signed char *\*address*, long *stride*, int8x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_int8x2xm4** (signed char *\*address*, long *stride*, int8x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_uint16x2xm1** (unsigned short *\*address*, long *stride*, uint16x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_uint16x2xm2** (unsigned short *\*address*, long *stride*, uint16x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_uint16x2xm4** (unsigned short *\*address*, long *stride*, uint16x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_uint32x2xm1** (unsigned int *\*address*, long *stride*, uint32x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_uint32x2xm2** (unsigned int *\*address*, long *stride*, uint32x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_uint32x2xm4** (unsigned int *\*address*, long *stride*, uint32x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_uint64x2xm1** (unsigned long *\*address*, long *stride*, uint64x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_uint64x2xm2** (unsigned long *\*address*, long *stride*, uint64x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_uint64x2xm4** (unsigned long *\*address*, long *stride*, uint64x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_uint8x2xm1** (unsigned char *\*address*, long *stride*, uint8x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_uint8x2xm2** (unsigned char *\*address*, long *stride*, uint8x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2bv\_uint8x2xm4** (unsigned char *\*address*, long *stride*, uint8x2xm4\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg2bv\_mask\_int16x2xm1** (short *\*address*, long *stride*, int16x2xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_int16x2xm2** (short *\*address*, long *stride*, int16x2xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_int16x2xm4** (short *\*address*, long *stride*, int16x2xm4\_t *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_int32x2xm1** (int *\*address*, long *stride*, int32x2xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_int32x2xm2** (int *\*address*, long *stride*, int32x2xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_int32x2xm4** (int *\*address*, long *stride*, int32x2xm4\_t *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)

- void **vssseg2bv\_mask\_int64x2xm1** (long *\*address*, long *stride*, int64x2xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_int64x2xm2** (long *\*address*, long *stride*, int64x2xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_int64x2xm4** (long *\*address*, long *stride*, int64x2xm4\_t *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_int8x2xm1** (signed char *\*address*, long *stride*, int8x2xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_int8x2xm2** (signed char *\*address*, long *stride*, int8x2xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_int8x2xm4** (signed char *\*address*, long *stride*, int8x2xm4\_t *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_uint16x2xm1** (unsigned short *\*address*, long *stride*, uint16x2xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_uint16x2xm2** (unsigned short *\*address*, long *stride*, uint16x2xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_uint16x2xm4** (unsigned short *\*address*, long *stride*, uint16x2xm4\_t *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_uint32x2xm1** (unsigned int *\*address*, long *stride*, uint32x2xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_uint32x2xm2** (unsigned int *\*address*, long *stride*, uint32x2xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_uint32x2xm4** (unsigned int *\*address*, long *stride*, uint32x2xm4\_t *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_uint64x2xm1** (unsigned long *\*address*, long *stride*, uint64x2xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_uint64x2xm2** (unsigned long *\*address*, long *stride*, uint64x2xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_uint64x2xm4** (unsigned long *\*address*, long *stride*, uint64x2xm4\_t *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_uint8x2xm1** (unsigned char *\*address*, long *stride*, uint8x2xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_uint8x2xm2** (unsigned char *\*address*, long *stride*, uint8x2xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2bv\_mask\_uint8x2xm4** (unsigned char *\*address*, long *stride*, uint8x2xm4\_t *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

### 2.10.177 Store 2 contiguous element fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [*'vssseg2e.v'*]

**Prototypes:**

- void **vssseg2ev\_float16x2xm1** (float16\_t \*address, long stride, float16x2xm1\_t a, unsigned int gvl)
- void **vssseg2ev\_float16x2xm2** (float16\_t \*address, long stride, float16x2xm2\_t a, unsigned int gvl)
- void **vssseg2ev\_float16x2xm4** (float16\_t \*address, long stride, float16x2xm4\_t a, unsigned int gvl)
- void **vssseg2ev\_float32x2xm1** (float \*address, long stride, float32x2xm1\_t a, unsigned int gvl)
- void **vssseg2ev\_float32x2xm2** (float \*address, long stride, float32x2xm2\_t a, unsigned int gvl)
- void **vssseg2ev\_float32x2xm4** (float \*address, long stride, float32x2xm4\_t a, unsigned int gvl)
- void **vssseg2ev\_float64x2xm1** (double \*address, long stride, float64x2xm1\_t a, unsigned int gvl)
- void **vssseg2ev\_float64x2xm2** (double \*address, long stride, float64x2xm2\_t a, unsigned int gvl)
- void **vssseg2ev\_float64x2xm4** (double \*address, long stride, float64x2xm4\_t a, unsigned int gvl)
- void **vssseg2ev\_int16x2xm1** (short \*address, long stride, int16x2xm1\_t a, unsigned int gvl)
- void **vssseg2ev\_int16x2xm2** (short \*address, long stride, int16x2xm2\_t a, unsigned int gvl)
- void **vssseg2ev\_int16x2xm4** (short \*address, long stride, int16x2xm4\_t a, unsigned int gvl)
- void **vssseg2ev\_int32x2xm1** (int \*address, long stride, int32x2xm1\_t a, unsigned int gvl)
- void **vssseg2ev\_int32x2xm2** (int \*address, long stride, int32x2xm2\_t a, unsigned int gvl)
- void **vssseg2ev\_int32x2xm4** (int \*address, long stride, int32x2xm4\_t a, unsigned int gvl)
- void **vssseg2ev\_int64x2xm1** (long \*address, long stride, int64x2xm1\_t a, unsigned int gvl)
- void **vssseg2ev\_int64x2xm2** (long \*address, long stride, int64x2xm2\_t a, unsigned int gvl)
- void **vssseg2ev\_int64x2xm4** (long \*address, long stride, int64x2xm4\_t a, unsigned int gvl)
- void **vssseg2ev\_int8x2xm1** (signed char \*address, long stride, int8x2xm1\_t a, unsigned int gvl)
- void **vssseg2ev\_int8x2xm2** (signed char \*address, long stride, int8x2xm2\_t a, unsigned int gvl)
- void **vssseg2ev\_int8x2xm4** (signed char \*address, long stride, int8x2xm4\_t a, unsigned int gvl)
- void **vssseg2ev\_uint16x2xm1** (unsigned short \*address, long stride, uint16x2xm1\_t a, unsigned int gvl)
- void **vssseg2ev\_uint16x2xm2** (unsigned short \*address, long stride, uint16x2xm2\_t a, unsigned int gvl)
- void **vssseg2ev\_uint16x2xm4** (unsigned short \*address, long stride, uint16x2xm4\_t a, unsigned int gvl)
- void **vssseg2ev\_uint32x2xm1** (unsigned int \*address, long stride, uint32x2xm1\_t a, unsigned int gvl)
- void **vssseg2ev\_uint32x2xm2** (unsigned int \*address, long stride, uint32x2xm2\_t a, unsigned int gvl)
- void **vssseg2ev\_uint32x2xm4** (unsigned int \*address, long stride, uint32x2xm4\_t a, unsigned int gvl)
- void **vssseg2ev\_uint64x2xm1** (unsigned long \*address, long stride, uint64x2xm1\_t a, unsigned int gvl)
- void **vssseg2ev\_uint64x2xm2** (unsigned long \*address, long stride, uint64x2xm2\_t a, unsigned int gvl)



- void **vssseg2ev\_uint64x2xm4** (unsigned long *\*address*, long *stride*, uint64x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2ev\_uint8x2xm1** (unsigned char *\*address*, long *stride*, uint8x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2ev\_uint8x2xm2** (unsigned char *\*address*, long *stride*, uint8x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2ev\_uint8x2xm4** (unsigned char *\*address*, long *stride*, uint8x2xm4\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg2ev\_mask\_float16x2xm1** (float16\_t *\*address*, long *stride*, float16x2xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_float16x2xm2** (float16\_t *\*address*, long *stride*, float16x2xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_float16x2xm4** (float16\_t *\*address*, long *stride*, float16x2xm4\_t *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_float32x2xm1** (float *\*address*, long *stride*, float32x2xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_float32x2xm2** (float *\*address*, long *stride*, float32x2xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_float32x2xm4** (float *\*address*, long *stride*, float32x2xm4\_t *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_float64x2xm1** (double *\*address*, long *stride*, float64x2xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_float64x2xm2** (double *\*address*, long *stride*, float64x2xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_float64x2xm4** (double *\*address*, long *stride*, float64x2xm4\_t *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_int16x2xm1** (short *\*address*, long *stride*, int16x2xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_int16x2xm2** (short *\*address*, long *stride*, int16x2xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_int16x2xm4** (short *\*address*, long *stride*, int16x2xm4\_t *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_int32x2xm1** (int *\*address*, long *stride*, int32x2xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_int32x2xm2** (int *\*address*, long *stride*, int32x2xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_int32x2xm4** (int *\*address*, long *stride*, int32x2xm4\_t *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_int64x2xm1** (long *\*address*, long *stride*, int64x2xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)

- void **vssseg2ev\_mask\_int64x2xm2** (long *\*address*, long *stride*, int64x2xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_int64x2xm4** (long *\*address*, long *stride*, int64x2xm4\_t *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_int8x2xm1** (signed char *\*address*, long *stride*, int8x2xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_int8x2xm2** (signed char *\*address*, long *stride*, int8x2xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_int8x2xm4** (signed char *\*address*, long *stride*, int8x2xm4\_t *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_uint16x2xm1** (unsigned short *\*address*, long *stride*, uint16x2xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_uint16x2xm2** (unsigned short *\*address*, long *stride*, uint16x2xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_uint16x2xm4** (unsigned short *\*address*, long *stride*, uint16x2xm4\_t *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_uint32x2xm1** (unsigned int *\*address*, long *stride*, uint32x2xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_uint32x2xm2** (unsigned int *\*address*, long *stride*, uint32x2xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_uint32x2xm4** (unsigned int *\*address*, long *stride*, uint32x2xm4\_t *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_uint64x2xm1** (unsigned long *\*address*, long *stride*, uint64x2xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_uint64x2xm2** (unsigned long *\*address*, long *stride*, uint64x2xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_uint64x2xm4** (unsigned long *\*address*, long *stride*, uint64x2xm4\_t *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_uint8x2xm1** (unsigned char *\*address*, long *stride*, uint8x2xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_uint8x2xm2** (unsigned char *\*address*, long *stride*, uint8x2xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2ev\_mask\_uint8x2xm4** (unsigned char *\*address*, long *stride*, uint8x2xm4\_t *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.178 Store 2 contiguous 16b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [*'vssseg2h.v'*]

**Prototypes:**

- void **vssseg2hv\_int16x2xm1** (short *\*address*, long *stride*, int16x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_int16x2xm2** (short *\*address*, long *stride*, int16x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_int16x2xm4** (short *\*address*, long *stride*, int16x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_int32x2xm1** (int *\*address*, long *stride*, int32x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_int32x2xm2** (int *\*address*, long *stride*, int32x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_int32x2xm4** (int *\*address*, long *stride*, int32x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_int64x2xm1** (long *\*address*, long *stride*, int64x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_int64x2xm2** (long *\*address*, long *stride*, int64x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_int64x2xm4** (long *\*address*, long *stride*, int64x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_int8x2xm1** (signed char *\*address*, long *stride*, int8x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_int8x2xm2** (signed char *\*address*, long *stride*, int8x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_int8x2xm4** (signed char *\*address*, long *stride*, int8x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_uint16x2xm1** (unsigned short *\*address*, long *stride*, uint16x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_uint16x2xm2** (unsigned short *\*address*, long *stride*, uint16x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_uint16x2xm4** (unsigned short *\*address*, long *stride*, uint16x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_uint32x2xm1** (unsigned int *\*address*, long *stride*, uint32x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_uint32x2xm2** (unsigned int *\*address*, long *stride*, uint32x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_uint32x2xm4** (unsigned int *\*address*, long *stride*, uint32x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_uint64x2xm1** (unsigned long *\*address*, long *stride*, uint64x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_uint64x2xm2** (unsigned long *\*address*, long *stride*, uint64x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_uint64x2xm4** (unsigned long *\*address*, long *stride*, uint64x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_uint8x2xm1** (unsigned char *\*address*, long *stride*, uint8x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_uint8x2xm2** (unsigned char *\*address*, long *stride*, uint8x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2hv\_uint8x2xm4** (unsigned char *\*address*, long *stride*, uint8x2xm4\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg2hv\_mask\_int16x2xm1** (short *\*address*, long *stride*, int16x2xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_int16x2xm2** (short *\*address*, long *stride*, int16x2xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_int16x2xm4** (short *\*address*, long *stride*, int16x2xm4\_t *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_int32x2xm1** (int *\*address*, long *stride*, int32x2xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_int32x2xm2** (int *\*address*, long *stride*, int32x2xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_int32x2xm4** (int *\*address*, long *stride*, int32x2xm4\_t *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_int64x2xm1** (long *\*address*, long *stride*, int64x2xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_int64x2xm2** (long *\*address*, long *stride*, int64x2xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_int64x2xm4** (long *\*address*, long *stride*, int64x2xm4\_t *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_int8x2xm1** (signed char *\*address*, long *stride*, int8x2xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_int8x2xm2** (signed char *\*address*, long *stride*, int8x2xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_int8x2xm4** (signed char *\*address*, long *stride*, int8x2xm4\_t *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_uint16x2xm1** (unsigned short *\*address*, long *stride*, uint16x2xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_uint16x2xm2** (unsigned short *\*address*, long *stride*, uint16x2xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_uint16x2xm4** (unsigned short *\*address*, long *stride*, uint16x2xm4\_t *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_uint32x2xm1** (unsigned int *\*address*, long *stride*, uint32x2xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_uint32x2xm2** (unsigned int *\*address*, long *stride*, uint32x2xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_uint32x2xm4** (unsigned int *\*address*, long *stride*, uint32x2xm4\_t *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_uint64x2xm1** (unsigned long *\*address*, long *stride*, uint64x2xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_uint64x2xm2** (unsigned long *\*address*, long *stride*, uint64x2xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_uint64x2xm4** (unsigned long *\*address*, long *stride*, uint64x2xm4\_t *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_uint8x2xm1** (unsigned char *\*address*, long *stride*, uint8x2xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2hv\_mask\_uint8x2xm2** (unsigned char *\*address*, long *stride*, uint8x2xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)

- void **vssseg2hv\_mask\_uint8x2xm4**(unsigned char \**address*, long *stride*, uint8x2xm4\_t *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.179 Store 2 contiguous 32b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [*'vssseg2w.v'*]

**Prototypes:**

- void **vssseg2wv\_int16x2xm1**(short \**address*, long *stride*, int16x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_int16x2xm2**(short \**address*, long *stride*, int16x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_int16x2xm4**(short \**address*, long *stride*, int16x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_int32x2xm1**(int \**address*, long *stride*, int32x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_int32x2xm2**(int \**address*, long *stride*, int32x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_int32x2xm4**(int \**address*, long *stride*, int32x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_int64x2xm1**(long \**address*, long *stride*, int64x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_int64x2xm2**(long \**address*, long *stride*, int64x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_int64x2xm4**(long \**address*, long *stride*, int64x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_int8x2xm1**(signed char \**address*, long *stride*, int8x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_int8x2xm2**(signed char \**address*, long *stride*, int8x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_int8x2xm4**(signed char \**address*, long *stride*, int8x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_uint16x2xm1**(unsigned short \**address*, long *stride*, uint16x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_uint16x2xm2**(unsigned short \**address*, long *stride*, uint16x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_uint16x2xm4**(unsigned short \**address*, long *stride*, uint16x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_uint32x2xm1**(unsigned int \**address*, long *stride*, uint32x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_uint32x2xm2**(unsigned int \**address*, long *stride*, uint32x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_uint32x2xm4**(unsigned int \**address*, long *stride*, uint32x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_uint64x2xm1**(unsigned long \**address*, long *stride*, uint64x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_uint64x2xm2**(unsigned long \**address*, long *stride*, uint64x2xm2\_t *a*, unsigned int *gvl*)

- void **vssseg2wv\_uint64x2xm4** (unsigned long *\*address*, long *stride*, uint64x2xm4\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_uint8x2xm1** (unsigned char *\*address*, long *stride*, uint8x2xm1\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_uint8x2xm2** (unsigned char *\*address*, long *stride*, uint8x2xm2\_t *a*, unsigned int *gvl*)
- void **vssseg2wv\_uint8x2xm4** (unsigned char *\*address*, long *stride*, uint8x2xm4\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg2wv\_mask\_int16x2xm1** (short *\*address*, long *stride*, int16x2xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_int16x2xm2** (short *\*address*, long *stride*, int16x2xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_int16x2xm4** (short *\*address*, long *stride*, int16x2xm4\_t *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_int32x2xm1** (int *\*address*, long *stride*, int32x2xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_int32x2xm2** (int *\*address*, long *stride*, int32x2xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_int32x2xm4** (int *\*address*, long *stride*, int32x2xm4\_t *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_int64x2xm1** (long *\*address*, long *stride*, int64x2xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_int64x2xm2** (long *\*address*, long *stride*, int64x2xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_int64x2xm4** (long *\*address*, long *stride*, int64x2xm4\_t *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_int8x2xm1** (signed char *\*address*, long *stride*, int8x2xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_int8x2xm2** (signed char *\*address*, long *stride*, int8x2xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_int8x2xm4** (signed char *\*address*, long *stride*, int8x2xm4\_t *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_uint16x2xm1** (unsigned short *\*address*, long *stride*, uint16x2xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_uint16x2xm2** (unsigned short *\*address*, long *stride*, uint16x2xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_uint16x2xm4** (unsigned short *\*address*, long *stride*, uint16x2xm4\_t *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_uint32x2xm1** (unsigned int *\*address*, long *stride*, uint32x2xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)



- void **vssseg2wv\_mask\_uint32x2xm2** (unsigned int *\*address*, long *stride*, uint32x2xm2\_t *a*, e32xm2\_t *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_uint32x2xm4** (unsigned int *\*address*, long *stride*, uint32x2xm4\_t *a*, e32xm4\_t *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_uint64x2xm1** (unsigned long *\*address*, long *stride*, uint64x2xm1\_t *a*, e64xm1\_t *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_uint64x2xm2** (unsigned long *\*address*, long *stride*, uint64x2xm2\_t *a*, e64xm2\_t *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_uint64x2xm4** (unsigned long *\*address*, long *stride*, uint64x2xm4\_t *a*, e64xm4\_t *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_uint8x2xm1** (unsigned char *\*address*, long *stride*, uint8x2xm1\_t *a*, e8xm1\_t *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_uint8x2xm2** (unsigned char *\*address*, long *stride*, uint8x2xm2\_t *a*, e8xm2\_t *mask*, unsigned int *gvl*)
- void **vssseg2wv\_mask\_uint8x2xm4** (unsigned char *\*address*, long *stride*, uint8x2xm4\_t *a*, e8xm4\_t *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

**2.10.180 Store 3 contiguous 8b fields in memory(strided) from consecutively numbered vector registers****Instruction:** [*vssseg3b.v*]**Prototypes:**

- void **vssseg3bv\_int16x3xm1** (short *\*address*, long *stride*, int16x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_int16x3xm2** (short *\*address*, long *stride*, int16x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_int32x3xm1** (int *\*address*, long *stride*, int32x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_int32x3xm2** (int *\*address*, long *stride*, int32x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_int64x3xm1** (long *\*address*, long *stride*, int64x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_int64x3xm2** (long *\*address*, long *stride*, int64x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_int8x3xm1** (signed char *\*address*, long *stride*, int8x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_int8x3xm2** (signed char *\*address*, long *stride*, int8x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_uint16x3xm1** (unsigned short *\*address*, long *stride*, uint16x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_uint16x3xm2** (unsigned short *\*address*, long *stride*, uint16x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_uint32x3xm1** (unsigned int *\*address*, long *stride*, uint32x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_uint32x3xm2** (unsigned int *\*address*, long *stride*, uint32x3xm2\_t *a*, unsigned int *gvl*)

- void **vssseg3bv\_uint64x3xm1** (unsigned long *\*address*, long *stride*, uint64x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_uint64x3xm2** (unsigned long *\*address*, long *stride*, uint64x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_uint8x3xm1** (unsigned char *\*address*, long *stride*, uint8x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3bv\_uint8x3xm2** (unsigned char *\*address*, long *stride*, uint8x3xm2\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg3bv\_mask\_int16x3xm1** (short *\*address*, long *stride*, int16x3xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_int16x3xm2** (short *\*address*, long *stride*, int16x3xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_int32x3xm1** (int *\*address*, long *stride*, int32x3xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_int32x3xm2** (int *\*address*, long *stride*, int32x3xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_int64x3xm1** (long *\*address*, long *stride*, int64x3xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_int64x3xm2** (long *\*address*, long *stride*, int64x3xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_int8x3xm1** (signed char *\*address*, long *stride*, int8x3xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_int8x3xm2** (signed char *\*address*, long *stride*, int8x3xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_uint16x3xm1** (unsigned short *\*address*, long *stride*, uint16x3xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_uint16x3xm2** (unsigned short *\*address*, long *stride*, uint16x3xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_uint32x3xm1** (unsigned int *\*address*, long *stride*, uint32x3xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_uint32x3xm2** (unsigned int *\*address*, long *stride*, uint32x3xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_uint64x3xm1** (unsigned long *\*address*, long *stride*, uint64x3xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_uint64x3xm2** (unsigned long *\*address*, long *stride*, uint64x3xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_uint8x3xm1** (unsigned char *\*address*, long *stride*, uint8x3xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3bv\_mask\_uint8x3xm2** (unsigned char *\*address*, long *stride*, uint8x3xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

**2.10.181 Store 3 contiguous element fields in memory(strided) from consecutively numbered vector registers****Instruction:** ['vssseg3e.v']**Prototypes:**

- void **vssseg3ev\_float16x3xm1** (float16\_t \*address, long stride, float16x3xm1\_t a, unsigned int gvl)
- void **vssseg3ev\_float16x3xm2** (float16\_t \*address, long stride, float16x3xm2\_t a, unsigned int gvl)
- void **vssseg3ev\_float32x3xm1** (float \*address, long stride, float32x3xm1\_t a, unsigned int gvl)
- void **vssseg3ev\_float32x3xm2** (float \*address, long stride, float32x3xm2\_t a, unsigned int gvl)
- void **vssseg3ev\_float64x3xm1** (double \*address, long stride, float64x3xm1\_t a, unsigned int gvl)
- void **vssseg3ev\_float64x3xm2** (double \*address, long stride, float64x3xm2\_t a, unsigned int gvl)
- void **vssseg3ev\_int16x3xm1** (short \*address, long stride, int16x3xm1\_t a, unsigned int gvl)
- void **vssseg3ev\_int16x3xm2** (short \*address, long stride, int16x3xm2\_t a, unsigned int gvl)
- void **vssseg3ev\_int32x3xm1** (int \*address, long stride, int32x3xm1\_t a, unsigned int gvl)
- void **vssseg3ev\_int32x3xm2** (int \*address, long stride, int32x3xm2\_t a, unsigned int gvl)
- void **vssseg3ev\_int64x3xm1** (long \*address, long stride, int64x3xm1\_t a, unsigned int gvl)
- void **vssseg3ev\_int64x3xm2** (long \*address, long stride, int64x3xm2\_t a, unsigned int gvl)
- void **vssseg3ev\_int8x3xm1** (signed char \*address, long stride, int8x3xm1\_t a, unsigned int gvl)
- void **vssseg3ev\_int8x3xm2** (signed char \*address, long stride, int8x3xm2\_t a, unsigned int gvl)
- void **vssseg3ev\_uint16x3xm1** (unsigned short \*address, long stride, uint16x3xm1\_t a, unsigned int gvl)
- void **vssseg3ev\_uint16x3xm2** (unsigned short \*address, long stride, uint16x3xm2\_t a, unsigned int gvl)
- void **vssseg3ev\_uint32x3xm1** (unsigned int \*address, long stride, uint32x3xm1\_t a, unsigned int gvl)
- void **vssseg3ev\_uint32x3xm2** (unsigned int \*address, long stride, uint32x3xm2\_t a, unsigned int gvl)
- void **vssseg3ev\_uint64x3xm1** (unsigned long \*address, long stride, uint64x3xm1\_t a, unsigned int gvl)
- void **vssseg3ev\_uint64x3xm2** (unsigned long \*address, long stride, uint64x3xm2\_t a, unsigned int gvl)
- void **vssseg3ev\_uint8x3xm1** (unsigned char \*address, long stride, uint8x3xm1\_t a, unsigned int gvl)

- void **vssseg3ev\_uint8x3xm2** (unsigned char \**address*, long *stride*, uint8x3xm2\_t *a*, unsigned int *gvl*)

#### Operation:

```
>>> for segment(3 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

#### Masked prototypes:

- void **vssseg3ev\_mask\_float16x3xm1** (float16\_t \**address*, long *stride*, float16x3xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_float16x3xm2** (float16\_t \**address*, long *stride*, float16x3xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_float32x3xm1** (float \**address*, long *stride*, float32x3xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_float32x3xm2** (float \**address*, long *stride*, float32x3xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_float64x3xm1** (double \**address*, long *stride*, float64x3xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_float64x3xm2** (double \**address*, long *stride*, float64x3xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_int16x3xm1** (short \**address*, long *stride*, int16x3xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_int16x3xm2** (short \**address*, long *stride*, int16x3xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_int32x3xm1** (int \**address*, long *stride*, int32x3xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_int32x3xm2** (int \**address*, long *stride*, int32x3xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_int64x3xm1** (long \**address*, long *stride*, int64x3xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_int64x3xm2** (long \**address*, long *stride*, int64x3xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_int8x3xm1** (signed char \**address*, long *stride*, int8x3xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_int8x3xm2** (signed char \**address*, long *stride*, int8x3xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_uint16x3xm1** (unsigned short \**address*, long *stride*, uint16x3xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_uint16x3xm2** (unsigned short \**address*, long *stride*, uint16x3xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_uint32x3xm1** (unsigned int \**address*, long *stride*, uint32x3xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_uint32x3xm2** (unsigned int \**address*, long *stride*, uint32x3xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_uint64x3xm1** (unsigned long \**address*, long *stride*, uint64x3xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)

- void **vssseg3ev\_mask\_uint64x3xm2** (unsigned long *\*address*, long *stride*, uint64x3xm2\_t *a*, e64xm2\_t *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_uint8x3xm1** (unsigned char *\*address*, long *stride*, uint8x3xm1\_t *a*, e8xm1\_t *mask*, unsigned int *gvl*)
- void **vssseg3ev\_mask\_uint8x3xm2** (unsigned char *\*address*, long *stride*, uint8x3xm2\_t *a*, e8xm2\_t *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.182 Store 3 contiguous 16b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** ['vssseg3h.v']

**Prototypes:**

- void **vssseg3hv\_int16x3xm1** (short *\*address*, long *stride*, int16x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_int16x3xm2** (short *\*address*, long *stride*, int16x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_int32x3xm1** (int *\*address*, long *stride*, int32x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_int32x3xm2** (int *\*address*, long *stride*, int32x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_int64x3xm1** (long *\*address*, long *stride*, int64x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_int64x3xm2** (long *\*address*, long *stride*, int64x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_int8x3xm1** (signed char *\*address*, long *stride*, int8x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_int8x3xm2** (signed char *\*address*, long *stride*, int8x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_uint16x3xm1** (unsigned short *\*address*, long *stride*, uint16x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_uint16x3xm2** (unsigned short *\*address*, long *stride*, uint16x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_uint32x3xm1** (unsigned int *\*address*, long *stride*, uint32x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_uint32x3xm2** (unsigned int *\*address*, long *stride*, uint32x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_uint64x3xm1** (unsigned long *\*address*, long *stride*, uint64x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_uint64x3xm2** (unsigned long *\*address*, long *stride*, uint64x3xm2\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_uint8x3xm1** (unsigned char *\*address*, long *stride*, uint8x3xm1\_t *a*, unsigned int *gvl*)
- void **vssseg3hv\_uint8x3xm2** (unsigned char *\*address*, long *stride*, uint8x3xm2\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

#### Masked prototypes:

- void **vssseg3hv\_mask\_int16x3xm1** (short \*address, long stride, int16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_int16x3xm2** (short \*address, long stride, int16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_int32x3xm1** (int \*address, long stride, int32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_int32x3xm2** (int \*address, long stride, int32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_int64x3xm1** (long \*address, long stride, int64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_int64x3xm2** (long \*address, long stride, int64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_int8x3xm1** (signed char \*address, long stride, int8x3xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_int8x3xm2** (signed char \*address, long stride, int8x3xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_uint16x3xm1** (unsigned short \*address, long stride, uint16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_uint16x3xm2** (unsigned short \*address, long stride, uint16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_uint32x3xm1** (unsigned int \*address, long stride, uint32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_uint32x3xm2** (unsigned int \*address, long stride, uint32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_uint64x3xm1** (unsigned long \*address, long stride, uint64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_uint64x3xm2** (unsigned long \*address, long stride, uint64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_uint8x3xm1** (unsigned char \*address, long stride, uint8x3xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vssseg3hv\_mask\_uint8x3xm2** (unsigned char \*address, long stride, uint8x3xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)

#### Masked operation:

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```



## 2.10.183 Store 3 contiguous 32b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [`'vssseg3w.v'`]

**Prototypes:**

- void `vssseg3wv_int16x3xm1` (short *\*address*, long *stride*, int16x3xm1\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_int16x3xm2` (short *\*address*, long *stride*, int16x3xm2\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_int32x3xm1` (int *\*address*, long *stride*, int32x3xm1\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_int32x3xm2` (int *\*address*, long *stride*, int32x3xm2\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_int64x3xm1` (long *\*address*, long *stride*, int64x3xm1\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_int64x3xm2` (long *\*address*, long *stride*, int64x3xm2\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_int8x3xm1` (signed char *\*address*, long *stride*, int8x3xm1\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_int8x3xm2` (signed char *\*address*, long *stride*, int8x3xm2\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_uint16x3xm1` (unsigned short *\*address*, long *stride*, uint16x3xm1\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_uint16x3xm2` (unsigned short *\*address*, long *stride*, uint16x3xm2\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_uint32x3xm1` (unsigned int *\*address*, long *stride*, uint32x3xm1\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_uint32x3xm2` (unsigned int *\*address*, long *stride*, uint32x3xm2\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_uint64x3xm1` (unsigned long *\*address*, long *stride*, uint64x3xm1\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_uint64x3xm2` (unsigned long *\*address*, long *stride*, uint64x3xm2\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_uint8x3xm1` (unsigned char *\*address*, long *stride*, uint8x3xm1\_t *a*, unsigned int *gvl*)
- void `vssseg3wv_uint8x3xm2` (unsigned char *\*address*, long *stride*, uint8x3xm2\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void `vssseg3wv_mask_int16x3xm1` (short *\*address*, long *stride*, int16x3xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void `vssseg3wv_mask_int16x3xm2` (short *\*address*, long *stride*, int16x3xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void `vssseg3wv_mask_int32x3xm1` (int *\*address*, long *stride*, int32x3xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void `vssseg3wv_mask_int32x3xm2` (int *\*address*, long *stride*, int32x3xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)

- void **vssseg3wv\_mask\_int64x3xm1** (long *\*address*, long *stride*, int64x3xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3wv\_mask\_int64x3xm2** (long *\*address*, long *stride*, int64x3xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3wv\_mask\_int8x3xm1** (signed char *\*address*, long *stride*, int8x3xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3wv\_mask\_int8x3xm2** (signed char *\*address*, long *stride*, int8x3xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3wv\_mask\_uint16x3xm1** (unsigned short *\*address*, long *stride*, uint16x3xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3wv\_mask\_uint16x3xm2** (unsigned short *\*address*, long *stride*, uint16x3xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3wv\_mask\_uint32x3xm1** (unsigned int *\*address*, long *stride*, uint32x3xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3wv\_mask\_uint32x3xm2** (unsigned int *\*address*, long *stride*, uint32x3xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3wv\_mask\_uint64x3xm1** (unsigned long *\*address*, long *stride*, uint64x3xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3wv\_mask\_uint64x3xm2** (unsigned long *\*address*, long *stride*, uint64x3xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg3wv\_mask\_uint8x3xm1** (unsigned char *\*address*, long *stride*, uint8x3xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg3wv\_mask\_uint8x3xm2** (unsigned char *\*address*, long *stride*, uint8x3xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

**2.10.184 Store 4 contiguous 8b fields in memory(strided) from consecutively numbered vector registers****Instruction:** [*'vssseg4b.v'*]**Prototypes:**

- void **vssseg4bv\_int16x4xm1** (short *\*address*, long *stride*, int16x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_int16x4xm2** (short *\*address*, long *stride*, int16x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_int32x4xm1** (int *\*address*, long *stride*, int32x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_int32x4xm2** (int *\*address*, long *stride*, int32x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_int64x4xm1** (long *\*address*, long *stride*, int64x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_int64x4xm2** (long *\*address*, long *stride*, int64x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_int8x4xm1** (signed char *\*address*, long *stride*, int8x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_int8x4xm2** (signed char *\*address*, long *stride*, int8x4xm2\_t *a*, unsigned int *gvl*)

- void **vssseg4bv\_uint16x4xm1** (unsigned short *\*address*, long *stride*, uint16x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_uint16x4xm2** (unsigned short *\*address*, long *stride*, uint16x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_uint32x4xm1** (unsigned int *\*address*, long *stride*, uint32x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_uint32x4xm2** (unsigned int *\*address*, long *stride*, uint32x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_uint64x4xm1** (unsigned long *\*address*, long *stride*, uint64x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_uint64x4xm2** (unsigned long *\*address*, long *stride*, uint64x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_uint8x4xm1** (unsigned char *\*address*, long *stride*, uint8x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4bv\_uint8x4xm2** (unsigned char *\*address*, long *stride*, uint8x4xm2\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg4bv\_mask\_int16x4xm1** (short *\*address*, long *stride*, int16x4xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_int16x4xm2** (short *\*address*, long *stride*, int16x4xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_int32x4xm1** (int *\*address*, long *stride*, int32x4xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_int32x4xm2** (int *\*address*, long *stride*, int32x4xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_int64x4xm1** (long *\*address*, long *stride*, int64x4xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_int64x4xm2** (long *\*address*, long *stride*, int64x4xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_int8x4xm1** (signed char *\*address*, long *stride*, int8x4xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_int8x4xm2** (signed char *\*address*, long *stride*, int8x4xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_uint16x4xm1** (unsigned short *\*address*, long *stride*, uint16x4xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_uint16x4xm2** (unsigned short *\*address*, long *stride*, uint16x4xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_uint32x4xm1** (unsigned int *\*address*, long *stride*, uint32x4xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_uint32x4xm2** (unsigned int *\*address*, long *stride*, uint32x4xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)

- void **vssseg4bv\_mask\_uint64x4xm1** (unsigned long *\*address*, long *stride*, uint64x4xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_uint64x4xm2** (unsigned long *\*address*, long *stride*, uint64x4xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_uint8x4xm1** (unsigned char *\*address*, long *stride*, uint8x4xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4bv\_mask\_uint8x4xm2** (unsigned char *\*address*, long *stride*, uint8x4xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.185 Store 4 contiguous element fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [*vssseg4e.v*]

**Prototypes:**

- void **vssseg4ev\_float16x4xm1** (float16\_t *\*address*, long *stride*, float16x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_float16x4xm2** (float16\_t *\*address*, long *stride*, float16x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_float32x4xm1** (float *\*address*, long *stride*, float32x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_float32x4xm2** (float *\*address*, long *stride*, float32x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_float64x4xm1** (double *\*address*, long *stride*, float64x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_float64x4xm2** (double *\*address*, long *stride*, float64x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_int16x4xm1** (short *\*address*, long *stride*, int16x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_int16x4xm2** (short *\*address*, long *stride*, int16x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_int32x4xm1** (int *\*address*, long *stride*, int32x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_int32x4xm2** (int *\*address*, long *stride*, int32x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_int64x4xm1** (long *\*address*, long *stride*, int64x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_int64x4xm2** (long *\*address*, long *stride*, int64x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_int8x4xm1** (signed char *\*address*, long *stride*, int8x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_int8x4xm2** (signed char *\*address*, long *stride*, int8x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_uint16x4xm1** (unsigned short *\*address*, long *stride*, uint16x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_uint16x4xm2** (unsigned short *\*address*, long *stride*, uint16x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_uint32x4xm1** (unsigned int *\*address*, long *stride*, uint32x4xm1\_t *a*, unsigned int *gvl*)

- void **vssseg4ev\_uint32x4xm2** (unsigned int *\*address*, long *stride*, uint32x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_uint64x4xm1** (unsigned long *\*address*, long *stride*, uint64x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_uint64x4xm2** (unsigned long *\*address*, long *stride*, uint64x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_uint8x4xm1** (unsigned char *\*address*, long *stride*, uint8x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4ev\_uint8x4xm2** (unsigned char *\*address*, long *stride*, uint8x4xm2\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg4ev\_mask\_float16x4xm1** (float16\_t *\*address*, long *stride*, float16x4xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_float16x4xm2** (float16\_t *\*address*, long *stride*, float16x4xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_float32x4xm1** (float *\*address*, long *stride*, float32x4xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_float32x4xm2** (float *\*address*, long *stride*, float32x4xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_float64x4xm1** (double *\*address*, long *stride*, float64x4xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_float64x4xm2** (double *\*address*, long *stride*, float64x4xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_int16x4xm1** (short *\*address*, long *stride*, int16x4xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_int16x4xm2** (short *\*address*, long *stride*, int16x4xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_int32x4xm1** (int *\*address*, long *stride*, int32x4xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_int32x4xm2** (int *\*address*, long *stride*, int32x4xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_int64x4xm1** (long *\*address*, long *stride*, int64x4xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_int64x4xm2** (long *\*address*, long *stride*, int64x4xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_int8x4xm1** (signed char *\*address*, long *stride*, int8x4xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_int8x4xm2** (signed char *\*address*, long *stride*, int8x4xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_uint16x4xm1** (unsigned short *\*address*, long *stride*, uint16x4xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)

- void **vssseg4ev\_mask\_uint16x4xm2** (unsigned short *\*address*, long *stride*, uint16x4xm2\_t *a*, *e16xm2\_t mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_uint32x4xm1** (unsigned int *\*address*, long *stride*, uint32x4xm1\_t *a*, *e32xm1\_t mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_uint32x4xm2** (unsigned int *\*address*, long *stride*, uint32x4xm2\_t *a*, *e32xm2\_t mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_uint64x4xm1** (unsigned long *\*address*, long *stride*, uint64x4xm1\_t *a*, *e64xm1\_t mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_uint64x4xm2** (unsigned long *\*address*, long *stride*, uint64x4xm2\_t *a*, *e64xm2\_t mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_uint8x4xm1** (unsigned char *\*address*, long *stride*, uint8x4xm1\_t *a*, *e8xm1\_t mask*, unsigned int *gvl*)
- void **vssseg4ev\_mask\_uint8x4xm2** (unsigned char *\*address*, long *stride*, uint8x4xm2\_t *a*, *e8xm2\_t mask*, unsigned int *gvl*)

Masked operation:

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.186 Store 4 contiguous 16b fields in memory(strided) from consecutively numbered vector registers

Instruction: ['vssseg4h.v']

Prototypes:

- void **vssseg4hv\_int16x4xm1** (short *\*address*, long *stride*, int16x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_int16x4xm2** (short *\*address*, long *stride*, int16x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_int32x4xm1** (int *\*address*, long *stride*, int32x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_int32x4xm2** (int *\*address*, long *stride*, int32x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_int64x4xm1** (long *\*address*, long *stride*, int64x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_int64x4xm2** (long *\*address*, long *stride*, int64x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_int8x4xm1** (signed char *\*address*, long *stride*, int8x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_int8x4xm2** (signed char *\*address*, long *stride*, int8x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_uint16x4xm1** (unsigned short *\*address*, long *stride*, uint16x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_uint16x4xm2** (unsigned short *\*address*, long *stride*, uint16x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_uint32x4xm1** (unsigned int *\*address*, long *stride*, uint32x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_uint32x4xm2** (unsigned int *\*address*, long *stride*, uint32x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_uint64x4xm1** (unsigned long *\*address*, long *stride*, uint64x4xm1\_t *a*, unsigned int *gvl*)



- void **vssseg4hv\_uint64x4xm2** (unsigned long *\*address*, long *stride*, uint64x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_uint8x4xm1** (unsigned char *\*address*, long *stride*, uint8x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4hv\_uint8x4xm2** (unsigned char *\*address*, long *stride*, uint8x4xm2\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg4hv\_mask\_int16x4xm1** (short *\*address*, long *stride*, int16x4xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_int16x4xm2** (short *\*address*, long *stride*, int16x4xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_int32x4xm1** (int *\*address*, long *stride*, int32x4xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_int32x4xm2** (int *\*address*, long *stride*, int32x4xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_int64x4xm1** (long *\*address*, long *stride*, int64x4xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_int64x4xm2** (long *\*address*, long *stride*, int64x4xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_int8x4xm1** (signed char *\*address*, long *stride*, int8x4xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_int8x4xm2** (signed char *\*address*, long *stride*, int8x4xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_uint16x4xm1** (unsigned short *\*address*, long *stride*, uint16x4xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_uint16x4xm2** (unsigned short *\*address*, long *stride*, uint16x4xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_uint32x4xm1** (unsigned int *\*address*, long *stride*, uint32x4xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_uint32x4xm2** (unsigned int *\*address*, long *stride*, uint32x4xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_uint64x4xm1** (unsigned long *\*address*, long *stride*, uint64x4xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_uint64x4xm2** (unsigned long *\*address*, long *stride*, uint64x4xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_uint8x4xm1** (unsigned char *\*address*, long *stride*, uint8x4xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4hv\_mask\_uint8x4xm2** (unsigned char *\*address*, long *stride*, uint8x4xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

### 2.10.187 Store 4 contiguous 32b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [`'vssseg4w.v'`]

**Prototypes:**

- void **vssseg4wv\_int16x4xm1** (short *\*address*, long *stride*, int16x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_int16x4xm2** (short *\*address*, long *stride*, int16x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_int32x4xm1** (int *\*address*, long *stride*, int32x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_int32x4xm2** (int *\*address*, long *stride*, int32x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_int64x4xm1** (long *\*address*, long *stride*, int64x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_int64x4xm2** (long *\*address*, long *stride*, int64x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_int8x4xm1** (signed char *\*address*, long *stride*, int8x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_int8x4xm2** (signed char *\*address*, long *stride*, int8x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_uint16x4xm1** (unsigned short *\*address*, long *stride*, uint16x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_uint16x4xm2** (unsigned short *\*address*, long *stride*, uint16x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_uint32x4xm1** (unsigned int *\*address*, long *stride*, uint32x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_uint32x4xm2** (unsigned int *\*address*, long *stride*, uint32x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_uint64x4xm1** (unsigned long *\*address*, long *stride*, uint64x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_uint64x4xm2** (unsigned long *\*address*, long *stride*, uint64x4xm2\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_uint8x4xm1** (unsigned char *\*address*, long *stride*, uint8x4xm1\_t *a*, unsigned int *gvl*)
- void **vssseg4wv\_uint8x4xm2** (unsigned char *\*address*, long *stride*, uint8x4xm2\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg4wv\_mask\_int16x4xm1** (short *\*address*, long *stride*, int16x4xm1\_t *a*, *e16xm1\_t mask*, unsigned int *gvl*)

- void **vssseg4wv\_mask\_int16x4xm2** (short *\*address*, long *stride*, int16x4xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_int32x4xm1** (int *\*address*, long *stride*, int32x4xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_int32x4xm2** (int *\*address*, long *stride*, int32x4xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_int64x4xm1** (long *\*address*, long *stride*, int64x4xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_int64x4xm2** (long *\*address*, long *stride*, int64x4xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_int8x4xm1** (signed char *\*address*, long *stride*, int8x4xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_int8x4xm2** (signed char *\*address*, long *stride*, int8x4xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_uint16x4xm1** (unsigned short *\*address*, long *stride*, uint16x4xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_uint16x4xm2** (unsigned short *\*address*, long *stride*, uint16x4xm2\_t *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_uint32x4xm1** (unsigned int *\*address*, long *stride*, uint32x4xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_uint32x4xm2** (unsigned int *\*address*, long *stride*, uint32x4xm2\_t *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_uint64x4xm1** (unsigned long *\*address*, long *stride*, uint64x4xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_uint64x4xm2** (unsigned long *\*address*, long *stride*, uint64x4xm2\_t *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_uint8x4xm1** (unsigned char *\*address*, long *stride*, uint8x4xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg4wv\_mask\_uint8x4xm2** (unsigned char *\*address*, long *stride*, uint8x4xm2\_t *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.188 Store 5 contiguous 8b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [*'vssseg5b.v'*]

**Prototypes:**

- void **vssseg5bv\_int16x5xm1** (short *\*address*, long *stride*, int16x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5bv\_int32x5xm1** (int *\*address*, long *stride*, int32x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5bv\_int64x5xm1** (long *\*address*, long *stride*, int64x5xm1\_t *a*, unsigned int *gvl*)

- void **vssseg5bv\_int8x5xm1** (signed char *\*address*, long *stride*, int8x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5bv\_uint16x5xm1** (unsigned short *\*address*, long *stride*, uint16x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5bv\_uint32x5xm1** (unsigned int *\*address*, long *stride*, uint32x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5bv\_uint64x5xm1** (unsigned long *\*address*, long *stride*, uint64x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5bv\_uint8x5xm1** (unsigned char *\*address*, long *stride*, uint8x5xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg5bv\_mask\_int16x5xm1** (short *\*address*, long *stride*, int16x5xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5bv\_mask\_int32x5xm1** (int *\*address*, long *stride*, int32x5xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5bv\_mask\_int64x5xm1** (long *\*address*, long *stride*, int64x5xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5bv\_mask\_int8x5xm1** (signed char *\*address*, long *stride*, int8x5xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5bv\_mask\_uint16x5xm1** (unsigned short *\*address*, long *stride*, uint16x5xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5bv\_mask\_uint32x5xm1** (unsigned int *\*address*, long *stride*, uint32x5xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5bv\_mask\_uint64x5xm1** (unsigned long *\*address*, long *stride*, uint64x5xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5bv\_mask\_uint8x5xm1** (unsigned char *\*address*, long *stride*, uint8x5xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.189 Store 5 contiguous element fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [*vssseg5e.v*']

**Prototypes:**

- void **vssseg5ev\_float16x5xm1** (float16\_t *\*address*, long *stride*, float16x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5ev\_float32x5xm1** (float *\*address*, long *stride*, float32x5xm1\_t *a*, unsigned int *gvl*)

- void **vssseg5ev\_float64x5xm1** (double *\*address*, long *stride*, float64x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5ev\_int16x5xm1** (short *\*address*, long *stride*, int16x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5ev\_int32x5xm1** (int *\*address*, long *stride*, int32x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5ev\_int64x5xm1** (long *\*address*, long *stride*, int64x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5ev\_int8x5xm1** (signed char *\*address*, long *stride*, int8x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5ev\_uint16x5xm1** (unsigned short *\*address*, long *stride*, uint16x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5ev\_uint32x5xm1** (unsigned int *\*address*, long *stride*, uint32x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5ev\_uint64x5xm1** (unsigned long *\*address*, long *stride*, uint64x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5ev\_uint8x5xm1** (unsigned char *\*address*, long *stride*, uint8x5xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg5ev\_mask\_float16x5xm1** (float16\_t *\*address*, long *stride*, float16x5xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5ev\_mask\_float32x5xm1** (float *\*address*, long *stride*, float32x5xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5ev\_mask\_float64x5xm1** (double *\*address*, long *stride*, float64x5xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5ev\_mask\_int16x5xm1** (short *\*address*, long *stride*, int16x5xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5ev\_mask\_int32x5xm1** (int *\*address*, long *stride*, int32x5xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5ev\_mask\_int64x5xm1** (long *\*address*, long *stride*, int64x5xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5ev\_mask\_int8x5xm1** (signed char *\*address*, long *stride*, int8x5xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5ev\_mask\_uint16x5xm1** (unsigned short *\*address*, long *stride*, uint16x5xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5ev\_mask\_uint32x5xm1** (unsigned int *\*address*, long *stride*, uint32x5xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5ev\_mask\_uint64x5xm1** (unsigned long *\*address*, long *stride*, uint64x5xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5ev\_mask\_uint8x5xm1** (unsigned char *\*address*, long *stride*, uint8x5xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

### 2.10.190 Store 5 contiguous 16b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [`'vssseg5h.v'`]

**Prototypes:**

- void **vssseg5hv\_int16x5xm1** (short *\*address*, long *stride*, int16x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5hv\_int32x5xm1** (int *\*address*, long *stride*, int32x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5hv\_int64x5xm1** (long *\*address*, long *stride*, int64x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5hv\_int8x5xm1** (signed char *\*address*, long *stride*, int8x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5hv\_uint16x5xm1** (unsigned short *\*address*, long *stride*, uint16x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5hv\_uint32x5xm1** (unsigned int *\*address*, long *stride*, uint32x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5hv\_uint64x5xm1** (unsigned long *\*address*, long *stride*, uint64x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5hv\_uint8x5xm1** (unsigned char *\*address*, long *stride*, uint8x5xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg5hv\_mask\_int16x5xm1** (short *\*address*, long *stride*, int16x5xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5hv\_mask\_int32x5xm1** (int *\*address*, long *stride*, int32x5xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5hv\_mask\_int64x5xm1** (long *\*address*, long *stride*, int64x5xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5hv\_mask\_int8x5xm1** (signed char *\*address*, long *stride*, int8x5xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5hv\_mask\_uint16x5xm1** (unsigned short *\*address*, long *stride*, uint16x5xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5hv\_mask\_uint32x5xm1** (unsigned int *\*address*, long *stride*, uint32x5xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5hv\_mask\_uint64x5xm1** (unsigned long *\*address*, long *stride*, uint64x5xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5hv\_mask\_uint8x5xm1** (unsigned char *\*address*, long *stride*, uint8x5xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)



**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

**2.10.191 Store 5 contiguous 32b fields in memory(strided) from consecutively numbered vector registers****Instruction:** [`'vssseg5w.v'`]**Prototypes:**

- void **vssseg5wv\_int16x5xm1** (short *\*address*, long *stride*, int16x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5wv\_int32x5xm1** (int *\*address*, long *stride*, int32x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5wv\_int64x5xm1** (long *\*address*, long *stride*, int64x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5wv\_int8x5xm1** (signed char *\*address*, long *stride*, int8x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5wv\_uint16x5xm1** (unsigned short *\*address*, long *stride*, uint16x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5wv\_uint32x5xm1** (unsigned int *\*address*, long *stride*, uint32x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5wv\_uint64x5xm1** (unsigned long *\*address*, long *stride*, uint64x5xm1\_t *a*, unsigned int *gvl*)
- void **vssseg5wv\_uint8x5xm1** (unsigned char *\*address*, long *stride*, uint8x5xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg5wv\_mask\_int16x5xm1** (short *\*address*, long *stride*, int16x5xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5wv\_mask\_int32x5xm1** (int *\*address*, long *stride*, int32x5xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5wv\_mask\_int64x5xm1** (long *\*address*, long *stride*, int64x5xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5wv\_mask\_int8x5xm1** (signed char *\*address*, long *stride*, int8x5xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5wv\_mask\_uint16x5xm1** (unsigned short *\*address*, long *stride*, uint16x5xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5wv\_mask\_uint32x5xm1** (unsigned int *\*address*, long *stride*, uint32x5xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5wv\_mask\_uint64x5xm1** (unsigned long *\*address*, long *stride*, uint64x5xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg5wv\_mask\_uint8x5xm1** (unsigned char *\*address*, long *stride*, uint8x5xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

**2.10.192 Store 6 contiguous 8b fields in memory(strided) from consecutively numbered vector registers****Instruction:** ['vssseg6b.v']**Prototypes:**

- void **vssseg6bv\_int16x6xm1** (short \*address, long stride, int16x6xm1\_t a, unsigned int gvl)
- void **vssseg6bv\_int32x6xm1** (int \*address, long stride, int32x6xm1\_t a, unsigned int gvl)
- void **vssseg6bv\_int64x6xm1** (long \*address, long stride, int64x6xm1\_t a, unsigned int gvl)
- void **vssseg6bv\_int8x6xm1** (signed char \*address, long stride, int8x6xm1\_t a, unsigned int gvl)
- void **vssseg6bv\_uint16x6xm1** (unsigned short \*address, long stride, uint16x6xm1\_t a, unsigned int gvl)
- void **vssseg6bv\_uint32x6xm1** (unsigned int \*address, long stride, uint32x6xm1\_t a, unsigned int gvl)
- void **vssseg6bv\_uint64x6xm1** (unsigned long \*address, long stride, uint64x6xm1\_t a, unsigned int gvl)
- void **vssseg6bv\_uint8x6xm1** (unsigned char \*address, long stride, uint8x6xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg6bv\_mask\_int16x6xm1** (short \*address, long stride, int16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg6bv\_mask\_int32x6xm1** (int \*address, long stride, int32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg6bv\_mask\_int64x6xm1** (long \*address, long stride, int64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg6bv\_mask\_int8x6xm1** (signed char \*address, long stride, int8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vssseg6bv\_mask\_uint16x6xm1** (unsigned short \*address, long stride, uint16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg6bv\_mask\_uint32x6xm1** (unsigned int \*address, long stride, uint32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg6bv\_mask\_uint64x6xm1** (unsigned long \*address, long stride, uint64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg6bv\_mask\_uint8x6xm1** (unsigned char \*address, long stride, uint8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

**2.10.193 Store 6 contiguous element fields in memory(strided) from consecutively numbered vector registers****Instruction:** ['vssseg6e.v']**Prototypes:**

- void **vssseg6ev\_float16x6xm1** (float16\_t \*address, long stride, float16x6xm1\_t a, unsigned int gvl)
- void **vssseg6ev\_float32x6xm1** (float \*address, long stride, float32x6xm1\_t a, unsigned int gvl)
- void **vssseg6ev\_float64x6xm1** (double \*address, long stride, float64x6xm1\_t a, unsigned int gvl)
- void **vssseg6ev\_int16x6xm1** (short \*address, long stride, int16x6xm1\_t a, unsigned int gvl)
- void **vssseg6ev\_int32x6xm1** (int \*address, long stride, int32x6xm1\_t a, unsigned int gvl)
- void **vssseg6ev\_int64x6xm1** (long \*address, long stride, int64x6xm1\_t a, unsigned int gvl)
- void **vssseg6ev\_int8x6xm1** (signed char \*address, long stride, int8x6xm1\_t a, unsigned int gvl)
- void **vssseg6ev\_uint16x6xm1** (unsigned short \*address, long stride, uint16x6xm1\_t a, unsigned int gvl)
- void **vssseg6ev\_uint32x6xm1** (unsigned int \*address, long stride, uint32x6xm1\_t a, unsigned int gvl)
- void **vssseg6ev\_uint64x6xm1** (unsigned long \*address, long stride, uint64x6xm1\_t a, unsigned int gvl)
- void **vssseg6ev\_uint8x6xm1** (unsigned char \*address, long stride, uint8x6xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg6ev\_mask\_float16x6xm1** (float16\_t \*address, long stride, float16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg6ev\_mask\_float32x6xm1** (float \*address, long stride, float32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg6ev\_mask\_float64x6xm1** (double \*address, long stride, float64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg6ev\_mask\_int16x6xm1** (short \*address, long stride, int16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg6ev\_mask\_int32x6xm1** (int \*address, long stride, int32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)

- void **vssseg6ev\_mask\_int64x6xm1** (long *\*address*, long *stride*, int64x6xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6ev\_mask\_int8x6xm1** (signed char *\*address*, long *stride*, int8x6xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6ev\_mask\_uint16x6xm1** (unsigned short *\*address*, long *stride*, uint16x6xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6ev\_mask\_uint32x6xm1** (unsigned int *\*address*, long *stride*, uint32x6xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6ev\_mask\_uint64x6xm1** (unsigned long *\*address*, long *stride*, uint64x6xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6ev\_mask\_uint8x6xm1** (unsigned char *\*address*, long *stride*, uint8x6xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

### 2.10.194 Store 6 contiguous 16b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [*'vssseg6h.v'*]

**Prototypes:**

- void **vssseg6hv\_int16x6xm1** (short *\*address*, long *stride*, int16x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6hv\_int32x6xm1** (int *\*address*, long *stride*, int32x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6hv\_int64x6xm1** (long *\*address*, long *stride*, int64x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6hv\_int8x6xm1** (signed char *\*address*, long *stride*, int8x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6hv\_uint16x6xm1** (unsigned short *\*address*, long *stride*, uint16x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6hv\_uint32x6xm1** (unsigned int *\*address*, long *stride*, uint32x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6hv\_uint64x6xm1** (unsigned long *\*address*, long *stride*, uint64x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6hv\_uint8x6xm1** (unsigned char *\*address*, long *stride*, uint8x6xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg6hv\_mask\_int16x6xm1** (short *\*address*, long *stride*, int16x6xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6hv\_mask\_int32x6xm1** (int *\*address*, long *stride*, int32x6xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)

- void **vssseg6hv\_mask\_int64x6xm1** (long *\*address*, long *stride*, int64x6xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6hv\_mask\_int8x6xm1** (signed char *\*address*, long *stride*, int8x6xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6hv\_mask\_uint16x6xm1** (unsigned short *\*address*, long *stride*, uint16x6xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6hv\_mask\_uint32x6xm1** (unsigned int *\*address*, long *stride*, uint32x6xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6hv\_mask\_uint64x6xm1** (unsigned long *\*address*, long *stride*, uint64x6xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6hv\_mask\_uint8x6xm1** (unsigned char *\*address*, long *stride*, uint8x6xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.195 Store 6 contiguous 32b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [*vssseg6w.v*]

**Prototypes:**

- void **vssseg6wv\_int16x6xm1** (short *\*address*, long *stride*, int16x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6wv\_int32x6xm1** (int *\*address*, long *stride*, int32x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6wv\_int64x6xm1** (long *\*address*, long *stride*, int64x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6wv\_int8x6xm1** (signed char *\*address*, long *stride*, int8x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6wv\_uint16x6xm1** (unsigned short *\*address*, long *stride*, uint16x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6wv\_uint32x6xm1** (unsigned int *\*address*, long *stride*, uint32x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6wv\_uint64x6xm1** (unsigned long *\*address*, long *stride*, uint64x6xm1\_t *a*, unsigned int *gvl*)
- void **vssseg6wv\_uint8x6xm1** (unsigned char *\*address*, long *stride*, uint8x6xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg6wv\_mask\_int16x6xm1** (short *\*address*, long *stride*, int16x6xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6wv\_mask\_int32x6xm1** (int *\*address*, long *stride*, int32x6xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)

- void **vssseg6wv\_mask\_int64x6xm1** (long *\*address*, long *stride*, int64x6xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6wv\_mask\_int8x6xm1** (signed char *\*address*, long *stride*, int8x6xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6wv\_mask\_uint16x6xm1** (unsigned short *\*address*, long *stride*, uint16x6xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6wv\_mask\_uint32x6xm1** (unsigned int *\*address*, long *stride*, uint32x6xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6wv\_mask\_uint64x6xm1** (unsigned long *\*address*, long *stride*, uint64x6xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg6wv\_mask\_uint8x6xm1** (unsigned char *\*address*, long *stride*, uint8x6xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

**2.10.196 Store 7 contiguous 8b fields in memory(strided) from consecutively numbered vector registers****Instruction:** [*vssseg7b.v'*]**Prototypes:**

- void **vssseg7bv\_int16x7xm1** (short *\*address*, long *stride*, int16x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7bv\_int32x7xm1** (int *\*address*, long *stride*, int32x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7bv\_int64x7xm1** (long *\*address*, long *stride*, int64x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7bv\_int8x7xm1** (signed char *\*address*, long *stride*, int8x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7bv\_uint16x7xm1** (unsigned short *\*address*, long *stride*, uint16x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7bv\_uint32x7xm1** (unsigned int *\*address*, long *stride*, uint32x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7bv\_uint64x7xm1** (unsigned long *\*address*, long *stride*, uint64x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7bv\_uint8x7xm1** (unsigned char *\*address*, long *stride*, uint8x7xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg7bv\_mask\_int16x7xm1** (short *\*address*, long *stride*, int16x7xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg7bv\_mask\_int32x7xm1** (int *\*address*, long *stride*, int32x7xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)



- void **vssseg7bv\_mask\_int64x7xm1** (long *\*address*, long *stride*, int64x7xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg7bv\_mask\_int8x7xm1** (signed char *\*address*, long *stride*, int8x7xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg7bv\_mask\_uint16x7xm1** (unsigned short *\*address*, long *stride*, uint16x7xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg7bv\_mask\_uint32x7xm1** (unsigned int *\*address*, long *stride*, uint32x7xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg7bv\_mask\_uint64x7xm1** (unsigned long *\*address*, long *stride*, uint64x7xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg7bv\_mask\_uint8x7xm1** (unsigned char *\*address*, long *stride*, uint8x7xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.197 Store 7 contiguous element fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [*'vssseg7e.v'*]

**Prototypes:**

- void **vssseg7ev\_float16x7xm1** (float16\_t *\*address*, long *stride*, float16x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7ev\_float32x7xm1** (float *\*address*, long *stride*, float32x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7ev\_float64x7xm1** (double *\*address*, long *stride*, float64x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7ev\_int16x7xm1** (short *\*address*, long *stride*, int16x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7ev\_int32x7xm1** (int *\*address*, long *stride*, int32x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7ev\_int64x7xm1** (long *\*address*, long *stride*, int64x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7ev\_int8x7xm1** (signed char *\*address*, long *stride*, int8x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7ev\_uint16x7xm1** (unsigned short *\*address*, long *stride*, uint16x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7ev\_uint32x7xm1** (unsigned int *\*address*, long *stride*, uint32x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7ev\_uint64x7xm1** (unsigned long *\*address*, long *stride*, uint64x7xm1\_t *a*, unsigned int *gvl*)
- void **vssseg7ev\_uint8x7xm1** (unsigned char *\*address*, long *stride*, uint8x7xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg7ev\_mask\_float16x7xm1** (float16\_t \*address, long stride, float16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg7ev\_mask\_float32x7xm1** (float \*address, long stride, float32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg7ev\_mask\_float64x7xm1** (double \*address, long stride, float64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg7ev\_mask\_int16x7xm1** (short \*address, long stride, int16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg7ev\_mask\_int32x7xm1** (int \*address, long stride, int32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg7ev\_mask\_int64x7xm1** (long \*address, long stride, int64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg7ev\_mask\_int8x7xm1** (signed char \*address, long stride, int8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vssseg7ev\_mask\_uint16x7xm1** (unsigned short \*address, long stride, uint16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg7ev\_mask\_uint32x7xm1** (unsigned int \*address, long stride, uint32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg7ev\_mask\_uint64x7xm1** (unsigned long \*address, long stride, uint64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg7ev\_mask\_uint8x7xm1** (unsigned char \*address, long stride, uint8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

**2.10.198 Store 7 contiguous 16b fields in memory(strided) from consecutively numbered vector registers****Instruction:** ['vssseg7h.v']**Prototypes:**

- void **vssseg7hv\_int16x7xm1** (short \*address, long stride, int16x7xm1\_t a, unsigned int gvl)
- void **vssseg7hv\_int32x7xm1** (int \*address, long stride, int32x7xm1\_t a, unsigned int gvl)
- void **vssseg7hv\_int64x7xm1** (long \*address, long stride, int64x7xm1\_t a, unsigned int gvl)
- void **vssseg7hv\_int8x7xm1** (signed char \*address, long stride, int8x7xm1\_t a, unsigned int gvl)
- void **vssseg7hv\_uint16x7xm1** (unsigned short \*address, long stride, uint16x7xm1\_t a, unsigned int gvl)
- void **vssseg7hv\_uint32x7xm1** (unsigned int \*address, long stride, uint32x7xm1\_t a, unsigned int gvl)
- void **vssseg7hv\_uint64x7xm1** (unsigned long \*address, long stride, uint64x7xm1\_t a, unsigned int gvl)

- void **vssseg7hv\_uint8x7xm1** (unsigned char \*address, long stride, uint8x7xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg7hv\_mask\_int16x7xm1** (short \*address, long stride, int16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg7hv\_mask\_int32x7xm1** (int \*address, long stride, int32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg7hv\_mask\_int64x7xm1** (long \*address, long stride, int64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg7hv\_mask\_int8x7xm1** (signed char \*address, long stride, int8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vssseg7hv\_mask\_uint16x7xm1** (unsigned short \*address, long stride, uint16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg7hv\_mask\_uint32x7xm1** (unsigned int \*address, long stride, uint32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg7hv\_mask\_uint64x7xm1** (unsigned long \*address, long stride, uint64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg7hv\_mask\_uint8x7xm1** (unsigned char \*address, long stride, uint8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.199 Store 7 contiguous 32b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [*vssseg7w.v*']

**Prototypes:**

- void **vssseg7wv\_int16x7xm1** (short \*address, long stride, int16x7xm1\_t a, unsigned int gvl)
- void **vssseg7wv\_int32x7xm1** (int \*address, long stride, int32x7xm1\_t a, unsigned int gvl)
- void **vssseg7wv\_int64x7xm1** (long \*address, long stride, int64x7xm1\_t a, unsigned int gvl)
- void **vssseg7wv\_int8x7xm1** (signed char \*address, long stride, int8x7xm1\_t a, unsigned int gvl)
- void **vssseg7wv\_uint16x7xm1** (unsigned short \*address, long stride, uint16x7xm1\_t a, unsigned int gvl)
- void **vssseg7wv\_uint32x7xm1** (unsigned int \*address, long stride, uint32x7xm1\_t a, unsigned int gvl)
- void **vssseg7wv\_uint64x7xm1** (unsigned long \*address, long stride, uint64x7xm1\_t a, unsigned int gvl)

- void **vssseg7wv\_uint8x7xm1** (unsigned char \*address, long stride, uint8x7xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg7wv\_mask\_int16x7xm1** (short \*address, long stride, int16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg7wv\_mask\_int32x7xm1** (int \*address, long stride, int32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg7wv\_mask\_int64x7xm1** (long \*address, long stride, int64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg7wv\_mask\_int8x7xm1** (signed char \*address, long stride, int8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vssseg7wv\_mask\_uint16x7xm1** (unsigned short \*address, long stride, uint16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg7wv\_mask\_uint32x7xm1** (unsigned int \*address, long stride, uint32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg7wv\_mask\_uint64x7xm1** (unsigned long \*address, long stride, uint64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg7wv\_mask\_uint8x7xm1** (unsigned char \*address, long stride, uint8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.200 Store 8 contiguous 8b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** ['vssseg8b.v']

**Prototypes:**

- void **vssseg8bv\_int16x8xm1** (short \*address, long stride, int16x8xm1\_t a, unsigned int gvl)
- void **vssseg8bv\_int32x8xm1** (int \*address, long stride, int32x8xm1\_t a, unsigned int gvl)
- void **vssseg8bv\_int64x8xm1** (long \*address, long stride, int64x8xm1\_t a, unsigned int gvl)
- void **vssseg8bv\_int8x8xm1** (signed char \*address, long stride, int8x8xm1\_t a, unsigned int gvl)
- void **vssseg8bv\_uint16x8xm1** (unsigned short \*address, long stride, uint16x8xm1\_t a, unsigned int gvl)
- void **vssseg8bv\_uint32x8xm1** (unsigned int \*address, long stride, uint32x8xm1\_t a, unsigned int gvl)
- void **vssseg8bv\_uint64x8xm1** (unsigned long \*address, long stride, uint64x8xm1\_t a, unsigned int gvl)

- void **vssseg8bv\_uint8x8xm1** (unsigned char \*address, long stride, uint8x8xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg8bv\_mask\_int16x8xm1** (short \*address, long stride, int16x8xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg8bv\_mask\_int32x8xm1** (int \*address, long stride, int32x8xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg8bv\_mask\_int64x8xm1** (long \*address, long stride, int64x8xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg8bv\_mask\_int8x8xm1** (signed char \*address, long stride, int8x8xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vssseg8bv\_mask\_uint16x8xm1** (unsigned short \*address, long stride, uint16x8xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vssseg8bv\_mask\_uint32x8xm1** (unsigned int \*address, long stride, uint32x8xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vssseg8bv\_mask\_uint64x8xm1** (unsigned long \*address, long stride, uint64x8xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vssseg8bv\_mask\_uint8x8xm1** (unsigned char \*address, long stride, uint8x8xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.201 Store 8 contiguous element fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [*vssseg8e.v*]

**Prototypes:**

- void **vssseg8ev\_float16x8xm1** (float16\_t \*address, long stride, float16x8xm1\_t a, unsigned int gvl)
- void **vssseg8ev\_float32x8xm1** (float \*address, long stride, float32x8xm1\_t a, unsigned int gvl)
- void **vssseg8ev\_float64x8xm1** (double \*address, long stride, float64x8xm1\_t a, unsigned int gvl)
- void **vssseg8ev\_int16x8xm1** (short \*address, long stride, int16x8xm1\_t a, unsigned int gvl)
- void **vssseg8ev\_int32x8xm1** (int \*address, long stride, int32x8xm1\_t a, unsigned int gvl)
- void **vssseg8ev\_int64x8xm1** (long \*address, long stride, int64x8xm1\_t a, unsigned int gvl)
- void **vssseg8ev\_int8x8xm1** (signed char \*address, long stride, int8x8xm1\_t a, unsigned int gvl)

- void **vssseg8ev\_uint16x8xm1** (unsigned short *\*address*, long *stride*, uint16x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8ev\_uint32x8xm1** (unsigned int *\*address*, long *stride*, uint32x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8ev\_uint64x8xm1** (unsigned long *\*address*, long *stride*, uint64x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8ev\_uint8x8xm1** (unsigned char *\*address*, long *stride*, uint8x8xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg8ev\_mask\_float16x8xm1** (float16\_t *\*address*, long *stride*, float16x8xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8ev\_mask\_float32x8xm1** (float *\*address*, long *stride*, float32x8xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8ev\_mask\_float64x8xm1** (double *\*address*, long *stride*, float64x8xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8ev\_mask\_int16x8xm1** (short *\*address*, long *stride*, int16x8xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8ev\_mask\_int32x8xm1** (int *\*address*, long *stride*, int32x8xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8ev\_mask\_int64x8xm1** (long *\*address*, long *stride*, int64x8xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8ev\_mask\_int8x8xm1** (signed char *\*address*, long *stride*, int8x8xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8ev\_mask\_uint16x8xm1** (unsigned short *\*address*, long *stride*, uint16x8xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8ev\_mask\_uint32x8xm1** (unsigned int *\*address*, long *stride*, uint32x8xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8ev\_mask\_uint64x8xm1** (unsigned long *\*address*, long *stride*, uint64x8xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8ev\_mask\_uint8x8xm1** (unsigned char *\*address*, long *stride*, uint8x8xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.202 Store 8 contiguous 16b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** [*'vssseg8h.v'*]



**Prototypes:**

- void **vssseg8hv\_int16x8xm1** (short *\*address*, long *stride*, int16x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8hv\_int32x8xm1** (int *\*address*, long *stride*, int32x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8hv\_int64x8xm1** (long *\*address*, long *stride*, int64x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8hv\_int8x8xm1** (signed char *\*address*, long *stride*, int8x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8hv\_uint16x8xm1** (unsigned short *\*address*, long *stride*, uint16x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8hv\_uint32x8xm1** (unsigned int *\*address*, long *stride*, uint32x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8hv\_uint64x8xm1** (unsigned long *\*address*, long *stride*, uint64x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8hv\_uint8x8xm1** (unsigned char *\*address*, long *stride*, uint8x8xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg8hv\_mask\_int16x8xm1** (short *\*address*, long *stride*, int16x8xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8hv\_mask\_int32x8xm1** (int *\*address*, long *stride*, int32x8xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8hv\_mask\_int64x8xm1** (long *\*address*, long *stride*, int64x8xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8hv\_mask\_int8x8xm1** (signed char *\*address*, long *stride*, int8x8xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8hv\_mask\_uint16x8xm1** (unsigned short *\*address*, long *stride*, uint16x8xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8hv\_mask\_uint32x8xm1** (unsigned int *\*address*, long *stride*, uint32x8xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8hv\_mask\_uint64x8xm1** (unsigned long *\*address*, long *stride*, uint64x8xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8hv\_mask\_uint8x8xm1** (unsigned char *\*address*, long *stride*, uint8x8xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

### 2.10.203 Store 8 contiguous 32b fields in memory(strided) from consecutively numbered vector registers

**Instruction:** ['vssseg8w.v']

**Prototypes:**

- void **vssseg8wv\_int16x8xm1** (short *\*address*, long *stride*, int16x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8wv\_int32x8xm1** (int *\*address*, long *stride*, int32x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8wv\_int64x8xm1** (long *\*address*, long *stride*, int64x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8wv\_int8x8xm1** (signed char *\*address*, long *stride*, int8x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8wv\_uint16x8xm1** (unsigned short *\*address*, long *stride*, uint16x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8wv\_uint32x8xm1** (unsigned int *\*address*, long *stride*, uint32x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8wv\_uint64x8xm1** (unsigned long *\*address*, long *stride*, uint64x8xm1\_t *a*, unsigned int *gvl*)
- void **vssseg8wv\_uint8x8xm1** (unsigned char *\*address*, long *stride*, uint8x8xm1\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + stride
```

**Masked prototypes:**

- void **vssseg8wv\_mask\_int16x8xm1** (short *\*address*, long *stride*, int16x8xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8wv\_mask\_int32x8xm1** (int *\*address*, long *stride*, int32x8xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8wv\_mask\_int64x8xm1** (long *\*address*, long *stride*, int64x8xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8wv\_mask\_int8x8xm1** (signed char *\*address*, long *stride*, int8x8xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8wv\_mask\_uint16x8xm1** (unsigned short *\*address*, long *stride*, uint16x8xm1\_t *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8wv\_mask\_uint32x8xm1** (unsigned int *\*address*, long *stride*, uint32x8xm1\_t *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8wv\_mask\_uint64x8xm1** (unsigned long *\*address*, long *stride*, uint64x8xm1\_t *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vssseg8wv\_mask\_uint8x8xm1** (unsigned char *\*address*, long *stride*, uint8x8xm1\_t *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + stride
```

## 2.10.204 Store 2 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg2b.v']

**Prototypes:**

- void **vsxseg2bv\_int16xm1\_int16x2xm1** (short *\*address*, *int16xm1\_t* index, int16x2xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_int16xm2\_int16x2xm2** (short *\*address*, *int16xm2\_t* index, int16x2xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_int16xm4\_int16x2xm4** (short *\*address*, *int16xm4\_t* index, int16x2xm4\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_int32xm1\_int32x2xm1** (int *\*address*, *int32xm1\_t* index, int32x2xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_int32xm2\_int32x2xm2** (int *\*address*, *int32xm2\_t* index, int32x2xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_int32xm4\_int32x2xm4** (int *\*address*, *int32xm4\_t* index, int32x2xm4\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_int64xm1\_int64x2xm1** (long *\*address*, *int64xm1\_t* index, int64x2xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_int64xm2\_int64x2xm2** (long *\*address*, *int64xm2\_t* index, int64x2xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_int64xm4\_int64x2xm4** (long *\*address*, *int64xm4\_t* index, int64x2xm4\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_int8xm1\_int8x2xm1** (signed char *\*address*, *int8xm1\_t* index, int8x2xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_int8xm2\_int8x2xm2** (signed char *\*address*, *int8xm2\_t* index, int8x2xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_int8xm4\_int8x2xm4** (signed char *\*address*, *int8xm4\_t* index, int8x2xm4\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_uint16xm1\_uint16x2xm1** (unsigned short *\*address*, *uint16xm1\_t* index, uint16x2xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_uint16xm2\_uint16x2xm2** (unsigned short *\*address*, *uint16xm2\_t* index, uint16x2xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_uint16xm4\_uint16x2xm4** (unsigned short *\*address*, *uint16xm4\_t* index, uint16x2xm4\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_uint32xm1\_uint32x2xm1** (unsigned int *\*address*, *uint32xm1\_t* index, uint32x2xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_uint32xm2\_uint32x2xm2** (unsigned int *\*address*, *uint32xm2\_t* index, uint32x2xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_uint32xm4\_uint32x2xm4** (unsigned int *\*address*, *uint32xm4\_t* index, uint32x2xm4\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_uint64xm1\_uint64x2xm1** (unsigned long *\*address*, *uint64xm1\_t* index, uint64x2xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_uint64xm2\_uint64x2xm2** (unsigned long *\*address*, *uint64xm2\_t* index, uint64x2xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_uint64xm4\_uint64x2xm4** (unsigned long *\*address*, *uint64xm4\_t* index, uint64x2xm4\_t *a*, unsigned int *gvl*)
- void **vsxseg2bv\_uint8xm1\_uint8x2xm1** (unsigned char *\*address*, *uint8xm1\_t* index, uint8x2xm1\_t *a*, unsigned int *gvl*)

- void **vsxseg2bv\_uint8xm2\_uint8x2xm2** (unsigned char \*address, uint8xm2\_t index, uint8x2xm2\_t a, unsigned int gvl)
- void **vsxseg2bv\_uint8xm4\_uint8x2xm4** (unsigned char \*address, uint8xm4\_t index, uint8x2xm4\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg2bv\_mask\_int16xm1\_int16x2xm1** (short \*address, int16xm1\_t index, int16x2xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_int16xm2\_int16x2xm2** (short \*address, int16xm2\_t index, int16x2xm2\_t a, e16xm2\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_int16xm4\_int16x2xm4** (short \*address, int16xm4\_t index, int16x2xm4\_t a, e16xm4\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_int32xm1\_int32x2xm1** (int \*address, int32xm1\_t index, int32x2xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_int32xm2\_int32x2xm2** (int \*address, int32xm2\_t index, int32x2xm2\_t a, e32xm2\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_int32xm4\_int32x2xm4** (int \*address, int32xm4\_t index, int32x2xm4\_t a, e32xm4\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_int64xm1\_int64x2xm1** (long \*address, int64xm1\_t index, int64x2xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_int64xm2\_int64x2xm2** (long \*address, int64xm2\_t index, int64x2xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_int64xm4\_int64x2xm4** (long \*address, int64xm4\_t index, int64x2xm4\_t a, e64xm4\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_int8xm1\_int8x2xm1** (signed char \*address, int8xm1\_t index, int8x2xm1\_t a, e8xm1\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_int8xm2\_int8x2xm2** (signed char \*address, int8xm2\_t index, int8x2xm2\_t a, e8xm2\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_int8xm4\_int8x2xm4** (signed char \*address, int8xm4\_t index, int8x2xm4\_t a, e8xm4\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_uint16xm1\_uint16x2xm1** (unsigned short \*address, uint16xm1\_t index, uint16x2xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsxseg2bv\_mask\_uint16xm2\_uint16x2xm2** (unsigned short \*address, uint16xm2\_t index, uint16x2xm2\_t a, e16xm2\_t mask, unsigned int gvl)

- void **vsxseg2bv\_mask\_uint16xm4\_uint16x2xm4** (unsigned short *\*address*, *uint16xm4\_t* *index*, *uint16x2xm4\_t* *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2bv\_mask\_uint32xm1\_uint32x2xm1** (unsigned int *\*address*, *uint32xm1\_t* *index*, *uint32x2xm1\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2bv\_mask\_uint32xm2\_uint32x2xm2** (unsigned int *\*address*, *uint32xm2\_t* *index*, *uint32x2xm2\_t* *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2bv\_mask\_uint32xm4\_uint32x2xm4** (unsigned int *\*address*, *uint32xm4\_t* *index*, *uint32x2xm4\_t* *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2bv\_mask\_uint64xm1\_uint64x2xm1** (unsigned long *\*address*, *uint64xm1\_t* *index*, *uint64x2xm1\_t* *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2bv\_mask\_uint64xm2\_uint64x2xm2** (unsigned long *\*address*, *uint64xm2\_t* *index*, *uint64x2xm2\_t* *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2bv\_mask\_uint64xm4\_uint64x2xm4** (unsigned long *\*address*, *uint64xm4\_t* *index*, *uint64x2xm4\_t* *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2bv\_mask\_uint8xm1\_uint8x2xm1** (unsigned char *\*address*, *uint8xm1\_t* *index*, *uint8x2xm1\_t* *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2bv\_mask\_uint8xm2\_uint8x2xm2** (unsigned char *\*address*, *uint8xm2\_t* *index*, *uint8x2xm2\_t* *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2bv\_mask\_uint8xm4\_uint8x2xm4** (unsigned char *\*address*, *uint8xm4\_t* *index*, *uint8x2xm4\_t* *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)

Masked operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.205 Store 2 contiguous element fields in memory(indexed) from consecutively numbered vector registers

Instruction: ['vsxseg2e.v']

Prototypes:

- void **vsxseg2ev\_float16xm1\_float16x2xm1** (*float16\_t* *\*address*, *float16xm1\_t* *index*, *float16x2xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg2ev\_float16xm2\_float16x2xm2** (*float16\_t* *\*address*, *float16xm2\_t* *index*, *float16x2xm2\_t* *a*, unsigned int *gvl*)
- void **vsxseg2ev\_float16xm4\_float16x2xm4** (*float16\_t* *\*address*, *float16xm4\_t* *index*, *float16x2xm4\_t* *a*, unsigned int *gvl*)

- void **vsxseg2ev\_float32xm1\_float32x2xm1** (float \*address, *float32xm1\_t* index, float32x2xm1\_t a, unsigned int gvl)
- void **vsxseg2ev\_float32xm2\_float32x2xm2** (float \*address, *float32xm2\_t* index, float32x2xm2\_t a, unsigned int gvl)
- void **vsxseg2ev\_float32xm4\_float32x2xm4** (float \*address, *float32xm4\_t* index, float32x2xm4\_t a, unsigned int gvl)
- void **vsxseg2ev\_float64xm1\_float64x2xm1** (double \*address, *float64xm1\_t* index, float64x2xm1\_t a, unsigned int gvl)
- void **vsxseg2ev\_float64xm2\_float64x2xm2** (double \*address, *float64xm2\_t* index, float64x2xm2\_t a, unsigned int gvl)
- void **vsxseg2ev\_float64xm4\_float64x2xm4** (double \*address, *float64xm4\_t* index, float64x2xm4\_t a, unsigned int gvl)
- void **vsxseg2ev\_int16xm1\_int16x2xm1** (short \*address, *int16xm1\_t* index, int16x2xm1\_t a, unsigned int gvl)
- void **vsxseg2ev\_int16xm2\_int16x2xm2** (short \*address, *int16xm2\_t* index, int16x2xm2\_t a, unsigned int gvl)
- void **vsxseg2ev\_int16xm4\_int16x2xm4** (short \*address, *int16xm4\_t* index, int16x2xm4\_t a, unsigned int gvl)
- void **vsxseg2ev\_int32xm1\_int32x2xm1** (int \*address, *int32xm1\_t* index, int32x2xm1\_t a, unsigned int gvl)
- void **vsxseg2ev\_int32xm2\_int32x2xm2** (int \*address, *int32xm2\_t* index, int32x2xm2\_t a, unsigned int gvl)
- void **vsxseg2ev\_int32xm4\_int32x2xm4** (int \*address, *int32xm4\_t* index, int32x2xm4\_t a, unsigned int gvl)
- void **vsxseg2ev\_int64xm1\_int64x2xm1** (long \*address, *int64xm1\_t* index, int64x2xm1\_t a, unsigned int gvl)
- void **vsxseg2ev\_int64xm2\_int64x2xm2** (long \*address, *int64xm2\_t* index, int64x2xm2\_t a, unsigned int gvl)
- void **vsxseg2ev\_int64xm4\_int64x2xm4** (long \*address, *int64xm4\_t* index, int64x2xm4\_t a, unsigned int gvl)
- void **vsxseg2ev\_int8xm1\_int8x2xm1** (signed char \*address, *int8xm1\_t* index, int8x2xm1\_t a, unsigned int gvl)
- void **vsxseg2ev\_int8xm2\_int8x2xm2** (signed char \*address, *int8xm2\_t* index, int8x2xm2\_t a, unsigned int gvl)
- void **vsxseg2ev\_int8xm4\_int8x2xm4** (signed char \*address, *int8xm4\_t* index, int8x2xm4\_t a, unsigned int gvl)
- void **vsxseg2ev\_uint16xm1\_uint16x2xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x2xm1\_t a, unsigned int gvl)
- void **vsxseg2ev\_uint16xm2\_uint16x2xm2** (unsigned short \*address, *uint16xm2\_t* index, uint16x2xm2\_t a, unsigned int gvl)
- void **vsxseg2ev\_uint16xm4\_uint16x2xm4** (unsigned short \*address, *uint16xm4\_t* index, uint16x2xm4\_t a, unsigned int gvl)
- void **vsxseg2ev\_uint32xm1\_uint32x2xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x2xm1\_t a, unsigned int gvl)
- void **vsxseg2ev\_uint32xm2\_uint32x2xm2** (unsigned int \*address, *uint32xm2\_t* index, uint32x2xm2\_t a, unsigned int gvl)



- void **vsxseg2ev\_uint32xm4\_uint32x2xm4** (unsigned int *\*address*, *uint32xm4\_t* *index*, *uint32x2xm4\_t* *a*, unsigned int *gvl*)
- void **vsxseg2ev\_uint64xm1\_uint64x2xm1** (unsigned long *\*address*, *uint64xm1\_t* *index*, *uint64x2xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg2ev\_uint64xm2\_uint64x2xm2** (unsigned long *\*address*, *uint64xm2\_t* *index*, *uint64x2xm2\_t* *a*, unsigned int *gvl*)
- void **vsxseg2ev\_uint64xm4\_uint64x2xm4** (unsigned long *\*address*, *uint64xm4\_t* *index*, *uint64x2xm4\_t* *a*, unsigned int *gvl*)
- void **vsxseg2ev\_uint8xm1\_uint8x2xm1** (unsigned char *\*address*, *uint8xm1\_t* *index*, *uint8x2xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg2ev\_uint8xm2\_uint8x2xm2** (unsigned char *\*address*, *uint8xm2\_t* *index*, *uint8x2xm2\_t* *a*, unsigned int *gvl*)
- void **vsxseg2ev\_uint8xm4\_uint8x2xm4** (unsigned char *\*address*, *uint8xm4\_t* *index*, *uint8x2xm4\_t* *a*, unsigned int *gvl*)

#### Operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

#### Masked prototypes:

- void **vsxseg2ev\_mask\_float16xm1\_float16x2xm1** (float16\_t *\*address*, *float16xm1\_t* *index*, *float16x2xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2ev\_mask\_float16xm2\_float16x2xm2** (float16\_t *\*address*, *float16xm2\_t* *index*, *float16x2xm2\_t* *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2ev\_mask\_float16xm4\_float16x2xm4** (float16\_t *\*address*, *float16xm4\_t* *index*, *float16x2xm4\_t* *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2ev\_mask\_float32xm1\_float32x2xm1** (float *\*address*, *float32xm1\_t* *index*, *float32x2xm1\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2ev\_mask\_float32xm2\_float32x2xm2** (float *\*address*, *float32xm2\_t* *index*, *float32x2xm2\_t* *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2ev\_mask\_float32xm4\_float32x2xm4** (float *\*address*, *float32xm4\_t* *index*, *float32x2xm4\_t* *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2ev\_mask\_float64xm1\_float64x2xm1** (double *\*address*, *float64xm1\_t* *index*, *float64x2xm1\_t* *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2ev\_mask\_float64xm2\_float64x2xm2** (double *\*address*, *float64xm2\_t* *index*, *float64x2xm2\_t* *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2ev\_mask\_float64xm4\_float64x2xm4** (double *\*address*, *float64xm4\_t* *index*, *float64x2xm4\_t* *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)

- void **vsxseg2ev\_mask\_int16xm1\_int16x2xm1** (short \*address, *int16xm1\_t* index, int16x2xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_int16xm2\_int16x2xm2** (short \*address, *int16xm2\_t* index, int16x2xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_int16xm4\_int16x2xm4** (short \*address, *int16xm4\_t* index, int16x2xm4\_t a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_int32xm1\_int32x2xm1** (int \*address, *int32xm1\_t* index, int32x2xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_int32xm2\_int32x2xm2** (int \*address, *int32xm2\_t* index, int32x2xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_int32xm4\_int32x2xm4** (int \*address, *int32xm4\_t* index, int32x2xm4\_t a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_int64xm1\_int64x2xm1** (long \*address, *int64xm1\_t* index, int64x2xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_int64xm2\_int64x2xm2** (long \*address, *int64xm2\_t* index, int64x2xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_int64xm4\_int64x2xm4** (long \*address, *int64xm4\_t* index, int64x2xm4\_t a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_int8xm1\_int8x2xm1** (signed char \*address, *int8xm1\_t* index, int8x2xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_int8xm2\_int8x2xm2** (signed char \*address, *int8xm2\_t* index, int8x2xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_int8xm4\_int8x2xm4** (signed char \*address, *int8xm4\_t* index, int8x2xm4\_t a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_uint16xm1\_uint16x2xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x2xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_uint16xm2\_uint16x2xm2** (unsigned short \*address, *uint16xm2\_t* index, uint16x2xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_uint16xm4\_uint16x2xm4** (unsigned short \*address, *uint16xm4\_t* index, uint16x2xm4\_t a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_uint32xm1\_uint32x2xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x2xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_uint32xm2\_uint32x2xm2** (unsigned int \*address, *uint32xm2\_t* index, uint32x2xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_uint32xm4\_uint32x2xm4** (unsigned int \*address, *uint32xm4\_t* index, uint32x2xm4\_t a, *e32xm4\_t* mask, unsigned int gvl)

- void **vsxseg2ev\_mask\_uint64xm1\_uint64x2xm1** (unsigned long \*address, [uint64xm1\\_t](#) index, uint64x2xm1\_t a, [e64xm1\\_t](#) mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_uint64xm2\_uint64x2xm2** (unsigned long \*address, [uint64xm2\\_t](#) index, uint64x2xm2\_t a, [e64xm2\\_t](#) mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_uint64xm4\_uint64x2xm4** (unsigned long \*address, [uint64xm4\\_t](#) index, uint64x2xm4\_t a, [e64xm4\\_t](#) mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_uint8xm1\_uint8x2xm1** (unsigned char \*address, [uint8xm1\\_t](#) index, uint8x2xm1\_t a, [e8xm1\\_t](#) mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_uint8xm2\_uint8x2xm2** (unsigned char \*address, [uint8xm2\\_t](#) index, uint8x2xm2\_t a, [e8xm2\\_t](#) mask, unsigned int gvl)
- void **vsxseg2ev\_mask\_uint8xm4\_uint8x2xm4** (unsigned char \*address, [uint8xm4\\_t](#) index, uint8x2xm4\_t a, [e8xm4\\_t](#) mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.206 Store 2 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** [`'vsxseg2h.v'`]

**Prototypes:**

- void **vsxseg2hv\_int16xm1\_int16x2xm1** (short \*address, [int16xm1\\_t](#) index, int16x2xm1\_t a, unsigned int gvl)
- void **vsxseg2hv\_int16xm2\_int16x2xm2** (short \*address, [int16xm2\\_t](#) index, int16x2xm2\_t a, unsigned int gvl)
- void **vsxseg2hv\_int16xm4\_int16x2xm4** (short \*address, [int16xm4\\_t](#) index, int16x2xm4\_t a, unsigned int gvl)
- void **vsxseg2hv\_int32xm1\_int32x2xm1** (int \*address, [int32xm1\\_t](#) index, int32x2xm1\_t a, unsigned int gvl)
- void **vsxseg2hv\_int32xm2\_int32x2xm2** (int \*address, [int32xm2\\_t](#) index, int32x2xm2\_t a, unsigned int gvl)
- void **vsxseg2hv\_int32xm4\_int32x2xm4** (int \*address, [int32xm4\\_t](#) index, int32x2xm4\_t a, unsigned int gvl)
- void **vsxseg2hv\_int64xm1\_int64x2xm1** (long \*address, [int64xm1\\_t](#) index, int64x2xm1\_t a, unsigned int gvl)
- void **vsxseg2hv\_int64xm2\_int64x2xm2** (long \*address, [int64xm2\\_t](#) index, int64x2xm2\_t a, unsigned int gvl)
- void **vsxseg2hv\_int64xm4\_int64x2xm4** (long \*address, [int64xm4\\_t](#) index, int64x2xm4\_t a, unsigned int gvl)

- void **vsxseg2hv\_int8xm1\_int8x2xm1** (signed char *\*address*, *int8xm1\_t* *index*, *int8x2xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_int8xm2\_int8x2xm2** (signed char *\*address*, *int8xm2\_t* *index*, *int8x2xm2\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_int8xm4\_int8x2xm4** (signed char *\*address*, *int8xm4\_t* *index*, *int8x2xm4\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_uint16xm1\_uint16x2xm1** (unsigned short *\*address*, *uint16xm1\_t* *index*, *uint16x2xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_uint16xm2\_uint16x2xm2** (unsigned short *\*address*, *uint16xm2\_t* *index*, *uint16x2xm2\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_uint16xm4\_uint16x2xm4** (unsigned short *\*address*, *uint16xm4\_t* *index*, *uint16x2xm4\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_uint32xm1\_uint32x2xm1** (unsigned int *\*address*, *uint32xm1\_t* *index*, *uint32x2xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_uint32xm2\_uint32x2xm2** (unsigned int *\*address*, *uint32xm2\_t* *index*, *uint32x2xm2\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_uint32xm4\_uint32x2xm4** (unsigned int *\*address*, *uint32xm4\_t* *index*, *uint32x2xm4\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_uint64xm1\_uint64x2xm1** (unsigned long *\*address*, *uint64xm1\_t* *index*, *uint64x2xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_uint64xm2\_uint64x2xm2** (unsigned long *\*address*, *uint64xm2\_t* *index*, *uint64x2xm2\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_uint64xm4\_uint64x2xm4** (unsigned long *\*address*, *uint64xm4\_t* *index*, *uint64x2xm4\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_uint8xm1\_uint8x2xm1** (unsigned char *\*address*, *uint8xm1\_t* *index*, *uint8x2xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_uint8xm2\_uint8x2xm2** (unsigned char *\*address*, *uint8xm2\_t* *index*, *uint8x2xm2\_t* *a*, unsigned int *gvl*)
- void **vsxseg2hv\_uint8xm4\_uint8x2xm4** (unsigned char *\*address*, *uint8xm4\_t* *index*, *uint8x2xm4\_t* *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg2hv\_mask\_int16xm1\_int16x2xm1** (short *\*address*, *int16xm1\_t* *index*, *int16x2xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2hv\_mask\_int16xm2\_int16x2xm2** (short *\*address*, *int16xm2\_t* *index*, *int16x2xm2\_t* *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2hv\_mask\_int16xm4\_int16x2xm4** (short *\*address*, *int16xm4\_t* *index*, *int16x2xm4\_t* *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- void **vsxseg2hv\_mask\_int32xm1\_int32x2xm1** (int *\*address*, *int32xm1\_t* *index*, *int32x2xm1\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)

- void **vsxseg2hv\_mask\_int32xm2\_int32x2xm2** (int \*address, *int32xm2\_t* index, int32x2xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_int32xm4\_int32x2xm4** (int \*address, *int32xm4\_t* index, int32x2xm4\_t a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_int64xm1\_int64x2xm1** (long \*address, *int64xm1\_t* index, int64x2xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_int64xm2\_int64x2xm2** (long \*address, *int64xm2\_t* index, int64x2xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_int64xm4\_int64x2xm4** (long \*address, *int64xm4\_t* index, int64x2xm4\_t a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_int8xm1\_int8x2xm1** (signed char \*address, *int8xm1\_t* index, int8x2xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_int8xm2\_int8x2xm2** (signed char \*address, *int8xm2\_t* index, int8x2xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_int8xm4\_int8x2xm4** (signed char \*address, *int8xm4\_t* index, int8x2xm4\_t a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_uint16xm1\_uint16x2xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x2xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_uint16xm2\_uint16x2xm2** (unsigned short \*address, *uint16xm2\_t* index, uint16x2xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_uint16xm4\_uint16x2xm4** (unsigned short \*address, *uint16xm4\_t* index, uint16x2xm4\_t a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_uint32xm1\_uint32x2xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x2xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_uint32xm2\_uint32x2xm2** (unsigned int \*address, *uint32xm2\_t* index, uint32x2xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_uint32xm4\_uint32x2xm4** (unsigned int \*address, *uint32xm4\_t* index, uint32x2xm4\_t a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_uint64xm1\_uint64x2xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x2xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_uint64xm2\_uint64x2xm2** (unsigned long \*address, *uint64xm2\_t* index, uint64x2xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_uint64xm4\_uint64x2xm4** (unsigned long \*address, *uint64xm4\_t* index, uint64x2xm4\_t a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_uint8xm1\_uint8x2xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x2xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

- void **vsxseg2hv\_mask\_uint8xm2\_uint8x2xm2** (unsigned char \*address, [uint8xm2\\_t](#) index, uint8x2xm2\_t a, [e8xm2\\_t](#) mask, unsigned int gvl)
- void **vsxseg2hv\_mask\_uint8xm4\_uint8x2xm4** (unsigned char \*address, [uint8xm4\\_t](#) index, uint8x2xm4\_t a, [e8xm4\\_t](#) mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.207 Store 2 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg2wv.v']

**Prototypes:**

- void **vsxseg2wv\_int16xm1\_int16x2xm1** (short \*address, [int16xm1\\_t](#) index, int16x2xm1\_t a, unsigned int gvl)
- void **vsxseg2wv\_int16xm2\_int16x2xm2** (short \*address, [int16xm2\\_t](#) index, int16x2xm2\_t a, unsigned int gvl)
- void **vsxseg2wv\_int16xm4\_int16x2xm4** (short \*address, [int16xm4\\_t](#) index, int16x2xm4\_t a, unsigned int gvl)
- void **vsxseg2wv\_int32xm1\_int32x2xm1** (int \*address, [int32xm1\\_t](#) index, int32x2xm1\_t a, unsigned int gvl)
- void **vsxseg2wv\_int32xm2\_int32x2xm2** (int \*address, [int32xm2\\_t](#) index, int32x2xm2\_t a, unsigned int gvl)
- void **vsxseg2wv\_int32xm4\_int32x2xm4** (int \*address, [int32xm4\\_t](#) index, int32x2xm4\_t a, unsigned int gvl)
- void **vsxseg2wv\_int64xm1\_int64x2xm1** (long \*address, [int64xm1\\_t](#) index, int64x2xm1\_t a, unsigned int gvl)
- void **vsxseg2wv\_int64xm2\_int64x2xm2** (long \*address, [int64xm2\\_t](#) index, int64x2xm2\_t a, unsigned int gvl)
- void **vsxseg2wv\_int64xm4\_int64x2xm4** (long \*address, [int64xm4\\_t](#) index, int64x2xm4\_t a, unsigned int gvl)
- void **vsxseg2wv\_int8xm1\_int8x2xm1** (signed char \*address, [int8xm1\\_t](#) index, int8x2xm1\_t a, unsigned int gvl)
- void **vsxseg2wv\_int8xm2\_int8x2xm2** (signed char \*address, [int8xm2\\_t](#) index, int8x2xm2\_t a, unsigned int gvl)
- void **vsxseg2wv\_int8xm4\_int8x2xm4** (signed char \*address, [int8xm4\\_t](#) index, int8x2xm4\_t a, unsigned int gvl)
- void **vsxseg2wv\_uint16xm1\_uint16x2xm1** (unsigned short \*address, [uint16xm1\\_t](#) index, uint16x2xm1\_t a, unsigned int gvl)
- void **vsxseg2wv\_uint16xm2\_uint16x2xm2** (unsigned short \*address, [uint16xm2\\_t](#) index, uint16x2xm2\_t a, unsigned int gvl)



- void **vsxseg2wv\_uint16xm4\_uint16x2xm4** (unsigned short \*address, *uint16xm4\_t* index, uint16x2xm4\_t a, unsigned int gvl)
- void **vsxseg2wv\_uint32xm1\_uint32x2xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x2xm1\_t a, unsigned int gvl)
- void **vsxseg2wv\_uint32xm2\_uint32x2xm2** (unsigned int \*address, *uint32xm2\_t* index, uint32x2xm2\_t a, unsigned int gvl)
- void **vsxseg2wv\_uint32xm4\_uint32x2xm4** (unsigned int \*address, *uint32xm4\_t* index, uint32x2xm4\_t a, unsigned int gvl)
- void **vsxseg2wv\_uint64xm1\_uint64x2xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x2xm1\_t a, unsigned int gvl)
- void **vsxseg2wv\_uint64xm2\_uint64x2xm2** (unsigned long \*address, *uint64xm2\_t* index, uint64x2xm2\_t a, unsigned int gvl)
- void **vsxseg2wv\_uint64xm4\_uint64x2xm4** (unsigned long \*address, *uint64xm4\_t* index, uint64x2xm4\_t a, unsigned int gvl)
- void **vsxseg2wv\_uint8xm1\_uint8x2xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x2xm1\_t a, unsigned int gvl)
- void **vsxseg2wv\_uint8xm2\_uint8x2xm2** (unsigned char \*address, *uint8xm2\_t* index, uint8x2xm2\_t a, unsigned int gvl)
- void **vsxseg2wv\_uint8xm4\_uint8x2xm4** (unsigned char \*address, *uint8xm4\_t* index, uint8x2xm4\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(2 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg2wv\_mask\_int16xm1\_int16x2xm1** (short \*address, *int16xm1\_t* index, int16x2xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_int16xm2\_int16x2xm2** (short \*address, *int16xm2\_t* index, int16x2xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_int16xm4\_int16x2xm4** (short \*address, *int16xm4\_t* index, int16x2xm4\_t a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_int32xm1\_int32x2xm1** (int \*address, *int32xm1\_t* index, int32x2xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_int32xm2\_int32x2xm2** (int \*address, *int32xm2\_t* index, int32x2xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_int32xm4\_int32x2xm4** (int \*address, *int32xm4\_t* index, int32x2xm4\_t a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_int64xm1\_int64x2xm1** (long \*address, *int64xm1\_t* index, int64x2xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_int64xm2\_int64x2xm2** (long \*address, *int64xm2\_t* index, int64x2xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)

- void **vsxseg2wv\_mask\_int64xm4\_int64x2xm4** (long \*address, *int64xm4\_t* index, *int64x2xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_int8xm1\_int8x2xm1** (signed char \*address, *int8xm1\_t* index, *int8x2xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_int8xm2\_int8x2xm2** (signed char \*address, *int8xm2\_t* index, *int8x2xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_int8xm4\_int8x2xm4** (signed char \*address, *int8xm4\_t* index, *int8x2xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_uint16xm1\_uint16x2xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16x2xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_uint16xm2\_uint16x2xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16x2xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_uint16xm4\_uint16x2xm4** (unsigned short \*address, *uint16xm4\_t* index, *uint16x2xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_uint32xm1\_uint32x2xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32x2xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_uint32xm2\_uint32x2xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32x2xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_uint32xm4\_uint32x2xm4** (unsigned int \*address, *uint32xm4\_t* index, *uint32x2xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_uint64xm1\_uint64x2xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64x2xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_uint64xm2\_uint64x2xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64x2xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_uint64xm4\_uint64x2xm4** (unsigned long \*address, *uint64xm4\_t* index, *uint64x2xm4\_t* a, *e64xm4\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_uint8xm1\_uint8x2xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8x2xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_uint8xm2\_uint8x2xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8x2xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxseg2wv\_mask\_uint8xm4\_uint8x2xm4** (unsigned char \*address, *uint8xm4\_t* index, *uint8x2xm4\_t* a, *e8xm4\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for segment(2 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.208 Store 3 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg3b.v']

**Prototypes:**

- void **vsxseg3bv\_int16xm1\_int16x3xm1** (short \*address, *int16xm1\_t* index, int16x3xm1\_t a, unsigned int gvl)
- void **vsxseg3bv\_int16xm2\_int16x3xm2** (short \*address, *int16xm2\_t* index, int16x3xm2\_t a, unsigned int gvl)
- void **vsxseg3bv\_int32xm1\_int32x3xm1** (int \*address, *int32xm1\_t* index, int32x3xm1\_t a, unsigned int gvl)
- void **vsxseg3bv\_int32xm2\_int32x3xm2** (int \*address, *int32xm2\_t* index, int32x3xm2\_t a, unsigned int gvl)
- void **vsxseg3bv\_int64xm1\_int64x3xm1** (long \*address, *int64xm1\_t* index, int64x3xm1\_t a, unsigned int gvl)
- void **vsxseg3bv\_int64xm2\_int64x3xm2** (long \*address, *int64xm2\_t* index, int64x3xm2\_t a, unsigned int gvl)
- void **vsxseg3bv\_int8xm1\_int8x3xm1** (signed char \*address, *int8xm1\_t* index, int8x3xm1\_t a, unsigned int gvl)
- void **vsxseg3bv\_int8xm2\_int8x3xm2** (signed char \*address, *int8xm2\_t* index, int8x3xm2\_t a, unsigned int gvl)
- void **vsxseg3bv\_uint16xm1\_uint16x3xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x3xm1\_t a, unsigned int gvl)
- void **vsxseg3bv\_uint16xm2\_uint16x3xm2** (unsigned short \*address, *uint16xm2\_t* index, uint16x3xm2\_t a, unsigned int gvl)
- void **vsxseg3bv\_uint32xm1\_uint32x3xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x3xm1\_t a, unsigned int gvl)
- void **vsxseg3bv\_uint32xm2\_uint32x3xm2** (unsigned int \*address, *uint32xm2\_t* index, uint32x3xm2\_t a, unsigned int gvl)
- void **vsxseg3bv\_uint64xm1\_uint64x3xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x3xm1\_t a, unsigned int gvl)
- void **vsxseg3bv\_uint64xm2\_uint64x3xm2** (unsigned long \*address, *uint64xm2\_t* index, uint64x3xm2\_t a, unsigned int gvl)
- void **vsxseg3bv\_uint8xm1\_uint8x3xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x3xm1\_t a, unsigned int gvl)
- void **vsxseg3bv\_uint8xm2\_uint8x3xm2** (unsigned char \*address, *uint8xm2\_t* index, uint8x3xm2\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

### Masked prototypes:

- void **vsxseg3bv\_mask\_int16xm1\_int16x3xm1** (short \*address, *int16xm1\_t* index, int16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_int16xm2\_int16x3xm2** (short \*address, *int16xm2\_t* index, int16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_int32xm1\_int32x3xm1** (int \*address, *int32xm1\_t* index, int32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_int32xm2\_int32x3xm2** (int \*address, *int32xm2\_t* index, int32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_int64xm1\_int64x3xm1** (long \*address, *int64xm1\_t* index, int64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_int64xm2\_int64x3xm2** (long \*address, *int64xm2\_t* index, int64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_int8xm1\_int8x3xm1** (signed char \*address, *int8xm1\_t* index, int8x3xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_int8xm2\_int8x3xm2** (signed char \*address, *int8xm2\_t* index, int8x3xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_uint16xm1\_uint16x3xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_uint16xm2\_uint16x3xm2** (unsigned short \*address, *uint16xm2\_t* index, uint16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_uint32xm1\_uint32x3xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_uint32xm2\_uint32x3xm2** (unsigned int \*address, *uint32xm2\_t* index, uint32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_uint64xm1\_uint64x3xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_uint64xm2\_uint64x3xm2** (unsigned long \*address, *uint64xm2\_t* index, uint64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg3bv\_mask\_uint8xm1\_uint8x3xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x3xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

- void **vsxseg3bv\_mask\_uint8xm2\_uint8x3xm2** (unsigned char \*address, *uint8xm2\_t* index, uint8x3xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.209 Store 3 contiguous element fields in memory(indexed) from consecutively numbered vector registers

Instruction: ['vsxseg3e.v']

Prototypes:

- void **vsxseg3ev\_float16xm1\_float16x3xm1** (float16\_t \*address, *float16xm1\_t* index, float16x3xm1\_t a, unsigned int gvl)
- void **vsxseg3ev\_float16xm2\_float16x3xm2** (float16\_t \*address, *float16xm2\_t* index, float16x3xm2\_t a, unsigned int gvl)
- void **vsxseg3ev\_float32xm1\_float32x3xm1** (float \*address, *float32xm1\_t* index, float32x3xm1\_t a, unsigned int gvl)
- void **vsxseg3ev\_float32xm2\_float32x3xm2** (float \*address, *float32xm2\_t* index, float32x3xm2\_t a, unsigned int gvl)
- void **vsxseg3ev\_float64xm1\_float64x3xm1** (double \*address, *float64xm1\_t* index, float64x3xm1\_t a, unsigned int gvl)
- void **vsxseg3ev\_float64xm2\_float64x3xm2** (double \*address, *float64xm2\_t* index, float64x3xm2\_t a, unsigned int gvl)
- void **vsxseg3ev\_int16xm1\_int16x3xm1** (short \*address, *int16xm1\_t* index, int16x3xm1\_t a, unsigned int gvl)
- void **vsxseg3ev\_int16xm2\_int16x3xm2** (short \*address, *int16xm2\_t* index, int16x3xm2\_t a, unsigned int gvl)
- void **vsxseg3ev\_int32xm1\_int32x3xm1** (int \*address, *int32xm1\_t* index, int32x3xm1\_t a, unsigned int gvl)
- void **vsxseg3ev\_int32xm2\_int32x3xm2** (int \*address, *int32xm2\_t* index, int32x3xm2\_t a, unsigned int gvl)
- void **vsxseg3ev\_int64xm1\_int64x3xm1** (long \*address, *int64xm1\_t* index, int64x3xm1\_t a, unsigned int gvl)
- void **vsxseg3ev\_int64xm2\_int64x3xm2** (long \*address, *int64xm2\_t* index, int64x3xm2\_t a, unsigned int gvl)
- void **vsxseg3ev\_int8xm1\_int8x3xm1** (signed char \*address, *int8xm1\_t* index, int8x3xm1\_t a, unsigned int gvl)
- void **vsxseg3ev\_int8xm2\_int8x3xm2** (signed char \*address, *int8xm2\_t* index, int8x3xm2\_t a, unsigned int gvl)
- void **vsxseg3ev\_uint16xm1\_uint16x3xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x3xm1\_t a, unsigned int gvl)
- void **vsxseg3ev\_uint16xm2\_uint16x3xm2** (unsigned short \*address, *uint16xm2\_t* index, uint16x3xm2\_t a, unsigned int gvl)

- void **vsxseg3ev\_uint32xm1\_uint32x3xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x3xm1\_t a, unsigned int gvl)
- void **vsxseg3ev\_uint32xm2\_uint32x3xm2** (unsigned int \*address, *uint32xm2\_t* index, uint32x3xm2\_t a, unsigned int gvl)
- void **vsxseg3ev\_uint64xm1\_uint64x3xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x3xm1\_t a, unsigned int gvl)
- void **vsxseg3ev\_uint64xm2\_uint64x3xm2** (unsigned long \*address, *uint64xm2\_t* index, uint64x3xm2\_t a, unsigned int gvl)
- void **vsxseg3ev\_uint8xm1\_uint8x3xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x3xm1\_t a, unsigned int gvl)
- void **vsxseg3ev\_uint8xm2\_uint8x3xm2** (unsigned char \*address, *uint8xm2\_t* index, uint8x3xm2\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg3ev\_mask\_float16xm1\_float16x3xm1** (float16\_t \*address, *float16xm1\_t* index, float16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_float16xm2\_float16x3xm2** (float16\_t \*address, *float16xm2\_t* index, float16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_float32xm1\_float32x3xm1** (float \*address, *float32xm1\_t* index, float32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_float32xm2\_float32x3xm2** (float \*address, *float32xm2\_t* index, float32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_float64xm1\_float64x3xm1** (double \*address, *float64xm1\_t* index, float64x3xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_float64xm2\_float64x3xm2** (double \*address, *float64xm2\_t* index, float64x3xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_int16xm1\_int16x3xm1** (short \*address, *int16xm1\_t* index, int16x3xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_int16xm2\_int16x3xm2** (short \*address, *int16xm2\_t* index, int16x3xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_int32xm1\_int32x3xm1** (int \*address, *int32xm1\_t* index, int32x3xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_int32xm2\_int32x3xm2** (int \*address, *int32xm2\_t* index, int32x3xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)



- void **vsxseg3ev\_mask\_int64xm1\_int64x3xm1** (long \*address, *int64xm1\_t* index, *int64x3xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_int64xm2\_int64x3xm2** (long \*address, *int64xm2\_t* index, *int64x3xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_int8xm1\_int8x3xm1** (signed char \*address, *int8xm1\_t* index, *int8x3xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_int8xm2\_int8x3xm2** (signed char \*address, *int8xm2\_t* index, *int8x3xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_uint16xm1\_uint16x3xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16x3xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_uint16xm2\_uint16x3xm2** (unsigned short \*address, *uint16xm2\_t* index, *uint16x3xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_uint32xm1\_uint32x3xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32x3xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_uint32xm2\_uint32x3xm2** (unsigned int \*address, *uint32xm2\_t* index, *uint32x3xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_uint64xm1\_uint64x3xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64x3xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_uint64xm2\_uint64x3xm2** (unsigned long \*address, *uint64xm2\_t* index, *uint64x3xm2\_t* a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_uint8xm1\_uint8x3xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8x3xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg3ev\_mask\_uint8xm2\_uint8x3xm2** (unsigned char \*address, *uint8xm2\_t* index, *uint8x3xm2\_t* a, *e8xm2\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

### 2.10.210 Store 3 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** [*'vsxseg3h.v'*]

**Prototypes:**

- void **vsxseg3hv\_int16xm1\_int16x3xm1** (short \*address, *int16xm1\_t* index, *int16x3xm1\_t* a, unsigned int gvl)

- void **vsxseg3hv\_int16xm2\_int16x3xm2** (short *\*address*, *int16xm2\_t* index, int16x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_int32xm1\_int32x3xm1** (int *\*address*, *int32xm1\_t* index, int32x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_int32xm2\_int32x3xm2** (int *\*address*, *int32xm2\_t* index, int32x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_int64xm1\_int64x3xm1** (long *\*address*, *int64xm1\_t* index, int64x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_int64xm2\_int64x3xm2** (long *\*address*, *int64xm2\_t* index, int64x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_int8xm1\_int8x3xm1** (signed char *\*address*, *int8xm1\_t* index, int8x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_int8xm2\_int8x3xm2** (signed char *\*address*, *int8xm2\_t* index, int8x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_uint16xm1\_uint16x3xm1** (unsigned short *\*address*, *uint16xm1\_t* index, uint16x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_uint16xm2\_uint16x3xm2** (unsigned short *\*address*, *uint16xm2\_t* index, uint16x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_uint32xm1\_uint32x3xm1** (unsigned int *\*address*, *uint32xm1\_t* index, uint32x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_uint32xm2\_uint32x3xm2** (unsigned int *\*address*, *uint32xm2\_t* index, uint32x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_uint64xm1\_uint64x3xm1** (unsigned long *\*address*, *uint64xm1\_t* index, uint64x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_uint64xm2\_uint64x3xm2** (unsigned long *\*address*, *uint64xm2\_t* index, uint64x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_uint8xm1\_uint8x3xm1** (unsigned char *\*address*, *uint8xm1\_t* index, uint8x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3hv\_uint8xm2\_uint8x3xm2** (unsigned char *\*address*, *uint8xm2\_t* index, uint8x3xm2\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg3hv\_mask\_int16xm1\_int16x3xm1** (short *\*address*, *int16xm1\_t* index, int16x3xm1\_t *a*, *e16xm1\_t* mask, unsigned int *gvl*)
- void **vsxseg3hv\_mask\_int16xm2\_int16x3xm2** (short *\*address*, *int16xm2\_t* index, int16x3xm2\_t *a*, *e16xm2\_t* mask, unsigned int *gvl*)
- void **vsxseg3hv\_mask\_int32xm1\_int32x3xm1** (int *\*address*, *int32xm1\_t* index, int32x3xm1\_t *a*, *e32xm1\_t* mask, unsigned int *gvl*)
- void **vsxseg3hv\_mask\_int32xm2\_int32x3xm2** (int *\*address*, *int32xm2\_t* index, int32x3xm2\_t *a*, *e32xm2\_t* mask, unsigned int *gvl*)

- void **vsxseg3hv\_mask\_int64xm1\_int64x3xm1** (long \*address, int64xm1\_t index, int64x3xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg3hv\_mask\_int64xm2\_int64x3xm2** (long \*address, int64xm2\_t index, int64x3xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsxseg3hv\_mask\_int8xm1\_int8x3xm1** (signed char \*address, int8xm1\_t index, int8x3xm1\_t a, e8xm1\_t mask, unsigned int gvl)
- void **vsxseg3hv\_mask\_int8xm2\_int8x3xm2** (signed char \*address, int8xm2\_t index, int8x3xm2\_t a, e8xm2\_t mask, unsigned int gvl)
- void **vsxseg3hv\_mask\_uint16xm1\_uint16x3xm1** (unsigned short \*address, uint16xm1\_t index, uint16x3xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsxseg3hv\_mask\_uint16xm2\_uint16x3xm2** (unsigned short \*address, uint16xm2\_t index, uint16x3xm2\_t a, e16xm2\_t mask, unsigned int gvl)
- void **vsxseg3hv\_mask\_uint32xm1\_uint32x3xm1** (unsigned int \*address, uint32xm1\_t index, uint32x3xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsxseg3hv\_mask\_uint32xm2\_uint32x3xm2** (unsigned int \*address, uint32xm2\_t index, uint32x3xm2\_t a, e32xm2\_t mask, unsigned int gvl)
- void **vsxseg3hv\_mask\_uint64xm1\_uint64x3xm1** (unsigned long \*address, uint64xm1\_t index, uint64x3xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg3hv\_mask\_uint64xm2\_uint64x3xm2** (unsigned long \*address, uint64xm2\_t index, uint64x3xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsxseg3hv\_mask\_uint8xm1\_uint8x3xm1** (unsigned char \*address, uint8xm1\_t index, uint8x3xm1\_t a, e8xm1\_t mask, unsigned int gvl)
- void **vsxseg3hv\_mask\_uint8xm2\_uint8x3xm2** (unsigned char \*address, uint8xm2\_t index, uint8x3xm2\_t a, e8xm2\_t mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

**2.10.211 Store 3 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers****Instruction:** [**vsxseg3w.v**]**Prototypes:**

- void **vsxseg3wv\_int16xm1\_int16x3xm1** (short \*address, int16xm1\_t index, int16x3xm1\_t a, unsigned int gvl)

- void **vsxseg3wv\_int16xm2\_int16x3xm2** (short *\*address*, *int16xm2\_t* index, int16x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_int32xm1\_int32x3xm1** (int *\*address*, *int32xm1\_t* index, int32x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_int32xm2\_int32x3xm2** (int *\*address*, *int32xm2\_t* index, int32x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_int64xm1\_int64x3xm1** (long *\*address*, *int64xm1\_t* index, int64x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_int64xm2\_int64x3xm2** (long *\*address*, *int64xm2\_t* index, int64x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_int8xm1\_int8x3xm1** (signed char *\*address*, *int8xm1\_t* index, int8x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_int8xm2\_int8x3xm2** (signed char *\*address*, *int8xm2\_t* index, int8x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_uint16xm1\_uint16x3xm1** (unsigned short *\*address*, *uint16xm1\_t* index, uint16x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_uint16xm2\_uint16x3xm2** (unsigned short *\*address*, *uint16xm2\_t* index, uint16x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_uint32xm1\_uint32x3xm1** (unsigned int *\*address*, *uint32xm1\_t* index, uint32x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_uint32xm2\_uint32x3xm2** (unsigned int *\*address*, *uint32xm2\_t* index, uint32x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_uint64xm1\_uint64x3xm1** (unsigned long *\*address*, *uint64xm1\_t* index, uint64x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_uint64xm2\_uint64x3xm2** (unsigned long *\*address*, *uint64xm2\_t* index, uint64x3xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_uint8xm1\_uint8x3xm1** (unsigned char *\*address*, *uint8xm1\_t* index, uint8x3xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg3wv\_uint8xm2\_uint8x3xm2** (unsigned char *\*address*, *uint8xm2\_t* index, uint8x3xm2\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg3wv\_mask\_int16xm1\_int16x3xm1** (short *\*address*, *int16xm1\_t* index, int16x3xm1\_t *a*, *e16xm1\_t* mask, unsigned int *gvl*)
- void **vsxseg3wv\_mask\_int16xm2\_int16x3xm2** (short *\*address*, *int16xm2\_t* index, int16x3xm2\_t *a*, *e16xm2\_t* mask, unsigned int *gvl*)
- void **vsxseg3wv\_mask\_int32xm1\_int32x3xm1** (int *\*address*, *int32xm1\_t* index, int32x3xm1\_t *a*, *e32xm1\_t* mask, unsigned int *gvl*)
- void **vsxseg3wv\_mask\_int32xm2\_int32x3xm2** (int *\*address*, *int32xm2\_t* index, int32x3xm2\_t *a*, *e32xm2\_t* mask, unsigned int *gvl*)

- void **vsxseg3wv\_mask\_int64xm1\_int64x3xm1** (long \*address, int64xm1\_t index, int64x3xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg3wv\_mask\_int64xm2\_int64x3xm2** (long \*address, int64xm2\_t index, int64x3xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsxseg3wv\_mask\_int8xm1\_int8x3xm1** (signed char \*address, int8xm1\_t index, int8x3xm1\_t a, e8xm1\_t mask, unsigned int gvl)
- void **vsxseg3wv\_mask\_int8xm2\_int8x3xm2** (signed char \*address, int8xm2\_t index, int8x3xm2\_t a, e8xm2\_t mask, unsigned int gvl)
- void **vsxseg3wv\_mask\_uint16xm1\_uint16x3xm1** (unsigned short \*address, uint16xm1\_t index, uint16x3xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsxseg3wv\_mask\_uint16xm2\_uint16x3xm2** (unsigned short \*address, uint16xm2\_t index, uint16x3xm2\_t a, e16xm2\_t mask, unsigned int gvl)
- void **vsxseg3wv\_mask\_uint32xm1\_uint32x3xm1** (unsigned int \*address, uint32xm1\_t index, uint32x3xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsxseg3wv\_mask\_uint32xm2\_uint32x3xm2** (unsigned int \*address, uint32xm2\_t index, uint32x3xm2\_t a, e32xm2\_t mask, unsigned int gvl)
- void **vsxseg3wv\_mask\_uint64xm1\_uint64x3xm1** (unsigned long \*address, uint64xm1\_t index, uint64x3xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg3wv\_mask\_uint64xm2\_uint64x3xm2** (unsigned long \*address, uint64xm2\_t index, uint64x3xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsxseg3wv\_mask\_uint8xm1\_uint8x3xm1** (unsigned char \*address, uint8xm1\_t index, uint8x3xm1\_t a, e8xm1\_t mask, unsigned int gvl)
- void **vsxseg3wv\_mask\_uint8xm2\_uint8x3xm2** (unsigned char \*address, uint8xm2\_t index, uint8x3xm2\_t a, e8xm2\_t mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(3 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

**2.10.212 Store 4 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers****Instruction:** ['vsxseg4b.v']**Prototypes:**

- void **vsxseg4bv\_int16xm1\_int16x4xm1** (short \*address, int16xm1\_t index, int16x4xm1\_t a, unsigned int gvl)

- void **vsxseg4bv\_int16xm2\_int16x4xm2** (short *\*address*, *int16xm2\_t* index, int16x4xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_int32xm1\_int32x4xm1** (int *\*address*, *int32xm1\_t* index, int32x4xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_int32xm2\_int32x4xm2** (int *\*address*, *int32xm2\_t* index, int32x4xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_int64xm1\_int64x4xm1** (long *\*address*, *int64xm1\_t* index, int64x4xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_int64xm2\_int64x4xm2** (long *\*address*, *int64xm2\_t* index, int64x4xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_int8xm1\_int8x4xm1** (signed char *\*address*, *int8xm1\_t* index, int8x4xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_int8xm2\_int8x4xm2** (signed char *\*address*, *int8xm2\_t* index, int8x4xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_uint16xm1\_uint16x4xm1** (unsigned short *\*address*, *uint16xm1\_t* index, uint16x4xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_uint16xm2\_uint16x4xm2** (unsigned short *\*address*, *uint16xm2\_t* index, uint16x4xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_uint32xm1\_uint32x4xm1** (unsigned int *\*address*, *uint32xm1\_t* index, uint32x4xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_uint32xm2\_uint32x4xm2** (unsigned int *\*address*, *uint32xm2\_t* index, uint32x4xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_uint64xm1\_uint64x4xm1** (unsigned long *\*address*, *uint64xm1\_t* index, uint64x4xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_uint64xm2\_uint64x4xm2** (unsigned long *\*address*, *uint64xm2\_t* index, uint64x4xm2\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_uint8xm1\_uint8x4xm1** (unsigned char *\*address*, *uint8xm1\_t* index, uint8x4xm1\_t *a*, unsigned int *gvl*)
- void **vsxseg4bv\_uint8xm2\_uint8x4xm2** (unsigned char *\*address*, *uint8xm2\_t* index, uint8x4xm2\_t *a*, unsigned int *gvl*)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg4bv\_mask\_int16xm1\_int16x4xm1** (short *\*address*, *int16xm1\_t* index, int16x4xm1\_t *a*, *e16xm1\_t* mask, unsigned int *gvl*)
- void **vsxseg4bv\_mask\_int16xm2\_int16x4xm2** (short *\*address*, *int16xm2\_t* index, int16x4xm2\_t *a*, *e16xm2\_t* mask, unsigned int *gvl*)
- void **vsxseg4bv\_mask\_int32xm1\_int32x4xm1** (int *\*address*, *int32xm1\_t* index, int32x4xm1\_t *a*, *e32xm1\_t* mask, unsigned int *gvl*)
- void **vsxseg4bv\_mask\_int32xm2\_int32x4xm2** (int *\*address*, *int32xm2\_t* index, int32x4xm2\_t *a*, *e32xm2\_t* mask, unsigned int *gvl*)



- void **vsxseg4bv\_mask\_int64xm1\_int64x4xm1** (long \*address, int64xm1\_t index, int64x4xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg4bv\_mask\_int64xm2\_int64x4xm2** (long \*address, int64xm2\_t index, int64x4xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsxseg4bv\_mask\_int8xm1\_int8x4xm1** (signed char \*address, int8xm1\_t index, int8x4xm1\_t a, e8xm1\_t mask, unsigned int gvl)
- void **vsxseg4bv\_mask\_int8xm2\_int8x4xm2** (signed char \*address, int8xm2\_t index, int8x4xm2\_t a, e8xm2\_t mask, unsigned int gvl)
- void **vsxseg4bv\_mask\_uint16xm1\_uint16x4xm1** (unsigned short \*address, uint16xm1\_t index, uint16x4xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsxseg4bv\_mask\_uint16xm2\_uint16x4xm2** (unsigned short \*address, uint16xm2\_t index, uint16x4xm2\_t a, e16xm2\_t mask, unsigned int gvl)
- void **vsxseg4bv\_mask\_uint32xm1\_uint32x4xm1** (unsigned int \*address, uint32xm1\_t index, uint32x4xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsxseg4bv\_mask\_uint32xm2\_uint32x4xm2** (unsigned int \*address, uint32xm2\_t index, uint32x4xm2\_t a, e32xm2\_t mask, unsigned int gvl)
- void **vsxseg4bv\_mask\_uint64xm1\_uint64x4xm1** (unsigned long \*address, uint64xm1\_t index, uint64x4xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg4bv\_mask\_uint64xm2\_uint64x4xm2** (unsigned long \*address, uint64xm2\_t index, uint64x4xm2\_t a, e64xm2\_t mask, unsigned int gvl)
- void **vsxseg4bv\_mask\_uint8xm1\_uint8x4xm1** (unsigned char \*address, uint8xm1\_t index, uint8x4xm1\_t a, e8xm1\_t mask, unsigned int gvl)
- void **vsxseg4bv\_mask\_uint8xm2\_uint8x4xm2** (unsigned char \*address, uint8xm2\_t index, uint8x4xm2\_t a, e8xm2\_t mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

**2.10.213 Store 4 contiguous element fields in memory(indexed) from consecutively numbered vector registers****Instruction:** [**vsxseg4e.v**]**Prototypes:**

- void **vsxseg4ev\_float16xm1\_float16x4xm1** (float16\_t \*address, float16xm1\_t index, float16x4xm1\_t a, unsigned int gvl)

- void **vsxseg4ev\_float16xm2\_float16x4xm2** (float16\_t \*address, float16xm2\_t index, float16x4xm2\_t a, unsigned int gvl)
- void **vsxseg4ev\_float32xm1\_float32x4xm1** (float \*address, float32xm1\_t index, float32x4xm1\_t a, unsigned int gvl)
- void **vsxseg4ev\_float32xm2\_float32x4xm2** (float \*address, float32xm2\_t index, float32x4xm2\_t a, unsigned int gvl)
- void **vsxseg4ev\_float64xm1\_float64x4xm1** (double \*address, float64xm1\_t index, float64x4xm1\_t a, unsigned int gvl)
- void **vsxseg4ev\_float64xm2\_float64x4xm2** (double \*address, float64xm2\_t index, float64x4xm2\_t a, unsigned int gvl)
- void **vsxseg4ev\_int16xm1\_int16x4xm1** (short \*address, int16xm1\_t index, int16x4xm1\_t a, unsigned int gvl)
- void **vsxseg4ev\_int16xm2\_int16x4xm2** (short \*address, int16xm2\_t index, int16x4xm2\_t a, unsigned int gvl)
- void **vsxseg4ev\_int32xm1\_int32x4xm1** (int \*address, int32xm1\_t index, int32x4xm1\_t a, unsigned int gvl)
- void **vsxseg4ev\_int32xm2\_int32x4xm2** (int \*address, int32xm2\_t index, int32x4xm2\_t a, unsigned int gvl)
- void **vsxseg4ev\_int64xm1\_int64x4xm1** (long \*address, int64xm1\_t index, int64x4xm1\_t a, unsigned int gvl)
- void **vsxseg4ev\_int64xm2\_int64x4xm2** (long \*address, int64xm2\_t index, int64x4xm2\_t a, unsigned int gvl)
- void **vsxseg4ev\_int8xm1\_int8x4xm1** (signed char \*address, int8xm1\_t index, int8x4xm1\_t a, unsigned int gvl)
- void **vsxseg4ev\_int8xm2\_int8x4xm2** (signed char \*address, int8xm2\_t index, int8x4xm2\_t a, unsigned int gvl)
- void **vsxseg4ev\_uint16xm1\_uint16x4xm1** (unsigned short \*address, uint16xm1\_t index, uint16x4xm1\_t a, unsigned int gvl)
- void **vsxseg4ev\_uint16xm2\_uint16x4xm2** (unsigned short \*address, uint16xm2\_t index, uint16x4xm2\_t a, unsigned int gvl)
- void **vsxseg4ev\_uint32xm1\_uint32x4xm1** (unsigned int \*address, uint32xm1\_t index, uint32x4xm1\_t a, unsigned int gvl)
- void **vsxseg4ev\_uint32xm2\_uint32x4xm2** (unsigned int \*address, uint32xm2\_t index, uint32x4xm2\_t a, unsigned int gvl)
- void **vsxseg4ev\_uint64xm1\_uint64x4xm1** (unsigned long \*address, uint64xm1\_t index, uint64x4xm1\_t a, unsigned int gvl)
- void **vsxseg4ev\_uint64xm2\_uint64x4xm2** (unsigned long \*address, uint64xm2\_t index, uint64x4xm2\_t a, unsigned int gvl)
- void **vsxseg4ev\_uint8xm1\_uint8x4xm1** (unsigned char \*address, uint8xm1\_t index, uint8x4xm1\_t a, unsigned int gvl)
- void **vsxseg4ev\_uint8xm2\_uint8x4xm2** (unsigned char \*address, uint8xm2\_t index, uint8x4xm2\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg4ev\_mask\_float16xm1\_float16x4xm1** (float16\_t \*address, *float16xm1\_t* index, float16x4xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_float16xm2\_float16x4xm2** (float16\_t \*address, *float16xm2\_t* index, float16x4xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_float32xm1\_float32x4xm1** (float \*address, *float32xm1\_t* index, float32x4xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_float32xm2\_float32x4xm2** (float \*address, *float32xm2\_t* index, float32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_float64xm1\_float64x4xm1** (double \*address, *float64xm1\_t* index, float64x4xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_float64xm2\_float64x4xm2** (double \*address, *float64xm2\_t* index, float64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_int16xm1\_int16x4xm1** (short \*address, *int16xm1\_t* index, int16x4xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_int16xm2\_int16x4xm2** (short \*address, *int16xm2\_t* index, int16x4xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_int32xm1\_int32x4xm1** (int \*address, *int32xm1\_t* index, int32x4xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_int32xm2\_int32x4xm2** (int \*address, *int32xm2\_t* index, int32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_int64xm1\_int64x4xm1** (long \*address, *int64xm1\_t* index, int64x4xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_int64xm2\_int64x4xm2** (long \*address, *int64xm2\_t* index, int64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_int8xm1\_int8x4xm1** (signed char \*address, *int8xm1\_t* index, int8x4xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_int8xm2\_int8x4xm2** (signed char \*address, *int8xm2\_t* index, int8x4xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_uint16xm1\_uint16x4xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x4xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_uint16xm2\_uint16x4xm2** (unsigned short \*address, *uint16xm2\_t* index, uint16x4xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_uint32xm1\_uint32x4xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x4xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)

- void **vsxseg4ev\_mask\_uint32xm2\_uint32x4xm2** (unsigned int \*address, *uint32xm2\_t* index, uint32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_uint64xm1\_uint64x4xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x4xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_uint64xm2\_uint64x4xm2** (unsigned long \*address, *uint64xm2\_t* index, uint64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_uint8xm1\_uint8x4xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x4xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg4ev\_mask\_uint8xm2\_uint8x4xm2** (unsigned char \*address, *uint8xm2\_t* index, uint8x4xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)

#### Masked operation:

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

### 2.10.214 Store 4 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg4h.v']

#### Prototypes:

- void **vsxseg4hv\_int16xm1\_int16x4xm1** (short \*address, *int16xm1\_t* index, int16x4xm1\_t a, unsigned int gvl)
- void **vsxseg4hv\_int16xm2\_int16x4xm2** (short \*address, *int16xm2\_t* index, int16x4xm2\_t a, unsigned int gvl)
- void **vsxseg4hv\_int32xm1\_int32x4xm1** (int \*address, *int32xm1\_t* index, int32x4xm1\_t a, unsigned int gvl)
- void **vsxseg4hv\_int32xm2\_int32x4xm2** (int \*address, *int32xm2\_t* index, int32x4xm2\_t a, unsigned int gvl)
- void **vsxseg4hv\_int64xm1\_int64x4xm1** (long \*address, *int64xm1\_t* index, int64x4xm1\_t a, unsigned int gvl)
- void **vsxseg4hv\_int64xm2\_int64x4xm2** (long \*address, *int64xm2\_t* index, int64x4xm2\_t a, unsigned int gvl)
- void **vsxseg4hv\_int8xm1\_int8x4xm1** (signed char \*address, *int8xm1\_t* index, int8x4xm1\_t a, unsigned int gvl)
- void **vsxseg4hv\_int8xm2\_int8x4xm2** (signed char \*address, *int8xm2\_t* index, int8x4xm2\_t a, unsigned int gvl)
- void **vsxseg4hv\_uint16xm1\_uint16x4xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x4xm1\_t a, unsigned int gvl)
- void **vsxseg4hv\_uint16xm2\_uint16x4xm2** (unsigned short \*address, *uint16xm2\_t* index, uint16x4xm2\_t a, unsigned int gvl)

- void **vsxseg4hv\_uint32xm1\_uint32x4xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x4xm1\_t a, unsigned int gvl)
- void **vsxseg4hv\_uint32xm2\_uint32x4xm2** (unsigned int \*address, *uint32xm2\_t* index, uint32x4xm2\_t a, unsigned int gvl)
- void **vsxseg4hv\_uint64xm1\_uint64x4xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x4xm1\_t a, unsigned int gvl)
- void **vsxseg4hv\_uint64xm2\_uint64x4xm2** (unsigned long \*address, *uint64xm2\_t* index, uint64x4xm2\_t a, unsigned int gvl)
- void **vsxseg4hv\_uint8xm1\_uint8x4xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x4xm1\_t a, unsigned int gvl)
- void **vsxseg4hv\_uint8xm2\_uint8x4xm2** (unsigned char \*address, *uint8xm2\_t* index, uint8x4xm2\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg4hv\_mask\_int16xm1\_int16x4xm1** (short \*address, *int16xm1\_t* index, int16x4xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_int16xm2\_int16x4xm2** (short \*address, *int16xm2\_t* index, int16x4xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_int32xm1\_int32x4xm1** (int \*address, *int32xm1\_t* index, int32x4xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_int32xm2\_int32x4xm2** (int \*address, *int32xm2\_t* index, int32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_int64xm1\_int64x4xm1** (long \*address, *int64xm1\_t* index, int64x4xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_int64xm2\_int64x4xm2** (long \*address, *int64xm2\_t* index, int64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_int8xm1\_int8x4xm1** (signed char \*address, *int8xm1\_t* index, int8x4xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_int8xm2\_int8x4xm2** (signed char \*address, *int8xm2\_t* index, int8x4xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_uint16xm1\_uint16x4xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x4xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_uint16xm2\_uint16x4xm2** (unsigned short \*address, *uint16xm2\_t* index, uint16x4xm2\_t a, *e16xm2\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_uint32xm1\_uint32x4xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x4xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)

- void **vsxseg4hv\_mask\_uint32xm2\_uint32x4xm2** (unsigned int \*address, *uint32xm2\_t* index, uint32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_uint64xm1\_uint64x4xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x4xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_uint64xm2\_uint64x4xm2** (unsigned long \*address, *uint64xm2\_t* index, uint64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_uint8xm1\_uint8x4xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x4xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg4hv\_mask\_uint8xm2\_uint8x4xm2** (unsigned char \*address, *uint8xm2\_t* index, uint8x4xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)

#### Masked operation:

```
>>> for segment(4 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

### 2.10.215 Store 4 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg4w.v']

#### Prototypes:

- void **vsxseg4wv\_int16xm1\_int16x4xm1** (short \*address, *int16xm1\_t* index, int16x4xm1\_t a, unsigned int gvl)
- void **vsxseg4wv\_int16xm2\_int16x4xm2** (short \*address, *int16xm2\_t* index, int16x4xm2\_t a, unsigned int gvl)
- void **vsxseg4wv\_int32xm1\_int32x4xm1** (int \*address, *int32xm1\_t* index, int32x4xm1\_t a, unsigned int gvl)
- void **vsxseg4wv\_int32xm2\_int32x4xm2** (int \*address, *int32xm2\_t* index, int32x4xm2\_t a, unsigned int gvl)
- void **vsxseg4wv\_int64xm1\_int64x4xm1** (long \*address, *int64xm1\_t* index, int64x4xm1\_t a, unsigned int gvl)
- void **vsxseg4wv\_int64xm2\_int64x4xm2** (long \*address, *int64xm2\_t* index, int64x4xm2\_t a, unsigned int gvl)
- void **vsxseg4wv\_int8xm1\_int8x4xm1** (signed char \*address, *int8xm1\_t* index, int8x4xm1\_t a, unsigned int gvl)
- void **vsxseg4wv\_int8xm2\_int8x4xm2** (signed char \*address, *int8xm2\_t* index, int8x4xm2\_t a, unsigned int gvl)
- void **vsxseg4wv\_uint16xm1\_uint16x4xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x4xm1\_t a, unsigned int gvl)
- void **vsxseg4wv\_uint16xm2\_uint16x4xm2** (unsigned short \*address, *uint16xm2\_t* index, uint16x4xm2\_t a, unsigned int gvl)



- void **vsxseg4wv\_uint32xm1\_uint32x4xm1** (unsigned int \*address, [uint32xm1\\_t](#) index, uint32x4xm1\_t a, unsigned int gvl)
- void **vsxseg4wv\_uint32xm2\_uint32x4xm2** (unsigned int \*address, [uint32xm2\\_t](#) index, uint32x4xm2\_t a, unsigned int gvl)
- void **vsxseg4wv\_uint64xm1\_uint64x4xm1** (unsigned long \*address, [uint64xm1\\_t](#) index, uint64x4xm1\_t a, unsigned int gvl)
- void **vsxseg4wv\_uint64xm2\_uint64x4xm2** (unsigned long \*address, [uint64xm2\\_t](#) index, uint64x4xm2\_t a, unsigned int gvl)
- void **vsxseg4wv\_uint8xm1\_uint8x4xm1** (unsigned char \*address, [uint8xm1\\_t](#) index, uint8x4xm1\_t a, unsigned int gvl)
- void **vsxseg4wv\_uint8xm2\_uint8x4xm2** (unsigned char \*address, [uint8xm2\\_t](#) index, uint8x4xm2\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg4wv\_mask\_int16xm1\_int16x4xm1** (short \*address, [int16xm1\\_t](#) index, int16x4xm1\_t a, [e16xm1\\_t](#) mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_int16xm2\_int16x4xm2** (short \*address, [int16xm2\\_t](#) index, int16x4xm2\_t a, [e16xm2\\_t](#) mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_int32xm1\_int32x4xm1** (int \*address, [int32xm1\\_t](#) index, int32x4xm1\_t a, [e32xm1\\_t](#) mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_int32xm2\_int32x4xm2** (int \*address, [int32xm2\\_t](#) index, int32x4xm2\_t a, [e32xm2\\_t](#) mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_int64xm1\_int64x4xm1** (long \*address, [int64xm1\\_t](#) index, int64x4xm1\_t a, [e64xm1\\_t](#) mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_int64xm2\_int64x4xm2** (long \*address, [int64xm2\\_t](#) index, int64x4xm2\_t a, [e64xm2\\_t](#) mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_int8xm1\_int8x4xm1** (signed char \*address, [int8xm1\\_t](#) index, int8x4xm1\_t a, [e8xm1\\_t](#) mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_int8xm2\_int8x4xm2** (signed char \*address, [int8xm2\\_t](#) index, int8x4xm2\_t a, [e8xm2\\_t](#) mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_uint16xm1\_uint16x4xm1** (unsigned short \*address, [uint16xm1\\_t](#) index, uint16x4xm1\_t a, [e16xm1\\_t](#) mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_uint16xm2\_uint16x4xm2** (unsigned short \*address, [uint16xm2\\_t](#) index, uint16x4xm2\_t a, [e16xm2\\_t](#) mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_uint32xm1\_uint32x4xm1** (unsigned int \*address, [uint32xm1\\_t](#) index, uint32x4xm1\_t a, [e32xm1\\_t](#) mask, unsigned int gvl)

- void **vsxseg4wv\_mask\_uint32xm2\_uint32x4xm2** (unsigned int \*address, *uint32xm2\_t* index, uint32x4xm2\_t a, *e32xm2\_t* mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_uint64xm1\_uint64x4xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x4xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_uint64xm2\_uint64x4xm2** (unsigned long \*address, *uint64xm2\_t* index, uint64x4xm2\_t a, *e64xm2\_t* mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_uint8xm1\_uint8x4xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x4xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg4wv\_mask\_uint8xm2\_uint8x4xm2** (unsigned char \*address, *uint8xm2\_t* index, uint8x4xm2\_t a, *e8xm2\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(4 fields) = 0 to gvl - 1
      if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

### 2.10.216 Store 5 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg5b.v']

**Prototypes:**

- void **vsxseg5bv\_int16xm1\_int16x5xm1** (short \*address, *int16xm1\_t* index, int16x5xm1\_t a, unsigned int gvl)
- void **vsxseg5bv\_int32xm1\_int32x5xm1** (int \*address, *int32xm1\_t* index, int32x5xm1\_t a, unsigned int gvl)
- void **vsxseg5bv\_int64xm1\_int64x5xm1** (long \*address, *int64xm1\_t* index, int64x5xm1\_t a, unsigned int gvl)
- void **vsxseg5bv\_int8xm1\_int8x5xm1** (signed char \*address, *int8xm1\_t* index, int8x5xm1\_t a, unsigned int gvl)
- void **vsxseg5bv\_uint16xm1\_uint16x5xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x5xm1\_t a, unsigned int gvl)
- void **vsxseg5bv\_uint32xm1\_uint32x5xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x5xm1\_t a, unsigned int gvl)
- void **vsxseg5bv\_uint64xm1\_uint64x5xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x5xm1\_t a, unsigned int gvl)
- void **vsxseg5bv\_uint8xm1\_uint8x5xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x5xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
      store_segment(address, a[segment])
      address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg5bv\_mask\_int16xm1\_int16x5xm1** (short *\*address*, *int16xm1\_t* *index*, *int16x5xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg5bv\_mask\_int32xm1\_int32x5xm1** (int *\*address*, *int32xm1\_t* *index*, *int32x5xm1\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg5bv\_mask\_int64xm1\_int64x5xm1** (long *\*address*, *int64xm1\_t* *index*, *int64x5xm1\_t* *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg5bv\_mask\_int8xm1\_int8x5xm1** (signed char *\*address*, *int8xm1\_t* *index*, *int8x5xm1\_t* *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg5bv\_mask\_uint16xm1\_uint16x5xm1** (unsigned short *\*address*, *uint16xm1\_t* *index*, *uint16x5xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg5bv\_mask\_uint32xm1\_uint32x5xm1** (unsigned int *\*address*, *uint32xm1\_t* *index*, *uint32x5xm1\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg5bv\_mask\_uint64xm1\_uint64x5xm1** (unsigned long *\*address*, *uint64xm1\_t* *index*, *uint64x5xm1\_t* *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg5bv\_mask\_uint8xm1\_uint8x5xm1** (unsigned char *\*address*, *uint8xm1\_t* *index*, *uint8x5xm1\_t* *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

**2.10.217 Store 5 contiguous element fields in memory(indexed) from consecutively numbered vector registers****Instruction:** ['vsxseg5e.v']**Prototypes:**

- void **vsxseg5ev\_float16xm1\_float16x5xm1** (float16\_t *\*address*, *float16xm1\_t* *index*, *float16x5xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg5ev\_float32xm1\_float32x5xm1** (float *\*address*, *float32xm1\_t* *index*, *float32x5xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg5ev\_float64xm1\_float64x5xm1** (double *\*address*, *float64xm1\_t* *index*, *float64x5xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg5ev\_int16xm1\_int16x5xm1** (short *\*address*, *int16xm1\_t* *index*, *int16x5xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg5ev\_int32xm1\_int32x5xm1** (int *\*address*, *int32xm1\_t* *index*, *int32x5xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg5ev\_int64xm1\_int64x5xm1** (long *\*address*, *int64xm1\_t* *index*, *int64x5xm1\_t* *a*, unsigned int *gvl*)

- void **vsxseg5ev\_int8xm1\_int8x5xm1** (signed char \*address, *int8xm1\_t* index, int8x5xm1\_t a, unsigned int gvl)
- void **vsxseg5ev\_uint16xm1\_uint16x5xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x5xm1\_t a, unsigned int gvl)
- void **vsxseg5ev\_uint32xm1\_uint32x5xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x5xm1\_t a, unsigned int gvl)
- void **vsxseg5ev\_uint64xm1\_uint64x5xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x5xm1\_t a, unsigned int gvl)
- void **vsxseg5ev\_uint8xm1\_uint8x5xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x5xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg5ev\_mask\_float16xm1\_float16x5xm1** (float16\_t \*address, *float16xm1\_t* index, float16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg5ev\_mask\_float32xm1\_float32x5xm1** (float \*address, *float32xm1\_t* index, float32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg5ev\_mask\_float64xm1\_float64x5xm1** (double \*address, *float64xm1\_t* index, float64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg5ev\_mask\_int16xm1\_int16x5xm1** (short \*address, *int16xm1\_t* index, int16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg5ev\_mask\_int32xm1\_int32x5xm1** (int \*address, *int32xm1\_t* index, int32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg5ev\_mask\_int64xm1\_int64x5xm1** (long \*address, *int64xm1\_t* index, int64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg5ev\_mask\_int8xm1\_int8x5xm1** (signed char \*address, *int8xm1\_t* index, int8x5xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg5ev\_mask\_uint16xm1\_uint16x5xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg5ev\_mask\_uint32xm1\_uint32x5xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg5ev\_mask\_uint64xm1\_uint64x5xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg5ev\_mask\_uint8xm1\_uint8x5xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x5xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

### 2.10.218 Store 5 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg5h.v']

**Prototypes:**

- void **vsxseg5hv\_int16xm1\_int16x5xm1** (short \*address, *int16xm1\_t* index, int16x5xm1\_t a, unsigned int gvl)
- void **vsxseg5hv\_int32xm1\_int32x5xm1** (int \*address, *int32xm1\_t* index, int32x5xm1\_t a, unsigned int gvl)
- void **vsxseg5hv\_int64xm1\_int64x5xm1** (long \*address, *int64xm1\_t* index, int64x5xm1\_t a, unsigned int gvl)
- void **vsxseg5hv\_int8xm1\_int8x5xm1** (signed char \*address, *int8xm1\_t* index, int8x5xm1\_t a, unsigned int gvl)
- void **vsxseg5hv\_uint16xm1\_uint16x5xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x5xm1\_t a, unsigned int gvl)
- void **vsxseg5hv\_uint32xm1\_uint32x5xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x5xm1\_t a, unsigned int gvl)
- void **vsxseg5hv\_uint64xm1\_uint64x5xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x5xm1\_t a, unsigned int gvl)
- void **vsxseg5hv\_uint8xm1\_uint8x5xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x5xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg5hv\_mask\_int16xm1\_int16x5xm1** (short \*address, *int16xm1\_t* index, int16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg5hv\_mask\_int32xm1\_int32x5xm1** (int \*address, *int32xm1\_t* index, int32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg5hv\_mask\_int64xm1\_int64x5xm1** (long \*address, *int64xm1\_t* index, int64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg5hv\_mask\_int8xm1\_int8x5xm1** (signed char \*address, *int8xm1\_t* index, int8x5xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg5hv\_mask\_uint16xm1\_uint16x5xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)

- void **vsxseg5hv\_mask\_uint32xm1\_uint32x5xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg5hv\_mask\_uint64xm1\_uint64x5xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg5hv\_mask\_uint8xm1\_uint8x5xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x5xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

### 2.10.219 Store 5 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg5w.v']

**Prototypes:**

- void **vsxseg5wv\_int16xm1\_int16x5xm1** (short \*address, *int16xm1\_t* index, int16x5xm1\_t a, unsigned int gvl)
- void **vsxseg5wv\_int32xm1\_int32x5xm1** (int \*address, *int32xm1\_t* index, int32x5xm1\_t a, unsigned int gvl)
- void **vsxseg5wv\_int64xm1\_int64x5xm1** (long \*address, *int64xm1\_t* index, int64x5xm1\_t a, unsigned int gvl)
- void **vsxseg5wv\_int8xm1\_int8x5xm1** (signed char \*address, *int8xm1\_t* index, int8x5xm1\_t a, unsigned int gvl)
- void **vsxseg5wv\_uint16xm1\_uint16x5xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x5xm1\_t a, unsigned int gvl)
- void **vsxseg5wv\_uint32xm1\_uint32x5xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x5xm1\_t a, unsigned int gvl)
- void **vsxseg5wv\_uint64xm1\_uint64x5xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x5xm1\_t a, unsigned int gvl)
- void **vsxseg5wv\_uint8xm1\_uint8x5xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x5xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(5 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg5wv\_mask\_int16xm1\_int16x5xm1** (short \*address, *int16xm1\_t* index, int16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)



- void **vsxseg5wv\_mask\_int32xm1\_int32x5xm1** (int \*address, *int32xm1\_t* index, int32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg5wv\_mask\_int64xm1\_int64x5xm1** (long \*address, *int64xm1\_t* index, int64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg5wv\_mask\_int8xm1\_int8x5xm1** (signed char \*address, *int8xm1\_t* index, int8x5xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg5wv\_mask\_uint16xm1\_uint16x5xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x5xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg5wv\_mask\_uint32xm1\_uint32x5xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x5xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg5wv\_mask\_uint64xm1\_uint64x5xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x5xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg5wv\_mask\_uint8xm1\_uint8x5xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x5xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

#### Masked operation:

```
>>> for segment(5 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.220 Store 6 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg6b.v']

#### Prototypes:

- void **vsxseg6bv\_int16xm1\_int16x6xm1** (short \*address, *int16xm1\_t* index, int16x6xm1\_t a, unsigned int gvl)
- void **vsxseg6bv\_int32xm1\_int32x6xm1** (int \*address, *int32xm1\_t* index, int32x6xm1\_t a, unsigned int gvl)
- void **vsxseg6bv\_int64xm1\_int64x6xm1** (long \*address, *int64xm1\_t* index, int64x6xm1\_t a, unsigned int gvl)
- void **vsxseg6bv\_int8xm1\_int8x6xm1** (signed char \*address, *int8xm1\_t* index, int8x6xm1\_t a, unsigned int gvl)
- void **vsxseg6bv\_uint16xm1\_uint16x6xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x6xm1\_t a, unsigned int gvl)
- void **vsxseg6bv\_uint32xm1\_uint32x6xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x6xm1\_t a, unsigned int gvl)
- void **vsxseg6bv\_uint64xm1\_uint64x6xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x6xm1\_t a, unsigned int gvl)
- void **vsxseg6bv\_uint8xm1\_uint8x6xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x6xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg6bv\_mask\_int16xm1\_int16x6xm1** (short \*address, *int16xm1\_t* index, int16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg6bv\_mask\_int32xm1\_int32x6xm1** (int \*address, *int32xm1\_t* index, int32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg6bv\_mask\_int64xm1\_int64x6xm1** (long \*address, *int64xm1\_t* index, int64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg6bv\_mask\_int8xm1\_int8x6xm1** (signed char \*address, *int8xm1\_t* index, int8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg6bv\_mask\_uint16xm1\_uint16x6xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg6bv\_mask\_uint32xm1\_uint32x6xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg6bv\_mask\_uint64xm1\_uint64x6xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg6bv\_mask\_uint8xm1\_uint8x6xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.221 Store 6 contiguous element fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg6e.v']

**Prototypes:**

- void **vsxseg6ev\_float16xm1\_float16x6xm1** (float16\_t \*address, *float16xm1\_t* index, float16x6xm1\_t a, unsigned int gvl)
- void **vsxseg6ev\_float32xm1\_float32x6xm1** (float \*address, *float32xm1\_t* index, float32x6xm1\_t a, unsigned int gvl)
- void **vsxseg6ev\_float64xm1\_float64x6xm1** (double \*address, *float64xm1\_t* index, float64x6xm1\_t a, unsigned int gvl)
- void **vsxseg6ev\_int16xm1\_int16x6xm1** (short \*address, *int16xm1\_t* index, int16x6xm1\_t a, unsigned int gvl)

- void **vsxseg6ev\_int32xm1\_int32x6xm1** (int \*address, *int32xm1\_t* index, int32x6xm1\_t a, unsigned int gvl)
- void **vsxseg6ev\_int64xm1\_int64x6xm1** (long \*address, *int64xm1\_t* index, int64x6xm1\_t a, unsigned int gvl)
- void **vsxseg6ev\_int8xm1\_int8x6xm1** (signed char \*address, *int8xm1\_t* index, int8x6xm1\_t a, unsigned int gvl)
- void **vsxseg6ev\_uint16xm1\_uint16x6xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x6xm1\_t a, unsigned int gvl)
- void **vsxseg6ev\_uint32xm1\_uint32x6xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x6xm1\_t a, unsigned int gvl)
- void **vsxseg6ev\_uint64xm1\_uint64x6xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x6xm1\_t a, unsigned int gvl)
- void **vsxseg6ev\_uint8xm1\_uint8x6xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x6xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg6ev\_mask\_float16xm1\_float16x6xm1** (float16\_t \*address, *float16xm1\_t* index, float16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg6ev\_mask\_float32xm1\_float32x6xm1** (float \*address, *float32xm1\_t* index, float32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg6ev\_mask\_float64xm1\_float64x6xm1** (double \*address, *float64xm1\_t* index, float64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg6ev\_mask\_int16xm1\_int16x6xm1** (short \*address, *int16xm1\_t* index, int16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg6ev\_mask\_int32xm1\_int32x6xm1** (int \*address, *int32xm1\_t* index, int32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg6ev\_mask\_int64xm1\_int64x6xm1** (long \*address, *int64xm1\_t* index, int64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg6ev\_mask\_int8xm1\_int8x6xm1** (signed char \*address, *int8xm1\_t* index, int8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg6ev\_mask\_uint16xm1\_uint16x6xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg6ev\_mask\_uint32xm1\_uint32x6xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg6ev\_mask\_uint64xm1\_uint64x6xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)

- void **vsxseg6ev\_mask\_uint8xm1\_uint8x6xm1** (unsigned char \*address, [uint8xm1\\_t](#) index, uint8x6xm1\_t a, [e8xm1\\_t](#) mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.222 Store 6 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** [`'vsxseg6h.v'`]

**Prototypes:**

- void **vsxseg6hv\_int16xm1\_int16x6xm1** (short \*address, [int16xm1\\_t](#) index, int16x6xm1\_t a, unsigned int gvl)
- void **vsxseg6hv\_int32xm1\_int32x6xm1** (int \*address, [int32xm1\\_t](#) index, int32x6xm1\_t a, unsigned int gvl)
- void **vsxseg6hv\_int64xm1\_int64x6xm1** (long \*address, [int64xm1\\_t](#) index, int64x6xm1\_t a, unsigned int gvl)
- void **vsxseg6hv\_int8xm1\_int8x6xm1** (signed char \*address, [int8xm1\\_t](#) index, int8x6xm1\_t a, unsigned int gvl)
- void **vsxseg6hv\_uint16xm1\_uint16x6xm1** (unsigned short \*address, [uint16xm1\\_t](#) index, uint16x6xm1\_t a, unsigned int gvl)
- void **vsxseg6hv\_uint32xm1\_uint32x6xm1** (unsigned int \*address, [uint32xm1\\_t](#) index, uint32x6xm1\_t a, unsigned int gvl)
- void **vsxseg6hv\_uint64xm1\_uint64x6xm1** (unsigned long \*address, [uint64xm1\\_t](#) index, uint64x6xm1\_t a, unsigned int gvl)
- void **vsxseg6hv\_uint8xm1\_uint8x6xm1** (unsigned char \*address, [uint8xm1\\_t](#) index, uint8x6xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(6 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg6hv\_mask\_int16xm1\_int16x6xm1** (short \*address, [int16xm1\\_t](#) index, int16x6xm1\_t a, [e16xm1\\_t](#) mask, unsigned int gvl)
- void **vsxseg6hv\_mask\_int32xm1\_int32x6xm1** (int \*address, [int32xm1\\_t](#) index, int32x6xm1\_t a, [e32xm1\\_t](#) mask, unsigned int gvl)
- void **vsxseg6hv\_mask\_int64xm1\_int64x6xm1** (long \*address, [int64xm1\\_t](#) index, int64x6xm1\_t a, [e64xm1\\_t](#) mask, unsigned int gvl)
- void **vsxseg6hv\_mask\_int8xm1\_int8x6xm1** (signed char \*address, [int8xm1\\_t](#) index, int8x6xm1\_t a, [e8xm1\\_t](#) mask, unsigned int gvl)

- void **vsxseg6hv\_mask\_uint16xm1\_uint16x6xm1** (unsigned short *\*address*, *uint16xm1\_t* *index*, *uint16x6xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg6hv\_mask\_uint32xm1\_uint32x6xm1** (unsigned int *\*address*, *uint32xm1\_t* *index*, *uint32x6xm1\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg6hv\_mask\_uint64xm1\_uint64x6xm1** (unsigned long *\*address*, *uint64xm1\_t* *index*, *uint64x6xm1\_t* *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg6hv\_mask\_uint8xm1\_uint8x6xm1** (unsigned char *\*address*, *uint8xm1\_t* *index*, *uint8x6xm1\_t* *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

Masked operation:

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.223 Store 6 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers

Instruction: ['vsxseg6w.v']

Prototypes:

- void **vsxseg6wv\_int16xm1\_int16x6xm1** (short *\*address*, *int16xm1\_t* *index*, *int16x6xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg6wv\_int32xm1\_int32x6xm1** (int *\*address*, *int32xm1\_t* *index*, *int32x6xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg6wv\_int64xm1\_int64x6xm1** (long *\*address*, *int64xm1\_t* *index*, *int64x6xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg6wv\_int8xm1\_int8x6xm1** (signed char *\*address*, *int8xm1\_t* *index*, *int8x6xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg6wv\_uint16xm1\_uint16x6xm1** (unsigned short *\*address*, *uint16xm1\_t* *index*, *uint16x6xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg6wv\_uint32xm1\_uint32x6xm1** (unsigned int *\*address*, *uint32xm1\_t* *index*, *uint32x6xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg6wv\_uint64xm1\_uint64x6xm1** (unsigned long *\*address*, *uint64xm1\_t* *index*, *uint64x6xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg6wv\_uint8xm1\_uint8x6xm1** (unsigned char *\*address*, *uint8xm1\_t* *index*, *uint8x6xm1\_t* *a*, unsigned int *gvl*)

Operation:

```
>>> for segment(6 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

Masked prototypes:

- void **vsxseg6wv\_mask\_int16xm1\_int16x6xm1** (short \*address, *int16xm1\_t* index, int16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg6wv\_mask\_int32xm1\_int32x6xm1** (int \*address, *int32xm1\_t* index, int32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg6wv\_mask\_int64xm1\_int64x6xm1** (long \*address, *int64xm1\_t* index, int64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg6wv\_mask\_int8xm1\_int8x6xm1** (signed char \*address, *int8xm1\_t* index, int8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg6wv\_mask\_uint16xm1\_uint16x6xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x6xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg6wv\_mask\_uint32xm1\_uint32x6xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x6xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg6wv\_mask\_uint64xm1\_uint64x6xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x6xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg6wv\_mask\_uint8xm1\_uint8x6xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x6xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

#### Masked operation:

```
>>> for segment(6 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

### 2.10.224 Store 7 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg7b.v']

#### Prototypes:

- void **vsxseg7bv\_int16xm1\_int16x7xm1** (short \*address, *int16xm1\_t* index, int16x7xm1\_t a, unsigned int gvl)
- void **vsxseg7bv\_int32xm1\_int32x7xm1** (int \*address, *int32xm1\_t* index, int32x7xm1\_t a, unsigned int gvl)
- void **vsxseg7bv\_int64xm1\_int64x7xm1** (long \*address, *int64xm1\_t* index, int64x7xm1\_t a, unsigned int gvl)
- void **vsxseg7bv\_int8xm1\_int8x7xm1** (signed char \*address, *int8xm1\_t* index, int8x7xm1\_t a, unsigned int gvl)
- void **vsxseg7bv\_uint16xm1\_uint16x7xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x7xm1\_t a, unsigned int gvl)
- void **vsxseg7bv\_uint32xm1\_uint32x7xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x7xm1\_t a, unsigned int gvl)
- void **vsxseg7bv\_uint64xm1\_uint64x7xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x7xm1\_t a, unsigned int gvl)



- void **vsxseg7bv\_uint8xm1\_uint8x7xm1** (unsigned char \*address, uint8xm1\_t index, uint8x7xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg7bv\_mask\_int16xm1\_int16x7xm1** (short \*address, int16xm1\_t index, int16x7xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsxseg7bv\_mask\_int32xm1\_int32x7xm1** (int \*address, int32xm1\_t index, int32x7xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsxseg7bv\_mask\_int64xm1\_int64x7xm1** (long \*address, int64xm1\_t index, int64x7xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg7bv\_mask\_int8xm1\_int8x7xm1** (signed char \*address, int8xm1\_t index, int8x7xm1\_t a, e8xm1\_t mask, unsigned int gvl)
- void **vsxseg7bv\_mask\_uint16xm1\_uint16x7xm1** (unsigned short \*address, uint16xm1\_t index, uint16x7xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsxseg7bv\_mask\_uint32xm1\_uint32x7xm1** (unsigned int \*address, uint32xm1\_t index, uint32x7xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsxseg7bv\_mask\_uint64xm1\_uint64x7xm1** (unsigned long \*address, uint64xm1\_t index, uint64x7xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg7bv\_mask\_uint8xm1\_uint8x7xm1** (unsigned char \*address, uint8xm1\_t index, uint8x7xm1\_t a, e8xm1\_t mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.225 Store 7 contiguous element fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** [`'vsxseg7e.v'`]

**Prototypes:**

- void **vsxseg7ev\_float16xm1\_float16x7xm1** (float16\_t \*address, float16xm1\_t index, float16x7xm1\_t a, unsigned int gvl)
- void **vsxseg7ev\_float32xm1\_float32x7xm1** (float \*address, float32xm1\_t index, float32x7xm1\_t a, unsigned int gvl)
- void **vsxseg7ev\_float64xm1\_float64x7xm1** (double \*address, float64xm1\_t index, float64x7xm1\_t a, unsigned int gvl)

- void **vsxseg7ev\_int16xm1\_int16x7xm1** (short \*address, *int16xm1\_t* index, int16x7xm1\_t a, unsigned int gvl)
- void **vsxseg7ev\_int32xm1\_int32x7xm1** (int \*address, *int32xm1\_t* index, int32x7xm1\_t a, unsigned int gvl)
- void **vsxseg7ev\_int64xm1\_int64x7xm1** (long \*address, *int64xm1\_t* index, int64x7xm1\_t a, unsigned int gvl)
- void **vsxseg7ev\_int8xm1\_int8x7xm1** (signed char \*address, *int8xm1\_t* index, int8x7xm1\_t a, unsigned int gvl)
- void **vsxseg7ev\_uint16xm1\_uint16x7xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x7xm1\_t a, unsigned int gvl)
- void **vsxseg7ev\_uint32xm1\_uint32x7xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x7xm1\_t a, unsigned int gvl)
- void **vsxseg7ev\_uint64xm1\_uint64x7xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x7xm1\_t a, unsigned int gvl)
- void **vsxseg7ev\_uint8xm1\_uint8x7xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x7xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg7ev\_mask\_float16xm1\_float16x7xm1** (float16\_t \*address, *float16xm1\_t* index, float16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg7ev\_mask\_float32xm1\_float32x7xm1** (float \*address, *float32xm1\_t* index, float32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg7ev\_mask\_float64xm1\_float64x7xm1** (double \*address, *float64xm1\_t* index, float64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg7ev\_mask\_int16xm1\_int16x7xm1** (short \*address, *int16xm1\_t* index, int16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg7ev\_mask\_int32xm1\_int32x7xm1** (int \*address, *int32xm1\_t* index, int32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg7ev\_mask\_int64xm1\_int64x7xm1** (long \*address, *int64xm1\_t* index, int64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg7ev\_mask\_int8xm1\_int8x7xm1** (signed char \*address, *int8xm1\_t* index, int8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg7ev\_mask\_uint16xm1\_uint16x7xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg7ev\_mask\_uint32xm1\_uint32x7xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)

- void **vsxseg7ev\_mask\_uint64xm1\_uint64x7xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg7ev\_mask\_uint8xm1\_uint8x7xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

Masked operation:

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.226 Store 7 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers

Instruction: ['vsxseg7h.v']

Prototypes:

- void **vsxseg7hv\_int16xm1\_int16x7xm1** (short \*address, *int16xm1\_t* index, int16x7xm1\_t a, unsigned int gvl)
- void **vsxseg7hv\_int32xm1\_int32x7xm1** (int \*address, *int32xm1\_t* index, int32x7xm1\_t a, unsigned int gvl)
- void **vsxseg7hv\_int64xm1\_int64x7xm1** (long \*address, *int64xm1\_t* index, int64x7xm1\_t a, unsigned int gvl)
- void **vsxseg7hv\_int8xm1\_int8x7xm1** (signed char \*address, *int8xm1\_t* index, int8x7xm1\_t a, unsigned int gvl)
- void **vsxseg7hv\_uint16xm1\_uint16x7xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x7xm1\_t a, unsigned int gvl)
- void **vsxseg7hv\_uint32xm1\_uint32x7xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x7xm1\_t a, unsigned int gvl)
- void **vsxseg7hv\_uint64xm1\_uint64x7xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x7xm1\_t a, unsigned int gvl)
- void **vsxseg7hv\_uint8xm1\_uint8x7xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x7xm1\_t a, unsigned int gvl)

Operation:

```
>>> for segment(7 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

Masked prototypes:

- void **vsxseg7hv\_mask\_int16xm1\_int16x7xm1** (short \*address, *int16xm1\_t* index, int16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg7hv\_mask\_int32xm1\_int32x7xm1** (int \*address, *int32xm1\_t* index, int32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)

- void **vsxseg7hv\_mask\_int64xm1\_int64x7xm1** (long \*address, *int64xm1\_t* index, *int64x7xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg7hv\_mask\_int8xm1\_int8x7xm1** (signed char \*address, *int8xm1\_t* index, *int8x7xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg7hv\_mask\_uint16xm1\_uint16x7xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16x7xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg7hv\_mask\_uint32xm1\_uint32x7xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32x7xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg7hv\_mask\_uint64xm1\_uint64x7xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64x7xm1\_t* a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg7hv\_mask\_uint8xm1\_uint8x7xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8x7xm1\_t* a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.227 Store 7 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** [*'vsxseg7w.v'*]

**Prototypes:**

- void **vsxseg7wv\_int16xm1\_int16x7xm1** (short \*address, *int16xm1\_t* index, *int16x7xm1\_t* a, unsigned int gvl)
- void **vsxseg7wv\_int32xm1\_int32x7xm1** (int \*address, *int32xm1\_t* index, *int32x7xm1\_t* a, unsigned int gvl)
- void **vsxseg7wv\_int64xm1\_int64x7xm1** (long \*address, *int64xm1\_t* index, *int64x7xm1\_t* a, unsigned int gvl)
- void **vsxseg7wv\_int8xm1\_int8x7xm1** (signed char \*address, *int8xm1\_t* index, *int8x7xm1\_t* a, unsigned int gvl)
- void **vsxseg7wv\_uint16xm1\_uint16x7xm1** (unsigned short \*address, *uint16xm1\_t* index, *uint16x7xm1\_t* a, unsigned int gvl)
- void **vsxseg7wv\_uint32xm1\_uint32x7xm1** (unsigned int \*address, *uint32xm1\_t* index, *uint32x7xm1\_t* a, unsigned int gvl)
- void **vsxseg7wv\_uint64xm1\_uint64x7xm1** (unsigned long \*address, *uint64xm1\_t* index, *uint64x7xm1\_t* a, unsigned int gvl)
- void **vsxseg7wv\_uint8xm1\_uint8x7xm1** (unsigned char \*address, *uint8xm1\_t* index, *uint8x7xm1\_t* a, unsigned int gvl)

**Operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg7wv\_mask\_int16xm1\_int16x7xm1** (short \*address, *int16xm1\_t* index, int16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg7wv\_mask\_int32xm1\_int32x7xm1** (int \*address, *int32xm1\_t* index, int32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg7wv\_mask\_int64xm1\_int64x7xm1** (long \*address, *int64xm1\_t* index, int64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg7wv\_mask\_int8xm1\_int8x7xm1** (signed char \*address, *int8xm1\_t* index, int8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg7wv\_mask\_uint16xm1\_uint16x7xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x7xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg7wv\_mask\_uint32xm1\_uint32x7xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x7xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg7wv\_mask\_uint64xm1\_uint64x7xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x7xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg7wv\_mask\_uint8xm1\_uint8x7xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x7xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(7 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

**2.10.228 Store 8 contiguous 8b fields in memory(indexed) from consecutively numbered vector registers****Instruction:** ['vsxseg8b.v']**Prototypes:**

- void **vsxseg8bv\_int16xm1\_int16x8xm1** (short \*address, *int16xm1\_t* index, int16x8xm1\_t a, unsigned int gvl)
- void **vsxseg8bv\_int32xm1\_int32x8xm1** (int \*address, *int32xm1\_t* index, int32x8xm1\_t a, unsigned int gvl)
- void **vsxseg8bv\_int64xm1\_int64x8xm1** (long \*address, *int64xm1\_t* index, int64x8xm1\_t a, unsigned int gvl)
- void **vsxseg8bv\_int8xm1\_int8x8xm1** (signed char \*address, *int8xm1\_t* index, int8x8xm1\_t a, unsigned int gvl)

- void **vsxseg8bv\_uint16xm1\_uint16x8xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x8xm1\_t a, unsigned int gvl)
- void **vsxseg8bv\_uint32xm1\_uint32x8xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x8xm1\_t a, unsigned int gvl)
- void **vsxseg8bv\_uint64xm1\_uint64x8xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x8xm1\_t a, unsigned int gvl)
- void **vsxseg8bv\_uint8xm1\_uint8x8xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x8xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg8bv\_mask\_int16xm1\_int16x8xm1** (short \*address, *int16xm1\_t* index, int16x8xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg8bv\_mask\_int32xm1\_int32x8xm1** (int \*address, *int32xm1\_t* index, int32x8xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg8bv\_mask\_int64xm1\_int64x8xm1** (long \*address, *int64xm1\_t* index, int64x8xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg8bv\_mask\_int8xm1\_int8x8xm1** (signed char \*address, *int8xm1\_t* index, int8x8xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg8bv\_mask\_uint16xm1\_uint16x8xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x8xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg8bv\_mask\_uint32xm1\_uint32x8xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x8xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg8bv\_mask\_uint64xm1\_uint64x8xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x8xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg8bv\_mask\_uint8xm1\_uint8x8xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x8xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.229 Store 8 contiguous element fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg8e.v']

**Prototypes:**



- void **vsxseg8ev\_float16xm1\_float16x8xm1** (float16\_t \*address, float16xm1\_t index, float16x8xm1\_t a, unsigned int gvl)
- void **vsxseg8ev\_float32xm1\_float32x8xm1** (float \*address, float32xm1\_t index, float32x8xm1\_t a, unsigned int gvl)
- void **vsxseg8ev\_float64xm1\_float64x8xm1** (double \*address, float64xm1\_t index, float64x8xm1\_t a, unsigned int gvl)
- void **vsxseg8ev\_int16xm1\_int16x8xm1** (short \*address, int16xm1\_t index, int16x8xm1\_t a, unsigned int gvl)
- void **vsxseg8ev\_int32xm1\_int32x8xm1** (int \*address, int32xm1\_t index, int32x8xm1\_t a, unsigned int gvl)
- void **vsxseg8ev\_int64xm1\_int64x8xm1** (long \*address, int64xm1\_t index, int64x8xm1\_t a, unsigned int gvl)
- void **vsxseg8ev\_int8xm1\_int8x8xm1** (signed char \*address, int8xm1\_t index, int8x8xm1\_t a, unsigned int gvl)
- void **vsxseg8ev\_uint16xm1\_uint16x8xm1** (unsigned short \*address, uint16xm1\_t index, uint16x8xm1\_t a, unsigned int gvl)
- void **vsxseg8ev\_uint32xm1\_uint32x8xm1** (unsigned int \*address, uint32xm1\_t index, uint32x8xm1\_t a, unsigned int gvl)
- void **vsxseg8ev\_uint64xm1\_uint64x8xm1** (unsigned long \*address, uint64xm1\_t index, uint64x8xm1\_t a, unsigned int gvl)
- void **vsxseg8ev\_uint8xm1\_uint8x8xm1** (unsigned char \*address, uint8xm1\_t index, uint8x8xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg8ev\_mask\_float16xm1\_float16x8xm1** (float16\_t \*address, float16xm1\_t index, float16x8xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsxseg8ev\_mask\_float32xm1\_float32x8xm1** (float \*address, float32xm1\_t index, float32x8xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsxseg8ev\_mask\_float64xm1\_float64x8xm1** (double \*address, float64xm1\_t index, float64x8xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg8ev\_mask\_int16xm1\_int16x8xm1** (short \*address, int16xm1\_t index, int16x8xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsxseg8ev\_mask\_int32xm1\_int32x8xm1** (int \*address, int32xm1\_t index, int32x8xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsxseg8ev\_mask\_int64xm1\_int64x8xm1** (long \*address, int64xm1\_t index, int64x8xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg8ev\_mask\_int8xm1\_int8x8xm1** (signed char \*address, int8xm1\_t index, int8x8xm1\_t a, e8xm1\_t mask, unsigned int gvl)

- void **vsxseg8ev\_mask\_uint16xm1\_uint16x8xm1** (unsigned short *\*address*, *uint16xm1\_t* *index*, *uint16x8xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg8ev\_mask\_uint32xm1\_uint32x8xm1** (unsigned int *\*address*, *uint32xm1\_t* *index*, *uint32x8xm1\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg8ev\_mask\_uint64xm1\_uint64x8xm1** (unsigned long *\*address*, *uint64xm1\_t* *index*, *uint64x8xm1\_t* *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- void **vsxseg8ev\_mask\_uint8xm1\_uint8x8xm1** (unsigned char *\*address*, *uint8xm1\_t* *index*, *uint8x8xm1\_t* *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)

Masked operation:

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.10.230 Store 8 contiguous 16b fields in memory(indexed) from consecutively numbered vector registers

Instruction: ['vsxseg8h.v']

Prototypes:

- void **vsxseg8hv\_int16xm1\_int16x8xm1** (short *\*address*, *int16xm1\_t* *index*, *int16x8xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg8hv\_int32xm1\_int32x8xm1** (int *\*address*, *int32xm1\_t* *index*, *int32x8xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg8hv\_int64xm1\_int64x8xm1** (long *\*address*, *int64xm1\_t* *index*, *int64x8xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg8hv\_int8xm1\_int8x8xm1** (signed char *\*address*, *int8xm1\_t* *index*, *int8x8xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg8hv\_uint16xm1\_uint16x8xm1** (unsigned short *\*address*, *uint16xm1\_t* *index*, *uint16x8xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg8hv\_uint32xm1\_uint32x8xm1** (unsigned int *\*address*, *uint32xm1\_t* *index*, *uint32x8xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg8hv\_uint64xm1\_uint64x8xm1** (unsigned long *\*address*, *uint64xm1\_t* *index*, *uint64x8xm1\_t* *a*, unsigned int *gvl*)
- void **vsxseg8hv\_uint8xm1\_uint8x8xm1** (unsigned char *\*address*, *uint8xm1\_t* *index*, *uint8x8xm1\_t* *a*, unsigned int *gvl*)

Operation:

```
>>> for segment(8 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

Masked prototypes:

- void **vsxseg8hv\_mask\_int16xm1\_int16x8xm1** (short \*address, *int16xm1\_t* index, int16x8xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg8hv\_mask\_int32xm1\_int32x8xm1** (int \*address, *int32xm1\_t* index, int32x8xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg8hv\_mask\_int64xm1\_int64x8xm1** (long \*address, *int64xm1\_t* index, int64x8xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg8hv\_mask\_int8xm1\_int8x8xm1** (signed char \*address, *int8xm1\_t* index, int8x8xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)
- void **vsxseg8hv\_mask\_uint16xm1\_uint16x8xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x8xm1\_t a, *e16xm1\_t* mask, unsigned int gvl)
- void **vsxseg8hv\_mask\_uint32xm1\_uint32x8xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x8xm1\_t a, *e32xm1\_t* mask, unsigned int gvl)
- void **vsxseg8hv\_mask\_uint64xm1\_uint64x8xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x8xm1\_t a, *e64xm1\_t* mask, unsigned int gvl)
- void **vsxseg8hv\_mask\_uint8xm1\_uint8x8xm1** (unsigned char \*address, *uint8xm1\_t* index, uint8x8xm1\_t a, *e8xm1\_t* mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

### 2.10.231 Store 8 contiguous 32b fields in memory(indexed) from consecutively numbered vector registers

**Instruction:** ['vsxseg8w.v']

**Prototypes:**

- void **vsxseg8wv\_int16xm1\_int16x8xm1** (short \*address, *int16xm1\_t* index, int16x8xm1\_t a, unsigned int gvl)
- void **vsxseg8wv\_int32xm1\_int32x8xm1** (int \*address, *int32xm1\_t* index, int32x8xm1\_t a, unsigned int gvl)
- void **vsxseg8wv\_int64xm1\_int64x8xm1** (long \*address, *int64xm1\_t* index, int64x8xm1\_t a, unsigned int gvl)
- void **vsxseg8wv\_int8xm1\_int8x8xm1** (signed char \*address, *int8xm1\_t* index, int8x8xm1\_t a, unsigned int gvl)
- void **vsxseg8wv\_uint16xm1\_uint16x8xm1** (unsigned short \*address, *uint16xm1\_t* index, uint16x8xm1\_t a, unsigned int gvl)
- void **vsxseg8wv\_uint32xm1\_uint32x8xm1** (unsigned int \*address, *uint32xm1\_t* index, uint32x8xm1\_t a, unsigned int gvl)
- void **vsxseg8wv\_uint64xm1\_uint64x8xm1** (unsigned long \*address, *uint64xm1\_t* index, uint64x8xm1\_t a, unsigned int gvl)

- void **vsxseg8wv\_uint8xm1\_uint8x8xm1** (unsigned char \*address, uint8xm1\_t index, uint8x8xm1\_t a, unsigned int gvl)

**Operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    store_segment(address, a[segment])
    address = address + sizeof(segment) + index[segment]
```

**Masked prototypes:**

- void **vsxseg8wv\_mask\_int16xm1\_int16x8xm1** (short \*address, int16xm1\_t index, int16x8xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsxseg8wv\_mask\_int32xm1\_int32x8xm1** (int \*address, int32xm1\_t index, int32x8xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsxseg8wv\_mask\_int64xm1\_int64x8xm1** (long \*address, int64xm1\_t index, int64x8xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg8wv\_mask\_int8xm1\_int8x8xm1** (signed char \*address, int8xm1\_t index, int8x8xm1\_t a, e8xm1\_t mask, unsigned int gvl)
- void **vsxseg8wv\_mask\_uint16xm1\_uint16x8xm1** (unsigned short \*address, uint16xm1\_t index, uint16x8xm1\_t a, e16xm1\_t mask, unsigned int gvl)
- void **vsxseg8wv\_mask\_uint32xm1\_uint32x8xm1** (unsigned int \*address, uint32xm1\_t index, uint32x8xm1\_t a, e32xm1\_t mask, unsigned int gvl)
- void **vsxseg8wv\_mask\_uint64xm1\_uint64x8xm1** (unsigned long \*address, uint64xm1\_t index, uint64x8xm1\_t a, e64xm1\_t mask, unsigned int gvl)
- void **vsxseg8wv\_mask\_uint8xm1\_uint8x8xm1** (unsigned char \*address, uint8xm1\_t index, uint8x8xm1\_t a, e8xm1\_t mask, unsigned int gvl)

**Masked operation:**

```
>>> for segment(8 fields) = 0 to gvl - 1
    if mask[segment] then
        store_segment(address, a[segment])
        address = address + sizeof(segment) + index[segment]
```

## 2.11 Operations with masks

### 2.11.1 Compute elementwise logical and between two masks

**Instruction:** ['vmmand.mm']**Prototypes:**

- e16xm1\_t **vmmandmm\_e16xm1** (e16xm1\_t a, e16xm1\_t b, unsigned int gvl)
- e16xm2\_t **vmmandmm\_e16xm2** (e16xm2\_t a, e16xm2\_t b, unsigned int gvl)
- e16xm4\_t **vmmandmm\_e16xm4** (e16xm4\_t a, e16xm4\_t b, unsigned int gvl)

- *e16xm8\_t vmandmm\_e16xm8* (*e16xm8\_t a, e16xm8\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmandmm\_e32xm1* (*e32xm1\_t a, e32xm1\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmandmm\_e32xm2* (*e32xm2\_t a, e32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmandmm\_e32xm4* (*e32xm4\_t a, e32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmandmm\_e32xm8* (*e32xm8\_t a, e32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmandmm\_e64xm1* (*e64xm1\_t a, e64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmandmm\_e64xm2* (*e64xm2\_t a, e64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmandmm\_e64xm4* (*e64xm4\_t a, e64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmandmm\_e64xm8* (*e64xm8\_t a, e64xm8\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmandmm\_e8xm1* (*e8xm1\_t a, e8xm1\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmandmm\_e8xm2* (*e8xm2\_t a, e8xm2\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmandmm\_e8xm4* (*e8xm4\_t a, e8xm4\_t b*, unsigned int *gvl*)
- *e8xm8\_t vmandmm\_e8xm8* (*e8xm8\_t a, e8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = logical_and (a[element], b[element])
result[gvl : VLMAX] = 0
```

**2.11.2 Compute elementwise logical andnot between two masks****Instruction:** ['vmandnot.mm']**Prototypes:**

- *e16xm1\_t vmandnotmm\_e16xm1* (*e16xm1\_t a, e16xm1\_t b*, unsigned int *gvl*)
- *e16xm2\_t vmandnotmm\_e16xm2* (*e16xm2\_t a, e16xm2\_t b*, unsigned int *gvl*)
- *e16xm4\_t vmandnotmm\_e16xm4* (*e16xm4\_t a, e16xm4\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmandnotmm\_e16xm8* (*e16xm8\_t a, e16xm8\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmandnotmm\_e32xm1* (*e32xm1\_t a, e32xm1\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmandnotmm\_e32xm2* (*e32xm2\_t a, e32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmandnotmm\_e32xm4* (*e32xm4\_t a, e32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmandnotmm\_e32xm8* (*e32xm8\_t a, e32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmandnotmm\_e64xm1* (*e64xm1\_t a, e64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmandnotmm\_e64xm2* (*e64xm2\_t a, e64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmandnotmm\_e64xm4* (*e64xm4\_t a, e64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmandnotmm\_e64xm8* (*e64xm8\_t a, e64xm8\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmandnotmm\_e8xm1* (*e8xm1\_t a, e8xm1\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmandnotmm\_e8xm2* (*e8xm2\_t a, e8xm2\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmandnotmm\_e8xm4* (*e8xm4\_t a, e8xm4\_t b*, unsigned int *gvl*)

- *e8xm8\_t* **vmmandnotmm\_e8xm8** (*e8xm8\_t* *a*, *e8xm8\_t* *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = logical_andnot (a[element], b[element])
result[gvl : VLMAX] = 0
```

### 2.11.3 Clear elementwise mask

**Instruction:** ['vmclr.m']**Prototypes:**

- *e16xm1\_t* **vmclrmm\_e16xm1** (unsigned int *gvl*)
- *e16xm2\_t* **vmclrmm\_e16xm2** (unsigned int *gvl*)
- *e16xm4\_t* **vmclrmm\_e16xm4** (unsigned int *gvl*)
- *e16xm8\_t* **vmclrmm\_e16xm8** (unsigned int *gvl*)
- *e32xm1\_t* **vmclrmm\_e32xm1** (unsigned int *gvl*)
- *e32xm2\_t* **vmclrmm\_e32xm2** (unsigned int *gvl*)
- *e32xm4\_t* **vmclrmm\_e32xm4** (unsigned int *gvl*)
- *e32xm8\_t* **vmclrmm\_e32xm8** (unsigned int *gvl*)
- *e64xm1\_t* **vmclrmm\_e64xm1** (unsigned int *gvl*)
- *e64xm2\_t* **vmclrmm\_e64xm2** (unsigned int *gvl*)
- *e64xm4\_t* **vmclrmm\_e64xm4** (unsigned int *gvl*)
- *e64xm8\_t* **vmclrmm\_e64xm8** (unsigned int *gvl*)
- *e8xm1\_t* **vmclrmm\_e8xm1** (unsigned int *gvl*)
- *e8xm2\_t* **vmclrmm\_e8xm2** (unsigned int *gvl*)
- *e8xm4\_t* **vmclrmm\_e8xm4** (unsigned int *gvl*)
- *e8xm8\_t* **vmclrmm\_e8xm8** (unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = logical_clr (result[element])
result[gvl : VLMAX] = 0
```

### 2.11.4 Copy elementwise mask

**Instruction:** ['vmcpy.m']**Prototypes:**

- *e16xm1\_t* **vmcpymm\_e16xm1** (*e16xm1\_t* *a*, unsigned int *gvl*)
- *e16xm2\_t* **vmcpymm\_e16xm2** (*e16xm2\_t* *a*, unsigned int *gvl*)
- *e16xm4\_t* **vmcpymm\_e16xm4** (*e16xm4\_t* *a*, unsigned int *gvl*)



- *e16xm8\_t vmcpym\_e16xm8* (*e16xm8\_t a*, unsigned int *gvl*)
- *e32xm1\_t vmcpym\_e32xm1* (*e32xm1\_t a*, unsigned int *gvl*)
- *e32xm2\_t vmcpym\_e32xm2* (*e32xm2\_t a*, unsigned int *gvl*)
- *e32xm4\_t vmcpym\_e32xm4* (*e32xm4\_t a*, unsigned int *gvl*)
- *e32xm8\_t vmcpym\_e32xm8* (*e32xm8\_t a*, unsigned int *gvl*)
- *e64xm1\_t vmcpym\_e64xm1* (*e64xm1\_t a*, unsigned int *gvl*)
- *e64xm2\_t vmcpym\_e64xm2* (*e64xm2\_t a*, unsigned int *gvl*)
- *e64xm4\_t vmcpym\_e64xm4* (*e64xm4\_t a*, unsigned int *gvl*)
- *e64xm8\_t vmcpym\_e64xm8* (*e64xm8\_t a*, unsigned int *gvl*)
- *e8xm1\_t vmcpym\_e8xm1* (*e8xm1\_t a*, unsigned int *gvl*)
- *e8xm2\_t vmcpym\_e8xm2* (*e8xm2\_t a*, unsigned int *gvl*)
- *e8xm4\_t vmcpym\_e8xm4* (*e8xm4\_t a*, unsigned int *gvl*)
- *e8xm8\_t vmcpym\_e8xm8* (*e8xm8\_t a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = logical_cpy (a[element])
result[gvl : VLMAX] = 0
```

## 2.11.5 Compute the index of the first enabled element

**Instruction:** ['vmfirst.m']**Prototypes:**

- long **vmfirstm\_e16xm1** (*e16xm1\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e16xm2** (*e16xm2\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e16xm4** (*e16xm4\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e16xm8** (*e16xm8\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e32xm1** (*e32xm1\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e32xm2** (*e32xm2\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e32xm4** (*e32xm4\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e32xm8** (*e32xm8\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e64xm1** (*e64xm1\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e64xm2** (*e64xm2\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e64xm4** (*e64xm4\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e64xm8** (*e64xm8\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e8xm1** (*e8xm1\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e8xm2** (*e8xm2\_t a*, unsigned int *gvl*)
- long **vmfirstm\_e8xm4** (*e8xm4\_t a*, unsigned int *gvl*)

- long **vmfirstm\_e8xm8** (*e8xm8\_t a*, unsigned int *gvl*)

**Operation:**

```
>>> result = -1
    for element = 0 to gvl - 1
        if a[element]
            result = element
            break
```

**Masked prototypes:**

- long **vmfirstm\_mask\_e16xm1** (*e16xm1\_t a*, *e16xm1\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e16xm2** (*e16xm2\_t a*, *e16xm2\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e16xm4** (*e16xm4\_t a*, *e16xm4\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e16xm8** (*e16xm8\_t a*, *e16xm8\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e32xm1** (*e32xm1\_t a*, *e32xm1\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e32xm2** (*e32xm2\_t a*, *e32xm2\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e32xm4** (*e32xm4\_t a*, *e32xm4\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e32xm8** (*e32xm8\_t a*, *e32xm8\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e64xm1** (*e64xm1\_t a*, *e64xm1\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e64xm2** (*e64xm2\_t a*, *e64xm2\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e64xm4** (*e64xm4\_t a*, *e64xm4\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e64xm8** (*e64xm8\_t a*, *e64xm8\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e8xm1** (*e8xm1\_t a*, *e8xm1\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e8xm2** (*e8xm2\_t a*, *e8xm2\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e8xm4** (*e8xm4\_t a*, *e8xm4\_t mask*, unsigned int *gvl*)
- long **vmfirstm\_mask\_e8xm8** (*e8xm8\_t a*, *e8xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> result = -1
    for element = 0 to gvl - 1
        if mask[element] and a[element] then
            result = element
            break
```

## 2.11.6 Compute elementwise logical negated and between two masks

**Instruction:** ['vmnand.mm']**Prototypes:**

- *e16xm1\_t* **vmnandmm\_e16xm1** (*e16xm1\_t a*, *e16xm1\_t b*, unsigned int *gvl*)
- *e16xm2\_t* **vmnandmm\_e16xm2** (*e16xm2\_t a*, *e16xm2\_t b*, unsigned int *gvl*)
- *e16xm4\_t* **vmnandmm\_e16xm4** (*e16xm4\_t a*, *e16xm4\_t b*, unsigned int *gvl*)
- *e16xm8\_t* **vmnandmm\_e16xm8** (*e16xm8\_t a*, *e16xm8\_t b*, unsigned int *gvl*)

- `e32xm1_t vmnandmm_e32xm1` (`e32xm1_t a`, `e32xm1_t b`, unsigned int `gvl`)
- `e32xm2_t vmnandmm_e32xm2` (`e32xm2_t a`, `e32xm2_t b`, unsigned int `gvl`)
- `e32xm4_t vmnandmm_e32xm4` (`e32xm4_t a`, `e32xm4_t b`, unsigned int `gvl`)
- `e32xm8_t vmnandmm_e32xm8` (`e32xm8_t a`, `e32xm8_t b`, unsigned int `gvl`)
- `e64xm1_t vmnandmm_e64xm1` (`e64xm1_t a`, `e64xm1_t b`, unsigned int `gvl`)
- `e64xm2_t vmnandmm_e64xm2` (`e64xm2_t a`, `e64xm2_t b`, unsigned int `gvl`)
- `e64xm4_t vmnandmm_e64xm4` (`e64xm4_t a`, `e64xm4_t b`, unsigned int `gvl`)
- `e64xm8_t vmnandmm_e64xm8` (`e64xm8_t a`, `e64xm8_t b`, unsigned int `gvl`)
- `e8xm1_t vmnandmm_e8xm1` (`e8xm1_t a`, `e8xm1_t b`, unsigned int `gvl`)
- `e8xm2_t vmnandmm_e8xm2` (`e8xm2_t a`, `e8xm2_t b`, unsigned int `gvl`)
- `e8xm4_t vmnandmm_e8xm4` (`e8xm4_t a`, `e8xm4_t b`, unsigned int `gvl`)
- `e8xm8_t vmnandmm_e8xm8` (`e8xm8_t a`, `e8xm8_t b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = logical_nand (a[element], b[element])
result[gvl : VLMAX] = 0
```

**2.11.7 Compute elementwise logical negated or between two masks****Instruction:** [`'vmnor.mm'`]**Prototypes:**

- `e16xm1_t vmnormm_e16xm1` (`e16xm1_t a`, `e16xm1_t b`, unsigned int `gvl`)
- `e16xm2_t vmnormm_e16xm2` (`e16xm2_t a`, `e16xm2_t b`, unsigned int `gvl`)
- `e16xm4_t vmnormm_e16xm4` (`e16xm4_t a`, `e16xm4_t b`, unsigned int `gvl`)
- `e16xm8_t vmnormm_e16xm8` (`e16xm8_t a`, `e16xm8_t b`, unsigned int `gvl`)
- `e32xm1_t vmnormm_e32xm1` (`e32xm1_t a`, `e32xm1_t b`, unsigned int `gvl`)
- `e32xm2_t vmnormm_e32xm2` (`e32xm2_t a`, `e32xm2_t b`, unsigned int `gvl`)
- `e32xm4_t vmnormm_e32xm4` (`e32xm4_t a`, `e32xm4_t b`, unsigned int `gvl`)
- `e32xm8_t vmnormm_e32xm8` (`e32xm8_t a`, `e32xm8_t b`, unsigned int `gvl`)
- `e64xm1_t vmnormm_e64xm1` (`e64xm1_t a`, `e64xm1_t b`, unsigned int `gvl`)
- `e64xm2_t vmnormm_e64xm2` (`e64xm2_t a`, `e64xm2_t b`, unsigned int `gvl`)
- `e64xm4_t vmnormm_e64xm4` (`e64xm4_t a`, `e64xm4_t b`, unsigned int `gvl`)
- `e64xm8_t vmnormm_e64xm8` (`e64xm8_t a`, `e64xm8_t b`, unsigned int `gvl`)
- `e8xm1_t vmnormm_e8xm1` (`e8xm1_t a`, `e8xm1_t b`, unsigned int `gvl`)
- `e8xm2_t vmnormm_e8xm2` (`e8xm2_t a`, `e8xm2_t b`, unsigned int `gvl`)
- `e8xm4_t vmnormm_e8xm4` (`e8xm4_t a`, `e8xm4_t b`, unsigned int `gvl`)
- `e8xm8_t vmnormm_e8xm8` (`e8xm8_t a`, `e8xm8_t b`, unsigned int `gvl`)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = logical_nor (a[element], b[element])
result[gvl : VLMAX] = 0
```

**2.11.8 Invert elementwise mask****Instruction:** ['vmnot.m']**Prototypes:**

- *e16xm1\_t vmnotm\_e16xm1* (*e16xm1\_t a*, unsigned int *gvl*)
- *e16xm2\_t vmnotm\_e16xm2* (*e16xm2\_t a*, unsigned int *gvl*)
- *e16xm4\_t vmnotm\_e16xm4* (*e16xm4\_t a*, unsigned int *gvl*)
- *e16xm8\_t vmnotm\_e16xm8* (*e16xm8\_t a*, unsigned int *gvl*)
- *e32xm1\_t vmnotm\_e32xm1* (*e32xm1\_t a*, unsigned int *gvl*)
- *e32xm2\_t vmnotm\_e32xm2* (*e32xm2\_t a*, unsigned int *gvl*)
- *e32xm4\_t vmnotm\_e32xm4* (*e32xm4\_t a*, unsigned int *gvl*)
- *e32xm8\_t vmnotm\_e32xm8* (*e32xm8\_t a*, unsigned int *gvl*)
- *e64xm1\_t vmnotm\_e64xm1* (*e64xm1\_t a*, unsigned int *gvl*)
- *e64xm2\_t vmnotm\_e64xm2* (*e64xm2\_t a*, unsigned int *gvl*)
- *e64xm4\_t vmnotm\_e64xm4* (*e64xm4\_t a*, unsigned int *gvl*)
- *e64xm8\_t vmnotm\_e64xm8* (*e64xm8\_t a*, unsigned int *gvl*)
- *e8xm1\_t vmnotm\_e8xm1* (*e8xm1\_t a*, unsigned int *gvl*)
- *e8xm2\_t vmnotm\_e8xm2* (*e8xm2\_t a*, unsigned int *gvl*)
- *e8xm4\_t vmnotm\_e8xm4* (*e8xm4\_t a*, unsigned int *gvl*)
- *e8xm8\_t vmnotm\_e8xm8* (*e8xm8\_t a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = logical_not (a[element])
result[gvl : VLMAX] = 0
```

**2.11.9 Compute elementwise logical or between two masks****Instruction:** ['vmor.mm']**Prototypes:**

- *e16xm1\_t vmorimm\_e16xm1* (*e16xm1\_t a*, *e16xm1\_t b*, unsigned int *gvl*)
- *e16xm2\_t vmorimm\_e16xm2* (*e16xm2\_t a*, *e16xm2\_t b*, unsigned int *gvl*)
- *e16xm4\_t vmorimm\_e16xm4* (*e16xm4\_t a*, *e16xm4\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmorimm\_e16xm8* (*e16xm8\_t a*, *e16xm8\_t b*, unsigned int *gvl*)

- *e32xm1\_t vmorrm\_e32xm1* (*e32xm1\_t a, e32xm1\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmorrm\_e32xm2* (*e32xm2\_t a, e32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmorrm\_e32xm4* (*e32xm4\_t a, e32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmorrm\_e32xm8* (*e32xm8\_t a, e32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmorrm\_e64xm1* (*e64xm1\_t a, e64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmorrm\_e64xm2* (*e64xm2\_t a, e64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmorrm\_e64xm4* (*e64xm4\_t a, e64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmorrm\_e64xm8* (*e64xm8\_t a, e64xm8\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmorrm\_e8xm1* (*e8xm1\_t a, e8xm1\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmorrm\_e8xm2* (*e8xm2\_t a, e8xm2\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmorrm\_e8xm4* (*e8xm4\_t a, e8xm4\_t b*, unsigned int *gvl*)
- *e8xm8\_t vmorrm\_e8xm8* (*e8xm8\_t a, e8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = logical_or (a[element], b[element])
result[gvl : VLMAX] = 0
```

**2.11.10 Compute elementwise logical ornot between two masks****Instruction:** [*'vmornot.mm'*]**Prototypes:**

- *e16xm1\_t vmornotmm\_e16xm1* (*e16xm1\_t a, e16xm1\_t b*, unsigned int *gvl*)
- *e16xm2\_t vmornotmm\_e16xm2* (*e16xm2\_t a, e16xm2\_t b*, unsigned int *gvl*)
- *e16xm4\_t vmornotmm\_e16xm4* (*e16xm4\_t a, e16xm4\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmornotmm\_e16xm8* (*e16xm8\_t a, e16xm8\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmornotmm\_e32xm1* (*e32xm1\_t a, e32xm1\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmornotmm\_e32xm2* (*e32xm2\_t a, e32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmornotmm\_e32xm4* (*e32xm4\_t a, e32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmornotmm\_e32xm8* (*e32xm8\_t a, e32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmornotmm\_e64xm1* (*e64xm1\_t a, e64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmornotmm\_e64xm2* (*e64xm2\_t a, e64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmornotmm\_e64xm4* (*e64xm4\_t a, e64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmornotmm\_e64xm8* (*e64xm8\_t a, e64xm8\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmornotmm\_e8xm1* (*e8xm1\_t a, e8xm1\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmornotmm\_e8xm2* (*e8xm2\_t a, e8xm2\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmornotmm\_e8xm4* (*e8xm4\_t a, e8xm4\_t b*, unsigned int *gvl*)
- *e8xm8\_t vmornotmm\_e8xm8* (*e8xm8\_t a, e8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = logical_ornot (a[element], b[element])
result[gvl : VLMAX] = 0
```

**2.11.11 Population count of a mask vector****Instruction:** ['vmpopc.m']**Prototypes:**

- unsigned long **vmpopcm\_e16xm1** (*e16xm1\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e16xm2** (*e16xm2\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e16xm4** (*e16xm4\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e16xm8** (*e16xm8\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e32xm1** (*e32xm1\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e32xm2** (*e32xm2\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e32xm4** (*e32xm4\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e32xm8** (*e32xm8\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e64xm1** (*e64xm1\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e64xm2** (*e64xm2\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e64xm4** (*e64xm4\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e64xm8** (*e64xm8\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e8xm1** (*e8xm1\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e8xm2** (*e8xm2\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e8xm4** (*e8xm4\_t* a, unsigned int gvl)
- unsigned long **vmpopcm\_e8xm8** (*e8xm8\_t* a, unsigned int gvl)

**Operation:**

```
>>> result = 0
for element = 0 to gvl - 1
  if a[element] then
    result = result + 1
```

**Masked prototypes:**

- unsigned long **vmpopcm\_mask\_e16xm1** (*e16xm1\_t* a, *e16xm1\_t* mask, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e16xm2** (*e16xm2\_t* a, *e16xm2\_t* mask, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e16xm4** (*e16xm4\_t* a, *e16xm4\_t* mask, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e16xm8** (*e16xm8\_t* a, *e16xm8\_t* mask, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e32xm1** (*e32xm1\_t* a, *e32xm1\_t* mask, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e32xm2** (*e32xm2\_t* a, *e32xm2\_t* mask, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e32xm4** (*e32xm4\_t* a, *e32xm4\_t* mask, unsigned int gvl)



- unsigned long **vmpopcm\_mask\_e32xm8** (*e32xm8\_t a, e32xm8\_t mask*, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e64xm1** (*e64xm1\_t a, e64xm1\_t mask*, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e64xm2** (*e64xm2\_t a, e64xm2\_t mask*, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e64xm4** (*e64xm4\_t a, e64xm4\_t mask*, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e64xm8** (*e64xm8\_t a, e64xm8\_t mask*, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e8xm1** (*e8xm1\_t a, e8xm1\_t mask*, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e8xm2** (*e8xm2\_t a, e8xm2\_t mask*, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e8xm4** (*e8xm4\_t a, e8xm4\_t mask*, unsigned int gvl)
- unsigned long **vmpopcm\_mask\_e8xm8** (*e8xm8\_t a, e8xm8\_t mask*, unsigned int gvl)

**Masked operation:**

```
>>> result = 0
    for element = 0 to gvl - 1
        if mask[element] and a[element] then
            result = result + 1
```

### 2.11.12 Enable elements before the first one enabled

**Instruction:** ['vmsbfm']

**Prototypes:**

- *e16xm1\_t vmsbfm\_e16xm1* (*e16xm1\_t a*, unsigned int gvl)
- *e16xm2\_t vmsbfm\_e16xm2* (*e16xm2\_t a*, unsigned int gvl)
- *e16xm4\_t vmsbfm\_e16xm4* (*e16xm4\_t a*, unsigned int gvl)
- *e16xm8\_t vmsbfm\_e16xm8* (*e16xm8\_t a*, unsigned int gvl)
- *e32xm1\_t vmsbfm\_e32xm1* (*e32xm1\_t a*, unsigned int gvl)
- *e32xm2\_t vmsbfm\_e32xm2* (*e32xm2\_t a*, unsigned int gvl)
- *e32xm4\_t vmsbfm\_e32xm4* (*e32xm4\_t a*, unsigned int gvl)
- *e32xm8\_t vmsbfm\_e32xm8* (*e32xm8\_t a*, unsigned int gvl)
- *e64xm1\_t vmsbfm\_e64xm1* (*e64xm1\_t a*, unsigned int gvl)
- *e64xm2\_t vmsbfm\_e64xm2* (*e64xm2\_t a*, unsigned int gvl)
- *e64xm4\_t vmsbfm\_e64xm4* (*e64xm4\_t a*, unsigned int gvl)
- *e64xm8\_t vmsbfm\_e64xm8* (*e64xm8\_t a*, unsigned int gvl)
- *e8xm1\_t vmsbfm\_e8xm1* (*e8xm1\_t a*, unsigned int gvl)
- *e8xm2\_t vmsbfm\_e8xm2* (*e8xm2\_t a*, unsigned int gvl)
- *e8xm4\_t vmsbfm\_e8xm4* (*e8xm4\_t a*, unsigned int gvl)
- *e8xm8\_t vmsbfm\_e8xm8* (*e8xm8\_t a*, unsigned int gvl)

**Operation:**

```
>>> for element = 0 to gvl - 1
      if not a[element] then
        result[element] = 1
      else
        break
result[element : VLMAX] = 0
```

#### Masked prototypes:

- *e16xm1\_t vmsbfm\_mask\_e16xm1* (*e16xm1\_t a, e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmsbfm\_mask\_e16xm2* (*e16xm2\_t a, e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmsbfm\_mask\_e16xm4* (*e16xm4\_t a, e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmsbfm\_mask\_e16xm8* (*e16xm8\_t a, e16xm8\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmsbfm\_mask\_e32xm1* (*e32xm1\_t a, e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm2\_t vmsbfm\_mask\_e32xm2* (*e32xm2\_t a, e32xm2\_t mask*, unsigned int *gvl*)
- *e32xm4\_t vmsbfm\_mask\_e32xm4* (*e32xm4\_t a, e32xm4\_t mask*, unsigned int *gvl*)
- *e32xm8\_t vmsbfm\_mask\_e32xm8* (*e32xm8\_t a, e32xm8\_t mask*, unsigned int *gvl*)
- *e64xm1\_t vmsbfm\_mask\_e64xm1* (*e64xm1\_t a, e64xm1\_t mask*, unsigned int *gvl*)
- *e64xm2\_t vmsbfm\_mask\_e64xm2* (*e64xm2\_t a, e64xm2\_t mask*, unsigned int *gvl*)
- *e64xm4\_t vmsbfm\_mask\_e64xm4* (*e64xm4\_t a, e64xm4\_t mask*, unsigned int *gvl*)
- *e64xm8\_t vmsbfm\_mask\_e64xm8* (*e64xm8\_t a, e64xm8\_t mask*, unsigned int *gvl*)
- *e8xm1\_t vmsbfm\_mask\_e8xm1* (*e8xm1\_t a, e8xm1\_t mask*, unsigned int *gvl*)
- *e8xm2\_t vmsbfm\_mask\_e8xm2* (*e8xm2\_t a, e8xm2\_t mask*, unsigned int *gvl*)
- *e8xm4\_t vmsbfm\_mask\_e8xm4* (*e8xm4\_t a, e8xm4\_t mask*, unsigned int *gvl*)
- *e8xm8\_t vmsbfm\_mask\_e8xm8* (*e8xm8\_t a, e8xm8\_t mask*, unsigned int *gvl*)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        if not a[element] then
          result[element] = 1
        else
          break
result[element : VLMAX] = 0
```

### 2.11.13 Set elementwise mask

**Instruction:** ['vmset.m']

#### Prototypes:

- *e16xm1\_t vmsetm\_e16xm1* (unsigned int *gvl*)
- *e16xm2\_t vmsetm\_e16xm2* (unsigned int *gvl*)
- *e16xm4\_t vmsetm\_e16xm4* (unsigned int *gvl*)
- *e16xm8\_t vmsetm\_e16xm8* (unsigned int *gvl*)

- *e32xm1\_t* **vmsetm\_e32xm1** (unsigned int *gvl*)
- *e32xm2\_t* **vmsetm\_e32xm2** (unsigned int *gvl*)
- *e32xm4\_t* **vmsetm\_e32xm4** (unsigned int *gvl*)
- *e32xm8\_t* **vmsetm\_e32xm8** (unsigned int *gvl*)
- *e64xm1\_t* **vmsetm\_e64xm1** (unsigned int *gvl*)
- *e64xm2\_t* **vmsetm\_e64xm2** (unsigned int *gvl*)
- *e64xm4\_t* **vmsetm\_e64xm4** (unsigned int *gvl*)
- *e64xm8\_t* **vmsetm\_e64xm8** (unsigned int *gvl*)
- *e8xm1\_t* **vmsetm\_e8xm1** (unsigned int *gvl*)
- *e8xm2\_t* **vmsetm\_e8xm2** (unsigned int *gvl*)
- *e8xm4\_t* **vmsetm\_e8xm4** (unsigned int *gvl*)
- *e8xm8\_t* **vmsetm\_e8xm8** (unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = logical_set (result[element])
result[gvl : VLMAX] = 0
```

### 2.11.14 Enable elements until the first one enabled

**Instruction:** [*'vmsifm.m'*]

**Prototypes:**

- *e16xm1\_t* **vmsifm\_e16xm1** (*e16xm1\_t a*, unsigned int *gvl*)
- *e16xm2\_t* **vmsifm\_e16xm2** (*e16xm2\_t a*, unsigned int *gvl*)
- *e16xm4\_t* **vmsifm\_e16xm4** (*e16xm4\_t a*, unsigned int *gvl*)
- *e16xm8\_t* **vmsifm\_e16xm8** (*e16xm8\_t a*, unsigned int *gvl*)
- *e32xm1\_t* **vmsifm\_e32xm1** (*e32xm1\_t a*, unsigned int *gvl*)
- *e32xm2\_t* **vmsifm\_e32xm2** (*e32xm2\_t a*, unsigned int *gvl*)
- *e32xm4\_t* **vmsifm\_e32xm4** (*e32xm4\_t a*, unsigned int *gvl*)
- *e32xm8\_t* **vmsifm\_e32xm8** (*e32xm8\_t a*, unsigned int *gvl*)
- *e64xm1\_t* **vmsifm\_e64xm1** (*e64xm1\_t a*, unsigned int *gvl*)
- *e64xm2\_t* **vmsifm\_e64xm2** (*e64xm2\_t a*, unsigned int *gvl*)
- *e64xm4\_t* **vmsifm\_e64xm4** (*e64xm4\_t a*, unsigned int *gvl*)
- *e64xm8\_t* **vmsifm\_e64xm8** (*e64xm8\_t a*, unsigned int *gvl*)
- *e8xm1\_t* **vmsifm\_e8xm1** (*e8xm1\_t a*, unsigned int *gvl*)
- *e8xm2\_t* **vmsifm\_e8xm2** (*e8xm2\_t a*, unsigned int *gvl*)
- *e8xm4\_t* **vmsifm\_e8xm4** (*e8xm4\_t a*, unsigned int *gvl*)
- *e8xm8\_t* **vmsifm\_e8xm8** (*e8xm8\_t a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = 1
      if a[element] then
        break
      result[element : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t vmsifm\_mask\_e16xm1* (*e16xm1\_t a, e16xm1\_t mask*, unsigned int *gvl*)
- *e16xm2\_t vmsifm\_mask\_e16xm2* (*e16xm2\_t a, e16xm2\_t mask*, unsigned int *gvl*)
- *e16xm4\_t vmsifm\_mask\_e16xm4* (*e16xm4\_t a, e16xm4\_t mask*, unsigned int *gvl*)
- *e16xm8\_t vmsifm\_mask\_e16xm8* (*e16xm8\_t a, e16xm8\_t mask*, unsigned int *gvl*)
- *e32xm1\_t vmsifm\_mask\_e32xm1* (*e32xm1\_t a, e32xm1\_t mask*, unsigned int *gvl*)
- *e32xm2\_t vmsifm\_mask\_e32xm2* (*e32xm2\_t a, e32xm2\_t mask*, unsigned int *gvl*)
- *e32xm4\_t vmsifm\_mask\_e32xm4* (*e32xm4\_t a, e32xm4\_t mask*, unsigned int *gvl*)
- *e32xm8\_t vmsifm\_mask\_e32xm8* (*e32xm8\_t a, e32xm8\_t mask*, unsigned int *gvl*)
- *e64xm1\_t vmsifm\_mask\_e64xm1* (*e64xm1\_t a, e64xm1\_t mask*, unsigned int *gvl*)
- *e64xm2\_t vmsifm\_mask\_e64xm2* (*e64xm2\_t a, e64xm2\_t mask*, unsigned int *gvl*)
- *e64xm4\_t vmsifm\_mask\_e64xm4* (*e64xm4\_t a, e64xm4\_t mask*, unsigned int *gvl*)
- *e64xm8\_t vmsifm\_mask\_e64xm8* (*e64xm8\_t a, e64xm8\_t mask*, unsigned int *gvl*)
- *e8xm1\_t vmsifm\_mask\_e8xm1* (*e8xm1\_t a, e8xm1\_t mask*, unsigned int *gvl*)
- *e8xm2\_t vmsifm\_mask\_e8xm2* (*e8xm2\_t a, e8xm2\_t mask*, unsigned int *gvl*)
- *e8xm4\_t vmsifm\_mask\_e8xm4* (*e8xm4\_t a, e8xm4\_t mask*, unsigned int *gvl*)
- *e8xm8\_t vmsifm\_mask\_e8xm8* (*e8xm8\_t a, e8xm8\_t mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = 1
        if a[element] then
          break
      result[element : VLMAX] = 0
```

**2.11.15 Enable only the first element enabled****Instruction:** ['vmsof.m']**Prototypes:**

- *e16xm1\_t vmsofm\_e16xm1* (*e16xm1\_t a*, unsigned int *gvl*)
- *e16xm2\_t vmsofm\_e16xm2* (*e16xm2\_t a*, unsigned int *gvl*)
- *e16xm4\_t vmsofm\_e16xm4* (*e16xm4\_t a*, unsigned int *gvl*)
- *e16xm8\_t vmsofm\_e16xm8* (*e16xm8\_t a*, unsigned int *gvl*)

- *e32xm1\_t* **vmsofm\_e32xm1** (*e32xm1\_t* *a*, unsigned int *gvl*)
- *e32xm2\_t* **vmsofm\_e32xm2** (*e32xm2\_t* *a*, unsigned int *gvl*)
- *e32xm4\_t* **vmsofm\_e32xm4** (*e32xm4\_t* *a*, unsigned int *gvl*)
- *e32xm8\_t* **vmsofm\_e32xm8** (*e32xm8\_t* *a*, unsigned int *gvl*)
- *e64xm1\_t* **vmsofm\_e64xm1** (*e64xm1\_t* *a*, unsigned int *gvl*)
- *e64xm2\_t* **vmsofm\_e64xm2** (*e64xm2\_t* *a*, unsigned int *gvl*)
- *e64xm4\_t* **vmsofm\_e64xm4** (*e64xm4\_t* *a*, unsigned int *gvl*)
- *e64xm8\_t* **vmsofm\_e64xm8** (*e64xm8\_t* *a*, unsigned int *gvl*)
- *e8xm1\_t* **vmsofm\_e8xm1** (*e8xm1\_t* *a*, unsigned int *gvl*)
- *e8xm2\_t* **vmsofm\_e8xm2** (*e8xm2\_t* *a*, unsigned int *gvl*)
- *e8xm4\_t* **vmsofm\_e8xm4** (*e8xm4\_t* *a*, unsigned int *gvl*)
- *e8xm8\_t* **vmsofm\_e8xm8** (*e8xm8\_t* *a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      if a[element] then
        result[element] = 1
        break
      else
        result[element] = 0
result[element : VLMAX] = 0
```

**Masked prototypes:**

- *e16xm1\_t* **vmsofm\_mask\_e16xm1** (*e16xm1\_t* *a*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *e16xm2\_t* **vmsofm\_mask\_e16xm2** (*e16xm2\_t* *a*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *e16xm4\_t* **vmsofm\_mask\_e16xm4** (*e16xm4\_t* *a*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *e16xm8\_t* **vmsofm\_mask\_e16xm8** (*e16xm8\_t* *a*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- *e32xm1\_t* **vmsofm\_mask\_e32xm1** (*e32xm1\_t* *a*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *e32xm2\_t* **vmsofm\_mask\_e32xm2** (*e32xm2\_t* *a*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *e32xm4\_t* **vmsofm\_mask\_e32xm4** (*e32xm4\_t* *a*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *e32xm8\_t* **vmsofm\_mask\_e32xm8** (*e32xm8\_t* *a*, *e32xm8\_t* *mask*, unsigned int *gvl*)
- *e64xm1\_t* **vmsofm\_mask\_e64xm1** (*e64xm1\_t* *a*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *e64xm2\_t* **vmsofm\_mask\_e64xm2** (*e64xm2\_t* *a*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *e64xm4\_t* **vmsofm\_mask\_e64xm4** (*e64xm4\_t* *a*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *e64xm8\_t* **vmsofm\_mask\_e64xm8** (*e64xm8\_t* *a*, *e64xm8\_t* *mask*, unsigned int *gvl*)
- *e8xm1\_t* **vmsofm\_mask\_e8xm1** (*e8xm1\_t* *a*, *e8xm1\_t* *mask*, unsigned int *gvl*)
- *e8xm2\_t* **vmsofm\_mask\_e8xm2** (*e8xm2\_t* *a*, *e8xm2\_t* *mask*, unsigned int *gvl*)
- *e8xm4\_t* **vmsofm\_mask\_e8xm4** (*e8xm4\_t* *a*, *e8xm4\_t* *mask*, unsigned int *gvl*)
- *e8xm8\_t* **vmsofm\_mask\_e8xm8** (*e8xm8\_t* *a*, *e8xm8\_t* *mask*, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        if a[element] then
            result[element] = 1
            break
        else
            result[element] = 0
    result[element : VLMAX] = 0
```

### 2.11.16 Compute elementwise logical xnor between two masks

**Instruction:** ['vmxnor.mm']

**Prototypes:**

- *e16xm1\_t vmxnor\_mm\_e16xm1* (*e16xm1\_t a, e16xm1\_t b*, unsigned int *gvl*)
- *e16xm2\_t vmxnor\_mm\_e16xm2* (*e16xm2\_t a, e16xm2\_t b*, unsigned int *gvl*)
- *e16xm4\_t vmxnor\_mm\_e16xm4* (*e16xm4\_t a, e16xm4\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmxnor\_mm\_e16xm8* (*e16xm8\_t a, e16xm8\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmxnor\_mm\_e32xm1* (*e32xm1\_t a, e32xm1\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmxnor\_mm\_e32xm2* (*e32xm2\_t a, e32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmxnor\_mm\_e32xm4* (*e32xm4\_t a, e32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmxnor\_mm\_e32xm8* (*e32xm8\_t a, e32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmxnor\_mm\_e64xm1* (*e64xm1\_t a, e64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmxnor\_mm\_e64xm2* (*e64xm2\_t a, e64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmxnor\_mm\_e64xm4* (*e64xm4\_t a, e64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmxnor\_mm\_e64xm8* (*e64xm8\_t a, e64xm8\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmxnor\_mm\_e8xm1* (*e8xm1\_t a, e8xm1\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmxnor\_mm\_e8xm2* (*e8xm2\_t a, e8xm2\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmxnor\_mm\_e8xm4* (*e8xm4\_t a, e8xm4\_t b*, unsigned int *gvl*)
- *e8xm8\_t vmxnor\_mm\_e8xm8* (*e8xm8\_t a, e8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = logical_xnor (a[element], b[element])
result[gvl : VLMAX] = 0
```

### 2.11.17 Compute elementwise logical xor between two masks

**Instruction:** ['vmxor.mm']

**Prototypes:**

- *e16xm1\_t vmxor\_mm\_e16xm1* (*e16xm1\_t a, e16xm1\_t b*, unsigned int *gvl*)
- *e16xm2\_t vmxor\_mm\_e16xm2* (*e16xm2\_t a, e16xm2\_t b*, unsigned int *gvl*)



- *e16xm4\_t vmxor\_mm\_e16xm4* (*e16xm4\_t a, e16xm4\_t b*, unsigned int *gvl*)
- *e16xm8\_t vmxor\_mm\_e16xm8* (*e16xm8\_t a, e16xm8\_t b*, unsigned int *gvl*)
- *e32xm1\_t vmxor\_mm\_e32xm1* (*e32xm1\_t a, e32xm1\_t b*, unsigned int *gvl*)
- *e32xm2\_t vmxor\_mm\_e32xm2* (*e32xm2\_t a, e32xm2\_t b*, unsigned int *gvl*)
- *e32xm4\_t vmxor\_mm\_e32xm4* (*e32xm4\_t a, e32xm4\_t b*, unsigned int *gvl*)
- *e32xm8\_t vmxor\_mm\_e32xm8* (*e32xm8\_t a, e32xm8\_t b*, unsigned int *gvl*)
- *e64xm1\_t vmxor\_mm\_e64xm1* (*e64xm1\_t a, e64xm1\_t b*, unsigned int *gvl*)
- *e64xm2\_t vmxor\_mm\_e64xm2* (*e64xm2\_t a, e64xm2\_t b*, unsigned int *gvl*)
- *e64xm4\_t vmxor\_mm\_e64xm4* (*e64xm4\_t a, e64xm4\_t b*, unsigned int *gvl*)
- *e64xm8\_t vmxor\_mm\_e64xm8* (*e64xm8\_t a, e64xm8\_t b*, unsigned int *gvl*)
- *e8xm1\_t vmxor\_mm\_e8xm1* (*e8xm1\_t a, e8xm1\_t b*, unsigned int *gvl*)
- *e8xm2\_t vmxor\_mm\_e8xm2* (*e8xm2\_t a, e8xm2\_t b*, unsigned int *gvl*)
- *e8xm4\_t vmxor\_mm\_e8xm4* (*e8xm4\_t a, e8xm4\_t b*, unsigned int *gvl*)
- *e8xm8\_t vmxor\_mm\_e8xm8* (*e8xm8\_t a, e8xm8\_t b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
    result[element] = logical_xor (a[element], b[element])
result[gvl : VLMAX] = 0
```

## 2.12 Vector elements manipulation

### 2.12.1 Pack elements contiguously

**Instruction:** [`'vcompress.vmm'`]**Prototypes:**

- *int16xm1\_t vcompressvm\_int16xm1\_e16xm1* (*int16xm1\_t a, e16xm1\_t b*, unsigned int *gvl*)
- *int16xm2\_t vcompressvm\_int16xm2\_e16xm2* (*int16xm2\_t a, e16xm2\_t b*, unsigned int *gvl*)
- *int16xm4\_t vcompressvm\_int16xm4\_e16xm4* (*int16xm4\_t a, e16xm4\_t b*, unsigned int *gvl*)
- *int16xm8\_t vcompressvm\_int16xm8\_e16xm8* (*int16xm8\_t a, e16xm8\_t b*, unsigned int *gvl*)
- *int32xm1\_t vcompressvm\_int32xm1\_e32xm1* (*int32xm1\_t a, e32xm1\_t b*, unsigned int *gvl*)
- *int32xm2\_t vcompressvm\_int32xm2\_e32xm2* (*int32xm2\_t a, e32xm2\_t b*, unsigned int *gvl*)
- *int32xm4\_t vcompressvm\_int32xm4\_e32xm4* (*int32xm4\_t a, e32xm4\_t b*, unsigned int *gvl*)
- *int32xm8\_t vcompressvm\_int32xm8\_e32xm8* (*int32xm8\_t a, e32xm8\_t b*, unsigned int *gvl*)
- *int64xm1\_t vcompressvm\_int64xm1\_e64xm1* (*int64xm1\_t a, e64xm1\_t b*, unsigned int *gvl*)
- *int64xm2\_t vcompressvm\_int64xm2\_e64xm2* (*int64xm2\_t a, e64xm2\_t b*, unsigned int *gvl*)
- *int64xm4\_t vcompressvm\_int64xm4\_e64xm4* (*int64xm4\_t a, e64xm4\_t b*, unsigned int *gvl*)
- *int64xm8\_t vcompressvm\_int64xm8\_e64xm8* (*int64xm8\_t a, e64xm8\_t b*, unsigned int *gvl*)

- `int8xm1_t vcompressvm_int8xm1_e8xm1 (int8xm1_t a, e8xm1_t b, unsigned int gvl)`
- `int8xm2_t vcompressvm_int8xm2_e8xm2 (int8xm2_t a, e8xm2_t b, unsigned int gvl)`
- `int8xm4_t vcompressvm_int8xm4_e8xm4 (int8xm4_t a, e8xm4_t b, unsigned int gvl)`
- `int8xm8_t vcompressvm_int8xm8_e8xm8 (int8xm8_t a, e8xm8_t b, unsigned int gvl)`
- `uint16xm1_t vcompressvm_uint16xm1_e16xm1 (uint16xm1_t a, e16xm1_t b, unsigned int gvl)`
- `uint16xm2_t vcompressvm_uint16xm2_e16xm2 (uint16xm2_t a, e16xm2_t b, unsigned int gvl)`
- `uint16xm4_t vcompressvm_uint16xm4_e16xm4 (uint16xm4_t a, e16xm4_t b, unsigned int gvl)`
- `uint16xm8_t vcompressvm_uint16xm8_e16xm8 (uint16xm8_t a, e16xm8_t b, unsigned int gvl)`
- `uint32xm1_t vcompressvm_uint32xm1_e32xm1 (uint32xm1_t a, e32xm1_t b, unsigned int gvl)`
- `uint32xm2_t vcompressvm_uint32xm2_e32xm2 (uint32xm2_t a, e32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vcompressvm_uint32xm4_e32xm4 (uint32xm4_t a, e32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vcompressvm_uint32xm8_e32xm8 (uint32xm8_t a, e32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vcompressvm_uint64xm1_e64xm1 (uint64xm1_t a, e64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vcompressvm_uint64xm2_e64xm2 (uint64xm2_t a, e64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vcompressvm_uint64xm4_e64xm4 (uint64xm4_t a, e64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vcompressvm_uint64xm8_e64xm8 (uint64xm8_t a, e64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vcompressvm_uint8xm1_e8xm1 (uint8xm1_t a, e8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vcompressvm_uint8xm2_e8xm2 (uint8xm2_t a, e8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vcompressvm_uint8xm4_e8xm4 (uint8xm4_t a, e8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vcompressvm_uint8xm8_e8xm8 (uint8xm8_t a, e8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> next_index = 0
    for element = 0 to gvl - 1
        if b[element] then
            result[next_index] = a[element]
            next_index = next_index + 1
    result[next_index : VLMAX] = 0
```

## 2.12.2 Extract integer element

**Instruction:** [`'vext.x.v'`]**Prototypes:**

- short `vextxv_int16xm1 (int16xm1_t a, unsigned int b, unsigned int gvl)`
- short `vextxv_int16xm2 (int16xm2_t a, unsigned int b, unsigned int gvl)`
- short `vextxv_int16xm4 (int16xm4_t a, unsigned int b, unsigned int gvl)`
- short `vextxv_int16xm8 (int16xm8_t a, unsigned int b, unsigned int gvl)`
- int `vextxv_int32xm1 (int32xm1_t a, unsigned int b, unsigned int gvl)`
- int `vextxv_int32xm2 (int32xm2_t a, unsigned int b, unsigned int gvl)`

- int **vectxv\_int32xm4** (*int32xm4\_t a*, unsigned int *b*, unsigned int *gvl*)
- int **vectxv\_int32xm8** (*int32xm8\_t a*, unsigned int *b*, unsigned int *gvl*)
- long **vectxv\_int64xm1** (*int64xm1\_t a*, unsigned int *b*, unsigned int *gvl*)
- long **vectxv\_int64xm2** (*int64xm2\_t a*, unsigned int *b*, unsigned int *gvl*)
- long **vectxv\_int64xm4** (*int64xm4\_t a*, unsigned int *b*, unsigned int *gvl*)
- long **vectxv\_int64xm8** (*int64xm8\_t a*, unsigned int *b*, unsigned int *gvl*)
- signed char **vectxv\_int8xm1** (*int8xm1\_t a*, unsigned int *b*, unsigned int *gvl*)
- signed char **vectxv\_int8xm2** (*int8xm2\_t a*, unsigned int *b*, unsigned int *gvl*)
- signed char **vectxv\_int8xm4** (*int8xm4\_t a*, unsigned int *b*, unsigned int *gvl*)
- signed char **vectxv\_int8xm8** (*int8xm8\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned short **vectxv\_uint16xm1** (*uint16xm1\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned short **vectxv\_uint16xm2** (*uint16xm2\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned short **vectxv\_uint16xm4** (*uint16xm4\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned short **vectxv\_uint16xm8** (*uint16xm8\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned int **vectxv\_uint32xm1** (*uint32xm1\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned int **vectxv\_uint32xm2** (*uint32xm2\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned int **vectxv\_uint32xm4** (*uint32xm4\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned int **vectxv\_uint32xm8** (*uint32xm8\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned long **vectxv\_uint64xm1** (*uint64xm1\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned long **vectxv\_uint64xm2** (*uint64xm2\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned long **vectxv\_uint64xm4** (*uint64xm4\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned long **vectxv\_uint64xm8** (*uint64xm8\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned char **vectxv\_uint8xm1** (*uint8xm1\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned char **vectxv\_uint8xm2** (*uint8xm2\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned char **vectxv\_uint8xm4** (*uint8xm4\_t a*, unsigned int *b*, unsigned int *gvl*)
- unsigned char **vectxv\_uint8xm8** (*uint8xm8\_t a*, unsigned int *b*, unsigned int *gvl*)

**Operation:**

```
>>> if b < VLEN/SEW
      result = a[b]
else
      result = 0
```

**2.12.3 Merge two floating-point vectors using a mask vector****Instruction:** [`'vfmerge.vfm'`]**Masked prototypes:**

- *float16xm1\_t* **vfmergevfm\_mask\_float16xm1** (*float16xm1\_t a*, float16\_t *b*, *e16xm1\_t mask*, unsigned int *gvl*)

- `float16xm2_t vfmergevfm_mask_float16xm2` (`float16xm2_t a`, `float16_t b`, `e16xm2_t mask`, unsigned int `gvl`)
- `float16xm4_t vfmergevfm_mask_float16xm4` (`float16xm4_t a`, `float16_t b`, `e16xm4_t mask`, unsigned int `gvl`)
- `float16xm8_t vfmergevfm_mask_float16xm8` (`float16xm8_t a`, `float16_t b`, `e16xm8_t mask`, unsigned int `gvl`)
- `float32xm1_t vfmergevfm_mask_float32xm1` (`float32xm1_t a`, `float b`, `e32xm1_t mask`, unsigned int `gvl`)
- `float32xm2_t vfmergevfm_mask_float32xm2` (`float32xm2_t a`, `float b`, `e32xm2_t mask`, unsigned int `gvl`)
- `float32xm4_t vfmergevfm_mask_float32xm4` (`float32xm4_t a`, `float b`, `e32xm4_t mask`, unsigned int `gvl`)
- `float32xm8_t vfmergevfm_mask_float32xm8` (`float32xm8_t a`, `float b`, `e32xm8_t mask`, unsigned int `gvl`)
- `float64xm1_t vfmergevfm_mask_float64xm1` (`float64xm1_t a`, `double b`, `e64xm1_t mask`, unsigned int `gvl`)
- `float64xm2_t vfmergevfm_mask_float64xm2` (`float64xm2_t a`, `double b`, `e64xm2_t mask`, unsigned int `gvl`)
- `float64xm4_t vfmergevfm_mask_float64xm4` (`float64xm4_t a`, `double b`, `e64xm4_t mask`, unsigned int `gvl`)
- `float64xm8_t vfmergevfm_mask_float64xm8` (`float64xm8_t a`, `double b`, `e64xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = 0 to gvl - 1
      result[element] = mask[element] ? b : a[element]
result[gvl : VLMAX] = 0
```

## 2.12.4 move the lowest element of a vector to the given floating-point

Instruction: [`'vfmv.f.s'`]

Prototypes:

- `float16_t vfmvfs_float16xm1` (`float16xm1_t a`, unsigned int `gvl`)
- `float16_t vfmvfs_float16xm2` (`float16xm2_t a`, unsigned int `gvl`)
- `float16_t vfmvfs_float16xm4` (`float16xm4_t a`, unsigned int `gvl`)
- `float16_t vfmvfs_float16xm8` (`float16xm8_t a`, unsigned int `gvl`)
- `float vfmvfs_float32xm1` (`float32xm1_t a`, unsigned int `gvl`)
- `float vfmvfs_float32xm2` (`float32xm2_t a`, unsigned int `gvl`)
- `float vfmvfs_float32xm4` (`float32xm4_t a`, unsigned int `gvl`)
- `float vfmvfs_float32xm8` (`float32xm8_t a`, unsigned int `gvl`)
- `double vfmvfs_float64xm1` (`float64xm1_t a`, unsigned int `gvl`)
- `double vfmvfs_float64xm2` (`float64xm2_t a`, unsigned int `gvl`)
- `double vfmvfs_float64xm4` (`float64xm4_t a`, unsigned int `gvl`)

- double **vfmvfs\_float64xm8** (*float64xm8\_t a*, unsigned int *gvl*)

**Operation:**

```
>>> result = a[0]
```

## 2.12.5 move a floating-point to the lowest element of the vector

**Instruction:** [`'vfmv.s.f'`]

**Prototypes:**

- *float16xm1\_t* **vfmvsv\_float16xm1** (*float16\_t a*, unsigned int *gvl*)
- *float16xm2\_t* **vfmvsv\_float16xm2** (*float16\_t a*, unsigned int *gvl*)
- *float16xm4\_t* **vfmvsv\_float16xm4** (*float16\_t a*, unsigned int *gvl*)
- *float16xm8\_t* **vfmvsv\_float16xm8** (*float16\_t a*, unsigned int *gvl*)
- *float32xm1\_t* **vfmvsv\_float32xm1** (*float a*, unsigned int *gvl*)
- *float32xm2\_t* **vfmvsv\_float32xm2** (*float a*, unsigned int *gvl*)
- *float32xm4\_t* **vfmvsv\_float32xm4** (*float a*, unsigned int *gvl*)
- *float32xm8\_t* **vfmvsv\_float32xm8** (*float a*, unsigned int *gvl*)
- *float64xm1\_t* **vfmvsv\_float64xm1** (*double a*, unsigned int *gvl*)
- *float64xm2\_t* **vfmvsv\_float64xm2** (*double a*, unsigned int *gvl*)
- *float64xm4\_t* **vfmvsv\_float64xm4** (*double a*, unsigned int *gvl*)
- *float64xm8\_t* **vfmvsv\_float64xm8** (*double a*, unsigned int *gvl*)

**Operation:**

```
>>> result[0] = a
```

## 2.12.6 Create a vector that all the elements the same as the given floating-point

**Instruction:** [`'vfmv.v.f'`]

**Prototypes:**

- *float16xm1\_t* **vfmvvf\_float16xm1** (*float16\_t a*, unsigned int *gvl*)
- *float16xm2\_t* **vfmvvf\_float16xm2** (*float16\_t a*, unsigned int *gvl*)
- *float16xm4\_t* **vfmvvf\_float16xm4** (*float16\_t a*, unsigned int *gvl*)
- *float16xm8\_t* **vfmvvf\_float16xm8** (*float16\_t a*, unsigned int *gvl*)
- *float32xm1\_t* **vfmvvf\_float32xm1** (*float a*, unsigned int *gvl*)
- *float32xm2\_t* **vfmvvf\_float32xm2** (*float a*, unsigned int *gvl*)
- *float32xm4\_t* **vfmvvf\_float32xm4** (*float a*, unsigned int *gvl*)
- *float32xm8\_t* **vfmvvf\_float32xm8** (*float a*, unsigned int *gvl*)
- *float64xm1\_t* **vfmvvf\_float64xm1** (*double a*, unsigned int *gvl*)
- *float64xm2\_t* **vfmvvf\_float64xm2** (*double a*, unsigned int *gvl*)

- `float64xm4_t vfmvuf_float64xm4` (double *a*, unsigned int *gvl*)
- `float64xm8_t vfmvuf_float64xm8` (double *a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a
      result[gvl : VLMAX] = 0
```

## 2.12.7 Compute index vector

**Instruction:** [`'vid.v'`]**Prototypes:**

- `uint16xm1_t vidv_uint16xm1` (unsigned int *gvl*)
- `uint16xm2_t vidv_uint16xm2` (unsigned int *gvl*)
- `uint16xm4_t vidv_uint16xm4` (unsigned int *gvl*)
- `uint16xm8_t vidv_uint16xm8` (unsigned int *gvl*)
- `uint32xm1_t vidv_uint32xm1` (unsigned int *gvl*)
- `uint32xm2_t vidv_uint32xm2` (unsigned int *gvl*)
- `uint32xm4_t vidv_uint32xm4` (unsigned int *gvl*)
- `uint32xm8_t vidv_uint32xm8` (unsigned int *gvl*)
- `uint64xm1_t vidv_uint64xm1` (unsigned int *gvl*)
- `uint64xm2_t vidv_uint64xm2` (unsigned int *gvl*)
- `uint64xm4_t vidv_uint64xm4` (unsigned int *gvl*)
- `uint64xm8_t vidv_uint64xm8` (unsigned int *gvl*)
- `uint8xm1_t vidv_uint8xm1` (unsigned int *gvl*)
- `uint8xm2_t vidv_uint8xm2` (unsigned int *gvl*)
- `uint8xm4_t vidv_uint8xm4` (unsigned int *gvl*)
- `uint8xm8_t vidv_uint8xm8` (unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = element
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t vidv_mask_uint16xm1` (`uint16xm1_t merge`, `e16xm1_t mask`, unsigned int *gvl*)
- `uint16xm2_t vidv_mask_uint16xm2` (`uint16xm2_t merge`, `e16xm2_t mask`, unsigned int *gvl*)
- `uint16xm4_t vidv_mask_uint16xm4` (`uint16xm4_t merge`, `e16xm4_t mask`, unsigned int *gvl*)
- `uint16xm8_t vidv_mask_uint16xm8` (`uint16xm8_t merge`, `e16xm8_t mask`, unsigned int *gvl*)
- `uint32xm1_t vidv_mask_uint32xm1` (`uint32xm1_t merge`, `e32xm1_t mask`, unsigned int *gvl*)



- `uint32xm2_t vidv_mask_uint32xm2 (uint32xm2_t merge, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vidv_mask_uint32xm4 (uint32xm4_t merge, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vidv_mask_uint32xm8 (uint32xm8_t merge, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vidv_mask_uint64xm1 (uint64xm1_t merge, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vidv_mask_uint64xm2 (uint64xm2_t merge, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vidv_mask_uint64xm4 (uint64xm4_t merge, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vidv_mask_uint64xm8 (uint64xm8_t merge, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vidv_mask_uint8xm1 (uint8xm1_t merge, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vidv_mask_uint8xm2 (uint8xm2_t merge, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vidv_mask_uint8xm4 (uint8xm4_t merge, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vidv_mask_uint8xm8 (uint8xm8_t merge, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = element
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.12.8 Compute a prefix sum of a mask

**Instruction:** [`'viota.m'`]

**Prototypes:**

- `uint16xm1_t viotam_uint16xm1_e16xm1 (e16xm1_t a, unsigned int gvl)`
- `uint16xm2_t viotam_uint16xm2_e16xm2 (e16xm2_t a, unsigned int gvl)`
- `uint16xm4_t viotam_uint16xm4_e16xm4 (e16xm4_t a, unsigned int gvl)`
- `uint16xm8_t viotam_uint16xm8_e16xm8 (e16xm8_t a, unsigned int gvl)`
- `uint32xm1_t viotam_uint32xm1_e32xm1 (e32xm1_t a, unsigned int gvl)`
- `uint32xm2_t viotam_uint32xm2_e32xm2 (e32xm2_t a, unsigned int gvl)`
- `uint32xm4_t viotam_uint32xm4_e32xm4 (e32xm4_t a, unsigned int gvl)`
- `uint32xm8_t viotam_uint32xm8_e32xm8 (e32xm8_t a, unsigned int gvl)`
- `uint64xm1_t viotam_uint64xm1_e64xm1 (e64xm1_t a, unsigned int gvl)`
- `uint64xm2_t viotam_uint64xm2_e64xm2 (e64xm2_t a, unsigned int gvl)`
- `uint64xm4_t viotam_uint64xm4_e64xm4 (e64xm4_t a, unsigned int gvl)`
- `uint64xm8_t viotam_uint64xm8_e64xm8 (e64xm8_t a, unsigned int gvl)`
- `uint8xm1_t viotam_uint8xm1_e8xm1 (e8xm1_t a, unsigned int gvl)`
- `uint8xm2_t viotam_uint8xm2_e8xm2 (e8xm2_t a, unsigned int gvl)`
- `uint8xm4_t viotam_uint8xm4_e8xm4 (e8xm4_t a, unsigned int gvl)`

- `uint8xm8_t viotam_uint8xm8_e8xm8 (e8xm8_t a, unsigned int gvl)`

**Operation:**

```
>>> prefix_sum = 0
    for element = 0 to gvl - 1
        result[element] = prefix_sum
        if a[element] then
            prefix_sum = prefix_sum + 1
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `uint16xm1_t viotam_mask_uint16xm1_e16xm1 (uint16xm1_t merge, e16xm1_t a, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t viotam_mask_uint16xm2_e16xm2 (uint16xm2_t merge, e16xm2_t a, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t viotam_mask_uint16xm4_e16xm4 (uint16xm4_t merge, e16xm4_t a, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t viotam_mask_uint16xm8_e16xm8 (uint16xm8_t merge, e16xm8_t a, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t viotam_mask_uint32xm1_e32xm1 (uint32xm1_t merge, e32xm1_t a, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t viotam_mask_uint32xm2_e32xm2 (uint32xm2_t merge, e32xm2_t a, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t viotam_mask_uint32xm4_e32xm4 (uint32xm4_t merge, e32xm4_t a, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t viotam_mask_uint32xm8_e32xm8 (uint32xm8_t merge, e32xm8_t a, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t viotam_mask_uint64xm1_e64xm1 (uint64xm1_t merge, e64xm1_t a, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t viotam_mask_uint64xm2_e64xm2 (uint64xm2_t merge, e64xm2_t a, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t viotam_mask_uint64xm4_e64xm4 (uint64xm4_t merge, e64xm4_t a, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t viotam_mask_uint64xm8_e64xm8 (uint64xm8_t merge, e64xm8_t a, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t viotam_mask_uint8xm1_e8xm1 (uint8xm1_t merge, e8xm1_t a, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t viotam_mask_uint8xm2_e8xm2 (uint8xm2_t merge, e8xm2_t a, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t viotam_mask_uint8xm4_e8xm4 (uint8xm4_t merge, e8xm4_t a, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t viotam_mask_uint8xm8_e8xm8 (uint8xm8_t merge, e8xm8_t a, e8xm8_t mask, unsigned int gvl)`

**Masked operation:**

```
>>> prefix_sum = 0
    for element = 0 to gvl - 1
        if mask[element] then
```

(continues on next page)

(continued from previous page)

```

result[element] = prefix_sum
if a[element] then
    prefix_sum = prefix_sum + 1
else
    result[element] = merge[element]
result[gvl : VLMAX] = 0

```

## 2.12.9 Elementwise vector-immediate integer merge

**Instruction:** ['vmerge.vim']

**Masked prototypes:**

- *int16xm1\_t* vmergevim\_mask\_int16xm1 (*int16xm1\_t* a, const short b, *e16xm1\_t* mask, unsigned int gvl)
- *int16xm2\_t* vmergevim\_mask\_int16xm2 (*int16xm2\_t* a, const short b, *e16xm2\_t* mask, unsigned int gvl)
- *int16xm4\_t* vmergevim\_mask\_int16xm4 (*int16xm4\_t* a, const short b, *e16xm4\_t* mask, unsigned int gvl)
- *int16xm8\_t* vmergevim\_mask\_int16xm8 (*int16xm8\_t* a, const short b, *e16xm8\_t* mask, unsigned int gvl)
- *int32xm1\_t* vmergevim\_mask\_int32xm1 (*int32xm1\_t* a, const int b, *e32xm1\_t* mask, unsigned int gvl)
- *int32xm2\_t* vmergevim\_mask\_int32xm2 (*int32xm2\_t* a, const int b, *e32xm2\_t* mask, unsigned int gvl)
- *int32xm4\_t* vmergevim\_mask\_int32xm4 (*int32xm4\_t* a, const int b, *e32xm4\_t* mask, unsigned int gvl)
- *int32xm8\_t* vmergevim\_mask\_int32xm8 (*int32xm8\_t* a, const int b, *e32xm8\_t* mask, unsigned int gvl)
- *int64xm1\_t* vmergevim\_mask\_int64xm1 (*int64xm1\_t* a, const long b, *e64xm1\_t* mask, unsigned int gvl)
- *int64xm2\_t* vmergevim\_mask\_int64xm2 (*int64xm2\_t* a, const long b, *e64xm2\_t* mask, unsigned int gvl)
- *int64xm4\_t* vmergevim\_mask\_int64xm4 (*int64xm4\_t* a, const long b, *e64xm4\_t* mask, unsigned int gvl)
- *int64xm8\_t* vmergevim\_mask\_int64xm8 (*int64xm8\_t* a, const long b, *e64xm8\_t* mask, unsigned int gvl)
- *int8xm1\_t* vmergevim\_mask\_int8xm1 (*int8xm1\_t* a, const signed char b, *e8xm1\_t* mask, unsigned int gvl)
- *int8xm2\_t* vmergevim\_mask\_int8xm2 (*int8xm2\_t* a, const signed char b, *e8xm2\_t* mask, unsigned int gvl)
- *int8xm4\_t* vmergevim\_mask\_int8xm4 (*int8xm4\_t* a, const signed char b, *e8xm4\_t* mask, unsigned int gvl)
- *int8xm8\_t* vmergevim\_mask\_int8xm8 (*int8xm8\_t* a, const signed char b, *e8xm8\_t* mask, unsigned int gvl)
- *uint16xm1\_t* vmergevim\_mask\_uint16xm1 (*uint16xm1\_t* a, const unsigned short b, *e16xm1\_t* mask, unsigned int gvl)

- `uint16xm2_t vmergevim_mask_uint16xm2` (`uint16xm2_t a`, const unsigned short `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint16xm4_t vmergevim_mask_uint16xm4` (`uint16xm4_t a`, const unsigned short `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vmergevim_mask_uint16xm8` (`uint16xm8_t a`, const unsigned short `b`, `e16xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vmergevim_mask_uint32xm1` (`uint32xm1_t a`, const unsigned int `b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vmergevim_mask_uint32xm2` (`uint32xm2_t a`, const unsigned int `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vmergevim_mask_uint32xm4` (`uint32xm4_t a`, const unsigned int `b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint32xm8_t vmergevim_mask_uint32xm8` (`uint32xm8_t a`, const unsigned int `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint64xm1_t vmergevim_mask_uint64xm1` (`uint64xm1_t a`, const unsigned long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vmergevim_mask_uint64xm2` (`uint64xm2_t a`, const unsigned long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vmergevim_mask_uint64xm4` (`uint64xm4_t a`, const unsigned long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vmergevim_mask_uint64xm8` (`uint64xm8_t a`, const unsigned long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vmergevim_mask_uint8xm1` (`uint8xm1_t a`, const unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vmergevim_mask_uint8xm2` (`uint8xm2_t a`, const unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vmergevim_mask_uint8xm4` (`uint8xm4_t a`, const unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vmergevim_mask_uint8xm8` (`uint8xm8_t a`, const unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = b
      else
        result[element] = a[element]
      result[gvl : VLMAX] = 0
```

### 2.12.10 Elementwise vector-vector integer merge

**Instruction:** [`'vmergevvm'`]

**Masked prototypes:**

- `int16xm1_t vmergevvm_mask_int16xm1` (`int16xm1_t a`, `int16xm1_t b`, `e16xm1_t mask`, unsigned int `gvl`)
- `int16xm2_t vmergevvm_mask_int16xm2` (`int16xm2_t a`, `int16xm2_t b`, `e16xm2_t mask`, unsigned int `gvl`)

- `int16xm4_t vmergevmm_mask_int16xm4` (`int16xm4_t a`, `int16xm4_t b`, `e16xm4_t mask`, unsigned int gvl)
- `int16xm8_t vmergevmm_mask_int16xm8` (`int16xm8_t a`, `int16xm8_t b`, `e16xm8_t mask`, unsigned int gvl)
- `int32xm1_t vmergevmm_mask_int32xm1` (`int32xm1_t a`, `int32xm1_t b`, `e32xm1_t mask`, unsigned int gvl)
- `int32xm2_t vmergevmm_mask_int32xm2` (`int32xm2_t a`, `int32xm2_t b`, `e32xm2_t mask`, unsigned int gvl)
- `int32xm4_t vmergevmm_mask_int32xm4` (`int32xm4_t a`, `int32xm4_t b`, `e32xm4_t mask`, unsigned int gvl)
- `int32xm8_t vmergevmm_mask_int32xm8` (`int32xm8_t a`, `int32xm8_t b`, `e32xm8_t mask`, unsigned int gvl)
- `int64xm1_t vmergevmm_mask_int64xm1` (`int64xm1_t a`, `int64xm1_t b`, `e64xm1_t mask`, unsigned int gvl)
- `int64xm2_t vmergevmm_mask_int64xm2` (`int64xm2_t a`, `int64xm2_t b`, `e64xm2_t mask`, unsigned int gvl)
- `int64xm4_t vmergevmm_mask_int64xm4` (`int64xm4_t a`, `int64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)
- `int64xm8_t vmergevmm_mask_int64xm8` (`int64xm8_t a`, `int64xm8_t b`, `e64xm8_t mask`, unsigned int gvl)
- `int8xm1_t vmergevmm_mask_int8xm1` (`int8xm1_t a`, `int8xm1_t b`, `e8xm1_t mask`, unsigned int gvl)
- `int8xm2_t vmergevmm_mask_int8xm2` (`int8xm2_t a`, `int8xm2_t b`, `e8xm2_t mask`, unsigned int gvl)
- `int8xm4_t vmergevmm_mask_int8xm4` (`int8xm4_t a`, `int8xm4_t b`, `e8xm4_t mask`, unsigned int gvl)
- `int8xm8_t vmergevmm_mask_int8xm8` (`int8xm8_t a`, `int8xm8_t b`, `e8xm8_t mask`, unsigned int gvl)
- `uint16xm1_t vmergevmm_mask_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, `e16xm1_t mask`, unsigned int gvl)
- `uint16xm2_t vmergevmm_mask_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, `e16xm2_t mask`, unsigned int gvl)
- `uint16xm4_t vmergevmm_mask_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, `e16xm4_t mask`, unsigned int gvl)
- `uint16xm8_t vmergevmm_mask_uint16xm8` (`uint16xm8_t a`, `uint16xm8_t b`, `e16xm8_t mask`, unsigned int gvl)
- `uint32xm1_t vmergevmm_mask_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, `e32xm1_t mask`, unsigned int gvl)
- `uint32xm2_t vmergevmm_mask_uint32xm2` (`uint32xm2_t a`, `uint32xm2_t b`, `e32xm2_t mask`, unsigned int gvl)
- `uint32xm4_t vmergevmm_mask_uint32xm4` (`uint32xm4_t a`, `uint32xm4_t b`, `e32xm4_t mask`, unsigned int gvl)
- `uint32xm8_t vmergevmm_mask_uint32xm8` (`uint32xm8_t a`, `uint32xm8_t b`, `e32xm8_t mask`, unsigned int gvl)
- `uint64xm1_t vmergevmm_mask_uint64xm1` (`uint64xm1_t a`, `uint64xm1_t b`, `e64xm1_t mask`, unsigned int gvl)
- `uint64xm2_t vmergevmm_mask_uint64xm2` (`uint64xm2_t a`, `uint64xm2_t b`, `e64xm2_t mask`, unsigned int gvl)
- `uint64xm4_t vmergevmm_mask_uint64xm4` (`uint64xm4_t a`, `uint64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)

- `uint64xm8_t vmergevmm_mask_uint64xm8 (uint64xm8_t a, uint64xm8_t b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vmergevmm_mask_uint8xm1 (uint8xm1_t a, uint8xm1_t b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vmergevmm_mask_uint8xm2 (uint8xm2_t a, uint8xm2_t b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vmergevmm_mask_uint8xm4 (uint8xm4_t a, uint8xm4_t b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vmergevmm_mask_uint8xm8 (uint8xm8_t a, uint8xm8_t b, e8xm8_t mask, unsigned int gvl)`

Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = b[element]
      else
        result[element] = a[element]
      result[gvl : VLMAX] = 0
```

### 2.12.11 Elementwise vector-scalar integer merge

Instruction: ['vmerge.vxm']

Masked prototypes:

- `int16xm1_t vmergevmm_mask_int16xm1 (int16xm1_t a, short b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vmergevmm_mask_int16xm2 (int16xm2_t a, short b, e16xm2_t mask, unsigned int gvl)`
- `int16xm4_t vmergevmm_mask_int16xm4 (int16xm4_t a, short b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vmergevmm_mask_int16xm8 (int16xm8_t a, short b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vmergevmm_mask_int32xm1 (int32xm1_t a, int b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vmergevmm_mask_int32xm2 (int32xm2_t a, int b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vmergevmm_mask_int32xm4 (int32xm4_t a, int b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vmergevmm_mask_int32xm8 (int32xm8_t a, int b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vmergevmm_mask_int64xm1 (int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vmergevmm_mask_int64xm2 (int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vmergevmm_mask_int64xm4 (int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vmergevmm_mask_int64xm8 (int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vmergevmm_mask_int8xm1 (int8xm1_t a, signed char b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vmergevmm_mask_int8xm2 (int8xm2_t a, signed char b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vmergevmm_mask_int8xm4 (int8xm4_t a, signed char b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vmergevmm_mask_int8xm8 (int8xm8_t a, signed char b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vmergevmm_mask_uint16xm1 (uint16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vmergevmm_mask_uint16xm2 (uint16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`



- `uint16xm4_t vmergevxm_mask_uint16xm4` (`uint16xm4_t a`, unsigned short `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vmergevxm_mask_uint16xm8` (`uint16xm8_t a`, unsigned short `b`, `e16xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vmergevxm_mask_uint32xm1` (`uint32xm1_t a`, unsigned int `b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vmergevxm_mask_uint32xm2` (`uint32xm2_t a`, unsigned int `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vmergevxm_mask_uint32xm4` (`uint32xm4_t a`, unsigned int `b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint32xm8_t vmergevxm_mask_uint32xm8` (`uint32xm8_t a`, unsigned int `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint64xm1_t vmergevxm_mask_uint64xm1` (`uint64xm1_t a`, unsigned long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vmergevxm_mask_uint64xm2` (`uint64xm2_t a`, unsigned long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vmergevxm_mask_uint64xm4` (`uint64xm4_t a`, unsigned long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vmergevxm_mask_uint64xm8` (`uint64xm8_t a`, unsigned long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vmergevxm_mask_uint8xm1` (`uint8xm1_t a`, unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vmergevxm_mask_uint8xm2` (`uint8xm2_t a`, unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vmergevxm_mask_uint8xm4` (`uint8xm4_t a`, unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vmergevxm_mask_uint8xm8` (`uint8xm8_t a`, unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        result[element] = b
      else
        result[element] = a[element]
      result[gvl : VLMAX] = 0
```

### 2.12.12 Set first integer element of integer vector

**Instruction:** [`'vmv.s.x'`]

#### Prototypes:

- `int16xm1_t vmvsx_int16xm1` (short `a`, unsigned int `gvl`)
- `int16xm2_t vmvsx_int16xm2` (short `a`, unsigned int `gvl`)
- `int16xm4_t vmvsx_int16xm4` (short `a`, unsigned int `gvl`)
- `int16xm8_t vmvsx_int16xm8` (short `a`, unsigned int `gvl`)
- `int32xm1_t vmvsx_int32xm1` (int `a`, unsigned int `gvl`)

- `int32xm2_t vmvsx_int32xm2` (int *a*, unsigned int *gvl*)
- `int32xm4_t vmvsx_int32xm4` (int *a*, unsigned int *gvl*)
- `int32xm8_t vmvsx_int32xm8` (int *a*, unsigned int *gvl*)
- `int64xm1_t vmvsx_int64xm1` (long *a*, unsigned int *gvl*)
- `int64xm2_t vmvsx_int64xm2` (long *a*, unsigned int *gvl*)
- `int64xm4_t vmvsx_int64xm4` (long *a*, unsigned int *gvl*)
- `int64xm8_t vmvsx_int64xm8` (long *a*, unsigned int *gvl*)
- `int8xm1_t vmvsx_int8xm1` (signed char *a*, unsigned int *gvl*)
- `int8xm2_t vmvsx_int8xm2` (signed char *a*, unsigned int *gvl*)
- `int8xm4_t vmvsx_int8xm4` (signed char *a*, unsigned int *gvl*)
- `int8xm8_t vmvsx_int8xm8` (signed char *a*, unsigned int *gvl*)
- `uint16xm1_t vmvsx_uint16xm1` (unsigned short *a*, unsigned int *gvl*)
- `uint16xm2_t vmvsx_uint16xm2` (unsigned short *a*, unsigned int *gvl*)
- `uint16xm4_t vmvsx_uint16xm4` (unsigned short *a*, unsigned int *gvl*)
- `uint16xm8_t vmvsx_uint16xm8` (unsigned short *a*, unsigned int *gvl*)
- `uint32xm1_t vmvsx_uint32xm1` (unsigned int *a*, unsigned int *gvl*)
- `uint32xm2_t vmvsx_uint32xm2` (unsigned int *a*, unsigned int *gvl*)
- `uint32xm4_t vmvsx_uint32xm4` (unsigned int *a*, unsigned int *gvl*)
- `uint32xm8_t vmvsx_uint32xm8` (unsigned int *a*, unsigned int *gvl*)
- `uint64xm1_t vmvsx_uint64xm1` (unsigned long *a*, unsigned int *gvl*)
- `uint64xm2_t vmvsx_uint64xm2` (unsigned long *a*, unsigned int *gvl*)
- `uint64xm4_t vmvsx_uint64xm4` (unsigned long *a*, unsigned int *gvl*)
- `uint64xm8_t vmvsx_uint64xm8` (unsigned long *a*, unsigned int *gvl*)
- `uint8xm1_t vmvsx_uint8xm1` (unsigned char *a*, unsigned int *gvl*)
- `uint8xm2_t vmvsx_uint8xm2` (unsigned char *a*, unsigned int *gvl*)
- `uint8xm4_t vmvsx_uint8xm4` (unsigned char *a*, unsigned int *gvl*)
- `uint8xm8_t vmvsx_uint8xm8` (unsigned char *a*, unsigned int *gvl*)

Operation:

```
>>> result[0] = a
```

### 2.12.13 Elementwise immediate integer move

Instruction: [`'vmv.v.i'`]

Prototypes:

- `int16xm1_t vmvvi_int16xm1` (const short *a*, unsigned int *gvl*)
- `int16xm2_t vmvvi_int16xm2` (const short *a*, unsigned int *gvl*)

- `int16xm4_t vmvvi_int16xm4` (const short *a*, unsigned int *gvl*)
- `int16xm8_t vmvvi_int16xm8` (const short *a*, unsigned int *gvl*)
- `int32xm1_t vmvvi_int32xm1` (const int *a*, unsigned int *gvl*)
- `int32xm2_t vmvvi_int32xm2` (const int *a*, unsigned int *gvl*)
- `int32xm4_t vmvvi_int32xm4` (const int *a*, unsigned int *gvl*)
- `int32xm8_t vmvvi_int32xm8` (const int *a*, unsigned int *gvl*)
- `int64xm1_t vmvvi_int64xm1` (const long *a*, unsigned int *gvl*)
- `int64xm2_t vmvvi_int64xm2` (const long *a*, unsigned int *gvl*)
- `int64xm4_t vmvvi_int64xm4` (const long *a*, unsigned int *gvl*)
- `int64xm8_t vmvvi_int64xm8` (const long *a*, unsigned int *gvl*)
- `int8xm1_t vmvvi_int8xm1` (const signed char *a*, unsigned int *gvl*)
- `int8xm2_t vmvvi_int8xm2` (const signed char *a*, unsigned int *gvl*)
- `int8xm4_t vmvvi_int8xm4` (const signed char *a*, unsigned int *gvl*)
- `int8xm8_t vmvvi_int8xm8` (const signed char *a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a
      result[gvl : VLMAX] = 0
```

## 2.12.14 Elementwise vector integer move

**Instruction:** [`'vmv.v.v'`]**Prototypes:**

- `float16xm1_t vmvvv_float16xm1` (`float16xm1_t` *a*, unsigned int *gvl*)
- `float16xm2_t vmvvv_float16xm2` (`float16xm2_t` *a*, unsigned int *gvl*)
- `float16xm4_t vmvvv_float16xm4` (`float16xm4_t` *a*, unsigned int *gvl*)
- `float16xm8_t vmvvv_float16xm8` (`float16xm8_t` *a*, unsigned int *gvl*)
- `float32xm1_t vmvvv_float32xm1` (`float32xm1_t` *a*, unsigned int *gvl*)
- `float32xm2_t vmvvv_float32xm2` (`float32xm2_t` *a*, unsigned int *gvl*)
- `float32xm4_t vmvvv_float32xm4` (`float32xm4_t` *a*, unsigned int *gvl*)
- `float32xm8_t vmvvv_float32xm8` (`float32xm8_t` *a*, unsigned int *gvl*)
- `float64xm1_t vmvvv_float64xm1` (`float64xm1_t` *a*, unsigned int *gvl*)
- `float64xm2_t vmvvv_float64xm2` (`float64xm2_t` *a*, unsigned int *gvl*)
- `float64xm4_t vmvvv_float64xm4` (`float64xm4_t` *a*, unsigned int *gvl*)
- `float64xm8_t vmvvv_float64xm8` (`float64xm8_t` *a*, unsigned int *gvl*)
- `int16xm1_t vmvvv_int16xm1` (`int16xm1_t` *a*, unsigned int *gvl*)
- `int16xm2_t vmvvv_int16xm2` (`int16xm2_t` *a*, unsigned int *gvl*)

- `int16xm4_t vmvvv_int16xm4 (int16xm4_t a, unsigned int gvl)`
- `int16xm8_t vmvvv_int16xm8 (int16xm8_t a, unsigned int gvl)`
- `int32xm1_t vmvvv_int32xm1 (int32xm1_t a, unsigned int gvl)`
- `int32xm2_t vmvvv_int32xm2 (int32xm2_t a, unsigned int gvl)`
- `int32xm4_t vmvvv_int32xm4 (int32xm4_t a, unsigned int gvl)`
- `int32xm8_t vmvvv_int32xm8 (int32xm8_t a, unsigned int gvl)`
- `int64xm1_t vmvvv_int64xm1 (int64xm1_t a, unsigned int gvl)`
- `int64xm2_t vmvvv_int64xm2 (int64xm2_t a, unsigned int gvl)`
- `int64xm4_t vmvvv_int64xm4 (int64xm4_t a, unsigned int gvl)`
- `int64xm8_t vmvvv_int64xm8 (int64xm8_t a, unsigned int gvl)`
- `int8xm1_t vmvvv_int8xm1 (int8xm1_t a, unsigned int gvl)`
- `int8xm2_t vmvvv_int8xm2 (int8xm2_t a, unsigned int gvl)`
- `int8xm4_t vmvvv_int8xm4 (int8xm4_t a, unsigned int gvl)`
- `int8xm8_t vmvvv_int8xm8 (int8xm8_t a, unsigned int gvl)`
- `uint16xm1_t vmvvv_uint16xm1 (uint16xm1_t a, unsigned int gvl)`
- `uint16xm2_t vmvvv_uint16xm2 (uint16xm2_t a, unsigned int gvl)`
- `uint16xm4_t vmvvv_uint16xm4 (uint16xm4_t a, unsigned int gvl)`
- `uint16xm8_t vmvvv_uint16xm8 (uint16xm8_t a, unsigned int gvl)`
- `uint32xm1_t vmvvv_uint32xm1 (uint32xm1_t a, unsigned int gvl)`
- `uint32xm2_t vmvvv_uint32xm2 (uint32xm2_t a, unsigned int gvl)`
- `uint32xm4_t vmvvv_uint32xm4 (uint32xm4_t a, unsigned int gvl)`
- `uint32xm8_t vmvvv_uint32xm8 (uint32xm8_t a, unsigned int gvl)`
- `uint64xm1_t vmvvv_uint64xm1 (uint64xm1_t a, unsigned int gvl)`
- `uint64xm2_t vmvvv_uint64xm2 (uint64xm2_t a, unsigned int gvl)`
- `uint64xm4_t vmvvv_uint64xm4 (uint64xm4_t a, unsigned int gvl)`
- `uint64xm8_t vmvvv_uint64xm8 (uint64xm8_t a, unsigned int gvl)`
- `uint8xm1_t vmvvv_uint8xm1 (uint8xm1_t a, unsigned int gvl)`
- `uint8xm2_t vmvvv_uint8xm2 (uint8xm2_t a, unsigned int gvl)`
- `uint8xm4_t vmvvv_uint8xm4 (uint8xm4_t a, unsigned int gvl)`
- `uint8xm8_t vmvvv_uint8xm8 (uint8xm8_t a, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a[element]
result[gvl : VLMAX] = 0
```

## 2.12.15 Elementwise scalar integer move

**Instruction:** [`'vmv.v.x'`]

**Prototypes:**

- `int16xm1_t vmvvx_int16xm1` (short *a*, unsigned int *gvl*)
- `int16xm2_t vmvvx_int16xm2` (short *a*, unsigned int *gvl*)
- `int16xm4_t vmvvx_int16xm4` (short *a*, unsigned int *gvl*)
- `int16xm8_t vmvvx_int16xm8` (short *a*, unsigned int *gvl*)
- `int32xm1_t vmvvx_int32xm1` (int *a*, unsigned int *gvl*)
- `int32xm2_t vmvvx_int32xm2` (int *a*, unsigned int *gvl*)
- `int32xm4_t vmvvx_int32xm4` (int *a*, unsigned int *gvl*)
- `int32xm8_t vmvvx_int32xm8` (int *a*, unsigned int *gvl*)
- `int64xm1_t vmvvx_int64xm1` (long *a*, unsigned int *gvl*)
- `int64xm2_t vmvvx_int64xm2` (long *a*, unsigned int *gvl*)
- `int64xm4_t vmvvx_int64xm4` (long *a*, unsigned int *gvl*)
- `int64xm8_t vmvvx_int64xm8` (long *a*, unsigned int *gvl*)
- `int8xm1_t vmvvx_int8xm1` (signed char *a*, unsigned int *gvl*)
- `int8xm2_t vmvvx_int8xm2` (signed char *a*, unsigned int *gvl*)
- `int8xm4_t vmvvx_int8xm4` (signed char *a*, unsigned int *gvl*)
- `int8xm8_t vmvvx_int8xm8` (signed char *a*, unsigned int *gvl*)
- `uint16xm1_t vmvvx_uint16xm1` (unsigned short *a*, unsigned int *gvl*)
- `uint16xm2_t vmvvx_uint16xm2` (unsigned short *a*, unsigned int *gvl*)
- `uint16xm4_t vmvvx_uint16xm4` (unsigned short *a*, unsigned int *gvl*)
- `uint16xm8_t vmvvx_uint16xm8` (unsigned short *a*, unsigned int *gvl*)
- `uint32xm1_t vmvvx_uint32xm1` (unsigned int *a*, unsigned int *gvl*)
- `uint32xm2_t vmvvx_uint32xm2` (unsigned int *a*, unsigned int *gvl*)
- `uint32xm4_t vmvvx_uint32xm4` (unsigned int *a*, unsigned int *gvl*)
- `uint32xm8_t vmvvx_uint32xm8` (unsigned int *a*, unsigned int *gvl*)
- `uint64xm1_t vmvvx_uint64xm1` (unsigned long *a*, unsigned int *gvl*)
- `uint64xm2_t vmvvx_uint64xm2` (unsigned long *a*, unsigned int *gvl*)
- `uint64xm4_t vmvvx_uint64xm4` (unsigned long *a*, unsigned int *gvl*)
- `uint64xm8_t vmvvx_uint64xm8` (unsigned long *a*, unsigned int *gvl*)
- `uint8xm1_t vmvvx_uint8xm1` (unsigned char *a*, unsigned int *gvl*)
- `uint8xm2_t vmvvx_uint8xm2` (unsigned char *a*, unsigned int *gvl*)
- `uint8xm4_t vmvvx_uint8xm4` (unsigned char *a*, unsigned int *gvl*)
- `uint8xm8_t vmvvx_uint8xm8` (unsigned char *a*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      result[element] = a
      result[gvl : VLMAX] = 0
```

### 2.12.16 Get first integer element of integer vector

**Instruction:** ['vmv.x.s']

**Prototypes:**

- short **vmvxs\_int16xm1** (*int16xm1\_t a*, unsigned int *gvl*)
- short **vmvxs\_int16xm2** (*int16xm2\_t a*, unsigned int *gvl*)
- short **vmvxs\_int16xm4** (*int16xm4\_t a*, unsigned int *gvl*)
- short **vmvxs\_int16xm8** (*int16xm8\_t a*, unsigned int *gvl*)
- int **vmvxs\_int32xm1** (*int32xm1\_t a*, unsigned int *gvl*)
- int **vmvxs\_int32xm2** (*int32xm2\_t a*, unsigned int *gvl*)
- int **vmvxs\_int32xm4** (*int32xm4\_t a*, unsigned int *gvl*)
- int **vmvxs\_int32xm8** (*int32xm8\_t a*, unsigned int *gvl*)
- long **vmvxs\_int64xm1** (*int64xm1\_t a*, unsigned int *gvl*)
- long **vmvxs\_int64xm2** (*int64xm2\_t a*, unsigned int *gvl*)
- long **vmvxs\_int64xm4** (*int64xm4\_t a*, unsigned int *gvl*)
- long **vmvxs\_int64xm8** (*int64xm8\_t a*, unsigned int *gvl*)
- signed char **vmvxs\_int8xm1** (*int8xm1\_t a*, unsigned int *gvl*)
- signed char **vmvxs\_int8xm2** (*int8xm2\_t a*, unsigned int *gvl*)
- signed char **vmvxs\_int8xm4** (*int8xm4\_t a*, unsigned int *gvl*)
- signed char **vmvxs\_int8xm8** (*int8xm8\_t a*, unsigned int *gvl*)
- unsigned short **vmvxs\_uint16xm1** (*uint16xm1\_t a*, unsigned int *gvl*)
- unsigned short **vmvxs\_uint16xm2** (*uint16xm2\_t a*, unsigned int *gvl*)
- unsigned short **vmvxs\_uint16xm4** (*uint16xm4\_t a*, unsigned int *gvl*)
- unsigned short **vmvxs\_uint16xm8** (*uint16xm8\_t a*, unsigned int *gvl*)
- unsigned int **vmvxs\_uint32xm1** (*uint32xm1\_t a*, unsigned int *gvl*)
- unsigned int **vmvxs\_uint32xm2** (*uint32xm2\_t a*, unsigned int *gvl*)
- unsigned int **vmvxs\_uint32xm4** (*uint32xm4\_t a*, unsigned int *gvl*)
- unsigned int **vmvxs\_uint32xm8** (*uint32xm8\_t a*, unsigned int *gvl*)
- unsigned long **vmvxs\_uint64xm1** (*uint64xm1\_t a*, unsigned int *gvl*)
- unsigned long **vmvxs\_uint64xm2** (*uint64xm2\_t a*, unsigned int *gvl*)
- unsigned long **vmvxs\_uint64xm4** (*uint64xm4\_t a*, unsigned int *gvl*)
- unsigned long **vmvxs\_uint64xm8** (*uint64xm8\_t a*, unsigned int *gvl*)
- unsigned char **vmvxs\_uint8xm1** (*uint8xm1\_t a*, unsigned int *gvl*)



- unsigned char **vmvxs\_uint8xm2** (*uint8xm2\_t a*, unsigned int *gvl*)
- unsigned char **vmvxs\_uint8xm4** (*uint8xm4\_t a*, unsigned int *gvl*)
- unsigned char **vmvxs\_uint8xm8** (*uint8xm8\_t a*, unsigned int *gvl*)

**Operation:**

```
>>> result = a[0]
```

## 2.12.17 Gather vector-immediate (index)

**Instruction:** [`'vrgather.vi'`]**Prototypes:**

- *float16xm1\_t vrgather.vi\_float16xm1* (*float16xm1\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *float16xm2\_t vrgather.vi\_float16xm2* (*float16xm2\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *float16xm4\_t vrgather.vi\_float16xm4* (*float16xm4\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *float16xm8\_t vrgather.vi\_float16xm8* (*float16xm8\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *float32xm1\_t vrgather.vi\_float32xm1* (*float32xm1\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *float32xm2\_t vrgather.vi\_float32xm2* (*float32xm2\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *float32xm4\_t vrgather.vi\_float32xm4* (*float32xm4\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *float32xm8\_t vrgather.vi\_float32xm8* (*float32xm8\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *float64xm1\_t vrgather.vi\_float64xm1* (*float64xm1\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *float64xm2\_t vrgather.vi\_float64xm2* (*float64xm2\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *float64xm4\_t vrgather.vi\_float64xm4* (*float64xm4\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *float64xm8\_t vrgather.vi\_float64xm8* (*float64xm8\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *int16xm1\_t vrgather.vi\_int16xm1* (*int16xm1\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *int16xm2\_t vrgather.vi\_int16xm2* (*int16xm2\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *int16xm4\_t vrgather.vi\_int16xm4* (*int16xm4\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *int16xm8\_t vrgather.vi\_int16xm8* (*int16xm8\_t a*, const unsigned short *b*, unsigned int *gvl*)
- *int32xm1\_t vrgather.vi\_int32xm1* (*int32xm1\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *int32xm2\_t vrgather.vi\_int32xm2* (*int32xm2\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *int32xm4\_t vrgather.vi\_int32xm4* (*int32xm4\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *int32xm8\_t vrgather.vi\_int32xm8* (*int32xm8\_t a*, const unsigned int *b*, unsigned int *gvl*)
- *int64xm1\_t vrgather.vi\_int64xm1* (*int64xm1\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *int64xm2\_t vrgather.vi\_int64xm2* (*int64xm2\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *int64xm4\_t vrgather.vi\_int64xm4* (*int64xm4\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *int64xm8\_t vrgather.vi\_int64xm8* (*int64xm8\_t a*, const unsigned long *b*, unsigned int *gvl*)
- *int8xm1\_t vrgather.vi\_int8xm1* (*int8xm1\_t a*, const unsigned char *b*, unsigned int *gvl*)
- *int8xm2\_t vrgather.vi\_int8xm2* (*int8xm2\_t a*, const unsigned char *b*, unsigned int *gvl*)

- `int8xm4_t vrgathervi_int8xm4 (int8xm4_t a, const unsigned char b, unsigned int gvl)`
- `int8xm8_t vrgathervi_int8xm8 (int8xm8_t a, const unsigned char b, unsigned int gvl)`
- `uint16xm1_t vrgathervi_uint16xm1 (uint16xm1_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm2_t vrgathervi_uint16xm2 (uint16xm2_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm4_t vrgathervi_uint16xm4 (uint16xm4_t a, const unsigned short b, unsigned int gvl)`
- `uint16xm8_t vrgathervi_uint16xm8 (uint16xm8_t a, const unsigned short b, unsigned int gvl)`
- `uint32xm1_t vrgathervi_uint32xm1 (uint32xm1_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm2_t vrgathervi_uint32xm2 (uint32xm2_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm4_t vrgathervi_uint32xm4 (uint32xm4_t a, const unsigned int b, unsigned int gvl)`
- `uint32xm8_t vrgathervi_uint32xm8 (uint32xm8_t a, const unsigned int b, unsigned int gvl)`
- `uint64xm1_t vrgathervi_uint64xm1 (uint64xm1_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm2_t vrgathervi_uint64xm2 (uint64xm2_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm4_t vrgathervi_uint64xm4 (uint64xm4_t a, const unsigned long b, unsigned int gvl)`
- `uint64xm8_t vrgathervi_uint64xm8 (uint64xm8_t a, const unsigned long b, unsigned int gvl)`
- `uint8xm1_t vrgathervi_uint8xm1 (uint8xm1_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm2_t vrgathervi_uint8xm2 (uint8xm2_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm4_t vrgathervi_uint8xm4 (uint8xm4_t a, const unsigned char b, unsigned int gvl)`
- `uint8xm8_t vrgathervi_uint8xm8 (uint8xm8_t a, const unsigned char b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      if b > VLMAX then
        result[element] = 0
      else
        result[element] = a[b]
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vrgathervi_mask_float16xm1 (float16xm1_t merge, float16xm1_t a, const unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `float16xm2_t vrgathervi_mask_float16xm2 (float16xm2_t merge, float16xm2_t a, const unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `float16xm4_t vrgathervi_mask_float16xm4 (float16xm4_t merge, float16xm4_t a, const unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `float16xm8_t vrgathervi_mask_float16xm8 (float16xm8_t merge, float16xm8_t a, const unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `float32xm1_t vrgathervi_mask_float32xm1 (float32xm1_t merge, float32xm1_t a, const unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `float32xm2_t vrgathervi_mask_float32xm2 (float32xm2_t merge, float32xm2_t a, const unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `float32xm4_t vrgathervi_mask_float32xm4 (float32xm4_t merge, float32xm4_t a, const unsigned int b, e32xm4_t mask, unsigned int gvl)`

- *float32xm8\_t vrgathervi\_mask\_float32xm8* (*float32xm8\_t merge, float32xm8\_t a*, const unsigned int *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *float64xm1\_t vrgathervi\_mask\_float64xm1* (*float64xm1\_t merge, float64xm1\_t a*, const unsigned long *b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *float64xm2\_t vrgathervi\_mask\_float64xm2* (*float64xm2\_t merge, float64xm2\_t a*, const unsigned long *b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *float64xm4\_t vrgathervi\_mask\_float64xm4* (*float64xm4\_t merge, float64xm4\_t a*, const unsigned long *b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *float64xm8\_t vrgathervi\_mask\_float64xm8* (*float64xm8\_t merge, float64xm8\_t a*, const unsigned long *b*, *e64xm8\_t mask*, unsigned int *gvl*)
- *int16xm1\_t vrgathervi\_mask\_int16xm1* (*int16xm1\_t merge, int16xm1\_t a*, const unsigned short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vrgathervi\_mask\_int16xm2* (*int16xm2\_t merge, int16xm2\_t a*, const unsigned short *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vrgathervi\_mask\_int16xm4* (*int16xm4\_t merge, int16xm4\_t a*, const unsigned short *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vrgathervi\_mask\_int16xm8* (*int16xm8\_t merge, int16xm8\_t a*, const unsigned short *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *int32xm1\_t vrgathervi\_mask\_int32xm1* (*int32xm1\_t merge, int32xm1\_t a*, const unsigned int *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vrgathervi\_mask\_int32xm2* (*int32xm2\_t merge, int32xm2\_t a*, const unsigned int *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vrgathervi\_mask\_int32xm4* (*int32xm4\_t merge, int32xm4\_t a*, const unsigned int *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *int32xm8\_t vrgathervi\_mask\_int32xm8* (*int32xm8\_t merge, int32xm8\_t a*, const unsigned int *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *int64xm1\_t vrgathervi\_mask\_int64xm1* (*int64xm1\_t merge, int64xm1\_t a*, const unsigned long *b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *int64xm2\_t vrgathervi\_mask\_int64xm2* (*int64xm2\_t merge, int64xm2\_t a*, const unsigned long *b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *int64xm4\_t vrgathervi\_mask\_int64xm4* (*int64xm4\_t merge, int64xm4\_t a*, const unsigned long *b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *int64xm8\_t vrgathervi\_mask\_int64xm8* (*int64xm8\_t merge, int64xm8\_t a*, const unsigned long *b*, *e64xm8\_t mask*, unsigned int *gvl*)
- *int8xm1\_t vrgathervi\_mask\_int8xm1* (*int8xm1\_t merge, int8xm1\_t a*, const unsigned char *b*, *e8xm1\_t mask*, unsigned int *gvl*)
- *int8xm2\_t vrgathervi\_mask\_int8xm2* (*int8xm2\_t merge, int8xm2\_t a*, const unsigned char *b*, *e8xm2\_t mask*, unsigned int *gvl*)
- *int8xm4\_t vrgathervi\_mask\_int8xm4* (*int8xm4\_t merge, int8xm4\_t a*, const unsigned char *b*, *e8xm4\_t mask*, unsigned int *gvl*)
- *int8xm8\_t vrgathervi\_mask\_int8xm8* (*int8xm8\_t merge, int8xm8\_t a*, const unsigned char *b*, *e8xm8\_t mask*, unsigned int *gvl*)
- *uint16xm1\_t vrgathervi\_mask\_uint16xm1* (*uint16xm1\_t merge, uint16xm1\_t a*, const unsigned short *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *uint16xm2\_t vrgathervi\_mask\_uint16xm2* (*uint16xm2\_t merge, uint16xm2\_t a*, const unsigned short *b*, *e16xm2\_t mask*, unsigned int *gvl*)

- `uint16xm4_t vrgathervi_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, const unsigned short `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vrgathervi_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, const unsigned short `b`, `e16xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vrgathervi_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, const unsigned int `b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vrgathervi_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, const unsigned int `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vrgathervi_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, const unsigned int `b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint32xm8_t vrgathervi_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, const unsigned int `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint64xm1_t vrgathervi_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, const unsigned long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vrgathervi_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, const unsigned long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vrgathervi_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, const unsigned long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vrgathervi_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, const unsigned long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vrgathervi_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, const unsigned char `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vrgathervi_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, const unsigned char `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vrgathervi_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, const unsigned char `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vrgathervi_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, const unsigned char `b`, `e8xm8_t mask`, unsigned int `gvl`)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] then
        if b > VLMAX then
            result[elemnt] = 0
        else
            result[element] = a[b]
        else
            result[element] = merge[element]
result[gvl : VLMAX] = 0
```

### 2.12.18 Gather vector-vector (index)

**Instruction:** [`'vrgather.vv'`]

**Prototypes:**

- `float16xm1_t vrgathervv_float16xm1_uint16xm1` (`float16xm1_t a`, `uint16xm1_t b`, unsigned int `gvl`)
- `float16xm2_t vrgathervv_float16xm2_uint16xm2` (`float16xm2_t a`, `uint16xm2_t b`, unsigned int `gvl`)

- `float16xm4_t vrgathervv_float16xm4_uint16xm4` (`float16xm4_t a`, `uint16xm4_t b`, unsigned int gvl)
- `float16xm8_t vrgathervv_float16xm8_uint16xm8` (`float16xm8_t a`, `uint16xm8_t b`, unsigned int gvl)
- `float32xm1_t vrgathervv_float32xm1_uint32xm1` (`float32xm1_t a`, `uint32xm1_t b`, unsigned int gvl)
- `float32xm2_t vrgathervv_float32xm2_uint32xm2` (`float32xm2_t a`, `uint32xm2_t b`, unsigned int gvl)
- `float32xm4_t vrgathervv_float32xm4_uint32xm4` (`float32xm4_t a`, `uint32xm4_t b`, unsigned int gvl)
- `float32xm8_t vrgathervv_float32xm8_uint32xm8` (`float32xm8_t a`, `uint32xm8_t b`, unsigned int gvl)
- `float64xm1_t vrgathervv_float64xm1_uint64xm1` (`float64xm1_t a`, `uint64xm1_t b`, unsigned int gvl)
- `float64xm2_t vrgathervv_float64xm2_uint64xm2` (`float64xm2_t a`, `uint64xm2_t b`, unsigned int gvl)
- `float64xm4_t vrgathervv_float64xm4_uint64xm4` (`float64xm4_t a`, `uint64xm4_t b`, unsigned int gvl)
- `float64xm8_t vrgathervv_float64xm8_uint64xm8` (`float64xm8_t a`, `uint64xm8_t b`, unsigned int gvl)
- `int16xm1_t vrgathervv_int16xm1_uint16xm1` (`int16xm1_t a`, `uint16xm1_t b`, unsigned int gvl)
- `int16xm2_t vrgathervv_int16xm2_uint16xm2` (`int16xm2_t a`, `uint16xm2_t b`, unsigned int gvl)
- `int16xm4_t vrgathervv_int16xm4_uint16xm4` (`int16xm4_t a`, `uint16xm4_t b`, unsigned int gvl)
- `int16xm8_t vrgathervv_int16xm8_uint16xm8` (`int16xm8_t a`, `uint16xm8_t b`, unsigned int gvl)
- `int32xm1_t vrgathervv_int32xm1_uint32xm1` (`int32xm1_t a`, `uint32xm1_t b`, unsigned int gvl)
- `int32xm2_t vrgathervv_int32xm2_uint32xm2` (`int32xm2_t a`, `uint32xm2_t b`, unsigned int gvl)
- `int32xm4_t vrgathervv_int32xm4_uint32xm4` (`int32xm4_t a`, `uint32xm4_t b`, unsigned int gvl)
- `int32xm8_t vrgathervv_int32xm8_uint32xm8` (`int32xm8_t a`, `uint32xm8_t b`, unsigned int gvl)
- `int64xm1_t vrgathervv_int64xm1_uint64xm1` (`int64xm1_t a`, `uint64xm1_t b`, unsigned int gvl)
- `int64xm2_t vrgathervv_int64xm2_uint64xm2` (`int64xm2_t a`, `uint64xm2_t b`, unsigned int gvl)
- `int64xm4_t vrgathervv_int64xm4_uint64xm4` (`int64xm4_t a`, `uint64xm4_t b`, unsigned int gvl)
- `int64xm8_t vrgathervv_int64xm8_uint64xm8` (`int64xm8_t a`, `uint64xm8_t b`, unsigned int gvl)
- `int8xm1_t vrgathervv_int8xm1_uint8xm1` (`int8xm1_t a`, `uint8xm1_t b`, unsigned int gvl)
- `int8xm2_t vrgathervv_int8xm2_uint8xm2` (`int8xm2_t a`, `uint8xm2_t b`, unsigned int gvl)
- `int8xm4_t vrgathervv_int8xm4_uint8xm4` (`int8xm4_t a`, `uint8xm4_t b`, unsigned int gvl)
- `int8xm8_t vrgathervv_int8xm8_uint8xm8` (`int8xm8_t a`, `uint8xm8_t b`, unsigned int gvl)
- `uint16xm1_t vrgathervv_uint16xm1` (`uint16xm1_t a`, `uint16xm1_t b`, unsigned int gvl)
- `uint16xm2_t vrgathervv_uint16xm2` (`uint16xm2_t a`, `uint16xm2_t b`, unsigned int gvl)
- `uint16xm4_t vrgathervv_uint16xm4` (`uint16xm4_t a`, `uint16xm4_t b`, unsigned int gvl)
- `uint16xm8_t vrgathervv_uint16xm8` (`uint16xm8_t a`, `uint16xm8_t b`, unsigned int gvl)
- `uint32xm1_t vrgathervv_uint32xm1` (`uint32xm1_t a`, `uint32xm1_t b`, unsigned int gvl)



- `uint32xm2_t vrgathervv_uint32xm2 (uint32xm2_t a, uint32xm2_t b, unsigned int gvl)`
- `uint32xm4_t vrgathervv_uint32xm4 (uint32xm4_t a, uint32xm4_t b, unsigned int gvl)`
- `uint32xm8_t vrgathervv_uint32xm8 (uint32xm8_t a, uint32xm8_t b, unsigned int gvl)`
- `uint64xm1_t vrgathervv_uint64xm1 (uint64xm1_t a, uint64xm1_t b, unsigned int gvl)`
- `uint64xm2_t vrgathervv_uint64xm2 (uint64xm2_t a, uint64xm2_t b, unsigned int gvl)`
- `uint64xm4_t vrgathervv_uint64xm4 (uint64xm4_t a, uint64xm4_t b, unsigned int gvl)`
- `uint64xm8_t vrgathervv_uint64xm8 (uint64xm8_t a, uint64xm8_t b, unsigned int gvl)`
- `uint8xm1_t vrgathervv_uint8xm1 (uint8xm1_t a, uint8xm1_t b, unsigned int gvl)`
- `uint8xm2_t vrgathervv_uint8xm2 (uint8xm2_t a, uint8xm2_t b, unsigned int gvl)`
- `uint8xm4_t vrgathervv_uint8xm4 (uint8xm4_t a, uint8xm4_t b, unsigned int gvl)`
- `uint8xm8_t vrgathervv_uint8xm8 (uint8xm8_t a, uint8xm8_t b, unsigned int gvl)`

**Operation:**

```
>>> for element = 0 to gvl - 1
      if b[element] > VLMAX then
        result[elemnt] = 0
      else
        result[element] = a[b[element]]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vrgathervv_mask_float16xm1_uint16xm1 (float16xm1_t merge, float16xm1_t a, uint16xm1_t b, e16xm1_t mask, unsigned int gvl)`
- `float16xm2_t vrgathervv_mask_float16xm2_uint16xm2 (float16xm2_t merge, float16xm2_t a, uint16xm2_t b, e16xm2_t mask, unsigned int gvl)`
- `float16xm4_t vrgathervv_mask_float16xm4_uint16xm4 (float16xm4_t merge, float16xm4_t a, uint16xm4_t b, e16xm4_t mask, unsigned int gvl)`
- `float16xm8_t vrgathervv_mask_float16xm8_uint16xm8 (float16xm8_t merge, float16xm8_t a, uint16xm8_t b, e16xm8_t mask, unsigned int gvl)`
- `float32xm1_t vrgathervv_mask_float32xm1_uint32xm1 (float32xm1_t merge, float32xm1_t a, uint32xm1_t b, e32xm1_t mask, unsigned int gvl)`
- `float32xm2_t vrgathervv_mask_float32xm2_uint32xm2 (float32xm2_t merge, float32xm2_t a, uint32xm2_t b, e32xm2_t mask, unsigned int gvl)`
- `float32xm4_t vrgathervv_mask_float32xm4_uint32xm4 (float32xm4_t merge, float32xm4_t a, uint32xm4_t b, e32xm4_t mask, unsigned int gvl)`
- `float32xm8_t vrgathervv_mask_float32xm8_uint32xm8 (float32xm8_t merge, float32xm8_t a, uint32xm8_t b, e32xm8_t mask, unsigned int gvl)`



- *float64xm1\_t vrgathervv\_mask\_float64xm1\_uint64xm1* (*float64xm1\_t merge, float64xm1\_t a, uint64xm1\_t b, e64xm1\_t mask*, unsigned int gvl)
- *float64xm2\_t vrgathervv\_mask\_float64xm2\_uint64xm2* (*float64xm2\_t merge, float64xm2\_t a, uint64xm2\_t b, e64xm2\_t mask*, unsigned int gvl)
- *float64xm4\_t vrgathervv\_mask\_float64xm4\_uint64xm4* (*float64xm4\_t merge, float64xm4\_t a, uint64xm4\_t b, e64xm4\_t mask*, unsigned int gvl)
- *float64xm8\_t vrgathervv\_mask\_float64xm8\_uint64xm8* (*float64xm8\_t merge, float64xm8\_t a, uint64xm8\_t b, e64xm8\_t mask*, unsigned int gvl)
- *int16xm1\_t vrgathervv\_mask\_int16xm1\_uint16xm1* (*int16xm1\_t merge, int16xm1\_t a, uint16xm1\_t b, e16xm1\_t mask*, unsigned int gvl)
- *int16xm2\_t vrgathervv\_mask\_int16xm2\_uint16xm2* (*int16xm2\_t merge, int16xm2\_t a, uint16xm2\_t b, e16xm2\_t mask*, unsigned int gvl)
- *int16xm4\_t vrgathervv\_mask\_int16xm4\_uint16xm4* (*int16xm4\_t merge, int16xm4\_t a, uint16xm4\_t b, e16xm4\_t mask*, unsigned int gvl)
- *int16xm8\_t vrgathervv\_mask\_int16xm8\_uint16xm8* (*int16xm8\_t merge, int16xm8\_t a, uint16xm8\_t b, e16xm8\_t mask*, unsigned int gvl)
- *int32xm1\_t vrgathervv\_mask\_int32xm1\_uint32xm1* (*int32xm1\_t merge, int32xm1\_t a, uint32xm1\_t b, e32xm1\_t mask*, unsigned int gvl)
- *int32xm2\_t vrgathervv\_mask\_int32xm2\_uint32xm2* (*int32xm2\_t merge, int32xm2\_t a, uint32xm2\_t b, e32xm2\_t mask*, unsigned int gvl)
- *int32xm4\_t vrgathervv\_mask\_int32xm4\_uint32xm4* (*int32xm4\_t merge, int32xm4\_t a, uint32xm4\_t b, e32xm4\_t mask*, unsigned int gvl)
- *int32xm8\_t vrgathervv\_mask\_int32xm8\_uint32xm8* (*int32xm8\_t merge, int32xm8\_t a, uint32xm8\_t b, e32xm8\_t mask*, unsigned int gvl)
- *int64xm1\_t vrgathervv\_mask\_int64xm1\_uint64xm1* (*int64xm1\_t merge, int64xm1\_t a, uint64xm1\_t b, e64xm1\_t mask*, unsigned int gvl)
- *int64xm2\_t vrgathervv\_mask\_int64xm2\_uint64xm2* (*int64xm2\_t merge, int64xm2\_t a, uint64xm2\_t b, e64xm2\_t mask*, unsigned int gvl)
- *int64xm4\_t vrgathervv\_mask\_int64xm4\_uint64xm4* (*int64xm4\_t merge, int64xm4\_t a, uint64xm4\_t b, e64xm4\_t mask*, unsigned int gvl)
- *int64xm8\_t vrgathervv\_mask\_int64xm8\_uint64xm8* (*int64xm8\_t merge, int64xm8\_t a, uint64xm8\_t b, e64xm8\_t mask*, unsigned int gvl)

- `int8xm1_t vrgathervv_mask_int8xm1_uint8xm1` (`int8xm1_t merge`, `int8xm1_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int gvl)
- `int8xm2_t vrgathervv_mask_int8xm2_uint8xm2` (`int8xm2_t merge`, `int8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int gvl)
- `int8xm4_t vrgathervv_mask_int8xm4_uint8xm4` (`int8xm4_t merge`, `int8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int gvl)
- `int8xm8_t vrgathervv_mask_int8xm8_uint8xm8` (`int8xm8_t merge`, `int8xm8_t a`, `uint8xm8_t b`, `e8xm8_t mask`, unsigned int gvl)
- `uint16xm1_t vrgathervv_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, `uint16xm1_t b`, `e16xm1_t mask`, unsigned int gvl)
- `uint16xm2_t vrgathervv_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, `uint16xm2_t b`, `e16xm2_t mask`, unsigned int gvl)
- `uint16xm4_t vrgathervv_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, `uint16xm4_t b`, `e16xm4_t mask`, unsigned int gvl)
- `uint16xm8_t vrgathervv_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, `uint16xm8_t b`, `e16xm8_t mask`, unsigned int gvl)
- `uint32xm1_t vrgathervv_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, `uint32xm1_t b`, `e32xm1_t mask`, unsigned int gvl)
- `uint32xm2_t vrgathervv_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, `uint32xm2_t b`, `e32xm2_t mask`, unsigned int gvl)
- `uint32xm4_t vrgathervv_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, `uint32xm4_t b`, `e32xm4_t mask`, unsigned int gvl)
- `uint32xm8_t vrgathervv_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, `uint32xm8_t b`, `e32xm8_t mask`, unsigned int gvl)
- `uint64xm1_t vrgathervv_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, `uint64xm1_t b`, `e64xm1_t mask`, unsigned int gvl)
- `uint64xm2_t vrgathervv_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, `uint64xm2_t b`, `e64xm2_t mask`, unsigned int gvl)
- `uint64xm4_t vrgathervv_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, `uint64xm4_t b`, `e64xm4_t mask`, unsigned int gvl)
- `uint64xm8_t vrgathervv_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, `uint64xm8_t b`, `e64xm8_t mask`, unsigned int gvl)
- `uint8xm1_t vrgathervv_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, `uint8xm1_t b`, `e8xm1_t mask`, unsigned int gvl)
- `uint8xm2_t vrgathervv_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, `uint8xm2_t b`, `e8xm2_t mask`, unsigned int gvl)
- `uint8xm4_t vrgathervv_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, `uint8xm4_t b`, `e8xm4_t mask`, unsigned int gvl)
- `uint8xm8_t vrgathervv_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, `uint8xm8_t b`, `e8xm8_t mask`, unsigned int gvl)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        if b[element] > VLMAX then
          result[elemnt] = 0
        else
```

(continues on next page)

(continued from previous page)

```

        result[element] = a[b[element]]
    else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0

```

## 2.12.19 Gather vector-scalar (index)

**Instruction:** [`'vrgather.vx'`]

**Prototypes:**

- `float16xm1_t vrgathervx_float16xm1` (`float16xm1_t a`, unsigned short `b`, unsigned int `gvl`)
- `float16xm2_t vrgathervx_float16xm2` (`float16xm2_t a`, unsigned short `b`, unsigned int `gvl`)
- `float16xm4_t vrgathervx_float16xm4` (`float16xm4_t a`, unsigned short `b`, unsigned int `gvl`)
- `float16xm8_t vrgathervx_float16xm8` (`float16xm8_t a`, unsigned short `b`, unsigned int `gvl`)
- `float32xm1_t vrgathervx_float32xm1` (`float32xm1_t a`, unsigned int `b`, unsigned int `gvl`)
- `float32xm2_t vrgathervx_float32xm2` (`float32xm2_t a`, unsigned int `b`, unsigned int `gvl`)
- `float32xm4_t vrgathervx_float32xm4` (`float32xm4_t a`, unsigned int `b`, unsigned int `gvl`)
- `float32xm8_t vrgathervx_float32xm8` (`float32xm8_t a`, unsigned int `b`, unsigned int `gvl`)
- `float64xm1_t vrgathervx_float64xm1` (`float64xm1_t a`, unsigned long `b`, unsigned int `gvl`)
- `float64xm2_t vrgathervx_float64xm2` (`float64xm2_t a`, unsigned long `b`, unsigned int `gvl`)
- `float64xm4_t vrgathervx_float64xm4` (`float64xm4_t a`, unsigned long `b`, unsigned int `gvl`)
- `float64xm8_t vrgathervx_float64xm8` (`float64xm8_t a`, unsigned long `b`, unsigned int `gvl`)
- `int16xm1_t vrgathervx_int16xm1` (`int16xm1_t a`, unsigned short `b`, unsigned int `gvl`)
- `int16xm2_t vrgathervx_int16xm2` (`int16xm2_t a`, unsigned short `b`, unsigned int `gvl`)
- `int16xm4_t vrgathervx_int16xm4` (`int16xm4_t a`, unsigned short `b`, unsigned int `gvl`)
- `int16xm8_t vrgathervx_int16xm8` (`int16xm8_t a`, unsigned short `b`, unsigned int `gvl`)
- `int32xm1_t vrgathervx_int32xm1` (`int32xm1_t a`, unsigned int `b`, unsigned int `gvl`)
- `int32xm2_t vrgathervx_int32xm2` (`int32xm2_t a`, unsigned int `b`, unsigned int `gvl`)
- `int32xm4_t vrgathervx_int32xm4` (`int32xm4_t a`, unsigned int `b`, unsigned int `gvl`)
- `int32xm8_t vrgathervx_int32xm8` (`int32xm8_t a`, unsigned int `b`, unsigned int `gvl`)
- `int64xm1_t vrgathervx_int64xm1` (`int64xm1_t a`, unsigned long `b`, unsigned int `gvl`)
- `int64xm2_t vrgathervx_int64xm2` (`int64xm2_t a`, unsigned long `b`, unsigned int `gvl`)
- `int64xm4_t vrgathervx_int64xm4` (`int64xm4_t a`, unsigned long `b`, unsigned int `gvl`)
- `int64xm8_t vrgathervx_int64xm8` (`int64xm8_t a`, unsigned long `b`, unsigned int `gvl`)
- `int8xm1_t vrgathervx_int8xm1` (`int8xm1_t a`, unsigned char `b`, unsigned int `gvl`)
- `int8xm2_t vrgathervx_int8xm2` (`int8xm2_t a`, unsigned char `b`, unsigned int `gvl`)
- `int8xm4_t vrgathervx_int8xm4` (`int8xm4_t a`, unsigned char `b`, unsigned int `gvl`)
- `int8xm8_t vrgathervx_int8xm8` (`int8xm8_t a`, unsigned char `b`, unsigned int `gvl`)

- `uint16xm1_t vrgathervx_uint16xm1 (uint16xm1_t a, unsigned short b, unsigned int gvl)`
- `uint16xm2_t vrgathervx_uint16xm2 (uint16xm2_t a, unsigned short b, unsigned int gvl)`
- `uint16xm4_t vrgathervx_uint16xm4 (uint16xm4_t a, unsigned short b, unsigned int gvl)`
- `uint16xm8_t vrgathervx_uint16xm8 (uint16xm8_t a, unsigned short b, unsigned int gvl)`
- `uint32xm1_t vrgathervx_uint32xm1 (uint32xm1_t a, unsigned int b, unsigned int gvl)`
- `uint32xm2_t vrgathervx_uint32xm2 (uint32xm2_t a, unsigned int b, unsigned int gvl)`
- `uint32xm4_t vrgathervx_uint32xm4 (uint32xm4_t a, unsigned int b, unsigned int gvl)`
- `uint32xm8_t vrgathervx_uint32xm8 (uint32xm8_t a, unsigned int b, unsigned int gvl)`
- `uint64xm1_t vrgathervx_uint64xm1 (uint64xm1_t a, unsigned long b, unsigned int gvl)`
- `uint64xm2_t vrgathervx_uint64xm2 (uint64xm2_t a, unsigned long b, unsigned int gvl)`
- `uint64xm4_t vrgathervx_uint64xm4 (uint64xm4_t a, unsigned long b, unsigned int gvl)`
- `uint64xm8_t vrgathervx_uint64xm8 (uint64xm8_t a, unsigned long b, unsigned int gvl)`
- `uint8xm1_t vrgathervx_uint8xm1 (uint8xm1_t a, unsigned char b, unsigned int gvl)`
- `uint8xm2_t vrgathervx_uint8xm2 (uint8xm2_t a, unsigned char b, unsigned int gvl)`
- `uint8xm4_t vrgathervx_uint8xm4 (uint8xm4_t a, unsigned char b, unsigned int gvl)`
- `uint8xm8_t vrgathervx_uint8xm8 (uint8xm8_t a, unsigned char b, unsigned int gvl)`

#### Operation:

```
>>> for element = 0 to gvl - 1
      if b > VLMAX then
        result[element] = 0
      else
        result[element] = a[b]
      result[gvl : VLMAX] = 0
```

#### Masked prototypes:

- `float16xm1_t vrgathervx_mask_float16xm1 (float16xm1_t merge, float16xm1_t a, unsigned short b, e16xm1_t mask, unsigned int gvl)`
- `float16xm2_t vrgathervx_mask_float16xm2 (float16xm2_t merge, float16xm2_t a, unsigned short b, e16xm2_t mask, unsigned int gvl)`
- `float16xm4_t vrgathervx_mask_float16xm4 (float16xm4_t merge, float16xm4_t a, unsigned short b, e16xm4_t mask, unsigned int gvl)`
- `float16xm8_t vrgathervx_mask_float16xm8 (float16xm8_t merge, float16xm8_t a, unsigned short b, e16xm8_t mask, unsigned int gvl)`
- `float32xm1_t vrgathervx_mask_float32xm1 (float32xm1_t merge, float32xm1_t a, unsigned int b, e32xm1_t mask, unsigned int gvl)`
- `float32xm2_t vrgathervx_mask_float32xm2 (float32xm2_t merge, float32xm2_t a, unsigned int b, e32xm2_t mask, unsigned int gvl)`
- `float32xm4_t vrgathervx_mask_float32xm4 (float32xm4_t merge, float32xm4_t a, unsigned int b, e32xm4_t mask, unsigned int gvl)`
- `float32xm8_t vrgathervx_mask_float32xm8 (float32xm8_t merge, float32xm8_t a, unsigned int b, e32xm8_t mask, unsigned int gvl)`
- `float64xm1_t vrgathervx_mask_float64xm1 (float64xm1_t merge, float64xm1_t a, unsigned long b, e64xm1_t mask, unsigned int gvl)`

- *float64xm2\_t vrgathervx\_mask\_float64xm2* (*float64xm2\_t* merge, *float64xm2\_t* *a*, unsigned long *b*, *e64xm2\_t* mask, unsigned int gvl)
- *float64xm4\_t vrgathervx\_mask\_float64xm4* (*float64xm4\_t* merge, *float64xm4\_t* *a*, unsigned long *b*, *e64xm4\_t* mask, unsigned int gvl)
- *float64xm8\_t vrgathervx\_mask\_float64xm8* (*float64xm8\_t* merge, *float64xm8\_t* *a*, unsigned long *b*, *e64xm8\_t* mask, unsigned int gvl)
- *int16xm1\_t vrgathervx\_mask\_int16xm1* (*int16xm1\_t* merge, *int16xm1\_t* *a*, unsigned short *b*, *e16xm1\_t* mask, unsigned int gvl)
- *int16xm2\_t vrgathervx\_mask\_int16xm2* (*int16xm2\_t* merge, *int16xm2\_t* *a*, unsigned short *b*, *e16xm2\_t* mask, unsigned int gvl)
- *int16xm4\_t vrgathervx\_mask\_int16xm4* (*int16xm4\_t* merge, *int16xm4\_t* *a*, unsigned short *b*, *e16xm4\_t* mask, unsigned int gvl)
- *int16xm8\_t vrgathervx\_mask\_int16xm8* (*int16xm8\_t* merge, *int16xm8\_t* *a*, unsigned short *b*, *e16xm8\_t* mask, unsigned int gvl)
- *int32xm1\_t vrgathervx\_mask\_int32xm1* (*int32xm1\_t* merge, *int32xm1\_t* *a*, unsigned int *b*, *e32xm1\_t* mask, unsigned int gvl)
- *int32xm2\_t vrgathervx\_mask\_int32xm2* (*int32xm2\_t* merge, *int32xm2\_t* *a*, unsigned int *b*, *e32xm2\_t* mask, unsigned int gvl)
- *int32xm4\_t vrgathervx\_mask\_int32xm4* (*int32xm4\_t* merge, *int32xm4\_t* *a*, unsigned int *b*, *e32xm4\_t* mask, unsigned int gvl)
- *int32xm8\_t vrgathervx\_mask\_int32xm8* (*int32xm8\_t* merge, *int32xm8\_t* *a*, unsigned int *b*, *e32xm8\_t* mask, unsigned int gvl)
- *int64xm1\_t vrgathervx\_mask\_int64xm1* (*int64xm1\_t* merge, *int64xm1\_t* *a*, unsigned long *b*, *e64xm1\_t* mask, unsigned int gvl)
- *int64xm2\_t vrgathervx\_mask\_int64xm2* (*int64xm2\_t* merge, *int64xm2\_t* *a*, unsigned long *b*, *e64xm2\_t* mask, unsigned int gvl)
- *int64xm4\_t vrgathervx\_mask\_int64xm4* (*int64xm4\_t* merge, *int64xm4\_t* *a*, unsigned long *b*, *e64xm4\_t* mask, unsigned int gvl)
- *int64xm8\_t vrgathervx\_mask\_int64xm8* (*int64xm8\_t* merge, *int64xm8\_t* *a*, unsigned long *b*, *e64xm8\_t* mask, unsigned int gvl)
- *int8xm1\_t vrgathervx\_mask\_int8xm1* (*int8xm1\_t* merge, *int8xm1\_t* *a*, unsigned char *b*, *e8xm1\_t* mask, unsigned int gvl)
- *int8xm2\_t vrgathervx\_mask\_int8xm2* (*int8xm2\_t* merge, *int8xm2\_t* *a*, unsigned char *b*, *e8xm2\_t* mask, unsigned int gvl)
- *int8xm4\_t vrgathervx\_mask\_int8xm4* (*int8xm4\_t* merge, *int8xm4\_t* *a*, unsigned char *b*, *e8xm4\_t* mask, unsigned int gvl)
- *int8xm8\_t vrgathervx\_mask\_int8xm8* (*int8xm8\_t* merge, *int8xm8\_t* *a*, unsigned char *b*, *e8xm8\_t* mask, unsigned int gvl)
- *uint16xm1\_t vrgathervx\_mask\_uint16xm1* (*uint16xm1\_t* merge, *uint16xm1\_t* *a*, unsigned short *b*, *e16xm1\_t* mask, unsigned int gvl)
- *uint16xm2\_t vrgathervx\_mask\_uint16xm2* (*uint16xm2\_t* merge, *uint16xm2\_t* *a*, unsigned short *b*, *e16xm2\_t* mask, unsigned int gvl)
- *uint16xm4\_t vrgathervx\_mask\_uint16xm4* (*uint16xm4\_t* merge, *uint16xm4\_t* *a*, unsigned short *b*, *e16xm4\_t* mask, unsigned int gvl)
- *uint16xm8\_t vrgathervx\_mask\_uint16xm8* (*uint16xm8\_t* merge, *uint16xm8\_t* *a*, unsigned short *b*, *e16xm8\_t* mask, unsigned int gvl)



- `uint32xm1_t vrgathervx_mask_uint32xm1` (`uint32xm1_t` merge, `uint32xm1_t` *a*, unsigned int *b*, `e32xm1_t` mask, unsigned int *gvl*)
- `uint32xm2_t vrgathervx_mask_uint32xm2` (`uint32xm2_t` merge, `uint32xm2_t` *a*, unsigned int *b*, `e32xm2_t` mask, unsigned int *gvl*)
- `uint32xm4_t vrgathervx_mask_uint32xm4` (`uint32xm4_t` merge, `uint32xm4_t` *a*, unsigned int *b*, `e32xm4_t` mask, unsigned int *gvl*)
- `uint32xm8_t vrgathervx_mask_uint32xm8` (`uint32xm8_t` merge, `uint32xm8_t` *a*, unsigned int *b*, `e32xm8_t` mask, unsigned int *gvl*)
- `uint64xm1_t vrgathervx_mask_uint64xm1` (`uint64xm1_t` merge, `uint64xm1_t` *a*, unsigned long *b*, `e64xm1_t` mask, unsigned int *gvl*)
- `uint64xm2_t vrgathervx_mask_uint64xm2` (`uint64xm2_t` merge, `uint64xm2_t` *a*, unsigned long *b*, `e64xm2_t` mask, unsigned int *gvl*)
- `uint64xm4_t vrgathervx_mask_uint64xm4` (`uint64xm4_t` merge, `uint64xm4_t` *a*, unsigned long *b*, `e64xm4_t` mask, unsigned int *gvl*)
- `uint64xm8_t vrgathervx_mask_uint64xm8` (`uint64xm8_t` merge, `uint64xm8_t` *a*, unsigned long *b*, `e64xm8_t` mask, unsigned int *gvl*)
- `uint8xm1_t vrgathervx_mask_uint8xm1` (`uint8xm1_t` merge, `uint8xm1_t` *a*, unsigned char *b*, `e8xm1_t` mask, unsigned int *gvl*)
- `uint8xm2_t vrgathervx_mask_uint8xm2` (`uint8xm2_t` merge, `uint8xm2_t` *a*, unsigned char *b*, `e8xm2_t` mask, unsigned int *gvl*)
- `uint8xm4_t vrgathervx_mask_uint8xm4` (`uint8xm4_t` merge, `uint8xm4_t` *a*, unsigned char *b*, `e8xm4_t` mask, unsigned int *gvl*)
- `uint8xm8_t vrgathervx_mask_uint8xm8` (`uint8xm8_t` merge, `uint8xm8_t` *a*, unsigned char *b*, `e8xm8_t` mask, unsigned int *gvl*)

#### Masked operation:

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        if b > VLMAX then
          result[element] = 0
        else
          result[element] = a[b]
        else
          result[element] = merge[element]
      result[gvl : VLMAX] = 0
```

### 2.12.20 Slide down one element of a vector

**Instruction:** [`'vslide1down.vx'`]

#### Prototypes:

- `float16xm1_t vslide1downvx_float16xm1` (`float16xm1_t` *a*, long *b*, unsigned int *gvl*)
- `float16xm2_t vslide1downvx_float16xm2` (`float16xm2_t` *a*, long *b*, unsigned int *gvl*)
- `float16xm4_t vslide1downvx_float16xm4` (`float16xm4_t` *a*, long *b*, unsigned int *gvl*)
- `float16xm8_t vslide1downvx_float16xm8` (`float16xm8_t` *a*, long *b*, unsigned int *gvl*)
- `float32xm1_t vslide1downvx_float32xm1` (`float32xm1_t` *a*, long *b*, unsigned int *gvl*)
- `float32xm2_t vslide1downvx_float32xm2` (`float32xm2_t` *a*, long *b*, unsigned int *gvl*)



- *float32xm4\_t vsldel1downvx\_float32xm4* (*float32xm4\_t a*, long *b*, unsigned int *gvl*)
- *float32xm8\_t vsldel1downvx\_float32xm8* (*float32xm8\_t a*, long *b*, unsigned int *gvl*)
- *float64xm1\_t vsldel1downvx\_float64xm1* (*float64xm1\_t a*, long *b*, unsigned int *gvl*)
- *float64xm2\_t vsldel1downvx\_float64xm2* (*float64xm2\_t a*, long *b*, unsigned int *gvl*)
- *float64xm4\_t vsldel1downvx\_float64xm4* (*float64xm4\_t a*, long *b*, unsigned int *gvl*)
- *float64xm8\_t vsldel1downvx\_float64xm8* (*float64xm8\_t a*, long *b*, unsigned int *gvl*)
- *int16xm1\_t vsldel1downvx\_int16xm1* (*int16xm1\_t a*, long *b*, unsigned int *gvl*)
- *int16xm2\_t vsldel1downvx\_int16xm2* (*int16xm2\_t a*, long *b*, unsigned int *gvl*)
- *int16xm4\_t vsldel1downvx\_int16xm4* (*int16xm4\_t a*, long *b*, unsigned int *gvl*)
- *int16xm8\_t vsldel1downvx\_int16xm8* (*int16xm8\_t a*, long *b*, unsigned int *gvl*)
- *int32xm1\_t vsldel1downvx\_int32xm1* (*int32xm1\_t a*, long *b*, unsigned int *gvl*)
- *int32xm2\_t vsldel1downvx\_int32xm2* (*int32xm2\_t a*, long *b*, unsigned int *gvl*)
- *int32xm4\_t vsldel1downvx\_int32xm4* (*int32xm4\_t a*, long *b*, unsigned int *gvl*)
- *int32xm8\_t vsldel1downvx\_int32xm8* (*int32xm8\_t a*, long *b*, unsigned int *gvl*)
- *int64xm1\_t vsldel1downvx\_int64xm1* (*int64xm1\_t a*, long *b*, unsigned int *gvl*)
- *int64xm2\_t vsldel1downvx\_int64xm2* (*int64xm2\_t a*, long *b*, unsigned int *gvl*)
- *int64xm4\_t vsldel1downvx\_int64xm4* (*int64xm4\_t a*, long *b*, unsigned int *gvl*)
- *int64xm8\_t vsldel1downvx\_int64xm8* (*int64xm8\_t a*, long *b*, unsigned int *gvl*)
- *int8xm1\_t vsldel1downvx\_int8xm1* (*int8xm1\_t a*, long *b*, unsigned int *gvl*)
- *int8xm2\_t vsldel1downvx\_int8xm2* (*int8xm2\_t a*, long *b*, unsigned int *gvl*)
- *int8xm4\_t vsldel1downvx\_int8xm4* (*int8xm4\_t a*, long *b*, unsigned int *gvl*)
- *int8xm8\_t vsldel1downvx\_int8xm8* (*int8xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm1\_t vsldel1downvx\_uint16xm1* (*uint16xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm2\_t vsldel1downvx\_uint16xm2* (*uint16xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm4\_t vsldel1downvx\_uint16xm4* (*uint16xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm8\_t vsldel1downvx\_uint16xm8* (*uint16xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm1\_t vsldel1downvx\_uint32xm1* (*uint32xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm2\_t vsldel1downvx\_uint32xm2* (*uint32xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm4\_t vsldel1downvx\_uint32xm4* (*uint32xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm8\_t vsldel1downvx\_uint32xm8* (*uint32xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint64xm1\_t vsldel1downvx\_uint64xm1* (*uint64xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint64xm2\_t vsldel1downvx\_uint64xm2* (*uint64xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint64xm4\_t vsldel1downvx\_uint64xm4* (*uint64xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint64xm8\_t vsldel1downvx\_uint64xm8* (*uint64xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint8xm1\_t vsldel1downvx\_uint8xm1* (*uint8xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint8xm2\_t vsldel1downvx\_uint8xm2* (*uint8xm2\_t a*, long *b*, unsigned int *gvl*)

- *uint8xm4\_t vslide1downvx\_uint8xm4* (*uint8xm4\_t* *a*, long *b*, unsigned int *gvl*)
- *uint8xm8\_t vslide1downvx\_uint8xm8* (*uint8xm8\_t* *a*, long *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = b to gvl - 1
      if element == gvl - 1 then result[element] = b
      else if element + 1 < VLMAX then
        result[element] = a[element + 1]
      else
        result[element] = 0
    result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vslide1downvx\_mask\_float16xm1* (*float16xm1\_t* *merge*, *float16xm1\_t* *a*, long *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *float16xm2\_t vslide1downvx\_mask\_float16xm2* (*float16xm2\_t* *merge*, *float16xm2\_t* *a*, long *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *float16xm4\_t vslide1downvx\_mask\_float16xm4* (*float16xm4\_t* *merge*, *float16xm4\_t* *a*, long *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *float16xm8\_t vslide1downvx\_mask\_float16xm8* (*float16xm8\_t* *merge*, *float16xm8\_t* *a*, long *b*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- *float32xm1\_t vslide1downvx\_mask\_float32xm1* (*float32xm1\_t* *merge*, *float32xm1\_t* *a*, long *b*, *e32xm1\_t* *mask*, unsigned int *gvl*)
- *float32xm2\_t vslide1downvx\_mask\_float32xm2* (*float32xm2\_t* *merge*, *float32xm2\_t* *a*, long *b*, *e32xm2\_t* *mask*, unsigned int *gvl*)
- *float32xm4\_t vslide1downvx\_mask\_float32xm4* (*float32xm4\_t* *merge*, *float32xm4\_t* *a*, long *b*, *e32xm4\_t* *mask*, unsigned int *gvl*)
- *float32xm8\_t vslide1downvx\_mask\_float32xm8* (*float32xm8\_t* *merge*, *float32xm8\_t* *a*, long *b*, *e32xm8\_t* *mask*, unsigned int *gvl*)
- *float64xm1\_t vslide1downvx\_mask\_float64xm1* (*float64xm1\_t* *merge*, *float64xm1\_t* *a*, long *b*, *e64xm1\_t* *mask*, unsigned int *gvl*)
- *float64xm2\_t vslide1downvx\_mask\_float64xm2* (*float64xm2\_t* *merge*, *float64xm2\_t* *a*, long *b*, *e64xm2\_t* *mask*, unsigned int *gvl*)
- *float64xm4\_t vslide1downvx\_mask\_float64xm4* (*float64xm4\_t* *merge*, *float64xm4\_t* *a*, long *b*, *e64xm4\_t* *mask*, unsigned int *gvl*)
- *float64xm8\_t vslide1downvx\_mask\_float64xm8* (*float64xm8\_t* *merge*, *float64xm8\_t* *a*, long *b*, *e64xm8\_t* *mask*, unsigned int *gvl*)
- *int16xm1\_t vslide1downvx\_mask\_int16xm1* (*int16xm1\_t* *merge*, *int16xm1\_t* *a*, long *b*, *e16xm1\_t* *mask*, unsigned int *gvl*)
- *int16xm2\_t vslide1downvx\_mask\_int16xm2* (*int16xm2\_t* *merge*, *int16xm2\_t* *a*, long *b*, *e16xm2\_t* *mask*, unsigned int *gvl*)
- *int16xm4\_t vslide1downvx\_mask\_int16xm4* (*int16xm4\_t* *merge*, *int16xm4\_t* *a*, long *b*, *e16xm4\_t* *mask*, unsigned int *gvl*)
- *int16xm8\_t vslide1downvx\_mask\_int16xm8* (*int16xm8\_t* *merge*, *int16xm8\_t* *a*, long *b*, *e16xm8\_t* *mask*, unsigned int *gvl*)
- *int32xm1\_t vslide1downvx\_mask\_int32xm1* (*int32xm1\_t* *merge*, *int32xm1\_t* *a*, long *b*, *e32xm1\_t* *mask*, unsigned int *gvl*)

- `int32xm2_t vslide1downvx_mask_int32xm2` (`int32xm2_t merge`, `int32xm2_t a`, long `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `int32xm4_t vslide1downvx_mask_int32xm4` (`int32xm4_t merge`, `int32xm4_t a`, long `b`, `e32xm4_t mask`, unsigned int `gvl`)
- `int32xm8_t vslide1downvx_mask_int32xm8` (`int32xm8_t merge`, `int32xm8_t a`, long `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `int64xm1_t vslide1downvx_mask_int64xm1` (`int64xm1_t merge`, `int64xm1_t a`, long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `int64xm2_t vslide1downvx_mask_int64xm2` (`int64xm2_t merge`, `int64xm2_t a`, long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `int64xm4_t vslide1downvx_mask_int64xm4` (`int64xm4_t merge`, `int64xm4_t a`, long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `int64xm8_t vslide1downvx_mask_int64xm8` (`int64xm8_t merge`, `int64xm8_t a`, long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `int8xm1_t vslide1downvx_mask_int8xm1` (`int8xm1_t merge`, `int8xm1_t a`, long `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `int8xm2_t vslide1downvx_mask_int8xm2` (`int8xm2_t merge`, `int8xm2_t a`, long `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `int8xm4_t vslide1downvx_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, long `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int8xm8_t vslide1downvx_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, long `b`, `e8xm8_t mask`, unsigned int `gvl`)
- `uint16xm1_t vslide1downvx_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, long `b`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint16xm2_t vslide1downvx_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, long `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint16xm4_t vslide1downvx_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, long `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vslide1downvx_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, long `b`, `e16xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vslide1downvx_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, long `b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vslide1downvx_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, long `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vslide1downvx_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, long `b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint32xm8_t vslide1downvx_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, long `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint64xm1_t vslide1downvx_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vslide1downvx_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vslide1downvx_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vslide1downvx_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, long `b`, `e64xm8_t mask`, unsigned int `gvl`)

- `uint8xm1_t vslide1downvx_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, long `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vslide1downvx_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, long `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vslide1downvx_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, long `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vslide1downvx_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, long `b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = b to gvl - 1
      if element == gvl - 1 then result[element] = b
      else if element + 1 < VLMAX then
        result[element] = a[element + 1]
      else
        result[element] = 0
result[gvl : VLMAX] = 0
```

### 2.12.21 Slide up one element of a vector

Instruction: [`'vslide1up.vx'`]

Prototypes:

- `float16xm1_t vslide1upvx_float16xm1` (`float16xm1_t a`, long `b`, unsigned int `gvl`)
- `float16xm2_t vslide1upvx_float16xm2` (`float16xm2_t a`, long `b`, unsigned int `gvl`)
- `float16xm4_t vslide1upvx_float16xm4` (`float16xm4_t a`, long `b`, unsigned int `gvl`)
- `float16xm8_t vslide1upvx_float16xm8` (`float16xm8_t a`, long `b`, unsigned int `gvl`)
- `float32xm1_t vslide1upvx_float32xm1` (`float32xm1_t a`, long `b`, unsigned int `gvl`)
- `float32xm2_t vslide1upvx_float32xm2` (`float32xm2_t a`, long `b`, unsigned int `gvl`)
- `float32xm4_t vslide1upvx_float32xm4` (`float32xm4_t a`, long `b`, unsigned int `gvl`)
- `float32xm8_t vslide1upvx_float32xm8` (`float32xm8_t a`, long `b`, unsigned int `gvl`)
- `float64xm1_t vslide1upvx_float64xm1` (`float64xm1_t a`, long `b`, unsigned int `gvl`)
- `float64xm2_t vslide1upvx_float64xm2` (`float64xm2_t a`, long `b`, unsigned int `gvl`)
- `float64xm4_t vslide1upvx_float64xm4` (`float64xm4_t a`, long `b`, unsigned int `gvl`)
- `float64xm8_t vslide1upvx_float64xm8` (`float64xm8_t a`, long `b`, unsigned int `gvl`)
- `int16xm1_t vslide1upvx_int16xm1` (`int16xm1_t a`, long `b`, unsigned int `gvl`)
- `int16xm2_t vslide1upvx_int16xm2` (`int16xm2_t a`, long `b`, unsigned int `gvl`)
- `int16xm4_t vslide1upvx_int16xm4` (`int16xm4_t a`, long `b`, unsigned int `gvl`)
- `int16xm8_t vslide1upvx_int16xm8` (`int16xm8_t a`, long `b`, unsigned int `gvl`)
- `int32xm1_t vslide1upvx_int32xm1` (`int32xm1_t a`, long `b`, unsigned int `gvl`)
- `int32xm2_t vslide1upvx_int32xm2` (`int32xm2_t a`, long `b`, unsigned int `gvl`)
- `int32xm4_t vslide1upvx_int32xm4` (`int32xm4_t a`, long `b`, unsigned int `gvl`)

- *int32xm8\_t vsldelupvx\_int32xm8* (*int32xm8\_t a*, long *b*, unsigned int *gvl*)
- *int64xm1\_t vsldelupvx\_int64xm1* (*int64xm1\_t a*, long *b*, unsigned int *gvl*)
- *int64xm2\_t vsldelupvx\_int64xm2* (*int64xm2\_t a*, long *b*, unsigned int *gvl*)
- *int64xm4\_t vsldelupvx\_int64xm4* (*int64xm4\_t a*, long *b*, unsigned int *gvl*)
- *int64xm8\_t vsldelupvx\_int64xm8* (*int64xm8\_t a*, long *b*, unsigned int *gvl*)
- *int8xm1\_t vsldelupvx\_int8xm1* (*int8xm1\_t a*, long *b*, unsigned int *gvl*)
- *int8xm2\_t vsldelupvx\_int8xm2* (*int8xm2\_t a*, long *b*, unsigned int *gvl*)
- *int8xm4\_t vsldelupvx\_int8xm4* (*int8xm4\_t a*, long *b*, unsigned int *gvl*)
- *int8xm8\_t vsldelupvx\_int8xm8* (*int8xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm1\_t vsldelupvx\_uint16xm1* (*uint16xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm2\_t vsldelupvx\_uint16xm2* (*uint16xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm4\_t vsldelupvx\_uint16xm4* (*uint16xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm8\_t vsldelupvx\_uint16xm8* (*uint16xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm1\_t vsldelupvx\_uint32xm1* (*uint32xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm2\_t vsldelupvx\_uint32xm2* (*uint32xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm4\_t vsldelupvx\_uint32xm4* (*uint32xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm8\_t vsldelupvx\_uint32xm8* (*uint32xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint64xm1\_t vsldelupvx\_uint64xm1* (*uint64xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint64xm2\_t vsldelupvx\_uint64xm2* (*uint64xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint64xm4\_t vsldelupvx\_uint64xm4* (*uint64xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint64xm8\_t vsldelupvx\_uint64xm8* (*uint64xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint8xm1\_t vsldelupvx\_uint8xm1* (*uint8xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint8xm2\_t vsldelupvx\_uint8xm2* (*uint8xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint8xm4\_t vsldelupvx\_uint8xm4* (*uint8xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint8xm8\_t vsldelupvx\_uint8xm8* (*uint8xm8\_t a*, long *b*, unsigned int *gvl*)

**Operation:**

```
>>> for element = 0 to gvl - 1
      if element == 0 then
        result[0] = b
      else
        result[element] = a[element - 1]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- *float16xm1\_t vsldelupvx\_mask\_float16xm1* (*float16xm1\_t merge*, *float16xm1\_t a*, long *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *float16xm2\_t vsldelupvx\_mask\_float16xm2* (*float16xm2\_t merge*, *float16xm2\_t a*, long *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *float16xm4\_t vsldelupvx\_mask\_float16xm4* (*float16xm4\_t merge*, *float16xm4\_t a*, long *b*, *e16xm4\_t mask*, unsigned int *gvl*)



- *float16xm8\_t vslidelupvx\_mask\_float16xm8* (*float16xm8\_t* merge, *float16xm8\_t* *a*, long *b*, *e16xm8\_t* mask, unsigned int gvl)
- *float32xm1\_t vslidelupvx\_mask\_float32xm1* (*float32xm1\_t* merge, *float32xm1\_t* *a*, long *b*, *e32xm1\_t* mask, unsigned int gvl)
- *float32xm2\_t vslidelupvx\_mask\_float32xm2* (*float32xm2\_t* merge, *float32xm2\_t* *a*, long *b*, *e32xm2\_t* mask, unsigned int gvl)
- *float32xm4\_t vslidelupvx\_mask\_float32xm4* (*float32xm4\_t* merge, *float32xm4\_t* *a*, long *b*, *e32xm4\_t* mask, unsigned int gvl)
- *float32xm8\_t vslidelupvx\_mask\_float32xm8* (*float32xm8\_t* merge, *float32xm8\_t* *a*, long *b*, *e32xm8\_t* mask, unsigned int gvl)
- *float64xm1\_t vslidelupvx\_mask\_float64xm1* (*float64xm1\_t* merge, *float64xm1\_t* *a*, long *b*, *e64xm1\_t* mask, unsigned int gvl)
- *float64xm2\_t vslidelupvx\_mask\_float64xm2* (*float64xm2\_t* merge, *float64xm2\_t* *a*, long *b*, *e64xm2\_t* mask, unsigned int gvl)
- *float64xm4\_t vslidelupvx\_mask\_float64xm4* (*float64xm4\_t* merge, *float64xm4\_t* *a*, long *b*, *e64xm4\_t* mask, unsigned int gvl)
- *float64xm8\_t vslidelupvx\_mask\_float64xm8* (*float64xm8\_t* merge, *float64xm8\_t* *a*, long *b*, *e64xm8\_t* mask, unsigned int gvl)
- *int16xm1\_t vslidelupvx\_mask\_int16xm1* (*int16xm1\_t* merge, *int16xm1\_t* *a*, long *b*, *e16xm1\_t* mask, unsigned int gvl)
- *int16xm2\_t vslidelupvx\_mask\_int16xm2* (*int16xm2\_t* merge, *int16xm2\_t* *a*, long *b*, *e16xm2\_t* mask, unsigned int gvl)
- *int16xm4\_t vslidelupvx\_mask\_int16xm4* (*int16xm4\_t* merge, *int16xm4\_t* *a*, long *b*, *e16xm4\_t* mask, unsigned int gvl)
- *int16xm8\_t vslidelupvx\_mask\_int16xm8* (*int16xm8\_t* merge, *int16xm8\_t* *a*, long *b*, *e16xm8\_t* mask, unsigned int gvl)
- *int32xm1\_t vslidelupvx\_mask\_int32xm1* (*int32xm1\_t* merge, *int32xm1\_t* *a*, long *b*, *e32xm1\_t* mask, unsigned int gvl)
- *int32xm2\_t vslidelupvx\_mask\_int32xm2* (*int32xm2\_t* merge, *int32xm2\_t* *a*, long *b*, *e32xm2\_t* mask, unsigned int gvl)
- *int32xm4\_t vslidelupvx\_mask\_int32xm4* (*int32xm4\_t* merge, *int32xm4\_t* *a*, long *b*, *e32xm4\_t* mask, unsigned int gvl)
- *int32xm8\_t vslidelupvx\_mask\_int32xm8* (*int32xm8\_t* merge, *int32xm8\_t* *a*, long *b*, *e32xm8\_t* mask, unsigned int gvl)
- *int64xm1\_t vslidelupvx\_mask\_int64xm1* (*int64xm1\_t* merge, *int64xm1\_t* *a*, long *b*, *e64xm1\_t* mask, unsigned int gvl)
- *int64xm2\_t vslidelupvx\_mask\_int64xm2* (*int64xm2\_t* merge, *int64xm2\_t* *a*, long *b*, *e64xm2\_t* mask, unsigned int gvl)
- *int64xm4\_t vslidelupvx\_mask\_int64xm4* (*int64xm4\_t* merge, *int64xm4\_t* *a*, long *b*, *e64xm4\_t* mask, unsigned int gvl)
- *int64xm8\_t vslidelupvx\_mask\_int64xm8* (*int64xm8\_t* merge, *int64xm8\_t* *a*, long *b*, *e64xm8\_t* mask, unsigned int gvl)
- *int8xm1\_t vslidelupvx\_mask\_int8xm1* (*int8xm1\_t* merge, *int8xm1\_t* *a*, long *b*, *e8xm1\_t* mask, unsigned int gvl)
- *int8xm2\_t vslidelupvx\_mask\_int8xm2* (*int8xm2\_t* merge, *int8xm2\_t* *a*, long *b*, *e8xm2\_t* mask, unsigned int gvl)



- `int8xm4_t vslidelupvx_mask_int8xm4` (`int8xm4_t merge`, `int8xm4_t a`, long `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `int8xm8_t vslidelupvx_mask_int8xm8` (`int8xm8_t merge`, `int8xm8_t a`, long `b`, `e8xm8_t mask`, unsigned int `gvl`)
- `uint16xm1_t vslidelupvx_mask_uint16xm1` (`uint16xm1_t merge`, `uint16xm1_t a`, long `b`, `e16xm1_t mask`, unsigned int `gvl`)
- `uint16xm2_t vslidelupvx_mask_uint16xm2` (`uint16xm2_t merge`, `uint16xm2_t a`, long `b`, `e16xm2_t mask`, unsigned int `gvl`)
- `uint16xm4_t vslidelupvx_mask_uint16xm4` (`uint16xm4_t merge`, `uint16xm4_t a`, long `b`, `e16xm4_t mask`, unsigned int `gvl`)
- `uint16xm8_t vslidelupvx_mask_uint16xm8` (`uint16xm8_t merge`, `uint16xm8_t a`, long `b`, `e16xm8_t mask`, unsigned int `gvl`)
- `uint32xm1_t vslidelupvx_mask_uint32xm1` (`uint32xm1_t merge`, `uint32xm1_t a`, long `b`, `e32xm1_t mask`, unsigned int `gvl`)
- `uint32xm2_t vslidelupvx_mask_uint32xm2` (`uint32xm2_t merge`, `uint32xm2_t a`, long `b`, `e32xm2_t mask`, unsigned int `gvl`)
- `uint32xm4_t vslidelupvx_mask_uint32xm4` (`uint32xm4_t merge`, `uint32xm4_t a`, long `b`, `e32xm4_t mask`, unsigned int `gvl`)
- `uint32xm8_t vslidelupvx_mask_uint32xm8` (`uint32xm8_t merge`, `uint32xm8_t a`, long `b`, `e32xm8_t mask`, unsigned int `gvl`)
- `uint64xm1_t vslidelupvx_mask_uint64xm1` (`uint64xm1_t merge`, `uint64xm1_t a`, long `b`, `e64xm1_t mask`, unsigned int `gvl`)
- `uint64xm2_t vslidelupvx_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, long `b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vslidelupvx_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, long `b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vslidelupvx_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, long `b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vslidelupvx_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, long `b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vslidelupvx_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, long `b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vslidelupvx_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, long `b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vslidelupvx_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, long `b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] then
        if element == 0 then result[element] = b
      else
        result[element] = a[element - 1]
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.12.22 Slide down elements of vector-immediate (indexed)

**Instruction:** [`vslidedown.vi`]

**Prototypes:**

- `float16xm1_t vslidedownvi_float16xm1 (float16xm1_t a, const long b, unsigned int gvl)`
- `float16xm2_t vslidedownvi_float16xm2 (float16xm2_t a, const long b, unsigned int gvl)`
- `float16xm4_t vslidedownvi_float16xm4 (float16xm4_t a, const long b, unsigned int gvl)`
- `float16xm8_t vslidedownvi_float16xm8 (float16xm8_t a, const long b, unsigned int gvl)`
- `float32xm1_t vslidedownvi_float32xm1 (float32xm1_t a, const long b, unsigned int gvl)`
- `float32xm2_t vslidedownvi_float32xm2 (float32xm2_t a, const long b, unsigned int gvl)`
- `float32xm4_t vslidedownvi_float32xm4 (float32xm4_t a, const long b, unsigned int gvl)`
- `float32xm8_t vslidedownvi_float32xm8 (float32xm8_t a, const long b, unsigned int gvl)`
- `float64xm1_t vslidedownvi_float64xm1 (float64xm1_t a, const long b, unsigned int gvl)`
- `float64xm2_t vslidedownvi_float64xm2 (float64xm2_t a, const long b, unsigned int gvl)`
- `float64xm4_t vslidedownvi_float64xm4 (float64xm4_t a, const long b, unsigned int gvl)`
- `float64xm8_t vslidedownvi_float64xm8 (float64xm8_t a, const long b, unsigned int gvl)`
- `int16xm1_t vslidedownvi_int16xm1 (int16xm1_t a, const long b, unsigned int gvl)`
- `int16xm2_t vslidedownvi_int16xm2 (int16xm2_t a, const long b, unsigned int gvl)`
- `int16xm4_t vslidedownvi_int16xm4 (int16xm4_t a, const long b, unsigned int gvl)`
- `int16xm8_t vslidedownvi_int16xm8 (int16xm8_t a, const long b, unsigned int gvl)`
- `int32xm1_t vslidedownvi_int32xm1 (int32xm1_t a, const long b, unsigned int gvl)`
- `int32xm2_t vslidedownvi_int32xm2 (int32xm2_t a, const long b, unsigned int gvl)`
- `int32xm4_t vslidedownvi_int32xm4 (int32xm4_t a, const long b, unsigned int gvl)`
- `int32xm8_t vslidedownvi_int32xm8 (int32xm8_t a, const long b, unsigned int gvl)`
- `int64xm1_t vslidedownvi_int64xm1 (int64xm1_t a, const long b, unsigned int gvl)`
- `int64xm2_t vslidedownvi_int64xm2 (int64xm2_t a, const long b, unsigned int gvl)`
- `int64xm4_t vslidedownvi_int64xm4 (int64xm4_t a, const long b, unsigned int gvl)`
- `int64xm8_t vslidedownvi_int64xm8 (int64xm8_t a, const long b, unsigned int gvl)`
- `int8xm1_t vslidedownvi_int8xm1 (int8xm1_t a, const long b, unsigned int gvl)`
- `int8xm2_t vslidedownvi_int8xm2 (int8xm2_t a, const long b, unsigned int gvl)`
- `int8xm4_t vslidedownvi_int8xm4 (int8xm4_t a, const long b, unsigned int gvl)`
- `int8xm8_t vslidedownvi_int8xm8 (int8xm8_t a, const long b, unsigned int gvl)`
- `uint16xm1_t vslidedownvi_uint16xm1 (uint16xm1_t a, const long b, unsigned int gvl)`
- `uint16xm2_t vslidedownvi_uint16xm2 (uint16xm2_t a, const long b, unsigned int gvl)`
- `uint16xm4_t vslidedownvi_uint16xm4 (uint16xm4_t a, const long b, unsigned int gvl)`
- `uint16xm8_t vslidedownvi_uint16xm8 (uint16xm8_t a, const long b, unsigned int gvl)`
- `uint32xm1_t vslidedownvi_uint32xm1 (uint32xm1_t a, const long b, unsigned int gvl)`

- `uint32xm2_t vslidedownvi_uint32xm2 (uint32xm2_t a, const long b, unsigned int gvl)`
- `uint32xm4_t vslidedownvi_uint32xm4 (uint32xm4_t a, const long b, unsigned int gvl)`
- `uint32xm8_t vslidedownvi_uint32xm8 (uint32xm8_t a, const long b, unsigned int gvl)`
- `uint64xm1_t vslidedownvi_uint64xm1 (uint64xm1_t a, const long b, unsigned int gvl)`
- `uint64xm2_t vslidedownvi_uint64xm2 (uint64xm2_t a, const long b, unsigned int gvl)`
- `uint64xm4_t vslidedownvi_uint64xm4 (uint64xm4_t a, const long b, unsigned int gvl)`
- `uint64xm8_t vslidedownvi_uint64xm8 (uint64xm8_t a, const long b, unsigned int gvl)`
- `uint8xm1_t vslidedownvi_uint8xm1 (uint8xm1_t a, const long b, unsigned int gvl)`
- `uint8xm2_t vslidedownvi_uint8xm2 (uint8xm2_t a, const long b, unsigned int gvl)`
- `uint8xm4_t vslidedownvi_uint8xm4 (uint8xm4_t a, const long b, unsigned int gvl)`
- `uint8xm8_t vslidedownvi_uint8xm8 (uint8xm8_t a, const long b, unsigned int gvl)`

**Operation:**

```
>>> for element = b to gvl - 1
      if element - b < 0
        result[element] = result[element]
      else
        result[element] = a[element - b]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vslidedownvi_mask_float16xm1 (float16xm1_t merge, float16xm1_t a, const long b, e16xm1_t mask, unsigned int gvl)`
- `float16xm2_t vslidedownvi_mask_float16xm2 (float16xm2_t merge, float16xm2_t a, const long b, e16xm2_t mask, unsigned int gvl)`
- `float16xm4_t vslidedownvi_mask_float16xm4 (float16xm4_t merge, float16xm4_t a, const long b, e16xm4_t mask, unsigned int gvl)`
- `float16xm8_t vslidedownvi_mask_float16xm8 (float16xm8_t merge, float16xm8_t a, const long b, e16xm8_t mask, unsigned int gvl)`
- `float32xm1_t vslidedownvi_mask_float32xm1 (float32xm1_t merge, float32xm1_t a, const long b, e32xm1_t mask, unsigned int gvl)`
- `float32xm2_t vslidedownvi_mask_float32xm2 (float32xm2_t merge, float32xm2_t a, const long b, e32xm2_t mask, unsigned int gvl)`
- `float32xm4_t vslidedownvi_mask_float32xm4 (float32xm4_t merge, float32xm4_t a, const long b, e32xm4_t mask, unsigned int gvl)`
- `float32xm8_t vslidedownvi_mask_float32xm8 (float32xm8_t merge, float32xm8_t a, const long b, e32xm8_t mask, unsigned int gvl)`
- `float64xm1_t vslidedownvi_mask_float64xm1 (float64xm1_t merge, float64xm1_t a, const long b, e64xm1_t mask, unsigned int gvl)`
- `float64xm2_t vslidedownvi_mask_float64xm2 (float64xm2_t merge, float64xm2_t a, const long b, e64xm2_t mask, unsigned int gvl)`
- `float64xm4_t vslidedownvi_mask_float64xm4 (float64xm4_t merge, float64xm4_t a, const long b, e64xm4_t mask, unsigned int gvl)`
- `float64xm8_t vslidedownvi_mask_float64xm8 (float64xm8_t merge, float64xm8_t a, const long b, e64xm8_t mask, unsigned int gvl)`

- *int16xm1\_t vslidedownvi\_mask\_int16xm1* (*int16xm1\_t merge*, *int16xm1\_t a*, const long *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *int16xm2\_t vslidedownvi\_mask\_int16xm2* (*int16xm2\_t merge*, *int16xm2\_t a*, const long *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *int16xm4\_t vslidedownvi\_mask\_int16xm4* (*int16xm4\_t merge*, *int16xm4\_t a*, const long *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *int16xm8\_t vslidedownvi\_mask\_int16xm8* (*int16xm8\_t merge*, *int16xm8\_t a*, const long *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *int32xm1\_t vslidedownvi\_mask\_int32xm1* (*int32xm1\_t merge*, *int32xm1\_t a*, const long *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *int32xm2\_t vslidedownvi\_mask\_int32xm2* (*int32xm2\_t merge*, *int32xm2\_t a*, const long *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *int32xm4\_t vslidedownvi\_mask\_int32xm4* (*int32xm4\_t merge*, *int32xm4\_t a*, const long *b*, *e32xm4\_t mask*, unsigned int *gvl*)
- *int32xm8\_t vslidedownvi\_mask\_int32xm8* (*int32xm8\_t merge*, *int32xm8\_t a*, const long *b*, *e32xm8\_t mask*, unsigned int *gvl*)
- *int64xm1\_t vslidedownvi\_mask\_int64xm1* (*int64xm1\_t merge*, *int64xm1\_t a*, const long *b*, *e64xm1\_t mask*, unsigned int *gvl*)
- *int64xm2\_t vslidedownvi\_mask\_int64xm2* (*int64xm2\_t merge*, *int64xm2\_t a*, const long *b*, *e64xm2\_t mask*, unsigned int *gvl*)
- *int64xm4\_t vslidedownvi\_mask\_int64xm4* (*int64xm4\_t merge*, *int64xm4\_t a*, const long *b*, *e64xm4\_t mask*, unsigned int *gvl*)
- *int64xm8\_t vslidedownvi\_mask\_int64xm8* (*int64xm8\_t merge*, *int64xm8\_t a*, const long *b*, *e64xm8\_t mask*, unsigned int *gvl*)
- *int8xm1\_t vslidedownvi\_mask\_int8xm1* (*int8xm1\_t merge*, *int8xm1\_t a*, const long *b*, *e8xm1\_t mask*, unsigned int *gvl*)
- *int8xm2\_t vslidedownvi\_mask\_int8xm2* (*int8xm2\_t merge*, *int8xm2\_t a*, const long *b*, *e8xm2\_t mask*, unsigned int *gvl*)
- *int8xm4\_t vslidedownvi\_mask\_int8xm4* (*int8xm4\_t merge*, *int8xm4\_t a*, const long *b*, *e8xm4\_t mask*, unsigned int *gvl*)
- *int8xm8\_t vslidedownvi\_mask\_int8xm8* (*int8xm8\_t merge*, *int8xm8\_t a*, const long *b*, *e8xm8\_t mask*, unsigned int *gvl*)
- *uint16xm1\_t vslidedownvi\_mask\_uint16xm1* (*uint16xm1\_t merge*, *uint16xm1\_t a*, const long *b*, *e16xm1\_t mask*, unsigned int *gvl*)
- *uint16xm2\_t vslidedownvi\_mask\_uint16xm2* (*uint16xm2\_t merge*, *uint16xm2\_t a*, const long *b*, *e16xm2\_t mask*, unsigned int *gvl*)
- *uint16xm4\_t vslidedownvi\_mask\_uint16xm4* (*uint16xm4\_t merge*, *uint16xm4\_t a*, const long *b*, *e16xm4\_t mask*, unsigned int *gvl*)
- *uint16xm8\_t vslidedownvi\_mask\_uint16xm8* (*uint16xm8\_t merge*, *uint16xm8\_t a*, const long *b*, *e16xm8\_t mask*, unsigned int *gvl*)
- *uint32xm1\_t vslidedownvi\_mask\_uint32xm1* (*uint32xm1\_t merge*, *uint32xm1\_t a*, const long *b*, *e32xm1\_t mask*, unsigned int *gvl*)
- *uint32xm2\_t vslidedownvi\_mask\_uint32xm2* (*uint32xm2\_t merge*, *uint32xm2\_t a*, const long *b*, *e32xm2\_t mask*, unsigned int *gvl*)
- *uint32xm4\_t vslidedownvi\_mask\_uint32xm4* (*uint32xm4\_t merge*, *uint32xm4\_t a*, const long *b*, *e32xm4\_t mask*, unsigned int *gvl*)

- `uint32xm8_t vslidedownvi_mask_uint32xm8` (`uint32xm8_t` merge, `uint32xm8_t` *a*, const long *b*, `e32xm8_t` mask, unsigned int *gvl*)
- `uint64xm1_t vslidedownvi_mask_uint64xm1` (`uint64xm1_t` merge, `uint64xm1_t` *a*, const long *b*, `e64xm1_t` mask, unsigned int *gvl*)
- `uint64xm2_t vslidedownvi_mask_uint64xm2` (`uint64xm2_t` merge, `uint64xm2_t` *a*, const long *b*, `e64xm2_t` mask, unsigned int *gvl*)
- `uint64xm4_t vslidedownvi_mask_uint64xm4` (`uint64xm4_t` merge, `uint64xm4_t` *a*, const long *b*, `e64xm4_t` mask, unsigned int *gvl*)
- `uint64xm8_t vslidedownvi_mask_uint64xm8` (`uint64xm8_t` merge, `uint64xm8_t` *a*, const long *b*, `e64xm8_t` mask, unsigned int *gvl*)
- `uint8xm1_t vslidedownvi_mask_uint8xm1` (`uint8xm1_t` merge, `uint8xm1_t` *a*, const long *b*, `e8xm1_t` mask, unsigned int *gvl*)
- `uint8xm2_t vslidedownvi_mask_uint8xm2` (`uint8xm2_t` merge, `uint8xm2_t` *a*, const long *b*, `e8xm2_t` mask, unsigned int *gvl*)
- `uint8xm4_t vslidedownvi_mask_uint8xm4` (`uint8xm4_t` merge, `uint8xm4_t` *a*, const long *b*, `e8xm4_t` mask, unsigned int *gvl*)
- `uint8xm8_t vslidedownvi_mask_uint8xm8` (`uint8xm8_t` merge, `uint8xm8_t` *a*, const long *b*, `e8xm8_t` mask, unsigned int *gvl*)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
      if mask[element] and element - b >= 0 then
        result[element] = a[element - b]
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

**2.12.23 Slide down elements of vector-scalar (indexed)****Instruction:** [`'vslidedown.vx'`]**Prototypes:**

- `float16xm1_t vslidedownvx_float16xm1` (`float16xm1_t` *a*, long *b*, unsigned int *gvl*)
- `float16xm2_t vslidedownvx_float16xm2` (`float16xm2_t` *a*, long *b*, unsigned int *gvl*)
- `float16xm4_t vslidedownvx_float16xm4` (`float16xm4_t` *a*, long *b*, unsigned int *gvl*)
- `float16xm8_t vslidedownvx_float16xm8` (`float16xm8_t` *a*, long *b*, unsigned int *gvl*)
- `float32xm1_t vslidedownvx_float32xm1` (`float32xm1_t` *a*, long *b*, unsigned int *gvl*)
- `float32xm2_t vslidedownvx_float32xm2` (`float32xm2_t` *a*, long *b*, unsigned int *gvl*)
- `float32xm4_t vslidedownvx_float32xm4` (`float32xm4_t` *a*, long *b*, unsigned int *gvl*)
- `float32xm8_t vslidedownvx_float32xm8` (`float32xm8_t` *a*, long *b*, unsigned int *gvl*)
- `float64xm1_t vslidedownvx_float64xm1` (`float64xm1_t` *a*, long *b*, unsigned int *gvl*)
- `float64xm2_t vslidedownvx_float64xm2` (`float64xm2_t` *a*, long *b*, unsigned int *gvl*)
- `float64xm4_t vslidedownvx_float64xm4` (`float64xm4_t` *a*, long *b*, unsigned int *gvl*)
- `float64xm8_t vslidedownvx_float64xm8` (`float64xm8_t` *a*, long *b*, unsigned int *gvl*)



- *int16xm1\_t vslidedownvx\_int16xm1* (*int16xm1\_t a*, long *b*, unsigned int *gvl*)
- *int16xm2\_t vslidedownvx\_int16xm2* (*int16xm2\_t a*, long *b*, unsigned int *gvl*)
- *int16xm4\_t vslidedownvx\_int16xm4* (*int16xm4\_t a*, long *b*, unsigned int *gvl*)
- *int16xm8\_t vslidedownvx\_int16xm8* (*int16xm8\_t a*, long *b*, unsigned int *gvl*)
- *int32xm1\_t vslidedownvx\_int32xm1* (*int32xm1\_t a*, long *b*, unsigned int *gvl*)
- *int32xm2\_t vslidedownvx\_int32xm2* (*int32xm2\_t a*, long *b*, unsigned int *gvl*)
- *int32xm4\_t vslidedownvx\_int32xm4* (*int32xm4\_t a*, long *b*, unsigned int *gvl*)
- *int32xm8\_t vslidedownvx\_int32xm8* (*int32xm8\_t a*, long *b*, unsigned int *gvl*)
- *int64xm1\_t vslidedownvx\_int64xm1* (*int64xm1\_t a*, long *b*, unsigned int *gvl*)
- *int64xm2\_t vslidedownvx\_int64xm2* (*int64xm2\_t a*, long *b*, unsigned int *gvl*)
- *int64xm4\_t vslidedownvx\_int64xm4* (*int64xm4\_t a*, long *b*, unsigned int *gvl*)
- *int64xm8\_t vslidedownvx\_int64xm8* (*int64xm8\_t a*, long *b*, unsigned int *gvl*)
- *int8xm1\_t vslidedownvx\_int8xm1* (*int8xm1\_t a*, long *b*, unsigned int *gvl*)
- *int8xm2\_t vslidedownvx\_int8xm2* (*int8xm2\_t a*, long *b*, unsigned int *gvl*)
- *int8xm4\_t vslidedownvx\_int8xm4* (*int8xm4\_t a*, long *b*, unsigned int *gvl*)
- *int8xm8\_t vslidedownvx\_int8xm8* (*int8xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm1\_t vslidedownvx\_uint16xm1* (*uint16xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm2\_t vslidedownvx\_uint16xm2* (*uint16xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm4\_t vslidedownvx\_uint16xm4* (*uint16xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm8\_t vslidedownvx\_uint16xm8* (*uint16xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm1\_t vslidedownvx\_uint32xm1* (*uint32xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm2\_t vslidedownvx\_uint32xm2* (*uint32xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm4\_t vslidedownvx\_uint32xm4* (*uint32xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm8\_t vslidedownvx\_uint32xm8* (*uint32xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint64xm1\_t vslidedownvx\_uint64xm1* (*uint64xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint64xm2\_t vslidedownvx\_uint64xm2* (*uint64xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint64xm4\_t vslidedownvx\_uint64xm4* (*uint64xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint64xm8\_t vslidedownvx\_uint64xm8* (*uint64xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint8xm1\_t vslidedownvx\_uint8xm1* (*uint8xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint8xm2\_t vslidedownvx\_uint8xm2* (*uint8xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint8xm4\_t vslidedownvx\_uint8xm4* (*uint8xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint8xm8\_t vslidedownvx\_uint8xm8* (*uint8xm8\_t a*, long *b*, unsigned int *gvl*)

**Operation:**



```

>>> for element = b to gvl - 1
      if element + b < VLMAX then
        result[element] = a[element + b]
      else
        result[element] = 0
      result[gvl : VLMAX] = 0

```

#### Masked prototypes:

- *float16xm1\_t vslidedownvx\_mask\_float16xm1* (*float16xm1\_t* merge, *float16xm1\_t* a, long b, *e16xm1\_t* mask, unsigned int gvl)
- *float16xm2\_t vslidedownvx\_mask\_float16xm2* (*float16xm2\_t* merge, *float16xm2\_t* a, long b, *e16xm2\_t* mask, unsigned int gvl)
- *float16xm4\_t vslidedownvx\_mask\_float16xm4* (*float16xm4\_t* merge, *float16xm4\_t* a, long b, *e16xm4\_t* mask, unsigned int gvl)
- *float16xm8\_t vslidedownvx\_mask\_float16xm8* (*float16xm8\_t* merge, *float16xm8\_t* a, long b, *e16xm8\_t* mask, unsigned int gvl)
- *float32xm1\_t vslidedownvx\_mask\_float32xm1* (*float32xm1\_t* merge, *float32xm1\_t* a, long b, *e32xm1\_t* mask, unsigned int gvl)
- *float32xm2\_t vslidedownvx\_mask\_float32xm2* (*float32xm2\_t* merge, *float32xm2\_t* a, long b, *e32xm2\_t* mask, unsigned int gvl)
- *float32xm4\_t vslidedownvx\_mask\_float32xm4* (*float32xm4\_t* merge, *float32xm4\_t* a, long b, *e32xm4\_t* mask, unsigned int gvl)
- *float32xm8\_t vslidedownvx\_mask\_float32xm8* (*float32xm8\_t* merge, *float32xm8\_t* a, long b, *e32xm8\_t* mask, unsigned int gvl)
- *float64xm1\_t vslidedownvx\_mask\_float64xm1* (*float64xm1\_t* merge, *float64xm1\_t* a, long b, *e64xm1\_t* mask, unsigned int gvl)
- *float64xm2\_t vslidedownvx\_mask\_float64xm2* (*float64xm2\_t* merge, *float64xm2\_t* a, long b, *e64xm2\_t* mask, unsigned int gvl)
- *float64xm4\_t vslidedownvx\_mask\_float64xm4* (*float64xm4\_t* merge, *float64xm4\_t* a, long b, *e64xm4\_t* mask, unsigned int gvl)
- *float64xm8\_t vslidedownvx\_mask\_float64xm8* (*float64xm8\_t* merge, *float64xm8\_t* a, long b, *e64xm8\_t* mask, unsigned int gvl)
- *int16xm1\_t vslidedownvx\_mask\_int16xm1* (*int16xm1\_t* merge, *int16xm1\_t* a, long b, *e16xm1\_t* mask, unsigned int gvl)
- *int16xm2\_t vslidedownvx\_mask\_int16xm2* (*int16xm2\_t* merge, *int16xm2\_t* a, long b, *e16xm2\_t* mask, unsigned int gvl)
- *int16xm4\_t vslidedownvx\_mask\_int16xm4* (*int16xm4\_t* merge, *int16xm4\_t* a, long b, *e16xm4\_t* mask, unsigned int gvl)
- *int16xm8\_t vslidedownvx\_mask\_int16xm8* (*int16xm8\_t* merge, *int16xm8\_t* a, long b, *e16xm8\_t* mask, unsigned int gvl)
- *int32xm1\_t vslidedownvx\_mask\_int32xm1* (*int32xm1\_t* merge, *int32xm1\_t* a, long b, *e32xm1\_t* mask, unsigned int gvl)
- *int32xm2\_t vslidedownvx\_mask\_int32xm2* (*int32xm2\_t* merge, *int32xm2\_t* a, long b, *e32xm2\_t* mask, unsigned int gvl)
- *int32xm4\_t vslidedownvx\_mask\_int32xm4* (*int32xm4\_t* merge, *int32xm4\_t* a, long b, *e32xm4\_t* mask, unsigned int gvl)

- `int32xm8_t vslidedownvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, long b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vslidedownvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vslidedownvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vslidedownvx_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vslidedownvx_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vslidedownvx_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, long b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vslidedownvx_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, long b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vslidedownvx_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, long b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vslidedownvx_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, long b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vslidedownvx_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, long b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vslidedownvx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, long b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vslidedownvx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, long b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vslidedownvx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, long b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vslidedownvx_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, long b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vslidedownvx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, long b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vslidedownvx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, long b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vslidedownvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, long b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vslidedownvx_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vslidedownvx_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vslidedownvx_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vslidedownvx_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vslidedownvx_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, long b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vslidedownvx_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, long b, e8xm2_t mask, unsigned int gvl)`

- `uint8xm4_t vslidedownvx_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, `long b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vslidedownvx_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, `long b`, `e8xm8_t mask`, unsigned int `gvl`)

Masked operation:

```
>>> for element = b to gvl - 1
      if mask[element] then
        if element + b < VLMAX then
          result[element] = a[element + b]
        else
          result[element] = 0
      else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

## 2.12.24 Slide up elements of vector-immediate (indexed)

Instruction: [`'vslideup.vi'`]

Prototypes:

- `float16xm1_t vslideupvi_float16xm1` (`float16xm1_t a`, const long `b`, unsigned int `gvl`)
- `float16xm2_t vslideupvi_float16xm2` (`float16xm2_t a`, const long `b`, unsigned int `gvl`)
- `float16xm4_t vslideupvi_float16xm4` (`float16xm4_t a`, const long `b`, unsigned int `gvl`)
- `float16xm8_t vslideupvi_float16xm8` (`float16xm8_t a`, const long `b`, unsigned int `gvl`)
- `float32xm1_t vslideupvi_float32xm1` (`float32xm1_t a`, const long `b`, unsigned int `gvl`)
- `float32xm2_t vslideupvi_float32xm2` (`float32xm2_t a`, const long `b`, unsigned int `gvl`)
- `float32xm4_t vslideupvi_float32xm4` (`float32xm4_t a`, const long `b`, unsigned int `gvl`)
- `float32xm8_t vslideupvi_float32xm8` (`float32xm8_t a`, const long `b`, unsigned int `gvl`)
- `float64xm1_t vslideupvi_float64xm1` (`float64xm1_t a`, const long `b`, unsigned int `gvl`)
- `float64xm2_t vslideupvi_float64xm2` (`float64xm2_t a`, const long `b`, unsigned int `gvl`)
- `float64xm4_t vslideupvi_float64xm4` (`float64xm4_t a`, const long `b`, unsigned int `gvl`)
- `float64xm8_t vslideupvi_float64xm8` (`float64xm8_t a`, const long `b`, unsigned int `gvl`)
- `int16xm1_t vslideupvi_int16xm1` (`int16xm1_t a`, const long `b`, unsigned int `gvl`)
- `int16xm2_t vslideupvi_int16xm2` (`int16xm2_t a`, const long `b`, unsigned int `gvl`)
- `int16xm4_t vslideupvi_int16xm4` (`int16xm4_t a`, const long `b`, unsigned int `gvl`)
- `int16xm8_t vslideupvi_int16xm8` (`int16xm8_t a`, const long `b`, unsigned int `gvl`)
- `int32xm1_t vslideupvi_int32xm1` (`int32xm1_t a`, const long `b`, unsigned int `gvl`)
- `int32xm2_t vslideupvi_int32xm2` (`int32xm2_t a`, const long `b`, unsigned int `gvl`)
- `int32xm4_t vslideupvi_int32xm4` (`int32xm4_t a`, const long `b`, unsigned int `gvl`)
- `int32xm8_t vslideupvi_int32xm8` (`int32xm8_t a`, const long `b`, unsigned int `gvl`)
- `int64xm1_t vslideupvi_int64xm1` (`int64xm1_t a`, const long `b`, unsigned int `gvl`)

- `int64xm2_t vslideupvi_int64xm2 (int64xm2_t a, const long b, unsigned int gvl)`
- `int64xm4_t vslideupvi_int64xm4 (int64xm4_t a, const long b, unsigned int gvl)`
- `int64xm8_t vslideupvi_int64xm8 (int64xm8_t a, const long b, unsigned int gvl)`
- `int8xm1_t vslideupvi_int8xm1 (int8xm1_t a, const long b, unsigned int gvl)`
- `int8xm2_t vslideupvi_int8xm2 (int8xm2_t a, const long b, unsigned int gvl)`
- `int8xm4_t vslideupvi_int8xm4 (int8xm4_t a, const long b, unsigned int gvl)`
- `int8xm8_t vslideupvi_int8xm8 (int8xm8_t a, const long b, unsigned int gvl)`
- `uint16xm1_t vslideupvi_uint16xm1 (uint16xm1_t a, const long b, unsigned int gvl)`
- `uint16xm2_t vslideupvi_uint16xm2 (uint16xm2_t a, const long b, unsigned int gvl)`
- `uint16xm4_t vslideupvi_uint16xm4 (uint16xm4_t a, const long b, unsigned int gvl)`
- `uint16xm8_t vslideupvi_uint16xm8 (uint16xm8_t a, const long b, unsigned int gvl)`
- `uint32xm1_t vslideupvi_uint32xm1 (uint32xm1_t a, const long b, unsigned int gvl)`
- `uint32xm2_t vslideupvi_uint32xm2 (uint32xm2_t a, const long b, unsigned int gvl)`
- `uint32xm4_t vslideupvi_uint32xm4 (uint32xm4_t a, const long b, unsigned int gvl)`
- `uint32xm8_t vslideupvi_uint32xm8 (uint32xm8_t a, const long b, unsigned int gvl)`
- `uint64xm1_t vslideupvi_uint64xm1 (uint64xm1_t a, const long b, unsigned int gvl)`
- `uint64xm2_t vslideupvi_uint64xm2 (uint64xm2_t a, const long b, unsigned int gvl)`
- `uint64xm4_t vslideupvi_uint64xm4 (uint64xm4_t a, const long b, unsigned int gvl)`
- `uint64xm8_t vslideupvi_uint64xm8 (uint64xm8_t a, const long b, unsigned int gvl)`
- `uint8xm1_t vslideupvi_uint8xm1 (uint8xm1_t a, const long b, unsigned int gvl)`
- `uint8xm2_t vslideupvi_uint8xm2 (uint8xm2_t a, const long b, unsigned int gvl)`
- `uint8xm4_t vslideupvi_uint8xm4 (uint8xm4_t a, const long b, unsigned int gvl)`
- `uint8xm8_t vslideupvi_uint8xm8 (uint8xm8_t a, const long b, unsigned int gvl)`

**Operation:**

```
>>> for element = b to gvl - 1
    if element - b < 0
        result[element] = result[element]
    else result[element] = a[element - b]
result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vslideupvi_mask_float16xm1 (float16xm1_t merge, float16xm1_t a, const long b, e16xm1_t mask, unsigned int gvl)`
- `float16xm2_t vslideupvi_mask_float16xm2 (float16xm2_t merge, float16xm2_t a, const long b, e16xm2_t mask, unsigned int gvl)`
- `float16xm4_t vslideupvi_mask_float16xm4 (float16xm4_t merge, float16xm4_t a, const long b, e16xm4_t mask, unsigned int gvl)`
- `float16xm8_t vslideupvi_mask_float16xm8 (float16xm8_t merge, float16xm8_t a, const long b, e16xm8_t mask, unsigned int gvl)`

- *float32xm1\_t vslideupvi\_mask\_float32xm1* (*float32xm1\_t* merge, *float32xm1\_t* *a*, const long *b*, *e32xm1\_t* mask, unsigned int gvl)
- *float32xm2\_t vslideupvi\_mask\_float32xm2* (*float32xm2\_t* merge, *float32xm2\_t* *a*, const long *b*, *e32xm2\_t* mask, unsigned int gvl)
- *float32xm4\_t vslideupvi\_mask\_float32xm4* (*float32xm4\_t* merge, *float32xm4\_t* *a*, const long *b*, *e32xm4\_t* mask, unsigned int gvl)
- *float32xm8\_t vslideupvi\_mask\_float32xm8* (*float32xm8\_t* merge, *float32xm8\_t* *a*, const long *b*, *e32xm8\_t* mask, unsigned int gvl)
- *float64xm1\_t vslideupvi\_mask\_float64xm1* (*float64xm1\_t* merge, *float64xm1\_t* *a*, const long *b*, *e64xm1\_t* mask, unsigned int gvl)
- *float64xm2\_t vslideupvi\_mask\_float64xm2* (*float64xm2\_t* merge, *float64xm2\_t* *a*, const long *b*, *e64xm2\_t* mask, unsigned int gvl)
- *float64xm4\_t vslideupvi\_mask\_float64xm4* (*float64xm4\_t* merge, *float64xm4\_t* *a*, const long *b*, *e64xm4\_t* mask, unsigned int gvl)
- *float64xm8\_t vslideupvi\_mask\_float64xm8* (*float64xm8\_t* merge, *float64xm8\_t* *a*, const long *b*, *e64xm8\_t* mask, unsigned int gvl)
- *int16xm1\_t vslideupvi\_mask\_int16xm1* (*int16xm1\_t* merge, *int16xm1\_t* *a*, const long *b*, *e16xm1\_t* mask, unsigned int gvl)
- *int16xm2\_t vslideupvi\_mask\_int16xm2* (*int16xm2\_t* merge, *int16xm2\_t* *a*, const long *b*, *e16xm2\_t* mask, unsigned int gvl)
- *int16xm4\_t vslideupvi\_mask\_int16xm4* (*int16xm4\_t* merge, *int16xm4\_t* *a*, const long *b*, *e16xm4\_t* mask, unsigned int gvl)
- *int16xm8\_t vslideupvi\_mask\_int16xm8* (*int16xm8\_t* merge, *int16xm8\_t* *a*, const long *b*, *e16xm8\_t* mask, unsigned int gvl)
- *int32xm1\_t vslideupvi\_mask\_int32xm1* (*int32xm1\_t* merge, *int32xm1\_t* *a*, const long *b*, *e32xm1\_t* mask, unsigned int gvl)
- *int32xm2\_t vslideupvi\_mask\_int32xm2* (*int32xm2\_t* merge, *int32xm2\_t* *a*, const long *b*, *e32xm2\_t* mask, unsigned int gvl)
- *int32xm4\_t vslideupvi\_mask\_int32xm4* (*int32xm4\_t* merge, *int32xm4\_t* *a*, const long *b*, *e32xm4\_t* mask, unsigned int gvl)
- *int32xm8\_t vslideupvi\_mask\_int32xm8* (*int32xm8\_t* merge, *int32xm8\_t* *a*, const long *b*, *e32xm8\_t* mask, unsigned int gvl)
- *int64xm1\_t vslideupvi\_mask\_int64xm1* (*int64xm1\_t* merge, *int64xm1\_t* *a*, const long *b*, *e64xm1\_t* mask, unsigned int gvl)
- *int64xm2\_t vslideupvi\_mask\_int64xm2* (*int64xm2\_t* merge, *int64xm2\_t* *a*, const long *b*, *e64xm2\_t* mask, unsigned int gvl)
- *int64xm4\_t vslideupvi\_mask\_int64xm4* (*int64xm4\_t* merge, *int64xm4\_t* *a*, const long *b*, *e64xm4\_t* mask, unsigned int gvl)
- *int64xm8\_t vslideupvi\_mask\_int64xm8* (*int64xm8\_t* merge, *int64xm8\_t* *a*, const long *b*, *e64xm8\_t* mask, unsigned int gvl)
- *int8xm1\_t vslideupvi\_mask\_int8xm1* (*int8xm1\_t* merge, *int8xm1\_t* *a*, const long *b*, *e8xm1\_t* mask, unsigned int gvl)
- *int8xm2\_t vslideupvi\_mask\_int8xm2* (*int8xm2\_t* merge, *int8xm2\_t* *a*, const long *b*, *e8xm2\_t* mask, unsigned int gvl)
- *int8xm4\_t vslideupvi\_mask\_int8xm4* (*int8xm4\_t* merge, *int8xm4\_t* *a*, const long *b*, *e8xm4\_t* mask, unsigned int gvl)



- `int8xm8_t vslideupvi_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, const long b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vslideupvi_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, const long b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vslideupvi_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, const long b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vslideupvi_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, const long b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vslideupvi_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, const long b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vslideupvi_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, const long b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vslideupvi_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, const long b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vslideupvi_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, const long b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vslideupvi_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, const long b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vslideupvi_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, const long b, e64xm1_t mask, unsigned int gvl)`
- `uint64xm2_t vslideupvi_mask_uint64xm2 (uint64xm2_t merge, uint64xm2_t a, const long b, e64xm2_t mask, unsigned int gvl)`
- `uint64xm4_t vslideupvi_mask_uint64xm4 (uint64xm4_t merge, uint64xm4_t a, const long b, e64xm4_t mask, unsigned int gvl)`
- `uint64xm8_t vslideupvi_mask_uint64xm8 (uint64xm8_t merge, uint64xm8_t a, const long b, e64xm8_t mask, unsigned int gvl)`
- `uint8xm1_t vslideupvi_mask_uint8xm1 (uint8xm1_t merge, uint8xm1_t a, const long b, e8xm1_t mask, unsigned int gvl)`
- `uint8xm2_t vslideupvi_mask_uint8xm2 (uint8xm2_t merge, uint8xm2_t a, const long b, e8xm2_t mask, unsigned int gvl)`
- `uint8xm4_t vslideupvi_mask_uint8xm4 (uint8xm4_t merge, uint8xm4_t a, const long b, e8xm4_t mask, unsigned int gvl)`
- `uint8xm8_t vslideupvi_mask_uint8xm8 (uint8xm8_t merge, uint8xm8_t a, const long b, e8xm8_t mask, unsigned int gvl)`

#### Masked operation:

```
>>> for element = 0 to gvl - 1
    if mask[element] and element - b >= 0 then
        result[element] = a[element - b]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

### 2.12.25 Slide up elements of vector-scalar (indexed)

**Instruction:** ['vslideup.vx']

**Prototypes:**



- *float16xm1\_t vslideupvx\_float16xm1* (*float16xm1\_t a*, long *b*, unsigned int *gvl*)
- *float16xm2\_t vslideupvx\_float16xm2* (*float16xm2\_t a*, long *b*, unsigned int *gvl*)
- *float16xm4\_t vslideupvx\_float16xm4* (*float16xm4\_t a*, long *b*, unsigned int *gvl*)
- *float16xm8\_t vslideupvx\_float16xm8* (*float16xm8\_t a*, long *b*, unsigned int *gvl*)
- *float32xm1\_t vslideupvx\_float32xm1* (*float32xm1\_t a*, long *b*, unsigned int *gvl*)
- *float32xm2\_t vslideupvx\_float32xm2* (*float32xm2\_t a*, long *b*, unsigned int *gvl*)
- *float32xm4\_t vslideupvx\_float32xm4* (*float32xm4\_t a*, long *b*, unsigned int *gvl*)
- *float32xm8\_t vslideupvx\_float32xm8* (*float32xm8\_t a*, long *b*, unsigned int *gvl*)
- *float64xm1\_t vslideupvx\_float64xm1* (*float64xm1\_t a*, long *b*, unsigned int *gvl*)
- *float64xm2\_t vslideupvx\_float64xm2* (*float64xm2\_t a*, long *b*, unsigned int *gvl*)
- *float64xm4\_t vslideupvx\_float64xm4* (*float64xm4\_t a*, long *b*, unsigned int *gvl*)
- *float64xm8\_t vslideupvx\_float64xm8* (*float64xm8\_t a*, long *b*, unsigned int *gvl*)
- *int16xm1\_t vslideupvx\_int16xm1* (*int16xm1\_t a*, long *b*, unsigned int *gvl*)
- *int16xm2\_t vslideupvx\_int16xm2* (*int16xm2\_t a*, long *b*, unsigned int *gvl*)
- *int16xm4\_t vslideupvx\_int16xm4* (*int16xm4\_t a*, long *b*, unsigned int *gvl*)
- *int16xm8\_t vslideupvx\_int16xm8* (*int16xm8\_t a*, long *b*, unsigned int *gvl*)
- *int32xm1\_t vslideupvx\_int32xm1* (*int32xm1\_t a*, long *b*, unsigned int *gvl*)
- *int32xm2\_t vslideupvx\_int32xm2* (*int32xm2\_t a*, long *b*, unsigned int *gvl*)
- *int32xm4\_t vslideupvx\_int32xm4* (*int32xm4\_t a*, long *b*, unsigned int *gvl*)
- *int32xm8\_t vslideupvx\_int32xm8* (*int32xm8\_t a*, long *b*, unsigned int *gvl*)
- *int64xm1\_t vslideupvx\_int64xm1* (*int64xm1\_t a*, long *b*, unsigned int *gvl*)
- *int64xm2\_t vslideupvx\_int64xm2* (*int64xm2\_t a*, long *b*, unsigned int *gvl*)
- *int64xm4\_t vslideupvx\_int64xm4* (*int64xm4\_t a*, long *b*, unsigned int *gvl*)
- *int64xm8\_t vslideupvx\_int64xm8* (*int64xm8\_t a*, long *b*, unsigned int *gvl*)
- *int8xm1\_t vslideupvx\_int8xm1* (*int8xm1\_t a*, long *b*, unsigned int *gvl*)
- *int8xm2\_t vslideupvx\_int8xm2* (*int8xm2\_t a*, long *b*, unsigned int *gvl*)
- *int8xm4\_t vslideupvx\_int8xm4* (*int8xm4\_t a*, long *b*, unsigned int *gvl*)
- *int8xm8\_t vslideupvx\_int8xm8* (*int8xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm1\_t vslideupvx\_uint16xm1* (*uint16xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm2\_t vslideupvx\_uint16xm2* (*uint16xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm4\_t vslideupvx\_uint16xm4* (*uint16xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint16xm8\_t vslideupvx\_uint16xm8* (*uint16xm8\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm1\_t vslideupvx\_uint32xm1* (*uint32xm1\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm2\_t vslideupvx\_uint32xm2* (*uint32xm2\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm4\_t vslideupvx\_uint32xm4* (*uint32xm4\_t a*, long *b*, unsigned int *gvl*)
- *uint32xm8\_t vslideupvx\_uint32xm8* (*uint32xm8\_t a*, long *b*, unsigned int *gvl*)

- `uint64xm1_t vslideupvx_uint64xm1 (uint64xm1_t a, long b, unsigned int gvl)`
- `uint64xm2_t vslideupvx_uint64xm2 (uint64xm2_t a, long b, unsigned int gvl)`
- `uint64xm4_t vslideupvx_uint64xm4 (uint64xm4_t a, long b, unsigned int gvl)`
- `uint64xm8_t vslideupvx_uint64xm8 (uint64xm8_t a, long b, unsigned int gvl)`
- `uint8xm1_t vslideupvx_uint8xm1 (uint8xm1_t a, long b, unsigned int gvl)`
- `uint8xm2_t vslideupvx_uint8xm2 (uint8xm2_t a, long b, unsigned int gvl)`
- `uint8xm4_t vslideupvx_uint8xm4 (uint8xm4_t a, long b, unsigned int gvl)`
- `uint8xm8_t vslideupvx_uint8xm8 (uint8xm8_t a, long b, unsigned int gvl)`

**Operation:**

```
>>> for element = b to gvl - 1
      if element - b < 0
        result[element] = result[element]
      else
        result[element] = a[element - b]
      result[gvl : VLMAX] = 0
```

**Masked prototypes:**

- `float16xm1_t vslideupvx_mask_float16xm1 (float16xm1_t merge, float16xm1_t a, long b, e16xm1_t mask, unsigned int gvl)`
- `float16xm2_t vslideupvx_mask_float16xm2 (float16xm2_t merge, float16xm2_t a, long b, e16xm2_t mask, unsigned int gvl)`
- `float16xm4_t vslideupvx_mask_float16xm4 (float16xm4_t merge, float16xm4_t a, long b, e16xm4_t mask, unsigned int gvl)`
- `float16xm8_t vslideupvx_mask_float16xm8 (float16xm8_t merge, float16xm8_t a, long b, e16xm8_t mask, unsigned int gvl)`
- `float32xm1_t vslideupvx_mask_float32xm1 (float32xm1_t merge, float32xm1_t a, long b, e32xm1_t mask, unsigned int gvl)`
- `float32xm2_t vslideupvx_mask_float32xm2 (float32xm2_t merge, float32xm2_t a, long b, e32xm2_t mask, unsigned int gvl)`
- `float32xm4_t vslideupvx_mask_float32xm4 (float32xm4_t merge, float32xm4_t a, long b, e32xm4_t mask, unsigned int gvl)`
- `float32xm8_t vslideupvx_mask_float32xm8 (float32xm8_t merge, float32xm8_t a, long b, e32xm8_t mask, unsigned int gvl)`
- `float64xm1_t vslideupvx_mask_float64xm1 (float64xm1_t merge, float64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `float64xm2_t vslideupvx_mask_float64xm2 (float64xm2_t merge, float64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `float64xm4_t vslideupvx_mask_float64xm4 (float64xm4_t merge, float64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `float64xm8_t vslideupvx_mask_float64xm8 (float64xm8_t merge, float64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int16xm1_t vslideupvx_mask_int16xm1 (int16xm1_t merge, int16xm1_t a, long b, e16xm1_t mask, unsigned int gvl)`
- `int16xm2_t vslideupvx_mask_int16xm2 (int16xm2_t merge, int16xm2_t a, long b, e16xm2_t mask, unsigned int gvl)`

- `int16xm4_t vslideupvx_mask_int16xm4 (int16xm4_t merge, int16xm4_t a, long b, e16xm4_t mask, unsigned int gvl)`
- `int16xm8_t vslideupvx_mask_int16xm8 (int16xm8_t merge, int16xm8_t a, long b, e16xm8_t mask, unsigned int gvl)`
- `int32xm1_t vslideupvx_mask_int32xm1 (int32xm1_t merge, int32xm1_t a, long b, e32xm1_t mask, unsigned int gvl)`
- `int32xm2_t vslideupvx_mask_int32xm2 (int32xm2_t merge, int32xm2_t a, long b, e32xm2_t mask, unsigned int gvl)`
- `int32xm4_t vslideupvx_mask_int32xm4 (int32xm4_t merge, int32xm4_t a, long b, e32xm4_t mask, unsigned int gvl)`
- `int32xm8_t vslideupvx_mask_int32xm8 (int32xm8_t merge, int32xm8_t a, long b, e32xm8_t mask, unsigned int gvl)`
- `int64xm1_t vslideupvx_mask_int64xm1 (int64xm1_t merge, int64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`
- `int64xm2_t vslideupvx_mask_int64xm2 (int64xm2_t merge, int64xm2_t a, long b, e64xm2_t mask, unsigned int gvl)`
- `int64xm4_t vslideupvx_mask_int64xm4 (int64xm4_t merge, int64xm4_t a, long b, e64xm4_t mask, unsigned int gvl)`
- `int64xm8_t vslideupvx_mask_int64xm8 (int64xm8_t merge, int64xm8_t a, long b, e64xm8_t mask, unsigned int gvl)`
- `int8xm1_t vslideupvx_mask_int8xm1 (int8xm1_t merge, int8xm1_t a, long b, e8xm1_t mask, unsigned int gvl)`
- `int8xm2_t vslideupvx_mask_int8xm2 (int8xm2_t merge, int8xm2_t a, long b, e8xm2_t mask, unsigned int gvl)`
- `int8xm4_t vslideupvx_mask_int8xm4 (int8xm4_t merge, int8xm4_t a, long b, e8xm4_t mask, unsigned int gvl)`
- `int8xm8_t vslideupvx_mask_int8xm8 (int8xm8_t merge, int8xm8_t a, long b, e8xm8_t mask, unsigned int gvl)`
- `uint16xm1_t vslideupvx_mask_uint16xm1 (uint16xm1_t merge, uint16xm1_t a, long b, e16xm1_t mask, unsigned int gvl)`
- `uint16xm2_t vslideupvx_mask_uint16xm2 (uint16xm2_t merge, uint16xm2_t a, long b, e16xm2_t mask, unsigned int gvl)`
- `uint16xm4_t vslideupvx_mask_uint16xm4 (uint16xm4_t merge, uint16xm4_t a, long b, e16xm4_t mask, unsigned int gvl)`
- `uint16xm8_t vslideupvx_mask_uint16xm8 (uint16xm8_t merge, uint16xm8_t a, long b, e16xm8_t mask, unsigned int gvl)`
- `uint32xm1_t vslideupvx_mask_uint32xm1 (uint32xm1_t merge, uint32xm1_t a, long b, e32xm1_t mask, unsigned int gvl)`
- `uint32xm2_t vslideupvx_mask_uint32xm2 (uint32xm2_t merge, uint32xm2_t a, long b, e32xm2_t mask, unsigned int gvl)`
- `uint32xm4_t vslideupvx_mask_uint32xm4 (uint32xm4_t merge, uint32xm4_t a, long b, e32xm4_t mask, unsigned int gvl)`
- `uint32xm8_t vslideupvx_mask_uint32xm8 (uint32xm8_t merge, uint32xm8_t a, long b, e32xm8_t mask, unsigned int gvl)`
- `uint64xm1_t vslideupvx_mask_uint64xm1 (uint64xm1_t merge, uint64xm1_t a, long b, e64xm1_t mask, unsigned int gvl)`

- `uint64xm2_t vslideupvx_mask_uint64xm2` (`uint64xm2_t merge`, `uint64xm2_t a`, `long b`, `e64xm2_t mask`, unsigned int `gvl`)
- `uint64xm4_t vslideupvx_mask_uint64xm4` (`uint64xm4_t merge`, `uint64xm4_t a`, `long b`, `e64xm4_t mask`, unsigned int `gvl`)
- `uint64xm8_t vslideupvx_mask_uint64xm8` (`uint64xm8_t merge`, `uint64xm8_t a`, `long b`, `e64xm8_t mask`, unsigned int `gvl`)
- `uint8xm1_t vslideupvx_mask_uint8xm1` (`uint8xm1_t merge`, `uint8xm1_t a`, `long b`, `e8xm1_t mask`, unsigned int `gvl`)
- `uint8xm2_t vslideupvx_mask_uint8xm2` (`uint8xm2_t merge`, `uint8xm2_t a`, `long b`, `e8xm2_t mask`, unsigned int `gvl`)
- `uint8xm4_t vslideupvx_mask_uint8xm4` (`uint8xm4_t merge`, `uint8xm4_t a`, `long b`, `e8xm4_t mask`, unsigned int `gvl`)
- `uint8xm8_t vslideupvx_mask_uint8xm8` (`uint8xm8_t merge`, `uint8xm8_t a`, `long b`, `e8xm8_t mask`, unsigned int `gvl`)

**Masked operation:**

```
>>> for element = 0 to gvl - 1
    if mask[element] and element - b >= 0 then
        result[element] = a[element - b]
    else
        result[element] = merge[element]
result[gvl : VLMAX] = 0
```

THEAD

**EXAMPLES**

The following are provided to help explain the vector ISA.

### 3.1 Vector-vector add

```
#include <riscv-vector.h>

void vv_add_int32(int number, int *a, int *b, int *c) {

    int32xm2_t va;
    int32xm2_t vb;
    int32xm2_t vc;

    unsigned int gvl = 0;

    for (int i = 0; i < number;) {

        gvl = vsetvli(number - i, RVV_E32, RVV_M2);

        va = vlev_int32xm2(&a[i], gvl);
        vb = vlev_int32xm2(&b[i], gvl);
        vc = vaddvv_int32xm2(va, vb, gvl);
        vsev_int32xm2(&c[i], vc, gvl);

        i += gvl;
    }
}
```

### 3.2 Vector-vector add with inner data type

```
#include <riscv-vector.h>

void wvv_add_int8 (int8xm1_t a, int8xm1_t b, unsigned int gvl,
                  int16xm1_t *oa, int16xm1_t *ob) {
```

(continues on next page)



(continued from previous page)

```
int16xm2_u rv;  
rv.v = vwaddvv_int16xm2_int8xm1 (a, b, gvl);  
*oa = rv.m1[0];  
*ob = rv.m1[1];  
}
```

THE END

## APPENDIX A: FCSR

Table 1: fcsr layout

Bits	Name	Description
10:9	vxrm	Fixed-point rounding mode
8	vxsat	Fixed-point accrued saturation flag
7:5	frm	Floating-point rounding mode
4:0	fflags	Floating-point accrued exception flags

### 4.1 Vector Floating Rounding Mode Register frm

Floating-point operations use either a static rounding mode encoded in the instruction, or a dynamic rounding mode held in frm. Rounding modes are encoded as shown below. A value of 111 in the instruction's rm field selects the dynamic rounding mode held in frm. If frm is set to an invalid value (101–111), any subsequent attempt to execute a floating-point operation with a dynamic rounding mode will cause an illegal instruction trap. Some instructions that have the rm field are nevertheless unaffected by the rounding mode; they should have their rm field set to RNE (000).

Table 2: Rounding mode encoding

Rounding Mode	Mnemonic	Meaning
000	RNE	Round to Nearest, ties to Even
001	RTZ	Round towards Zero
010	RDN	Round Down (towards -infinity)
011	RUP	Round Up (towards +infinity)
100	RMM	Round to Nearest, ties to Max Magnitude
101		Invalid. Reserved for future use.
110		Invalid. Reserved for future use.
111		In instruction's rm field, selects dynamic rounding mode; In Rounding Mode register, Invalid.

### 4.2 Vector Fixed-Point Rounding Mode Register vxrm

The vector fixed-point rounding-mode register holds a two-bit read-write rounding-mode field. The vector fixed-point rounding-mode is given a separate CSR address to allow independent access, but is also reflected as a field in the upper bits of fcsr. Systems without floating-point must add fcsr when adding the vector extension.

The fixed-point rounding algorithm is specified as follows. Suppose the pre-rounding result is  $v$ , and  $d$  bits of that result are to be rounded off. Then the rounded result is  $(v \gg d) + r$ , where  $r$  depends on the rounding mode as specified in the following table.

Table 3: vxrm encoding

Bits [1:0]	Abbreviation	Rounding Mode	Rounding increment, $r$
0:0	rnu	round-to-nearest-up (add +0.5 LSB)	$v[d-1]$
0:1	rne	round-to-nearest-even	$v[d-1] \ \& \ (v[d-2:0] \neq 0 \mid v[d])$
1:0	rdn	round-down (truncate)	0
1:1	rod	round-to-odd (OR bits into LSB, aka “jam”)	$!v[d] \ \& \ v[d-1:0] \neq 0$

The rounding function:

$\text{roundoff}(v, d) = (v \gg d) + r$

## E

e16xm1\_t (C type), 2  
e16xm2\_t (C type), 2  
e16xm4\_t (C type), 2  
e16xm8\_t (C type), 2  
e32xm1\_t (C type), 2  
e32xm2\_t (C type), 2  
e32xm4\_t (C type), 2  
e32xm8\_t (C type), 2  
e64xm1\_t (C type), 2  
e64xm2\_t (C type), 2  
e64xm4\_t (C type), 2  
e64xm8\_t (C type), 2  
e8xm1\_t (C type), 2  
e8xm2\_t (C type), 2  
e8xm4\_t (C type), 2  
e8xm8\_t (C type), 2

## F

float16xm1\_t (C type), 2  
float16xm2\_t (C type), 2  
float16xm4\_t (C type), 2  
float16xm8\_t (C type), 2  
float32xm1\_t (C type), 2  
float32xm2\_t (C type), 2  
float32xm4\_t (C type), 2  
float32xm8\_t (C type), 2  
float64xm1\_t (C type), 2  
float64xm2\_t (C type), 2  
float64xm4\_t (C type), 2  
float64xm8\_t (C type), 2

## I

int16xm1\_t (C type), 2  
int16xm2\_t (C type), 2  
int16xm4\_t (C type), 2  
int16xm8\_t (C type), 2  
int32xm1\_t (C type), 2  
int32xm2\_t (C type), 2  
int32xm4\_t (C type), 2  
int32xm8\_t (C type), 2  
int64xm1\_t (C type), 2

int64xm2\_t (C type), 2  
int64xm4\_t (C type), 2  
int64xm8\_t (C type), 2  
int8xm1\_t (C type), 2  
int8xm2\_t (C type), 2  
int8xm4\_t (C type), 2  
int8xm8\_t (C type), 2

## U

uint16xm1\_t (C type), 2  
uint16xm2\_t (C type), 2  
uint16xm4\_t (C type), 2  
uint16xm8\_t (C type), 2  
uint32xm1\_t (C type), 2  
uint32xm2\_t (C type), 2  
uint32xm4\_t (C type), 2  
uint32xm8\_t (C type), 2  
uint64xm1\_t (C type), 2  
uint64xm2\_t (C type), 2  
uint64xm4\_t (C type), 2  
uint64xm8\_t (C type), 2  
uint8xm1\_t (C type), 2  
uint8xm2\_t (C type), 2  
uint8xm4\_t (C type), 2  
uint8xm8\_t (C type), 2

## V

vaaddvi\_int16xm1 (C function), 158  
vaaddvi\_int16xm2 (C function), 158  
vaaddvi\_int16xm4 (C function), 158  
vaaddvi\_int16xm8 (C function), 158  
vaaddvi\_int32xm1 (C function), 158  
vaaddvi\_int32xm2 (C function), 158  
vaaddvi\_int32xm4 (C function), 158  
vaaddvi\_int32xm8 (C function), 158  
vaaddvi\_int64xm1 (C function), 158  
vaaddvi\_int64xm2 (C function), 158  
vaaddvi\_int64xm4 (C function), 158  
vaaddvi\_int64xm8 (C function), 158  
vaaddvi\_int8xm1 (C function), 158  
vaaddvi\_int8xm2 (C function), 158  
vaaddvi\_int8xm4 (C function), 158







vaddvi\_mask\_uint32xm8 (C function), 169  
vaddvi\_mask\_uint64xm1 (C function), 169  
vaddvi\_mask\_uint64xm2 (C function), 169  
vaddvi\_mask\_uint64xm4 (C function), 169  
vaddvi\_mask\_uint64xm8 (C function), 169  
vaddvi\_mask\_uint8xm1 (C function), 169  
vaddvi\_mask\_uint8xm2 (C function), 169  
vaddvi\_mask\_uint8xm4 (C function), 169  
vaddvi\_mask\_uint8xm8 (C function), 169  
vaddvi\_uint16xm1 (C function), 167  
vaddvi\_uint16xm2 (C function), 167  
vaddvi\_uint16xm4 (C function), 167  
vaddvi\_uint16xm8 (C function), 167  
vaddvi\_uint32xm1 (C function), 167  
vaddvi\_uint32xm2 (C function), 167  
vaddvi\_uint32xm4 (C function), 167  
vaddvi\_uint32xm8 (C function), 167  
vaddvi\_uint64xm1 (C function), 168  
vaddvi\_uint64xm2 (C function), 168  
vaddvi\_uint64xm4 (C function), 168  
vaddvi\_uint64xm8 (C function), 168  
vaddvi\_uint8xm1 (C function), 168  
vaddvi\_uint8xm2 (C function), 168  
vaddvi\_uint8xm4 (C function), 168  
vaddvi\_uint8xm8 (C function), 168  
vaddvv\_int16xm1 (C function), 170  
vaddvv\_int16xm2 (C function), 170  
vaddvv\_int16xm4 (C function), 170  
vaddvv\_int16xm8 (C function), 170  
vaddvv\_int32xm1 (C function), 170  
vaddvv\_int32xm2 (C function), 170  
vaddvv\_int32xm4 (C function), 170  
vaddvv\_int32xm8 (C function), 170  
vaddvv\_int64xm1 (C function), 170  
vaddvv\_int64xm2 (C function), 170  
vaddvv\_int64xm4 (C function), 170  
vaddvv\_int64xm8 (C function), 170  
vaddvv\_int8xm1 (C function), 170  
vaddvv\_int8xm2 (C function), 170  
vaddvv\_int8xm4 (C function), 170  
vaddvv\_int8xm8 (C function), 170  
vaddvv\_mask\_int16xm1 (C function), 171  
vaddvv\_mask\_int16xm2 (C function), 171  
vaddvv\_mask\_int16xm4 (C function), 171  
vaddvv\_mask\_int16xm8 (C function), 171  
vaddvv\_mask\_int32xm1 (C function), 171  
vaddvv\_mask\_int32xm2 (C function), 171  
vaddvv\_mask\_int32xm4 (C function), 171  
vaddvv\_mask\_int32xm8 (C function), 171  
vaddvv\_mask\_int64xm1 (C function), 171  
vaddvv\_mask\_int64xm2 (C function), 171  
vaddvv\_mask\_int64xm4 (C function), 171  
vaddvv\_mask\_int64xm8 (C function), 171  
vaddvv\_mask\_int8xm1 (C function), 171  
vaddvv\_mask\_int8xm2 (C function), 171  
vaddvv\_mask\_int8xm4 (C function), 171  
vaddvv\_mask\_int8xm8 (C function), 171  
vaddvv\_mask\_uint16xm1 (C function), 171  
vaddvv\_mask\_uint16xm2 (C function), 171  
vaddvv\_mask\_uint16xm4 (C function), 171  
vaddvv\_mask\_uint16xm8 (C function), 171  
vaddvv\_mask\_uint32xm1 (C function), 171  
vaddvv\_mask\_uint32xm2 (C function), 171  
vaddvv\_mask\_uint32xm4 (C function), 172  
vaddvv\_mask\_uint32xm8 (C function), 172  
vaddvv\_mask\_uint64xm1 (C function), 172  
vaddvv\_mask\_uint64xm2 (C function), 172  
vaddvv\_mask\_uint64xm4 (C function), 172  
vaddvv\_mask\_uint64xm8 (C function), 172  
vaddvv\_mask\_uint8xm1 (C function), 172  
vaddvv\_mask\_uint8xm2 (C function), 172  
vaddvv\_mask\_uint8xm4 (C function), 172  
vaddvv\_mask\_uint8xm8 (C function), 172  
vaddvv\_uint16xm1 (C function), 170  
vaddvv\_uint16xm2 (C function), 170  
vaddvv\_uint16xm4 (C function), 170  
vaddvv\_uint16xm8 (C function), 170  
vaddvv\_uint32xm1 (C function), 170  
vaddvv\_uint32xm2 (C function), 170  
vaddvv\_uint32xm4 (C function), 170  
vaddvv\_uint32xm8 (C function), 170  
vaddvv\_uint64xm1 (C function), 170  
vaddvv\_uint64xm2 (C function), 170  
vaddvv\_uint64xm4 (C function), 170  
vaddvv\_uint64xm8 (C function), 170  
vaddvv\_uint8xm1 (C function), 170  
vaddvv\_uint8xm2 (C function), 170  
vaddvv\_uint8xm4 (C function), 170  
vaddvv\_uint8xm8 (C function), 170  
vaddvx\_int16xm1 (C function), 172  
vaddvx\_int16xm2 (C function), 172  
vaddvx\_int16xm4 (C function), 172  
vaddvx\_int16xm8 (C function), 172  
vaddvx\_int32xm1 (C function), 172  
vaddvx\_int32xm2 (C function), 172  
vaddvx\_int32xm4 (C function), 172  
vaddvx\_int32xm8 (C function), 172  
vaddvx\_int64xm1 (C function), 172  
vaddvx\_int64xm2 (C function), 172  
vaddvx\_int64xm4 (C function), 172  
vaddvx\_int64xm8 (C function), 173  
vaddvx\_int8xm1 (C function), 173  
vaddvx\_int8xm2 (C function), 173  
vaddvx\_int8xm4 (C function), 173  
vaddvx\_int8xm8 (C function), 173  
vaddvx\_mask\_int16xm1 (C function), 173  
vaddvx\_mask\_int16xm2 (C function), 173  
vaddvx\_mask\_int16xm4 (C function), 173

vaddvx\_mask\_int16xm8 (C function), 173  
 vaddvx\_mask\_int32xm1 (C function), 173  
 vaddvx\_mask\_int32xm2 (C function), 173  
 vaddvx\_mask\_int32xm4 (C function), 173  
 vaddvx\_mask\_int32xm8 (C function), 174  
 vaddvx\_mask\_int64xm1 (C function), 174  
 vaddvx\_mask\_int64xm2 (C function), 174  
 vaddvx\_mask\_int64xm4 (C function), 174  
 vaddvx\_mask\_int64xm8 (C function), 174  
 vaddvx\_mask\_int8xm1 (C function), 174  
 vaddvx\_mask\_int8xm2 (C function), 174  
 vaddvx\_mask\_int8xm4 (C function), 174  
 vaddvx\_mask\_int8xm8 (C function), 174  
 vaddvx\_mask\_uint16xm1 (C function), 174  
 vaddvx\_mask\_uint16xm2 (C function), 174  
 vaddvx\_mask\_uint16xm4 (C function), 174  
 vaddvx\_mask\_uint16xm8 (C function), 174  
 vaddvx\_mask\_uint32xm1 (C function), 174  
 vaddvx\_mask\_uint32xm2 (C function), 174  
 vaddvx\_mask\_uint32xm4 (C function), 174  
 vaddvx\_mask\_uint32xm8 (C function), 174  
 vaddvx\_mask\_uint64xm1 (C function), 174  
 vaddvx\_mask\_uint64xm2 (C function), 174  
 vaddvx\_mask\_uint64xm4 (C function), 174  
 vaddvx\_mask\_uint64xm8 (C function), 174  
 vaddvx\_mask\_uint8xm1 (C function), 174  
 vaddvx\_mask\_uint8xm2 (C function), 174  
 vaddvx\_mask\_uint8xm4 (C function), 175  
 vaddvx\_mask\_uint8xm8 (C function), 175  
 vaddvx\_uint16xm1 (C function), 173  
 vaddvx\_uint16xm2 (C function), 173  
 vaddvx\_uint16xm4 (C function), 173  
 vaddvx\_uint16xm8 (C function), 173  
 vaddvx\_uint32xm1 (C function), 173  
 vaddvx\_uint32xm2 (C function), 173  
 vaddvx\_uint32xm4 (C function), 173  
 vaddvx\_uint32xm8 (C function), 173  
 vaddvx\_uint64xm1 (C function), 173  
 vaddvx\_uint64xm2 (C function), 173  
 vaddvx\_uint64xm4 (C function), 173  
 vaddvx\_uint64xm8 (C function), 173  
 vaddvx\_uint8xm1 (C function), 173  
 vaddvx\_uint8xm2 (C function), 173  
 vaddvx\_uint8xm4 (C function), 173  
 vaddvx\_uint8xm8 (C function), 173  
 vandvi\_int16xm1 (C function), 6  
 vandvi\_int16xm2 (C function), 6  
 vandvi\_int16xm4 (C function), 6  
 vandvi\_int16xm8 (C function), 6  
 vandvi\_int32xm1 (C function), 6  
 vandvi\_int32xm2 (C function), 6  
 vandvi\_int32xm4 (C function), 6  
 vandvi\_int32xm8 (C function), 6  
 vandvi\_int64xm1 (C function), 6  
 vandvi\_int64xm2 (C function), 6  
 vandvi\_int64xm4 (C function), 6  
 vandvi\_int64xm8 (C function), 6  
 vandvi\_int8xm1 (C function), 6  
 vandvi\_int8xm2 (C function), 6  
 vandvi\_int8xm4 (C function), 6  
 vandvi\_int8xm8 (C function), 6  
 vandvi\_mask\_int16xm1 (C function), 6  
 vandvi\_mask\_int16xm2 (C function), 7  
 vandvi\_mask\_int16xm4 (C function), 7  
 vandvi\_mask\_int16xm8 (C function), 7  
 vandvi\_mask\_int32xm1 (C function), 7  
 vandvi\_mask\_int32xm2 (C function), 7  
 vandvi\_mask\_int32xm4 (C function), 7  
 vandvi\_mask\_int32xm8 (C function), 7  
 vandvi\_mask\_int64xm1 (C function), 7  
 vandvi\_mask\_int64xm2 (C function), 7  
 vandvi\_mask\_int64xm4 (C function), 7  
 vandvi\_mask\_int64xm8 (C function), 7  
 vandvi\_mask\_int8xm1 (C function), 7  
 vandvi\_mask\_int8xm2 (C function), 7  
 vandvi\_mask\_int8xm4 (C function), 7  
 vandvi\_mask\_int8xm8 (C function), 7  
 vandvi\_mask\_uint16xm1 (C function), 7  
 vandvi\_mask\_uint16xm2 (C function), 7  
 vandvi\_mask\_uint16xm4 (C function), 7  
 vandvi\_mask\_uint16xm8 (C function), 7  
 vandvi\_mask\_uint32xm1 (C function), 7  
 vandvi\_mask\_uint32xm2 (C function), 7  
 vandvi\_mask\_uint32xm4 (C function), 7  
 vandvi\_mask\_uint32xm8 (C function), 7  
 vandvi\_mask\_uint64xm1 (C function), 8  
 vandvi\_mask\_uint64xm2 (C function), 8  
 vandvi\_mask\_uint64xm4 (C function), 8  
 vandvi\_mask\_uint64xm8 (C function), 8  
 vandvi\_mask\_uint8xm1 (C function), 8  
 vandvi\_mask\_uint8xm2 (C function), 8  
 vandvi\_mask\_uint8xm4 (C function), 8  
 vandvi\_mask\_uint8xm8 (C function), 8  
 vandvi\_uint16xm1 (C function), 6  
 vandvi\_uint16xm2 (C function), 6  
 vandvi\_uint16xm4 (C function), 6  
 vandvi\_uint16xm8 (C function), 6  
 vandvi\_uint32xm1 (C function), 6  
 vandvi\_uint32xm2 (C function), 6  
 vandvi\_uint32xm4 (C function), 6  
 vandvi\_uint32xm8 (C function), 6  
 vandvi\_uint64xm1 (C function), 6  
 vandvi\_uint64xm2 (C function), 6  
 vandvi\_uint64xm4 (C function), 6  
 vandvi\_uint64xm8 (C function), 6  
 vandvi\_uint8xm1 (C function), 6  
 vandvi\_uint8xm2 (C function), 6  
 vandvi\_uint8xm4 (C function), 6

vandvi\_uint8xm8 (C function), 6  
vandvv\_int16xm1 (C function), 8  
vandvv\_int16xm2 (C function), 8  
vandvv\_int16xm4 (C function), 8  
vandvv\_int16xm8 (C function), 8  
vandvv\_int32xm1 (C function), 8  
vandvv\_int32xm2 (C function), 8  
vandvv\_int32xm4 (C function), 8  
vandvv\_int32xm8 (C function), 8  
vandvv\_int64xm1 (C function), 8  
vandvv\_int64xm2 (C function), 8  
vandvv\_int64xm4 (C function), 8  
vandvv\_int64xm8 (C function), 8  
vandvv\_int8xm1 (C function), 8  
vandvv\_int8xm2 (C function), 8  
vandvv\_int8xm4 (C function), 9  
vandvv\_int8xm8 (C function), 9  
vandvv\_mask\_int16xm1 (C function), 9  
vandvv\_mask\_int16xm2 (C function), 9  
vandvv\_mask\_int16xm4 (C function), 9  
vandvv\_mask\_int16xm8 (C function), 9  
vandvv\_mask\_int32xm1 (C function), 9  
vandvv\_mask\_int32xm2 (C function), 9  
vandvv\_mask\_int32xm4 (C function), 9  
vandvv\_mask\_int32xm8 (C function), 9  
vandvv\_mask\_int64xm1 (C function), 10  
vandvv\_mask\_int64xm2 (C function), 10  
vandvv\_mask\_int64xm4 (C function), 10  
vandvv\_mask\_int64xm8 (C function), 10  
vandvv\_mask\_int8xm1 (C function), 10  
vandvv\_mask\_int8xm2 (C function), 10  
vandvv\_mask\_int8xm4 (C function), 10  
vandvv\_mask\_int8xm8 (C function), 10  
vandvv\_mask\_uint16xm1 (C function), 10  
vandvv\_mask\_uint16xm2 (C function), 10  
vandvv\_mask\_uint16xm4 (C function), 10  
vandvv\_mask\_uint16xm8 (C function), 10  
vandvv\_mask\_uint32xm1 (C function), 10  
vandvv\_mask\_uint32xm2 (C function), 10  
vandvv\_mask\_uint32xm4 (C function), 10  
vandvv\_mask\_uint32xm8 (C function), 10  
vandvv\_mask\_uint64xm1 (C function), 10  
vandvv\_mask\_uint64xm2 (C function), 10  
vandvv\_mask\_uint64xm4 (C function), 10  
vandvv\_mask\_uint64xm8 (C function), 10  
vandvv\_mask\_uint8xm1 (C function), 10  
vandvv\_mask\_uint8xm2 (C function), 10  
vandvv\_mask\_uint8xm4 (C function), 10  
vandvv\_mask\_uint8xm8 (C function), 11  
vandvv\_uint16xm1 (C function), 9  
vandvv\_uint16xm2 (C function), 9  
vandvv\_uint16xm4 (C function), 9  
vandvv\_uint16xm8 (C function), 9  
vandvv\_uint32xm1 (C function), 9  
vandvv\_uint32xm2 (C function), 9  
vandvv\_uint32xm4 (C function), 9  
vandvv\_uint32xm8 (C function), 9  
vandvv\_uint64xm1 (C function), 9  
vandvv\_uint64xm2 (C function), 9  
vandvv\_uint64xm4 (C function), 9  
vandvv\_uint64xm8 (C function), 9  
vandvv\_uint8xm1 (C function), 9  
vandvv\_uint8xm2 (C function), 9  
vandvv\_uint8xm4 (C function), 9  
vandvv\_uint8xm8 (C function), 9  
vandvx\_int16xm1 (C function), 11  
vandvx\_int16xm2 (C function), 11  
vandvx\_int16xm4 (C function), 11  
vandvx\_int16xm8 (C function), 11  
vandvx\_int32xm1 (C function), 11  
vandvx\_int32xm2 (C function), 11  
vandvx\_int32xm4 (C function), 11  
vandvx\_int32xm8 (C function), 11  
vandvx\_int64xm1 (C function), 11  
vandvx\_int64xm2 (C function), 11  
vandvx\_int64xm4 (C function), 11  
vandvx\_int64xm8 (C function), 11  
vandvx\_int8xm1 (C function), 11  
vandvx\_int8xm2 (C function), 11  
vandvx\_int8xm4 (C function), 11  
vandvx\_int8xm8 (C function), 11  
vandvx\_mask\_int16xm1 (C function), 12  
vandvx\_mask\_int16xm2 (C function), 12  
vandvx\_mask\_int16xm4 (C function), 12  
vandvx\_mask\_int16xm8 (C function), 12  
vandvx\_mask\_int32xm1 (C function), 12  
vandvx\_mask\_int32xm2 (C function), 12  
vandvx\_mask\_int32xm4 (C function), 12  
vandvx\_mask\_int32xm8 (C function), 12  
vandvx\_mask\_int64xm1 (C function), 12  
vandvx\_mask\_int64xm2 (C function), 12  
vandvx\_mask\_int64xm4 (C function), 12  
vandvx\_mask\_int64xm8 (C function), 12  
vandvx\_mask\_int8xm1 (C function), 12  
vandvx\_mask\_int8xm2 (C function), 12  
vandvx\_mask\_int8xm4 (C function), 12  
vandvx\_mask\_int8xm8 (C function), 12  
vandvx\_mask\_uint16xm1 (C function), 13  
vandvx\_mask\_uint16xm2 (C function), 13  
vandvx\_mask\_uint16xm4 (C function), 13  
vandvx\_mask\_uint16xm8 (C function), 13  
vandvx\_mask\_uint32xm1 (C function), 13  
vandvx\_mask\_uint32xm2 (C function), 13  
vandvx\_mask\_uint32xm4 (C function), 13  
vandvx\_mask\_uint32xm8 (C function), 13  
vandvx\_mask\_uint64xm1 (C function), 13  
vandvx\_mask\_uint64xm2 (C function), 13  
vandvx\_mask\_uint64xm4 (C function), 13

vandvx\_mask\_uint64xm8 (C function), 13  
 vandvx\_mask\_uint8xm1 (C function), 13  
 vandvx\_mask\_uint8xm2 (C function), 13  
 vandvx\_mask\_uint8xm4 (C function), 13  
 vandvx\_mask\_uint8xm8 (C function), 13  
 vandvx\_uint16xm1 (C function), 11  
 vandvx\_uint16xm2 (C function), 11  
 vandvx\_uint16xm4 (C function), 11  
 vandvx\_uint16xm8 (C function), 11  
 vandvx\_uint32xm1 (C function), 11  
 vandvx\_uint32xm2 (C function), 11  
 vandvx\_uint32xm4 (C function), 11  
 vandvx\_uint32xm8 (C function), 11  
 vandvx\_uint64xm1 (C function), 11  
 vandvx\_uint64xm2 (C function), 12  
 vandvx\_uint64xm4 (C function), 12  
 vandvx\_uint64xm8 (C function), 12  
 vandvx\_uint8xm1 (C function), 12  
 vandvx\_uint8xm2 (C function), 12  
 vandvx\_uint8xm4 (C function), 12  
 vandvx\_uint8xm8 (C function), 12  
 vasubvv\_int16xm1 (C function), 175  
 vasubvv\_int16xm2 (C function), 175  
 vasubvv\_int16xm4 (C function), 175  
 vasubvv\_int16xm8 (C function), 175  
 vasubvv\_int32xm1 (C function), 175  
 vasubvv\_int32xm2 (C function), 175  
 vasubvv\_int32xm4 (C function), 175  
 vasubvv\_int32xm8 (C function), 175  
 vasubvv\_int64xm1 (C function), 175  
 vasubvv\_int64xm2 (C function), 175  
 vasubvv\_int64xm4 (C function), 175  
 vasubvv\_int64xm8 (C function), 175  
 vasubvv\_int8xm1 (C function), 175  
 vasubvv\_int8xm2 (C function), 175  
 vasubvv\_int8xm4 (C function), 175  
 vasubvv\_int8xm8 (C function), 175  
 vasubvv\_mask\_int16xm1 (C function), 175  
 vasubvv\_mask\_int16xm2 (C function), 176  
 vasubvv\_mask\_int16xm4 (C function), 176  
 vasubvv\_mask\_int16xm8 (C function), 176  
 vasubvv\_mask\_int32xm1 (C function), 176  
 vasubvv\_mask\_int32xm2 (C function), 176  
 vasubvv\_mask\_int32xm4 (C function), 176  
 vasubvv\_mask\_int32xm8 (C function), 176  
 vasubvv\_mask\_int64xm1 (C function), 176  
 vasubvv\_mask\_int64xm2 (C function), 176  
 vasubvv\_mask\_int64xm4 (C function), 176  
 vasubvv\_mask\_int64xm8 (C function), 176  
 vasubvv\_mask\_int8xm1 (C function), 176  
 vasubvv\_mask\_int8xm2 (C function), 176  
 vasubvv\_mask\_int8xm4 (C function), 176  
 vasubvv\_mask\_int8xm8 (C function), 176  
 vasubvx\_int16xm1 (C function), 176  
 vasubvx\_int16xm2 (C function), 176  
 vasubvx\_int16xm4 (C function), 176  
 vasubvx\_int16xm8 (C function), 177  
 vasubvx\_int32xm1 (C function), 177  
 vasubvx\_int32xm2 (C function), 177  
 vasubvx\_int32xm4 (C function), 177  
 vasubvx\_int32xm8 (C function), 177  
 vasubvx\_int64xm1 (C function), 177  
 vasubvx\_int64xm2 (C function), 177  
 vasubvx\_int64xm4 (C function), 177  
 vasubvx\_int64xm8 (C function), 177  
 vasubvx\_int8xm1 (C function), 177  
 vasubvx\_int8xm2 (C function), 177  
 vasubvx\_int8xm4 (C function), 177  
 vasubvx\_int8xm8 (C function), 177  
 vasubvx\_mask\_int16xm1 (C function), 177  
 vasubvx\_mask\_int16xm2 (C function), 177  
 vasubvx\_mask\_int16xm4 (C function), 177  
 vasubvx\_mask\_int16xm8 (C function), 177  
 vasubvx\_mask\_int32xm1 (C function), 177  
 vasubvx\_mask\_int32xm2 (C function), 177  
 vasubvx\_mask\_int32xm4 (C function), 177  
 vasubvx\_mask\_int32xm8 (C function), 177  
 vasubvx\_mask\_int64xm1 (C function), 177  
 vasubvx\_mask\_int64xm2 (C function), 177  
 vasubvx\_mask\_int64xm4 (C function), 177  
 vasubvx\_mask\_int64xm8 (C function), 177  
 vasubvx\_mask\_int8xm1 (C function), 178  
 vasubvx\_mask\_int8xm2 (C function), 178  
 vasubvx\_mask\_int8xm4 (C function), 178  
 vasubvx\_mask\_int8xm8 (C function), 178  
 vcompressvm\_int16xm1\_e16xm1 (C function), 814  
 vcompressvm\_int16xm2\_e16xm2 (C function), 814  
 vcompressvm\_int16xm4\_e16xm4 (C function), 814  
 vcompressvm\_int16xm8\_e16xm8 (C function), 814  
 vcompressvm\_int32xm1\_e32xm1 (C function), 814  
 vcompressvm\_int32xm2\_e32xm2 (C function), 814  
 vcompressvm\_int32xm4\_e32xm4 (C function), 814  
 vcompressvm\_int32xm8\_e32xm8 (C function), 814  
 vcompressvm\_int64xm1\_e64xm1 (C function), 814  
 vcompressvm\_int64xm2\_e64xm2 (C function), 814  
 vcompressvm\_int64xm4\_e64xm4 (C function), 814  
 vcompressvm\_int64xm8\_e64xm8 (C function), 814  
 vcompressvm\_int8xm1\_e8xm1 (C function), 815  
 vcompressvm\_int8xm2\_e8xm2 (C function), 815  
 vcompressvm\_int8xm4\_e8xm4 (C function), 815  
 vcompressvm\_int8xm8\_e8xm8 (C function), 815  
 vcompressvm\_uint16xm1\_e16xm1 (C function), 815  
 vcompressvm\_uint16xm2\_e16xm2 (C function), 815  
 vcompressvm\_uint16xm4\_e16xm4 (C function), 815  
 vcompressvm\_uint16xm8\_e16xm8 (C function), 815  
 vcompressvm\_uint32xm1\_e32xm1 (C function), 815  
 vcompressvm\_uint32xm2\_e32xm2 (C function), 815  
 vcompressvm\_uint32xm4\_e32xm4 (C function), 815



vcompressvm\_uint32xm8\_e32xm8 (C function), 815  
vcompressvm\_uint64xm1\_e64xm1 (C function), 815  
vcompressvm\_uint64xm2\_e64xm2 (C function), 815  
vcompressvm\_uint64xm4\_e64xm4 (C function), 815  
vcompressvm\_uint64xm8\_e64xm8 (C function), 815  
vcompressvm\_uint8xm1\_e8xm1 (C function), 815  
vcompressvm\_uint8xm2\_e8xm2 (C function), 815  
vcompressvm\_uint8xm4\_e8xm4 (C function), 815  
vcompressvm\_uint8xm8\_e8xm8 (C function), 815  
vdivuvv\_mask\_uint16xm1 (C function), 182  
vdivuvv\_mask\_uint16xm2 (C function), 182  
vdivuvv\_mask\_uint16xm4 (C function), 182  
vdivuvv\_mask\_uint16xm8 (C function), 182  
vdivuvv\_mask\_uint32xm1 (C function), 182  
vdivuvv\_mask\_uint32xm2 (C function), 182  
vdivuvv\_mask\_uint32xm4 (C function), 182  
vdivuvv\_mask\_uint32xm8 (C function), 182  
vdivuvv\_mask\_uint64xm1 (C function), 182  
vdivuvv\_mask\_uint64xm2 (C function), 182  
vdivuvv\_mask\_uint64xm4 (C function), 182  
vdivuvv\_mask\_uint64xm8 (C function), 182  
vdivuvv\_mask\_uint8xm1 (C function), 182  
vdivuvv\_mask\_uint8xm2 (C function), 182  
vdivuvv\_mask\_uint8xm4 (C function), 182  
vdivuvv\_mask\_uint8xm8 (C function), 182  
vdivuvv\_uint16xm1 (C function), 181  
vdivuvv\_uint16xm2 (C function), 181  
vdivuvv\_uint16xm4 (C function), 181  
vdivuvv\_uint16xm8 (C function), 181  
vdivuvv\_uint32xm1 (C function), 181  
vdivuvv\_uint32xm2 (C function), 181  
vdivuvv\_uint32xm4 (C function), 181  
vdivuvv\_uint32xm8 (C function), 181  
vdivuvv\_uint64xm1 (C function), 181  
vdivuvv\_uint64xm2 (C function), 181  
vdivuvv\_uint64xm4 (C function), 181  
vdivuvv\_uint64xm8 (C function), 181  
vdivuvv\_uint8xm1 (C function), 181  
vdivuvv\_uint8xm2 (C function), 181  
vdivuvv\_uint8xm4 (C function), 181  
vdivuvv\_uint8xm8 (C function), 181  
vdivuvx\_mask\_uint16xm1 (C function), 183  
vdivuvx\_mask\_uint16xm2 (C function), 183  
vdivuvx\_mask\_uint16xm4 (C function), 183  
vdivuvx\_mask\_uint16xm8 (C function), 183  
vdivuvx\_mask\_uint32xm1 (C function), 183  
vdivuvx\_mask\_uint32xm2 (C function), 183  
vdivuvx\_mask\_uint32xm4 (C function), 183  
vdivuvx\_mask\_uint32xm8 (C function), 183  
vdivuvx\_mask\_uint64xm1 (C function), 184  
vdivuvx\_mask\_uint64xm2 (C function), 184  
vdivuvx\_mask\_uint64xm4 (C function), 184  
vdivuvx\_mask\_uint64xm8 (C function), 184  
vdivuvx\_mask\_uint8xm1 (C function), 184  
vdivuvx\_mask\_uint8xm2 (C function), 184  
vdivuvx\_mask\_uint8xm4 (C function), 184  
vdivuvx\_mask\_uint8xm8 (C function), 184  
vdivuvx\_uint16xm1 (C function), 183  
vdivuvx\_uint16xm2 (C function), 183  
vdivuvx\_uint16xm4 (C function), 183  
vdivuvx\_uint16xm8 (C function), 183  
vdivuvx\_uint32xm1 (C function), 183  
vdivuvx\_uint32xm2 (C function), 183  
vdivuvx\_uint32xm4 (C function), 183  
vdivuvx\_uint32xm8 (C function), 183  
vdivuvx\_uint64xm1 (C function), 183  
vdivuvx\_uint64xm2 (C function), 183  
vdivuvx\_uint64xm4 (C function), 183  
vdivuvx\_uint64xm8 (C function), 183  
vdivuvx\_uint8xm1 (C function), 183  
vdivuvx\_uint8xm2 (C function), 183  
vdivuvx\_uint8xm4 (C function), 183  
vdivuvx\_uint8xm8 (C function), 183  
vdivvv\_int16xm1 (C function), 178  
vdivvv\_int16xm2 (C function), 178  
vdivvv\_int16xm4 (C function), 178  
vdivvv\_int16xm8 (C function), 178  
vdivvv\_int32xm1 (C function), 178  
vdivvv\_int32xm2 (C function), 178  
vdivvv\_int32xm4 (C function), 178  
vdivvv\_int32xm8 (C function), 178  
vdivvv\_int64xm1 (C function), 178  
vdivvv\_int64xm2 (C function), 178  
vdivvv\_int64xm4 (C function), 178  
vdivvv\_int64xm8 (C function), 178  
vdivvv\_int8xm1 (C function), 178  
vdivvv\_int8xm2 (C function), 178  
vdivvv\_int8xm4 (C function), 178  
vdivvv\_int8xm8 (C function), 178  
vdivvv\_mask\_int16xm1 (C function), 179  
vdivvv\_mask\_int16xm2 (C function), 179  
vdivvv\_mask\_int16xm4 (C function), 179  
vdivvv\_mask\_int16xm8 (C function), 179  
vdivvv\_mask\_int32xm1 (C function), 179  
vdivvv\_mask\_int32xm2 (C function), 179  
vdivvv\_mask\_int32xm4 (C function), 179  
vdivvv\_mask\_int32xm8 (C function), 179  
vdivvv\_mask\_int64xm1 (C function), 179  
vdivvv\_mask\_int64xm2 (C function), 179  
vdivvv\_mask\_int64xm4 (C function), 179  
vdivvv\_mask\_int64xm8 (C function), 179  
vdivvv\_mask\_int8xm1 (C function), 179  
vdivvv\_mask\_int8xm2 (C function), 179  
vdivvv\_mask\_int8xm4 (C function), 179  
vdivvv\_mask\_int8xm8 (C function), 179  
vdivvx\_int16xm1 (C function), 180  
vdivvx\_int16xm2 (C function), 180  
vdivvx\_int16xm4 (C function), 180

vdivvx\_int16xm8 (C function), 180  
 vdivvx\_int32xm1 (C function), 180  
 vdivvx\_int32xm2 (C function), 180  
 vdivvx\_int32xm4 (C function), 180  
 vdivvx\_int32xm8 (C function), 180  
 vdivvx\_int64xm1 (C function), 180  
 vdivvx\_int64xm2 (C function), 180  
 vdivvx\_int64xm4 (C function), 180  
 vdivvx\_int64xm8 (C function), 180  
 vdivvx\_int8xm1 (C function), 180  
 vdivvx\_int8xm2 (C function), 180  
 vdivvx\_int8xm4 (C function), 180  
 vdivvx\_int8xm8 (C function), 180  
 vdivvx\_mask\_int16xm1 (C function), 180  
 vdivvx\_mask\_int16xm2 (C function), 180  
 vdivvx\_mask\_int16xm4 (C function), 180  
 vdivvx\_mask\_int16xm8 (C function), 180  
 vdivvx\_mask\_int32xm1 (C function), 180  
 vdivvx\_mask\_int32xm2 (C function), 180  
 vdivvx\_mask\_int32xm4 (C function), 180  
 vdivvx\_mask\_int32xm8 (C function), 180  
 vdivvx\_mask\_int64xm1 (C function), 180  
 vdivvx\_mask\_int64xm2 (C function), 180  
 vdivvx\_mask\_int64xm4 (C function), 181  
 vdivvx\_mask\_int64xm8 (C function), 181  
 vdivvx\_mask\_int8xm1 (C function), 181  
 vdivvx\_mask\_int8xm2 (C function), 181  
 vdivvx\_mask\_int8xm4 (C function), 181  
 vdivvx\_mask\_int8xm8 (C function), 181  
 vdotvv\_int16xm1 (C function), 184  
 vdotvv\_int16xm2 (C function), 184  
 vdotvv\_int16xm4 (C function), 184  
 vdotvv\_int16xm8 (C function), 184  
 vdotvv\_int32xm1 (C function), 184  
 vdotvv\_int32xm2 (C function), 184  
 vdotvv\_int32xm4 (C function), 184  
 vdotvv\_int32xm8 (C function), 184  
 vdotvv\_int64xm1 (C function), 184  
 vdotvv\_int64xm2 (C function), 184  
 vdotvv\_int64xm4 (C function), 184  
 vdotvv\_int64xm8 (C function), 184  
 vdotvv\_int8xm1 (C function), 184  
 vdotvv\_int8xm2 (C function), 184  
 vdotvv\_int8xm4 (C function), 185  
 vdotvv\_int8xm8 (C function), 185  
 vdotvv\_mask\_int16xm1 (C function), 185  
 vdotvv\_mask\_int16xm2 (C function), 185  
 vdotvv\_mask\_int16xm4 (C function), 185  
 vdotvv\_mask\_int16xm8 (C function), 185  
 vdotvv\_mask\_int32xm1 (C function), 185  
 vdotvv\_mask\_int32xm2 (C function), 185  
 vdotvv\_mask\_int32xm4 (C function), 185  
 vdotvv\_mask\_int32xm8 (C function), 185  
 vdotvv\_mask\_int64xm1 (C function), 186

vdotvv\_mask\_int64xm2 (C function), 186  
 vdotvv\_mask\_int64xm4 (C function), 186  
 vdotvv\_mask\_int64xm8 (C function), 186  
 vdotvv\_mask\_int8xm1 (C function), 186  
 vdotvv\_mask\_int8xm2 (C function), 186  
 vdotvv\_mask\_int8xm4 (C function), 186  
 vdotvv\_mask\_int8xm8 (C function), 186  
 vdotvv\_mask\_uint16xm1 (C function), 186  
 vdotvv\_mask\_uint16xm2 (C function), 186  
 vdotvv\_mask\_uint16xm4 (C function), 186  
 vdotvv\_mask\_uint16xm8 (C function), 186  
 vdotvv\_mask\_uint32xm1 (C function), 186  
 vdotvv\_mask\_uint32xm2 (C function), 186  
 vdotvv\_mask\_uint32xm4 (C function), 186  
 vdotvv\_mask\_uint32xm8 (C function), 186  
 vdotvv\_mask\_uint64xm1 (C function), 186  
 vdotvv\_mask\_uint64xm2 (C function), 186  
 vdotvv\_mask\_uint64xm4 (C function), 186  
 vdotvv\_mask\_uint64xm8 (C function), 186  
 vdotvv\_mask\_uint8xm1 (C function), 186  
 vdotvv\_mask\_uint8xm2 (C function), 186  
 vdotvv\_mask\_uint8xm4 (C function), 186  
 vdotvv\_mask\_uint8xm8 (C function), 187  
 vdotvv\_uint16xm1 (C function), 185  
 vdotvv\_uint16xm2 (C function), 185  
 vdotvv\_uint16xm4 (C function), 185  
 vdotvv\_uint16xm8 (C function), 185  
 vdotvv\_uint32xm1 (C function), 185  
 vdotvv\_uint32xm2 (C function), 185  
 vdotvv\_uint32xm4 (C function), 185  
 vdotvv\_uint32xm8 (C function), 185  
 vdotvv\_uint64xm1 (C function), 185  
 vdotvv\_uint64xm2 (C function), 185  
 vdotvv\_uint64xm4 (C function), 185  
 vdotvv\_uint64xm8 (C function), 185  
 vdotvv\_uint8xm1 (C function), 185  
 vdotvv\_uint8xm2 (C function), 185  
 vdotvv\_uint8xm4 (C function), 185  
 vdotvv\_uint8xm8 (C function), 185  
 vextxv\_int16xm1 (C function), 815  
 vextxv\_int16xm2 (C function), 815  
 vextxv\_int16xm4 (C function), 815  
 vextxv\_int16xm8 (C function), 815  
 vextxv\_int32xm1 (C function), 815  
 vextxv\_int32xm2 (C function), 815  
 vextxv\_int32xm4 (C function), 816  
 vextxv\_int32xm8 (C function), 816  
 vextxv\_int64xm1 (C function), 816  
 vextxv\_int64xm2 (C function), 816  
 vextxv\_int64xm4 (C function), 816  
 vextxv\_int64xm8 (C function), 816  
 vextxv\_int8xm1 (C function), 816  
 vextxv\_int8xm2 (C function), 816  
 vextxv\_int8xm4 (C function), 816



vextxv\_int8xm8 (C function), 816  
 vextxv\_uint16xm1 (C function), 816  
 vextxv\_uint16xm2 (C function), 816  
 vextxv\_uint16xm4 (C function), 816  
 vextxv\_uint16xm8 (C function), 816  
 vextxv\_uint32xm1 (C function), 816  
 vextxv\_uint32xm2 (C function), 816  
 vextxv\_uint32xm4 (C function), 816  
 vextxv\_uint32xm8 (C function), 816  
 vextxv\_uint64xm1 (C function), 816  
 vextxv\_uint64xm2 (C function), 816  
 vextxv\_uint64xm4 (C function), 816  
 vextxv\_uint64xm8 (C function), 816  
 vextxv\_uint8xm1 (C function), 816  
 vextxv\_uint8xm2 (C function), 816  
 vextxv\_uint8xm4 (C function), 816  
 vextxv\_uint8xm8 (C function), 816  
 vfaddvf\_float16xm1 (C function), 64  
 vfaddvf\_float16xm2 (C function), 64  
 vfaddvf\_float16xm4 (C function), 64  
 vfaddvf\_float16xm8 (C function), 64  
 vfaddvf\_float32xm1 (C function), 64  
 vfaddvf\_float32xm2 (C function), 64  
 vfaddvf\_float32xm4 (C function), 64  
 vfaddvf\_float32xm8 (C function), 65  
 vfaddvf\_float64xm1 (C function), 65  
 vfaddvf\_float64xm2 (C function), 65  
 vfaddvf\_float64xm4 (C function), 65  
 vfaddvf\_float64xm8 (C function), 65  
 vfaddvf\_mask\_float16xm1 (C function), 65  
 vfaddvf\_mask\_float16xm2 (C function), 65  
 vfaddvf\_mask\_float16xm4 (C function), 65  
 vfaddvf\_mask\_float16xm8 (C function), 65  
 vfaddvf\_mask\_float32xm1 (C function), 65  
 vfaddvf\_mask\_float32xm2 (C function), 65  
 vfaddvf\_mask\_float32xm4 (C function), 65  
 vfaddvf\_mask\_float32xm8 (C function), 65  
 vfaddvf\_mask\_float64xm1 (C function), 65  
 vfaddvf\_mask\_float64xm2 (C function), 65  
 vfaddvf\_mask\_float64xm4 (C function), 65  
 vfaddvf\_mask\_float64xm8 (C function), 65  
 vfaddvv\_float16xm1 (C function), 66  
 vfaddvv\_float16xm2 (C function), 66  
 vfaddvv\_float16xm4 (C function), 66  
 vfaddvv\_float16xm8 (C function), 66  
 vfaddvv\_float32xm1 (C function), 66  
 vfaddvv\_float32xm2 (C function), 66  
 vfaddvv\_float32xm4 (C function), 66  
 vfaddvv\_float32xm8 (C function), 66  
 vfaddvv\_float64xm1 (C function), 66  
 vfaddvv\_float64xm2 (C function), 66  
 vfaddvv\_float64xm4 (C function), 66  
 vfaddvv\_float64xm8 (C function), 66  
 vfaddvv\_mask\_float16xm1 (C function), 66

vfaddvv\_mask\_float16xm2 (C function), 66  
 vfaddvv\_mask\_float16xm4 (C function), 66  
 vfaddvv\_mask\_float16xm8 (C function), 66  
 vfaddvv\_mask\_float32xm1 (C function), 66  
 vfaddvv\_mask\_float32xm2 (C function), 66  
 vfaddvv\_mask\_float32xm4 (C function), 66  
 vfaddvv\_mask\_float32xm8 (C function), 66  
 vfaddvv\_mask\_float64xm1 (C function), 66  
 vfaddvv\_mask\_float64xm2 (C function), 66  
 vfaddvv\_mask\_float64xm4 (C function), 67  
 vfaddvv\_mask\_float64xm8 (C function), 67  
 vfclassv\_mask\_uint16xm1\_float16xm1 (C function), 67  
 vfclassv\_mask\_uint16xm2\_float16xm2 (C function), 67  
 vfclassv\_mask\_uint16xm4\_float16xm4 (C function), 67  
 vfclassv\_mask\_uint16xm8\_float16xm8 (C function), 67  
 vfclassv\_mask\_uint32xm1\_float32xm1 (C function), 68  
 vfclassv\_mask\_uint32xm2\_float32xm2 (C function), 68  
 vfclassv\_mask\_uint32xm4\_float32xm4 (C function), 68  
 vfclassv\_mask\_uint32xm8\_float32xm8 (C function), 68  
 vfclassv\_mask\_uint64xm1\_float64xm1 (C function), 68  
 vfclassv\_mask\_uint64xm2\_float64xm2 (C function), 68  
 vfclassv\_mask\_uint64xm4\_float64xm4 (C function), 68  
 vfclassv\_mask\_uint64xm8\_float64xm8 (C function), 68  
 vfclassv\_uint16xm1\_float16xm1 (C function), 67  
 vfclassv\_uint16xm2\_float16xm2 (C function), 67  
 vfclassv\_uint16xm4\_float16xm4 (C function), 67  
 vfclassv\_uint16xm8\_float16xm8 (C function), 67  
 vfclassv\_uint32xm1\_float32xm1 (C function), 67  
 vfclassv\_uint32xm2\_float32xm2 (C function), 67  
 vfclassv\_uint32xm4\_float32xm4 (C function), 67  
 vfclassv\_uint32xm8\_float32xm8 (C function), 67  
 vfclassv\_uint64xm1\_float64xm1 (C function), 67  
 vfclassv\_uint64xm2\_float64xm2 (C function), 67  
 vfclassv\_uint64xm4\_float64xm4 (C function), 67  
 vfclassv\_uint64xm8\_float64xm8 (C function), 67  
 vfcvtfxuv\_float16xm1\_uint16xm1 (C function), 53  
 vfcvtfxuv\_float16xm2\_uint16xm2 (C function), 53  
 vfcvtfxuv\_float16xm4\_uint16xm4 (C function), 53  
 vfcvtfxuv\_float16xm8\_uint16xm8 (C function), 53  
 vfcvtfxuv\_float32xm1\_uint32xm1 (C function), 53  
 vfcvtfxuv\_float32xm2\_uint32xm2 (C function), 53  
 vfcvtfxuv\_float32xm4\_uint32xm4 (C function), 53  
 vfcvtfxuv\_float32xm8\_uint32xm8 (C function), 53  
 vfcvtfxuv\_float64xm1\_uint64xm1 (C function), 53  
 vfcvtfxuv\_float64xm2\_uint64xm2 (C function), 53  
 vfcvtfxuv\_float64xm4\_uint64xm4 (C function), 53  
 vfcvtfxuv\_float64xm8\_uint64xm8 (C function), 53  
 vfcvtfxuv\_mask\_float16xm1\_uint16xm1 (C function), 53  
 vfcvtfxuv\_mask\_float16xm2\_uint16xm2 (C function), 54  
 vfcvtfxuv\_mask\_float16xm4\_uint16xm4 (C function), 54  
 vfcvtfxuv\_mask\_float16xm8\_uint16xm8 (C function), 54  
 vfcvtfxuv\_mask\_float32xm1\_uint32xm1 (C function), 54  
 vfcvtfxuv\_mask\_float32xm2\_uint32xm2 (C function), 54  
 vfcvtfxuv\_mask\_float32xm4\_uint32xm4 (C function), 54

vfevtfxuv\_mask\_float32xm8\_uint32xm8 (C function), 54  
 vfevtfxuv\_mask\_float64xm1\_uint64xm1 (C function), 54  
 vfevtfxuv\_mask\_float64xm2\_uint64xm2 (C function), 54  
 vfevtfxuv\_mask\_float64xm4\_uint64xm4 (C function), 54  
 vfevtfxuv\_mask\_float64xm8\_uint64xm8 (C function), 54  
 vfevtfxv\_float16xm1\_int16xm1 (C function), 52  
 vfevtfxv\_float16xm2\_int16xm2 (C function), 52  
 vfevtfxv\_float16xm4\_int16xm4 (C function), 52  
 vfevtfxv\_float16xm8\_int16xm8 (C function), 52  
 vfevtfxv\_float32xm1\_int32xm1 (C function), 52  
 vfevtfxv\_float32xm2\_int32xm2 (C function), 52  
 vfevtfxv\_float32xm4\_int32xm4 (C function), 52  
 vfevtfxv\_float32xm8\_int32xm8 (C function), 52  
 vfevtfxv\_float64xm1\_int64xm1 (C function), 52  
 vfevtfxv\_float64xm2\_int64xm2 (C function), 52  
 vfevtfxv\_float64xm4\_int64xm4 (C function), 52  
 vfevtfxv\_float64xm8\_int64xm8 (C function), 52  
 vfevtfxv\_mask\_float16xm1\_int16xm1 (C function), 52  
 vfevtfxv\_mask\_float16xm2\_int16xm2 (C function), 52  
 vfevtfxv\_mask\_float16xm4\_int16xm4 (C function), 52  
 vfevtfxv\_mask\_float16xm8\_int16xm8 (C function), 52  
 vfevtfxv\_mask\_float32xm1\_int32xm1 (C function), 52  
 vfevtfxv\_mask\_float32xm2\_int32xm2 (C function), 52  
 vfevtfxv\_mask\_float32xm4\_int32xm4 (C function), 52  
 vfevtfxv\_mask\_float32xm8\_int32xm8 (C function), 53  
 vfevtfxv\_mask\_float64xm1\_int64xm1 (C function), 53  
 vfevtfxv\_mask\_float64xm2\_int64xm2 (C function), 53  
 vfevtfxv\_mask\_float64xm4\_int64xm4 (C function), 53  
 vfevtfxv\_mask\_float64xm8\_int64xm8 (C function), 53  
 vfevtxfv\_int16xm1\_float16xm1 (C function), 54  
 vfevtxfv\_int16xm2\_float16xm2 (C function), 54  
 vfevtxfv\_int16xm4\_float16xm4 (C function), 54  
 vfevtxfv\_int16xm8\_float16xm8 (C function), 54  
 vfevtxfv\_int32xm1\_float32xm1 (C function), 54  
 vfevtxfv\_int32xm2\_float32xm2 (C function), 54  
 vfevtxfv\_int32xm4\_float32xm4 (C function), 54  
 vfevtxfv\_int32xm8\_float32xm8 (C function), 54  
 vfevtxfv\_int64xm1\_float64xm1 (C function), 54  
 vfevtxfv\_int64xm2\_float64xm2 (C function), 55  
 vfevtxfv\_int64xm4\_float64xm4 (C function), 55  
 vfevtxfv\_int64xm8\_float64xm8 (C function), 55  
 vfevtxfv\_mask\_int16xm1\_float16xm1 (C function), 55  
 vfevtxfv\_mask\_int16xm2\_float16xm2 (C function), 55  
 vfevtxfv\_mask\_int16xm4\_float16xm4 (C function), 55  
 vfevtxfv\_mask\_int16xm8\_float16xm8 (C function), 55  
 vfevtxfv\_mask\_int32xm1\_float32xm1 (C function), 55  
 vfevtxfv\_mask\_int32xm2\_float32xm2 (C function), 55  
 vfevtxfv\_mask\_int32xm4\_float32xm4 (C function), 55  
 vfevtxfv\_mask\_int32xm8\_float32xm8 (C function), 55  
 vfevtxfv\_mask\_int64xm1\_float64xm1 (C function), 55  
 vfevtxfv\_mask\_int64xm2\_float64xm2 (C function), 55  
 vfevtxfv\_mask\_int64xm4\_float64xm4 (C function), 55  
 vfevtxfv\_mask\_int64xm8\_float64xm8 (C function), 55  
 vfevtxufv\_mask\_uint16xm1\_float16xm1 (C function), 56  
 vfevtxufv\_mask\_uint16xm2\_float16xm2 (C function), 56  
 vfevtxufv\_mask\_uint16xm4\_float16xm4 (C function), 56  
 vfevtxufv\_mask\_uint16xm8\_float16xm8 (C function), 56  
 vfevtxufv\_mask\_uint32xm1\_float32xm1 (C function), 56  
 vfevtxufv\_mask\_uint32xm2\_float32xm2 (C function), 56  
 vfevtxufv\_mask\_uint32xm4\_float32xm4 (C function), 56  
 vfevtxufv\_mask\_uint32xm8\_float32xm8 (C function), 56  
 vfevtxufv\_mask\_uint64xm1\_float64xm1 (C function), 56  
 vfevtxufv\_mask\_uint64xm2\_float64xm2 (C function), 56  
 vfevtxufv\_mask\_uint64xm4\_float64xm4 (C function), 56  
 vfevtxufv\_mask\_uint64xm8\_float64xm8 (C function), 56  
 vfevtxufv\_uint16xm1\_float16xm1 (C function), 56  
 vfevtxufv\_uint16xm2\_float16xm2 (C function), 56  
 vfevtxufv\_uint16xm4\_float16xm4 (C function), 56  
 vfevtxufv\_uint16xm8\_float16xm8 (C function), 56  
 vfevtxufv\_uint32xm1\_float32xm1 (C function), 56  
 vfevtxufv\_uint32xm2\_float32xm2 (C function), 56  
 vfevtxufv\_uint32xm4\_float32xm4 (C function), 56  
 vfevtxufv\_uint32xm8\_float32xm8 (C function), 56  
 vfevtxufv\_uint64xm1\_float64xm1 (C function), 56  
 vfevtxufv\_uint64xm2\_float64xm2 (C function), 56  
 vfevtxufv\_uint64xm4\_float64xm4 (C function), 56  
 vfevtxufv\_uint64xm8\_float64xm8 (C function), 56  
 vfdivvf\_float16xm1 (C function), 68  
 vfdivvf\_float16xm2 (C function), 68  
 vfdivvf\_float16xm4 (C function), 68  
 vfdivvf\_float16xm8 (C function), 68  
 vfdivvf\_float32xm1 (C function), 68  
 vfdivvf\_float32xm2 (C function), 68  
 vfdivvf\_float32xm4 (C function), 68  
 vfdivvf\_float32xm8 (C function), 68  
 vfdivvf\_float64xm1 (C function), 68  
 vfdivvf\_float64xm2 (C function), 68  
 vfdivvf\_float64xm4 (C function), 68  
 vfdivvf\_float64xm8 (C function), 68  
 vfdivvf\_mask\_float16xm1 (C function), 69  
 vfdivvf\_mask\_float16xm2 (C function), 69  
 vfdivvf\_mask\_float16xm4 (C function), 69  
 vfdivvf\_mask\_float16xm8 (C function), 69  
 vfdivvf\_mask\_float32xm1 (C function), 69  
 vfdivvf\_mask\_float32xm2 (C function), 69  
 vfdivvf\_mask\_float32xm4 (C function), 69  
 vfdivvf\_mask\_float32xm8 (C function), 69  
 vfdivvf\_mask\_float64xm1 (C function), 69  
 vfdivvf\_mask\_float64xm2 (C function), 69  
 vfdivvf\_mask\_float64xm4 (C function), 69  
 vfdivvf\_mask\_float64xm8 (C function), 69  
 vfdivvv\_float16xm1 (C function), 69  
 vfdivvv\_float16xm2 (C function), 69  
 vfdivvv\_float16xm4 (C function), 69  
 vfdivvv\_float16xm8 (C function), 69  
 vfdivvv\_float32xm1 (C function), 70  
 vfdivvv\_float32xm2 (C function), 70  
 vfdivvv\_float32xm4 (C function), 70









vfmulvf\_float32xm8 (C function), 87  
 vfmulvf\_float64xm1 (C function), 87  
 vfmulvf\_float64xm2 (C function), 87  
 vfmulvf\_float64xm4 (C function), 87  
 vfmulvf\_float64xm8 (C function), 87  
 vfmulvf\_mask\_float16xm1 (C function), 87  
 vfmulvf\_mask\_float16xm2 (C function), 87  
 vfmulvf\_mask\_float16xm4 (C function), 87  
 vfmulvf\_mask\_float16xm8 (C function), 87  
 vfmulvf\_mask\_float32xm1 (C function), 87  
 vfmulvf\_mask\_float32xm2 (C function), 87  
 vfmulvf\_mask\_float32xm4 (C function), 87  
 vfmulvf\_mask\_float32xm8 (C function), 88  
 vfmulvf\_mask\_float64xm1 (C function), 88  
 vfmulvf\_mask\_float64xm2 (C function), 88  
 vfmulvf\_mask\_float64xm4 (C function), 88  
 vfmulvf\_mask\_float64xm8 (C function), 88  
 vfmulvv\_float16xm1 (C function), 88  
 vfmulvv\_float16xm2 (C function), 88  
 vfmulvv\_float16xm4 (C function), 88  
 vfmulvv\_float16xm8 (C function), 88  
 vfmulvv\_float32xm1 (C function), 88  
 vfmulvv\_float32xm2 (C function), 88  
 vfmulvv\_float32xm4 (C function), 88  
 vfmulvv\_float32xm8 (C function), 88  
 vfmulvv\_float64xm1 (C function), 88  
 vfmulvv\_float64xm2 (C function), 88  
 vfmulvv\_float64xm4 (C function), 88  
 vfmulvv\_float64xm8 (C function), 88  
 vfmulvv\_mask\_float16xm1 (C function), 88  
 vfmulvv\_mask\_float16xm2 (C function), 89  
 vfmulvv\_mask\_float16xm4 (C function), 89  
 vfmulvv\_mask\_float16xm8 (C function), 89  
 vfmulvv\_mask\_float32xm1 (C function), 89  
 vfmulvv\_mask\_float32xm2 (C function), 89  
 vfmulvv\_mask\_float32xm4 (C function), 89  
 vfmulvv\_mask\_float32xm8 (C function), 89  
 vfmulvv\_mask\_float64xm1 (C function), 89  
 vfmulvv\_mask\_float64xm2 (C function), 89  
 vfmulvv\_mask\_float64xm4 (C function), 89  
 vfmulvv\_mask\_float64xm8 (C function), 89  
 vfmvfs\_float16xm1 (C function), 817  
 vfmvfs\_float16xm2 (C function), 817  
 vfmvfs\_float16xm4 (C function), 817  
 vfmvfs\_float16xm8 (C function), 817  
 vfmvfs\_float32xm1 (C function), 817  
 vfmvfs\_float32xm2 (C function), 817  
 vfmvfs\_float32xm4 (C function), 817  
 vfmvfs\_float32xm8 (C function), 817  
 vfmvfs\_float64xm1 (C function), 817  
 vfmvfs\_float64xm2 (C function), 817  
 vfmvfs\_float64xm4 (C function), 817  
 vfmvfs\_float64xm8 (C function), 818  
 vfmvsf\_float16xm1 (C function), 818  
 vfmvsf\_float16xm2 (C function), 818  
 vfmvsf\_float16xm4 (C function), 818  
 vfmvsf\_float16xm8 (C function), 818  
 vfmvsf\_float32xm1 (C function), 818  
 vfmvsf\_float32xm2 (C function), 818  
 vfmvsf\_float32xm4 (C function), 818  
 vfmvsf\_float32xm8 (C function), 818  
 vfmvsf\_float64xm1 (C function), 818  
 vfmvsf\_float64xm2 (C function), 818  
 vfmvsf\_float64xm4 (C function), 818  
 vfmvsf\_float64xm8 (C function), 818  
 vfmvvf\_float16xm1 (C function), 818  
 vfmvvf\_float16xm2 (C function), 818  
 vfmvvf\_float16xm4 (C function), 818  
 vfmvvf\_float16xm8 (C function), 818  
 vfmvvf\_float32xm1 (C function), 818  
 vfmvvf\_float32xm2 (C function), 818  
 vfmvvf\_float32xm4 (C function), 818  
 vfmvvf\_float32xm8 (C function), 818  
 vfmvvf\_float64xm1 (C function), 818  
 vfmvvf\_float64xm2 (C function), 818  
 vfmvvf\_float64xm4 (C function), 819  
 vfmvvf\_float64xm8 (C function), 819  
 vfncvtfv\_float16xm1\_float32xm2 (C function), 50  
 vfncvtfv\_float16xm2\_float32xm4 (C function), 50  
 vfncvtfv\_float16xm4\_float32xm8 (C function), 50  
 vfncvtfv\_float32xm1\_float64xm2 (C function), 50  
 vfncvtfv\_float32xm2\_float64xm4 (C function), 50  
 vfncvtfv\_float32xm4\_float64xm8 (C function), 50  
 vfncvtfv\_mask\_float16xm1\_float32xm2 (C function), 50  
 vfncvtfv\_mask\_float16xm2\_float32xm4 (C function), 50  
 vfncvtfv\_mask\_float16xm4\_float32xm8 (C function), 50  
 vfncvtfv\_mask\_float32xm1\_float64xm2 (C function), 50  
 vfncvtfv\_mask\_float32xm2\_float64xm4 (C function), 50  
 vfncvtfv\_mask\_float32xm4\_float64xm8 (C function), 51  
 vfncvtfxuv\_float16xm1\_uint32xm2 (C function), 58  
 vfncvtfxuv\_float16xm2\_uint32xm4 (C function), 58  
 vfncvtfxuv\_float16xm4\_uint32xm8 (C function), 58  
 vfncvtfxuv\_float32xm1\_uint64xm2 (C function), 58  
 vfncvtfxuv\_float32xm2\_uint64xm4 (C function), 58  
 vfncvtfxuv\_float32xm4\_uint64xm8 (C function), 58  
 vfncvtfxuv\_mask\_float16xm1\_uint32xm2 (C function), 58  
 vfncvtfxuv\_mask\_float16xm2\_uint32xm4 (C function), 58  
 vfncvtfxuv\_mask\_float16xm4\_uint32xm8 (C function), 58  
 vfncvtfxuv\_mask\_float32xm1\_uint64xm2 (C function), 58  
 vfncvtfxuv\_mask\_float32xm2\_uint64xm4 (C function), 58  
 vfncvtfxuv\_mask\_float32xm4\_uint64xm8 (C function), 58  
 vfncvtfxuv\_mask\_float16xm1\_uint32xm2 (C function), 58  
 vfncvtfxuv\_mask\_float16xm2\_uint32xm4 (C function), 58  
 vfncvtfxuv\_mask\_float16xm4\_uint32xm8 (C function), 58  
 vfncvtfxuv\_mask\_float32xm1\_uint64xm2 (C function), 58  
 vfncvtfxuv\_mask\_float32xm2\_uint64xm4 (C function), 58  
 vfncvtfxuv\_mask\_float32xm4\_uint64xm8 (C function), 58  
 vfncvtfxv\_float16xm1\_int32xm2 (C function), 57



vfncvtxfv\_float16xm2\_int32xm4 (C function), 57  
 vfncvtxfv\_float16xm4\_int32xm8 (C function), 57  
 vfncvtxfv\_float32xm1\_int64xm2 (C function), 57  
 vfncvtxfv\_float32xm2\_int64xm4 (C function), 57  
 vfncvtxfv\_float32xm4\_int64xm8 (C function), 57  
 vfncvtxfv\_mask\_float16xm1\_int32xm2 (C function), 57  
 vfncvtxfv\_mask\_float16xm2\_int32xm4 (C function), 57  
 vfncvtxfv\_mask\_float16xm4\_int32xm8 (C function), 57  
 vfncvtxfv\_mask\_float32xm1\_int64xm2 (C function), 57  
 vfncvtxfv\_mask\_float32xm2\_int64xm4 (C function), 57  
 vfncvtxfv\_mask\_float32xm4\_int64xm8 (C function), 57  
 vfncvtxfv\_int16xm1\_float32xm2 (C function), 58  
 vfncvtxfv\_int16xm2\_float32xm4 (C function), 58  
 vfncvtxfv\_int16xm4\_float32xm8 (C function), 58  
 vfncvtxfv\_int32xm1\_float64xm2 (C function), 59  
 vfncvtxfv\_int32xm2\_float64xm4 (C function), 59  
 vfncvtxfv\_int32xm4\_float64xm8 (C function), 59  
 vfncvtxfv\_int8xm1\_float16xm2 (C function), 59  
 vfncvtxfv\_int8xm2\_float16xm4 (C function), 59  
 vfncvtxfv\_int8xm4\_float16xm8 (C function), 59  
 vfncvtxfv\_mask\_int16xm1\_float32xm2 (C function), 59  
 vfncvtxfv\_mask\_int16xm2\_float32xm4 (C function), 59  
 vfncvtxfv\_mask\_int16xm4\_float32xm8 (C function), 59  
 vfncvtxfv\_mask\_int32xm1\_float64xm2 (C function), 59  
 vfncvtxfv\_mask\_int32xm2\_float64xm4 (C function), 59  
 vfncvtxfv\_mask\_int32xm4\_float64xm8 (C function), 59  
 vfncvtxfv\_mask\_int8xm1\_float16xm2 (C function), 59  
 vfncvtxfv\_mask\_int8xm2\_float16xm4 (C function), 59  
 vfncvtxfv\_mask\_int8xm4\_float16xm8 (C function), 59  
 vfncvtxufv\_mask\_uint16xm1\_float32xm2 (C function), 60  
 vfncvtxufv\_mask\_uint16xm2\_float32xm4 (C function), 60  
 vfncvtxufv\_mask\_uint16xm4\_float32xm8 (C function), 60  
 vfncvtxufv\_mask\_uint32xm1\_float64xm2 (C function), 60  
 vfncvtxufv\_mask\_uint32xm2\_float64xm4 (C function), 60  
 vfncvtxufv\_mask\_uint32xm4\_float64xm8 (C function), 60  
 vfncvtxufv\_mask\_uint8xm1\_float16xm2 (C function), 60  
 vfncvtxufv\_mask\_uint8xm2\_float16xm4 (C function), 60  
 vfncvtxufv\_mask\_uint8xm4\_float16xm8 (C function), 60  
 vfnmaccvf\_float16xm1 (C function), 89  
 vfnmaccvf\_float16xm2 (C function), 89  
 vfnmaccvf\_float16xm4 (C function), 89  
 vfnmaccvf\_float16xm8 (C function), 89  
 vfnmaccvf\_float32xm1 (C function), 89  
 vfnmaccvf\_float32xm2 (C function), 89  
 vfnmaccvf\_float32xm4 (C function), 89  
 vfnmaccvf\_float32xm8 (C function), 90  
 vfnmaccvf\_float64xm1 (C function), 90  
 vfnmaccvf\_float64xm2 (C function), 90  
 vfnmaccvf\_float64xm4 (C function), 90  
 vfnmaccvf\_float64xm8 (C function), 90  
 vfnmaccvf\_mask\_float16xm1 (C function), 90  
 vfnmaccvf\_mask\_float16xm2 (C function), 90  
 vfnmaccvf\_mask\_float16xm4 (C function), 90  
 vfnmaccvf\_mask\_float32xm1 (C function), 90  
 vfnmaccvf\_mask\_float32xm2 (C function), 90  
 vfnmaccvf\_mask\_float32xm4 (C function), 90  
 vfnmaccvf\_mask\_float64xm1 (C function), 90  
 vfnmaccvf\_mask\_float64xm2 (C function), 90  
 vfnmaccvf\_mask\_float64xm4 (C function), 90  
 vfnmaccvv\_float16xm1 (C function), 90  
 vfnmaccvv\_float16xm2 (C function), 90  
 vfnmaccvv\_float16xm4 (C function), 91  
 vfnmaccvv\_float16xm8 (C function), 91  
 vfnmaccvv\_float32xm1 (C function), 91  
 vfnmaccvv\_float32xm2 (C function), 91  
 vfnmaccvv\_float32xm4 (C function), 91  
 vfnmaccvv\_float32xm8 (C function), 91  
 vfnmaccvv\_float64xm1 (C function), 91  
 vfnmaccvv\_float64xm2 (C function), 91  
 vfnmaccvv\_float64xm4 (C function), 91  
 vfnmaccvv\_float64xm8 (C function), 91  
 vfnmaccvv\_mask\_float16xm1 (C function), 91  
 vfnmaccvv\_mask\_float16xm2 (C function), 91  
 vfnmaccvv\_mask\_float16xm4 (C function), 91  
 vfnmaccvv\_mask\_float32xm1 (C function), 91  
 vfnmaccvv\_mask\_float32xm2 (C function), 91  
 vfnmaccvv\_mask\_float32xm4 (C function), 91  
 vfnmaccvv\_mask\_float64xm1 (C function), 91  
 vfnmaccvv\_mask\_float64xm2 (C function), 91  
 vfnmaccvv\_mask\_float64xm4 (C function), 91  
 vfnmaddvf\_float16xm1 (C function), 92  
 vfnmaddvf\_float16xm2 (C function), 92  
 vfnmaddvf\_float16xm4 (C function), 92  
 vfnmaddvf\_float16xm8 (C function), 92  
 vfnmaddvf\_float32xm1 (C function), 92  
 vfnmaddvf\_float32xm2 (C function), 92  
 vfnmaddvf\_float32xm4 (C function), 92  
 vfnmaddvf\_float32xm8 (C function), 92  
 vfnmaddvf\_float64xm1 (C function), 92  
 vfnmaddvf\_float64xm2 (C function), 92  
 vfnmaddvf\_float64xm4 (C function), 92  
 vfnmaddvf\_float64xm8 (C function), 92  
 vfnmaddvf\_mask\_float16xm1 (C function), 92









[illegible]

- vsubvf\_float32xm8 (C function), 116  
 vsubvf\_float64xm1 (C function), 116  
 vsubvf\_float64xm2 (C function), 116  
 vsubvf\_float64xm4 (C function), 116  
 vsubvf\_float64xm8 (C function), 116  
 vsubvf\_mask\_float16xm1 (C function), 116  
 vsubvf\_mask\_float16xm2 (C function), 116  
 vsubvf\_mask\_float16xm4 (C function), 116  
 vsubvf\_mask\_float16xm8 (C function), 116  
 vsubvf\_mask\_float32xm1 (C function), 116  
 vsubvf\_mask\_float32xm2 (C function), 116  
 vsubvf\_mask\_float32xm4 (C function), 116  
 vsubvf\_mask\_float32xm8 (C function), 116  
 vsubvf\_mask\_float64xm1 (C function), 116  
 vsubvf\_mask\_float64xm2 (C function), 116  
 vsubvf\_mask\_float64xm4 (C function), 116  
 vsubvf\_mask\_float64xm8 (C function), 116  
 vsubvv\_float16xm1 (C function), 117  
 vsubvv\_float16xm2 (C function), 117  
 vsubvv\_float16xm4 (C function), 117  
 vsubvv\_float16xm8 (C function), 117  
 vsubvv\_float32xm1 (C function), 117  
 vsubvv\_float32xm2 (C function), 117  
 vsubvv\_float32xm4 (C function), 117  
 vsubvv\_float32xm8 (C function), 117  
 vsubvv\_float64xm1 (C function), 117  
 vsubvv\_float64xm2 (C function), 117  
 vsubvv\_float64xm4 (C function), 117  
 vsubvv\_float64xm8 (C function), 117  
 vsubvv\_mask\_float16xm1 (C function), 117  
 vsubvv\_mask\_float16xm2 (C function), 117  
 vsubvv\_mask\_float16xm4 (C function), 117  
 vsubvv\_mask\_float16xm8 (C function), 117  
 vsubvv\_mask\_float32xm1 (C function), 117  
 vsubvv\_mask\_float32xm2 (C function), 117  
 vsubvv\_mask\_float32xm4 (C function), 117  
 vsubvv\_mask\_float32xm8 (C function), 118  
 vsubvv\_mask\_float64xm1 (C function), 118  
 vsubvv\_mask\_float64xm2 (C function), 118  
 vsubvv\_mask\_float64xm4 (C function), 118  
 vsubvv\_mask\_float64xm8 (C function), 118  
 vfwaddvf\_float32xm2\_float16xm1 (C function), 118  
 vfwaddvf\_float32xm4\_float16xm2 (C function), 118  
 vfwaddvf\_float32xm8\_float16xm4 (C function), 118  
 vfwaddvf\_float64xm2\_float32xm1 (C function), 118  
 vfwaddvf\_float64xm4\_float32xm2 (C function), 118  
 vfwaddvf\_float64xm8\_float32xm4 (C function), 118  
 vfwaddvf\_mask\_float32xm2\_float16xm1 (C function), 118  
 vfwaddvf\_mask\_float32xm4\_float16xm2 (C function), 118  
 vfwaddvf\_mask\_float32xm8\_float16xm4 (C function), 118  
 vfwaddvf\_mask\_float64xm2\_float32xm1 (C function), 119  
 vfwaddvf\_mask\_float64xm4\_float32xm2 (C function), 119  
 vfwaddvf\_mask\_float64xm8\_float32xm4 (C function), 119  
 vfwaddvv\_float32xm2\_float16xm1 (C function), 119  
 vfwaddvv\_float32xm4\_float16xm2 (C function), 119  
 vfwaddvv\_float32xm8\_float16xm4 (C function), 119  
 vfwaddvv\_float64xm2\_float32xm1 (C function), 119  
 vfwaddvv\_float64xm4\_float32xm2 (C function), 119  
 vfwaddvv\_float64xm8\_float32xm4 (C function), 119  
 vfwaddvv\_mask\_float32xm2\_float16xm1 (C function), 119  
 vfwaddvv\_mask\_float32xm4\_float16xm2 (C function), 119  
 vfwaddvv\_mask\_float32xm8\_float16xm4 (C function), 120  
 vfwaddvv\_mask\_float64xm2\_float32xm1 (C function), 120  
 vfwaddvv\_mask\_float64xm4\_float32xm2 (C function), 120  
 vfwaddvv\_mask\_float64xm8\_float32xm4 (C function), 120  
 vfwaddwf\_float32xm2 (C function), 120  
 vfwaddwf\_float32xm4 (C function), 120  
 vfwaddwf\_float32xm8 (C function), 120  
 vfwaddwf\_float64xm2 (C function), 120  
 vfwaddwf\_float64xm4 (C function), 120  
 vfwaddwf\_float64xm8 (C function), 120  
 vfwaddwf\_mask\_float32xm2 (C function), 120  
 vfwaddwf\_mask\_float32xm4 (C function), 120  
 vfwaddwf\_mask\_float32xm8 (C function), 120  
 vfwaddwf\_mask\_float64xm2 (C function), 120  
 vfwaddwf\_mask\_float64xm4 (C function), 121  
 vfwaddwf\_mask\_float64xm8 (C function), 121  
 vfwaddwv\_float32xm2\_float16xm1 (C function), 121  
 vfwaddwv\_float32xm4\_float16xm2 (C function), 121  
 vfwaddwv\_float32xm8\_float16xm4 (C function), 121  
 vfwaddwv\_float64xm2\_float32xm1 (C function), 121  
 vfwaddwv\_float64xm4\_float32xm2 (C function), 121  
 vfwaddwv\_float64xm8\_float32xm4 (C function), 121  
 vfwaddwv\_mask\_float32xm2\_float16xm1 (C function), 121  
 vfwaddwv\_mask\_float32xm4\_float16xm2 (C function), 121  
 vfwaddwv\_mask\_float32xm8\_float16xm4 (C function), 121  
 vfwaddwv\_mask\_float64xm2\_float32xm1 (C function), 122  
 vfwaddwv\_mask\_float64xm4\_float32xm2 (C function), 122  
 vfwaddwv\_mask\_float64xm8\_float32xm4 (C function), 122



- vfwcvtffv\_float32xm2\_float16xm1 (C function), 51  
 vfwcvtffv\_float32xm4\_float16xm2 (C function), 51  
 vfwcvtffv\_float32xm8\_float16xm4 (C function), 51  
 vfwcvtffv\_float64xm2\_float32xm1 (C function), 51  
 vfwcvtffv\_float64xm4\_float32xm2 (C function), 51  
 vfwcvtffv\_float64xm8\_float32xm4 (C function), 51  
 vfwcvtffv\_mask\_float32xm2\_float16xm1 (C function), 51  
 vfwcvtffv\_mask\_float32xm4\_float16xm2 (C function), 51  
 vfwcvtffv\_mask\_float32xm8\_float16xm4 (C function), 51  
 vfwcvtffv\_mask\_float64xm2\_float32xm1 (C function), 51  
 vfwcvtffv\_mask\_float64xm4\_float32xm2 (C function), 51  
 vfwcvtffv\_mask\_float64xm8\_float32xm4 (C function), 51  
 vfwcvtfxuv\_float16xm2\_uint8xm1 (C function), 62  
 vfwcvtfxuv\_float16xm4\_uint8xm2 (C function), 62  
 vfwcvtfxuv\_float16xm8\_uint8xm4 (C function), 62  
 vfwcvtfxuv\_float32xm2\_uint16xm1 (C function), 62  
 vfwcvtfxuv\_float32xm4\_uint16xm2 (C function), 62  
 vfwcvtfxuv\_float32xm8\_uint16xm4 (C function), 62  
 vfwcvtfxuv\_float64xm2\_uint32xm1 (C function), 62  
 vfwcvtfxuv\_float64xm4\_uint32xm2 (C function), 62  
 vfwcvtfxuv\_float64xm8\_uint32xm4 (C function), 62  
 vfwcvtfxuv\_mask\_float16xm2\_uint8xm1 (C function), 62  
 vfwcvtfxuv\_mask\_float16xm4\_uint8xm2 (C function), 62  
 vfwcvtfxuv\_mask\_float16xm8\_uint8xm4 (C function), 62  
 vfwcvtfxuv\_mask\_float32xm2\_uint16xm1 (C function), 62  
 vfwcvtfxuv\_mask\_float32xm4\_uint16xm2 (C function), 62  
 vfwcvtfxuv\_mask\_float32xm8\_uint16xm4 (C function), 62  
 vfwcvtfxuv\_mask\_float64xm2\_uint32xm1 (C function), 62  
 vfwcvtfxuv\_mask\_float64xm4\_uint32xm2 (C function), 62  
 vfwcvtfxuv\_mask\_float64xm8\_uint32xm4 (C function), 62  
 vfwcvtfxv\_float16xm2\_int8xm1 (C function), 60  
 vfwcvtfxv\_float16xm4\_int8xm2 (C function), 61  
 vfwcvtfxv\_float16xm8\_int8xm4 (C function), 61  
 vfwcvtfxv\_float32xm2\_int16xm1 (C function), 61  
 vfwcvtfxv\_float32xm4\_int16xm2 (C function), 61  
 vfwcvtfxv\_float32xm8\_int16xm4 (C function), 61  
 vfwcvtfxv\_float64xm2\_int32xm1 (C function), 61  
 vfwcvtfxv\_float64xm4\_int32xm2 (C function), 61  
 vfwcvtfxv\_float64xm8\_int32xm4 (C function), 61  
 vfwcvtfxv\_mask\_float16xm2\_int8xm1 (C function), 61  
 vfwcvtfxv\_mask\_float16xm4\_int8xm2 (C function), 61  
 vfwcvtfxv\_mask\_float16xm8\_int8xm4 (C function), 61  
 vfwcvtfxv\_mask\_float32xm2\_int16xm1 (C function), 61  
 vfwcvtfxv\_mask\_float32xm4\_int16xm2 (C function), 61  
 vfwcvtfxv\_mask\_float32xm8\_int16xm4 (C function), 61  
 vfwcvtfxv\_mask\_float64xm2\_int32xm1 (C function), 61  
 vfwcvtfxv\_mask\_float64xm4\_int32xm2 (C function), 61  
 vfwcvtfxv\_mask\_float64xm8\_int32xm4 (C function), 61  
 vfwcvtfxv\_mask\_int32xm2\_float16xm1 (C function), 63  
 vfwcvtfxv\_mask\_int32xm4\_float16xm2 (C function), 63  
 vfwcvtfxv\_mask\_int32xm8\_float16xm4 (C function), 63  
 vfwcvtfxv\_mask\_int64xm2\_float32xm1 (C function), 63  
 vfwcvtfxv\_mask\_int64xm4\_float32xm2 (C function), 63  
 vfwcvtfxv\_mask\_int64xm8\_float32xm4 (C function), 63  
 vfwcvtfxv\_mask\_int32xm2\_float16xm1 (C function), 63  
 vfwcvtfxv\_mask\_int32xm4\_float16xm2 (C function), 63  
 vfwcvtfxv\_mask\_int32xm8\_float16xm4 (C function), 63  
 vfwcvtfxv\_mask\_int64xm2\_float32xm1 (C function), 63  
 vfwcvtfxv\_mask\_int64xm4\_float32xm2 (C function), 63  
 vfwcvtfxv\_mask\_int64xm8\_float32xm4 (C function), 63  
 vfwcvtxufv\_mask\_uint32xm2\_float16xm1 (C function), 64  
 vfwcvtxufv\_mask\_uint32xm4\_float16xm2 (C function), 64  
 vfwcvtxufv\_mask\_uint32xm8\_float16xm4 (C function), 64  
 vfwcvtxufv\_mask\_uint64xm2\_float32xm1 (C function), 64  
 vfwcvtxufv\_mask\_uint64xm4\_float32xm2 (C function), 64  
 vfwcvtxufv\_mask\_uint64xm8\_float32xm4 (C function), 64  
 vfwcvtxufv\_uint32xm2\_float16xm1 (C function), 63  
 vfwcvtxufv\_uint32xm4\_float16xm2 (C function), 63  
 vfwcvtxufv\_uint32xm8\_float16xm4 (C function), 63  
 vfwcvtxufv\_uint64xm2\_float32xm1 (C function), 64  
 vfwcvtxufv\_uint64xm4\_float32xm2 (C function), 64  
 vfwcvtxufv\_uint64xm8\_float32xm4 (C function), 64  
 vfwmaccvf\_float32xm2\_float16xm1 (C function), 122  
 vfwmaccvf\_float32xm4\_float16xm2 (C function), 122  
 vfwmaccvf\_float32xm8\_float16xm4 (C function), 122  
 vfwmaccvf\_float64xm2\_float32xm1 (C function), 122  
 vfwmaccvf\_float64xm4\_float32xm2 (C function), 122  
 vfwmaccvf\_float64xm8\_float32xm4 (C function), 122  
 vfwmaccvf\_mask\_float32xm2\_float16xm1 (C function), 122  
 vfwmaccvf\_mask\_float32xm4\_float16xm2 (C function), 122  
 vfwmaccvf\_mask\_float32xm8\_float16xm4 (C function), 122  
 vfwmaccvf\_mask\_float64xm2\_float32xm1 (C function), 123

- [vfwmaccvf\\_mask\\_float64xm4\\_float32xm2 \(C function\), 123](#)  
[vfwmaccvf\\_mask\\_float64xm8\\_float32xm4 \(C function\), 123](#)  
[vfwmaccvv\\_float32xm2\\_float16xm1 \(C function\), 123](#)  
[vfwmaccvv\\_float32xm4\\_float16xm2 \(C function\), 123](#)  
[vfwmaccvv\\_float32xm8\\_float16xm4 \(C function\), 123](#)  
[vfwmaccvv\\_float64xm2\\_float32xm1 \(C function\), 123](#)  
[vfwmaccvv\\_float64xm4\\_float32xm2 \(C function\), 123](#)  
[vfwmaccvv\\_float64xm8\\_float32xm4 \(C function\), 123](#)  
[vfwmaccvv\\_mask\\_float32xm2\\_float16xm1 \(C function\), 123](#)  
[vfwmaccvv\\_mask\\_float32xm4\\_float16xm2 \(C function\), 123](#)  
[vfwmaccvv\\_mask\\_float32xm8\\_float16xm4 \(C function\), 124](#)  
[vfwmaccvv\\_mask\\_float64xm2\\_float32xm1 \(C function\), 124](#)  
[vfwmaccvv\\_mask\\_float64xm4\\_float32xm2 \(C function\), 124](#)  
[vfwmaccvv\\_mask\\_float64xm8\\_float32xm4 \(C function\), 124](#)  
[vfwmsacvf\\_float32xm2\\_float16xm1 \(C function\), 124](#)  
[vfwmsacvf\\_float32xm4\\_float16xm2 \(C function\), 124](#)  
[vfwmsacvf\\_float32xm8\\_float16xm4 \(C function\), 124](#)  
[vfwmsacvf\\_float64xm2\\_float32xm1 \(C function\), 124](#)  
[vfwmsacvf\\_float64xm4\\_float32xm2 \(C function\), 124](#)  
[vfwmsacvf\\_float64xm8\\_float32xm4 \(C function\), 124](#)  
[vfwmsacvf\\_mask\\_float32xm2\\_float16xm1 \(C function\), 124](#)  
[vfwmsacvf\\_mask\\_float32xm4\\_float16xm2 \(C function\), 125](#)  
[vfwmsacvf\\_mask\\_float32xm8\\_float16xm4 \(C function\), 125](#)  
[vfwmsacvf\\_mask\\_float64xm2\\_float32xm1 \(C function\), 125](#)  
[vfwmsacvf\\_mask\\_float64xm4\\_float32xm2 \(C function\), 125](#)  
[vfwmsacvf\\_mask\\_float64xm8\\_float32xm4 \(C function\), 125](#)  
[vfwmsacvv\\_float32xm2\\_float16xm1 \(C function\), 125](#)  
[vfwmsacvv\\_float32xm4\\_float16xm2 \(C function\), 125](#)  
[vfwmsacvv\\_float32xm8\\_float16xm4 \(C function\), 125](#)  
[vfwmsacvv\\_float64xm2\\_float32xm1 \(C function\), 125](#)  
[vfwmsacvv\\_float64xm4\\_float32xm2 \(C function\), 125](#)  
[vfwmsacvv\\_float64xm8\\_float32xm4 \(C function\), 125](#)  
[vfwmsacvv\\_mask\\_float32xm2\\_float16xm1 \(C function\), 126](#)  
[vfwmsacvv\\_mask\\_float32xm4\\_float16xm2 \(C function\), 126](#)  
[vfwmsacvv\\_mask\\_float32xm8\\_float16xm4 \(C function\), 126](#)  
[vfwmsacvv\\_mask\\_float64xm2\\_float32xm1 \(C function\), 126](#)  
[vfwmsacvv\\_mask\\_float64xm4\\_float32xm2 \(C function\), 126](#)  
[vfwmsacvv\\_mask\\_float64xm8\\_float32xm4 \(C function\), 126](#)  
[vfwmsacvv\\_mask\\_float32xm2\\_float16xm1 \(C function\), 126](#)  
[vfwmsacvv\\_mask\\_float32xm4\\_float16xm2 \(C function\), 127](#)  
[vfwmsacvv\\_mask\\_float32xm8\\_float16xm4 \(C function\), 127](#)  
[vfwmsacvv\\_mask\\_float64xm2\\_float32xm1 \(C function\), 127](#)  
[vfwmsacvv\\_mask\\_float64xm4\\_float32xm2 \(C function\), 127](#)  
[vfwmsacvv\\_mask\\_float64xm8\\_float32xm4 \(C function\), 127](#)  
[vfwmulvf\\_float32xm2\\_float16xm1 \(C function\), 126](#)  
[vfwmulvf\\_float32xm4\\_float16xm2 \(C function\), 126](#)  
[vfwmulvf\\_float32xm8\\_float16xm4 \(C function\), 126](#)  
[vfwmulvf\\_float64xm2\\_float32xm1 \(C function\), 126](#)  
[vfwmulvf\\_float64xm4\\_float32xm2 \(C function\), 126](#)  
[vfwmulvf\\_float64xm8\\_float32xm4 \(C function\), 126](#)  
[vfwmulvf\\_mask\\_float32xm2\\_float16xm1 \(C function\), 126](#)  
[vfwmulvf\\_mask\\_float32xm4\\_float16xm2 \(C function\), 127](#)  
[vfwmulvf\\_mask\\_float32xm8\\_float16xm4 \(C function\), 127](#)  
[vfwmulvf\\_mask\\_float64xm2\\_float32xm1 \(C function\), 127](#)  
[vfwmulvf\\_mask\\_float64xm4\\_float32xm2 \(C function\), 127](#)  
[vfwmulvf\\_mask\\_float64xm8\\_float32xm4 \(C function\), 127](#)  
[vfwmulvv\\_float32xm2\\_float16xm1 \(C function\), 127](#)  
[vfwmulvv\\_float32xm4\\_float16xm2 \(C function\), 127](#)  
[vfwmulvv\\_float32xm8\\_float16xm4 \(C function\), 127](#)  
[vfwmulvv\\_float64xm2\\_float32xm1 \(C function\), 127](#)  
[vfwmulvv\\_float64xm4\\_float32xm2 \(C function\), 127](#)  
[vfwmulvv\\_float64xm8\\_float32xm4 \(C function\), 127](#)  
[vfwmulvv\\_mask\\_float32xm2\\_float16xm1 \(C function\), 128](#)  
[vfwmulvv\\_mask\\_float32xm4\\_float16xm2 \(C function\), 128](#)  
[vfwmulvv\\_mask\\_float32xm8\\_float16xm4 \(C function\), 128](#)  
[vfwmulvv\\_mask\\_float64xm2\\_float32xm1 \(C function\), 128](#)  
[vfwmulvv\\_mask\\_float64xm4\\_float32xm2 \(C function\), 128](#)  
[vfwmulvv\\_mask\\_float64xm8\\_float32xm4 \(C function\), 128](#)  
[vfwnmaccvf\\_float32xm2\\_float16xm1 \(C function\), 128](#)  
[vfwnmaccvf\\_float32xm4\\_float16xm2 \(C function\), 128](#)  
[vfwnmaccvf\\_float32xm8\\_float16xm4 \(C function\), 128](#)  
[vfwnmaccvf\\_float64xm2\\_float32xm1 \(C function\), 128](#)  
[vfwnmaccvf\\_float64xm4\\_float32xm2 \(C function\), 128](#)  
[vfwnmaccvf\\_float64xm8\\_float32xm4 \(C function\), 128](#)  
[vfwnmaccvf\\_mask\\_float32xm2\\_float16xm1 \(C function\), 129](#)  
[vfwnmaccvf\\_mask\\_float32xm4\\_float16xm2 \(C function\), 129](#)  
[vfwnmaccvf\\_mask\\_float32xm8\\_float16xm4 \(C function\), 129](#)  
[vfwnmaccvf\\_mask\\_float64xm2\\_float32xm1 \(C function\), 129](#)

<code>vfwnmaccvf_mask_float64xm4_float32xm2</code> (C function), 129	<code>vfwnmsacvv_mask_float64xm4_float32xm2</code> (C function), 132
<code>vfwnmaccvf_mask_float64xm8_float32xm4</code> (C function), 129	<code>vfwnmsacvv_mask_float64xm8_float32xm4</code> (C function), 132
<code>vfwnmaccvv_float32xm2_float16xm1</code> (C function), 129	<code>vfwwredosumvs_float32xm2_float16xm1</code> (C function), 133
<code>vfwnmaccvv_float32xm4_float16xm2</code> (C function), 129	<code>vfwwredosumvs_float32xm4_float16xm2</code> (C function), 133
<code>vfwnmaccvv_float32xm8_float16xm4</code> (C function), 129	<code>vfwwredosumvs_float32xm8_float16xm4</code> (C function), 133
<code>vfwnmaccvv_float64xm2_float32xm1</code> (C function), 129	<code>vfwwredosumvs_float64xm2_float32xm1</code> (C function), 133
<code>vfwnmaccvv_float64xm4_float32xm2</code> (C function), 129	<code>vfwwredosumvs_float64xm4_float32xm2</code> (C function), 133
<code>vfwnmaccvv_float64xm8_float32xm4</code> (C function), 130	<code>vfwwredosumvs_float64xm8_float32xm4</code> (C function), 133
<code>vfwnmaccvv_mask_float32xm2_float16xm1</code> (C function), 130	<code>vfwwredosumvs_mask_float32xm2_float16xm1</code> (C function), 133
<code>vfwnmaccvv_mask_float32xm4_float16xm2</code> (C function), 130	<code>vfwwredosumvs_mask_float32xm4_float16xm2</code> (C function), 133
<code>vfwnmaccvv_mask_float32xm8_float16xm4</code> (C function), 130	<code>vfwwredosumvs_mask_float32xm8_float16xm4</code> (C function), 133
<code>vfwnmaccvv_mask_float64xm2_float32xm1</code> (C function), 130	<code>vfwwredosumvs_mask_float64xm2_float32xm1</code> (C function), 133
<code>vfwnmaccvv_mask_float64xm4_float32xm2</code> (C function), 130	<code>vfwwredosumvs_mask_float64xm4_float32xm2</code> (C function), 133
<code>vfwnmaccvv_mask_float64xm8_float32xm4</code> (C function), 130	<code>vfwwredosumvs_mask_float64xm8_float32xm4</code> (C function), 133
<code>vfwnmsacvf_float32xm2_float16xm1</code> (C function), 130	<code>vfwwredsumvs_float32xm2_float16xm1</code> (C function), 134
<code>vfwnmsacvf_float32xm4_float16xm2</code> (C function), 130	<code>vfwwredsumvs_float32xm4_float16xm2</code> (C function), 134
<code>vfwnmsacvf_float32xm8_float16xm4</code> (C function), 131	<code>vfwwredsumvs_float32xm8_float16xm4</code> (C function), 134
<code>vfwnmsacvf_float64xm2_float32xm1</code> (C function), 131	<code>vfwwredsumvs_float64xm2_float32xm1</code> (C function), 134
<code>vfwnmsacvf_float64xm4_float32xm2</code> (C function), 131	<code>vfwwredsumvs_float64xm4_float32xm2</code> (C function), 134
<code>vfwnmsacvf_float64xm8_float32xm4</code> (C function), 131	<code>vfwwredsumvs_float64xm8_float32xm4</code> (C function), 134
<code>vfwnmsacvf_mask_float32xm2_float16xm1</code> (C function), 131	<code>vfwwredsumvs_mask_float32xm2_float16xm1</code> (C function), 134
<code>vfwnmsacvf_mask_float32xm4_float16xm2</code> (C function), 131	<code>vfwwredsumvs_mask_float32xm4_float16xm2</code> (C function), 134
<code>vfwnmsacvf_mask_float32xm8_float16xm4</code> (C function), 131	<code>vfwwredsumvs_mask_float32xm8_float16xm4</code> (C function), 134
<code>vfwnmsacvf_mask_float64xm2_float32xm1</code> (C function), 131	<code>vfwwredsumvs_mask_float64xm2_float32xm1</code> (C function), 134
<code>vfwnmsacvf_mask_float64xm4_float32xm2</code> (C function), 131	<code>vfwwredsumvs_mask_float64xm4_float32xm2</code> (C function), 134
<code>vfwnmsacvf_mask_float64xm8_float32xm4</code> (C function), 131	<code>vfwwredsumvs_mask_float64xm8_float32xm4</code> (C function), 134
<code>vfwnmsacvv_float32xm2_float16xm1</code> (C function), 132	<code>vfwwsubvf_float32xm2_float16xm1</code> (C function), 135
<code>vfwnmsacvv_float32xm4_float16xm2</code> (C function), 132	<code>vfwwsubvf_float32xm4_float16xm2</code> (C function), 135
<code>vfwnmsacvv_float32xm8_float16xm4</code> (C function), 132	<code>vfwwsubvf_float32xm8_float16xm4</code> (C function), 135
<code>vfwnmsacvv_float64xm2_float32xm1</code> (C function), 132	<code>vfwwsubvf_float64xm2_float32xm1</code> (C function), 135
<code>vfwnmsacvv_float64xm4_float32xm2</code> (C function), 132	<code>vfwwsubvf_float64xm4_float32xm2</code> (C function), 135
<code>vfwnmsacvv_float64xm8_float32xm4</code> (C function), 132	<code>vfwwsubvf_float64xm8_float32xm4</code> (C function), 135
<code>vfwnmsacvv_mask_float32xm2_float16xm1</code> (C function), 132	<code>vfwwsubvf_mask_float32xm2_float16xm1</code> (C function), 135
<code>vfwnmsacvv_mask_float32xm4_float16xm2</code> (C function), 132	
<code>vfwnmsacvv_mask_float32xm8_float16xm4</code> (C function), 132	
<code>vfwnmsacvv_mask_float64xm2_float32xm1</code> (C function), 132	

vfwsbvf\_mask\_float32xm4\_float16xm2 (C function), 135  
 vfwsbvf\_mask\_float32xm8\_float16xm4 (C function), 135  
 vfwsbvf\_mask\_float64xm2\_float32xm1 (C function), 135  
 vfwsbvf\_mask\_float64xm4\_float32xm2 (C function), 135  
 vfwsbvf\_mask\_float64xm8\_float32xm4 (C function), 135  
 vfwsbvfv\_float32xm2\_float16xm1 (C function), 136  
 vfwsbvfv\_float32xm4\_float16xm2 (C function), 136  
 vfwsbvfv\_float32xm8\_float16xm4 (C function), 136  
 vfwsbvfv\_float64xm2\_float32xm1 (C function), 136  
 vfwsbvfv\_float64xm4\_float32xm2 (C function), 136  
 vfwsbvfv\_float64xm8\_float32xm4 (C function), 136  
 vfwsbvfv\_mask\_float32xm2\_float16xm1 (C function), 136  
 vfwsbvfv\_mask\_float32xm4\_float16xm2 (C function), 136  
 vfwsbvfv\_mask\_float32xm8\_float16xm4 (C function), 136  
 vfwsbvfv\_mask\_float64xm2\_float32xm1 (C function), 136  
 vfwsbvfv\_mask\_float64xm4\_float32xm2 (C function), 136  
 vfwsbvfv\_mask\_float64xm8\_float32xm4 (C function), 136  
 vfwsbwf\_float32xm2 (C function), 137  
 vfwsbwf\_float32xm4 (C function), 137  
 vfwsbwf\_float32xm8 (C function), 137  
 vfwsbwf\_float64xm2 (C function), 137  
 vfwsbwf\_float64xm4 (C function), 137  
 vfwsbwf\_float64xm8 (C function), 137  
 vfwsbwf\_mask\_float32xm2 (C function), 137  
 vfwsbwf\_mask\_float32xm4 (C function), 137  
 vfwsbwf\_mask\_float32xm8 (C function), 137  
 vfwsbwf\_mask\_float64xm2 (C function), 137  
 vfwsbwf\_mask\_float64xm4 (C function), 137  
 vfwsbwf\_mask\_float64xm8 (C function), 137  
 vfwsbwv\_float32xm2\_float16xm1 (C function), 138  
 vfwsbwv\_float32xm4\_float16xm2 (C function), 138  
 vfwsbwv\_float32xm8\_float16xm4 (C function), 138  
 vfwsbwv\_float64xm2\_float32xm1 (C function), 138  
 vfwsbwv\_float64xm4\_float32xm2 (C function), 138  
 vfwsbwv\_float64xm8\_float32xm4 (C function), 138  
 vfwsbwv\_mask\_float32xm2\_float16xm1 (C function), 138  
 vfwsbwv\_mask\_float32xm4\_float16xm2 (C function), 138  
 vfwsbwv\_mask\_float32xm8\_float16xm4 (C function), 138  
 vfwsbwv\_mask\_float64xm2\_float32xm1 (C function), 138  
 vfwsbwv\_mask\_float64xm4\_float32xm2 (C function), 138  
 vfwsbwv\_mask\_float64xm8\_float32xm4 (C function), 138  
 vidv\_mask\_uint16xm1 (C function), 819  
 vidv\_mask\_uint16xm2 (C function), 819  
 vidv\_mask\_uint16xm4 (C function), 819  
 vidv\_mask\_uint16xm8 (C function), 819  
 vidv\_mask\_uint32xm1 (C function), 819  
 vidv\_mask\_uint32xm2 (C function), 820  
 vidv\_mask\_uint32xm4 (C function), 820  
 vidv\_mask\_uint32xm8 (C function), 820  
 vidv\_mask\_uint64xm1 (C function), 820  
 vidv\_mask\_uint64xm2 (C function), 820  
 vidv\_mask\_uint64xm4 (C function), 820  
 vidv\_mask\_uint64xm8 (C function), 820  
 vidv\_mask\_uint8xm1 (C function), 820  
 vidv\_mask\_uint8xm2 (C function), 820  
 vidv\_mask\_uint8xm4 (C function), 820  
 vidv\_mask\_uint8xm8 (C function), 820  
 vidv\_uint16xm1 (C function), 819  
 vidv\_uint16xm2 (C function), 819  
 vidv\_uint16xm4 (C function), 819  
 vidv\_uint16xm8 (C function), 819  
 vidv\_uint32xm1 (C function), 819  
 vidv\_uint32xm2 (C function), 819  
 vidv\_uint32xm4 (C function), 819  
 vidv\_uint32xm8 (C function), 819  
 vidv\_uint64xm1 (C function), 819  
 vidv\_uint64xm2 (C function), 819  
 vidv\_uint64xm4 (C function), 819  
 vidv\_uint64xm8 (C function), 819  
 vidv\_uint8xm1 (C function), 819  
 vidv\_uint8xm2 (C function), 819  
 vidv\_uint8xm4 (C function), 819  
 vidv\_uint8xm8 (C function), 819  
 viotam\_mask\_uint16xm1\_e16xm1 (C function), 821  
 viotam\_mask\_uint16xm2\_e16xm2 (C function), 821  
 viotam\_mask\_uint16xm4\_e16xm4 (C function), 821  
 viotam\_mask\_uint16xm8\_e16xm8 (C function), 821  
 viotam\_mask\_uint32xm1\_e32xm1 (C function), 821  
 viotam\_mask\_uint32xm2\_e32xm2 (C function), 821  
 viotam\_mask\_uint32xm4\_e32xm4 (C function), 821  
 viotam\_mask\_uint32xm8\_e32xm8 (C function), 821  
 viotam\_mask\_uint64xm1\_e64xm1 (C function), 821  
 viotam\_mask\_uint64xm2\_e64xm2 (C function), 821  
 viotam\_mask\_uint64xm4\_e64xm4 (C function), 821  
 viotam\_mask\_uint64xm8\_e64xm8 (C function), 821  
 viotam\_mask\_uint8xm1\_e8xm1 (C function), 821  
 viotam\_mask\_uint8xm2\_e8xm2 (C function), 821  
 viotam\_mask\_uint8xm4\_e8xm4 (C function), 821  
 viotam\_mask\_uint8xm8\_e8xm8 (C function), 821  
 viotam\_uint16xm1\_e16xm1 (C function), 820  
 viotam\_uint16xm2\_e16xm2 (C function), 820

viotam\_uint16xm4\_e16xm4 (C function), 820  
viotam\_uint16xm8\_e16xm8 (C function), 820  
viotam\_uint32xm1\_e32xm1 (C function), 820  
viotam\_uint32xm2\_e32xm2 (C function), 820  
viotam\_uint32xm4\_e32xm4 (C function), 820  
viotam\_uint32xm8\_e32xm8 (C function), 820  
viotam\_uint64xm1\_e64xm1 (C function), 820  
viotam\_uint64xm2\_e64xm2 (C function), 820  
viotam\_uint64xm4\_e64xm4 (C function), 820  
viotam\_uint64xm8\_e64xm8 (C function), 820  
viotam\_uint8xm1\_e8xm1 (C function), 820  
viotam\_uint8xm2\_e8xm2 (C function), 820  
viotam\_uint8xm4\_e8xm4 (C function), 820  
viotam\_uint8xm8\_e8xm8 (C function), 821  
vlbu\_mask\_uint16xm1 (C function), 413  
vlbu\_mask\_uint16xm2 (C function), 413  
vlbu\_mask\_uint16xm4 (C function), 413  
vlbu\_mask\_uint16xm8 (C function), 413  
vlbu\_mask\_uint32xm1 (C function), 413  
vlbu\_mask\_uint32xm2 (C function), 413  
vlbu\_mask\_uint32xm4 (C function), 413  
vlbu\_mask\_uint32xm8 (C function), 413  
vlbu\_mask\_uint64xm1 (C function), 413  
vlbu\_mask\_uint64xm2 (C function), 413  
vlbu\_mask\_uint64xm4 (C function), 413  
vlbu\_mask\_uint64xm8 (C function), 413  
vlbu\_mask\_uint8xm1 (C function), 414  
vlbu\_mask\_uint8xm2 (C function), 414  
vlbu\_mask\_uint8xm4 (C function), 414  
vlbu\_mask\_uint8xm8 (C function), 414  
vlbu\_uint16xm1 (C function), 412  
vlbu\_uint16xm2 (C function), 412  
vlbu\_uint16xm4 (C function), 412  
vlbu\_uint16xm8 (C function), 412  
vlbu\_uint32xm1 (C function), 413  
vlbu\_uint32xm2 (C function), 413  
vlbu\_uint32xm4 (C function), 413  
vlbu\_uint32xm8 (C function), 413  
vlbu\_uint64xm1 (C function), 413  
vlbu\_uint64xm2 (C function), 413  
vlbu\_uint64xm4 (C function), 413  
vlbu\_uint64xm8 (C function), 413  
vlbu\_uint8xm1 (C function), 413  
vlbu\_uint8xm2 (C function), 413  
vlbu\_uint8xm4 (C function), 413  
vlbu\_uint8xm8 (C function), 413  
vlbv\_int16xm1 (C function), 411  
vlbv\_int16xm2 (C function), 411  
vlbv\_int16xm4 (C function), 411  
vlbv\_int16xm8 (C function), 411  
vlbv\_int32xm1 (C function), 411  
vlbv\_int32xm2 (C function), 411  
vlbv\_int32xm4 (C function), 411  
vlbv\_int32xm8 (C function), 411  
vlbv\_int64xm1 (C function), 411  
vlbv\_int64xm2 (C function), 411  
vlbv\_int64xm4 (C function), 411  
vlbv\_int64xm8 (C function), 411  
vlbv\_int8xm1 (C function), 411  
vlbv\_int8xm2 (C function), 411  
vlbv\_int8xm4 (C function), 411  
vlbv\_int8xm8 (C function), 411  
vlbv\_mask\_int16xm1 (C function), 411  
vlbv\_mask\_int16xm2 (C function), 411  
vlbv\_mask\_int16xm4 (C function), 412  
vlbv\_mask\_int16xm8 (C function), 412  
vlbv\_mask\_int32xm1 (C function), 412  
vlbv\_mask\_int32xm2 (C function), 412  
vlbv\_mask\_int32xm4 (C function), 412  
vlbv\_mask\_int32xm8 (C function), 412  
vlbv\_mask\_int64xm1 (C function), 412  
vlbv\_mask\_int64xm2 (C function), 412  
vlbv\_mask\_int64xm4 (C function), 412  
vlbv\_mask\_int64xm8 (C function), 412  
vlbv\_mask\_int8xm1 (C function), 412  
vlbv\_mask\_int8xm2 (C function), 412  
vlbv\_mask\_int8xm4 (C function), 412  
vlbv\_mask\_int8xm8 (C function), 412  
vlev\_float16xm1 (C function), 414  
vlev\_float16xm2 (C function), 414  
vlev\_float16xm4 (C function), 414  
vlev\_float16xm8 (C function), 414  
vlev\_float32xm1 (C function), 414  
vlev\_float32xm2 (C function), 414  
vlev\_float32xm4 (C function), 414  
vlev\_float32xm8 (C function), 414  
vlev\_float64xm1 (C function), 414  
vlev\_float64xm2 (C function), 414  
vlev\_float64xm4 (C function), 414  
vlev\_float64xm8 (C function), 414  
vlev\_int16xm1 (C function), 414  
vlev\_int16xm2 (C function), 414  
vlev\_int16xm4 (C function), 414  
vlev\_int16xm8 (C function), 414  
vlev\_int32xm1 (C function), 414  
vlev\_int32xm2 (C function), 414  
vlev\_int32xm4 (C function), 414  
vlev\_int32xm8 (C function), 415  
vlev\_int64xm1 (C function), 415  
vlev\_int64xm2 (C function), 415  
vlev\_int64xm4 (C function), 415  
vlev\_int64xm8 (C function), 415  
vlev\_int8xm1 (C function), 415  
vlev\_int8xm2 (C function), 415  
vlev\_int8xm4 (C function), 415  
vlev\_int8xm8 (C function), 415  
vlev\_mask\_float16xm1 (C function), 415  
vlev\_mask\_float16xm2 (C function), 415



vlev\_mask\_float16xm4 (C function), 415  
 vlev\_mask\_float16xm8 (C function), 415  
 vlev\_mask\_float32xm1 (C function), 416  
 vlev\_mask\_float32xm2 (C function), 416  
 vlev\_mask\_float32xm4 (C function), 416  
 vlev\_mask\_float32xm8 (C function), 416  
 vlev\_mask\_float64xm1 (C function), 416  
 vlev\_mask\_float64xm2 (C function), 416  
 vlev\_mask\_float64xm4 (C function), 416  
 vlev\_mask\_float64xm8 (C function), 416  
 vlev\_mask\_int16xm1 (C function), 416  
 vlev\_mask\_int16xm2 (C function), 416  
 vlev\_mask\_int16xm4 (C function), 416  
 vlev\_mask\_int16xm8 (C function), 416  
 vlev\_mask\_int32xm1 (C function), 416  
 vlev\_mask\_int32xm2 (C function), 416  
 vlev\_mask\_int32xm4 (C function), 416  
 vlev\_mask\_int32xm8 (C function), 416  
 vlev\_mask\_int64xm1 (C function), 416  
 vlev\_mask\_int64xm2 (C function), 416  
 vlev\_mask\_int64xm4 (C function), 416  
 vlev\_mask\_int64xm8 (C function), 416  
 vlev\_mask\_int8xm1 (C function), 416  
 vlev\_mask\_int8xm2 (C function), 416  
 vlev\_mask\_int8xm4 (C function), 416  
 vlev\_mask\_int8xm8 (C function), 417  
 vlev\_mask\_uint16xm1 (C function), 417  
 vlev\_mask\_uint16xm2 (C function), 417  
 vlev\_mask\_uint16xm4 (C function), 417  
 vlev\_mask\_uint16xm8 (C function), 417  
 vlev\_mask\_uint32xm1 (C function), 417  
 vlev\_mask\_uint32xm2 (C function), 417  
 vlev\_mask\_uint32xm4 (C function), 417  
 vlev\_mask\_uint32xm8 (C function), 417  
 vlev\_mask\_uint64xm1 (C function), 417  
 vlev\_mask\_uint64xm2 (C function), 417  
 vlev\_mask\_uint64xm4 (C function), 417  
 vlev\_mask\_uint64xm8 (C function), 417  
 vlev\_mask\_uint8xm1 (C function), 417  
 vlev\_mask\_uint8xm2 (C function), 417  
 vlev\_mask\_uint8xm4 (C function), 417  
 vlev\_mask\_uint8xm8 (C function), 417  
 vlev\_uint16xm1 (C function), 415  
 vlev\_uint16xm2 (C function), 415  
 vlev\_uint16xm4 (C function), 415  
 vlev\_uint16xm8 (C function), 415  
 vlev\_uint32xm1 (C function), 415  
 vlev\_uint32xm2 (C function), 415  
 vlev\_uint32xm4 (C function), 415  
 vlev\_uint32xm8 (C function), 415  
 vlev\_uint64xm1 (C function), 415  
 vlev\_uint64xm2 (C function), 415  
 vlev\_uint64xm4 (C function), 415  
 vlev\_uint64xm8 (C function), 415

vlev\_uint8xm1 (C function), 415  
 vlev\_uint8xm2 (C function), 415  
 vlev\_uint8xm4 (C function), 415  
 vlev\_uint8xm8 (C function), 415  
 vlhuv\_mask\_uint16xm1 (C function), 420  
 vlhuv\_mask\_uint16xm2 (C function), 420  
 vlhuv\_mask\_uint16xm4 (C function), 420  
 vlhuv\_mask\_uint16xm8 (C function), 420  
 vlhuv\_mask\_uint32xm1 (C function), 420  
 vlhuv\_mask\_uint32xm2 (C function), 420  
 vlhuv\_mask\_uint32xm4 (C function), 420  
 vlhuv\_mask\_uint32xm8 (C function), 420  
 vlhuv\_mask\_uint64xm1 (C function), 420  
 vlhuv\_mask\_uint64xm2 (C function), 420  
 vlhuv\_mask\_uint64xm4 (C function), 420  
 vlhuv\_mask\_uint64xm8 (C function), 420  
 vlhuv\_mask\_uint8xm1 (C function), 420  
 vlhuv\_mask\_uint8xm2 (C function), 420  
 vlhuv\_mask\_uint8xm4 (C function), 420  
 vlhuv\_mask\_uint8xm8 (C function), 420  
 vlhuv\_uint16xm1 (C function), 419  
 vlhuv\_uint16xm2 (C function), 419  
 vlhuv\_uint16xm4 (C function), 419  
 vlhuv\_uint16xm8 (C function), 419  
 vlhuv\_uint32xm1 (C function), 419  
 vlhuv\_uint32xm2 (C function), 419  
 vlhuv\_uint32xm4 (C function), 419  
 vlhuv\_uint32xm8 (C function), 419  
 vlhuv\_uint64xm1 (C function), 419  
 vlhuv\_uint64xm2 (C function), 419  
 vlhuv\_uint64xm4 (C function), 419  
 vlhuv\_uint64xm8 (C function), 419  
 vlhuv\_uint8xm1 (C function), 419  
 vlhuv\_uint8xm2 (C function), 419  
 vlhuv\_uint8xm4 (C function), 419  
 vlhuv\_uint8xm8 (C function), 420  
 vlhv\_int16xm1 (C function), 418  
 vlhv\_int16xm2 (C function), 418  
 vlhv\_int16xm4 (C function), 418  
 vlhv\_int16xm8 (C function), 418  
 vlhv\_int32xm1 (C function), 418  
 vlhv\_int32xm2 (C function), 418  
 vlhv\_int32xm4 (C function), 418  
 vlhv\_int32xm8 (C function), 418  
 vlhv\_int64xm1 (C function), 418  
 vlhv\_int64xm2 (C function), 418  
 vlhv\_int64xm4 (C function), 418  
 vlhv\_int64xm8 (C function), 418  
 vlhv\_int8xm1 (C function), 418  
 vlhv\_int8xm2 (C function), 418  
 vlhv\_int8xm4 (C function), 418  
 vlhv\_int8xm8 (C function), 418  
 vlhv\_mask\_int16xm1 (C function), 418  
 vlhv\_mask\_int16xm2 (C function), 418



vlhv\_mask\_int16xm4 (C function), 418  
vlhv\_mask\_int16xm8 (C function), 418  
vlhv\_mask\_int32xm1 (C function), 418  
vlhv\_mask\_int32xm2 (C function), 418  
vlhv\_mask\_int32xm4 (C function), 418  
vlhv\_mask\_int32xm8 (C function), 418  
vlhv\_mask\_int64xm1 (C function), 418  
vlhv\_mask\_int64xm2 (C function), 419  
vlhv\_mask\_int64xm4 (C function), 419  
vlhv\_mask\_int64xm8 (C function), 419  
vlhv\_mask\_int8xm1 (C function), 419  
vlhv\_mask\_int8xm2 (C function), 419  
vlhv\_mask\_int8xm4 (C function), 419  
vlhv\_mask\_int8xm8 (C function), 419  
vlsbuv\_mask\_uint16xm1 (C function), 423  
vlsbuv\_mask\_uint16xm2 (C function), 423  
vlsbuv\_mask\_uint16xm4 (C function), 423  
vlsbuv\_mask\_uint16xm8 (C function), 423  
vlsbuv\_mask\_uint32xm1 (C function), 423  
vlsbuv\_mask\_uint32xm2 (C function), 423  
vlsbuv\_mask\_uint32xm4 (C function), 423  
vlsbuv\_mask\_uint32xm8 (C function), 423  
vlsbuv\_mask\_uint64xm1 (C function), 423  
vlsbuv\_mask\_uint64xm2 (C function), 423  
vlsbuv\_mask\_uint64xm4 (C function), 423  
vlsbuv\_mask\_uint64xm8 (C function), 423  
vlsbuv\_mask\_uint8xm1 (C function), 423  
vlsbuv\_mask\_uint8xm2 (C function), 423  
vlsbuv\_mask\_uint8xm4 (C function), 423  
vlsbuv\_mask\_uint8xm8 (C function), 424  
vlsbuv\_uint16xm1 (C function), 422  
vlsbuv\_uint16xm2 (C function), 422  
vlsbuv\_uint16xm4 (C function), 422  
vlsbuv\_uint16xm8 (C function), 422  
vlsbuv\_uint32xm1 (C function), 422  
vlsbuv\_uint32xm2 (C function), 422  
vlsbuv\_uint32xm4 (C function), 422  
vlsbuv\_uint32xm8 (C function), 422  
vlsbuv\_uint64xm1 (C function), 422  
vlsbuv\_uint64xm2 (C function), 423  
vlsbuv\_uint64xm4 (C function), 423  
vlsbuv\_uint64xm8 (C function), 423  
vlsbuv\_uint8xm1 (C function), 423  
vlsbuv\_uint8xm2 (C function), 423  
vlsbuv\_uint8xm4 (C function), 423  
vlsbuv\_uint8xm8 (C function), 423  
vlsbv\_int16xm1 (C function), 421  
vlsbv\_int16xm2 (C function), 421  
vlsbv\_int16xm4 (C function), 421  
vlsbv\_int16xm8 (C function), 421  
vlsbv\_int32xm1 (C function), 421  
vlsbv\_int32xm2 (C function), 421  
vlsbv\_int32xm4 (C function), 421  
vlsbv\_int32xm8 (C function), 421  
vlsbv\_int64xm1 (C function), 421  
vlsbv\_int64xm2 (C function), 421  
vlsbv\_int64xm4 (C function), 421  
vlsbv\_int64xm8 (C function), 421  
vlsbv\_int8xm1 (C function), 421  
vlsbv\_int8xm2 (C function), 421  
vlsbv\_int8xm4 (C function), 421  
vlsbv\_int8xm8 (C function), 421  
vlsbv\_mask\_int16xm1 (C function), 421  
vlsbv\_mask\_int16xm2 (C function), 421  
vlsbv\_mask\_int16xm4 (C function), 421  
vlsbv\_mask\_int16xm8 (C function), 421  
vlsbv\_mask\_int32xm1 (C function), 421  
vlsbv\_mask\_int32xm2 (C function), 422  
vlsbv\_mask\_int32xm4 (C function), 422  
vlsbv\_mask\_int32xm8 (C function), 422  
vlsbv\_mask\_int64xm1 (C function), 422  
vlsbv\_mask\_int64xm2 (C function), 422  
vlsbv\_mask\_int64xm4 (C function), 422  
vlsbv\_mask\_int64xm8 (C function), 422  
vlsbv\_mask\_int8xm1 (C function), 422  
vlsbv\_mask\_int8xm2 (C function), 422  
vlsbv\_mask\_int8xm4 (C function), 422  
vlsbv\_mask\_int8xm8 (C function), 422  
vlseg2buv\_mask\_uint16x2xm1 (C function), 497  
vlseg2buv\_mask\_uint16x2xm2 (C function), 497  
vlseg2buv\_mask\_uint16x2xm4 (C function), 497  
vlseg2buv\_mask\_uint32x2xm1 (C function), 497  
vlseg2buv\_mask\_uint32x2xm2 (C function), 497  
vlseg2buv\_mask\_uint32x2xm4 (C function), 497  
vlseg2buv\_mask\_uint64x2xm1 (C function), 497  
vlseg2buv\_mask\_uint64x2xm2 (C function), 497  
vlseg2buv\_mask\_uint64x2xm4 (C function), 497  
vlseg2buv\_mask\_uint8x2xm1 (C function), 497  
vlseg2buv\_mask\_uint8x2xm2 (C function), 497  
vlseg2buv\_mask\_uint8x2xm4 (C function), 497  
vlseg2buv\_uint16x2xm1 (C function), 496  
vlseg2buv\_uint16x2xm2 (C function), 496  
vlseg2buv\_uint16x2xm4 (C function), 496  
vlseg2buv\_uint32x2xm1 (C function), 496  
vlseg2buv\_uint32x2xm2 (C function), 496  
vlseg2buv\_uint32x2xm4 (C function), 496  
vlseg2buv\_uint64x2xm1 (C function), 496  
vlseg2buv\_uint64x2xm2 (C function), 496  
vlseg2buv\_uint64x2xm4 (C function), 497  
vlseg2buv\_uint8x2xm1 (C function), 497  
vlseg2buv\_uint8x2xm2 (C function), 497  
vlseg2buv\_uint8x2xm4 (C function), 497  
vlseg2bv\_int16x2xm1 (C function), 495  
vlseg2bv\_int16x2xm2 (C function), 495  
vlseg2bv\_int16x2xm4 (C function), 495  
vlseg2bv\_int32x2xm1 (C function), 495  
vlseg2bv\_int32x2xm2 (C function), 495  
vlseg2bv\_int32x2xm4 (C function), 495

vlseg2bv\_int64x2xm1 (C function), 495  
 vlseg2bv\_int64x2xm2 (C function), 495  
 vlseg2bv\_int64x2xm4 (C function), 495  
 vlseg2bv\_int8x2xm1 (C function), 495  
 vlseg2bv\_int8x2xm2 (C function), 495  
 vlseg2bv\_int8x2xm4 (C function), 495  
 vlseg2bv\_mask\_int16x2xm1 (C function), 495  
 vlseg2bv\_mask\_int16x2xm2 (C function), 496  
 vlseg2bv\_mask\_int16x2xm4 (C function), 496  
 vlseg2bv\_mask\_int32x2xm1 (C function), 496  
 vlseg2bv\_mask\_int32x2xm2 (C function), 496  
 vlseg2bv\_mask\_int32x2xm4 (C function), 496  
 vlseg2bv\_mask\_int64x2xm1 (C function), 496  
 vlseg2bv\_mask\_int64x2xm2 (C function), 496  
 vlseg2bv\_mask\_int64x2xm4 (C function), 496  
 vlseg2bv\_mask\_int8x2xm1 (C function), 496  
 vlseg2bv\_mask\_int8x2xm2 (C function), 496  
 vlseg2bv\_mask\_int8x2xm4 (C function), 496  
 vlseg2ev\_float16x2xm1 (C function), 498  
 vlseg2ev\_float16x2xm2 (C function), 498  
 vlseg2ev\_float16x2xm4 (C function), 498  
 vlseg2ev\_float32x2xm1 (C function), 498  
 vlseg2ev\_float32x2xm2 (C function), 498  
 vlseg2ev\_float32x2xm4 (C function), 498  
 vlseg2ev\_float64x2xm1 (C function), 498  
 vlseg2ev\_float64x2xm2 (C function), 498  
 vlseg2ev\_float64x2xm4 (C function), 498  
 vlseg2ev\_int16x2xm1 (C function), 498  
 vlseg2ev\_int16x2xm2 (C function), 498  
 vlseg2ev\_int16x2xm4 (C function), 498  
 vlseg2ev\_int32x2xm1 (C function), 498  
 vlseg2ev\_int32x2xm2 (C function), 498  
 vlseg2ev\_int32x2xm4 (C function), 498  
 vlseg2ev\_int64x2xm1 (C function), 498  
 vlseg2ev\_int64x2xm2 (C function), 498  
 vlseg2ev\_int64x2xm4 (C function), 498  
 vlseg2ev\_int8x2xm1 (C function), 498  
 vlseg2ev\_int8x2xm2 (C function), 498  
 vlseg2ev\_int8x2xm4 (C function), 498  
 vlseg2ev\_mask\_float16x2xm1 (C function), 499  
 vlseg2ev\_mask\_float16x2xm2 (C function), 499  
 vlseg2ev\_mask\_float16x2xm4 (C function), 499  
 vlseg2ev\_mask\_float32x2xm1 (C function), 499  
 vlseg2ev\_mask\_float32x2xm2 (C function), 499  
 vlseg2ev\_mask\_float32x2xm4 (C function), 499  
 vlseg2ev\_mask\_float64x2xm1 (C function), 499  
 vlseg2ev\_mask\_float64x2xm2 (C function), 499  
 vlseg2ev\_mask\_float64x2xm4 (C function), 499  
 vlseg2ev\_mask\_int16x2xm1 (C function), 499  
 vlseg2ev\_mask\_int16x2xm2 (C function), 499  
 vlseg2ev\_mask\_int16x2xm4 (C function), 499  
 vlseg2ev\_mask\_int32x2xm1 (C function), 499  
 vlseg2ev\_mask\_int32x2xm2 (C function), 499  
 vlseg2ev\_mask\_int32x2xm4 (C function), 499  
 vlseg2ev\_mask\_int64x2xm1 (C function), 499  
 vlseg2ev\_mask\_int64x2xm2 (C function), 499  
 vlseg2ev\_mask\_int64x2xm4 (C function), 499  
 vlseg2ev\_mask\_int8x2xm1 (C function), 499  
 vlseg2ev\_mask\_int8x2xm2 (C function), 500  
 vlseg2ev\_mask\_int8x2xm4 (C function), 500  
 vlseg2ev\_mask\_uint16x2xm1 (C function), 500  
 vlseg2ev\_mask\_uint16x2xm2 (C function), 500  
 vlseg2ev\_mask\_uint16x2xm4 (C function), 500  
 vlseg2ev\_mask\_uint32x2xm1 (C function), 500  
 vlseg2ev\_mask\_uint32x2xm2 (C function), 500  
 vlseg2ev\_mask\_uint32x2xm4 (C function), 500  
 vlseg2ev\_mask\_uint64x2xm1 (C function), 500  
 vlseg2ev\_mask\_uint64x2xm2 (C function), 500  
 vlseg2ev\_mask\_uint64x2xm4 (C function), 500  
 vlseg2ev\_mask\_uint8x2xm1 (C function), 500  
 vlseg2ev\_mask\_uint8x2xm2 (C function), 500  
 vlseg2ev\_mask\_uint8x2xm4 (C function), 500  
 vlseg2ev\_uint16x2xm1 (C function), 498  
 vlseg2ev\_uint16x2xm2 (C function), 498  
 vlseg2ev\_uint16x2xm4 (C function), 498  
 vlseg2ev\_uint32x2xm1 (C function), 498  
 vlseg2ev\_uint32x2xm2 (C function), 498  
 vlseg2ev\_uint32x2xm4 (C function), 498  
 vlseg2ev\_uint64x2xm1 (C function), 498  
 vlseg2ev\_uint64x2xm2 (C function), 498  
 vlseg2ev\_uint64x2xm4 (C function), 498  
 vlseg2ev\_uint8x2xm1 (C function), 498  
 vlseg2ev\_uint8x2xm2 (C function), 498  
 vlseg2ev\_uint8x2xm4 (C function), 499  
 vlseg2huv\_mask\_uint16x2xm1 (C function), 502  
 vlseg2huv\_mask\_uint16x2xm2 (C function), 502  
 vlseg2huv\_mask\_uint16x2xm4 (C function), 502  
 vlseg2huv\_mask\_uint32x2xm1 (C function), 502  
 vlseg2huv\_mask\_uint32x2xm2 (C function), 502  
 vlseg2huv\_mask\_uint32x2xm4 (C function), 503  
 vlseg2huv\_mask\_uint64x2xm1 (C function), 503  
 vlseg2huv\_mask\_uint64x2xm2 (C function), 503  
 vlseg2huv\_mask\_uint64x2xm4 (C function), 503  
 vlseg2huv\_mask\_uint8x2xm1 (C function), 503  
 vlseg2huv\_mask\_uint8x2xm2 (C function), 503  
 vlseg2huv\_mask\_uint8x2xm4 (C function), 503  
 vlseg2huv\_uint16x2xm1 (C function), 502  
 vlseg2huv\_uint16x2xm2 (C function), 502  
 vlseg2huv\_uint16x2xm4 (C function), 502  
 vlseg2huv\_uint32x2xm1 (C function), 502  
 vlseg2huv\_uint32x2xm2 (C function), 502  
 vlseg2huv\_uint32x2xm4 (C function), 502  
 vlseg2huv\_uint64x2xm1 (C function), 502  
 vlseg2huv\_uint64x2xm2 (C function), 502  
 vlseg2huv\_uint64x2xm4 (C function), 502  
 vlseg2huv\_uint8x2xm1 (C function), 502  
 vlseg2huv\_uint8x2xm2 (C function), 502  
 vlseg2huv\_uint8x2xm4 (C function), 502

vlseg2hvv\_int16x2xm1 (C function), 500  
vlseg2hvv\_int16x2xm2 (C function), 500  
vlseg2hvv\_int16x2xm4 (C function), 500  
vlseg2hvv\_int32x2xm1 (C function), 501  
vlseg2hvv\_int32x2xm2 (C function), 501  
vlseg2hvv\_int32x2xm4 (C function), 501  
vlseg2hvv\_int64x2xm1 (C function), 501  
vlseg2hvv\_int64x2xm2 (C function), 501  
vlseg2hvv\_int64x2xm4 (C function), 501  
vlseg2hvv\_int8x2xm1 (C function), 501  
vlseg2hvv\_int8x2xm2 (C function), 501  
vlseg2hvv\_int8x2xm4 (C function), 501  
vlseg2hvv\_mask\_int16x2xm1 (C function), 501  
vlseg2hvv\_mask\_int16x2xm2 (C function), 501  
vlseg2hvv\_mask\_int16x2xm4 (C function), 501  
vlseg2hvv\_mask\_int32x2xm1 (C function), 501  
vlseg2hvv\_mask\_int32x2xm2 (C function), 501  
vlseg2hvv\_mask\_int32x2xm4 (C function), 501  
vlseg2hvv\_mask\_int64x2xm1 (C function), 501  
vlseg2hvv\_mask\_int64x2xm2 (C function), 501  
vlseg2hvv\_mask\_int64x2xm4 (C function), 501  
vlseg2hvv\_mask\_int8x2xm1 (C function), 501  
vlseg2hvv\_mask\_int8x2xm2 (C function), 501  
vlseg2hvv\_mask\_int8x2xm4 (C function), 501  
vlseg2wuv\_mask\_uint16x2xm1 (C function), 505  
vlseg2wuv\_mask\_uint16x2xm2 (C function), 505  
vlseg2wuv\_mask\_uint16x2xm4 (C function), 505  
vlseg2wuv\_mask\_uint32x2xm1 (C function), 505  
vlseg2wuv\_mask\_uint32x2xm2 (C function), 505  
vlseg2wuv\_mask\_uint32x2xm4 (C function), 505  
vlseg2wuv\_mask\_uint64x2xm1 (C function), 505  
vlseg2wuv\_mask\_uint64x2xm2 (C function), 505  
vlseg2wuv\_mask\_uint64x2xm4 (C function), 505  
vlseg2wuv\_mask\_uint8x2xm1 (C function), 505  
vlseg2wuv\_mask\_uint8x2xm2 (C function), 505  
vlseg2wuv\_mask\_uint8x2xm4 (C function), 505  
vlseg2wuv\_uint16x2xm1 (C function), 504  
vlseg2wuv\_uint16x2xm2 (C function), 504  
vlseg2wuv\_uint16x2xm4 (C function), 505  
vlseg2wuv\_uint32x2xm1 (C function), 505  
vlseg2wuv\_uint32x2xm2 (C function), 505  
vlseg2wuv\_uint32x2xm4 (C function), 505  
vlseg2wuv\_uint64x2xm1 (C function), 505  
vlseg2wuv\_uint64x2xm2 (C function), 505  
vlseg2wuv\_uint64x2xm4 (C function), 505  
vlseg2wuv\_uint8x2xm1 (C function), 505  
vlseg2wuv\_uint8x2xm2 (C function), 505  
vlseg2wuv\_uint8x2xm4 (C function), 505  
vlseg2wvv\_int16x2xm1 (C function), 503  
vlseg2wvv\_int16x2xm2 (C function), 503  
vlseg2wvv\_int16x2xm4 (C function), 503  
vlseg2wvv\_int32x2xm1 (C function), 503  
vlseg2wvv\_int32x2xm2 (C function), 503  
vlseg2wvv\_int32x2xm4 (C function), 503  
vlseg2wvv\_int64x2xm1 (C function), 503  
vlseg2wvv\_int64x2xm2 (C function), 503  
vlseg2wvv\_int64x2xm4 (C function), 503  
vlseg2wvv\_int8x2xm1 (C function), 503  
vlseg2wvv\_int8x2xm2 (C function), 503  
vlseg2wvv\_int8x2xm4 (C function), 503  
vlseg2wvv\_mask\_int16x2xm1 (C function), 504  
vlseg2wvv\_mask\_int16x2xm2 (C function), 504  
vlseg2wvv\_mask\_int16x2xm4 (C function), 504  
vlseg2wvv\_mask\_int32x2xm1 (C function), 504  
vlseg2wvv\_mask\_int32x2xm2 (C function), 504  
vlseg2wvv\_mask\_int32x2xm4 (C function), 504  
vlseg2wvv\_mask\_int64x2xm1 (C function), 504  
vlseg2wvv\_mask\_int64x2xm2 (C function), 504  
vlseg2wvv\_mask\_int64x2xm4 (C function), 504  
vlseg2wvv\_mask\_int8x2xm1 (C function), 504  
vlseg2wvv\_mask\_int8x2xm2 (C function), 504  
vlseg2wvv\_mask\_int8x2xm4 (C function), 504  
vlseg3buv\_mask\_uint16x3xm1 (C function), 507  
vlseg3buv\_mask\_uint16x3xm2 (C function), 507  
vlseg3buv\_mask\_uint32x3xm1 (C function), 507  
vlseg3buv\_mask\_uint32x3xm2 (C function), 507  
vlseg3buv\_mask\_uint64x3xm1 (C function), 507  
vlseg3buv\_mask\_uint64x3xm2 (C function), 507  
vlseg3buv\_mask\_uint8x3xm1 (C function), 507  
vlseg3buv\_mask\_uint8x3xm2 (C function), 508  
vlseg3buv\_uint16x3xm1 (C function), 507  
vlseg3buv\_uint16x3xm2 (C function), 507  
vlseg3buv\_uint32x3xm1 (C function), 507  
vlseg3buv\_uint32x3xm2 (C function), 507  
vlseg3buv\_uint64x3xm1 (C function), 507  
vlseg3buv\_uint64x3xm2 (C function), 507  
vlseg3buv\_uint8x3xm1 (C function), 507  
vlseg3buv\_uint8x3xm2 (C function), 507  
vlseg3bv\_int16x3xm1 (C function), 506  
vlseg3bv\_int16x3xm2 (C function), 506  
vlseg3bv\_int32x3xm1 (C function), 506  
vlseg3bv\_int32x3xm2 (C function), 506  
vlseg3bv\_int64x3xm1 (C function), 506  
vlseg3bv\_int64x3xm2 (C function), 506  
vlseg3bv\_int8x3xm1 (C function), 506  
vlseg3bv\_int8x3xm2 (C function), 506  
vlseg3bv\_mask\_int16x3xm1 (C function), 506  
vlseg3bv\_mask\_int16x3xm2 (C function), 506  
vlseg3bv\_mask\_int32x3xm1 (C function), 506  
vlseg3bv\_mask\_int32x3xm2 (C function), 506  
vlseg3bv\_mask\_int64x3xm1 (C function), 506  
vlseg3bv\_mask\_int64x3xm2 (C function), 506  
vlseg3bv\_mask\_int8x3xm1 (C function), 506  
vlseg3bv\_mask\_int8x3xm2 (C function), 506  
vlseg3ev\_float16x3xm1 (C function), 508  
vlseg3ev\_float16x3xm2 (C function), 508  
vlseg3ev\_float32x3xm1 (C function), 508  
vlseg3ev\_float32x3xm2 (C function), 508

vlseg3ev\_float64x3xm1 (C function), 508  
 vlseg3ev\_float64x3xm2 (C function), 508  
 vlseg3ev\_int16x3xm1 (C function), 508  
 vlseg3ev\_int16x3xm2 (C function), 508  
 vlseg3ev\_int32x3xm1 (C function), 508  
 vlseg3ev\_int32x3xm2 (C function), 508  
 vlseg3ev\_int64x3xm1 (C function), 508  
 vlseg3ev\_int64x3xm2 (C function), 508  
 vlseg3ev\_int8x3xm1 (C function), 508  
 vlseg3ev\_int8x3xm2 (C function), 508  
 vlseg3ev\_mask\_float16x3xm1 (C function), 509  
 vlseg3ev\_mask\_float16x3xm2 (C function), 509  
 vlseg3ev\_mask\_float32x3xm1 (C function), 509  
 vlseg3ev\_mask\_float32x3xm2 (C function), 509  
 vlseg3ev\_mask\_float64x3xm1 (C function), 509  
 vlseg3ev\_mask\_float64x3xm2 (C function), 509  
 vlseg3ev\_mask\_int16x3xm1 (C function), 509  
 vlseg3ev\_mask\_int16x3xm2 (C function), 509  
 vlseg3ev\_mask\_int32x3xm1 (C function), 509  
 vlseg3ev\_mask\_int32x3xm2 (C function), 509  
 vlseg3ev\_mask\_int64x3xm1 (C function), 509  
 vlseg3ev\_mask\_int64x3xm2 (C function), 509  
 vlseg3ev\_mask\_int8x3xm1 (C function), 509  
 vlseg3ev\_mask\_int8x3xm2 (C function), 509  
 vlseg3ev\_mask\_uint16x3xm1 (C function), 509  
 vlseg3ev\_mask\_uint16x3xm2 (C function), 509  
 vlseg3ev\_mask\_uint32x3xm1 (C function), 509  
 vlseg3ev\_mask\_uint32x3xm2 (C function), 509  
 vlseg3ev\_mask\_uint64x3xm1 (C function), 509  
 vlseg3ev\_mask\_uint64x3xm2 (C function), 509  
 vlseg3ev\_mask\_uint8x3xm1 (C function), 510  
 vlseg3ev\_mask\_uint8x3xm2 (C function), 510  
 vlseg3ev\_uint16x3xm1 (C function), 508  
 vlseg3ev\_uint16x3xm2 (C function), 508  
 vlseg3ev\_uint32x3xm1 (C function), 508  
 vlseg3ev\_uint32x3xm2 (C function), 508  
 vlseg3ev\_uint64x3xm1 (C function), 508  
 vlseg3ev\_uint64x3xm2 (C function), 508  
 vlseg3ev\_uint8x3xm1 (C function), 508  
 vlseg3ev\_uint8x3xm2 (C function), 508  
 vlseg3huv\_mask\_uint16x3xm1 (C function), 511  
 vlseg3huv\_mask\_uint16x3xm2 (C function), 511  
 vlseg3huv\_mask\_uint32x3xm1 (C function), 511  
 vlseg3huv\_mask\_uint32x3xm2 (C function), 511  
 vlseg3huv\_mask\_uint64x3xm1 (C function), 512  
 vlseg3huv\_mask\_uint64x3xm2 (C function), 512  
 vlseg3huv\_mask\_uint8x3xm1 (C function), 512  
 vlseg3huv\_mask\_uint8x3xm2 (C function), 512  
 vlseg3huv\_uint16x3xm1 (C function), 511  
 vlseg3huv\_uint16x3xm2 (C function), 511  
 vlseg3huv\_uint32x3xm1 (C function), 511  
 vlseg3huv\_uint32x3xm2 (C function), 511  
 vlseg3huv\_uint64x3xm1 (C function), 511  
 vlseg3huv\_uint64x3xm2 (C function), 511  
 vlseg3huv\_uint8x3xm1 (C function), 511  
 vlseg3huv\_uint8x3xm2 (C function), 511  
 vlseg3huv\_mask\_int16x3xm1 (C function), 510  
 vlseg3huv\_mask\_int16x3xm2 (C function), 510  
 vlseg3huv\_mask\_int32x3xm1 (C function), 510  
 vlseg3huv\_mask\_int32x3xm2 (C function), 510  
 vlseg3huv\_mask\_int64x3xm1 (C function), 510  
 vlseg3huv\_mask\_int64x3xm2 (C function), 511  
 vlseg3huv\_mask\_int8x3xm1 (C function), 511  
 vlseg3huv\_mask\_int8x3xm2 (C function), 511  
 vlseg3wuv\_mask\_uint16x3xm1 (C function), 513  
 vlseg3wuv\_mask\_uint16x3xm2 (C function), 513  
 vlseg3wuv\_mask\_uint32x3xm1 (C function), 514  
 vlseg3wuv\_mask\_uint32x3xm2 (C function), 514  
 vlseg3wuv\_mask\_uint64x3xm1 (C function), 514  
 vlseg3wuv\_mask\_uint64x3xm2 (C function), 514  
 vlseg3wuv\_mask\_uint8x3xm1 (C function), 514  
 vlseg3wuv\_mask\_uint8x3xm2 (C function), 514  
 vlseg3wuv\_uint16x3xm1 (C function), 513  
 vlseg3wuv\_uint16x3xm2 (C function), 513  
 vlseg3wuv\_uint32x3xm1 (C function), 513  
 vlseg3wuv\_uint32x3xm2 (C function), 513  
 vlseg3wuv\_uint64x3xm1 (C function), 513  
 vlseg3wuv\_uint64x3xm2 (C function), 513  
 vlseg3wuv\_uint8x3xm1 (C function), 513  
 vlseg3wuv\_uint8x3xm2 (C function), 513  
 vlseg3wv\_int16x3xm1 (C function), 512  
 vlseg3wv\_int16x3xm2 (C function), 512  
 vlseg3wv\_int32x3xm1 (C function), 512  
 vlseg3wv\_int32x3xm2 (C function), 512  
 vlseg3wv\_int64x3xm1 (C function), 512  
 vlseg3wv\_int64x3xm2 (C function), 512  
 vlseg3wv\_int8x3xm1 (C function), 512  
 vlseg3wv\_int8x3xm2 (C function), 512  
 vlseg3wv\_mask\_int16x3xm1 (C function), 512  
 vlseg3wv\_mask\_int16x3xm2 (C function), 512  
 vlseg3wv\_mask\_int32x3xm1 (C function), 512  
 vlseg3wv\_mask\_int32x3xm2 (C function), 513  
 vlseg3wv\_mask\_int64x3xm1 (C function), 513  
 vlseg3wv\_mask\_int64x3xm2 (C function), 513  
 vlseg3wv\_mask\_int8x3xm1 (C function), 513  
 vlseg3wv\_mask\_int8x3xm2 (C function), 513  
 vlseg4buv\_mask\_uint16x4xm1 (C function), 516  
 vlseg4buv\_mask\_uint16x4xm2 (C function), 516  
 vlseg4buv\_mask\_uint32x4xm1 (C function), 516  
 vlseg4buv\_mask\_uint32x4xm2 (C function), 516



vlseg4buv\_mask\_uint64x4xm1 (C function), 516  
vlseg4buv\_mask\_uint64x4xm2 (C function), 516  
vlseg4buv\_mask\_uint8x4xm1 (C function), 516  
vlseg4buv\_mask\_uint8x4xm2 (C function), 516  
vlseg4buv\_uint16x4xm1 (C function), 515  
vlseg4buv\_uint16x4xm2 (C function), 515  
vlseg4buv\_uint32x4xm1 (C function), 515  
vlseg4buv\_uint32x4xm2 (C function), 515  
vlseg4buv\_uint64x4xm1 (C function), 515  
vlseg4buv\_uint64x4xm2 (C function), 515  
vlseg4buv\_uint8x4xm1 (C function), 515  
vlseg4buv\_uint8x4xm2 (C function), 515  
vlseg4bv\_int16x4xm1 (C function), 514  
vlseg4bv\_int16x4xm2 (C function), 514  
vlseg4bv\_int32x4xm1 (C function), 514  
vlseg4bv\_int32x4xm2 (C function), 514  
vlseg4bv\_int64x4xm1 (C function), 514  
vlseg4bv\_int64x4xm2 (C function), 514  
vlseg4bv\_int8x4xm1 (C function), 514  
vlseg4bv\_int8x4xm2 (C function), 514  
vlseg4bv\_mask\_int16x4xm1 (C function), 514  
vlseg4bv\_mask\_int16x4xm2 (C function), 515  
vlseg4bv\_mask\_int32x4xm1 (C function), 515  
vlseg4bv\_mask\_int32x4xm2 (C function), 515  
vlseg4bv\_mask\_int64x4xm1 (C function), 515  
vlseg4bv\_mask\_int64x4xm2 (C function), 515  
vlseg4bv\_mask\_int8x4xm1 (C function), 515  
vlseg4bv\_mask\_int8x4xm2 (C function), 515  
vlseg4ev\_float16x4xm1 (C function), 516  
vlseg4ev\_float16x4xm2 (C function), 516  
vlseg4ev\_float32x4xm1 (C function), 516  
vlseg4ev\_float32x4xm2 (C function), 516  
vlseg4ev\_float64x4xm1 (C function), 516  
vlseg4ev\_float64x4xm2 (C function), 516  
vlseg4ev\_int16x4xm1 (C function), 516  
vlseg4ev\_int16x4xm2 (C function), 516  
vlseg4ev\_int32x4xm1 (C function), 516  
vlseg4ev\_int32x4xm2 (C function), 516  
vlseg4ev\_int64x4xm1 (C function), 516  
vlseg4ev\_int64x4xm2 (C function), 516  
vlseg4ev\_int8x4xm1 (C function), 516  
vlseg4ev\_int8x4xm2 (C function), 517  
vlseg4ev\_mask\_float16x4xm1 (C function), 517  
vlseg4ev\_mask\_float16x4xm2 (C function), 517  
vlseg4ev\_mask\_float32x4xm1 (C function), 517  
vlseg4ev\_mask\_float32x4xm2 (C function), 517  
vlseg4ev\_mask\_float64x4xm1 (C function), 517  
vlseg4ev\_mask\_float64x4xm2 (C function), 517  
vlseg4ev\_mask\_int16x4xm1 (C function), 517  
vlseg4ev\_mask\_int16x4xm2 (C function), 517  
vlseg4ev\_mask\_int32x4xm1 (C function), 517  
vlseg4ev\_mask\_int32x4xm2 (C function), 517  
vlseg4ev\_mask\_int64x4xm1 (C function), 517  
vlseg4ev\_mask\_int64x4xm2 (C function), 517  
vlseg4ev\_mask\_int8x4xm1 (C function), 517  
vlseg4ev\_mask\_int8x4xm2 (C function), 517  
vlseg4ev\_uint16x4xm1 (C function), 517  
vlseg4ev\_uint16x4xm2 (C function), 517  
vlseg4ev\_uint32x4xm1 (C function), 517  
vlseg4ev\_uint32x4xm2 (C function), 517  
vlseg4ev\_uint64x4xm1 (C function), 517  
vlseg4ev\_uint64x4xm2 (C function), 517  
vlseg4ev\_uint8x4xm1 (C function), 517  
vlseg4ev\_uint8x4xm2 (C function), 517  
vlseg4huv\_mask\_uint16x4xm1 (C function), 520  
vlseg4huv\_mask\_uint16x4xm2 (C function), 520  
vlseg4huv\_mask\_uint32x4xm1 (C function), 520  
vlseg4huv\_mask\_uint32x4xm2 (C function), 520  
vlseg4huv\_mask\_uint64x4xm1 (C function), 520  
vlseg4huv\_mask\_uint64x4xm2 (C function), 520  
vlseg4huv\_mask\_uint8x4xm1 (C function), 520  
vlseg4huv\_mask\_uint8x4xm2 (C function), 520  
vlseg4huv\_uint16x4xm1 (C function), 519  
vlseg4huv\_uint16x4xm2 (C function), 519  
vlseg4huv\_uint32x4xm1 (C function), 519  
vlseg4huv\_uint32x4xm2 (C function), 519  
vlseg4huv\_uint64x4xm1 (C function), 519  
vlseg4huv\_uint64x4xm2 (C function), 519  
vlseg4huv\_uint8x4xm1 (C function), 519  
vlseg4huv\_uint8x4xm2 (C function), 519  
vlseg4hv\_int16x4xm1 (C function), 518  
vlseg4hv\_int16x4xm2 (C function), 518  
vlseg4hv\_int32x4xm1 (C function), 518  
vlseg4hv\_int32x4xm2 (C function), 518  
vlseg4hv\_int64x4xm1 (C function), 518  
vlseg4hv\_int64x4xm2 (C function), 518  
vlseg4hv\_int8x4xm1 (C function), 518  
vlseg4hv\_int8x4xm2 (C function), 518  
vlseg4hv\_mask\_int16x4xm1 (C function), 519  
vlseg4hv\_mask\_int16x4xm2 (C function), 519  
vlseg4hv\_mask\_int32x4xm1 (C function), 519  
vlseg4hv\_mask\_int32x4xm2 (C function), 519  
vlseg4hv\_mask\_int64x4xm1 (C function), 519  
vlseg4hv\_mask\_int64x4xm2 (C function), 519  
vlseg4hv\_mask\_int8x4xm1 (C function), 519  
vlseg4hv\_mask\_int8x4xm2 (C function), 519  
vlseg4wuv\_mask\_uint16x4xm1 (C function), 522  
vlseg4wuv\_mask\_uint16x4xm2 (C function), 522  
vlseg4wuv\_mask\_uint32x4xm1 (C function), 522  
vlseg4wuv\_mask\_uint32x4xm2 (C function), 522

vlseg4wuv\_mask\_uint64x4xm1 (C function), 522  
 vlseg4wuv\_mask\_uint64x4xm2 (C function), 522  
 vlseg4wuv\_mask\_uint8x4xm1 (C function), 522  
 vlseg4wuv\_mask\_uint8x4xm2 (C function), 522  
 vlseg4wuv\_uint16x4xm1 (C function), 521  
 vlseg4wuv\_uint16x4xm2 (C function), 521  
 vlseg4wuv\_uint32x4xm1 (C function), 521  
 vlseg4wuv\_uint32x4xm2 (C function), 521  
 vlseg4wuv\_uint64x4xm1 (C function), 521  
 vlseg4wuv\_uint64x4xm2 (C function), 521  
 vlseg4wuv\_uint8x4xm1 (C function), 521  
 vlseg4wuv\_uint8x4xm2 (C function), 522  
 vlseg4wv\_int16x4xm1 (C function), 520  
 vlseg4wv\_int16x4xm2 (C function), 520  
 vlseg4wv\_int32x4xm1 (C function), 520  
 vlseg4wv\_int32x4xm2 (C function), 520  
 vlseg4wv\_int64x4xm1 (C function), 520  
 vlseg4wv\_int64x4xm2 (C function), 520  
 vlseg4wv\_int8x4xm1 (C function), 520  
 vlseg4wv\_int8x4xm2 (C function), 520  
 vlseg4wv\_mask\_int16x4xm1 (C function), 521  
 vlseg4wv\_mask\_int16x4xm2 (C function), 521  
 vlseg4wv\_mask\_int32x4xm1 (C function), 521  
 vlseg4wv\_mask\_int32x4xm2 (C function), 521  
 vlseg4wv\_mask\_int64x4xm1 (C function), 521  
 vlseg4wv\_mask\_int64x4xm2 (C function), 521  
 vlseg4wv\_mask\_int8x4xm1 (C function), 521  
 vlseg4wv\_mask\_int8x4xm2 (C function), 521  
 vlseg5buv\_mask\_uint16x5xm1 (C function), 523  
 vlseg5buv\_mask\_uint32x5xm1 (C function), 523  
 vlseg5buv\_mask\_uint64x5xm1 (C function), 523  
 vlseg5buv\_mask\_uint8x5xm1 (C function), 524  
 vlseg5buv\_uint16x5xm1 (C function), 523  
 vlseg5buv\_uint32x5xm1 (C function), 523  
 vlseg5buv\_uint64x5xm1 (C function), 523  
 vlseg5buv\_uint8x5xm1 (C function), 523  
 vlseg5bv\_int16x5xm1 (C function), 522  
 vlseg5bv\_int32x5xm1 (C function), 522  
 vlseg5bv\_int64x5xm1 (C function), 522  
 vlseg5bv\_int8x5xm1 (C function), 522  
 vlseg5bv\_mask\_int16x5xm1 (C function), 523  
 vlseg5bv\_mask\_int32x5xm1 (C function), 523  
 vlseg5bv\_mask\_int64x5xm1 (C function), 523  
 vlseg5bv\_mask\_int8x5xm1 (C function), 523  
 vlseg5ev\_float16x5xm1 (C function), 524  
 vlseg5ev\_float32x5xm1 (C function), 524  
 vlseg5ev\_float64x5xm1 (C function), 524  
 vlseg5ev\_int16x5xm1 (C function), 524  
 vlseg5ev\_int32x5xm1 (C function), 524  
 vlseg5ev\_int64x5xm1 (C function), 524  
 vlseg5ev\_int8x5xm1 (C function), 524  
 vlseg5ev\_mask\_float16x5xm1 (C function), 524  
 vlseg5ev\_mask\_float32x5xm1 (C function), 524  
 vlseg5ev\_mask\_float64x5xm1 (C function), 524  
 vlseg5ev\_mask\_int16x5xm1 (C function), 524  
 vlseg5ev\_mask\_int32x5xm1 (C function), 525  
 vlseg5ev\_mask\_int64x5xm1 (C function), 525  
 vlseg5ev\_mask\_int8x5xm1 (C function), 525  
 vlseg5ev\_mask\_uint16x5xm1 (C function), 525  
 vlseg5ev\_mask\_uint32x5xm1 (C function), 525  
 vlseg5ev\_mask\_uint64x5xm1 (C function), 525  
 vlseg5ev\_mask\_uint8x5xm1 (C function), 525  
 vlseg5ev\_uint16x5xm1 (C function), 524  
 vlseg5ev\_uint32x5xm1 (C function), 524  
 vlseg5ev\_uint64x5xm1 (C function), 524  
 vlseg5ev\_uint8x5xm1 (C function), 524  
 vlseg5huv\_mask\_uint16x5xm1 (C function), 526  
 vlseg5huv\_mask\_uint32x5xm1 (C function), 526  
 vlseg5huv\_mask\_uint64x5xm1 (C function), 526  
 vlseg5huv\_mask\_uint8x5xm1 (C function), 526  
 vlseg5huv\_uint16x5xm1 (C function), 526  
 vlseg5huv\_uint32x5xm1 (C function), 526  
 vlseg5huv\_uint64x5xm1 (C function), 526  
 vlseg5huv\_uint8x5xm1 (C function), 526  
 vlseg5hv\_int16x5xm1 (C function), 525  
 vlseg5hv\_int32x5xm1 (C function), 525  
 vlseg5hv\_int64x5xm1 (C function), 525  
 vlseg5hv\_int8x5xm1 (C function), 525  
 vlseg5hv\_mask\_int16x5xm1 (C function), 525  
 vlseg5hv\_mask\_int32x5xm1 (C function), 525  
 vlseg5hv\_mask\_int64x5xm1 (C function), 525  
 vlseg5hv\_mask\_int8x5xm1 (C function), 526  
 vlseg5wuv\_mask\_uint16x5xm1 (C function), 528  
 vlseg5wuv\_mask\_uint32x5xm1 (C function), 528  
 vlseg5wuv\_mask\_uint64x5xm1 (C function), 528  
 vlseg5wuv\_mask\_uint8x5xm1 (C function), 528  
 vlseg5wuv\_uint16x5xm1 (C function), 527  
 vlseg5wuv\_uint32x5xm1 (C function), 527  
 vlseg5wuv\_uint64x5xm1 (C function), 527  
 vlseg5wuv\_uint8x5xm1 (C function), 527  
 vlseg5wv\_int16x5xm1 (C function), 527  
 vlseg5wv\_int32x5xm1 (C function), 527  
 vlseg5wv\_int64x5xm1 (C function), 527  
 vlseg5wv\_int8x5xm1 (C function), 527  
 vlseg5wv\_mask\_int16x5xm1 (C function), 527  
 vlseg5wv\_mask\_int32x5xm1 (C function), 527  
 vlseg5wv\_mask\_int64x5xm1 (C function), 527  
 vlseg5wv\_mask\_int8x5xm1 (C function), 527  
 vlseg6buv\_mask\_uint16x6xm1 (C function), 529  
 vlseg6buv\_mask\_uint32x6xm1 (C function), 529  
 vlseg6buv\_mask\_uint64x6xm1 (C function), 529  
 vlseg6buv\_mask\_uint8x6xm1 (C function), 529  
 vlseg6buv\_uint16x6xm1 (C function), 529  
 vlseg6buv\_uint32x6xm1 (C function), 529  
 vlseg6buv\_uint64x6xm1 (C function), 529  
 vlseg6buv\_uint8x6xm1 (C function), 529  
 vlseg6bv\_int16x6xm1 (C function), 528  
 vlseg6bv\_int32x6xm1 (C function), 528



vlseg6bv\_int64x6xm1 (C function), 528  
vlseg6bv\_int8x6xm1 (C function), 528  
vlseg6bv\_mask\_int16x6xm1 (C function), 528  
vlseg6bv\_mask\_int32x6xm1 (C function), 528  
vlseg6bv\_mask\_int64x6xm1 (C function), 528  
vlseg6bv\_mask\_int8x6xm1 (C function), 529  
vlseg6ev\_float16x6xm1 (C function), 530  
vlseg6ev\_float32x6xm1 (C function), 530  
vlseg6ev\_float64x6xm1 (C function), 530  
vlseg6ev\_int16x6xm1 (C function), 530  
vlseg6ev\_int32x6xm1 (C function), 530  
vlseg6ev\_int64x6xm1 (C function), 530  
vlseg6ev\_int8x6xm1 (C function), 530  
vlseg6ev\_mask\_float16x6xm1 (C function), 530  
vlseg6ev\_mask\_float32x6xm1 (C function), 530  
vlseg6ev\_mask\_float64x6xm1 (C function), 530  
vlseg6ev\_mask\_int16x6xm1 (C function), 530  
vlseg6ev\_mask\_int32x6xm1 (C function), 530  
vlseg6ev\_mask\_int64x6xm1 (C function), 530  
vlseg6ev\_mask\_int8x6xm1 (C function), 530  
vlseg6ev\_mask\_uint16x6xm1 (C function), 530  
vlseg6ev\_mask\_uint32x6xm1 (C function), 530  
vlseg6ev\_mask\_uint64x6xm1 (C function), 530  
vlseg6ev\_mask\_uint8x6xm1 (C function), 531  
vlseg6ev\_uint16x6xm1 (C function), 530  
vlseg6ev\_uint32x6xm1 (C function), 530  
vlseg6ev\_uint64x6xm1 (C function), 530  
vlseg6ev\_uint8x6xm1 (C function), 530  
vlseg6huv\_mask\_uint16x6xm1 (C function), 532  
vlseg6huv\_mask\_uint32x6xm1 (C function), 532  
vlseg6huv\_mask\_uint64x6xm1 (C function), 532  
vlseg6huv\_mask\_uint8x6xm1 (C function), 532  
vlseg6huv\_uint16x6xm1 (C function), 532  
vlseg6huv\_uint32x6xm1 (C function), 532  
vlseg6huv\_uint64x6xm1 (C function), 532  
vlseg6huv\_uint8x6xm1 (C function), 532  
vlseg6hv\_int16x6xm1 (C function), 531  
vlseg6hv\_int32x6xm1 (C function), 531  
vlseg6hv\_int64x6xm1 (C function), 531  
vlseg6hv\_int8x6xm1 (C function), 531  
vlseg6hv\_mask\_int16x6xm1 (C function), 531  
vlseg6hv\_mask\_int32x6xm1 (C function), 531  
vlseg6hv\_mask\_int64x6xm1 (C function), 531  
vlseg6hv\_mask\_int8x6xm1 (C function), 531  
vlseg6wuv\_mask\_uint16x6xm1 (C function), 533  
vlseg6wuv\_mask\_uint32x6xm1 (C function), 533  
vlseg6wuv\_mask\_uint64x6xm1 (C function), 533  
vlseg6wuv\_mask\_uint8x6xm1 (C function), 534  
vlseg6wuv\_uint16x6xm1 (C function), 533  
vlseg6wuv\_uint32x6xm1 (C function), 533  
vlseg6wuv\_uint64x6xm1 (C function), 533  
vlseg6wuv\_uint8x6xm1 (C function), 533  
vlseg6wv\_int16x6xm1 (C function), 532  
vlseg6wv\_int32x6xm1 (C function), 532  
vlseg6wv\_int64x6xm1 (C function), 532  
vlseg6wv\_int8x6xm1 (C function), 532  
vlseg6wv\_mask\_int16x6xm1 (C function), 533  
vlseg6wv\_mask\_int32x6xm1 (C function), 533  
vlseg6wv\_mask\_int64x6xm1 (C function), 533  
vlseg6wv\_mask\_int8x6xm1 (C function), 533  
vlseg7buv\_mask\_uint16x7xm1 (C function), 535  
vlseg7buv\_mask\_uint32x7xm1 (C function), 535  
vlseg7buv\_mask\_uint64x7xm1 (C function), 535  
vlseg7buv\_mask\_uint8x7xm1 (C function), 535  
vlseg7buv\_uint16x7xm1 (C function), 535  
vlseg7buv\_uint32x7xm1 (C function), 535  
vlseg7buv\_uint64x7xm1 (C function), 535  
vlseg7buv\_uint8x7xm1 (C function), 535  
vlseg7bv\_int16x7xm1 (C function), 534  
vlseg7bv\_int32x7xm1 (C function), 534  
vlseg7bv\_int64x7xm1 (C function), 534  
vlseg7bv\_int8x7xm1 (C function), 534  
vlseg7bv\_mask\_int16x7xm1 (C function), 534  
vlseg7bv\_mask\_int32x7xm1 (C function), 534  
vlseg7bv\_mask\_int64x7xm1 (C function), 534  
vlseg7bv\_mask\_int8x7xm1 (C function), 534  
vlseg7ev\_float16x7xm1 (C function), 535  
vlseg7ev\_float32x7xm1 (C function), 535  
vlseg7ev\_float64x7xm1 (C function), 535  
vlseg7ev\_int16x7xm1 (C function), 535  
vlseg7ev\_int32x7xm1 (C function), 535  
vlseg7ev\_int64x7xm1 (C function), 536  
vlseg7ev\_int8x7xm1 (C function), 536  
vlseg7ev\_mask\_float16x7xm1 (C function), 536  
vlseg7ev\_mask\_float32x7xm1 (C function), 536  
vlseg7ev\_mask\_float64x7xm1 (C function), 536  
vlseg7ev\_mask\_int16x7xm1 (C function), 536  
vlseg7ev\_mask\_int32x7xm1 (C function), 536  
vlseg7ev\_mask\_int64x7xm1 (C function), 536  
vlseg7ev\_mask\_int8x7xm1 (C function), 536  
vlseg7ev\_mask\_uint16x7xm1 (C function), 536  
vlseg7ev\_mask\_uint32x7xm1 (C function), 536  
vlseg7ev\_mask\_uint64x7xm1 (C function), 536  
vlseg7ev\_mask\_uint8x7xm1 (C function), 536  
vlseg7ev\_uint16x7xm1 (C function), 536  
vlseg7ev\_uint32x7xm1 (C function), 536  
vlseg7ev\_uint64x7xm1 (C function), 536  
vlseg7ev\_uint8x7xm1 (C function), 536  
vlseg7huv\_mask\_uint16x7xm1 (C function), 538  
vlseg7huv\_mask\_uint32x7xm1 (C function), 538  
vlseg7huv\_mask\_uint64x7xm1 (C function), 538  
vlseg7huv\_mask\_uint8x7xm1 (C function), 538  
vlseg7huv\_uint16x7xm1 (C function), 537  
vlseg7huv\_uint32x7xm1 (C function), 537  
vlseg7huv\_uint64x7xm1 (C function), 537  
vlseg7huv\_uint8x7xm1 (C function), 537  
vlseg7hv\_int16x7xm1 (C function), 537  
vlseg7hv\_int32x7xm1 (C function), 537

vlseg7hvv\_int64x7xm1 (C function), 537  
 vlseg7hvv\_int8x7xm1 (C function), 537  
 vlseg7hvv\_mask\_int16x7xm1 (C function), 537  
 vlseg7hvv\_mask\_int32x7xm1 (C function), 537  
 vlseg7hvv\_mask\_int64x7xm1 (C function), 537  
 vlseg7hvv\_mask\_int8x7xm1 (C function), 537  
 vlseg7wuv\_mask\_uint16x7xm1 (C function), 539  
 vlseg7wuv\_mask\_uint32x7xm1 (C function), 539  
 vlseg7wuv\_mask\_uint64x7xm1 (C function), 539  
 vlseg7wuv\_mask\_uint8x7xm1 (C function), 539  
 vlseg7wuv\_uint16x7xm1 (C function), 539  
 vlseg7wuv\_uint32x7xm1 (C function), 539  
 vlseg7wuv\_uint64x7xm1 (C function), 539  
 vlseg7wuv\_uint8x7xm1 (C function), 539  
 vlseg7wv\_int16x7xm1 (C function), 538  
 vlseg7wv\_int32x7xm1 (C function), 538  
 vlseg7wv\_int64x7xm1 (C function), 538  
 vlseg7wv\_int8x7xm1 (C function), 538  
 vlseg7wv\_mask\_int16x7xm1 (C function), 538  
 vlseg7wv\_mask\_int32x7xm1 (C function), 538  
 vlseg7wv\_mask\_int64x7xm1 (C function), 538  
 vlseg7wv\_mask\_int8x7xm1 (C function), 539  
 vlseg8buv\_mask\_uint16x8xm1 (C function), 541  
 vlseg8buv\_mask\_uint32x8xm1 (C function), 541  
 vlseg8buv\_mask\_uint64x8xm1 (C function), 541  
 vlseg8buv\_mask\_uint8x8xm1 (C function), 541  
 vlseg8buv\_uint16x8xm1 (C function), 540  
 vlseg8buv\_uint32x8xm1 (C function), 540  
 vlseg8buv\_uint64x8xm1 (C function), 540  
 vlseg8buv\_uint8x8xm1 (C function), 540  
 vlseg8bv\_int16x8xm1 (C function), 540  
 vlseg8bv\_int32x8xm1 (C function), 540  
 vlseg8bv\_int64x8xm1 (C function), 540  
 vlseg8bv\_int8x8xm1 (C function), 540  
 vlseg8bv\_mask\_int16x8xm1 (C function), 540  
 vlseg8bv\_mask\_int32x8xm1 (C function), 540  
 vlseg8bv\_mask\_int64x8xm1 (C function), 540  
 vlseg8bv\_mask\_int8x8xm1 (C function), 540  
 vlseg8ev\_float16x8xm1 (C function), 541  
 vlseg8ev\_float32x8xm1 (C function), 541  
 vlseg8ev\_float64x8xm1 (C function), 541  
 vlseg8ev\_int16x8xm1 (C function), 541  
 vlseg8ev\_int32x8xm1 (C function), 541  
 vlseg8ev\_int64x8xm1 (C function), 541  
 vlseg8ev\_int8x8xm1 (C function), 541  
 vlseg8ev\_mask\_float16x8xm1 (C function), 542  
 vlseg8ev\_mask\_float32x8xm1 (C function), 542  
 vlseg8ev\_mask\_float64x8xm1 (C function), 542  
 vlseg8ev\_mask\_int16x8xm1 (C function), 542  
 vlseg8ev\_mask\_int32x8xm1 (C function), 542  
 vlseg8ev\_mask\_int64x8xm1 (C function), 542  
 vlseg8ev\_mask\_int8x8xm1 (C function), 542  
 vlseg8ev\_mask\_uint16x8xm1 (C function), 542  
 vlseg8ev\_mask\_uint32x8xm1 (C function), 542  
 vlseg8ev\_mask\_uint64x8xm1 (C function), 542  
 vlseg8ev\_mask\_uint8x8xm1 (C function), 542  
 vlseg8huv\_mask\_uint16x8xm1 (C function), 543  
 vlseg8huv\_mask\_uint32x8xm1 (C function), 543  
 vlseg8huv\_mask\_uint64x8xm1 (C function), 544  
 vlseg8huv\_mask\_uint8x8xm1 (C function), 544  
 vlseg8huv\_uint16x8xm1 (C function), 543  
 vlseg8huv\_uint32x8xm1 (C function), 543  
 vlseg8huv\_uint64x8xm1 (C function), 543  
 vlseg8huv\_uint8x8xm1 (C function), 543  
 vlseg8hv\_int16x8xm1 (C function), 542  
 vlseg8hv\_int32x8xm1 (C function), 542  
 vlseg8hv\_int64x8xm1 (C function), 542  
 vlseg8hv\_int8x8xm1 (C function), 543  
 vlseg8hv\_mask\_int16x8xm1 (C function), 543  
 vlseg8hv\_mask\_int32x8xm1 (C function), 543  
 vlseg8hv\_mask\_int64x8xm1 (C function), 543  
 vlseg8hv\_mask\_int8x8xm1 (C function), 543  
 vlseg8wuv\_mask\_uint16x8xm1 (C function), 545  
 vlseg8wuv\_mask\_uint32x8xm1 (C function), 545  
 vlseg8wuv\_mask\_uint64x8xm1 (C function), 545  
 vlseg8wuv\_mask\_uint8x8xm1 (C function), 545  
 vlseg8wuv\_uint16x8xm1 (C function), 545  
 vlseg8wuv\_uint32x8xm1 (C function), 545  
 vlseg8wuv\_uint64x8xm1 (C function), 545  
 vlseg8wuv\_uint8x8xm1 (C function), 545  
 vlseg8wv\_int16x8xm1 (C function), 544  
 vlseg8wv\_int32x8xm1 (C function), 544  
 vlseg8wv\_int64x8xm1 (C function), 544  
 vlseg8wv\_int8x8xm1 (C function), 544  
 vlseg8wv\_mask\_int16x8xm1 (C function), 544  
 vlseg8wv\_mask\_int32x8xm1 (C function), 544  
 vlseg8wv\_mask\_int64x8xm1 (C function), 544  
 vlseg8wv\_mask\_int8x8xm1 (C function), 544  
 vlsev\_float16xm1 (C function), 424  
 vlsev\_float16xm2 (C function), 424  
 vlsev\_float16xm4 (C function), 424  
 vlsev\_float16xm8 (C function), 424  
 vlsev\_float32xm1 (C function), 424  
 vlsev\_float32xm2 (C function), 424  
 vlsev\_float32xm4 (C function), 424  
 vlsev\_float32xm8 (C function), 424  
 vlsev\_float64xm1 (C function), 424  
 vlsev\_float64xm2 (C function), 424  
 vlsev\_float64xm4 (C function), 424  
 vlsev\_float64xm8 (C function), 424  
 vlsev\_int16xm1 (C function), 424  
 vlsev\_int16xm2 (C function), 424  
 vlsev\_int16xm4 (C function), 424  
 vlsev\_int16xm8 (C function), 424

vlsev\_int32xm1 (C function), 424  
vlsev\_int32xm2 (C function), 424  
vlsev\_int32xm4 (C function), 424  
vlsev\_int32xm8 (C function), 424  
vlsev\_int64xm1 (C function), 424  
vlsev\_int64xm2 (C function), 424  
vlsev\_int64xm4 (C function), 424  
vlsev\_int64xm8 (C function), 424  
vlsev\_int8xm1 (C function), 425  
vlsev\_int8xm2 (C function), 425  
vlsev\_int8xm4 (C function), 425  
vlsev\_int8xm8 (C function), 425  
vlsev\_mask\_float16xm1 (C function), 425  
vlsev\_mask\_float16xm2 (C function), 425  
vlsev\_mask\_float16xm4 (C function), 425  
vlsev\_mask\_float16xm8 (C function), 425  
vlsev\_mask\_float32xm1 (C function), 425  
vlsev\_mask\_float32xm2 (C function), 425  
vlsev\_mask\_float32xm4 (C function), 425  
vlsev\_mask\_float32xm8 (C function), 426  
vlsev\_mask\_float64xm1 (C function), 426  
vlsev\_mask\_float64xm2 (C function), 426  
vlsev\_mask\_float64xm4 (C function), 426  
vlsev\_mask\_float64xm8 (C function), 426  
vlsev\_mask\_int16xm1 (C function), 426  
vlsev\_mask\_int16xm2 (C function), 426  
vlsev\_mask\_int16xm4 (C function), 426  
vlsev\_mask\_int16xm8 (C function), 426  
vlsev\_mask\_int32xm1 (C function), 426  
vlsev\_mask\_int32xm2 (C function), 426  
vlsev\_mask\_int32xm4 (C function), 426  
vlsev\_mask\_int32xm8 (C function), 426  
vlsev\_mask\_int64xm1 (C function), 426  
vlsev\_mask\_int64xm2 (C function), 426  
vlsev\_mask\_int64xm4 (C function), 426  
vlsev\_mask\_int64xm8 (C function), 426  
vlsev\_mask\_int8xm1 (C function), 426  
vlsev\_mask\_int8xm2 (C function), 426  
vlsev\_mask\_int8xm4 (C function), 426  
vlsev\_mask\_int8xm8 (C function), 426  
vlsev\_mask\_uint16xm1 (C function), 426  
vlsev\_mask\_uint16xm2 (C function), 426  
vlsev\_mask\_uint16xm4 (C function), 427  
vlsev\_mask\_uint16xm8 (C function), 427  
vlsev\_mask\_uint32xm1 (C function), 427  
vlsev\_mask\_uint32xm2 (C function), 427  
vlsev\_mask\_uint32xm4 (C function), 427  
vlsev\_mask\_uint32xm8 (C function), 427  
vlsev\_mask\_uint64xm1 (C function), 427  
vlsev\_mask\_uint64xm2 (C function), 427  
vlsev\_mask\_uint64xm4 (C function), 427  
vlsev\_mask\_uint64xm8 (C function), 427  
vlsev\_mask\_uint8xm1 (C function), 427  
vlsev\_mask\_uint8xm2 (C function), 427

vlsev\_mask\_uint8xm4 (C function), 427  
vlsev\_mask\_uint8xm8 (C function), 427  
vlsev\_uint16xm1 (C function), 425  
vlsev\_uint16xm2 (C function), 425  
vlsev\_uint16xm4 (C function), 425  
vlsev\_uint16xm8 (C function), 425  
vlsev\_uint32xm1 (C function), 425  
vlsev\_uint32xm2 (C function), 425  
vlsev\_uint32xm4 (C function), 425  
vlsev\_uint32xm8 (C function), 425  
vlsev\_uint64xm1 (C function), 425  
vlsev\_uint64xm2 (C function), 425  
vlsev\_uint64xm4 (C function), 425  
vlsev\_uint64xm8 (C function), 425  
vlsev\_uint8xm1 (C function), 425  
vlsev\_uint8xm2 (C function), 425  
vlsev\_uint8xm4 (C function), 425  
vlsev\_uint8xm8 (C function), 425  
vlshuv\_mask\_uint16xm1 (C function), 430  
vlshuv\_mask\_uint16xm2 (C function), 430  
vlshuv\_mask\_uint16xm4 (C function), 430  
vlshuv\_mask\_uint16xm8 (C function), 430  
vlshuv\_mask\_uint32xm1 (C function), 430  
vlshuv\_mask\_uint32xm2 (C function), 430  
vlshuv\_mask\_uint32xm4 (C function), 430  
vlshuv\_mask\_uint32xm8 (C function), 430  
vlshuv\_mask\_uint64xm1 (C function), 430  
vlshuv\_mask\_uint64xm2 (C function), 430  
vlshuv\_mask\_uint64xm4 (C function), 430  
vlshuv\_mask\_uint64xm8 (C function), 430  
vlshuv\_mask\_uint8xm1 (C function), 430  
vlshuv\_mask\_uint8xm2 (C function), 430  
vlshuv\_mask\_uint8xm4 (C function), 430  
vlshuv\_mask\_uint8xm8 (C function), 430  
vlshuv\_uint16xm1 (C function), 429  
vlshuv\_uint16xm2 (C function), 429  
vlshuv\_uint16xm4 (C function), 429  
vlshuv\_uint16xm8 (C function), 429  
vlshuv\_uint32xm1 (C function), 429  
vlshuv\_uint32xm2 (C function), 429  
vlshuv\_uint32xm4 (C function), 429  
vlshuv\_uint32xm8 (C function), 429  
vlshuv\_uint64xm1 (C function), 429  
vlshuv\_uint64xm2 (C function), 429  
vlshuv\_uint64xm4 (C function), 429  
vlshuv\_uint64xm8 (C function), 429  
vlshuv\_uint8xm1 (C function), 429  
vlshuv\_uint8xm2 (C function), 429  
vlshuv\_uint8xm4 (C function), 429  
vlshuv\_uint8xm8 (C function), 429  
vlshv\_int16xm1 (C function), 427  
vlshv\_int16xm2 (C function), 427  
vlshv\_int16xm4 (C function), 427  
vlshv\_int16xm8 (C function), 427

vlshv\_int32xm1 (C function), 428  
 vlshv\_int32xm2 (C function), 428  
 vlshv\_int32xm4 (C function), 428  
 vlshv\_int32xm8 (C function), 428  
 vlshv\_int64xm1 (C function), 428  
 vlshv\_int64xm2 (C function), 428  
 vlshv\_int64xm4 (C function), 428  
 vlshv\_int64xm8 (C function), 428  
 vlshv\_int8xm1 (C function), 428  
 vlshv\_int8xm2 (C function), 428  
 vlshv\_int8xm4 (C function), 428  
 vlshv\_int8xm8 (C function), 428  
 vlshv\_mask\_int16xm1 (C function), 428  
 vlshv\_mask\_int16xm2 (C function), 428  
 vlshv\_mask\_int16xm4 (C function), 428  
 vlshv\_mask\_int16xm8 (C function), 428  
 vlshv\_mask\_int32xm1 (C function), 428  
 vlshv\_mask\_int32xm2 (C function), 428  
 vlshv\_mask\_int32xm4 (C function), 428  
 vlshv\_mask\_int32xm8 (C function), 428  
 vlshv\_mask\_int64xm1 (C function), 428  
 vlshv\_mask\_int64xm2 (C function), 428  
 vlshv\_mask\_int64xm4 (C function), 428  
 vlshv\_mask\_int64xm8 (C function), 428  
 vlshv\_mask\_int8xm1 (C function), 429  
 vlshv\_mask\_int8xm2 (C function), 429  
 vlshv\_mask\_int8xm4 (C function), 429  
 vlshv\_mask\_int8xm8 (C function), 429  
 vlsseg2buv\_mask\_uint16x2xm1 (C function), 547  
 vlsseg2buv\_mask\_uint16x2xm2 (C function), 547  
 vlsseg2buv\_mask\_uint16x2xm4 (C function), 548  
 vlsseg2buv\_mask\_uint32x2xm1 (C function), 548  
 vlsseg2buv\_mask\_uint32x2xm2 (C function), 548  
 vlsseg2buv\_mask\_uint32x2xm4 (C function), 548  
 vlsseg2buv\_mask\_uint64x2xm1 (C function), 548  
 vlsseg2buv\_mask\_uint64x2xm2 (C function), 548  
 vlsseg2buv\_mask\_uint64x2xm4 (C function), 548  
 vlsseg2buv\_mask\_uint8x2xm1 (C function), 548  
 vlsseg2buv\_mask\_uint8x2xm2 (C function), 548  
 vlsseg2buv\_mask\_uint8x2xm4 (C function), 548  
 vlsseg2buv\_uint16x2xm1 (C function), 547  
 vlsseg2buv\_uint16x2xm2 (C function), 547  
 vlsseg2buv\_uint16x2xm4 (C function), 547  
 vlsseg2buv\_uint32x2xm1 (C function), 547  
 vlsseg2buv\_uint32x2xm2 (C function), 547  
 vlsseg2buv\_uint32x2xm4 (C function), 547  
 vlsseg2buv\_uint64x2xm1 (C function), 547  
 vlsseg2buv\_uint64x2xm2 (C function), 547  
 vlsseg2buv\_uint64x2xm4 (C function), 547  
 vlsseg2buv\_uint8x2xm1 (C function), 547  
 vlsseg2buv\_uint8x2xm2 (C function), 547  
 vlsseg2buv\_uint8x2xm4 (C function), 547  
 vlsseg2bv\_int16x2xm1 (C function), 545  
 vlsseg2bv\_int16x2xm2 (C function), 545  
 vlsseg2bv\_int16x2xm4 (C function), 545  
 vlsseg2bv\_int32x2xm1 (C function), 545  
 vlsseg2bv\_int32x2xm2 (C function), 545  
 vlsseg2bv\_int32x2xm4 (C function), 546  
 vlsseg2bv\_int64x2xm1 (C function), 546  
 vlsseg2bv\_int64x2xm2 (C function), 546  
 vlsseg2bv\_int64x2xm4 (C function), 546  
 vlsseg2bv\_int8x2xm1 (C function), 546  
 vlsseg2bv\_int8x2xm2 (C function), 546  
 vlsseg2bv\_int8x2xm4 (C function), 546  
 vlsseg2bv\_mask\_int16x2xm1 (C function), 546  
 vlsseg2bv\_mask\_int16x2xm2 (C function), 546  
 vlsseg2bv\_mask\_int16x2xm4 (C function), 546  
 vlsseg2bv\_mask\_int32x2xm1 (C function), 546  
 vlsseg2bv\_mask\_int32x2xm2 (C function), 546  
 vlsseg2bv\_mask\_int32x2xm4 (C function), 546  
 vlsseg2bv\_mask\_int64x2xm1 (C function), 546  
 vlsseg2bv\_mask\_int64x2xm2 (C function), 546  
 vlsseg2bv\_mask\_int64x2xm4 (C function), 546  
 vlsseg2bv\_mask\_int8x2xm1 (C function), 546  
 vlsseg2bv\_mask\_int8x2xm2 (C function), 546  
 vlsseg2bv\_mask\_int8x2xm4 (C function), 546  
 vlsseg2ev\_float16x2xm1 (C function), 548  
 vlsseg2ev\_float16x2xm2 (C function), 548  
 vlsseg2ev\_float16x2xm4 (C function), 548  
 vlsseg2ev\_float32x2xm1 (C function), 548  
 vlsseg2ev\_float32x2xm2 (C function), 548  
 vlsseg2ev\_float32x2xm4 (C function), 549  
 vlsseg2ev\_float64x2xm1 (C function), 549  
 vlsseg2ev\_float64x2xm2 (C function), 549  
 vlsseg2ev\_float64x2xm4 (C function), 549  
 vlsseg2ev\_int16x2xm1 (C function), 549  
 vlsseg2ev\_int16x2xm2 (C function), 549  
 vlsseg2ev\_int16x2xm4 (C function), 549  
 vlsseg2ev\_int32x2xm1 (C function), 549  
 vlsseg2ev\_int32x2xm2 (C function), 549  
 vlsseg2ev\_int32x2xm4 (C function), 549  
 vlsseg2ev\_int64x2xm1 (C function), 549  
 vlsseg2ev\_int64x2xm2 (C function), 549  
 vlsseg2ev\_int64x2xm4 (C function), 549  
 vlsseg2ev\_int8x2xm1 (C function), 549  
 vlsseg2ev\_int8x2xm2 (C function), 549  
 vlsseg2ev\_int8x2xm4 (C function), 549  
 vlsseg2ev\_mask\_float16x2xm1 (C function), 550  
 vlsseg2ev\_mask\_float16x2xm2 (C function), 550  
 vlsseg2ev\_mask\_float16x2xm4 (C function), 550  
 vlsseg2ev\_mask\_float32x2xm1 (C function), 550  
 vlsseg2ev\_mask\_float32x2xm2 (C function), 550  
 vlsseg2ev\_mask\_float32x2xm4 (C function), 550  
 vlsseg2ev\_mask\_float64x2xm1 (C function), 550  
 vlsseg2ev\_mask\_float64x2xm2 (C function), 550  
 vlsseg2ev\_mask\_float64x2xm4 (C function), 550  
 vlsseg2ev\_mask\_int16x2xm1 (C function), 550  
 vlsseg2ev\_mask\_int16x2xm2 (C function), 550



[illegible]

vlsseg2wv\_int16x2xm4 (C function), 555  
 vlsseg2wv\_int32x2xm1 (C function), 555  
 vlsseg2wv\_int32x2xm2 (C function), 555  
 vlsseg2wv\_int32x2xm4 (C function), 555  
 vlsseg2wv\_int64x2xm1 (C function), 555  
 vlsseg2wv\_int64x2xm2 (C function), 555  
 vlsseg2wv\_int64x2xm4 (C function), 555  
 vlsseg2wv\_int8x2xm1 (C function), 555  
 vlsseg2wv\_int8x2xm2 (C function), 555  
 vlsseg2wv\_int8x2xm4 (C function), 555  
 vlsseg2wv\_mask\_int16x2xm1 (C function), 555  
 vlsseg2wv\_mask\_int16x2xm2 (C function), 555  
 vlsseg2wv\_mask\_int16x2xm4 (C function), 555  
 vlsseg2wv\_mask\_int32x2xm1 (C function), 555  
 vlsseg2wv\_mask\_int32x2xm2 (C function), 555  
 vlsseg2wv\_mask\_int32x2xm4 (C function), 555  
 vlsseg2wv\_mask\_int64x2xm1 (C function), 555  
 vlsseg2wv\_mask\_int64x2xm2 (C function), 555  
 vlsseg2wv\_mask\_int64x2xm4 (C function), 555  
 vlsseg2wv\_mask\_int8x2xm1 (C function), 556  
 vlsseg2wv\_mask\_int8x2xm2 (C function), 556  
 vlsseg2wv\_mask\_int8x2xm4 (C function), 556  
 vlsseg3buv\_mask\_uint16x3xm1 (C function), 559  
 vlsseg3buv\_mask\_uint16x3xm2 (C function), 559  
 vlsseg3buv\_mask\_uint32x3xm1 (C function), 559  
 vlsseg3buv\_mask\_uint32x3xm2 (C function), 559  
 vlsseg3buv\_mask\_uint64x3xm1 (C function), 559  
 vlsseg3buv\_mask\_uint64x3xm2 (C function), 559  
 vlsseg3buv\_mask\_uint8x3xm1 (C function), 560  
 vlsseg3buv\_mask\_uint8x3xm2 (C function), 560  
 vlsseg3buv\_uint16x3xm1 (C function), 559  
 vlsseg3buv\_uint16x3xm2 (C function), 559  
 vlsseg3buv\_uint32x3xm1 (C function), 559  
 vlsseg3buv\_uint32x3xm2 (C function), 559  
 vlsseg3buv\_uint64x3xm1 (C function), 559  
 vlsseg3buv\_uint64x3xm2 (C function), 559  
 vlsseg3buv\_uint8x3xm1 (C function), 559  
 vlsseg3buv\_uint8x3xm2 (C function), 559  
 vlsseg3bv\_int16x3xm1 (C function), 558  
 vlsseg3bv\_int16x3xm2 (C function), 558  
 vlsseg3bv\_int32x3xm1 (C function), 558  
 vlsseg3bv\_int32x3xm2 (C function), 558  
 vlsseg3bv\_int64x3xm1 (C function), 558  
 vlsseg3bv\_int64x3xm2 (C function), 558  
 vlsseg3bv\_int8x3xm1 (C function), 558  
 vlsseg3bv\_int8x3xm2 (C function), 558  
 vlsseg3bv\_mask\_int16x3xm1 (C function), 558  
 vlsseg3bv\_mask\_int16x3xm2 (C function), 558  
 vlsseg3bv\_mask\_int32x3xm1 (C function), 558  
 vlsseg3bv\_mask\_int32x3xm2 (C function), 558  
 vlsseg3bv\_mask\_int64x3xm1 (C function), 558  
 vlsseg3bv\_mask\_int64x3xm2 (C function), 558  
 vlsseg3bv\_mask\_int8x3xm1 (C function), 558  
 vlsseg3bv\_mask\_int8x3xm2 (C function), 558  
 vlsseg3ev\_float16x3xm1 (C function), 560  
 vlsseg3ev\_float16x3xm2 (C function), 560  
 vlsseg3ev\_float32x3xm1 (C function), 560  
 vlsseg3ev\_float32x3xm2 (C function), 560  
 vlsseg3ev\_float64x3xm1 (C function), 560  
 vlsseg3ev\_float64x3xm2 (C function), 560  
 vlsseg3ev\_int16x3xm1 (C function), 560  
 vlsseg3ev\_int16x3xm2 (C function), 560  
 vlsseg3ev\_int32x3xm1 (C function), 560  
 vlsseg3ev\_int32x3xm2 (C function), 560  
 vlsseg3ev\_int64x3xm1 (C function), 560  
 vlsseg3ev\_int64x3xm2 (C function), 560  
 vlsseg3ev\_int8x3xm1 (C function), 560  
 vlsseg3ev\_int8x3xm2 (C function), 560  
 vlsseg3ev\_mask\_float16x3xm1 (C function), 561  
 vlsseg3ev\_mask\_float16x3xm2 (C function), 561  
 vlsseg3ev\_mask\_float32x3xm1 (C function), 561  
 vlsseg3ev\_mask\_float32x3xm2 (C function), 561  
 vlsseg3ev\_mask\_float64x3xm1 (C function), 561  
 vlsseg3ev\_mask\_float64x3xm2 (C function), 561  
 vlsseg3ev\_mask\_int16x3xm1 (C function), 561  
 vlsseg3ev\_mask\_int16x3xm2 (C function), 561  
 vlsseg3ev\_mask\_int32x3xm1 (C function), 561  
 vlsseg3ev\_mask\_int32x3xm2 (C function), 561  
 vlsseg3ev\_mask\_int64x3xm1 (C function), 561  
 vlsseg3ev\_mask\_int64x3xm2 (C function), 561  
 vlsseg3ev\_mask\_int8x3xm1 (C function), 561  
 vlsseg3ev\_mask\_int8x3xm2 (C function), 561  
 vlsseg3ev\_mask\_uint16x3xm1 (C function), 561  
 vlsseg3ev\_mask\_uint16x3xm2 (C function), 561  
 vlsseg3ev\_mask\_uint32x3xm1 (C function), 562  
 vlsseg3ev\_mask\_uint32x3xm2 (C function), 562  
 vlsseg3ev\_mask\_uint64x3xm1 (C function), 562  
 vlsseg3ev\_mask\_uint64x3xm2 (C function), 562  
 vlsseg3ev\_mask\_uint8x3xm1 (C function), 562  
 vlsseg3ev\_mask\_uint8x3xm2 (C function), 562  
 vlsseg3ev\_uint16x3xm1 (C function), 560  
 vlsseg3ev\_uint16x3xm2 (C function), 560  
 vlsseg3ev\_uint32x3xm1 (C function), 560  
 vlsseg3ev\_uint32x3xm2 (C function), 560  
 vlsseg3ev\_uint64x3xm1 (C function), 560  
 vlsseg3ev\_uint64x3xm2 (C function), 561  
 vlsseg3ev\_uint8x3xm1 (C function), 561  
 vlsseg3ev\_uint8x3xm2 (C function), 561  
 vlsseg3huv\_mask\_uint16x3xm1 (C function), 564  
 vlsseg3huv\_mask\_uint16x3xm2 (C function), 564  
 vlsseg3huv\_mask\_uint32x3xm1 (C function), 564  
 vlsseg3huv\_mask\_uint32x3xm2 (C function), 564  
 vlsseg3huv\_mask\_uint64x3xm1 (C function), 564  
 vlsseg3huv\_mask\_uint64x3xm2 (C function), 564  
 vlsseg3huv\_mask\_uint8x3xm1 (C function), 564  
 vlsseg3huv\_mask\_uint8x3xm2 (C function), 564  
 vlsseg3huv\_uint16x3xm1 (C function), 563  
 vlsseg3huv\_uint16x3xm2 (C function), 563



vlssseg3huv\_uint32x3xm1 (C function), 563  
vlssseg3huv\_uint32x3xm2 (C function), 563  
vlssseg3huv\_uint64x3xm1 (C function), 563  
vlssseg3huv\_uint64x3xm2 (C function), 563  
vlssseg3huv\_uint8x3xm1 (C function), 563  
vlssseg3huv\_uint8x3xm2 (C function), 563  
vlssseg3hv\_int16x3xm1 (C function), 562  
vlssseg3hv\_int16x3xm2 (C function), 562  
vlssseg3hv\_int32x3xm1 (C function), 562  
vlssseg3hv\_int32x3xm2 (C function), 562  
vlssseg3hv\_int64x3xm1 (C function), 562  
vlssseg3hv\_int64x3xm2 (C function), 562  
vlssseg3hv\_int8x3xm1 (C function), 562  
vlssseg3hv\_int8x3xm2 (C function), 562  
vlssseg3hv\_mask\_int16x3xm1 (C function), 563  
vlssseg3hv\_mask\_int16x3xm2 (C function), 563  
vlssseg3hv\_mask\_int32x3xm1 (C function), 563  
vlssseg3hv\_mask\_int32x3xm2 (C function), 563  
vlssseg3hv\_mask\_int64x3xm1 (C function), 563  
vlssseg3hv\_mask\_int64x3xm2 (C function), 563  
vlssseg3hv\_mask\_int8x3xm1 (C function), 563  
vlssseg3hv\_mask\_int8x3xm2 (C function), 563  
vlssseg3wuv\_mask\_uint16x3xm1 (C function), 566  
vlssseg3wuv\_mask\_uint16x3xm2 (C function), 566  
vlssseg3wuv\_mask\_uint32x3xm1 (C function), 566  
vlssseg3wuv\_mask\_uint32x3xm2 (C function), 566  
vlssseg3wuv\_mask\_uint64x3xm1 (C function), 566  
vlssseg3wuv\_mask\_uint64x3xm2 (C function), 566  
vlssseg3wuv\_mask\_uint8x3xm1 (C function), 566  
vlssseg3wuv\_mask\_uint8x3xm2 (C function), 566  
vlssseg3wuv\_uint16x3xm1 (C function), 565  
vlssseg3wuv\_uint16x3xm2 (C function), 566  
vlssseg3wuv\_uint32x3xm1 (C function), 566  
vlssseg3wuv\_uint32x3xm2 (C function), 566  
vlssseg3wuv\_uint64x3xm1 (C function), 566  
vlssseg3wuv\_uint64x3xm2 (C function), 566  
vlssseg3wuv\_uint8x3xm1 (C function), 566  
vlssseg3wuv\_uint8x3xm2 (C function), 566  
vlssseg3wv\_int16x3xm1 (C function), 564  
vlssseg3wv\_int16x3xm2 (C function), 564  
vlssseg3wv\_int32x3xm1 (C function), 564  
vlssseg3wv\_int32x3xm2 (C function), 565  
vlssseg3wv\_int64x3xm1 (C function), 565  
vlssseg3wv\_int64x3xm2 (C function), 565  
vlssseg3wv\_int8x3xm1 (C function), 565  
vlssseg3wv\_int8x3xm2 (C function), 565  
vlssseg3wv\_mask\_int16x3xm1 (C function), 565  
vlssseg3wv\_mask\_int16x3xm2 (C function), 565  
vlssseg3wv\_mask\_int32x3xm1 (C function), 565  
vlssseg3wv\_mask\_int32x3xm2 (C function), 565  
vlssseg3wv\_mask\_int64x3xm1 (C function), 565  
vlssseg3wv\_mask\_int64x3xm2 (C function), 565  
vlssseg3wv\_mask\_int8x3xm1 (C function), 565  
vlssseg3wv\_mask\_int8x3xm2 (C function), 565  
vlssseg4buv\_mask\_uint16x4xm1 (C function), 568  
vlssseg4buv\_mask\_uint16x4xm2 (C function), 568  
vlssseg4buv\_mask\_uint32x4xm1 (C function), 568  
vlssseg4buv\_mask\_uint32x4xm2 (C function), 569  
vlssseg4buv\_mask\_uint64x4xm1 (C function), 569  
vlssseg4buv\_mask\_uint64x4xm2 (C function), 569  
vlssseg4buv\_mask\_uint8x4xm1 (C function), 569  
vlssseg4buv\_mask\_uint8x4xm2 (C function), 569  
vlssseg4buv\_uint16x4xm1 (C function), 568  
vlssseg4buv\_uint16x4xm2 (C function), 568  
vlssseg4buv\_uint32x4xm1 (C function), 568  
vlssseg4buv\_uint32x4xm2 (C function), 568  
vlssseg4buv\_uint64x4xm1 (C function), 568  
vlssseg4buv\_uint64x4xm2 (C function), 568  
vlssseg4buv\_uint8x4xm1 (C function), 568  
vlssseg4buv\_uint8x4xm2 (C function), 568  
vlssseg4bv\_int16x4xm1 (C function), 567  
vlssseg4bv\_int16x4xm2 (C function), 567  
vlssseg4bv\_int32x4xm1 (C function), 567  
vlssseg4bv\_int32x4xm2 (C function), 567  
vlssseg4bv\_int64x4xm1 (C function), 567  
vlssseg4bv\_int64x4xm2 (C function), 567  
vlssseg4bv\_int8x4xm1 (C function), 567  
vlssseg4bv\_int8x4xm2 (C function), 567  
vlssseg4bv\_mask\_int16x4xm1 (C function), 567  
vlssseg4bv\_mask\_int16x4xm2 (C function), 567  
vlssseg4bv\_mask\_int32x4xm1 (C function), 567  
vlssseg4bv\_mask\_int32x4xm2 (C function), 567  
vlssseg4bv\_mask\_int64x4xm1 (C function), 567  
vlssseg4bv\_mask\_int64x4xm2 (C function), 567  
vlssseg4bv\_mask\_int8x4xm1 (C function), 567  
vlssseg4bv\_mask\_int8x4xm2 (C function), 567  
vlssseg4ev\_float16x4xm1 (C function), 569  
vlssseg4ev\_float16x4xm2 (C function), 569  
vlssseg4ev\_float32x4xm1 (C function), 569  
vlssseg4ev\_float32x4xm2 (C function), 569  
vlssseg4ev\_float64x4xm1 (C function), 569  
vlssseg4ev\_float64x4xm2 (C function), 569  
vlssseg4ev\_int16x4xm1 (C function), 569  
vlssseg4ev\_int16x4xm2 (C function), 569  
vlssseg4ev\_int32x4xm1 (C function), 569  
vlssseg4ev\_int32x4xm2 (C function), 569  
vlssseg4ev\_int64x4xm1 (C function), 569  
vlssseg4ev\_int64x4xm2 (C function), 569  
vlssseg4ev\_int8x4xm1 (C function), 569  
vlssseg4ev\_int8x4xm2 (C function), 569  
vlssseg4ev\_mask\_float16x4xm1 (C function), 570  
vlssseg4ev\_mask\_float16x4xm2 (C function), 570  
vlssseg4ev\_mask\_float32x4xm1 (C function), 570  
vlssseg4ev\_mask\_float32x4xm2 (C function), 570  
vlssseg4ev\_mask\_float64x4xm1 (C function), 570  
vlssseg4ev\_mask\_float64x4xm2 (C function), 570  
vlssseg4ev\_mask\_int16x4xm1 (C function), 570  
vlssseg4ev\_mask\_int16x4xm2 (C function), 570

vlsseg4ev\_mask\_int32x4xm1 (C function), 570  
 vlsseg4ev\_mask\_int32x4xm2 (C function), 570  
 vlsseg4ev\_mask\_int64x4xm1 (C function), 570  
 vlsseg4ev\_mask\_int64x4xm2 (C function), 570  
 vlsseg4ev\_mask\_int8x4xm1 (C function), 570  
 vlsseg4ev\_mask\_int8x4xm2 (C function), 571  
 vlsseg4ev\_mask\_uint16x4xm1 (C function), 571  
 vlsseg4ev\_mask\_uint16x4xm2 (C function), 571  
 vlsseg4ev\_mask\_uint32x4xm1 (C function), 571  
 vlsseg4ev\_mask\_uint32x4xm2 (C function), 571  
 vlsseg4ev\_mask\_uint64x4xm1 (C function), 571  
 vlsseg4ev\_mask\_uint64x4xm2 (C function), 571  
 vlsseg4ev\_mask\_uint8x4xm1 (C function), 571  
 vlsseg4ev\_mask\_uint8x4xm2 (C function), 571  
 vlsseg4ev\_uint16x4xm1 (C function), 569  
 vlsseg4ev\_uint16x4xm2 (C function), 570  
 vlsseg4ev\_uint32x4xm1 (C function), 570  
 vlsseg4ev\_uint32x4xm2 (C function), 570  
 vlsseg4ev\_uint64x4xm1 (C function), 570  
 vlsseg4ev\_uint64x4xm2 (C function), 570  
 vlsseg4ev\_uint8x4xm1 (C function), 570  
 vlsseg4ev\_uint8x4xm2 (C function), 570  
 vlsseg4huv\_mask\_uint16x4xm1 (C function), 573  
 vlsseg4huv\_mask\_uint16x4xm2 (C function), 573  
 vlsseg4huv\_mask\_uint32x4xm1 (C function), 573  
 vlsseg4huv\_mask\_uint32x4xm2 (C function), 573  
 vlsseg4huv\_mask\_uint64x4xm1 (C function), 573  
 vlsseg4huv\_mask\_uint64x4xm2 (C function), 573  
 vlsseg4huv\_mask\_uint8x4xm1 (C function), 573  
 vlsseg4huv\_mask\_uint8x4xm2 (C function), 573  
 vlsseg4huv\_uint16x4xm1 (C function), 572  
 vlsseg4huv\_uint16x4xm2 (C function), 572  
 vlsseg4huv\_uint32x4xm1 (C function), 572  
 vlsseg4huv\_uint32x4xm2 (C function), 572  
 vlsseg4huv\_uint64x4xm1 (C function), 573  
 vlsseg4huv\_uint64x4xm2 (C function), 573  
 vlsseg4huv\_uint8x4xm1 (C function), 573  
 vlsseg4huv\_uint8x4xm2 (C function), 573  
 vlsseg4hv\_int16x4xm1 (C function), 571  
 vlsseg4hv\_int16x4xm2 (C function), 571  
 vlsseg4hv\_int32x4xm1 (C function), 571  
 vlsseg4hv\_int32x4xm2 (C function), 571  
 vlsseg4hv\_int64x4xm1 (C function), 571  
 vlsseg4hv\_int64x4xm2 (C function), 571  
 vlsseg4hv\_int8x4xm1 (C function), 571  
 vlsseg4hv\_int8x4xm2 (C function), 572  
 vlsseg4hv\_mask\_int16x4xm1 (C function), 572  
 vlsseg4hv\_mask\_int16x4xm2 (C function), 572  
 vlsseg4hv\_mask\_int32x4xm1 (C function), 572  
 vlsseg4hv\_mask\_int32x4xm2 (C function), 572  
 vlsseg4hv\_mask\_int64x4xm1 (C function), 572  
 vlsseg4hv\_mask\_int64x4xm2 (C function), 572  
 vlsseg4hv\_mask\_int8x4xm1 (C function), 572  
 vlsseg4hv\_mask\_int8x4xm2 (C function), 572  
 vlsseg4wuv\_mask\_uint16x4xm1 (C function), 575  
 vlsseg4wuv\_mask\_uint16x4xm2 (C function), 575  
 vlsseg4wuv\_mask\_uint32x4xm1 (C function), 575  
 vlsseg4wuv\_mask\_uint32x4xm2 (C function), 575  
 vlsseg4wuv\_mask\_uint64x4xm1 (C function), 575  
 vlsseg4wuv\_mask\_uint64x4xm2 (C function), 575  
 vlsseg4wuv\_mask\_uint8x4xm1 (C function), 576  
 vlsseg4wuv\_mask\_uint8x4xm2 (C function), 576  
 vlsseg4wuv\_uint16x4xm1 (C function), 575  
 vlsseg4wuv\_uint16x4xm2 (C function), 575  
 vlsseg4wuv\_uint32x4xm1 (C function), 575  
 vlsseg4wuv\_uint32x4xm2 (C function), 575  
 vlsseg4wuv\_uint64x4xm1 (C function), 575  
 vlsseg4wuv\_uint64x4xm2 (C function), 575  
 vlsseg4wuv\_uint8x4xm1 (C function), 575  
 vlsseg4wuv\_uint8x4xm2 (C function), 575  
 vlsseg4wv\_int16x4xm1 (C function), 574  
 vlsseg4wv\_int16x4xm2 (C function), 574  
 vlsseg4wv\_int32x4xm1 (C function), 574  
 vlsseg4wv\_int32x4xm2 (C function), 574  
 vlsseg4wv\_int64x4xm1 (C function), 574  
 vlsseg4wv\_int64x4xm2 (C function), 574  
 vlsseg4wv\_int8x4xm1 (C function), 574  
 vlsseg4wv\_int8x4xm2 (C function), 574  
 vlsseg4wv\_mask\_int16x4xm1 (C function), 574  
 vlsseg4wv\_mask\_int16x4xm2 (C function), 574  
 vlsseg4wv\_mask\_int32x4xm1 (C function), 574  
 vlsseg4wv\_mask\_int32x4xm2 (C function), 574  
 vlsseg4wv\_mask\_int64x4xm1 (C function), 574  
 vlsseg4wv\_mask\_int64x4xm2 (C function), 574  
 vlsseg4wv\_mask\_int8x4xm1 (C function), 574  
 vlsseg4wv\_mask\_int8x4xm2 (C function), 574  
 vlsseg5buv\_mask\_uint16x5xm1 (C function), 577  
 vlsseg5buv\_mask\_uint32x5xm1 (C function), 577  
 vlsseg5buv\_mask\_uint64x5xm1 (C function), 577  
 vlsseg5buv\_mask\_uint8x5xm1 (C function), 577  
 vlsseg5buv\_uint16x5xm1 (C function), 577  
 vlsseg5buv\_uint32x5xm1 (C function), 577  
 vlsseg5buv\_uint64x5xm1 (C function), 577  
 vlsseg5buv\_uint8x5xm1 (C function), 577  
 vlsseg5bv\_int16x5xm1 (C function), 576  
 vlsseg5bv\_int32x5xm1 (C function), 576  
 vlsseg5bv\_int64x5xm1 (C function), 576  
 vlsseg5bv\_int8x5xm1 (C function), 576  
 vlsseg5bv\_mask\_int16x5xm1 (C function), 576  
 vlsseg5bv\_mask\_int32x5xm1 (C function), 576  
 vlsseg5bv\_mask\_int64x5xm1 (C function), 576  
 vlsseg5bv\_mask\_int8x5xm1 (C function), 576  
 vlsseg5ev\_float16x5xm1 (C function), 577  
 vlsseg5ev\_float32x5xm1 (C function), 578  
 vlsseg5ev\_float64x5xm1 (C function), 578  
 vlsseg5ev\_int16x5xm1 (C function), 578  
 vlsseg5ev\_int32x5xm1 (C function), 578  
 vlsseg5ev\_int64x5xm1 (C function), 578

vlssseg5ev\_int8x5xm1 (C function), 578  
vlssseg5ev\_mask\_float16x5xm1 (C function), 578  
vlssseg5ev\_mask\_float32x5xm1 (C function), 578  
vlssseg5ev\_mask\_float64x5xm1 (C function), 578  
vlssseg5ev\_mask\_int16x5xm1 (C function), 578  
vlssseg5ev\_mask\_int32x5xm1 (C function), 578  
vlssseg5ev\_mask\_int64x5xm1 (C function), 578  
vlssseg5ev\_mask\_int8x5xm1 (C function), 578  
vlssseg5ev\_mask\_uint16x5xm1 (C function), 578  
vlssseg5ev\_mask\_uint32x5xm1 (C function), 578  
vlssseg5ev\_mask\_uint64x5xm1 (C function), 578  
vlssseg5ev\_mask\_uint8x5xm1 (C function), 579  
vlssseg5ev\_uint16x5xm1 (C function), 578  
vlssseg5ev\_uint32x5xm1 (C function), 578  
vlssseg5ev\_uint64x5xm1 (C function), 578  
vlssseg5ev\_uint8x5xm1 (C function), 578  
vlssseg5huv\_mask\_uint16x5xm1 (C function), 580  
vlssseg5huv\_mask\_uint32x5xm1 (C function), 580  
vlssseg5huv\_mask\_uint64x5xm1 (C function), 580  
vlssseg5huv\_mask\_uint8x5xm1 (C function), 580  
vlssseg5huv\_uint16x5xm1 (C function), 580  
vlssseg5huv\_uint32x5xm1 (C function), 580  
vlssseg5huv\_uint64x5xm1 (C function), 580  
vlssseg5huv\_uint8x5xm1 (C function), 580  
vlssseg5hv\_int16x5xm1 (C function), 579  
vlssseg5hv\_int32x5xm1 (C function), 579  
vlssseg5hv\_int64x5xm1 (C function), 579  
vlssseg5hv\_int8x5xm1 (C function), 579  
vlssseg5hv\_mask\_int16x5xm1 (C function), 579  
vlssseg5hv\_mask\_int32x5xm1 (C function), 579  
vlssseg5hv\_mask\_int64x5xm1 (C function), 579  
vlssseg5hv\_mask\_int8x5xm1 (C function), 579  
vlssseg5wuv\_mask\_uint16x5xm1 (C function), 582  
vlssseg5wuv\_mask\_uint32x5xm1 (C function), 582  
vlssseg5wuv\_mask\_uint64x5xm1 (C function), 582  
vlssseg5wuv\_mask\_uint8x5xm1 (C function), 582  
vlssseg5wuv\_uint16x5xm1 (C function), 581  
vlssseg5wuv\_uint32x5xm1 (C function), 581  
vlssseg5wuv\_uint64x5xm1 (C function), 581  
vlssseg5wuv\_uint8x5xm1 (C function), 581  
vlssseg5wv\_int16x5xm1 (C function), 580  
vlssseg5wv\_int32x5xm1 (C function), 581  
vlssseg5wv\_int64x5xm1 (C function), 581  
vlssseg5wv\_int8x5xm1 (C function), 581  
vlssseg5wv\_mask\_int16x5xm1 (C function), 581  
vlssseg5wv\_mask\_int32x5xm1 (C function), 581  
vlssseg5wv\_mask\_int64x5xm1 (C function), 581  
vlssseg5wv\_mask\_int8x5xm1 (C function), 581  
vlssseg6buv\_mask\_uint16x6xm1 (C function), 583  
vlssseg6buv\_mask\_uint32x6xm1 (C function), 583  
vlssseg6buv\_mask\_uint64x6xm1 (C function), 583  
vlssseg6buv\_mask\_uint8x6xm1 (C function), 583  
vlssseg6buv\_uint16x6xm1 (C function), 583  
vlssseg6buv\_uint32x6xm1 (C function), 583  
vlssseg6buv\_uint64x6xm1 (C function), 583  
vlssseg6bv\_int16x6xm1 (C function), 582  
vlssseg6bv\_int32x6xm1 (C function), 582  
vlssseg6bv\_int64x6xm1 (C function), 582  
vlssseg6bv\_int8x6xm1 (C function), 582  
vlssseg6bv\_mask\_int16x6xm1 (C function), 582  
vlssseg6bv\_mask\_int32x6xm1 (C function), 582  
vlssseg6bv\_mask\_int64x6xm1 (C function), 582  
vlssseg6bv\_mask\_int8x6xm1 (C function), 582  
vlssseg6ev\_float16x6xm1 (C function), 584  
vlssseg6ev\_float32x6xm1 (C function), 584  
vlssseg6ev\_float64x6xm1 (C function), 584  
vlssseg6ev\_int16x6xm1 (C function), 584  
vlssseg6ev\_int32x6xm1 (C function), 584  
vlssseg6ev\_int64x6xm1 (C function), 584  
vlssseg6ev\_int8x6xm1 (C function), 584  
vlssseg6ev\_mask\_float16x6xm1 (C function), 584  
vlssseg6ev\_mask\_float32x6xm1 (C function), 584  
vlssseg6ev\_mask\_float64x6xm1 (C function), 584  
vlssseg6ev\_mask\_int16x6xm1 (C function), 584  
vlssseg6ev\_mask\_int32x6xm1 (C function), 584  
vlssseg6ev\_mask\_int64x6xm1 (C function), 584  
vlssseg6ev\_mask\_int8x6xm1 (C function), 585  
vlssseg6ev\_mask\_uint16x6xm1 (C function), 585  
vlssseg6ev\_mask\_uint32x6xm1 (C function), 585  
vlssseg6ev\_mask\_uint64x6xm1 (C function), 585  
vlssseg6ev\_mask\_uint8x6xm1 (C function), 585  
vlssseg6ev\_uint16x6xm1 (C function), 584  
vlssseg6ev\_uint32x6xm1 (C function), 584  
vlssseg6ev\_uint64x6xm1 (C function), 584  
vlssseg6ev\_uint8x6xm1 (C function), 584  
vlssseg6huv\_mask\_uint16x6xm1 (C function), 586  
vlssseg6huv\_mask\_uint32x6xm1 (C function), 586  
vlssseg6huv\_mask\_uint64x6xm1 (C function), 586  
vlssseg6huv\_mask\_uint8x6xm1 (C function), 586  
vlssseg6huv\_uint16x6xm1 (C function), 586  
vlssseg6huv\_uint32x6xm1 (C function), 586  
vlssseg6huv\_uint64x6xm1 (C function), 586  
vlssseg6huv\_uint8x6xm1 (C function), 586  
vlssseg6hv\_int16x6xm1 (C function), 585  
vlssseg6hv\_int32x6xm1 (C function), 585  
vlssseg6hv\_int64x6xm1 (C function), 585  
vlssseg6hv\_int8x6xm1 (C function), 585  
vlssseg6hv\_mask\_int16x6xm1 (C function), 585  
vlssseg6hv\_mask\_int32x6xm1 (C function), 585  
vlssseg6hv\_mask\_int64x6xm1 (C function), 585  
vlssseg6hv\_mask\_int8x6xm1 (C function), 585  
vlssseg6wuv\_mask\_uint16x6xm1 (C function), 588  
vlssseg6wuv\_mask\_uint32x6xm1 (C function), 588  
vlssseg6wuv\_mask\_uint64x6xm1 (C function), 588  
vlssseg6wuv\_mask\_uint8x6xm1 (C function), 588  
vlssseg6wuv\_uint16x6xm1 (C function), 587  
vlssseg6wuv\_uint32x6xm1 (C function), 588

vlsseg6wuv\_uint64x6xm1 (C function), 588  
 vlsseg6wuv\_uint8x6xm1 (C function), 588  
 vlsseg6wv\_int16x6xm1 (C function), 587  
 vlsseg6wv\_int32x6xm1 (C function), 587  
 vlsseg6wv\_int64x6xm1 (C function), 587  
 vlsseg6wv\_int8x6xm1 (C function), 587  
 vlsseg6wv\_mask\_int16x6xm1 (C function), 587  
 vlsseg6wv\_mask\_int32x6xm1 (C function), 587  
 vlsseg6wv\_mask\_int64x6xm1 (C function), 587  
 vlsseg6wv\_mask\_int8x6xm1 (C function), 587  
 vlsseg7buv\_mask\_uint16x7xm1 (C function), 589  
 vlsseg7buv\_mask\_uint32x7xm1 (C function), 590  
 vlsseg7buv\_mask\_uint64x7xm1 (C function), 590  
 vlsseg7buv\_mask\_uint8x7xm1 (C function), 590  
 vlsseg7buv\_uint16x7xm1 (C function), 589  
 vlsseg7buv\_uint32x7xm1 (C function), 589  
 vlsseg7buv\_uint64x7xm1 (C function), 589  
 vlsseg7buv\_uint8x7xm1 (C function), 589  
 vlsseg7bv\_int16x7xm1 (C function), 588  
 vlsseg7bv\_int32x7xm1 (C function), 588  
 vlsseg7bv\_int64x7xm1 (C function), 588  
 vlsseg7bv\_int8x7xm1 (C function), 588  
 vlsseg7bv\_mask\_int16x7xm1 (C function), 589  
 vlsseg7bv\_mask\_int32x7xm1 (C function), 589  
 vlsseg7bv\_mask\_int64x7xm1 (C function), 589  
 vlsseg7bv\_mask\_int8x7xm1 (C function), 589  
 vlsseg7ev\_float16x7xm1 (C function), 590  
 vlsseg7ev\_float32x7xm1 (C function), 590  
 vlsseg7ev\_float64x7xm1 (C function), 590  
 vlsseg7ev\_int16x7xm1 (C function), 590  
 vlsseg7ev\_int32x7xm1 (C function), 590  
 vlsseg7ev\_int64x7xm1 (C function), 590  
 vlsseg7ev\_int8x7xm1 (C function), 590  
 vlsseg7ev\_mask\_float16x7xm1 (C function), 591  
 vlsseg7ev\_mask\_float32x7xm1 (C function), 591  
 vlsseg7ev\_mask\_float64x7xm1 (C function), 591  
 vlsseg7ev\_mask\_int16x7xm1 (C function), 591  
 vlsseg7ev\_mask\_int32x7xm1 (C function), 591  
 vlsseg7ev\_mask\_int64x7xm1 (C function), 591  
 vlsseg7ev\_mask\_int8x7xm1 (C function), 591  
 vlsseg7ev\_mask\_uint16x7xm1 (C function), 591  
 vlsseg7ev\_mask\_uint32x7xm1 (C function), 591  
 vlsseg7ev\_mask\_uint64x7xm1 (C function), 591  
 vlsseg7ev\_mask\_uint8x7xm1 (C function), 591  
 vlsseg7ev\_uint16x7xm1 (C function), 590  
 vlsseg7ev\_uint32x7xm1 (C function), 590  
 vlsseg7ev\_uint64x7xm1 (C function), 590  
 vlsseg7ev\_uint8x7xm1 (C function), 590  
 vlsseg7huv\_mask\_uint16x7xm1 (C function), 592  
 vlsseg7huv\_mask\_uint32x7xm1 (C function), 593  
 vlsseg7huv\_mask\_uint64x7xm1 (C function), 593  
 vlsseg7huv\_mask\_uint8x7xm1 (C function), 593  
 vlsseg7huv\_uint16x7xm1 (C function), 592  
 vlsseg7huv\_uint32x7xm1 (C function), 592  
 vlsseg7hv\_int16x7xm1 (C function), 591  
 vlsseg7hv\_int32x7xm1 (C function), 591  
 vlsseg7hv\_int64x7xm1 (C function), 591  
 vlsseg7hv\_int8x7xm1 (C function), 591  
 vlsseg7hv\_mask\_int16x7xm1 (C function), 592  
 vlsseg7hv\_mask\_int32x7xm1 (C function), 592  
 vlsseg7hv\_mask\_int64x7xm1 (C function), 592  
 vlsseg7hv\_mask\_int8x7xm1 (C function), 592  
 vlsseg7wuv\_mask\_uint16x7xm1 (C function), 594  
 vlsseg7wuv\_mask\_uint32x7xm1 (C function), 594  
 vlsseg7wuv\_mask\_uint64x7xm1 (C function), 594  
 vlsseg7wuv\_mask\_uint8x7xm1 (C function), 594  
 vlsseg7wuv\_uint16x7xm1 (C function), 594  
 vlsseg7wuv\_uint32x7xm1 (C function), 594  
 vlsseg7wuv\_uint64x7xm1 (C function), 594  
 vlsseg7wuv\_uint8x7xm1 (C function), 594  
 vlsseg7wv\_int16x7xm1 (C function), 593  
 vlsseg7wv\_int32x7xm1 (C function), 593  
 vlsseg7wv\_int64x7xm1 (C function), 593  
 vlsseg7wv\_int8x7xm1 (C function), 593  
 vlsseg7wv\_mask\_int16x7xm1 (C function), 593  
 vlsseg7wv\_mask\_int32x7xm1 (C function), 593  
 vlsseg7wv\_mask\_int64x7xm1 (C function), 593  
 vlsseg7wv\_mask\_int8x7xm1 (C function), 593  
 vlsseg8buv\_mask\_uint16x8xm1 (C function), 596  
 vlsseg8buv\_mask\_uint32x8xm1 (C function), 596  
 vlsseg8buv\_mask\_uint64x8xm1 (C function), 596  
 vlsseg8buv\_mask\_uint8x8xm1 (C function), 596  
 vlsseg8buv\_uint16x8xm1 (C function), 595  
 vlsseg8buv\_uint32x8xm1 (C function), 595  
 vlsseg8buv\_uint64x8xm1 (C function), 595  
 vlsseg8buv\_uint8x8xm1 (C function), 596  
 vlsseg8bv\_int16x8xm1 (C function), 595  
 vlsseg8bv\_int32x8xm1 (C function), 595  
 vlsseg8bv\_int64x8xm1 (C function), 595  
 vlsseg8bv\_int8x8xm1 (C function), 595  
 vlsseg8bv\_mask\_int16x8xm1 (C function), 595  
 vlsseg8bv\_mask\_int32x8xm1 (C function), 595  
 vlsseg8bv\_mask\_int64x8xm1 (C function), 595  
 vlsseg8bv\_mask\_int8x8xm1 (C function), 595  
 vlsseg8ev\_float16x8xm1 (C function), 596  
 vlsseg8ev\_float32x8xm1 (C function), 596  
 vlsseg8ev\_float64x8xm1 (C function), 596  
 vlsseg8ev\_int16x8xm1 (C function), 596  
 vlsseg8ev\_int32x8xm1 (C function), 596  
 vlsseg8ev\_int64x8xm1 (C function), 596  
 vlsseg8ev\_int8x8xm1 (C function), 596  
 vlsseg8ev\_mask\_float16x8xm1 (C function), 597  
 vlsseg8ev\_mask\_float32x8xm1 (C function), 597  
 vlsseg8ev\_mask\_float64x8xm1 (C function), 597  
 vlsseg8ev\_mask\_int16x8xm1 (C function), 597  
 vlsseg8ev\_mask\_int32x8xm1 (C function), 597



vlsseg8ev\_mask\_int64x8xm1 (C function), 597  
 vlsseg8ev\_mask\_int8x8xm1 (C function), 597  
 vlsseg8ev\_mask\_uint16x8xm1 (C function), 597  
 vlsseg8ev\_mask\_uint32x8xm1 (C function), 597  
 vlsseg8ev\_mask\_uint64x8xm1 (C function), 597  
 vlsseg8ev\_mask\_uint8x8xm1 (C function), 597  
 vlsseg8ev\_uint16x8xm1 (C function), 596  
 vlsseg8ev\_uint32x8xm1 (C function), 596  
 vlsseg8ev\_uint64x8xm1 (C function), 597  
 vlsseg8ev\_uint8x8xm1 (C function), 597  
 vlsseg8huv\_mask\_uint16x8xm1 (C function), 599  
 vlsseg8huv\_mask\_uint32x8xm1 (C function), 599  
 vlsseg8huv\_mask\_uint64x8xm1 (C function), 599  
 vlsseg8huv\_mask\_uint8x8xm1 (C function), 599  
 vlsseg8huv\_uint16x8xm1 (C function), 598  
 vlsseg8huv\_uint32x8xm1 (C function), 598  
 vlsseg8huv\_uint64x8xm1 (C function), 598  
 vlsseg8huv\_uint8x8xm1 (C function), 599  
 vlsseg8hv\_int16x8xm1 (C function), 598  
 vlsseg8hv\_int32x8xm1 (C function), 598  
 vlsseg8hv\_int64x8xm1 (C function), 598  
 vlsseg8hv\_int8x8xm1 (C function), 598  
 vlsseg8hv\_mask\_int16x8xm1 (C function), 598  
 vlsseg8hv\_mask\_int32x8xm1 (C function), 598  
 vlsseg8hv\_mask\_int64x8xm1 (C function), 598  
 vlsseg8hv\_mask\_int8x8xm1 (C function), 598  
 vlsseg8wuv\_mask\_uint16x8xm1 (C function), 600  
 vlsseg8wuv\_mask\_uint32x8xm1 (C function), 600  
 vlsseg8wuv\_mask\_uint64x8xm1 (C function), 600  
 vlsseg8wuv\_mask\_uint8x8xm1 (C function), 601  
 vlsseg8wuv\_uint16x8xm1 (C function), 600  
 vlsseg8wuv\_uint32x8xm1 (C function), 600  
 vlsseg8wuv\_uint64x8xm1 (C function), 600  
 vlsseg8wuv\_uint8x8xm1 (C function), 600  
 vlsseg8wv\_int16x8xm1 (C function), 599  
 vlsseg8wv\_int32x8xm1 (C function), 599  
 vlsseg8wv\_int64x8xm1 (C function), 599  
 vlsseg8wv\_int8x8xm1 (C function), 599  
 vlsseg8wv\_mask\_int16x8xm1 (C function), 600  
 vlsseg8wv\_mask\_int32x8xm1 (C function), 600  
 vlsseg8wv\_mask\_int64x8xm1 (C function), 600  
 vlsseg8wv\_mask\_int8x8xm1 (C function), 600  
 vlswuv\_mask\_uint16xm1 (C function), 433  
 vlswuv\_mask\_uint16xm2 (C function), 433  
 vlswuv\_mask\_uint16xm4 (C function), 433  
 vlswuv\_mask\_uint16xm8 (C function), 433  
 vlswuv\_uint16xm1 (C function), 432  
 vlswuv\_uint16xm2 (C function), 432  
 vlswuv\_uint16xm4 (C function), 432  
 vlswuv\_uint16xm8 (C function), 432  
 vlswuv\_uint32xm1 (C function), 432  
 vlswuv\_uint32xm2 (C function), 432  
 vlswuv\_uint32xm4 (C function), 432  
 vlswuv\_uint32xm8 (C function), 432  
 vlswuv\_uint64xm1 (C function), 432  
 vlswuv\_uint64xm2 (C function), 432  
 vlswuv\_uint64xm4 (C function), 432  
 vlswuv\_uint64xm8 (C function), 432  
 vlswuv\_uint8xm1 (C function), 433  
 vlswuv\_uint8xm2 (C function), 433  
 vlswuv\_uint8xm4 (C function), 433  
 vlswuv\_uint8xm8 (C function), 433  
 vlswv\_int16xm1 (C function), 431  
 vlswv\_int16xm2 (C function), 431  
 vlswv\_int16xm4 (C function), 431  
 vlswv\_int16xm8 (C function), 431  
 vlswv\_int32xm1 (C function), 431  
 vlswv\_int32xm2 (C function), 431  
 vlswv\_int32xm4 (C function), 431  
 vlswv\_int32xm8 (C function), 431  
 vlswv\_int64xm1 (C function), 431  
 vlswv\_int64xm2 (C function), 431  
 vlswv\_int64xm4 (C function), 431  
 vlswv\_int64xm8 (C function), 431  
 vlswv\_int8xm1 (C function), 431  
 vlswv\_int8xm2 (C function), 431  
 vlswv\_int8xm4 (C function), 431  
 vlswv\_int8xm8 (C function), 431  
 vlswv\_mask\_int16xm1 (C function), 431  
 vlswv\_mask\_int16xm2 (C function), 431  
 vlswv\_mask\_int16xm4 (C function), 431  
 vlswv\_mask\_int16xm8 (C function), 431  
 vlswv\_mask\_int32xm1 (C function), 431  
 vlswv\_mask\_int32xm2 (C function), 431  
 vlswv\_mask\_int32xm4 (C function), 431  
 vlswv\_mask\_int32xm8 (C function), 432  
 vlswv\_mask\_int64xm1 (C function), 432  
 vlswv\_mask\_int64xm2 (C function), 432  
 vlswv\_mask\_int64xm4 (C function), 432  
 vlswv\_mask\_int64xm8 (C function), 432  
 vlswv\_mask\_int8xm1 (C function), 432  
 vlswv\_mask\_int8xm2 (C function), 432  
 vlswv\_mask\_int8xm4 (C function), 432  
 vlswv\_mask\_int8xm8 (C function), 432  
 vlwuv\_mask\_uint16xm1 (C function), 436  
 vlwuv\_mask\_uint16xm2 (C function), 436

vlwuv\_mask\_uint16xm4 (C function), 436  
 vlwuv\_mask\_uint16xm8 (C function), 436  
 vlwuv\_mask\_uint32xm1 (C function), 436  
 vlwuv\_mask\_uint32xm2 (C function), 436  
 vlwuv\_mask\_uint32xm4 (C function), 436  
 vlwuv\_mask\_uint32xm8 (C function), 436  
 vlwuv\_mask\_uint64xm1 (C function), 436  
 vlwuv\_mask\_uint64xm2 (C function), 436  
 vlwuv\_mask\_uint64xm4 (C function), 436  
 vlwuv\_mask\_uint64xm8 (C function), 436  
 vlwuv\_mask\_uint8xm1 (C function), 436  
 vlwuv\_mask\_uint8xm2 (C function), 437  
 vlwuv\_mask\_uint8xm4 (C function), 437  
 vlwuv\_mask\_uint8xm8 (C function), 437  
 vlwuv\_uint16xm1 (C function), 435  
 vlwuv\_uint16xm2 (C function), 435  
 vlwuv\_uint16xm4 (C function), 435  
 vlwuv\_uint16xm8 (C function), 435  
 vlwuv\_uint32xm1 (C function), 435  
 vlwuv\_uint32xm2 (C function), 435  
 vlwuv\_uint32xm4 (C function), 436  
 vlwuv\_uint32xm8 (C function), 436  
 vlwuv\_uint64xm1 (C function), 436  
 vlwuv\_uint64xm2 (C function), 436  
 vlwuv\_uint64xm4 (C function), 436  
 vlwuv\_uint64xm8 (C function), 436  
 vlwuv\_uint8xm1 (C function), 436  
 vlwuv\_uint8xm2 (C function), 436  
 vlwuv\_uint8xm4 (C function), 436  
 vlwuv\_uint8xm8 (C function), 436  
 vlwv\_int16xm1 (C function), 434  
 vlwv\_int16xm2 (C function), 434  
 vlwv\_int16xm4 (C function), 434  
 vlwv\_int16xm8 (C function), 434  
 vlwv\_int32xm1 (C function), 434  
 vlwv\_int32xm2 (C function), 434  
 vlwv\_int32xm4 (C function), 434  
 vlwv\_int32xm8 (C function), 434  
 vlwv\_int64xm1 (C function), 434  
 vlwv\_int64xm2 (C function), 434  
 vlwv\_int64xm4 (C function), 434  
 vlwv\_int64xm8 (C function), 434  
 vlwv\_int8xm1 (C function), 434  
 vlwv\_int8xm2 (C function), 434  
 vlwv\_int8xm4 (C function), 434  
 vlwv\_int8xm8 (C function), 434  
 vlwv\_mask\_int16xm1 (C function), 434  
 vlwv\_mask\_int16xm2 (C function), 434  
 vlwv\_mask\_int16xm4 (C function), 434  
 vlwv\_mask\_int16xm8 (C function), 435  
 vlwv\_mask\_int32xm1 (C function), 435  
 vlwv\_mask\_int32xm2 (C function), 435  
 vlwv\_mask\_int32xm4 (C function), 435  
 vlwv\_mask\_int32xm8 (C function), 435

vlwv\_mask\_int64xm1 (C function), 435  
 vlwv\_mask\_int64xm2 (C function), 435  
 vlwv\_mask\_int64xm4 (C function), 435  
 vlwv\_mask\_int64xm8 (C function), 435  
 vlwv\_mask\_int8xm1 (C function), 435  
 vlwv\_mask\_int8xm2 (C function), 435  
 vlwv\_mask\_int8xm4 (C function), 435  
 vlwv\_mask\_int8xm8 (C function), 435  
 vlxbuv\_mask\_uint16xm1 (C function), 439  
 vlxbuv\_mask\_uint16xm2 (C function), 439  
 vlxbuv\_mask\_uint16xm4 (C function), 439  
 vlxbuv\_mask\_uint16xm8 (C function), 439  
 vlxbuv\_mask\_uint32xm1 (C function), 439  
 vlxbuv\_mask\_uint32xm2 (C function), 439  
 vlxbuv\_mask\_uint32xm4 (C function), 439  
 vlxbuv\_mask\_uint32xm8 (C function), 439  
 vlxbuv\_mask\_uint64xm1 (C function), 440  
 vlxbuv\_mask\_uint64xm2 (C function), 440  
 vlxbuv\_mask\_uint64xm4 (C function), 440  
 vlxbuv\_mask\_uint64xm8 (C function), 440  
 vlxbuv\_mask\_uint8xm1 (C function), 440  
 vlxbuv\_mask\_uint8xm2 (C function), 440  
 vlxbuv\_mask\_uint8xm4 (C function), 440  
 vlxbuv\_mask\_uint8xm8 (C function), 440  
 vlxbuv\_uint16xm1 (C function), 439  
 vlxbuv\_uint16xm2 (C function), 439  
 vlxbuv\_uint16xm4 (C function), 439  
 vlxbuv\_uint16xm8 (C function), 439  
 vlxbuv\_uint32xm1 (C function), 439  
 vlxbuv\_uint32xm2 (C function), 439  
 vlxbuv\_uint32xm4 (C function), 439  
 vlxbuv\_uint32xm8 (C function), 439  
 vlxbuv\_uint64xm1 (C function), 439  
 vlxbuv\_uint64xm2 (C function), 439  
 vlxbuv\_uint64xm4 (C function), 439  
 vlxbuv\_uint64xm8 (C function), 439  
 vlxbuv\_uint8xm1 (C function), 439  
 vlxbuv\_uint8xm2 (C function), 439  
 vlxbuv\_uint8xm4 (C function), 439  
 vlxbuv\_uint8xm8 (C function), 439  
 vlxbv\_int16xm1 (C function), 437  
 vlxbv\_int16xm2 (C function), 437  
 vlxbv\_int16xm4 (C function), 437  
 vlxbv\_int16xm8 (C function), 437  
 vlxbv\_int32xm1 (C function), 437  
 vlxbv\_int32xm2 (C function), 437  
 vlxbv\_int32xm4 (C function), 437  
 vlxbv\_int32xm8 (C function), 437  
 vlxbv\_int64xm1 (C function), 437  
 vlxbv\_int64xm2 (C function), 437  
 vlxbv\_int64xm4 (C function), 437  
 vlxbv\_int64xm8 (C function), 437  
 vlxbv\_int8xm1 (C function), 437  
 vlxbv\_int8xm2 (C function), 437



vlxbv\_int8xm4 (C function), 437  
vlxbv\_int8xm8 (C function), 437  
vlxbv\_mask\_int16xm1 (C function), 438  
vlxbv\_mask\_int16xm2 (C function), 438  
vlxbv\_mask\_int16xm4 (C function), 438  
vlxbv\_mask\_int16xm8 (C function), 438  
vlxbv\_mask\_int32xm1 (C function), 438  
vlxbv\_mask\_int32xm2 (C function), 438  
vlxbv\_mask\_int32xm4 (C function), 438  
vlxbv\_mask\_int32xm8 (C function), 438  
vlxbv\_mask\_int64xm1 (C function), 438  
vlxbv\_mask\_int64xm2 (C function), 438  
vlxbv\_mask\_int64xm4 (C function), 438  
vlxbv\_mask\_int64xm8 (C function), 438  
vlxbv\_mask\_int8xm1 (C function), 438  
vlxbv\_mask\_int8xm2 (C function), 438  
vlxbv\_mask\_int8xm4 (C function), 438  
vlxbv\_mask\_int8xm8 (C function), 438  
vlxev\_float16xm1 (C function), 440  
vlxev\_float16xm2 (C function), 440  
vlxev\_float16xm4 (C function), 440  
vlxev\_float16xm8 (C function), 440  
vlxev\_float32xm1 (C function), 440  
vlxev\_float32xm2 (C function), 440  
vlxev\_float32xm4 (C function), 440  
vlxev\_float32xm8 (C function), 440  
vlxev\_float64xm1 (C function), 440  
vlxev\_float64xm2 (C function), 440  
vlxev\_float64xm4 (C function), 440  
vlxev\_float64xm8 (C function), 440  
vlxev\_int16xm1 (C function), 440  
vlxev\_int16xm2 (C function), 441  
vlxev\_int16xm4 (C function), 441  
vlxev\_int16xm8 (C function), 441  
vlxev\_int32xm1 (C function), 441  
vlxev\_int32xm2 (C function), 441  
vlxev\_int32xm4 (C function), 441  
vlxev\_int32xm8 (C function), 441  
vlxev\_int64xm1 (C function), 441  
vlxev\_int64xm2 (C function), 441  
vlxev\_int64xm4 (C function), 441  
vlxev\_int64xm8 (C function), 441  
vlxev\_int8xm1 (C function), 441  
vlxev\_int8xm2 (C function), 441  
vlxev\_int8xm4 (C function), 441  
vlxev\_int8xm8 (C function), 441  
vlxev\_mask\_float16xm1 (C function), 442  
vlxev\_mask\_float16xm2 (C function), 442  
vlxev\_mask\_float16xm4 (C function), 442  
vlxev\_mask\_float16xm8 (C function), 442  
vlxev\_mask\_float32xm1 (C function), 442  
vlxev\_mask\_float32xm2 (C function), 442  
vlxev\_mask\_float32xm4 (C function), 442  
vlxev\_mask\_float32xm8 (C function), 442

vlxev\_mask\_float64xm1 (C function), 442  
vlxev\_mask\_float64xm2 (C function), 442  
vlxev\_mask\_float64xm4 (C function), 442  
vlxev\_mask\_float64xm8 (C function), 442  
vlxev\_mask\_int16xm1 (C function), 442  
vlxev\_mask\_int16xm2 (C function), 442  
vlxev\_mask\_int16xm4 (C function), 442  
vlxev\_mask\_int16xm8 (C function), 442  
vlxev\_mask\_int32xm1 (C function), 442  
vlxev\_mask\_int32xm2 (C function), 442  
vlxev\_mask\_int32xm4 (C function), 442  
vlxev\_mask\_int32xm8 (C function), 442  
vlxev\_mask\_int64xm1 (C function), 442  
vlxev\_mask\_int64xm2 (C function), 442  
vlxev\_mask\_int64xm4 (C function), 442  
vlxev\_mask\_int64xm8 (C function), 443  
vlxev\_mask\_int8xm1 (C function), 443  
vlxev\_mask\_int8xm2 (C function), 443  
vlxev\_mask\_int8xm4 (C function), 443  
vlxev\_mask\_int8xm8 (C function), 443  
vlxev\_mask\_uint16xm1 (C function), 443  
vlxev\_mask\_uint16xm2 (C function), 443  
vlxev\_mask\_uint16xm4 (C function), 443  
vlxev\_mask\_uint16xm8 (C function), 443  
vlxev\_mask\_uint32xm1 (C function), 443  
vlxev\_mask\_uint32xm2 (C function), 443  
vlxev\_mask\_uint32xm4 (C function), 443  
vlxev\_mask\_uint32xm8 (C function), 443  
vlxev\_mask\_uint64xm1 (C function), 443  
vlxev\_mask\_uint64xm2 (C function), 443  
vlxev\_mask\_uint64xm4 (C function), 443  
vlxev\_mask\_uint64xm8 (C function), 443  
vlxev\_mask\_uint8xm1 (C function), 443  
vlxev\_mask\_uint8xm2 (C function), 443  
vlxev\_mask\_uint8xm4 (C function), 443  
vlxev\_mask\_uint8xm8 (C function), 443  
vlxev\_uint16xm1 (C function), 441  
vlxev\_uint16xm2 (C function), 441  
vlxev\_uint16xm4 (C function), 441  
vlxev\_uint16xm8 (C function), 441  
vlxev\_uint32xm1 (C function), 441  
vlxev\_uint32xm2 (C function), 441  
vlxev\_uint32xm4 (C function), 441  
vlxev\_uint32xm8 (C function), 441  
vlxev\_uint64xm1 (C function), 441  
vlxev\_uint64xm2 (C function), 441  
vlxev\_uint64xm4 (C function), 441  
vlxev\_uint64xm8 (C function), 441  
vlxev\_uint8xm1 (C function), 441  
vlxev\_uint8xm2 (C function), 441  
vlxev\_uint8xm4 (C function), 441  
vlxev\_uint8xm8 (C function), 441  
vlxhuv\_mask\_uint16xm1 (C function), 446  
vlxhuv\_mask\_uint16xm2 (C function), 446

- vlxhuv\_mask\_uint16xm4 (C function), 446  
 vlxhuv\_mask\_uint16xm8 (C function), 446  
 vlxhuv\_mask\_uint32xm1 (C function), 446  
 vlxhuv\_mask\_uint32xm2 (C function), 446  
 vlxhuv\_mask\_uint32xm4 (C function), 446  
 vlxhuv\_mask\_uint32xm8 (C function), 446  
 vlxhuv\_mask\_uint64xm1 (C function), 446  
 vlxhuv\_mask\_uint64xm2 (C function), 446  
 vlxhuv\_mask\_uint64xm4 (C function), 446  
 vlxhuv\_mask\_uint64xm8 (C function), 446  
 vlxhuv\_mask\_uint8xm1 (C function), 447  
 vlxhuv\_mask\_uint8xm2 (C function), 447  
 vlxhuv\_mask\_uint8xm4 (C function), 447  
 vlxhuv\_mask\_uint8xm8 (C function), 447  
 vlxhuv\_uint16xm1 (C function), 445  
 vlxhuv\_uint16xm2 (C function), 445  
 vlxhuv\_uint16xm4 (C function), 445  
 vlxhuv\_uint16xm8 (C function), 445  
 vlxhuv\_uint32xm1 (C function), 446  
 vlxhuv\_uint32xm2 (C function), 446  
 vlxhuv\_uint32xm4 (C function), 446  
 vlxhuv\_uint32xm8 (C function), 446  
 vlxhuv\_uint64xm1 (C function), 446  
 vlxhuv\_uint64xm2 (C function), 446  
 vlxhuv\_uint64xm4 (C function), 446  
 vlxhuv\_uint64xm8 (C function), 446  
 vlxhuv\_uint8xm1 (C function), 446  
 vlxhuv\_uint8xm2 (C function), 446  
 vlxhuv\_uint8xm4 (C function), 446  
 vlxhuv\_uint8xm8 (C function), 446  
 vlxhv\_int16xm1 (C function), 444  
 vlxhv\_int16xm2 (C function), 444  
 vlxhv\_int16xm4 (C function), 444  
 vlxhv\_int16xm8 (C function), 444  
 vlxhv\_int32xm1 (C function), 444  
 vlxhv\_int32xm2 (C function), 444  
 vlxhv\_int32xm4 (C function), 444  
 vlxhv\_int32xm8 (C function), 444  
 vlxhv\_int64xm1 (C function), 444  
 vlxhv\_int64xm2 (C function), 444  
 vlxhv\_int64xm4 (C function), 444  
 vlxhv\_int64xm8 (C function), 444  
 vlxhv\_int8xm1 (C function), 444  
 vlxhv\_int8xm2 (C function), 444  
 vlxhv\_int8xm4 (C function), 444  
 vlxhv\_int8xm8 (C function), 444  
 vlxhv\_mask\_int16xm1 (C function), 444  
 vlxhv\_mask\_int16xm2 (C function), 444  
 vlxhv\_mask\_int16xm4 (C function), 444  
 vlxhv\_mask\_int16xm8 (C function), 444  
 vlxhv\_mask\_int32xm1 (C function), 445  
 vlxhv\_mask\_int32xm2 (C function), 445  
 vlxhv\_mask\_int32xm4 (C function), 445  
 vlxhv\_mask\_int32xm8 (C function), 445  
 vlxhv\_mask\_int64xm1 (C function), 445  
 vlxhv\_mask\_int64xm2 (C function), 445  
 vlxhv\_mask\_int64xm4 (C function), 445  
 vlxhv\_mask\_int64xm8 (C function), 445  
 vlxhv\_mask\_int8xm1 (C function), 445  
 vlxhv\_mask\_int8xm2 (C function), 445  
 vlxhv\_mask\_int8xm4 (C function), 445  
 vlxhv\_mask\_int8xm8 (C function), 445  
 vlxseg2buv\_mask\_uint16x2xm1\_uint16xm1 (C function), 603  
 vlxseg2buv\_mask\_uint16x2xm2\_uint16xm2 (C function), 603  
 vlxseg2buv\_mask\_uint16x2xm4\_uint16xm4 (C function), 604  
 vlxseg2buv\_mask\_uint32x2xm1\_uint32xm1 (C function), 604  
 vlxseg2buv\_mask\_uint32x2xm2\_uint32xm2 (C function), 604  
 vlxseg2buv\_mask\_uint32x2xm4\_uint32xm4 (C function), 604  
 vlxseg2buv\_mask\_uint64x2xm1\_uint64xm1 (C function), 604  
 vlxseg2buv\_mask\_uint64x2xm2\_uint64xm2 (C function), 604  
 vlxseg2buv\_mask\_uint64x2xm4\_uint64xm4 (C function), 604  
 vlxseg2buv\_mask\_uint8x2xm1\_uint8xm1 (C function), 604  
 vlxseg2buv\_mask\_uint8x2xm2\_uint8xm2 (C function), 604  
 vlxseg2buv\_mask\_uint8x2xm4\_uint8xm4 (C function), 604  
 vlxseg2buv\_uint16x2xm1\_uint16xm1 (C function), 603  
 vlxseg2buv\_uint16x2xm2\_uint16xm2 (C function), 603  
 vlxseg2buv\_uint16x2xm4\_uint16xm4 (C function), 603  
 vlxseg2buv\_uint32x2xm1\_uint32xm1 (C function), 603  
 vlxseg2buv\_uint32x2xm2\_uint32xm2 (C function), 603  
 vlxseg2buv\_uint32x2xm4\_uint32xm4 (C function), 603  
 vlxseg2buv\_uint64x2xm1\_uint64xm1 (C function), 603  
 vlxseg2buv\_uint64x2xm2\_uint64xm2 (C function), 603  
 vlxseg2buv\_uint64x2xm4\_uint64xm4 (C function), 603  
 vlxseg2buv\_uint8x2xm1\_uint8xm1 (C function), 603  
 vlxseg2buv\_uint8x2xm2\_uint8xm2 (C function), 603  
 vlxseg2buv\_uint8x2xm4\_uint8xm4 (C function), 603  
 vlxseg2bv\_int16x2xm1\_int16xm1 (C function), 601  
 vlxseg2bv\_int16x2xm2\_int16xm2 (C function), 601  
 vlxseg2bv\_int16x2xm4\_int16xm4 (C function), 601  
 vlxseg2bv\_int32x2xm1\_int32xm1 (C function), 601  
 vlxseg2bv\_int32x2xm2\_int32xm2 (C function), 601  
 vlxseg2bv\_int32x2xm4\_int32xm4 (C function), 601  
 vlxseg2bv\_int64x2xm1\_int64xm1 (C function), 601  
 vlxseg2bv\_int64x2xm2\_int64xm2 (C function), 601  
 vlxseg2bv\_int64x2xm4\_int64xm4 (C function), 601  
 vlxseg2bv\_int8x2xm1\_int8xm1 (C function), 601

vlxsseg2bv\_int8x2xm2\_int8xm2 (C function), 601  
 vlxsseg2bv\_int8x2xm4\_int8xm4 (C function), 601  
 vlxsseg2bv\_mask\_int16x2xm1\_int16xm1 (C function), 602  
 vlxsseg2bv\_mask\_int16x2xm2\_int16xm2 (C function), 602  
 vlxsseg2bv\_mask\_int16x2xm4\_int16xm4 (C function), 602  
 vlxsseg2bv\_mask\_int32x2xm1\_int32xm1 (C function), 602  
 vlxsseg2bv\_mask\_int32x2xm2\_int32xm2 (C function), 602  
 vlxsseg2bv\_mask\_int32x2xm4\_int32xm4 (C function), 602  
 vlxsseg2bv\_mask\_int64x2xm1\_int64xm1 (C function), 602  
 vlxsseg2bv\_mask\_int64x2xm2\_int64xm2 (C function), 602  
 vlxsseg2bv\_mask\_int64x2xm4\_int64xm4 (C function), 602  
 vlxsseg2bv\_mask\_int8x2xm1\_int8xm1 (C function), 602  
 vlxsseg2bv\_mask\_int8x2xm2\_int8xm2 (C function), 602  
 vlxsseg2bv\_mask\_int8x2xm4\_int8xm4 (C function), 602  
 vlxsseg2ev\_float16x2xm1\_float16xm1 (C function), 605  
 vlxsseg2ev\_float16x2xm2\_float16xm2 (C function), 605  
 vlxsseg2ev\_float16x2xm4\_float16xm4 (C function), 605  
 vlxsseg2ev\_float32x2xm1\_float32xm1 (C function), 605  
 vlxsseg2ev\_float32x2xm2\_float32xm2 (C function), 605  
 vlxsseg2ev\_float32x2xm4\_float32xm4 (C function), 605  
 vlxsseg2ev\_float64x2xm1\_float64xm1 (C function), 605  
 vlxsseg2ev\_float64x2xm2\_float64xm2 (C function), 605  
 vlxsseg2ev\_float64x2xm4\_float64xm4 (C function), 605  
 vlxsseg2ev\_int16x2xm1\_int16xm1 (C function), 605  
 vlxsseg2ev\_int16x2xm2\_int16xm2 (C function), 605  
 vlxsseg2ev\_int16x2xm4\_int16xm4 (C function), 605  
 vlxsseg2ev\_int32x2xm1\_int32xm1 (C function), 605  
 vlxsseg2ev\_int32x2xm2\_int32xm2 (C function), 605  
 vlxsseg2ev\_int32x2xm4\_int32xm4 (C function), 605  
 vlxsseg2ev\_int64x2xm1\_int64xm1 (C function), 605  
 vlxsseg2ev\_int64x2xm2\_int64xm2 (C function), 605  
 vlxsseg2ev\_int64x2xm4\_int64xm4 (C function), 605  
 vlxsseg2ev\_int8x2xm1\_int8xm1 (C function), 606  
 vlxsseg2ev\_int8x2xm2\_int8xm2 (C function), 606  
 vlxsseg2ev\_int8x2xm4\_int8xm4 (C function), 606  
 vlxsseg2ev\_mask\_float16x2xm1\_float16xm1 (C function), 606  
 vlxsseg2ev\_mask\_float16x2xm2\_float16xm2 (C function), 606  
 vlxsseg2ev\_mask\_float16x2xm4\_float16xm4 (C function), 607  
 vlxsseg2ev\_mask\_float32x2xm1\_float32xm1 (C function), 607  
 vlxsseg2ev\_mask\_float32x2xm2\_float32xm2 (C function), 607  
 vlxsseg2ev\_mask\_float32x2xm4\_float32xm4 (C function), 607  
 vlxsseg2ev\_mask\_float64x2xm1\_float64xm1 (C function), 607  
 vlxsseg2ev\_mask\_float64x2xm2\_float64xm2 (C function), 607  
 vlxsseg2ev\_mask\_float64x2xm4\_float64xm4 (C function), 607  
 vlxsseg2ev\_mask\_int16x2xm1\_int16xm1 (C function), 607  
 vlxsseg2ev\_mask\_int16x2xm2\_int16xm2 (C function), 607  
 vlxsseg2ev\_mask\_int16x2xm4\_int16xm4 (C function), 607  
 vlxsseg2ev\_mask\_int32x2xm1\_int32xm1 (C function), 607  
 vlxsseg2ev\_mask\_int32x2xm2\_int32xm2 (C function), 607  
 vlxsseg2ev\_mask\_int32x2xm4\_int32xm4 (C function), 608  
 vlxsseg2ev\_mask\_int64x2xm1\_int64xm1 (C function), 608  
 vlxsseg2ev\_mask\_int64x2xm2\_int64xm2 (C function), 608  
 vlxsseg2ev\_mask\_int64x2xm4\_int64xm4 (C function), 608  
 vlxsseg2ev\_mask\_int8x2xm1\_int8xm1 (C function), 608  
 vlxsseg2ev\_mask\_int8x2xm2\_int8xm2 (C function), 608  
 vlxsseg2ev\_mask\_int8x2xm4\_int8xm4 (C function), 608  
 vlxsseg2ev\_mask\_uint16x2xm1\_uint16xm1 (C function), 608  
 vlxsseg2ev\_mask\_uint16x2xm2\_uint16xm2 (C function), 608  
 vlxsseg2ev\_mask\_uint16x2xm4\_uint16xm4 (C function), 608  
 vlxsseg2ev\_mask\_uint32x2xm1\_uint32xm1 (C function), 608  
 vlxsseg2ev\_mask\_uint32x2xm2\_uint32xm2 (C function), 608  
 vlxsseg2ev\_mask\_uint32x2xm4\_uint32xm4 (C function), 608  
 vlxsseg2ev\_mask\_uint64x2xm1\_uint64xm1 (C function), 608  
 vlxsseg2ev\_mask\_uint64x2xm2\_uint64xm2 (C function), 609  
 vlxsseg2ev\_mask\_uint64x2xm4\_uint64xm4 (C function), 609  
 vlxsseg2ev\_mask\_uint8x2xm1\_uint8xm1 (C function), 609  
 vlxsseg2ev\_mask\_uint8x2xm2\_uint8xm2 (C function), 609  
 vlxsseg2ev\_mask\_uint8x2xm4\_uint8xm4 (C function), 609  
 vlxsseg2ev\_uint16x2xm1\_uint16xm1 (C function), 606

vlxsseg2ev\_uint16x2xm2\_uint16xm2 (C function), 606  
 vlxsseg2ev\_uint16x2xm4\_uint16xm4 (C function), 606  
 vlxsseg2ev\_uint32x2xm1\_uint32xm1 (C function), 606  
 vlxsseg2ev\_uint32x2xm2\_uint32xm2 (C function), 606  
 vlxsseg2ev\_uint32x2xm4\_uint32xm4 (C function), 606  
 vlxsseg2ev\_uint64x2xm1\_uint64xm1 (C function), 606  
 vlxsseg2ev\_uint64x2xm2\_uint64xm2 (C function), 606  
 vlxsseg2ev\_uint64x2xm4\_uint64xm4 (C function), 606  
 vlxsseg2ev\_uint8x2xm1\_uint8xm1 (C function), 606  
 vlxsseg2ev\_uint8x2xm2\_uint8xm2 (C function), 606  
 vlxsseg2ev\_uint8x2xm4\_uint8xm4 (C function), 606  
 vlxsseg2huv\_mask\_uint16x2xm1\_uint16xm1 (C function), 612  
 vlxsseg2huv\_mask\_uint16x2xm2\_uint16xm2 (C function), 612  
 vlxsseg2huv\_mask\_uint16x2xm4\_uint16xm4 (C function), 612  
 vlxsseg2huv\_mask\_uint32x2xm1\_uint32xm1 (C function), 612  
 vlxsseg2huv\_mask\_uint32x2xm2\_uint32xm2 (C function), 612  
 vlxsseg2huv\_mask\_uint32x2xm4\_uint32xm4 (C function), 612  
 vlxsseg2huv\_mask\_uint64x2xm1\_uint64xm1 (C function), 612  
 vlxsseg2huv\_mask\_uint64x2xm2\_uint64xm2 (C function), 612  
 vlxsseg2huv\_mask\_uint64x2xm4\_uint64xm4 (C function), 612  
 vlxsseg2huv\_mask\_uint8x2xm1\_uint8xm1 (C function), 612  
 vlxsseg2huv\_mask\_uint8x2xm2\_uint8xm2 (C function), 612  
 vlxsseg2huv\_mask\_uint8x2xm4\_uint8xm4 (C function), 613  
 vlxsseg2huv\_uint16x2xm1\_uint16xm1 (C function), 611  
 vlxsseg2huv\_uint16x2xm2\_uint16xm2 (C function), 611  
 vlxsseg2huv\_uint16x2xm4\_uint16xm4 (C function), 611  
 vlxsseg2huv\_uint32x2xm1\_uint32xm1 (C function), 611  
 vlxsseg2huv\_uint32x2xm2\_uint32xm2 (C function), 611  
 vlxsseg2huv\_uint32x2xm4\_uint32xm4 (C function), 611  
 vlxsseg2huv\_uint64x2xm1\_uint64xm1 (C function), 611  
 vlxsseg2huv\_uint64x2xm2\_uint64xm2 (C function), 611  
 vlxsseg2huv\_uint64x2xm4\_uint64xm4 (C function), 611  
 vlxsseg2huv\_uint8x2xm1\_uint8xm1 (C function), 611  
 vlxsseg2huv\_uint8x2xm2\_uint8xm2 (C function), 611  
 vlxsseg2huv\_uint8x2xm4\_uint8xm4 (C function), 611  
 vlxsseg2hv\_int16x2xm1\_int16xm1 (C function), 609  
 vlxsseg2hv\_int16x2xm2\_int16xm2 (C function), 609  
 vlxsseg2hv\_int16x2xm4\_int16xm4 (C function), 609  
 vlxsseg2hv\_int32x2xm1\_int32xm1 (C function), 609  
 vlxsseg2hv\_int32x2xm2\_int32xm2 (C function), 609  
 vlxsseg2hv\_int32x2xm4\_int32xm4 (C function), 609  
 vlxsseg2hv\_int64x2xm1\_int64xm1 (C function), 609  
 vlxsseg2hv\_int64x2xm2\_int64xm2 (C function), 609  
 vlxsseg2hv\_int64x2xm4\_int64xm4 (C function), 610  
 vlxsseg2hv\_int8x2xm1\_int8xm1 (C function), 610  
 vlxsseg2hv\_int8x2xm2\_int8xm2 (C function), 610  
 vlxsseg2hv\_int8x2xm4\_int8xm4 (C function), 610  
 vlxsseg2hv\_mask\_int16x2xm1\_int16xm1 (C function), 610  
 vlxsseg2hv\_mask\_int16x2xm2\_int16xm2 (C function), 610  
 vlxsseg2hv\_mask\_int16x2xm4\_int16xm4 (C function), 610  
 vlxsseg2hv\_mask\_int32x2xm1\_int32xm1 (C function), 610  
 vlxsseg2hv\_mask\_int32x2xm2\_int32xm2 (C function), 610  
 vlxsseg2hv\_mask\_int32x2xm4\_int32xm4 (C function), 610  
 vlxsseg2hv\_mask\_int64x2xm1\_int64xm1 (C function), 610  
 vlxsseg2hv\_mask\_int64x2xm2\_int64xm2 (C function), 610  
 vlxsseg2hv\_mask\_int64x2xm4\_int64xm4 (C function), 610  
 vlxsseg2hv\_mask\_int8x2xm1\_int8xm1 (C function), 610  
 vlxsseg2hv\_mask\_int8x2xm2\_int8xm2 (C function), 610  
 vlxsseg2hv\_mask\_int8x2xm4\_int8xm4 (C function), 611  
 vlxsseg2wuv\_mask\_uint16x2xm1\_uint16xm1 (C function), 615  
 vlxsseg2wuv\_mask\_uint16x2xm2\_uint16xm2 (C function), 616  
 vlxsseg2wuv\_mask\_uint16x2xm4\_uint16xm4 (C function), 616  
 vlxsseg2wuv\_mask\_uint32x2xm1\_uint32xm1 (C function), 616  
 vlxsseg2wuv\_mask\_uint32x2xm2\_uint32xm2 (C function), 616  
 vlxsseg2wuv\_mask\_uint32x2xm4\_uint32xm4 (C function), 616  
 vlxsseg2wuv\_mask\_uint64x2xm1\_uint64xm1 (C function), 616  
 vlxsseg2wuv\_mask\_uint64x2xm2\_uint64xm2 (C function), 616  
 vlxsseg2wuv\_mask\_uint64x2xm4\_uint64xm4 (C function), 616  
 vlxsseg2wuv\_mask\_uint8x2xm1\_uint8xm1 (C function), 616  
 vlxsseg2wuv\_mask\_uint8x2xm2\_uint8xm2 (C function), 616  
 vlxsseg2wuv\_mask\_uint8x2xm4\_uint8xm4 (C function), 616  
 vlxsseg2wuv\_uint16x2xm1\_uint16xm1 (C function), 615  
 vlxsseg2wuv\_uint16x2xm2\_uint16xm2 (C function), 615  
 vlxsseg2wuv\_uint16x2xm4\_uint16xm4 (C function), 615  
 vlxsseg2wuv\_uint32x2xm1\_uint32xm1 (C function), 615

vlxsseg2wuv\_uint32x2xm2\_uint32xm2 (C function), 615  
 vlxsseg2wuv\_uint32x2xm4\_uint32xm4 (C function), 615  
 vlxsseg2wuv\_uint64x2xm1\_uint64xm1 (C function), 615  
 vlxsseg2wuv\_uint64x2xm2\_uint64xm2 (C function), 615  
 vlxsseg2wuv\_uint64x2xm4\_uint64xm4 (C function), 615  
 vlxsseg2wuv\_uint8x2xm1\_uint8xm1 (C function), 615  
 vlxsseg2wuv\_uint8x2xm2\_uint8xm2 (C function), 615  
 vlxsseg2wuv\_uint8x2xm4\_uint8xm4 (C function), 615  
 vlxsseg2wv\_int16x2xm1\_int16xm1 (C function), 613  
 vlxsseg2wv\_int16x2xm2\_int16xm2 (C function), 613  
 vlxsseg2wv\_int16x2xm4\_int16xm4 (C function), 613  
 vlxsseg2wv\_int32x2xm1\_int32xm1 (C function), 613  
 vlxsseg2wv\_int32x2xm2\_int32xm2 (C function), 613  
 vlxsseg2wv\_int32x2xm4\_int32xm4 (C function), 613  
 vlxsseg2wv\_int64x2xm1\_int64xm1 (C function), 613  
 vlxsseg2wv\_int64x2xm2\_int64xm2 (C function), 613  
 vlxsseg2wv\_int64x2xm4\_int64xm4 (C function), 613  
 vlxsseg2wv\_int8x2xm1\_int8xm1 (C function), 613  
 vlxsseg2wv\_int8x2xm2\_int8xm2 (C function), 613  
 vlxsseg2wv\_int8x2xm4\_int8xm4 (C function), 613  
 vlxsseg2wv\_mask\_int16x2xm1\_int16xm1 (C function), 614  
 vlxsseg2wv\_mask\_int16x2xm2\_int16xm2 (C function), 614  
 vlxsseg2wv\_mask\_int16x2xm4\_int16xm4 (C function), 614  
 vlxsseg2wv\_mask\_int32x2xm1\_int32xm1 (C function), 614  
 vlxsseg2wv\_mask\_int32x2xm2\_int32xm2 (C function), 614  
 vlxsseg2wv\_mask\_int32x2xm4\_int32xm4 (C function), 614  
 vlxsseg2wv\_mask\_int64x2xm1\_int64xm1 (C function), 614  
 vlxsseg2wv\_mask\_int64x2xm2\_int64xm2 (C function), 614  
 vlxsseg2wv\_mask\_int64x2xm4\_int64xm4 (C function), 614  
 vlxsseg2wv\_mask\_int8x2xm1\_int8xm1 (C function), 614  
 vlxsseg2wv\_mask\_int8x2xm2\_int8xm2 (C function), 614  
 vlxsseg2wv\_mask\_int8x2xm4\_int8xm4 (C function), 614  
 vlxsseg3buv\_mask\_uint16x3xm1\_int16xm1 (C function), 619  
 vlxsseg3buv\_mask\_uint16x3xm2\_int16xm2 (C function), 619  
 vlxsseg3buv\_mask\_uint32x3xm1\_int32xm1 (C function), 619  
 vlxsseg3buv\_mask\_uint32x3xm2\_int32xm2 (C function), 619  
 vlxsseg3buv\_mask\_uint64x3xm1\_int64xm1 (C function), 619  
 vlxsseg3buv\_mask\_uint64x3xm2\_int64xm2 (C function), 619  
 vlxsseg3buv\_mask\_uint8x3xm1\_int8xm1 (C function), 619  
 vlxsseg3buv\_mask\_uint8x3xm2\_int8xm2 (C function), 619  
 vlxsseg3bv\_int16x3xm1\_int16xm1 (C function), 617  
 vlxsseg3bv\_int16x3xm2\_int16xm2 (C function), 617  
 vlxsseg3bv\_int32x3xm1\_int32xm1 (C function), 617  
 vlxsseg3bv\_int32x3xm2\_int32xm2 (C function), 617  
 vlxsseg3bv\_int64x3xm1\_int64xm1 (C function), 617  
 vlxsseg3bv\_int64x3xm2\_int64xm2 (C function), 617  
 vlxsseg3bv\_int8x3xm1\_int8xm1 (C function), 617  
 vlxsseg3bv\_int8x3xm2\_int8xm2 (C function), 617  
 vlxsseg3bv\_mask\_int16x3xm1\_int16xm1 (C function), 617  
 vlxsseg3bv\_mask\_int16x3xm2\_int16xm2 (C function), 617  
 vlxsseg3bv\_mask\_int32x3xm1\_int32xm1 (C function), 617  
 vlxsseg3bv\_mask\_int32x3xm2\_int32xm2 (C function), 618  
 vlxsseg3bv\_mask\_int64x3xm1\_int64xm1 (C function), 618  
 vlxsseg3bv\_mask\_int64x3xm2\_int64xm2 (C function), 618  
 vlxsseg3bv\_mask\_int8x3xm1\_int8xm1 (C function), 618  
 vlxsseg3bv\_mask\_int8x3xm2\_int8xm2 (C function), 618  
 vlxsseg3ev\_float16x3xm1\_float16xm1 (C function), 620  
 vlxsseg3ev\_float16x3xm2\_float16xm2 (C function), 620  
 vlxsseg3ev\_float32x3xm1\_float32xm1 (C function), 620  
 vlxsseg3ev\_float32x3xm2\_float32xm2 (C function), 620  
 vlxsseg3ev\_float64x3xm1\_float64xm1 (C function), 620  
 vlxsseg3ev\_float64x3xm2\_float64xm2 (C function), 620  
 vlxsseg3ev\_int16x3xm1\_int16xm1 (C function), 620  
 vlxsseg3ev\_int16x3xm2\_int16xm2 (C function), 620  
 vlxsseg3ev\_int32x3xm1\_int32xm1 (C function), 620  
 vlxsseg3ev\_int32x3xm2\_int32xm2 (C function), 620  
 vlxsseg3ev\_int64x3xm1\_int64xm1 (C function), 620  
 vlxsseg3ev\_int64x3xm2\_int64xm2 (C function), 620  
 vlxsseg3ev\_int8x3xm1\_int8xm1 (C function), 620  
 vlxsseg3ev\_int8x3xm2\_int8xm2 (C function), 620  
 vlxsseg3ev\_mask\_float16x3xm1\_float16xm1 (C function), 621  
 vlxsseg3ev\_mask\_float16x3xm2\_float16xm2 (C function), 621  
 vlxsseg3ev\_mask\_float32x3xm1\_float32xm1 (C function), 621



<code>vlxseg3ev_mask_float32x3xm2_float32xm2</code> (C function), <a href="#">621</a>	<code>vlxseg3huv_mask_uint64x3xm2_uint64xm2</code> (C function), <a href="#">625</a>
<code>vlxseg3ev_mask_float64x3xm1_float64xm1</code> (C function), <a href="#">621</a>	<code>vlxseg3huv_mask_uint8x3xm1_uint8xm1</code> (C function), <a href="#">625</a>
<code>vlxseg3ev_mask_float64x3xm2_float64xm2</code> (C function), <a href="#">621</a>	<code>vlxseg3huv_mask_uint8x3xm2_uint8xm2</code> (C function), <a href="#">625</a>
<code>vlxseg3ev_mask_int16x3xm1_int16xm1</code> (C function), <a href="#">621</a>	<code>vlxseg3huv_uint16x3xm1_uint16xm1</code> (C function), <a href="#">624</a>
<code>vlxseg3ev_mask_int16x3xm2_int16xm2</code> (C function), <a href="#">621</a>	<code>vlxseg3huv_uint16x3xm2_uint16xm2</code> (C function), <a href="#">624</a>
<code>vlxseg3ev_mask_int32x3xm1_int32xm1</code> (C function), <a href="#">621</a>	<code>vlxseg3huv_uint32x3xm1_uint32xm1</code> (C function), <a href="#">624</a>
<code>vlxseg3ev_mask_int32x3xm2_int32xm2</code> (C function), <a href="#">622</a>	<code>vlxseg3huv_uint32x3xm2_uint32xm2</code> (C function), <a href="#">624</a>
<code>vlxseg3ev_mask_int64x3xm1_int64xm1</code> (C function), <a href="#">622</a>	<code>vlxseg3huv_uint64x3xm1_uint64xm1</code> (C function), <a href="#">624</a>
<code>vlxseg3ev_mask_int64x3xm2_int64xm2</code> (C function), <a href="#">622</a>	<code>vlxseg3huv_uint64x3xm2_uint64xm2</code> (C function), <a href="#">624</a>
<code>vlxseg3ev_mask_int8x3xm1_int8xm1</code> (C function), <a href="#">622</a>	<code>vlxseg3huv_uint8x3xm1_uint8xm1</code> (C function), <a href="#">624</a>
<code>vlxseg3ev_mask_int8x3xm2_int8xm2</code> (C function), <a href="#">622</a>	<code>vlxseg3huv_uint8x3xm2_uint8xm2</code> (C function), <a href="#">624</a>
<code>vlxseg3ev_mask_uint16x3xm1_uint16xm1</code> (C function), <a href="#">622</a>	<code>vlxseg3hv_int16x3xm1_int16xm1</code> (C function), <a href="#">623</a>
<code>vlxseg3ev_mask_uint16x3xm2_uint16xm2</code> (C function), <a href="#">622</a>	<code>vlxseg3hv_int16x3xm2_int16xm2</code> (C function), <a href="#">623</a>
<code>vlxseg3ev_mask_uint32x3xm1_uint32xm1</code> (C function), <a href="#">622</a>	<code>vlxseg3hv_int32x3xm1_int32xm1</code> (C function), <a href="#">623</a>
<code>vlxseg3ev_mask_uint32x3xm2_uint32xm2</code> (C function), <a href="#">622</a>	<code>vlxseg3hv_int32x3xm2_int32xm2</code> (C function), <a href="#">623</a>
<code>vlxseg3ev_mask_uint64x3xm1_uint64xm1</code> (C function), <a href="#">622</a>	<code>vlxseg3hv_int64x3xm1_int64xm1</code> (C function), <a href="#">623</a>
<code>vlxseg3ev_mask_uint64x3xm2_uint64xm2</code> (C function), <a href="#">622</a>	<code>vlxseg3hv_int64x3xm2_int64xm2</code> (C function), <a href="#">623</a>
<code>vlxseg3ev_mask_uint8x3xm1_uint8xm1</code> (C function), <a href="#">622</a>	<code>vlxseg3hv_int8x3xm1_int8xm1</code> (C function), <a href="#">623</a>
<code>vlxseg3ev_mask_uint8x3xm2_uint8xm2</code> (C function), <a href="#">622</a>	<code>vlxseg3hv_int8x3xm2_int8xm2</code> (C function), <a href="#">623</a>
<code>vlxseg3ev_uint16x3xm1_uint16xm1</code> (C function), <a href="#">620</a>	<code>vlxseg3hv_mask_int16x3xm1_int16xm1</code> (C function), <a href="#">623</a>
<code>vlxseg3ev_uint16x3xm2_uint16xm2</code> (C function), <a href="#">620</a>	<code>vlxseg3hv_mask_int16x3xm2_int16xm2</code> (C function), <a href="#">623</a>
<code>vlxseg3ev_uint32x3xm1_uint32xm1</code> (C function), <a href="#">620</a>	<code>vlxseg3hv_mask_int32x3xm1_int32xm1</code> (C function), <a href="#">623</a>
<code>vlxseg3ev_uint32x3xm2_uint32xm2</code> (C function), <a href="#">620</a>	<code>vlxseg3hv_mask_int32x3xm2_int32xm2</code> (C function), <a href="#">623</a>
<code>vlxseg3ev_uint64x3xm1_uint64xm1</code> (C function), <a href="#">620</a>	<code>vlxseg3hv_mask_int64x3xm1_int64xm1</code> (C function), <a href="#">624</a>
<code>vlxseg3ev_uint64x3xm2_uint64xm2</code> (C function), <a href="#">620</a>	<code>vlxseg3hv_mask_int64x3xm2_int64xm2</code> (C function), <a href="#">624</a>
<code>vlxseg3ev_uint8x3xm1_uint8xm1</code> (C function), <a href="#">621</a>	<code>vlxseg3hv_mask_int8x3xm1_int8xm1</code> (C function), <a href="#">624</a>
<code>vlxseg3ev_uint8x3xm2_uint8xm2</code> (C function), <a href="#">621</a>	<code>vlxseg3hv_mask_int8x3xm2_int8xm2</code> (C function), <a href="#">624</a>
<code>vlxseg3huv_mask_uint16x3xm1_uint16xm1</code> (C function), <a href="#">625</a>	<code>vlxseg3wuv_mask_uint16x3xm1_uint16xm1</code> (C function), <a href="#">628</a>
<code>vlxseg3huv_mask_uint16x3xm2_uint16xm2</code> (C function), <a href="#">625</a>	<code>vlxseg3wuv_mask_uint16x3xm2_uint16xm2</code> (C function), <a href="#">628</a>
<code>vlxseg3huv_mask_uint32x3xm1_uint32xm1</code> (C function), <a href="#">625</a>	<code>vlxseg3wuv_mask_uint32x3xm1_uint32xm1</code> (C function), <a href="#">628</a>
<code>vlxseg3huv_mask_uint32x3xm2_uint32xm2</code> (C function), <a href="#">625</a>	<code>vlxseg3wuv_mask_uint32x3xm2_uint32xm2</code> (C function), <a href="#">628</a>
<code>vlxseg3huv_mask_uint64x3xm1_uint64xm1</code> (C function), <a href="#">625</a>	<code>vlxseg3wuv_mask_uint64x3xm1_uint64xm1</code> (C function), <a href="#">628</a>
	<code>vlxseg3wuv_mask_uint64x3xm2_uint64xm2</code> (C function), <a href="#">628</a>
	<code>vlxseg3wuv_mask_uint8x3xm1_uint8xm1</code> (C function), <a href="#">628</a>
	<code>vlxseg3wuv_mask_uint8x3xm2_uint8xm2</code> (C function), <a href="#">628</a>
	<code>vlxseg3wuv_uint16x3xm1_uint16xm1</code> (C function), <a href="#">627</a>
	<code>vlxseg3wuv_uint16x3xm2_uint16xm2</code> (C function), <a href="#">627</a>



vlxsseg3wuv\_uint32x3xm1\_uint32xm1 (C function), 627  
 vlxsseg3wuv\_uint32x3xm2\_uint32xm2 (C function), 627  
 vlxsseg3wuv\_uint64x3xm1\_uint64xm1 (C function), 627  
 vlxsseg3wuv\_uint64x3xm2\_uint64xm2 (C function), 627  
 vlxsseg3wuv\_uint8x3xm1\_uint8xm1 (C function), 627  
 vlxsseg3wuv\_uint8x3xm2\_uint8xm2 (C function), 627  
 vlxsseg3wv\_int16x3xm1\_int16xm1 (C function), 626  
 vlxsseg3wv\_int16x3xm2\_int16xm2 (C function), 626  
 vlxsseg3wv\_int32x3xm1\_int32xm1 (C function), 626  
 vlxsseg3wv\_int32x3xm2\_int32xm2 (C function), 626  
 vlxsseg3wv\_int64x3xm1\_int64xm1 (C function), 626  
 vlxsseg3wv\_int64x3xm2\_int64xm2 (C function), 626  
 vlxsseg3wv\_int8x3xm1\_int8xm1 (C function), 626  
 vlxsseg3wv\_int8x3xm2\_int8xm2 (C function), 626  
 vlxsseg3wv\_mask\_int16x3xm1\_int16xm1 (C function), 626  
 vlxsseg3wv\_mask\_int16x3xm2\_int16xm2 (C function), 626  
 vlxsseg3wv\_mask\_int32x3xm1\_int32xm1 (C function), 626  
 vlxsseg3wv\_mask\_int32x3xm2\_int32xm2 (C function), 626  
 vlxsseg3wv\_mask\_int64x3xm1\_int64xm1 (C function), 626  
 vlxsseg3wv\_mask\_int64x3xm2\_int64xm2 (C function), 626  
 vlxsseg3wv\_mask\_int8x3xm1\_int8xm1 (C function), 627  
 vlxsseg3wv\_mask\_int8x3xm2\_int8xm2 (C function), 627  
 vlxsseg4buv\_mask\_uint16x4xm1\_uint16xm1 (C function), 630  
 vlxsseg4buv\_mask\_uint16x4xm2\_uint16xm2 (C function), 630  
 vlxsseg4buv\_mask\_uint32x4xm1\_uint32xm1 (C function), 631  
 vlxsseg4buv\_mask\_uint32x4xm2\_uint32xm2 (C function), 631  
 vlxsseg4buv\_mask\_uint64x4xm1\_uint64xm1 (C function), 631  
 vlxsseg4buv\_mask\_uint64x4xm2\_uint64xm2 (C function), 631  
 vlxsseg4buv\_mask\_uint8x4xm1\_uint8xm1 (C function), 631  
 vlxsseg4buv\_mask\_uint8x4xm2\_uint8xm2 (C function), 631  
 vlxsseg4buv\_uint16x4xm1\_uint16xm1 (C function), 630  
 vlxsseg4buv\_uint16x4xm2\_uint16xm2 (C function), 630  
 vlxsseg4buv\_uint32x4xm1\_uint32xm1 (C function), 630  
 vlxsseg4buv\_uint32x4xm2\_uint32xm2 (C function), 630  
 vlxsseg4buv\_uint64x4xm1\_uint64xm1 (C function), 630  
 vlxsseg4buv\_uint64x4xm2\_uint64xm2 (C function), 630  
 vlxsseg4buv\_uint8x4xm1\_uint8xm1 (C function), 630  
 vlxsseg4buv\_uint8x4xm2\_uint8xm2 (C function), 630  
 vlxsseg4bv\_int16x4xm1\_int16xm1 (C function), 629  
 vlxsseg4bv\_int16x4xm2\_int16xm2 (C function), 629  
 vlxsseg4bv\_int32x4xm1\_int32xm1 (C function), 629  
 vlxsseg4bv\_int32x4xm2\_int32xm2 (C function), 629  
 vlxsseg4bv\_int64x4xm1\_int64xm1 (C function), 629  
 vlxsseg4bv\_int64x4xm2\_int64xm2 (C function), 629  
 vlxsseg4bv\_mask\_int16x4xm1\_int16xm1 (C function), 629  
 vlxsseg4bv\_mask\_int16x4xm2\_int16xm2 (C function), 629  
 vlxsseg4bv\_mask\_int32x4xm1\_int32xm1 (C function), 629  
 vlxsseg4bv\_mask\_int32x4xm2\_int32xm2 (C function), 629  
 vlxsseg4bv\_mask\_int64x4xm1\_int64xm1 (C function), 629  
 vlxsseg4bv\_mask\_int64x4xm2\_int64xm2 (C function), 629  
 vlxsseg4bv\_mask\_int8x4xm1\_int8xm1 (C function), 629  
 vlxsseg4bv\_mask\_int8x4xm2\_int8xm2 (C function), 629  
 vlxsseg4ev\_float16x4xm1\_float16xm1 (C function), 631  
 vlxsseg4ev\_float16x4xm2\_float16xm2 (C function), 631  
 vlxsseg4ev\_float32x4xm1\_float32xm1 (C function), 631  
 vlxsseg4ev\_float32x4xm2\_float32xm2 (C function), 631  
 vlxsseg4ev\_float64x4xm1\_float64xm1 (C function), 632  
 vlxsseg4ev\_float64x4xm2\_float64xm2 (C function), 632  
 vlxsseg4ev\_int16x4xm1\_int16xm1 (C function), 632  
 vlxsseg4ev\_int16x4xm2\_int16xm2 (C function), 632  
 vlxsseg4ev\_int32x4xm1\_int32xm1 (C function), 632  
 vlxsseg4ev\_int32x4xm2\_int32xm2 (C function), 632  
 vlxsseg4ev\_int64x4xm1\_int64xm1 (C function), 632  
 vlxsseg4ev\_int64x4xm2\_int64xm2 (C function), 632  
 vlxsseg4ev\_int8x4xm1\_int8xm1 (C function), 632  
 vlxsseg4ev\_int8x4xm2\_int8xm2 (C function), 632  
 vlxsseg4ev\_mask\_float16x4xm1\_float16xm1 (C function), 633  
 vlxsseg4ev\_mask\_float16x4xm2\_float16xm2 (C function), 633  
 vlxsseg4ev\_mask\_float32x4xm1\_float32xm1 (C function), 633  
 vlxsseg4ev\_mask\_float32x4xm2\_float32xm2 (C function), 633  
 vlxsseg4ev\_mask\_float64x4xm1\_float64xm1 (C function), 633  
 vlxsseg4ev\_mask\_float64x4xm2\_float64xm2 (C function), 633  
 vlxsseg4ev\_mask\_int16x4xm1\_int16xm1 (C function), 633  
 vlxsseg4ev\_mask\_int16x4xm2\_int16xm2 (C function), 633  
 vlxsseg4ev\_mask\_int32x4xm1\_int32xm1 (C function), 633  
 vlxsseg4ev\_mask\_int32x4xm2\_int32xm2 (C function), 633

vlxsseg4ev\_mask\_int64x4xm1\_int64xm1 (C function), 633  
 vlxsseg4ev\_mask\_int64x4xm2\_int64xm2 (C function), 633  
 vlxsseg4ev\_mask\_int8x4xm1\_int8xm1 (C function), 633  
 vlxsseg4ev\_mask\_int8x4xm2\_int8xm2 (C function), 634  
 vlxsseg4ev\_mask\_uint16x4xm1\_uint16xm1 (C function), 634  
 vlxsseg4ev\_mask\_uint16x4xm2\_uint16xm2 (C function), 634  
 vlxsseg4ev\_mask\_uint32x4xm1\_uint32xm1 (C function), 634  
 vlxsseg4ev\_mask\_uint32x4xm2\_uint32xm2 (C function), 634  
 vlxsseg4ev\_mask\_uint64x4xm1\_uint64xm1 (C function), 634  
 vlxsseg4ev\_mask\_uint64x4xm2\_uint64xm2 (C function), 634  
 vlxsseg4ev\_mask\_uint8x4xm1\_uint8xm1 (C function), 634  
 vlxsseg4ev\_mask\_uint8x4xm2\_uint8xm2 (C function), 634  
 vlxsseg4ev\_uint16x4xm1\_uint16xm1 (C function), 632  
 vlxsseg4ev\_uint16x4xm2\_uint16xm2 (C function), 632  
 vlxsseg4ev\_uint32x4xm1\_uint32xm1 (C function), 632  
 vlxsseg4ev\_uint32x4xm2\_uint32xm2 (C function), 632  
 vlxsseg4ev\_uint64x4xm1\_uint64xm1 (C function), 632  
 vlxsseg4ev\_uint64x4xm2\_uint64xm2 (C function), 632  
 vlxsseg4ev\_uint8x4xm1\_uint8xm1 (C function), 632  
 vlxsseg4ev\_uint8x4xm2\_uint8xm2 (C function), 632  
 vlxsseg4huv\_mask\_uint16x4xm1\_uint16xm1 (C function), 636  
 vlxsseg4huv\_mask\_uint16x4xm2\_uint16xm2 (C function), 636  
 vlxsseg4huv\_mask\_uint32x4xm1\_uint32xm1 (C function), 637  
 vlxsseg4huv\_mask\_uint32x4xm2\_uint32xm2 (C function), 637  
 vlxsseg4huv\_mask\_uint64x4xm1\_uint64xm1 (C function), 637  
 vlxsseg4huv\_mask\_uint64x4xm2\_uint64xm2 (C function), 637  
 vlxsseg4huv\_mask\_uint8x4xm1\_uint8xm1 (C function), 637  
 vlxsseg4huv\_mask\_uint8x4xm2\_uint8xm2 (C function), 637  
 vlxsseg4huv\_uint16x4xm1\_uint16xm1 (C function), 636  
 vlxsseg4huv\_uint16x4xm2\_uint16xm2 (C function), 636  
 vlxsseg4huv\_uint32x4xm1\_uint32xm1 (C function), 636  
 vlxsseg4huv\_uint32x4xm2\_uint32xm2 (C function), 636  
 vlxsseg4huv\_uint64x4xm1\_uint64xm1 (C function), 636  
 vlxsseg4huv\_uint64x4xm2\_uint64xm2 (C function), 636  
 vlxsseg4huv\_uint8x4xm1\_uint8xm1 (C function), 636  
 vlxsseg4huv\_uint8x4xm2\_uint8xm2 (C function), 636  
 vlxsseg4hv\_int16x4xm1\_int16xm1 (C function), 635  
 vlxsseg4hv\_int16x4xm2\_int16xm2 (C function), 635  
 vlxsseg4hv\_int32x4xm1\_int32xm1 (C function), 635  
 vlxsseg4hv\_int32x4xm2\_int32xm2 (C function), 635  
 vlxsseg4hv\_int64x4xm1\_int64xm1 (C function), 635  
 vlxsseg4hv\_int64x4xm2\_int64xm2 (C function), 635  
 vlxsseg4hv\_int8x4xm1\_int8xm1 (C function), 635  
 vlxsseg4hv\_int8x4xm2\_int8xm2 (C function), 635  
 vlxsseg4hv\_mask\_int16x4xm1\_int16xm1 (C function), 635  
 vlxsseg4hv\_mask\_int16x4xm2\_int16xm2 (C function), 635  
 vlxsseg4hv\_mask\_int32x4xm1\_int32xm1 (C function), 635  
 vlxsseg4hv\_mask\_int32x4xm2\_int32xm2 (C function), 635  
 vlxsseg4hv\_mask\_int64x4xm1\_int64xm1 (C function), 635  
 vlxsseg4hv\_mask\_int64x4xm2\_int64xm2 (C function), 635  
 vlxsseg4hv\_mask\_int8x4xm1\_int8xm1 (C function), 635  
 vlxsseg4hv\_mask\_int8x4xm2\_int8xm2 (C function), 635  
 vlxsseg4hv\_mask\_uint16x4xm1\_uint16xm1 (C function), 639  
 vlxsseg4hv\_mask\_uint16x4xm2\_uint16xm2 (C function), 639  
 vlxsseg4hv\_mask\_uint32x4xm1\_uint32xm1 (C function), 639  
 vlxsseg4hv\_mask\_uint32x4xm2\_uint32xm2 (C function), 639  
 vlxsseg4hv\_mask\_uint64x4xm1\_uint64xm1 (C function), 640  
 vlxsseg4hv\_mask\_uint64x4xm2\_uint64xm2 (C function), 640  
 vlxsseg4hv\_mask\_uint8x4xm1\_uint8xm1 (C function), 640  
 vlxsseg4hv\_mask\_uint8x4xm2\_uint8xm2 (C function), 640  
 vlxsseg4wuv\_uint16x4xm1\_uint16xm1 (C function), 639  
 vlxsseg4wuv\_uint16x4xm2\_uint16xm2 (C function), 639  
 vlxsseg4wuv\_uint32x4xm1\_uint32xm1 (C function), 639  
 vlxsseg4wuv\_uint32x4xm2\_uint32xm2 (C function), 639  
 vlxsseg4wuv\_uint64x4xm1\_uint64xm1 (C function), 639  
 vlxsseg4wuv\_uint64x4xm2\_uint64xm2 (C function), 639  
 vlxsseg4wuv\_uint8x4xm1\_uint8xm1 (C function), 639  
 vlxsseg4wuv\_uint8x4xm2\_uint8xm2 (C function), 639  
 vlxsseg4wv\_int16x4xm1\_int16xm1 (C function), 637  
 vlxsseg4wv\_int16x4xm2\_int16xm2 (C function), 637  
 vlxsseg4wv\_int32x4xm1\_int32xm1 (C function), 637  
 vlxsseg4wv\_int32x4xm2\_int32xm2 (C function), 637  
 vlxsseg4wv\_int64x4xm1\_int64xm1 (C function), 638  
 vlxsseg4wv\_int64x4xm2\_int64xm2 (C function), 638  
 vlxsseg4wv\_int8x4xm1\_int8xm1 (C function), 638  
 vlxsseg4wv\_int8x4xm2\_int8xm2 (C function), 638

vlxsseg4wv\_mask\_int16x4xm1\_int16xm1 (C function), 638  
 vlxsseg4wv\_mask\_int16x4xm2\_int16xm2 (C function), 638  
 vlxsseg4wv\_mask\_int32x4xm1\_int32xm1 (C function), 638  
 vlxsseg4wv\_mask\_int32x4xm2\_int32xm2 (C function), 638  
 vlxsseg4wv\_mask\_int64x4xm1\_int64xm1 (C function), 638  
 vlxsseg4wv\_mask\_int64x4xm2\_int64xm2 (C function), 638  
 vlxsseg4wv\_mask\_int8x4xm1\_int8xm1 (C function), 638  
 vlxsseg4wv\_mask\_int8x4xm2\_int8xm2 (C function), 638  
 vlxsseg5buv\_mask\_uint16x5xm1\_uint16xm1 (C function), 641  
 vlxsseg5buv\_mask\_uint32x5xm1\_uint32xm1 (C function), 641  
 vlxsseg5buv\_mask\_uint64x5xm1\_uint64xm1 (C function), 642  
 vlxsseg5buv\_mask\_uint8x5xm1\_uint8xm1 (C function), 642  
 vlxsseg5buv\_uint16x5xm1\_uint16xm1 (C function), 641  
 vlxsseg5buv\_uint32x5xm1\_uint32xm1 (C function), 641  
 vlxsseg5buv\_uint64x5xm1\_uint64xm1 (C function), 641  
 vlxsseg5buv\_uint8x5xm1\_uint8xm1 (C function), 641  
 vlxsseg5bv\_int16x5xm1\_int16xm1 (C function), 640  
 vlxsseg5bv\_int32x5xm1\_int32xm1 (C function), 640  
 vlxsseg5bv\_int64x5xm1\_int64xm1 (C function), 640  
 vlxsseg5bv\_int8x5xm1\_int8xm1 (C function), 640  
 vlxsseg5bv\_mask\_int16x5xm1\_int16xm1 (C function), 640  
 vlxsseg5bv\_mask\_int32x5xm1\_int32xm1 (C function), 641  
 vlxsseg5bv\_mask\_int64x5xm1\_int64xm1 (C function), 641  
 vlxsseg5bv\_mask\_int8x5xm1\_int8xm1 (C function), 641  
 vlxsseg5ev\_float16x5xm1\_float16xm1 (C function), 642  
 vlxsseg5ev\_float32x5xm1\_float32xm1 (C function), 642  
 vlxsseg5ev\_float64x5xm1\_float64xm1 (C function), 642  
 vlxsseg5ev\_int16x5xm1\_int16xm1 (C function), 642  
 vlxsseg5ev\_int32x5xm1\_int32xm1 (C function), 642  
 vlxsseg5ev\_int64x5xm1\_int64xm1 (C function), 642  
 vlxsseg5ev\_int8x5xm1\_int8xm1 (C function), 642  
 vlxsseg5ev\_mask\_float16x5xm1\_float16xm1 (C function), 643  
 vlxsseg5ev\_mask\_float32x5xm1\_float32xm1 (C function), 643  
 vlxsseg5ev\_mask\_float64x5xm1\_float64xm1 (C function), 643  
 vlxsseg5ev\_mask\_int16x5xm1\_int16xm1 (C function), 643  
 vlxsseg5ev\_mask\_int32x5xm1\_int32xm1 (C function), 643  
 vlxsseg5ev\_mask\_int64x5xm1\_int64xm1 (C function), 643  
 vlxsseg5ev\_mask\_int8x5xm1\_int8xm1 (C function), 643  
 vlxsseg5ev\_uint16x5xm1\_uint16xm1 (C function), 642  
 vlxsseg5ev\_uint32x5xm1\_uint32xm1 (C function), 642  
 vlxsseg5ev\_uint64x5xm1\_uint64xm1 (C function), 642  
 vlxsseg5ev\_uint8x5xm1\_uint8xm1 (C function), 642  
 vlxsseg5huv\_mask\_uint16x5xm1\_uint16xm1 (C function), 645  
 vlxsseg5huv\_mask\_uint32x5xm1\_uint32xm1 (C function), 645  
 vlxsseg5huv\_mask\_uint64x5xm1\_uint64xm1 (C function), 645  
 vlxsseg5huv\_mask\_uint8x5xm1\_uint8xm1 (C function), 645  
 vlxsseg5huv\_uint16x5xm1\_uint16xm1 (C function), 645  
 vlxsseg5huv\_uint32x5xm1\_uint32xm1 (C function), 645  
 vlxsseg5huv\_uint64x5xm1\_uint64xm1 (C function), 645  
 vlxsseg5huv\_uint8x5xm1\_uint8xm1 (C function), 645  
 vlxsseg5hv\_int16x5xm1\_int16xm1 (C function), 644  
 vlxsseg5hv\_int32x5xm1\_int32xm1 (C function), 644  
 vlxsseg5hv\_int64x5xm1\_int64xm1 (C function), 644  
 vlxsseg5hv\_int8x5xm1\_int8xm1 (C function), 644  
 vlxsseg5hv\_mask\_int16x5xm1\_int16xm1 (C function), 644  
 vlxsseg5hv\_mask\_int32x5xm1\_int32xm1 (C function), 644  
 vlxsseg5hv\_mask\_int64x5xm1\_int64xm1 (C function), 644  
 vlxsseg5hv\_mask\_int8x5xm1\_int8xm1 (C function), 644  
 vlxsseg5wuv\_mask\_uint16x5xm1\_uint16xm1 (C function), 647  
 vlxsseg5wuv\_mask\_uint32x5xm1\_uint32xm1 (C function), 647  
 vlxsseg5wuv\_mask\_uint64x5xm1\_uint64xm1 (C function), 647  
 vlxsseg5wuv\_mask\_uint8x5xm1\_uint8xm1 (C function), 647  
 vlxsseg5wuv\_uint16x5xm1\_uint16xm1 (C function), 647  
 vlxsseg5wuv\_uint32x5xm1\_uint32xm1 (C function), 647  
 vlxsseg5wuv\_uint64x5xm1\_uint64xm1 (C function), 647  
 vlxsseg5wuv\_uint8x5xm1\_uint8xm1 (C function), 647  
 vlxsseg5wv\_int16x5xm1\_int16xm1 (C function), 646  
 vlxsseg5wv\_int32x5xm1\_int32xm1 (C function), 646  
 vlxsseg5wv\_int64x5xm1\_int64xm1 (C function), 646  
 vlxsseg5wv\_int8x5xm1\_int8xm1 (C function), 646



vlxsseg7buv\_mask\_uint16x7xm1\_uint16xm1 (C function), 656  
 vlxsseg7buv\_mask\_uint32x7xm1\_uint32xm1 (C function), 656  
 vlxsseg7buv\_mask\_uint64x7xm1\_uint64xm1 (C function), 656  
 vlxsseg7buv\_mask\_uint8x7xm1\_uint8xm1 (C function), 657  
 vlxsseg7buv\_uint16x7xm1\_uint16xm1 (C function), 656  
 vlxsseg7buv\_uint32x7xm1\_uint32xm1 (C function), 656  
 vlxsseg7buv\_uint64x7xm1\_uint64xm1 (C function), 656  
 vlxsseg7buv\_uint8x7xm1\_uint8xm1 (C function), 656  
 vlxsseg7bv\_int16x7xm1\_int16xm1 (C function), 655  
 vlxsseg7bv\_int32x7xm1\_int32xm1 (C function), 655  
 vlxsseg7bv\_int64x7xm1\_int64xm1 (C function), 655  
 vlxsseg7bv\_int8x7xm1\_int8xm1 (C function), 655  
 vlxsseg7bv\_mask\_int16x7xm1\_int16xm1 (C function), 655  
 vlxsseg7bv\_mask\_int32x7xm1\_int32xm1 (C function), 655  
 vlxsseg7bv\_mask\_int64x7xm1\_int64xm1 (C function), 656  
 vlxsseg7bv\_mask\_int8x7xm1\_int8xm1 (C function), 656  
 vlxsseg7ev\_float16x7xm1\_float16xm1 (C function), 657  
 vlxsseg7ev\_float32x7xm1\_float32xm1 (C function), 657  
 vlxsseg7ev\_float64x7xm1\_float64xm1 (C function), 657  
 vlxsseg7ev\_int16x7xm1\_int16xm1 (C function), 657  
 vlxsseg7ev\_int32x7xm1\_int32xm1 (C function), 657  
 vlxsseg7ev\_int64x7xm1\_int64xm1 (C function), 657  
 vlxsseg7ev\_int8x7xm1\_int8xm1 (C function), 657  
 vlxsseg7ev\_mask\_float16x7xm1\_float16xm1 (C function), 658  
 vlxsseg7ev\_mask\_float32x7xm1\_float32xm1 (C function), 658  
 vlxsseg7ev\_mask\_float64x7xm1\_float64xm1 (C function), 658  
 vlxsseg7ev\_mask\_int16x7xm1\_int16xm1 (C function), 658  
 vlxsseg7ev\_mask\_int32x7xm1\_int32xm1 (C function), 658  
 vlxsseg7ev\_mask\_int64x7xm1\_int64xm1 (C function), 658  
 vlxsseg7ev\_mask\_int8x7xm1\_int8xm1 (C function), 658  
 vlxsseg7ev\_mask\_uint16x7xm1\_uint16xm1 (C function), 658  
 vlxsseg7ev\_mask\_uint32x7xm1\_uint32xm1 (C function), 658  
 vlxsseg7ev\_mask\_uint64x7xm1\_uint64xm1 (C function), 658  
 vlxsseg7ev\_mask\_uint8x7xm1\_uint8xm1 (C function), 658  
 vlxsseg7ev\_uint16x7xm1\_uint16xm1 (C function), 657  
 vlxsseg7ev\_uint32x7xm1\_uint32xm1 (C function), 657  
 vlxsseg7ev\_uint64x7xm1\_uint64xm1 (C function), 657  
 vlxsseg7ev\_uint8x7xm1\_uint8xm1 (C function), 657  
 vlxsseg7huv\_mask\_uint16x7xm1\_uint16xm1 (C function), 660  
 vlxsseg7huv\_mask\_uint32x7xm1\_uint32xm1 (C function), 660  
 vlxsseg7huv\_mask\_uint64x7xm1\_uint64xm1 (C function), 660  
 vlxsseg7huv\_mask\_uint8x7xm1\_uint8xm1 (C function), 660  
 vlxsseg7huv\_uint16x7xm1\_uint16xm1 (C function), 660  
 vlxsseg7huv\_uint32x7xm1\_uint32xm1 (C function), 660  
 vlxsseg7huv\_uint64x7xm1\_uint64xm1 (C function), 660  
 vlxsseg7huv\_uint8x7xm1\_uint8xm1 (C function), 660  
 vlxsseg7hv\_int16x7xm1\_int16xm1 (C function), 659  
 vlxsseg7hv\_int32x7xm1\_int32xm1 (C function), 659  
 vlxsseg7hv\_int64x7xm1\_int64xm1 (C function), 659  
 vlxsseg7hv\_int8x7xm1\_int8xm1 (C function), 659  
 vlxsseg7hv\_mask\_int16x7xm1\_int16xm1 (C function), 659  
 vlxsseg7hv\_mask\_int32x7xm1\_int32xm1 (C function), 659  
 vlxsseg7hv\_mask\_int64x7xm1\_int64xm1 (C function), 659  
 vlxsseg7hv\_mask\_int8x7xm1\_int8xm1 (C function), 659  
 vlxsseg7wuv\_mask\_uint16x7xm1\_uint16xm1 (C function), 662  
 vlxsseg7wuv\_mask\_uint32x7xm1\_uint32xm1 (C function), 662  
 vlxsseg7wuv\_mask\_uint64x7xm1\_uint64xm1 (C function), 662  
 vlxsseg7wuv\_mask\_uint8x7xm1\_uint8xm1 (C function), 662  
 vlxsseg7wuv\_uint16x7xm1\_uint16xm1 (C function), 662  
 vlxsseg7wuv\_uint32x7xm1\_uint32xm1 (C function), 662  
 vlxsseg7wuv\_uint64x7xm1\_uint64xm1 (C function), 662  
 vlxsseg7wuv\_uint8x7xm1\_uint8xm1 (C function), 662  
 vlxsseg7wv\_int16x7xm1\_int16xm1 (C function), 661  
 vlxsseg7wv\_int32x7xm1\_int32xm1 (C function), 661  
 vlxsseg7wv\_int64x7xm1\_int64xm1 (C function), 661  
 vlxsseg7wv\_int8x7xm1\_int8xm1 (C function), 661  
 vlxsseg7wv\_mask\_int16x7xm1\_int16xm1 (C function), 661  
 vlxsseg7wv\_mask\_int32x7xm1\_int32xm1 (C function), 661  
 vlxsseg7wv\_mask\_int64x7xm1\_int64xm1 (C function), 661  
 vlxsseg7wv\_mask\_int8x7xm1\_int8xm1 (C function), 661  
 vlxsseg8buv\_mask\_uint16x8xm1\_uint16xm1 (C function), 664  
 vlxsseg8buv\_mask\_uint32x8xm1\_uint32xm1 (C function), 664  
 vlxsseg8buv\_mask\_uint64x8xm1\_uint64xm1 (C function), 664  
 vlxsseg8buv\_mask\_uint8x8xm1\_uint8xm1 (C function), 664



- 664  
 vlxsseg8buv\_uint16x8xm1\_uint16xm1 (C function), 664  
 vlxsseg8buv\_uint32x8xm1\_uint32xm1 (C function), 664  
 vlxsseg8buv\_uint64x8xm1\_uint64xm1 (C function), 664  
 vlxsseg8buv\_uint8x8xm1\_uint8xm1 (C function), 664  
 vlxsseg8bv\_int16x8xm1\_int16xm1 (C function), 663  
 vlxsseg8bv\_int32x8xm1\_int32xm1 (C function), 663  
 vlxsseg8bv\_int64x8xm1\_int64xm1 (C function), 663  
 vlxsseg8bv\_int8x8xm1\_int8xm1 (C function), 663  
 vlxsseg8bv\_mask\_int16x8xm1\_int16xm1 (C function), 663  
 vlxsseg8bv\_mask\_int32x8xm1\_int32xm1 (C function), 663  
 vlxsseg8bv\_mask\_int64x8xm1\_int64xm1 (C function), 663  
 vlxsseg8bv\_mask\_int8x8xm1\_int8xm1 (C function), 663  
 vlxsseg8ev\_float16x8xm1\_float16xm1 (C function), 665  
 vlxsseg8ev\_float32x8xm1\_float32xm1 (C function), 665  
 vlxsseg8ev\_float64x8xm1\_float64xm1 (C function), 665  
 vlxsseg8ev\_int16x8xm1\_int16xm1 (C function), 665  
 vlxsseg8ev\_int32x8xm1\_int32xm1 (C function), 665  
 vlxsseg8ev\_int64x8xm1\_int64xm1 (C function), 665  
 vlxsseg8ev\_int8x8xm1\_int8xm1 (C function), 665  
 vlxsseg8ev\_mask\_float16x8xm1\_float16xm1 (C function), 665  
 vlxsseg8ev\_mask\_float32x8xm1\_float32xm1 (C function), 665  
 vlxsseg8ev\_mask\_float64x8xm1\_float64xm1 (C function), 665  
 vlxsseg8ev\_mask\_int16x8xm1\_int16xm1 (C function), 665  
 vlxsseg8ev\_mask\_int32x8xm1\_int32xm1 (C function), 666  
 vlxsseg8ev\_mask\_int64x8xm1\_int64xm1 (C function), 666  
 vlxsseg8ev\_mask\_int8x8xm1\_int8xm1 (C function), 666  
 vlxsseg8ev\_mask\_uint16x8xm1\_uint16xm1 (C function), 666  
 vlxsseg8ev\_mask\_uint32x8xm1\_uint32xm1 (C function), 666  
 vlxsseg8ev\_mask\_uint64x8xm1\_uint64xm1 (C function), 666  
 vlxsseg8ev\_mask\_uint8x8xm1\_uint8xm1 (C function), 666  
 vlxsseg8ev\_uint16x8xm1\_uint16xm1 (C function), 665  
 vlxsseg8ev\_uint32x8xm1\_uint32xm1 (C function), 665  
 vlxsseg8ev\_uint64x8xm1\_uint64xm1 (C function), 665  
 vlxsseg8ev\_uint8x8xm1\_uint8xm1 (C function), 665  
 vlxsseg8huv\_mask\_uint16x8xm1\_uint16xm1 (C function), 668  
 vlxsseg8huv\_mask\_uint32x8xm1\_uint32xm1 (C function), 668  
 vlxsseg8huv\_mask\_uint64x8xm1\_uint64xm1 (C function), 668  
 vlxsseg8huv\_mask\_uint8x8xm1\_uint8xm1 (C function), 668  
 vlxsseg8huv\_uint16x8xm1\_uint16xm1 (C function), 667  
 vlxsseg8huv\_uint32x8xm1\_uint32xm1 (C function), 667  
 vlxsseg8huv\_uint64x8xm1\_uint64xm1 (C function), 667  
 vlxsseg8huv\_uint8x8xm1\_uint8xm1 (C function), 667  
 vlxsseg8hv\_int16x8xm1\_int16xm1 (C function), 666  
 vlxsseg8hv\_int32x8xm1\_int32xm1 (C function), 666  
 vlxsseg8hv\_int64x8xm1\_int64xm1 (C function), 666  
 vlxsseg8hv\_int8x8xm1\_int8xm1 (C function), 666  
 vlxsseg8hv\_mask\_int16x8xm1\_int16xm1 (C function), 667  
 vlxsseg8hv\_mask\_int32x8xm1\_int32xm1 (C function), 667  
 vlxsseg8hv\_mask\_int64x8xm1\_int64xm1 (C function), 667  
 vlxsseg8hv\_mask\_int8x8xm1\_int8xm1 (C function), 667  
 vlxsseg8wuv\_mask\_uint16x8xm1\_uint16xm1 (C function), 669  
 vlxsseg8wuv\_mask\_uint32x8xm1\_uint32xm1 (C function), 670  
 vlxsseg8wuv\_mask\_uint64x8xm1\_uint64xm1 (C function), 670  
 vlxsseg8wuv\_mask\_uint8x8xm1\_uint8xm1 (C function), 670  
 vlxsseg8wuv\_uint16x8xm1\_uint16xm1 (C function), 669  
 vlxsseg8wuv\_uint32x8xm1\_uint32xm1 (C function), 669  
 vlxsseg8wuv\_uint64x8xm1\_uint64xm1 (C function), 669  
 vlxsseg8wuv\_uint8x8xm1\_uint8xm1 (C function), 669  
 vlxsseg8wv\_int16x8xm1\_int16xm1 (C function), 668  
 vlxsseg8wv\_int32x8xm1\_int32xm1 (C function), 668  
 vlxsseg8wv\_int64x8xm1\_int64xm1 (C function), 668  
 vlxsseg8wv\_int8x8xm1\_int8xm1 (C function), 668  
 vlxsseg8wv\_mask\_int16x8xm1\_int16xm1 (C function), 669  
 vlxsseg8wv\_mask\_int32x8xm1\_int32xm1 (C function), 669  
 vlxsseg8wv\_mask\_int64x8xm1\_int64xm1 (C function), 669  
 vlxsseg8wv\_mask\_int8x8xm1\_int8xm1 (C function), 669  
 vlxsseg8wv\_uint16x8xm1\_uint16xm1 (C function), 449  
 vlxsseg8wv\_uint32x8xm1\_uint32xm1 (C function), 449  
 vlxsseg8wv\_uint64x8xm1\_uint64xm1 (C function), 449  
 vlxsseg8wv\_uint8x8xm1\_uint8xm1 (C function), 449  
 vlxsseg8wv\_uint16xm2 (C function), 449  
 vlxsseg8wv\_uint16xm4 (C function), 449  
 vlxsseg8wv\_uint16xm8 (C function), 449  
 vlxsseg8wv\_uint32xm1 (C function), 449  
 vlxsseg8wv\_uint32xm2 (C function), 449  
 vlxsseg8wv\_uint32xm4 (C function), 450  
 vlxsseg8wv\_uint32xm8 (C function), 450  
 vlxsseg8wv\_uint64xm1 (C function), 450  
 vlxsseg8wv\_uint64xm2 (C function), 450  
 vlxsseg8wv\_uint64xm4 (C function), 450  
 vlxsseg8wv\_uint64xm8 (C function), 450  
 vlxsseg8wv\_uint8xm1 (C function), 450  
 vlxsseg8wv\_uint8xm2 (C function), 450



vlxwuv\_mask\_uint8xm4 (C function), 450  
vlxwuv\_mask\_uint8xm8 (C function), 450  
vlxwuv\_uint16xm1 (C function), 449  
vlxwuv\_uint16xm2 (C function), 449  
vlxwuv\_uint16xm4 (C function), 449  
vlxwuv\_uint16xm8 (C function), 449  
vlxwuv\_uint32xm1 (C function), 449  
vlxwuv\_uint32xm2 (C function), 449  
vlxwuv\_uint32xm4 (C function), 449  
vlxwuv\_uint32xm8 (C function), 449  
vlxwuv\_uint64xm1 (C function), 449  
vlxwuv\_uint64xm2 (C function), 449  
vlxwuv\_uint64xm4 (C function), 449  
vlxwuv\_uint64xm8 (C function), 449  
vlxwuv\_uint8xm1 (C function), 449  
vlxwuv\_uint8xm2 (C function), 449  
vlxwuv\_uint8xm4 (C function), 449  
vlxwuv\_uint8xm8 (C function), 449  
vlxwv\_int16xm1 (C function), 447  
vlxwv\_int16xm2 (C function), 447  
vlxwv\_int16xm4 (C function), 447  
vlxwv\_int16xm8 (C function), 447  
vlxwv\_int32xm1 (C function), 447  
vlxwv\_int32xm2 (C function), 447  
vlxwv\_int32xm4 (C function), 447  
vlxwv\_int32xm8 (C function), 447  
vlxwv\_int64xm1 (C function), 447  
vlxwv\_int64xm2 (C function), 447  
vlxwv\_int64xm4 (C function), 447  
vlxwv\_int64xm8 (C function), 447  
vlxwv\_int8xm1 (C function), 447  
vlxwv\_int8xm2 (C function), 447  
vlxwv\_int8xm4 (C function), 447  
vlxwv\_int8xm8 (C function), 447  
vlxwv\_mask\_int16xm1 (C function), 448  
vlxwv\_mask\_int16xm2 (C function), 448  
vlxwv\_mask\_int16xm4 (C function), 448  
vlxwv\_mask\_int16xm8 (C function), 448  
vlxwv\_mask\_int32xm1 (C function), 448  
vlxwv\_mask\_int32xm2 (C function), 448  
vlxwv\_mask\_int32xm4 (C function), 448  
vlxwv\_mask\_int32xm8 (C function), 448  
vlxwv\_mask\_int64xm1 (C function), 448  
vlxwv\_mask\_int64xm2 (C function), 448  
vlxwv\_mask\_int64xm4 (C function), 448  
vlxwv\_mask\_int64xm8 (C function), 448  
vlxwv\_mask\_int8xm1 (C function), 448  
vlxwv\_mask\_int8xm2 (C function), 448  
vlxwv\_mask\_int8xm4 (C function), 448  
vlxwv\_mask\_int8xm8 (C function), 448  
vmaccvv\_int16xm1 (C function), 187  
vmaccvv\_int16xm2 (C function), 187  
vmaccvv\_int16xm4 (C function), 187  
vmaccvv\_int16xm8 (C function), 187  
vmaccvv\_int32xm1 (C function), 187  
vmaccvv\_int32xm2 (C function), 187  
vmaccvv\_int32xm4 (C function), 187  
vmaccvv\_int32xm8 (C function), 187  
vmaccvv\_int64xm1 (C function), 187  
vmaccvv\_int64xm2 (C function), 187  
vmaccvv\_int64xm4 (C function), 187  
vmaccvv\_int64xm8 (C function), 187  
vmaccvv\_int8xm1 (C function), 187  
vmaccvv\_int8xm2 (C function), 187  
vmaccvv\_int8xm4 (C function), 187  
vmaccvv\_int8xm8 (C function), 187  
vmaccvv\_mask\_int16xm1 (C function), 188  
vmaccvv\_mask\_int16xm2 (C function), 188  
vmaccvv\_mask\_int16xm4 (C function), 188  
vmaccvv\_mask\_int32xm1 (C function), 188  
vmaccvv\_mask\_int32xm2 (C function), 188  
vmaccvv\_mask\_int32xm4 (C function), 188  
vmaccvv\_mask\_int64xm1 (C function), 188  
vmaccvv\_mask\_int64xm2 (C function), 188  
vmaccvv\_mask\_int64xm4 (C function), 188  
vmaccvv\_mask\_int8xm1 (C function), 188  
vmaccvv\_mask\_int8xm2 (C function), 188  
vmaccvv\_mask\_int8xm4 (C function), 188  
vmaccvv\_mask\_uint16xm1 (C function), 189  
vmaccvv\_mask\_uint16xm2 (C function), 189  
vmaccvv\_mask\_uint16xm4 (C function), 189  
vmaccvv\_mask\_uint32xm1 (C function), 189  
vmaccvv\_mask\_uint32xm2 (C function), 189  
vmaccvv\_mask\_uint32xm4 (C function), 189  
vmaccvv\_mask\_uint64xm1 (C function), 189  
vmaccvv\_mask\_uint64xm2 (C function), 189  
vmaccvv\_mask\_uint64xm4 (C function), 189  
vmaccvv\_mask\_uint8xm1 (C function), 189  
vmaccvv\_mask\_uint8xm2 (C function), 189  
vmaccvv\_mask\_uint8xm4 (C function), 189  
vmaccvv\_uint16xm1 (C function), 187  
vmaccvv\_uint16xm2 (C function), 187  
vmaccvv\_uint16xm4 (C function), 187  
vmaccvv\_uint16xm8 (C function), 187  
vmaccvv\_uint32xm1 (C function), 187  
vmaccvv\_uint32xm2 (C function), 187  
vmaccvv\_uint32xm4 (C function), 188  
vmaccvv\_uint32xm8 (C function), 188  
vmaccvv\_uint64xm1 (C function), 188  
vmaccvv\_uint64xm2 (C function), 188  
vmaccvv\_uint64xm4 (C function), 188  
vmaccvv\_uint64xm8 (C function), 188  
vmaccvv\_uint8xm1 (C function), 188  
vmaccvv\_uint8xm2 (C function), 188  
vmaccvv\_uint8xm4 (C function), 188  
vmaccvv\_uint8xm8 (C function), 188  
vmaccvx\_int16xm1 (C function), 189  
vmaccvx\_int16xm2 (C function), 189



vmadcvvm\_mask\_e64xm8\_int64xm8 (C function), 194  
vmadcvvm\_mask\_e64xm8\_uint64xm8 (C function), 194  
vmadcvvm\_mask\_e8xm1\_int8xm1 (C function), 195  
vmadcvvm\_mask\_e8xm1\_uint8xm1 (C function), 195  
vmadcvvm\_mask\_e8xm2\_int8xm2 (C function), 195  
vmadcvvm\_mask\_e8xm2\_uint8xm2 (C function), 195  
vmadcvvm\_mask\_e8xm4\_int8xm4 (C function), 195  
vmadcvvm\_mask\_e8xm4\_uint8xm4 (C function), 195  
vmadcvvm\_mask\_e8xm8\_int8xm8 (C function), 195  
vmadcvvm\_mask\_e8xm8\_uint8xm8 (C function), 195  
vmadcvvm\_mask\_e16xm1\_int16xm1 (C function), 195  
vmadcvvm\_mask\_e16xm1\_uint16xm1 (C function), 195  
vmadcvvm\_mask\_e16xm2\_int16xm2 (C function), 195  
vmadcvvm\_mask\_e16xm2\_uint16xm2 (C function), 195  
vmadcvvm\_mask\_e16xm4\_int16xm4 (C function), 195  
vmadcvvm\_mask\_e16xm4\_uint16xm4 (C function), 195  
vmadcvvm\_mask\_e16xm8\_int16xm8 (C function), 195  
vmadcvvm\_mask\_e16xm8\_uint16xm8 (C function), 195  
vmadcvvm\_mask\_e32xm1\_int32xm1 (C function), 195  
vmadcvvm\_mask\_e32xm1\_uint32xm1 (C function), 195  
vmadcvvm\_mask\_e32xm2\_int32xm2 (C function), 196  
vmadcvvm\_mask\_e32xm2\_uint32xm2 (C function), 196  
vmadcvvm\_mask\_e32xm4\_int32xm4 (C function), 196  
vmadcvvm\_mask\_e32xm4\_uint32xm4 (C function), 196  
vmadcvvm\_mask\_e32xm8\_int32xm8 (C function), 196  
vmadcvvm\_mask\_e32xm8\_uint32xm8 (C function), 196  
vmadcvvm\_mask\_e64xm1\_int64xm1 (C function), 196  
vmadcvvm\_mask\_e64xm1\_uint64xm1 (C function), 196  
vmadcvvm\_mask\_e64xm2\_int64xm2 (C function), 196  
vmadcvvm\_mask\_e64xm2\_uint64xm2 (C function), 196  
vmadcvvm\_mask\_e64xm4\_int64xm4 (C function), 196  
vmadcvvm\_mask\_e64xm4\_uint64xm4 (C function), 196  
vmadcvvm\_mask\_e64xm8\_int64xm8 (C function), 196  
vmadcvvm\_mask\_e64xm8\_uint64xm8 (C function), 196  
vmadcvvm\_mask\_e8xm1\_int8xm1 (C function), 196  
vmadcvvm\_mask\_e8xm1\_uint8xm1 (C function), 196  
vmadcvvm\_mask\_e8xm2\_int8xm2 (C function), 196  
vmadcvvm\_mask\_e8xm2\_uint8xm2 (C function), 196  
vmadcvvm\_mask\_e8xm4\_int8xm4 (C function), 196  
vmadcvvm\_mask\_e8xm4\_uint8xm4 (C function), 196  
vmadcvvm\_mask\_e8xm8\_int8xm8 (C function), 196  
vmadcvvm\_mask\_e8xm8\_uint8xm8 (C function), 196  
vmaddvv\_int16xm1 (C function), 197  
vmaddvv\_int16xm2 (C function), 197  
vmaddvv\_int16xm4 (C function), 197  
vmaddvv\_int16xm8 (C function), 197  
vmaddvv\_int32xm1 (C function), 197  
vmaddvv\_int32xm2 (C function), 197  
vmaddvv\_int32xm4 (C function), 197  
vmaddvv\_int32xm8 (C function), 197  
vmaddvv\_int64xm1 (C function), 197  
vmaddvv\_int64xm2 (C function), 197  
vmaddvv\_int64xm4 (C function), 197  
vmaddvv\_int64xm8 (C function), 197  
vmaddvv\_int8xm1 (C function), 197  
vmaddvv\_int8xm2 (C function), 197  
vmaddvv\_int8xm4 (C function), 197  
vmaddvv\_int8xm8 (C function), 197  
vmaddvv\_mask\_int16xm1 (C function), 198  
vmaddvv\_mask\_int16xm2 (C function), 198  
vmaddvv\_mask\_int16xm4 (C function), 198  
vmaddvv\_mask\_int32xm1 (C function), 198  
vmaddvv\_mask\_int32xm2 (C function), 198  
vmaddvv\_mask\_int32xm4 (C function), 198  
vmaddvv\_mask\_int64xm1 (C function), 198  
vmaddvv\_mask\_int64xm2 (C function), 198  
vmaddvv\_mask\_int64xm4 (C function), 198  
vmaddvv\_mask\_int8xm1 (C function), 198  
vmaddvv\_mask\_int8xm2 (C function), 198  
vmaddvv\_mask\_int8xm4 (C function), 198  
vmaddvv\_mask\_uint16xm1 (C function), 198  
vmaddvv\_mask\_uint16xm2 (C function), 198  
vmaddvv\_mask\_uint16xm4 (C function), 199  
vmaddvv\_mask\_uint32xm1 (C function), 199  
vmaddvv\_mask\_uint32xm2 (C function), 199  
vmaddvv\_mask\_uint32xm4 (C function), 199  
vmaddvv\_mask\_uint64xm1 (C function), 199  
vmaddvv\_mask\_uint64xm2 (C function), 199  
vmaddvv\_mask\_uint64xm4 (C function), 199  
vmaddvv\_mask\_uint8xm1 (C function), 199  
vmaddvv\_mask\_uint8xm2 (C function), 199  
vmaddvv\_mask\_uint8xm4 (C function), 199  
vmaddvv\_uint16xm1 (C function), 197  
vmaddvv\_uint16xm2 (C function), 197  
vmaddvv\_uint16xm4 (C function), 197  
vmaddvv\_uint16xm8 (C function), 197  
vmaddvv\_uint32xm1 (C function), 197  
vmaddvv\_uint32xm2 (C function), 197  
vmaddvv\_uint32xm4 (C function), 197  
vmaddvv\_uint32xm8 (C function), 197  
vmaddvv\_uint64xm1 (C function), 198  
vmaddvv\_uint64xm2 (C function), 198  
vmaddvv\_uint64xm4 (C function), 198  
vmaddvv\_uint64xm8 (C function), 198  
vmaddvv\_uint8xm1 (C function), 198  
vmaddvv\_uint8xm2 (C function), 198  
vmaddvv\_uint8xm4 (C function), 198  
vmaddvv\_uint8xm8 (C function), 198  
vmaddvx\_int16xm1 (C function), 199  
vmaddvx\_int16xm2 (C function), 199  
vmaddvx\_int16xm4 (C function), 199  
vmaddvx\_int16xm8 (C function), 199  
vmaddvx\_int32xm1 (C function), 199  
vmaddvx\_int32xm2 (C function), 199  
vmaddvx\_int32xm4 (C function), 199  
vmaddvx\_int32xm8 (C function), 199  
vmaddvx\_int64xm1 (C function), 199  
vmaddvx\_int64xm2 (C function), 199

vmaddvx\_int64xm4 (C function), 199  
 vmaddvx\_int64xm8 (C function), 200  
 vmaddvx\_int8xm1 (C function), 200  
 vmaddvx\_int8xm2 (C function), 200  
 vmaddvx\_int8xm4 (C function), 200  
 vmaddvx\_int8xm8 (C function), 200  
 vmaddvx\_mask\_int16xm1 (C function), 200  
 vmaddvx\_mask\_int16xm2 (C function), 200  
 vmaddvx\_mask\_int16xm4 (C function), 201  
 vmaddvx\_mask\_int32xm1 (C function), 201  
 vmaddvx\_mask\_int32xm2 (C function), 201  
 vmaddvx\_mask\_int32xm4 (C function), 201  
 vmaddvx\_mask\_int64xm1 (C function), 201  
 vmaddvx\_mask\_int64xm2 (C function), 201  
 vmaddvx\_mask\_int64xm4 (C function), 201  
 vmaddvx\_mask\_int8xm1 (C function), 201  
 vmaddvx\_mask\_int8xm2 (C function), 201  
 vmaddvx\_mask\_int8xm4 (C function), 201  
 vmaddvx\_mask\_uint16xm1 (C function), 201  
 vmaddvx\_mask\_uint16xm2 (C function), 201  
 vmaddvx\_mask\_uint16xm4 (C function), 201  
 vmaddvx\_mask\_uint32xm1 (C function), 201  
 vmaddvx\_mask\_uint32xm2 (C function), 201  
 vmaddvx\_mask\_uint32xm4 (C function), 201  
 vmaddvx\_mask\_uint64xm1 (C function), 201  
 vmaddvx\_mask\_uint64xm2 (C function), 201  
 vmaddvx\_mask\_uint64xm4 (C function), 201  
 vmaddvx\_mask\_uint8xm1 (C function), 201  
 vmaddvx\_mask\_uint8xm2 (C function), 201  
 vmaddvx\_mask\_uint8xm4 (C function), 201  
 vmaddvx\_uint16xm1 (C function), 200  
 vmaddvx\_uint16xm2 (C function), 200  
 vmaddvx\_uint16xm4 (C function), 200  
 vmaddvx\_uint16xm8 (C function), 200  
 vmaddvx\_uint32xm1 (C function), 200  
 vmaddvx\_uint32xm2 (C function), 200  
 vmaddvx\_uint32xm4 (C function), 200  
 vmaddvx\_uint32xm8 (C function), 200  
 vmaddvx\_uint64xm1 (C function), 200  
 vmaddvx\_uint64xm2 (C function), 200  
 vmaddvx\_uint64xm4 (C function), 200  
 vmaddvx\_uint64xm8 (C function), 200  
 vmaddvx\_uint8xm1 (C function), 200  
 vmaddvx\_uint8xm2 (C function), 200  
 vmaddvx\_uint8xm4 (C function), 200  
 vmaddvx\_uint8xm8 (C function), 200  
 vmandmm\_e16xm1 (C function), 799  
 vmandmm\_e16xm2 (C function), 799  
 vmandmm\_e16xm4 (C function), 799  
 vmandmm\_e16xm8 (C function), 800  
 vmandmm\_e32xm1 (C function), 800  
 vmandmm\_e32xm2 (C function), 800  
 vmandmm\_e32xm4 (C function), 800  
 vmandmm\_e32xm8 (C function), 800  
 vmandmm\_e64xm1 (C function), 800  
 vmandmm\_e64xm2 (C function), 800  
 vmandmm\_e64xm4 (C function), 800  
 vmandmm\_e64xm8 (C function), 800  
 vmandmm\_e8xm1 (C function), 800  
 vmandmm\_e8xm2 (C function), 800  
 vmandmm\_e8xm4 (C function), 800  
 vmandmm\_e8xm8 (C function), 800  
 vmandnotmm\_e16xm1 (C function), 800  
 vmandnotmm\_e16xm2 (C function), 800  
 vmandnotmm\_e16xm4 (C function), 800  
 vmandnotmm\_e16xm8 (C function), 800  
 vmandnotmm\_e32xm1 (C function), 800  
 vmandnotmm\_e32xm2 (C function), 800  
 vmandnotmm\_e32xm4 (C function), 800  
 vmandnotmm\_e32xm8 (C function), 800  
 vmandnotmm\_e64xm1 (C function), 800  
 vmandnotmm\_e64xm2 (C function), 800  
 vmandnotmm\_e64xm4 (C function), 800  
 vmandnotmm\_e64xm8 (C function), 800  
 vmandnotmm\_e8xm1 (C function), 800  
 vmandnotmm\_e8xm2 (C function), 800  
 vmandnotmm\_e8xm4 (C function), 800  
 vmandnotmm\_e8xm8 (C function), 801  
 vmaxuvv\_mask\_uint16xm1 (C function), 205  
 vmaxuvv\_mask\_uint16xm2 (C function), 205  
 vmaxuvv\_mask\_uint16xm4 (C function), 206  
 vmaxuvv\_mask\_uint16xm8 (C function), 206  
 vmaxuvv\_mask\_uint32xm1 (C function), 206  
 vmaxuvv\_mask\_uint32xm2 (C function), 206  
 vmaxuvv\_mask\_uint32xm4 (C function), 206  
 vmaxuvv\_mask\_uint32xm8 (C function), 206  
 vmaxuvv\_mask\_uint64xm1 (C function), 206  
 vmaxuvv\_mask\_uint64xm2 (C function), 206  
 vmaxuvv\_mask\_uint64xm4 (C function), 206  
 vmaxuvv\_mask\_uint64xm8 (C function), 206  
 vmaxuvv\_mask\_uint8xm1 (C function), 206  
 vmaxuvv\_mask\_uint8xm2 (C function), 206  
 vmaxuvv\_mask\_uint8xm4 (C function), 206  
 vmaxuvv\_mask\_uint8xm8 (C function), 206  
 vmaxuvv\_uint16xm1 (C function), 205  
 vmaxuvv\_uint16xm2 (C function), 205  
 vmaxuvv\_uint16xm4 (C function), 205  
 vmaxuvv\_uint16xm8 (C function), 205  
 vmaxuvv\_uint32xm1 (C function), 205  
 vmaxuvv\_uint32xm2 (C function), 205  
 vmaxuvv\_uint32xm4 (C function), 205  
 vmaxuvv\_uint32xm8 (C function), 205  
 vmaxuvv\_uint64xm1 (C function), 205  
 vmaxuvv\_uint64xm2 (C function), 205  
 vmaxuvv\_uint64xm4 (C function), 205  
 vmaxuvv\_uint64xm8 (C function), 205  
 vmaxuvv\_uint8xm1 (C function), 205  
 vmaxuvv\_uint8xm2 (C function), 205



vmaxuvv\_uint8xm4 (C function), 205  
vmaxuvv\_uint8xm8 (C function), 205  
vmaxuvx\_mask\_uint16xm1 (C function), 207  
vmaxuvx\_mask\_uint16xm2 (C function), 207  
vmaxuvx\_mask\_uint16xm4 (C function), 207  
vmaxuvx\_mask\_uint16xm8 (C function), 207  
vmaxuvx\_mask\_uint32xm1 (C function), 207  
vmaxuvx\_mask\_uint32xm2 (C function), 207  
vmaxuvx\_mask\_uint32xm4 (C function), 207  
vmaxuvx\_mask\_uint32xm8 (C function), 207  
vmaxuvx\_mask\_uint64xm1 (C function), 207  
vmaxuvx\_mask\_uint64xm2 (C function), 207  
vmaxuvx\_mask\_uint64xm4 (C function), 207  
vmaxuvx\_mask\_uint64xm8 (C function), 207  
vmaxuvx\_mask\_uint8xm1 (C function), 207  
vmaxuvx\_mask\_uint8xm2 (C function), 208  
vmaxuvx\_mask\_uint8xm4 (C function), 208  
vmaxuvx\_mask\_uint8xm8 (C function), 208  
vmaxuvx\_uint16xm1 (C function), 206  
vmaxuvx\_uint16xm2 (C function), 206  
vmaxuvx\_uint16xm4 (C function), 206  
vmaxuvx\_uint16xm8 (C function), 206  
vmaxuvx\_uint32xm1 (C function), 206  
vmaxuvx\_uint32xm2 (C function), 207  
vmaxuvx\_uint32xm4 (C function), 207  
vmaxuvx\_uint32xm8 (C function), 207  
vmaxuvx\_uint64xm1 (C function), 207  
vmaxuvx\_uint64xm2 (C function), 207  
vmaxuvx\_uint64xm4 (C function), 207  
vmaxuvx\_uint64xm8 (C function), 207  
vmaxuvx\_uint8xm1 (C function), 207  
vmaxuvx\_uint8xm2 (C function), 207  
vmaxuvx\_uint8xm4 (C function), 207  
vmaxuvx\_uint8xm8 (C function), 207  
vmaxvv\_int16xm1 (C function), 202  
vmaxvv\_int16xm2 (C function), 202  
vmaxvv\_int16xm4 (C function), 202  
vmaxvv\_int16xm8 (C function), 202  
vmaxvv\_int32xm1 (C function), 202  
vmaxvv\_int32xm2 (C function), 202  
vmaxvv\_int32xm4 (C function), 202  
vmaxvv\_int32xm8 (C function), 202  
vmaxvv\_int64xm1 (C function), 202  
vmaxvv\_int64xm2 (C function), 202  
vmaxvv\_int64xm4 (C function), 202  
vmaxvv\_int64xm8 (C function), 202  
vmaxvv\_int8xm1 (C function), 202  
vmaxvv\_int8xm2 (C function), 202  
vmaxvv\_int8xm4 (C function), 202  
vmaxvv\_int8xm8 (C function), 202  
vmaxvv\_mask\_int16xm1 (C function), 202  
vmaxvv\_mask\_int16xm2 (C function), 202  
vmaxvv\_mask\_int16xm4 (C function), 202  
vmaxvv\_mask\_int16xm8 (C function), 202

vmaxvv\_mask\_int32xm1 (C function), 203  
vmaxvv\_mask\_int32xm2 (C function), 203  
vmaxvv\_mask\_int32xm4 (C function), 203  
vmaxvv\_mask\_int32xm8 (C function), 203  
vmaxvv\_mask\_int64xm1 (C function), 203  
vmaxvv\_mask\_int64xm2 (C function), 203  
vmaxvv\_mask\_int64xm4 (C function), 203  
vmaxvv\_mask\_int64xm8 (C function), 203  
vmaxvv\_mask\_int8xm1 (C function), 203  
vmaxvv\_mask\_int8xm2 (C function), 203  
vmaxvv\_mask\_int8xm4 (C function), 203  
vmaxvv\_mask\_int8xm8 (C function), 203  
vmaxvx\_int16xm1 (C function), 203  
vmaxvx\_int16xm2 (C function), 203  
vmaxvx\_int16xm4 (C function), 203  
vmaxvx\_int16xm8 (C function), 203  
vmaxvx\_int32xm1 (C function), 203  
vmaxvx\_int32xm2 (C function), 203  
vmaxvx\_int32xm4 (C function), 203  
vmaxvx\_int32xm8 (C function), 203  
vmaxvx\_int64xm1 (C function), 204  
vmaxvx\_int64xm2 (C function), 204  
vmaxvx\_int64xm4 (C function), 204  
vmaxvx\_int64xm8 (C function), 204  
vmaxvx\_int8xm1 (C function), 204  
vmaxvx\_int8xm2 (C function), 204  
vmaxvx\_int8xm4 (C function), 204  
vmaxvx\_int8xm8 (C function), 204  
vmaxvx\_mask\_int16xm1 (C function), 204  
vmaxvx\_mask\_int16xm2 (C function), 204  
vmaxvx\_mask\_int16xm4 (C function), 204  
vmaxvx\_mask\_int16xm8 (C function), 204  
vmaxvx\_mask\_int32xm1 (C function), 204  
vmaxvx\_mask\_int32xm2 (C function), 204  
vmaxvx\_mask\_int32xm4 (C function), 204  
vmaxvx\_mask\_int32xm8 (C function), 204  
vmaxvx\_mask\_int64xm1 (C function), 204  
vmaxvx\_mask\_int64xm2 (C function), 204  
vmaxvx\_mask\_int64xm4 (C function), 204  
vmaxvx\_mask\_int64xm8 (C function), 204  
vmaxvx\_mask\_int8xm1 (C function), 204  
vmaxvx\_mask\_int8xm2 (C function), 204  
vmaxvx\_mask\_int8xm4 (C function), 204  
vmaxvx\_mask\_int8xm8 (C function), 205  
vmclrm\_e16xm1 (C function), 801  
vmclrm\_e16xm2 (C function), 801  
vmclrm\_e16xm4 (C function), 801  
vmclrm\_e16xm8 (C function), 801  
vmclrm\_e32xm1 (C function), 801  
vmclrm\_e32xm2 (C function), 801  
vmclrm\_e32xm4 (C function), 801  
vmclrm\_e32xm8 (C function), 801  
vmclrm\_e64xm1 (C function), 801  
vmclrm\_e64xm2 (C function), 801





vmmergevxm\_mask\_uint32xm4 (C function), 826  
vmmergevxm\_mask\_uint32xm8 (C function), 826  
vmmergevxm\_mask\_uint64xm1 (C function), 826  
vmmergevxm\_mask\_uint64xm2 (C function), 826  
vmmergevxm\_mask\_uint64xm4 (C function), 826  
vmmergevxm\_mask\_uint64xm8 (C function), 826  
vmmergevxm\_mask\_uint8xm1 (C function), 826  
vmmergevxm\_mask\_uint8xm2 (C function), 826  
vmmergevxm\_mask\_uint8xm4 (C function), 826  
vmmergevxm\_mask\_uint8xm8 (C function), 826  
vmfeqvfe16xm1\_float16xm1 (C function), 139  
vmfeqvfe16xm2\_float16xm2 (C function), 139  
vmfeqvfe16xm4\_float16xm4 (C function), 139  
vmfeqvfe16xm8\_float16xm8 (C function), 139  
vmfeqvfe32xm1\_float32xm1 (C function), 139  
vmfeqvfe32xm2\_float32xm2 (C function), 139  
vmfeqvfe32xm4\_float32xm4 (C function), 139  
vmfeqvfe32xm8\_float32xm8 (C function), 139  
vmfeqvfe64xm1\_float64xm1 (C function), 139  
vmfeqvfe64xm2\_float64xm2 (C function), 139  
vmfeqvfe64xm4\_float64xm4 (C function), 139  
vmfeqvfe64xm8\_float64xm8 (C function), 139  
vmfeqvfe\_mask\_e16xm1\_float16xm1 (C function), 139  
vmfeqvfe\_mask\_e16xm2\_float16xm2 (C function), 139  
vmfeqvfe\_mask\_e16xm4\_float16xm4 (C function), 139  
vmfeqvfe\_mask\_e16xm8\_float16xm8 (C function), 139  
vmfeqvfe\_mask\_e32xm1\_float32xm1 (C function), 139  
vmfeqvfe\_mask\_e32xm2\_float32xm2 (C function), 139  
vmfeqvfe\_mask\_e32xm4\_float32xm4 (C function), 139  
vmfeqvfe\_mask\_e32xm8\_float32xm8 (C function), 139  
vmfeqvfe\_mask\_e64xm1\_float64xm1 (C function), 139  
vmfeqvfe\_mask\_e64xm2\_float64xm2 (C function), 140  
vmfeqvfe\_mask\_e64xm4\_float64xm4 (C function), 140  
vmfeqvfe\_mask\_e64xm8\_float64xm8 (C function), 140  
vmfeqvve16xm1\_float16xm1 (C function), 140  
vmfeqvve16xm2\_float16xm2 (C function), 140  
vmfeqvve16xm4\_float16xm4 (C function), 140  
vmfeqvve16xm8\_float16xm8 (C function), 140  
vmfeqvve32xm1\_float32xm1 (C function), 140  
vmfeqvve32xm2\_float32xm2 (C function), 140  
vmfeqvve32xm4\_float32xm4 (C function), 140  
vmfeqvve32xm8\_float32xm8 (C function), 140  
vmfeqvve64xm1\_float64xm1 (C function), 140  
vmfeqvve64xm2\_float64xm2 (C function), 140  
vmfeqvve64xm4\_float64xm4 (C function), 140  
vmfeqvve64xm8\_float64xm8 (C function), 140  
vmfeqvve\_mask\_e16xm1\_float16xm1 (C function), 140  
vmfeqvve\_mask\_e16xm2\_float16xm2 (C function), 140  
vmfeqvve\_mask\_e16xm4\_float16xm4 (C function), 141  
vmfeqvve\_mask\_e16xm8\_float16xm8 (C function), 141  
vmfeqvve\_mask\_e32xm1\_float32xm1 (C function), 141  
vmfeqvve\_mask\_e32xm2\_float32xm2 (C function), 141  
vmfeqvve\_mask\_e32xm4\_float32xm4 (C function), 141  
vmfeqvve\_mask\_e32xm8\_float32xm8 (C function), 141  
vmfeqvve\_mask\_e64xm1\_float64xm1 (C function), 141  
vmfeqvve\_mask\_e64xm2\_float64xm2 (C function), 141  
vmfeqvve\_mask\_e64xm4\_float64xm4 (C function), 141  
vmfeqvve\_mask\_e64xm8\_float64xm8 (C function), 141  
vmfeqvve\_mask\_e64xm1\_float64xm1 (C function), 142  
vmfeqvve\_mask\_e64xm2\_float64xm2 (C function), 142  
vmfeqvve\_mask\_e64xm4\_float64xm4 (C function), 142  
vmfeqvve\_mask\_e64xm8\_float64xm8 (C function), 142  
vmfeqvve\_mask\_e32xm1\_float32xm1 (C function), 142  
vmfeqvve\_mask\_e32xm2\_float32xm2 (C function), 142  
vmfeqvve\_mask\_e32xm4\_float32xm4 (C function), 142  
vmfeqvve\_mask\_e32xm8\_float32xm8 (C function), 142  
vmfeqvve\_mask\_e64xm1\_float64xm1 (C function), 142  
vmfeqvve\_mask\_e64xm2\_float64xm2 (C function), 142  
vmfeqvve\_mask\_e64xm4\_float64xm4 (C function), 142  
vmfeqvve\_mask\_e64xm8\_float64xm8 (C function), 142  
vmfeqvve\_e16xm1\_float16xm1 (C function), 143  
vmfeqvve\_e16xm2\_float16xm2 (C function), 143  
vmfeqvve\_e16xm4\_float16xm4 (C function), 143  
vmfeqvve\_e16xm8\_float16xm8 (C function), 143  
vmfeqvve\_e32xm1\_float32xm1 (C function), 143  
vmfeqvve\_e32xm2\_float32xm2 (C function), 143  
vmfeqvve\_e32xm4\_float32xm4 (C function), 143  
vmfeqvve\_e32xm8\_float32xm8 (C function), 143  
vmfeqvve\_e64xm1\_float64xm1 (C function), 143  
vmfeqvve\_e64xm2\_float64xm2 (C function), 143  
vmfeqvve\_e64xm4\_float64xm4 (C function), 143  
vmfeqvve\_e64xm8\_float64xm8 (C function), 143  
vmfeqvve\_mask\_e16xm1\_float16xm1 (C function), 143  
vmfeqvve\_mask\_e16xm2\_float16xm2 (C function), 143  
vmfeqvve\_mask\_e16xm4\_float16xm4 (C function), 143  
vmfeqvve\_mask\_e16xm8\_float16xm8 (C function), 143  
vmfeqvve\_mask\_e32xm1\_float32xm1 (C function), 143  
vmfeqvve\_mask\_e32xm2\_float32xm2 (C function), 144  
vmfeqvve\_mask\_e32xm4\_float32xm4 (C function), 144  
vmfeqvve\_mask\_e32xm8\_float32xm8 (C function), 144  
vmfeqvve\_mask\_e64xm1\_float64xm1 (C function), 144  
vmfeqvve\_mask\_e64xm2\_float64xm2 (C function), 144  
vmfeqvve\_mask\_e64xm4\_float64xm4 (C function), 144  
vmfeqvve\_mask\_e64xm8\_float64xm8 (C function), 144  
vmfgetvfe16xm1\_float16xm1 (C function), 144  
vmfgetvfe16xm2\_float16xm2 (C function), 144

vmfgtvf\_e16xm4\_float16xm4 (C function), 144  
 vmfgtvf\_e16xm8\_float16xm8 (C function), 144  
 vmfgtvf\_e32xm1\_float32xm1 (C function), 144  
 vmfgtvf\_e32xm2\_float32xm2 (C function), 144  
 vmfgtvf\_e32xm4\_float32xm4 (C function), 144  
 vmfgtvf\_e32xm8\_float32xm8 (C function), 144  
 vmfgtvf\_e64xm1\_float64xm1 (C function), 144  
 vmfgtvf\_e64xm2\_float64xm2 (C function), 144  
 vmfgtvf\_e64xm4\_float64xm4 (C function), 144  
 vmfgtvf\_e64xm8\_float64xm8 (C function), 145  
 vmfgtvf\_mask\_e16xm1\_float16xm1 (C function), 145  
 vmfgtvf\_mask\_e16xm2\_float16xm2 (C function), 145  
 vmfgtvf\_mask\_e16xm4\_float16xm4 (C function), 145  
 vmfgtvf\_mask\_e16xm8\_float16xm8 (C function), 145  
 vmfgtvf\_mask\_e32xm1\_float32xm1 (C function), 145  
 vmfgtvf\_mask\_e32xm2\_float32xm2 (C function), 145  
 vmfgtvf\_mask\_e32xm4\_float32xm4 (C function), 145  
 vmfgtvf\_mask\_e32xm8\_float32xm8 (C function), 145  
 vmfgtvf\_mask\_e64xm1\_float64xm1 (C function), 145  
 vmfgtvf\_mask\_e64xm2\_float64xm2 (C function), 145  
 vmfgtvf\_mask\_e64xm4\_float64xm4 (C function), 145  
 vmfgtvf\_mask\_e64xm8\_float64xm8 (C function), 145  
 vmfgtvv\_e16xm1\_float16xm1 (C function), 145  
 vmfgtvv\_e16xm2\_float16xm2 (C function), 145  
 vmfgtvv\_e16xm4\_float16xm4 (C function), 146  
 vmfgtvv\_e16xm8\_float16xm8 (C function), 146  
 vmfgtvv\_e32xm1\_float32xm1 (C function), 146  
 vmfgtvv\_e32xm2\_float32xm2 (C function), 146  
 vmfgtvv\_e32xm4\_float32xm4 (C function), 146  
 vmfgtvv\_e32xm8\_float32xm8 (C function), 146  
 vmfgtvv\_e64xm1\_float64xm1 (C function), 146  
 vmfgtvv\_e64xm2\_float64xm2 (C function), 146  
 vmfgtvv\_e64xm4\_float64xm4 (C function), 146  
 vmfgtvv\_e64xm8\_float64xm8 (C function), 146  
 vmfgtvv\_mask\_e16xm1\_float16xm1 (C function), 146  
 vmfgtvv\_mask\_e16xm2\_float16xm2 (C function), 146  
 vmfgtvv\_mask\_e16xm4\_float16xm4 (C function), 146  
 vmfgtvv\_mask\_e16xm8\_float16xm8 (C function), 146  
 vmfgtvv\_mask\_e32xm1\_float32xm1 (C function), 146  
 vmfgtvv\_mask\_e32xm2\_float32xm2 (C function), 146  
 vmfgtvv\_mask\_e32xm4\_float32xm4 (C function), 146  
 vmfgtvv\_mask\_e32xm8\_float32xm8 (C function), 146  
 vmfgtvv\_mask\_e64xm1\_float64xm1 (C function), 146  
 vmfgtvv\_mask\_e64xm2\_float64xm2 (C function), 147  
 vmfgtvv\_mask\_e64xm4\_float64xm4 (C function), 147  
 vmfgtvv\_mask\_e64xm8\_float64xm8 (C function), 147  
 vmfirstm\_e16xm1 (C function), 802  
 vmfirstm\_e16xm2 (C function), 802  
 vmfirstm\_e16xm4 (C function), 802  
 vmfirstm\_e16xm8 (C function), 802  
 vmfirstm\_e32xm1 (C function), 802  
 vmfirstm\_e32xm2 (C function), 802  
 vmfirstm\_e32xm4 (C function), 802  
 vmfirstm\_e32xm8 (C function), 802  
 vmfirstm\_e64xm1 (C function), 802  
 vmfirstm\_e64xm2 (C function), 802  
 vmfirstm\_e64xm4 (C function), 802  
 vmfirstm\_e64xm8 (C function), 802  
 vmfirstm\_e8xm1 (C function), 802  
 vmfirstm\_e8xm2 (C function), 802  
 vmfirstm\_e8xm4 (C function), 802  
 vmfirstm\_e8xm8 (C function), 803  
 vmfirstm\_mask\_e16xm1 (C function), 803  
 vmfirstm\_mask\_e16xm2 (C function), 803  
 vmfirstm\_mask\_e16xm4 (C function), 803  
 vmfirstm\_mask\_e16xm8 (C function), 803  
 vmfirstm\_mask\_e32xm1 (C function), 803  
 vmfirstm\_mask\_e32xm2 (C function), 803  
 vmfirstm\_mask\_e32xm4 (C function), 803  
 vmfirstm\_mask\_e32xm8 (C function), 803  
 vmfirstm\_mask\_e64xm1 (C function), 803  
 vmfirstm\_mask\_e64xm2 (C function), 803  
 vmfirstm\_mask\_e64xm4 (C function), 803  
 vmfirstm\_mask\_e64xm8 (C function), 803  
 vmfirstm\_mask\_e8xm1 (C function), 803  
 vmfirstm\_mask\_e8xm2 (C function), 803  
 vmfirstm\_mask\_e8xm4 (C function), 803  
 vmfirstm\_mask\_e8xm8 (C function), 803  
 vmflevf\_e16xm1\_float16xm1 (C function), 147  
 vmflevf\_e16xm2\_float16xm2 (C function), 147  
 vmflevf\_e16xm4\_float16xm4 (C function), 147  
 vmflevf\_e16xm8\_float16xm8 (C function), 147  
 vmflevf\_e32xm1\_float32xm1 (C function), 147  
 vmflevf\_e32xm2\_float32xm2 (C function), 147  
 vmflevf\_e32xm4\_float32xm4 (C function), 147  
 vmflevf\_e32xm8\_float32xm8 (C function), 147  
 vmflevf\_e64xm1\_float64xm1 (C function), 147  
 vmflevf\_e64xm2\_float64xm2 (C function), 147  
 vmflevf\_e64xm4\_float64xm4 (C function), 147  
 vmflevf\_e64xm8\_float64xm8 (C function), 147  
 vmflevf\_mask\_e16xm1\_float16xm1 (C function), 147  
 vmflevf\_mask\_e16xm2\_float16xm2 (C function), 147  
 vmflevf\_mask\_e16xm4\_float16xm4 (C function), 148  
 vmflevf\_mask\_e16xm8\_float16xm8 (C function), 148  
 vmflevf\_mask\_e32xm1\_float32xm1 (C function), 148  
 vmflevf\_mask\_e32xm2\_float32xm2 (C function), 148  
 vmflevf\_mask\_e32xm4\_float32xm4 (C function), 148  
 vmflevf\_mask\_e32xm8\_float32xm8 (C function), 148  
 vmflevf\_mask\_e64xm1\_float64xm1 (C function), 148  
 vmflevf\_mask\_e64xm2\_float64xm2 (C function), 148  
 vmflevf\_mask\_e64xm4\_float64xm4 (C function), 148  
 vmflevf\_mask\_e64xm8\_float64xm8 (C function), 148  
 vmflevv\_e16xm1\_float16xm1 (C function), 148  
 vmflevv\_e16xm2\_float16xm2 (C function), 148  
 vmflevv\_e16xm4\_float16xm4 (C function), 148  
 vmflevv\_e16xm8\_float16xm8 (C function), 148  
 vmflevv\_e32xm1\_float32xm1 (C function), 148  
 vmflevv\_e32xm2\_float32xm2 (C function), 148



- vmfnevv\_mask\_e32xm4\_float32xm4 (C function), 155  
 vmfnevv\_mask\_e32xm8\_float32xm8 (C function), 155  
 vmfnevv\_mask\_e64xm1\_float64xm1 (C function), 155  
 vmfnevv\_mask\_e64xm2\_float64xm2 (C function), 155  
 vmfnevv\_mask\_e64xm4\_float64xm4 (C function), 155  
 vmfnevv\_mask\_e64xm8\_float64xm8 (C function), 155  
 vmfordvf\_e16xm1\_float16xm1 (C function), 155  
 vmfordvf\_e16xm2\_float16xm2 (C function), 155  
 vmfordvf\_e16xm4\_float16xm4 (C function), 155  
 vmfordvf\_e16xm8\_float16xm8 (C function), 155  
 vmfordvf\_e32xm1\_float32xm1 (C function), 155  
 vmfordvf\_e32xm2\_float32xm2 (C function), 155  
 vmfordvf\_e32xm4\_float32xm4 (C function), 155  
 vmfordvf\_e32xm8\_float32xm8 (C function), 155  
 vmfordvf\_e64xm1\_float64xm1 (C function), 155  
 vmfordvf\_e64xm2\_float64xm2 (C function), 156  
 vmfordvf\_e64xm4\_float64xm4 (C function), 156  
 vmfordvf\_e64xm8\_float64xm8 (C function), 156  
 vmfordvf\_mask\_e16xm1\_float16xm1 (C function), 156  
 vmfordvf\_mask\_e16xm2\_float16xm2 (C function), 156  
 vmfordvf\_mask\_e16xm4\_float16xm4 (C function), 156  
 vmfordvf\_mask\_e16xm8\_float16xm8 (C function), 156  
 vmfordvf\_mask\_e32xm1\_float32xm1 (C function), 156  
 vmfordvf\_mask\_e32xm2\_float32xm2 (C function), 156  
 vmfordvf\_mask\_e32xm4\_float32xm4 (C function), 156  
 vmfordvf\_mask\_e32xm8\_float32xm8 (C function), 156  
 vmfordvf\_mask\_e64xm1\_float64xm1 (C function), 156  
 vmfordvf\_mask\_e64xm2\_float64xm2 (C function), 156  
 vmfordvf\_mask\_e64xm4\_float64xm4 (C function), 156  
 vmfordvf\_mask\_e64xm8\_float64xm8 (C function), 156  
 vmfordvv\_e16xm1\_float16xm1 (C function), 157  
 vmfordvv\_e16xm2\_float16xm2 (C function), 157  
 vmfordvv\_e16xm4\_float16xm4 (C function), 157  
 vmfordvv\_e16xm8\_float16xm8 (C function), 157  
 vmfordvv\_e32xm1\_float32xm1 (C function), 157  
 vmfordvv\_e32xm2\_float32xm2 (C function), 157  
 vmfordvv\_e32xm4\_float32xm4 (C function), 157  
 vmfordvv\_e32xm8\_float32xm8 (C function), 157  
 vmfordvv\_e64xm1\_float64xm1 (C function), 157  
 vmfordvv\_e64xm2\_float64xm2 (C function), 157  
 vmfordvv\_e64xm4\_float64xm4 (C function), 157  
 vmfordvv\_e64xm8\_float64xm8 (C function), 157  
 vmfordvv\_mask\_e16xm1\_float16xm1 (C function), 157  
 vmfordvv\_mask\_e16xm2\_float16xm2 (C function), 157  
 vmfordvv\_mask\_e16xm4\_float16xm4 (C function), 157  
 vmfordvv\_mask\_e16xm8\_float16xm8 (C function), 157  
 vmfordvv\_mask\_e32xm1\_float32xm1 (C function), 157  
 vmfordvv\_mask\_e32xm2\_float32xm2 (C function), 157  
 vmfordvv\_mask\_e32xm4\_float32xm4 (C function), 157  
 vmfordvv\_mask\_e32xm8\_float32xm8 (C function), 157  
 vmfordvv\_mask\_e64xm1\_float64xm1 (C function), 157  
 vmfordvv\_mask\_e64xm2\_float64xm2 (C function), 158  
 vmfordvv\_mask\_e64xm4\_float64xm4 (C function), 158  
 vmfordvv\_mask\_e64xm8\_float64xm8 (C function), 158  
 vminuvv\_mask\_uint16xm1 (C function), 212  
 vminuvv\_mask\_uint16xm2 (C function), 212  
 vminuvv\_mask\_uint16xm4 (C function), 212  
 vminuvv\_mask\_uint16xm8 (C function), 212  
 vminuvv\_mask\_uint32xm1 (C function), 212  
 vminuvv\_mask\_uint32xm2 (C function), 212  
 vminuvv\_mask\_uint32xm4 (C function), 212  
 vminuvv\_mask\_uint32xm8 (C function), 212  
 vminuvv\_mask\_uint64xm1 (C function), 212  
 vminuvv\_mask\_uint64xm2 (C function), 212  
 vminuvv\_mask\_uint64xm4 (C function), 212  
 vminuvv\_mask\_uint64xm8 (C function), 212  
 vminuvv\_mask\_uint8xm1 (C function), 212  
 vminuvv\_mask\_uint8xm2 (C function), 212  
 vminuvv\_mask\_uint8xm4 (C function), 212  
 vminuvv\_mask\_uint8xm8 (C function), 212  
 vminuvv\_uint16xm1 (C function), 211  
 vminuvv\_uint16xm2 (C function), 211  
 vminuvv\_uint16xm4 (C function), 211  
 vminuvv\_uint16xm8 (C function), 211  
 vminuvv\_uint32xm1 (C function), 211  
 vminuvv\_uint32xm2 (C function), 211  
 vminuvv\_uint32xm4 (C function), 211  
 vminuvv\_uint32xm8 (C function), 211  
 vminuvv\_uint64xm1 (C function), 211  
 vminuvv\_uint64xm2 (C function), 211  
 vminuvv\_uint64xm4 (C function), 211  
 vminuvv\_uint64xm8 (C function), 211  
 vminuvv\_uint8xm1 (C function), 211  
 vminuvv\_uint8xm2 (C function), 211  
 vminuvv\_uint8xm4 (C function), 211  
 vminuvv\_uint8xm8 (C function), 211  
 vminuvx\_mask\_uint16xm1 (C function), 213  
 vminuvx\_mask\_uint16xm2 (C function), 213  
 vminuvx\_mask\_uint16xm4 (C function), 213  
 vminuvx\_mask\_uint16xm8 (C function), 213  
 vminuvx\_mask\_uint32xm1 (C function), 213  
 vminuvx\_mask\_uint32xm2 (C function), 213  
 vminuvx\_mask\_uint32xm4 (C function), 213  
 vminuvx\_mask\_uint32xm8 (C function), 213  
 vminuvx\_mask\_uint64xm1 (C function), 214  
 vminuvx\_mask\_uint64xm2 (C function), 214  
 vminuvx\_mask\_uint64xm4 (C function), 214  
 vminuvx\_mask\_uint64xm8 (C function), 214  
 vminuvx\_mask\_uint8xm1 (C function), 214  
 vminuvx\_mask\_uint8xm2 (C function), 214  
 vminuvx\_mask\_uint8xm4 (C function), 214  
 vminuvx\_mask\_uint8xm8 (C function), 214  
 vminuvx\_uint16xm1 (C function), 213  
 vminuvx\_uint16xm2 (C function), 213  
 vminuvx\_uint16xm4 (C function), 213  
 vminuvx\_uint16xm8 (C function), 213  
 vminuvx\_uint32xm1 (C function), 213  
 vminuvx\_uint32xm2 (C function), 213



`vminuvx_uint32xm4` (C function), 213  
`vminuvx_uint32xm8` (C function), 213  
`vminuvx_uint64xm1` (C function), 213  
`vminuvx_uint64xm2` (C function), 213  
`vminuvx_uint64xm4` (C function), 213  
`vminuvx_uint64xm8` (C function), 213  
`vminvv_int16xm1` (C function), 208  
`vminvv_int16xm2` (C function), 208  
`vminvv_int16xm4` (C function), 208  
`vminvv_int16xm8` (C function), 208  
`vminvv_int32xm1` (C function), 208  
`vminvv_int32xm2` (C function), 208  
`vminvv_int32xm4` (C function), 208  
`vminvv_int32xm8` (C function), 208  
`vminvv_int64xm1` (C function), 208  
`vminvv_int64xm2` (C function), 208  
`vminvv_int64xm4` (C function), 208  
`vminvv_int64xm8` (C function), 208  
`vminvv_int8xm1` (C function), 208  
`vminvv_int8xm2` (C function), 208  
`vminvv_int8xm4` (C function), 208  
`vminvv_int8xm8` (C function), 208  
`vminvv_mask_int16xm1` (C function), 209  
`vminvv_mask_int16xm2` (C function), 209  
`vminvv_mask_int16xm4` (C function), 209  
`vminvv_mask_int16xm8` (C function), 209  
`vminvv_mask_int32xm1` (C function), 209  
`vminvv_mask_int32xm2` (C function), 209  
`vminvv_mask_int32xm4` (C function), 209  
`vminvv_mask_int32xm8` (C function), 209  
`vminvv_mask_int64xm1` (C function), 209  
`vminvv_mask_int64xm2` (C function), 209  
`vminvv_mask_int64xm4` (C function), 209  
`vminvv_mask_int64xm8` (C function), 209  
`vminvv_mask_int8xm1` (C function), 209  
`vminvv_mask_int8xm2` (C function), 209  
`vminvv_mask_int8xm4` (C function), 209  
`vminvv_mask_int8xm8` (C function), 209  
`vminvx_int16xm1` (C function), 209  
`vminvx_int16xm2` (C function), 210  
`vminvx_int16xm4` (C function), 210  
`vminvx_int16xm8` (C function), 210  
`vminvx_int32xm1` (C function), 210  
`vminvx_int32xm2` (C function), 210  
`vminvx_int32xm4` (C function), 210  
`vminvx_int32xm8` (C function), 210  
`vminvx_int64xm1` (C function), 210  
`vminvx_int64xm2` (C function), 210  
`vminvx_int64xm4` (C function), 210  
`vminvx_int64xm8` (C function), 210  
`vminvx_int8xm1` (C function), 210  
`vminvx_int8xm2` (C function), 210  
`vminvx_int8xm4` (C function), 210  
`vminvx_int8xm8` (C function), 210  
`vminvx_mask_int16xm1` (C function), 210  
`vminvx_mask_int16xm2` (C function), 210  
`vminvx_mask_int16xm4` (C function), 210  
`vminvx_mask_int16xm8` (C function), 210  
`vminvx_mask_int32xm1` (C function), 210  
`vminvx_mask_int32xm2` (C function), 210  
`vminvx_mask_int32xm4` (C function), 210  
`vminvx_mask_int32xm8` (C function), 210  
`vminvx_mask_int64xm1` (C function), 211  
`vminvx_mask_int64xm2` (C function), 211  
`vminvx_mask_int64xm4` (C function), 211  
`vminvx_mask_int64xm8` (C function), 211  
`vminvx_mask_int8xm1` (C function), 211  
`vminvx_mask_int8xm2` (C function), 211  
`vminvx_mask_int8xm4` (C function), 211  
`vminvx_mask_int8xm8` (C function), 211  
`vmnandmm_e16xm1` (C function), 803  
`vmnandmm_e16xm2` (C function), 803  
`vmnandmm_e16xm4` (C function), 803  
`vmnandmm_e16xm8` (C function), 803  
`vmnandmm_e32xm1` (C function), 804  
`vmnandmm_e32xm2` (C function), 804  
`vmnandmm_e32xm4` (C function), 804  
`vmnandmm_e32xm8` (C function), 804  
`vmnandmm_e64xm1` (C function), 804  
`vmnandmm_e64xm2` (C function), 804  
`vmnandmm_e64xm4` (C function), 804  
`vmnandmm_e64xm8` (C function), 804  
`vmnandmm_e8xm1` (C function), 804  
`vmnandmm_e8xm2` (C function), 804  
`vmnandmm_e8xm4` (C function), 804  
`vmnandmm_e8xm8` (C function), 804  
`vmnormmm_e16xm1` (C function), 804  
`vmnormmm_e16xm2` (C function), 804  
`vmnormmm_e16xm4` (C function), 804  
`vmnormmm_e16xm8` (C function), 804  
`vmnormmm_e32xm1` (C function), 804  
`vmnormmm_e32xm2` (C function), 804  
`vmnormmm_e32xm4` (C function), 804  
`vmnormmm_e32xm8` (C function), 804  
`vmnormmm_e64xm1` (C function), 804  
`vmnormmm_e64xm2` (C function), 804  
`vmnormmm_e64xm4` (C function), 804  
`vmnormmm_e64xm8` (C function), 804  
`vmnormmm_e8xm1` (C function), 804  
`vmnormmm_e8xm2` (C function), 804  
`vmnormmm_e8xm4` (C function), 804  
`vmnormmm_e8xm8` (C function), 804  
`vmnotm_e16xm1` (C function), 805  
`vmnotm_e16xm2` (C function), 805

vmnotm\_e16xm4 (C function), 805  
 vmnotm\_e16xm8 (C function), 805  
 vmnotm\_e32xm1 (C function), 805  
 vmnotm\_e32xm2 (C function), 805  
 vmnotm\_e32xm4 (C function), 805  
 vmnotm\_e32xm8 (C function), 805  
 vmnotm\_e64xm1 (C function), 805  
 vmnotm\_e64xm2 (C function), 805  
 vmnotm\_e64xm4 (C function), 805  
 vmnotm\_e64xm8 (C function), 805  
 vmnotm\_e8xm1 (C function), 805  
 vmnotm\_e8xm2 (C function), 805  
 vmnotm\_e8xm4 (C function), 805  
 vmnotm\_e8xm8 (C function), 805  
 vmormm\_e16xm1 (C function), 805  
 vmormm\_e16xm2 (C function), 805  
 vmormm\_e16xm4 (C function), 805  
 vmormm\_e16xm8 (C function), 805  
 vmormm\_e32xm1 (C function), 806  
 vmormm\_e32xm2 (C function), 806  
 vmormm\_e32xm4 (C function), 806  
 vmormm\_e32xm8 (C function), 806  
 vmormm\_e64xm1 (C function), 806  
 vmormm\_e64xm2 (C function), 806  
 vmormm\_e64xm4 (C function), 806  
 vmormm\_e64xm8 (C function), 806  
 vmormm\_e8xm1 (C function), 806  
 vmormm\_e8xm2 (C function), 806  
 vmormm\_e8xm4 (C function), 806  
 vmormm\_e8xm8 (C function), 806  
 vmornotmm\_e16xm1 (C function), 806  
 vmornotmm\_e16xm2 (C function), 806  
 vmornotmm\_e16xm4 (C function), 806  
 vmornotmm\_e16xm8 (C function), 806  
 vmornotmm\_e32xm1 (C function), 806  
 vmornotmm\_e32xm2 (C function), 806  
 vmornotmm\_e32xm4 (C function), 806  
 vmornotmm\_e32xm8 (C function), 806  
 vmornotmm\_e64xm1 (C function), 806  
 vmornotmm\_e64xm2 (C function), 806  
 vmornotmm\_e64xm4 (C function), 806  
 vmornotmm\_e64xm8 (C function), 806  
 vmornotmm\_e8xm1 (C function), 806  
 vmornotmm\_e8xm2 (C function), 806  
 vmornotmm\_e8xm4 (C function), 806  
 vmornotmm\_e8xm8 (C function), 806  
 vmpopcm\_e16xm1 (C function), 807  
 vmpopcm\_e16xm2 (C function), 807  
 vmpopcm\_e16xm4 (C function), 807  
 vmpopcm\_e16xm8 (C function), 807  
 vmpopcm\_e32xm1 (C function), 807  
 vmpopcm\_e32xm2 (C function), 807  
 vmpopcm\_e32xm4 (C function), 807  
 vmpopcm\_e32xm8 (C function), 807  
 vmpopcm\_e64xm1 (C function), 807  
 vmpopcm\_e64xm2 (C function), 807  
 vmpopcm\_e64xm4 (C function), 807  
 vmpopcm\_e64xm8 (C function), 807  
 vmpopcm\_e8xm1 (C function), 807  
 vmpopcm\_e8xm2 (C function), 807  
 vmpopcm\_e8xm4 (C function), 807  
 vmpopcm\_e8xm8 (C function), 807  
 vmpopcm\_mask\_e16xm1 (C function), 807  
 vmpopcm\_mask\_e16xm2 (C function), 807  
 vmpopcm\_mask\_e16xm4 (C function), 807  
 vmpopcm\_mask\_e16xm8 (C function), 807  
 vmpopcm\_mask\_e32xm1 (C function), 807  
 vmpopcm\_mask\_e32xm2 (C function), 807  
 vmpopcm\_mask\_e32xm4 (C function), 807  
 vmpopcm\_mask\_e32xm8 (C function), 808  
 vmpopcm\_mask\_e64xm1 (C function), 808  
 vmpopcm\_mask\_e64xm2 (C function), 808  
 vmpopcm\_mask\_e64xm4 (C function), 808  
 vmpopcm\_mask\_e64xm8 (C function), 808  
 vmpopcm\_mask\_e8xm1 (C function), 808  
 vmpopcm\_mask\_e8xm2 (C function), 808  
 vmpopcm\_mask\_e8xm4 (C function), 808  
 vmpopcm\_mask\_e8xm8 (C function), 808  
 vmsbcvmm\_mask\_e16xm1\_int16xm1 (C function), 214  
 vmsbcvmm\_mask\_e16xm1\_uint16xm1 (C function), 214  
 vmsbcvmm\_mask\_e16xm2\_int16xm2 (C function), 214  
 vmsbcvmm\_mask\_e16xm2\_uint16xm2 (C function), 214  
 vmsbcvmm\_mask\_e16xm4\_int16xm4 (C function), 214  
 vmsbcvmm\_mask\_e16xm4\_uint16xm4 (C function), 214  
 vmsbcvmm\_mask\_e16xm8\_int16xm8 (C function), 214  
 vmsbcvmm\_mask\_e16xm8\_uint16xm8 (C function), 214  
 vmsbcvmm\_mask\_e32xm1\_int32xm1 (C function), 215  
 vmsbcvmm\_mask\_e32xm1\_uint32xm1 (C function), 215  
 vmsbcvmm\_mask\_e32xm2\_int32xm2 (C function), 215  
 vmsbcvmm\_mask\_e32xm2\_uint32xm2 (C function), 215  
 vmsbcvmm\_mask\_e32xm4\_int32xm4 (C function), 215  
 vmsbcvmm\_mask\_e32xm4\_uint32xm4 (C function), 215  
 vmsbcvmm\_mask\_e32xm8\_int32xm8 (C function), 215  
 vmsbcvmm\_mask\_e32xm8\_uint32xm8 (C function), 215  
 vmsbcvmm\_mask\_e64xm1\_int64xm1 (C function), 215  
 vmsbcvmm\_mask\_e64xm1\_uint64xm1 (C function), 215  
 vmsbcvmm\_mask\_e64xm2\_int64xm2 (C function), 215  
 vmsbcvmm\_mask\_e64xm2\_uint64xm2 (C function), 215  
 vmsbcvmm\_mask\_e64xm4\_int64xm4 (C function), 215  
 vmsbcvmm\_mask\_e64xm4\_uint64xm4 (C function), 215  
 vmsbcvmm\_mask\_e64xm8\_int64xm8 (C function), 215  
 vmsbcvmm\_mask\_e64xm8\_uint64xm8 (C function), 215  
 vmsbcvmm\_mask\_e8xm1\_int8xm1 (C function), 215  
 vmsbcvmm\_mask\_e8xm1\_uint8xm1 (C function), 215  
 vmsbcvmm\_mask\_e8xm2\_int8xm2 (C function), 215  
 vmsbcvmm\_mask\_e8xm2\_uint8xm2 (C function), 215  
 vmsbcvmm\_mask\_e8xm4\_int8xm4 (C function), 215  
 vmsbcvmm\_mask\_e8xm4\_uint8xm4 (C function), 215



vmsbcbvm\_mask\_e8xm8\_int8xm8 (C function), 215  
vmsbcbvm\_mask\_e8xm8\_uint8xm8 (C function), 216  
vmsbcbvm\_mask\_e16xm1\_int16xm1 (C function), 216  
vmsbcbvm\_mask\_e16xm1\_uint16xm1 (C function), 216  
vmsbcbvm\_mask\_e16xm2\_int16xm2 (C function), 216  
vmsbcbvm\_mask\_e16xm2\_uint16xm2 (C function), 216  
vmsbcbvm\_mask\_e16xm4\_int16xm4 (C function), 216  
vmsbcbvm\_mask\_e16xm4\_uint16xm4 (C function), 216  
vmsbcbvm\_mask\_e16xm8\_int16xm8 (C function), 216  
vmsbcbvm\_mask\_e16xm8\_uint16xm8 (C function), 216  
vmsbcbvm\_mask\_e32xm1\_int32xm1 (C function), 216  
vmsbcbvm\_mask\_e32xm1\_uint32xm1 (C function), 216  
vmsbcbvm\_mask\_e32xm2\_int32xm2 (C function), 216  
vmsbcbvm\_mask\_e32xm2\_uint32xm2 (C function), 216  
vmsbcbvm\_mask\_e32xm4\_int32xm4 (C function), 216  
vmsbcbvm\_mask\_e32xm4\_uint32xm4 (C function), 216  
vmsbcbvm\_mask\_e32xm8\_int32xm8 (C function), 216  
vmsbcbvm\_mask\_e32xm8\_uint32xm8 (C function), 216  
vmsbcbvm\_mask\_e64xm1\_int64xm1 (C function), 216  
vmsbcbvm\_mask\_e64xm1\_uint64xm1 (C function), 217  
vmsbcbvm\_mask\_e64xm2\_int64xm2 (C function), 217  
vmsbcbvm\_mask\_e64xm2\_uint64xm2 (C function), 217  
vmsbcbvm\_mask\_e64xm4\_int64xm4 (C function), 217  
vmsbcbvm\_mask\_e64xm4\_uint64xm4 (C function), 217  
vmsbcbvm\_mask\_e64xm8\_int64xm8 (C function), 217  
vmsbcbvm\_mask\_e64xm8\_uint64xm8 (C function), 217  
vmsbcbvm\_mask\_e8xm1\_int8xm1 (C function), 217  
vmsbcbvm\_mask\_e8xm1\_uint8xm1 (C function), 217  
vmsbcbvm\_mask\_e8xm2\_int8xm2 (C function), 217  
vmsbcbvm\_mask\_e8xm2\_uint8xm2 (C function), 217  
vmsbcbvm\_mask\_e8xm4\_int8xm4 (C function), 217  
vmsbcbvm\_mask\_e8xm4\_uint8xm4 (C function), 217  
vmsbcbvm\_mask\_e8xm8\_int8xm8 (C function), 217  
vmsbcbvm\_mask\_e8xm8\_uint8xm8 (C function), 217  
vmsbfm\_e16xm1 (C function), 808  
vmsbfm\_e16xm2 (C function), 808  
vmsbfm\_e16xm4 (C function), 808  
vmsbfm\_e16xm8 (C function), 808  
vmsbfm\_e32xm1 (C function), 808  
vmsbfm\_e32xm2 (C function), 808  
vmsbfm\_e32xm4 (C function), 808  
vmsbfm\_e32xm8 (C function), 808  
vmsbfm\_e64xm1 (C function), 808  
vmsbfm\_e64xm2 (C function), 808  
vmsbfm\_e64xm4 (C function), 808  
vmsbfm\_e64xm8 (C function), 808  
vmsbfm\_e8xm1 (C function), 808  
vmsbfm\_e8xm2 (C function), 808  
vmsbfm\_e8xm4 (C function), 808  
vmsbfm\_e8xm8 (C function), 808  
vmsbfm\_mask\_e16xm1 (C function), 809  
vmsbfm\_mask\_e16xm2 (C function), 809  
vmsbfm\_mask\_e16xm4 (C function), 809  
vmsbfm\_mask\_e16xm8 (C function), 809  
vmsbfm\_mask\_e32xm1 (C function), 809  
vmsbfm\_mask\_e32xm2 (C function), 809  
vmsbfm\_mask\_e32xm4 (C function), 809  
vmsbfm\_mask\_e32xm8 (C function), 809  
vmsbfm\_mask\_e64xm1 (C function), 809  
vmsbfm\_mask\_e64xm2 (C function), 809  
vmsbfm\_mask\_e64xm4 (C function), 809  
vmsbfm\_mask\_e64xm8 (C function), 809  
vmsbfm\_mask\_e8xm1 (C function), 809  
vmsbfm\_mask\_e8xm2 (C function), 809  
vmsbfm\_mask\_e8xm4 (C function), 809  
vmsbfm\_mask\_e8xm8 (C function), 809  
vmseqvi\_e16xm1\_int16xm1 (C function), 370  
vmseqvi\_e16xm1\_uint16xm1 (C function), 370  
vmseqvi\_e16xm2\_int16xm2 (C function), 370  
vmseqvi\_e16xm2\_uint16xm2 (C function), 370  
vmseqvi\_e16xm4\_int16xm4 (C function), 370  
vmseqvi\_e16xm4\_uint16xm4 (C function), 370  
vmseqvi\_e16xm8\_int16xm8 (C function), 370  
vmseqvi\_e16xm8\_uint16xm8 (C function), 370  
vmseqvi\_e32xm1\_int32xm1 (C function), 370  
vmseqvi\_e32xm1\_uint32xm1 (C function), 370  
vmseqvi\_e32xm2\_int32xm2 (C function), 370  
vmseqvi\_e32xm2\_uint32xm2 (C function), 370  
vmseqvi\_e32xm4\_int32xm4 (C function), 370  
vmseqvi\_e32xm4\_uint32xm4 (C function), 370  
vmseqvi\_e32xm8\_int32xm8 (C function), 370  
vmseqvi\_e32xm8\_uint32xm8 (C function), 370  
vmseqvi\_e64xm1\_int64xm1 (C function), 370  
vmseqvi\_e64xm1\_uint64xm1 (C function), 370  
vmseqvi\_e64xm2\_int64xm2 (C function), 370  
vmseqvi\_e64xm2\_uint64xm2 (C function), 370  
vmseqvi\_e64xm4\_int64xm4 (C function), 370  
vmseqvi\_e64xm4\_uint64xm4 (C function), 370  
vmseqvi\_e64xm8\_int64xm8 (C function), 370  
vmseqvi\_e64xm8\_uint64xm8 (C function), 370  
vmseqvi\_e8xm1\_int8xm1 (C function), 370  
vmseqvi\_e8xm1\_uint8xm1 (C function), 370  
vmseqvi\_e8xm2\_int8xm2 (C function), 370  
vmseqvi\_e8xm2\_uint8xm2 (C function), 370  
vmseqvi\_e8xm4\_int8xm4 (C function), 370  
vmseqvi\_e8xm4\_uint8xm4 (C function), 370  
vmseqvi\_e8xm8\_int8xm8 (C function), 370  
vmseqvi\_e8xm8\_uint8xm8 (C function), 371  
vmseqvi\_mask\_e16xm1\_int16xm1 (C function), 371  
vmseqvi\_mask\_e16xm1\_uint16xm1 (C function), 371  
vmseqvi\_mask\_e16xm2\_int16xm2 (C function), 371  
vmseqvi\_mask\_e16xm2\_uint16xm2 (C function), 371  
vmseqvi\_mask\_e16xm4\_int16xm4 (C function), 371  
vmseqvi\_mask\_e16xm4\_uint16xm4 (C function), 371  
vmseqvi\_mask\_e16xm8\_int16xm8 (C function), 371  
vmseqvi\_mask\_e16xm8\_uint16xm8 (C function), 371  
vmseqvi\_mask\_e32xm1\_int32xm1 (C function), 371  
vmseqvi\_mask\_e32xm1\_uint32xm1 (C function), 371



vmseqvx\_e64xm8\_int64xm8 (C function), 376  
 vmseqvx\_e64xm8\_uint64xm8 (C function), 376  
 vmseqvx\_e8xm1\_int8xm1 (C function), 376  
 vmseqvx\_e8xm1\_uint8xm1 (C function), 376  
 vmseqvx\_e8xm2\_int8xm2 (C function), 376  
 vmseqvx\_e8xm2\_uint8xm2 (C function), 376  
 vmseqvx\_e8xm4\_int8xm4 (C function), 376  
 vmseqvx\_e8xm4\_uint8xm4 (C function), 376  
 vmseqvx\_e8xm8\_int8xm8 (C function), 376  
 vmseqvx\_e8xm8\_uint8xm8 (C function), 376  
 vmseqvx\_mask\_e16xm1\_int16xm1 (C function), 376  
 vmseqvx\_mask\_e16xm1\_uint16xm1 (C function), 376  
 vmseqvx\_mask\_e16xm2\_int16xm2 (C function), 376  
 vmseqvx\_mask\_e16xm2\_uint16xm2 (C function), 376  
 vmseqvx\_mask\_e16xm4\_int16xm4 (C function), 376  
 vmseqvx\_mask\_e16xm4\_uint16xm4 (C function), 376  
 vmseqvx\_mask\_e16xm8\_int16xm8 (C function), 376  
 vmseqvx\_mask\_e16xm8\_uint16xm8 (C function), 376  
 vmseqvx\_mask\_e32xm1\_int32xm1 (C function), 376  
 vmseqvx\_mask\_e32xm1\_uint32xm1 (C function), 376  
 vmseqvx\_mask\_e32xm2\_int32xm2 (C function), 377  
 vmseqvx\_mask\_e32xm2\_uint32xm2 (C function), 377  
 vmseqvx\_mask\_e32xm4\_int32xm4 (C function), 377  
 vmseqvx\_mask\_e32xm4\_uint32xm4 (C function), 377  
 vmseqvx\_mask\_e32xm8\_int32xm8 (C function), 377  
 vmseqvx\_mask\_e32xm8\_uint32xm8 (C function), 377  
 vmseqvx\_mask\_e64xm1\_int64xm1 (C function), 377  
 vmseqvx\_mask\_e64xm1\_uint64xm1 (C function), 377  
 vmseqvx\_mask\_e64xm2\_int64xm2 (C function), 377  
 vmseqvx\_mask\_e64xm2\_uint64xm2 (C function), 377  
 vmseqvx\_mask\_e64xm4\_int64xm4 (C function), 377  
 vmseqvx\_mask\_e64xm4\_uint64xm4 (C function), 377  
 vmseqvx\_mask\_e64xm8\_int64xm8 (C function), 377  
 vmseqvx\_mask\_e64xm8\_uint64xm8 (C function), 377  
 vmseqvx\_mask\_e8xm1\_int8xm1 (C function), 377  
 vmseqvx\_mask\_e8xm1\_uint8xm1 (C function), 377  
 vmseqvx\_mask\_e8xm2\_int8xm2 (C function), 377  
 vmseqvx\_mask\_e8xm2\_uint8xm2 (C function), 377  
 vmseqvx\_mask\_e8xm4\_int8xm4 (C function), 377  
 vmseqvx\_mask\_e8xm4\_uint8xm4 (C function), 377  
 vmseqvx\_mask\_e8xm8\_int8xm8 (C function), 377  
 vmseqvx\_mask\_e8xm8\_uint8xm8 (C function), 377  
 vmsetm\_e16xm1 (C function), 809  
 vmsetm\_e16xm2 (C function), 809  
 vmsetm\_e16xm4 (C function), 809  
 vmsetm\_e16xm8 (C function), 809  
 vmsetm\_e32xm1 (C function), 810  
 vmsetm\_e32xm2 (C function), 810  
 vmsetm\_e32xm4 (C function), 810  
 vmsetm\_e32xm8 (C function), 810  
 vmsetm\_e64xm1 (C function), 810  
 vmsetm\_e64xm2 (C function), 810  
 vmsetm\_e64xm4 (C function), 810  
 vmsetm\_e64xm8 (C function), 810  
 vmsetm\_e8xm1 (C function), 810  
 vmsetm\_e8xm2 (C function), 810  
 vmsetm\_e8xm4 (C function), 810  
 vmsetm\_e8xm8 (C function), 810  
 vmsgeuvx\_e16xm1\_uint16xm1 (C function), 379  
 vmsgeuvx\_e16xm2\_uint16xm2 (C function), 379  
 vmsgeuvx\_e16xm4\_uint16xm4 (C function), 379  
 vmsgeuvx\_e16xm8\_uint16xm8 (C function), 379  
 vmsgeuvx\_e32xm1\_uint32xm1 (C function), 379  
 vmsgeuvx\_e32xm2\_uint32xm2 (C function), 379  
 vmsgeuvx\_e32xm4\_uint32xm4 (C function), 379  
 vmsgeuvx\_e32xm8\_uint32xm8 (C function), 380  
 vmsgeuvx\_e64xm1\_uint64xm1 (C function), 380  
 vmsgeuvx\_e64xm2\_uint64xm2 (C function), 380  
 vmsgeuvx\_e64xm4\_uint64xm4 (C function), 380  
 vmsgeuvx\_e64xm8\_uint64xm8 (C function), 380  
 vmsgeuvx\_e8xm1\_uint8xm1 (C function), 380  
 vmsgeuvx\_e8xm2\_uint8xm2 (C function), 380  
 vmsgeuvx\_e8xm4\_uint8xm4 (C function), 380  
 vmsgeuvx\_e8xm8\_uint8xm8 (C function), 380  
 vmsgeuvx\_mask\_e16xm1\_int16xm1 (C function), 380  
 vmsgeuvx\_mask\_e16xm2\_int16xm2 (C function), 380  
 vmsgeuvx\_mask\_e16xm4\_int16xm4 (C function), 380  
 vmsgeuvx\_mask\_e16xm8\_int16xm8 (C function), 380  
 vmsgeuvx\_mask\_e32xm1\_int32xm1 (C function), 380  
 vmsgeuvx\_mask\_e32xm2\_int32xm2 (C function), 380  
 vmsgeuvx\_mask\_e32xm4\_int32xm4 (C function), 380  
 vmsgeuvx\_mask\_e32xm8\_int32xm8 (C function), 380  
 vmsgeuvx\_mask\_e64xm1\_int64xm1 (C function), 380  
 vmsgeuvx\_mask\_e64xm2\_int64xm2 (C function), 380  
 vmsgeuvx\_mask\_e64xm4\_int64xm4 (C function), 380  
 vmsgeuvx\_mask\_e64xm8\_int64xm8 (C function), 380  
 vmsgeuvx\_mask\_e8xm1\_int8xm1 (C function), 380  
 vmsgeuvx\_mask\_e8xm2\_int8xm2 (C function), 380  
 vmsgeuvx\_mask\_e8xm4\_int8xm4 (C function), 381  
 vmsgeuvx\_mask\_e8xm8\_int8xm8 (C function), 381  
 vmsgevx\_e16xm1\_int16xm1 (C function), 378  
 vmsgevx\_e16xm2\_int16xm2 (C function), 378  
 vmsgevx\_e16xm4\_int16xm4 (C function), 378  
 vmsgevx\_e16xm8\_int16xm8 (C function), 378  
 vmsgevx\_e32xm1\_int32xm1 (C function), 378  
 vmsgevx\_e32xm2\_int32xm2 (C function), 378  
 vmsgevx\_e32xm4\_int32xm4 (C function), 378  
 vmsgevx\_e32xm8\_int32xm8 (C function), 378  
 vmsgevx\_e64xm1\_int64xm1 (C function), 378  
 vmsgevx\_e64xm2\_int64xm2 (C function), 378  
 vmsgevx\_e64xm4\_int64xm4 (C function), 378  
 vmsgevx\_e64xm8\_int64xm8 (C function), 378  
 vmsgevx\_e8xm1\_int8xm1 (C function), 378  
 vmsgevx\_e8xm2\_int8xm2 (C function), 378  
 vmsgevx\_e8xm4\_int8xm4 (C function), 378  
 vmsgevx\_e8xm8\_int8xm8 (C function), 378  
 vmsgevx\_mask\_e16xm1\_int16xm1 (C function), 378  
 vmsgevx\_mask\_e16xm2\_int16xm2 (C function), 378

vmsgevx\_mask\_e16xm4\_int16xm4 (C function), 378  
 vmsgevx\_mask\_e16xm8\_int16xm8 (C function), 378  
 vmsgevx\_mask\_e32xm1\_int32xm1 (C function), 379  
 vmsgevx\_mask\_e32xm2\_int32xm2 (C function), 379  
 vmsgevx\_mask\_e32xm4\_int32xm4 (C function), 379  
 vmsgevx\_mask\_e32xm8\_int32xm8 (C function), 379  
 vmsgevx\_mask\_e64xm1\_int64xm1 (C function), 379  
 vmsgevx\_mask\_e64xm2\_int64xm2 (C function), 379  
 vmsgevx\_mask\_e64xm4\_int64xm4 (C function), 379  
 vmsgevx\_mask\_e64xm8\_int64xm8 (C function), 379  
 vmsgevx\_mask\_e8xm1\_int8xm1 (C function), 379  
 vmsgevx\_mask\_e8xm2\_int8xm2 (C function), 379  
 vmsgevx\_mask\_e8xm4\_int8xm4 (C function), 379  
 vmsgevx\_mask\_e8xm8\_int8xm8 (C function), 379  
 vmsgtuvi\_e16xm1\_uint16xm1 (C function), 384  
 vmsgtuvi\_e16xm2\_uint16xm2 (C function), 384  
 vmsgtuvi\_e16xm4\_uint16xm4 (C function), 384  
 vmsgtuvi\_e16xm8\_uint16xm8 (C function), 384  
 vmsgtuvi\_e32xm1\_uint32xm1 (C function), 384  
 vmsgtuvi\_e32xm2\_uint32xm2 (C function), 384  
 vmsgtuvi\_e32xm4\_uint32xm4 (C function), 384  
 vmsgtuvi\_e32xm8\_uint32xm8 (C function), 384  
 vmsgtuvi\_e64xm1\_uint64xm1 (C function), 384  
 vmsgtuvi\_e64xm2\_uint64xm2 (C function), 384  
 vmsgtuvi\_e64xm4\_uint64xm4 (C function), 384  
 vmsgtuvi\_e64xm8\_uint64xm8 (C function), 384  
 vmsgtuvi\_e8xm1\_uint8xm1 (C function), 384  
 vmsgtuvi\_e8xm2\_uint8xm2 (C function), 384  
 vmsgtuvi\_e8xm4\_uint8xm4 (C function), 384  
 vmsgtuvi\_e8xm8\_uint8xm8 (C function), 384  
 vmsgtuvi\_mask\_e16xm1\_uint16xm1 (C function), 385  
 vmsgtuvi\_mask\_e16xm2\_uint16xm2 (C function), 385  
 vmsgtuvi\_mask\_e16xm4\_uint16xm4 (C function), 385  
 vmsgtuvi\_mask\_e16xm8\_uint16xm8 (C function), 385  
 vmsgtuvi\_mask\_e32xm1\_uint32xm1 (C function), 385  
 vmsgtuvi\_mask\_e32xm2\_uint32xm2 (C function), 385  
 vmsgtuvi\_mask\_e32xm4\_uint32xm4 (C function), 385  
 vmsgtuvi\_mask\_e32xm8\_uint32xm8 (C function), 385  
 vmsgtuvi\_mask\_e64xm1\_uint64xm1 (C function), 385  
 vmsgtuvi\_mask\_e64xm2\_uint64xm2 (C function), 385  
 vmsgtuvi\_mask\_e64xm4\_uint64xm4 (C function), 385  
 vmsgtuvi\_mask\_e64xm8\_uint64xm8 (C function), 385  
 vmsgtuvi\_mask\_e8xm1\_uint8xm1 (C function), 385  
 vmsgtuvi\_mask\_e8xm2\_uint8xm2 (C function), 385  
 vmsgtuvi\_mask\_e8xm4\_uint8xm4 (C function), 385  
 vmsgtuvi\_mask\_e8xm8\_uint8xm8 (C function), 385  
 vmsgtuvx\_e16xm1\_uint16xm1 (C function), 386  
 vmsgtuvx\_e16xm2\_uint16xm2 (C function), 386  
 vmsgtuvx\_e16xm4\_uint16xm4 (C function), 386  
 vmsgtuvx\_e16xm8\_uint16xm8 (C function), 386  
 vmsgtuvx\_e32xm1\_uint32xm1 (C function), 386  
 vmsgtuvx\_e32xm2\_uint32xm2 (C function), 386  
 vmsgtuvx\_e32xm4\_uint32xm4 (C function), 386  
 vmsgtuvx\_e32xm8\_uint32xm8 (C function), 386  
 vmsgtuvx\_e64xm1\_uint64xm1 (C function), 386  
 vmsgtuvx\_e64xm2\_uint64xm2 (C function), 386  
 vmsgtuvx\_e64xm4\_uint64xm4 (C function), 386  
 vmsgtuvx\_e64xm8\_uint64xm8 (C function), 386  
 vmsgtuvx\_e8xm1\_uint8xm1 (C function), 386  
 vmsgtuvx\_e8xm2\_uint8xm2 (C function), 386  
 vmsgtuvx\_e8xm4\_uint8xm4 (C function), 386  
 vmsgtuvx\_e8xm8\_uint8xm8 (C function), 386  
 vmsgtvi\_e16xm1\_int16xm1 (C function), 381  
 vmsgtvi\_e16xm2\_int16xm2 (C function), 381  
 vmsgtvi\_e16xm4\_int16xm4 (C function), 381  
 vmsgtvi\_e16xm8\_int16xm8 (C function), 381  
 vmsgtvi\_e32xm1\_int32xm1 (C function), 381  
 vmsgtvi\_e32xm2\_int32xm2 (C function), 381  
 vmsgtvi\_e32xm4\_int32xm4 (C function), 381  
 vmsgtvi\_e32xm8\_int32xm8 (C function), 381  
 vmsgtvi\_e64xm1\_int64xm1 (C function), 381  
 vmsgtvi\_e64xm2\_int64xm2 (C function), 381  
 vmsgtvi\_e64xm4\_int64xm4 (C function), 381  
 vmsgtvi\_e64xm8\_int64xm8 (C function), 381  
 vmsgtvi\_e8xm1\_int8xm1 (C function), 381  
 vmsgtvi\_e8xm2\_int8xm2 (C function), 381  
 vmsgtvi\_e8xm4\_int8xm4 (C function), 381  
 vmsgtvi\_e8xm8\_int8xm8 (C function), 381  
 vmsgtvi\_mask\_e16xm1\_int16xm1 (C function), 381  
 vmsgtvi\_mask\_e16xm2\_int16xm2 (C function), 382  
 vmsgtvi\_mask\_e16xm4\_int16xm4 (C function), 382  
 vmsgtvi\_mask\_e16xm8\_int16xm8 (C function), 382  
 vmsgtvi\_mask\_e32xm1\_int32xm1 (C function), 382  
 vmsgtvi\_mask\_e32xm2\_int32xm2 (C function), 382  
 vmsgtvi\_mask\_e32xm4\_int32xm4 (C function), 382  
 vmsgtvi\_mask\_e32xm8\_int32xm8 (C function), 382  
 vmsgtvi\_mask\_e64xm1\_int64xm1 (C function), 382  
 vmsgtvi\_mask\_e64xm2\_int64xm2 (C function), 382  
 vmsgtvi\_mask\_e64xm4\_int64xm4 (C function), 382  
 vmsgtvi\_mask\_e64xm8\_int64xm8 (C function), 382  
 vmsgtvi\_mask\_e8xm1\_int8xm1 (C function), 382  
 vmsgtvi\_mask\_e8xm2\_int8xm2 (C function), 382



vmsgtvi\_mask\_e8xm4\_int8xm4 (C function), 382  
vmsgtvi\_mask\_e8xm8\_int8xm8 (C function), 382  
vmsgtvx\_e16xm1\_int16xm1 (C function), 382  
vmsgtvx\_e16xm2\_int16xm2 (C function), 382  
vmsgtvx\_e16xm4\_int16xm4 (C function), 383  
vmsgtvx\_e16xm8\_int16xm8 (C function), 383  
vmsgtvx\_e32xm1\_int32xm1 (C function), 383  
vmsgtvx\_e32xm2\_int32xm2 (C function), 383  
vmsgtvx\_e32xm4\_int32xm4 (C function), 383  
vmsgtvx\_e32xm8\_int32xm8 (C function), 383  
vmsgtvx\_e64xm1\_int64xm1 (C function), 383  
vmsgtvx\_e64xm2\_int64xm2 (C function), 383  
vmsgtvx\_e64xm4\_int64xm4 (C function), 383  
vmsgtvx\_e64xm8\_int64xm8 (C function), 383  
vmsgtvx\_e8xm1\_int8xm1 (C function), 383  
vmsgtvx\_e8xm2\_int8xm2 (C function), 383  
vmsgtvx\_e8xm4\_int8xm4 (C function), 383  
vmsgtvx\_e8xm8\_int8xm8 (C function), 383  
vmsgtvx\_mask\_e16xm1\_int16xm1 (C function), 383  
vmsgtvx\_mask\_e16xm2\_int16xm2 (C function), 383  
vmsgtvx\_mask\_e16xm4\_int16xm4 (C function), 383  
vmsgtvx\_mask\_e16xm8\_int16xm8 (C function), 383  
vmsgtvx\_mask\_e32xm1\_int32xm1 (C function), 383  
vmsgtvx\_mask\_e32xm2\_int32xm2 (C function), 383  
vmsgtvx\_mask\_e32xm4\_int32xm4 (C function), 383  
vmsgtvx\_mask\_e32xm8\_int32xm8 (C function), 383  
vmsgtvx\_mask\_e64xm1\_int64xm1 (C function), 383  
vmsgtvx\_mask\_e64xm2\_int64xm2 (C function), 383  
vmsgtvx\_mask\_e64xm4\_int64xm4 (C function), 383  
vmsgtvx\_mask\_e64xm8\_int64xm8 (C function), 384  
vmsgtvx\_mask\_e8xm1\_int8xm1 (C function), 384  
vmsgtvx\_mask\_e8xm2\_int8xm2 (C function), 384  
vmsgtvx\_mask\_e8xm4\_int8xm4 (C function), 384  
vmsgtvx\_mask\_e8xm8\_int8xm8 (C function), 384  
vmsifm\_e16xm1 (C function), 810  
vmsifm\_e16xm2 (C function), 810  
vmsifm\_e16xm4 (C function), 810  
vmsifm\_e16xm8 (C function), 810  
vmsifm\_e32xm1 (C function), 810  
vmsifm\_e32xm2 (C function), 810  
vmsifm\_e32xm4 (C function), 810  
vmsifm\_e32xm8 (C function), 810  
vmsifm\_e64xm1 (C function), 810  
vmsifm\_e64xm2 (C function), 810  
vmsifm\_e64xm4 (C function), 810  
vmsifm\_e64xm8 (C function), 810  
vmsifm\_e8xm1 (C function), 810  
vmsifm\_e8xm2 (C function), 810  
vmsifm\_e8xm4 (C function), 810  
vmsifm\_e8xm8 (C function), 810  
vmsifm\_mask\_e16xm1 (C function), 811  
vmsifm\_mask\_e16xm2 (C function), 811  
vmsifm\_mask\_e16xm4 (C function), 811  
vmsifm\_mask\_e16xm8 (C function), 811  
vmsifm\_mask\_e32xm1 (C function), 811  
vmsifm\_mask\_e32xm2 (C function), 811  
vmsifm\_mask\_e32xm4 (C function), 811  
vmsifm\_mask\_e32xm8 (C function), 811  
vmsifm\_mask\_e64xm1 (C function), 811  
vmsifm\_mask\_e64xm2 (C function), 811  
vmsifm\_mask\_e64xm4 (C function), 811  
vmsifm\_mask\_e64xm8 (C function), 811  
vmsifm\_mask\_e8xm1 (C function), 811  
vmsifm\_mask\_e8xm2 (C function), 811  
vmsifm\_mask\_e8xm4 (C function), 811  
vmsifm\_mask\_e8xm8 (C function), 811  
vmsleuvi\_e16xm1\_uint16xm1 (C function), 392  
vmsleuvi\_e16xm2\_uint16xm2 (C function), 392  
vmsleuvi\_e16xm4\_uint16xm4 (C function), 392  
vmsleuvi\_e16xm8\_uint16xm8 (C function), 392  
vmsleuvi\_e32xm1\_uint32xm1 (C function), 392  
vmsleuvi\_e32xm2\_uint32xm2 (C function), 392  
vmsleuvi\_e32xm4\_uint32xm4 (C function), 392  
vmsleuvi\_e32xm8\_uint32xm8 (C function), 392  
vmsleuvi\_e64xm1\_uint64xm1 (C function), 392  
vmsleuvi\_e64xm2\_uint64xm2 (C function), 392  
vmsleuvi\_e64xm4\_uint64xm4 (C function), 392  
vmsleuvi\_e64xm8\_uint64xm8 (C function), 392  
vmsleuvi\_e8xm1\_uint8xm1 (C function), 392  
vmsleuvi\_e8xm2\_uint8xm2 (C function), 392  
vmsleuvi\_e8xm4\_uint8xm4 (C function), 392  
vmsleuvi\_e8xm8\_uint8xm8 (C function), 392  
vmsleuvi\_mask\_e16xm1\_uint16xm1 (C function), 392  
vmsleuvi\_mask\_e16xm2\_uint16xm2 (C function), 392  
vmsleuvi\_mask\_e16xm4\_uint16xm4 (C function), 393  
vmsleuvi\_mask\_e16xm8\_uint16xm8 (C function), 393  
vmsleuvi\_mask\_e32xm1\_uint32xm1 (C function), 393  
vmsleuvi\_mask\_e32xm2\_uint32xm2 (C function), 393  
vmsleuvi\_mask\_e32xm4\_uint32xm4 (C function), 393  
vmsleuvi\_mask\_e32xm8\_uint32xm8 (C function), 393  
vmsleuvi\_mask\_e64xm1\_uint64xm1 (C function), 393  
vmsleuvi\_mask\_e64xm2\_uint64xm2 (C function), 393  
vmsleuvi\_mask\_e64xm4\_uint64xm4 (C function), 393  
vmsleuvi\_mask\_e64xm8\_uint64xm8 (C function), 393  
vmsleuvi\_mask\_e8xm1\_uint8xm1 (C function), 393  
vmsleuvi\_mask\_e8xm2\_uint8xm2 (C function), 393  
vmsleuvi\_mask\_e8xm4\_uint8xm4 (C function), 393  
vmsleuvi\_mask\_e8xm8\_uint8xm8 (C function), 393  
vmsleuvv\_e16xm1\_uint16xm1 (C function), 393  
vmsleuvv\_e16xm2\_uint16xm2 (C function), 393  
vmsleuvv\_e16xm4\_uint16xm4 (C function), 393  
vmsleuvv\_e16xm8\_uint16xm8 (C function), 393  
vmsleuvv\_e32xm1\_uint32xm1 (C function), 393  
vmsleuvv\_e32xm2\_uint32xm2 (C function), 394  
vmsleuvv\_e32xm4\_uint32xm4 (C function), 394  
vmsleuvv\_e32xm8\_uint32xm8 (C function), 394  
vmsleuvv\_e64xm1\_uint64xm1 (C function), 394  
vmsleuvv\_e64xm2\_uint64xm2 (C function), 394

vmsleuvv\_e64xm4\_uint64xm4 (C function), 394  
 vmsleuvv\_e64xm8\_uint64xm8 (C function), 394  
 vmsleuvv\_e8xm1\_uint8xm1 (C function), 394  
 vmsleuvv\_e8xm2\_uint8xm2 (C function), 394  
 vmsleuvv\_e8xm4\_uint8xm4 (C function), 394  
 vmsleuvv\_e8xm8\_uint8xm8 (C function), 394  
 vmsleuvv\_mask\_e16xm1\_uint16xm1 (C function), 394  
 vmsleuvv\_mask\_e16xm2\_uint16xm2 (C function), 394  
 vmsleuvv\_mask\_e16xm4\_uint16xm4 (C function), 394  
 vmsleuvv\_mask\_e16xm8\_uint16xm8 (C function), 394  
 vmsleuvv\_mask\_e32xm1\_uint32xm1 (C function), 394  
 vmsleuvv\_mask\_e32xm2\_uint32xm2 (C function), 394  
 vmsleuvv\_mask\_e32xm4\_uint32xm4 (C function), 394  
 vmsleuvv\_mask\_e32xm8\_uint32xm8 (C function), 394  
 vmsleuvv\_mask\_e64xm1\_uint64xm1 (C function), 394  
 vmsleuvv\_mask\_e64xm2\_uint64xm2 (C function), 394  
 vmsleuvv\_mask\_e64xm4\_uint64xm4 (C function), 394  
 vmsleuvv\_mask\_e64xm8\_uint64xm8 (C function), 394  
 vmsleuvv\_mask\_e8xm1\_uint8xm1 (C function), 394  
 vmsleuvv\_mask\_e8xm2\_uint8xm2 (C function), 395  
 vmsleuvv\_mask\_e8xm4\_uint8xm4 (C function), 395  
 vmsleuvv\_mask\_e8xm8\_uint8xm8 (C function), 395  
 vmsleuvx\_e16xm1\_uint16xm1 (C function), 395  
 vmsleuvx\_e16xm2\_uint16xm2 (C function), 395  
 vmsleuvx\_e16xm4\_uint16xm4 (C function), 395  
 vmsleuvx\_e16xm8\_uint16xm8 (C function), 395  
 vmsleuvx\_e32xm1\_uint32xm1 (C function), 395  
 vmsleuvx\_e32xm2\_uint32xm2 (C function), 395  
 vmsleuvx\_e32xm4\_uint32xm4 (C function), 395  
 vmsleuvx\_e32xm8\_uint32xm8 (C function), 395  
 vmsleuvx\_e64xm1\_uint64xm1 (C function), 395  
 vmsleuvx\_e64xm2\_uint64xm2 (C function), 395  
 vmsleuvx\_e64xm4\_uint64xm4 (C function), 395  
 vmsleuvx\_e64xm8\_uint64xm8 (C function), 395  
 vmsleuvx\_e8xm1\_uint8xm1 (C function), 395  
 vmsleuvx\_e8xm2\_uint8xm2 (C function), 395  
 vmsleuvx\_e8xm4\_uint8xm4 (C function), 395  
 vmsleuvx\_e8xm8\_uint8xm8 (C function), 395  
 vmsleuvx\_mask\_e16xm1\_uint16xm1 (C function), 396  
 vmsleuvx\_mask\_e16xm2\_uint16xm2 (C function), 396  
 vmsleuvx\_mask\_e16xm4\_uint16xm4 (C function), 396  
 vmsleuvx\_mask\_e16xm8\_uint16xm8 (C function), 396  
 vmsleuvx\_mask\_e32xm1\_uint32xm1 (C function), 396  
 vmsleuvx\_mask\_e32xm2\_uint32xm2 (C function), 396  
 vmsleuvx\_mask\_e32xm4\_uint32xm4 (C function), 396  
 vmsleuvx\_mask\_e32xm8\_uint32xm8 (C function), 396  
 vmsleuvx\_mask\_e64xm1\_uint64xm1 (C function), 396  
 vmsleuvx\_mask\_e64xm2\_uint64xm2 (C function), 396  
 vmsleuvx\_mask\_e64xm4\_uint64xm4 (C function), 396  
 vmsleuvx\_mask\_e64xm8\_uint64xm8 (C function), 396  
 vmsleuvx\_mask\_e8xm1\_uint8xm1 (C function), 396  
 vmsleuvx\_mask\_e8xm2\_uint8xm2 (C function), 396  
 vmsleuvx\_mask\_e8xm4\_uint8xm4 (C function), 396  
 vmsleuvx\_mask\_e8xm8\_uint8xm8 (C function), 396  
 vmslevi\_e16xm1\_int16xm1 (C function), 387  
 vmslevi\_e16xm2\_int16xm2 (C function), 387  
 vmslevi\_e16xm4\_int16xm4 (C function), 387  
 vmslevi\_e16xm8\_int16xm8 (C function), 387  
 vmslevi\_e32xm1\_int32xm1 (C function), 387  
 vmslevi\_e32xm2\_int32xm2 (C function), 387  
 vmslevi\_e32xm4\_int32xm4 (C function), 387  
 vmslevi\_e32xm8\_int32xm8 (C function), 387  
 vmslevi\_e64xm1\_int64xm1 (C function), 387  
 vmslevi\_e64xm2\_int64xm2 (C function), 387  
 vmslevi\_e64xm4\_int64xm4 (C function), 387  
 vmslevi\_e64xm8\_int64xm8 (C function), 387  
 vmslevi\_e8xm1\_int8xm1 (C function), 388  
 vmslevi\_e8xm2\_int8xm2 (C function), 388  
 vmslevi\_e8xm4\_int8xm4 (C function), 388  
 vmslevi\_e8xm8\_int8xm8 (C function), 388  
 vmslevi\_mask\_e16xm1\_int16xm1 (C function), 388  
 vmslevi\_mask\_e16xm2\_int16xm2 (C function), 388  
 vmslevi\_mask\_e16xm4\_int16xm4 (C function), 388  
 vmslevi\_mask\_e16xm8\_int16xm8 (C function), 388  
 vmslevi\_mask\_e32xm1\_int32xm1 (C function), 388  
 vmslevi\_mask\_e32xm2\_int32xm2 (C function), 388  
 vmslevi\_mask\_e32xm4\_int32xm4 (C function), 388  
 vmslevi\_mask\_e32xm8\_int32xm8 (C function), 388  
 vmslevi\_mask\_e64xm1\_int64xm1 (C function), 388  
 vmslevi\_mask\_e64xm2\_int64xm2 (C function), 388  
 vmslevi\_mask\_e64xm4\_int64xm4 (C function), 388  
 vmslevi\_mask\_e64xm8\_int64xm8 (C function), 388  
 vmslevi\_mask\_e8xm1\_int8xm1 (C function), 388  
 vmslevi\_mask\_e8xm2\_int8xm2 (C function), 388  
 vmslevi\_mask\_e8xm4\_int8xm4 (C function), 388  
 vmslevi\_mask\_e8xm8\_int8xm8 (C function), 388  
 vmslevv\_e16xm1\_int16xm1 (C function), 389  
 vmslevv\_e16xm2\_int16xm2 (C function), 389  
 vmslevv\_e16xm4\_int16xm4 (C function), 389  
 vmslevv\_e16xm8\_int16xm8 (C function), 389  
 vmslevv\_e32xm1\_int32xm1 (C function), 389  
 vmslevv\_e32xm2\_int32xm2 (C function), 389  
 vmslevv\_e32xm4\_int32xm4 (C function), 389  
 vmslevv\_e32xm8\_int32xm8 (C function), 389  
 vmslevv\_e64xm1\_int64xm1 (C function), 389  
 vmslevv\_e64xm2\_int64xm2 (C function), 389  
 vmslevv\_e64xm4\_int64xm4 (C function), 389  
 vmslevv\_e64xm8\_int64xm8 (C function), 389  
 vmslevv\_e8xm1\_int8xm1 (C function), 389  
 vmslevv\_e8xm2\_int8xm2 (C function), 389  
 vmslevv\_e8xm4\_int8xm4 (C function), 389  
 vmslevv\_e8xm8\_int8xm8 (C function), 389  
 vmslevv\_mask\_e16xm1\_int16xm1 (C function), 389  
 vmslevv\_mask\_e16xm2\_int16xm2 (C function), 389  
 vmslevv\_mask\_e16xm4\_int16xm4 (C function), 389  
 vmslevv\_mask\_e16xm8\_int16xm8 (C function), 389  
 vmslevv\_mask\_e32xm1\_int32xm1 (C function), 390  
 vmslevv\_mask\_e32xm2\_int32xm2 (C function), 390



vmslevv\_mask\_e32xm4\_int32xm4 (C function), 390  
vmslevv\_mask\_e32xm8\_int32xm8 (C function), 390  
vmslevv\_mask\_e64xm1\_int64xm1 (C function), 390  
vmslevv\_mask\_e64xm2\_int64xm2 (C function), 390  
vmslevv\_mask\_e64xm4\_int64xm4 (C function), 390  
vmslevv\_mask\_e64xm8\_int64xm8 (C function), 390  
vmslevv\_mask\_e8xm1\_int8xm1 (C function), 390  
vmslevv\_mask\_e8xm2\_int8xm2 (C function), 390  
vmslevv\_mask\_e8xm4\_int8xm4 (C function), 390  
vmslevv\_mask\_e8xm8\_int8xm8 (C function), 390  
vmslevx\_e16xm1\_int16xm1 (C function), 390  
vmslevx\_e16xm2\_int16xm2 (C function), 390  
vmslevx\_e16xm4\_int16xm4 (C function), 390  
vmslevx\_e16xm8\_int16xm8 (C function), 390  
vmslevx\_e32xm1\_int32xm1 (C function), 390  
vmslevx\_e32xm2\_int32xm2 (C function), 390  
vmslevx\_e32xm4\_int32xm4 (C function), 390  
vmslevx\_e32xm8\_int32xm8 (C function), 390  
vmslevx\_e64xm1\_int64xm1 (C function), 391  
vmslevx\_e64xm2\_int64xm2 (C function), 391  
vmslevx\_e64xm4\_int64xm4 (C function), 391  
vmslevx\_e64xm8\_int64xm8 (C function), 391  
vmslevx\_e8xm1\_int8xm1 (C function), 391  
vmslevx\_e8xm2\_int8xm2 (C function), 391  
vmslevx\_e8xm4\_int8xm4 (C function), 391  
vmslevx\_e8xm8\_int8xm8 (C function), 391  
vmslevx\_mask\_e16xm1\_int16xm1 (C function), 391  
vmslevx\_mask\_e16xm2\_int16xm2 (C function), 391  
vmslevx\_mask\_e16xm4\_int16xm4 (C function), 391  
vmslevx\_mask\_e16xm8\_int16xm8 (C function), 391  
vmslevx\_mask\_e32xm1\_int32xm1 (C function), 391  
vmslevx\_mask\_e32xm2\_int32xm2 (C function), 391  
vmslevx\_mask\_e32xm4\_int32xm4 (C function), 391  
vmslevx\_mask\_e32xm8\_int32xm8 (C function), 391  
vmslevx\_mask\_e64xm1\_int64xm1 (C function), 391  
vmslevx\_mask\_e64xm2\_int64xm2 (C function), 391  
vmslevx\_mask\_e64xm4\_int64xm4 (C function), 391  
vmslevx\_mask\_e64xm8\_int64xm8 (C function), 391  
vmslevx\_mask\_e8xm1\_int8xm1 (C function), 391  
vmslevx\_mask\_e8xm2\_int8xm2 (C function), 391  
vmslevx\_mask\_e8xm4\_int8xm4 (C function), 391  
vmslevx\_mask\_e8xm8\_int8xm8 (C function), 392  
vmsltuvv\_e16xm1\_uint16xm1 (C function), 400  
vmsltuvv\_e16xm2\_uint16xm2 (C function), 400  
vmsltuvv\_e16xm4\_uint16xm4 (C function), 400  
vmsltuvv\_e16xm8\_uint16xm8 (C function), 400  
vmsltuvv\_e32xm1\_uint32xm1 (C function), 400  
vmsltuvv\_e32xm2\_uint32xm2 (C function), 400  
vmsltuvv\_e32xm4\_uint32xm4 (C function), 400  
vmsltuvv\_e32xm8\_uint32xm8 (C function), 400  
vmsltuvv\_e64xm1\_uint64xm1 (C function), 400  
vmsltuvv\_e64xm2\_uint64xm2 (C function), 400  
vmsltuvv\_e64xm4\_uint64xm4 (C function), 400  
vmsltuvv\_e64xm8\_uint64xm8 (C function), 400  
vmsltuvv\_e8xm1\_uint8xm1 (C function), 400  
vmsltuvv\_e8xm2\_uint8xm2 (C function), 400  
vmsltuvv\_e8xm4\_uint8xm4 (C function), 400  
vmsltuvv\_e8xm8\_uint8xm8 (C function), 400  
vmsltuvv\_mask\_e16xm1\_uint16xm1 (C function), 400  
vmsltuvv\_mask\_e16xm2\_uint16xm2 (C function), 400  
vmsltuvv\_mask\_e16xm4\_uint16xm4 (C function), 400  
vmsltuvv\_mask\_e16xm8\_uint16xm8 (C function), 400  
vmsltuvv\_mask\_e32xm1\_uint32xm1 (C function), 400  
vmsltuvv\_mask\_e32xm2\_uint32xm2 (C function), 400  
vmsltuvv\_mask\_e32xm4\_uint32xm4 (C function), 400  
vmsltuvv\_mask\_e32xm8\_uint32xm8 (C function), 400  
vmsltuvv\_mask\_e64xm1\_uint64xm1 (C function), 401  
vmsltuvv\_mask\_e64xm2\_uint64xm2 (C function), 401  
vmsltuvv\_mask\_e64xm4\_uint64xm4 (C function), 401  
vmsltuvv\_mask\_e64xm8\_uint64xm8 (C function), 401  
vmsltuvv\_mask\_e8xm1\_uint8xm1 (C function), 401  
vmsltuvv\_mask\_e8xm2\_uint8xm2 (C function), 401  
vmsltuvv\_mask\_e8xm4\_uint8xm4 (C function), 401  
vmsltuvv\_mask\_e8xm8\_uint8xm8 (C function), 401  
vmsltuvx\_e16xm1\_uint16xm1 (C function), 401  
vmsltuvx\_e16xm2\_uint16xm2 (C function), 401  
vmsltuvx\_e16xm4\_uint16xm4 (C function), 401  
vmsltuvx\_e16xm8\_uint16xm8 (C function), 401  
vmsltuvx\_e32xm1\_uint32xm1 (C function), 401  
vmsltuvx\_e32xm2\_uint32xm2 (C function), 401  
vmsltuvx\_e32xm4\_uint32xm4 (C function), 401  
vmsltuvx\_e32xm8\_uint32xm8 (C function), 401  
vmsltuvx\_e64xm1\_uint64xm1 (C function), 401  
vmsltuvx\_e64xm2\_uint64xm2 (C function), 401  
vmsltuvx\_e64xm4\_uint64xm4 (C function), 401  
vmsltuvx\_e64xm8\_uint64xm8 (C function), 401  
vmsltuvx\_e8xm1\_uint8xm1 (C function), 401  
vmsltuvx\_e8xm2\_uint8xm2 (C function), 401  
vmsltuvx\_e8xm4\_uint8xm4 (C function), 402  
vmsltuvx\_e8xm8\_uint8xm8 (C function), 402  
vmsltuvx\_mask\_e16xm1\_uint16xm1 (C function), 402  
vmsltuvx\_mask\_e16xm2\_uint16xm2 (C function), 402  
vmsltuvx\_mask\_e16xm4\_uint16xm4 (C function), 402  
vmsltuvx\_mask\_e16xm8\_uint16xm8 (C function), 402  
vmsltuvx\_mask\_e32xm1\_uint32xm1 (C function), 402  
vmsltuvx\_mask\_e32xm2\_uint32xm2 (C function), 402  
vmsltuvx\_mask\_e32xm4\_uint32xm4 (C function), 402  
vmsltuvx\_mask\_e32xm8\_uint32xm8 (C function), 402  
vmsltuvx\_mask\_e64xm1\_uint64xm1 (C function), 402  
vmsltuvx\_mask\_e64xm2\_uint64xm2 (C function), 402  
vmsltuvx\_mask\_e64xm4\_uint64xm4 (C function), 402  
vmsltuvx\_mask\_e64xm8\_uint64xm8 (C function), 402  
vmsltuvx\_mask\_e8xm1\_uint8xm1 (C function), 402  
vmsltuvx\_mask\_e8xm2\_uint8xm2 (C function), 402  
vmsltuvx\_mask\_e8xm4\_uint8xm4 (C function), 402  
vmsltuvx\_mask\_e8xm8\_uint8xm8 (C function), 402  
vmsltvv\_e16xm1\_int16xm1 (C function), 396  
vmsltvv\_e16xm2\_int16xm2 (C function), 397

vmsltvv\_e16xm4\_int16xm4 (C function), 397  
 vmsltvv\_e16xm8\_int16xm8 (C function), 397  
 vmsltvv\_e32xm1\_int32xm1 (C function), 397  
 vmsltvv\_e32xm2\_int32xm2 (C function), 397  
 vmsltvv\_e32xm4\_int32xm4 (C function), 397  
 vmsltvv\_e32xm8\_int32xm8 (C function), 397  
 vmsltvv\_e64xm1\_int64xm1 (C function), 397  
 vmsltvv\_e64xm2\_int64xm2 (C function), 397  
 vmsltvv\_e64xm4\_int64xm4 (C function), 397  
 vmsltvv\_e64xm8\_int64xm8 (C function), 397  
 vmsltvv\_e8xm1\_int8xm1 (C function), 397  
 vmsltvv\_e8xm2\_int8xm2 (C function), 397  
 vmsltvv\_e8xm4\_int8xm4 (C function), 397  
 vmsltvv\_e8xm8\_int8xm8 (C function), 397  
 vmsltvv\_mask\_e16xm1\_int16xm1 (C function), 397  
 vmsltvv\_mask\_e16xm2\_int16xm2 (C function), 397  
 vmsltvv\_mask\_e16xm4\_int16xm4 (C function), 397  
 vmsltvv\_mask\_e16xm8\_int16xm8 (C function), 397  
 vmsltvv\_mask\_e32xm1\_int32xm1 (C function), 397  
 vmsltvv\_mask\_e32xm2\_int32xm2 (C function), 397  
 vmsltvv\_mask\_e32xm4\_int32xm4 (C function), 397  
 vmsltvv\_mask\_e32xm8\_int32xm8 (C function), 397  
 vmsltvv\_mask\_e64xm1\_int64xm1 (C function), 397  
 vmsltvv\_mask\_e64xm2\_int64xm2 (C function), 397  
 vmsltvv\_mask\_e64xm4\_int64xm4 (C function), 398  
 vmsltvv\_mask\_e64xm8\_int64xm8 (C function), 398  
 vmsltvv\_mask\_e8xm1\_int8xm1 (C function), 398  
 vmsltvv\_mask\_e8xm2\_int8xm2 (C function), 398  
 vmsltvv\_mask\_e8xm4\_int8xm4 (C function), 398  
 vmsltvv\_mask\_e8xm8\_int8xm8 (C function), 398  
 vmsltvx\_e16xm1\_int16xm1 (C function), 398  
 vmsltvx\_e16xm2\_int16xm2 (C function), 398  
 vmsltvx\_e16xm4\_int16xm4 (C function), 398  
 vmsltvx\_e16xm8\_int16xm8 (C function), 398  
 vmsltvx\_e32xm1\_int32xm1 (C function), 398  
 vmsltvx\_e32xm2\_int32xm2 (C function), 398  
 vmsltvx\_e32xm4\_int32xm4 (C function), 398  
 vmsltvx\_e32xm8\_int32xm8 (C function), 398  
 vmsltvx\_e64xm1\_int64xm1 (C function), 398  
 vmsltvx\_e64xm2\_int64xm2 (C function), 398  
 vmsltvx\_e64xm4\_int64xm4 (C function), 398  
 vmsltvx\_e64xm8\_int64xm8 (C function), 398  
 vmsltvx\_e8xm1\_int8xm1 (C function), 398  
 vmsltvx\_e8xm2\_int8xm2 (C function), 398  
 vmsltvx\_e8xm4\_int8xm4 (C function), 398  
 vmsltvx\_e8xm8\_int8xm8 (C function), 398  
 vmsltvx\_mask\_e16xm1\_int16xm1 (C function), 399  
 vmsltvx\_mask\_e16xm2\_int16xm2 (C function), 399  
 vmsltvx\_mask\_e16xm4\_int16xm4 (C function), 399  
 vmsltvx\_mask\_e16xm8\_int16xm8 (C function), 399  
 vmsltvx\_mask\_e32xm1\_int32xm1 (C function), 399  
 vmsltvx\_mask\_e32xm2\_int32xm2 (C function), 399  
 vmsltvx\_mask\_e32xm4\_int32xm4 (C function), 399  
 vmsltvx\_mask\_e32xm8\_int32xm8 (C function), 399  
 vmsltvx\_mask\_e64xm1\_int64xm1 (C function), 399  
 vmsltvx\_mask\_e64xm2\_int64xm2 (C function), 399  
 vmsltvx\_mask\_e64xm4\_int64xm4 (C function), 399  
 vmsltvx\_mask\_e64xm8\_int64xm8 (C function), 399  
 vmsltvx\_mask\_e8xm1\_int8xm1 (C function), 399  
 vmsltvx\_mask\_e8xm2\_int8xm2 (C function), 399  
 vmsltvx\_mask\_e8xm4\_int8xm4 (C function), 399  
 vmsltvx\_mask\_e8xm8\_int8xm8 (C function), 399  
 vmsnevi\_e16xm1\_int16xm1 (C function), 403  
 vmsnevi\_e16xm1\_uint16xm1 (C function), 403  
 vmsnevi\_e16xm2\_int16xm2 (C function), 403  
 vmsnevi\_e16xm2\_uint16xm2 (C function), 403  
 vmsnevi\_e16xm4\_int16xm4 (C function), 403  
 vmsnevi\_e16xm4\_uint16xm4 (C function), 403  
 vmsnevi\_e16xm8\_int16xm8 (C function), 403  
 vmsnevi\_e16xm8\_uint16xm8 (C function), 403  
 vmsnevi\_e32xm1\_int32xm1 (C function), 403  
 vmsnevi\_e32xm1\_uint32xm1 (C function), 403  
 vmsnevi\_e32xm2\_int32xm2 (C function), 403  
 vmsnevi\_e32xm2\_uint32xm2 (C function), 403  
 vmsnevi\_e32xm4\_int32xm4 (C function), 403  
 vmsnevi\_e32xm4\_uint32xm4 (C function), 403  
 vmsnevi\_e32xm8\_int32xm8 (C function), 403  
 vmsnevi\_e32xm8\_uint32xm8 (C function), 403  
 vmsnevi\_e64xm1\_int64xm1 (C function), 403  
 vmsnevi\_e64xm1\_uint64xm1 (C function), 403  
 vmsnevi\_e64xm2\_int64xm2 (C function), 403  
 vmsnevi\_e64xm2\_uint64xm2 (C function), 403  
 vmsnevi\_e64xm4\_int64xm4 (C function), 403  
 vmsnevi\_e64xm4\_uint64xm4 (C function), 403  
 vmsnevi\_e64xm8\_int64xm8 (C function), 403  
 vmsnevi\_e64xm8\_uint64xm8 (C function), 403  
 vmsnevi\_e8xm1\_int8xm1 (C function), 403  
 vmsnevi\_e8xm1\_uint8xm1 (C function), 403  
 vmsnevi\_e8xm2\_int8xm2 (C function), 403  
 vmsnevi\_e8xm2\_uint8xm2 (C function), 403  
 vmsnevi\_e8xm4\_int8xm4 (C function), 403  
 vmsnevi\_e8xm4\_uint8xm4 (C function), 404  
 vmsnevi\_e8xm8\_int8xm8 (C function), 404  
 vmsnevi\_e8xm8\_uint8xm8 (C function), 404  
 vmsnevi\_mask\_e16xm1\_int16xm1 (C function), 404  
 vmsnevi\_mask\_e16xm1\_uint16xm1 (C function), 404  
 vmsnevi\_mask\_e16xm2\_int16xm2 (C function), 404  
 vmsnevi\_mask\_e16xm2\_uint16xm2 (C function), 404  
 vmsnevi\_mask\_e16xm4\_int16xm4 (C function), 404  
 vmsnevi\_mask\_e16xm4\_uint16xm4 (C function), 404  
 vmsnevi\_mask\_e16xm8\_int16xm8 (C function), 404  
 vmsnevi\_mask\_e16xm8\_uint16xm8 (C function), 404  
 vmsnevi\_mask\_e32xm1\_int32xm1 (C function), 404  
 vmsnevi\_mask\_e32xm1\_uint32xm1 (C function), 404  
 vmsnevi\_mask\_e32xm2\_int32xm2 (C function), 404  
 vmsnevi\_mask\_e32xm2\_uint32xm2 (C function), 404  
 vmsnevi\_mask\_e32xm4\_int32xm4 (C function), 404  
 vmsnevi\_mask\_e32xm4\_uint32xm4 (C function), 404

vmsnevi\_mask\_e32xm8\_int32xm8 (C function), 404  
vmsnevi\_mask\_e32xm8\_uint32xm8 (C function), 404  
vmsnevi\_mask\_e64xm1\_int64xm1 (C function), 404  
vmsnevi\_mask\_e64xm1\_uint64xm1 (C function), 404  
vmsnevi\_mask\_e64xm2\_int64xm2 (C function), 405  
vmsnevi\_mask\_e64xm2\_uint64xm2 (C function), 405  
vmsnevi\_mask\_e64xm4\_int64xm4 (C function), 405  
vmsnevi\_mask\_e64xm4\_uint64xm4 (C function), 405  
vmsnevi\_mask\_e64xm8\_int64xm8 (C function), 405  
vmsnevi\_mask\_e64xm8\_uint64xm8 (C function), 405  
vmsnevi\_mask\_e8xm1\_int8xm1 (C function), 405  
vmsnevi\_mask\_e8xm1\_uint8xm1 (C function), 405  
vmsnevi\_mask\_e8xm2\_int8xm2 (C function), 405  
vmsnevi\_mask\_e8xm2\_uint8xm2 (C function), 405  
vmsnevi\_mask\_e8xm4\_int8xm4 (C function), 405  
vmsnevi\_mask\_e8xm4\_uint8xm4 (C function), 405  
vmsnevi\_mask\_e8xm8\_int8xm8 (C function), 405  
vmsnevi\_mask\_e8xm8\_uint8xm8 (C function), 405  
vmsnevv\_e16xm1\_int16xm1 (C function), 405  
vmsnevv\_e16xm1\_uint16xm1 (C function), 405  
vmsnevv\_e16xm2\_int16xm2 (C function), 405  
vmsnevv\_e16xm2\_uint16xm2 (C function), 405  
vmsnevv\_e16xm4\_int16xm4 (C function), 405  
vmsnevv\_e16xm4\_uint16xm4 (C function), 406  
vmsnevv\_e16xm8\_int16xm8 (C function), 406  
vmsnevv\_e16xm8\_uint16xm8 (C function), 406  
vmsnevv\_e32xm1\_int32xm1 (C function), 406  
vmsnevv\_e32xm1\_uint32xm1 (C function), 406  
vmsnevv\_e32xm2\_int32xm2 (C function), 406  
vmsnevv\_e32xm2\_uint32xm2 (C function), 406  
vmsnevv\_e32xm4\_int32xm4 (C function), 406  
vmsnevv\_e32xm4\_uint32xm4 (C function), 406  
vmsnevv\_e32xm8\_int32xm8 (C function), 406  
vmsnevv\_e32xm8\_uint32xm8 (C function), 406  
vmsnevv\_e64xm1\_int64xm1 (C function), 406  
vmsnevv\_e64xm1\_uint64xm1 (C function), 406  
vmsnevv\_e64xm2\_int64xm2 (C function), 406  
vmsnevv\_e64xm2\_uint64xm2 (C function), 406  
vmsnevv\_e64xm4\_int64xm4 (C function), 406  
vmsnevv\_e64xm4\_uint64xm4 (C function), 406  
vmsnevv\_e64xm8\_int64xm8 (C function), 406  
vmsnevv\_e64xm8\_uint64xm8 (C function), 406  
vmsnevv\_e8xm1\_int8xm1 (C function), 406  
vmsnevv\_e8xm1\_uint8xm1 (C function), 406  
vmsnevv\_e8xm2\_int8xm2 (C function), 406  
vmsnevv\_e8xm2\_uint8xm2 (C function), 406  
vmsnevv\_e8xm4\_int8xm4 (C function), 406  
vmsnevv\_e8xm4\_uint8xm4 (C function), 406  
vmsnevv\_e8xm8\_int8xm8 (C function), 406  
vmsnevv\_e8xm8\_uint8xm8 (C function), 406  
vmsnevv\_mask\_e16xm1\_int16xm1 (C function), 406  
vmsnevv\_mask\_e16xm1\_uint16xm1 (C function), 406  
vmsnevv\_mask\_e16xm2\_int16xm2 (C function), 406  
vmsnevv\_mask\_e16xm2\_uint16xm2 (C function), 407  
vmsnevv\_mask\_e16xm4\_int16xm4 (C function), 407  
vmsnevv\_mask\_e16xm4\_uint16xm4 (C function), 407  
vmsnevv\_mask\_e16xm8\_int16xm8 (C function), 407  
vmsnevv\_mask\_e16xm8\_uint16xm8 (C function), 407  
vmsnevv\_mask\_e32xm1\_int32xm1 (C function), 407  
vmsnevv\_mask\_e32xm1\_uint32xm1 (C function), 407  
vmsnevv\_mask\_e32xm2\_int32xm2 (C function), 407  
vmsnevv\_mask\_e32xm2\_uint32xm2 (C function), 407  
vmsnevv\_mask\_e32xm4\_int32xm4 (C function), 407  
vmsnevv\_mask\_e32xm4\_uint32xm4 (C function), 407  
vmsnevv\_mask\_e32xm8\_int32xm8 (C function), 407  
vmsnevv\_mask\_e32xm8\_uint32xm8 (C function), 407  
vmsnevv\_mask\_e64xm1\_int64xm1 (C function), 407  
vmsnevv\_mask\_e64xm1\_uint64xm1 (C function), 407  
vmsnevv\_mask\_e64xm2\_int64xm2 (C function), 407  
vmsnevv\_mask\_e64xm2\_uint64xm2 (C function), 407  
vmsnevv\_mask\_e64xm4\_int64xm4 (C function), 407  
vmsnevv\_mask\_e64xm4\_uint64xm4 (C function), 407  
vmsnevv\_mask\_e64xm8\_int64xm8 (C function), 407  
vmsnevv\_mask\_e64xm8\_uint64xm8 (C function), 407  
vmsnevv\_mask\_e8xm1\_int8xm1 (C function), 407  
vmsnevv\_mask\_e8xm1\_uint8xm1 (C function), 407  
vmsnevv\_mask\_e8xm2\_int8xm2 (C function), 408  
vmsnevv\_mask\_e8xm2\_uint8xm2 (C function), 408  
vmsnevv\_mask\_e8xm4\_int8xm4 (C function), 408  
vmsnevv\_mask\_e8xm4\_uint8xm4 (C function), 408  
vmsnevv\_mask\_e8xm8\_int8xm8 (C function), 408  
vmsnevv\_mask\_e8xm8\_uint8xm8 (C function), 408  
vmsnevx\_e16xm1\_int16xm1 (C function), 408  
vmsnevx\_e16xm1\_uint16xm1 (C function), 408  
vmsnevx\_e16xm2\_int16xm2 (C function), 408  
vmsnevx\_e16xm2\_uint16xm2 (C function), 408  
vmsnevx\_e16xm4\_int16xm4 (C function), 408  
vmsnevx\_e16xm4\_uint16xm4 (C function), 408  
vmsnevx\_e16xm8\_int16xm8 (C function), 408  
vmsnevx\_e16xm8\_uint16xm8 (C function), 408  
vmsnevx\_e32xm1\_int32xm1 (C function), 408  
vmsnevx\_e32xm1\_uint32xm1 (C function), 408  
vmsnevx\_e32xm2\_int32xm2 (C function), 408  
vmsnevx\_e32xm2\_uint32xm2 (C function), 408  
vmsnevx\_e32xm4\_int32xm4 (C function), 408  
vmsnevx\_e32xm4\_uint32xm4 (C function), 408  
vmsnevx\_e32xm8\_int32xm8 (C function), 408  
vmsnevx\_e32xm8\_uint32xm8 (C function), 408  
vmsnevx\_e64xm1\_int64xm1 (C function), 408  
vmsnevx\_e64xm1\_uint64xm1 (C function), 409  
vmsnevx\_e64xm2\_int64xm2 (C function), 409  
vmsnevx\_e64xm2\_uint64xm2 (C function), 409  
vmsnevx\_e64xm4\_int64xm4 (C function), 409  
vmsnevx\_e64xm4\_uint64xm4 (C function), 409  
vmsnevx\_e64xm8\_int64xm8 (C function), 409  
vmsnevx\_e64xm8\_uint64xm8 (C function), 409  
vmsnevx\_e8xm1\_int8xm1 (C function), 409  
vmsnevx\_e8xm1\_uint8xm1 (C function), 409

- vmsnevx\_e8xm2\_int8xm2 (C function), 409  
 vmsnevx\_e8xm2\_uint8xm2 (C function), 409  
 vmsnevx\_e8xm4\_int8xm4 (C function), 409  
 vmsnevx\_e8xm4\_uint8xm4 (C function), 409  
 vmsnevx\_e8xm8\_int8xm8 (C function), 409  
 vmsnevx\_e8xm8\_uint8xm8 (C function), 409  
 vmsnevx\_mask\_e16xm1\_int16xm1 (C function), 409  
 vmsnevx\_mask\_e16xm1\_uint16xm1 (C function), 409  
 vmsnevx\_mask\_e16xm2\_int16xm2 (C function), 409  
 vmsnevx\_mask\_e16xm2\_uint16xm2 (C function), 409  
 vmsnevx\_mask\_e16xm4\_int16xm4 (C function), 409  
 vmsnevx\_mask\_e16xm4\_uint16xm4 (C function), 409  
 vmsnevx\_mask\_e16xm8\_int16xm8 (C function), 409  
 vmsnevx\_mask\_e16xm8\_uint16xm8 (C function), 409  
 vmsnevx\_mask\_e32xm1\_int32xm1 (C function), 409  
 vmsnevx\_mask\_e32xm1\_uint32xm1 (C function), 409  
 vmsnevx\_mask\_e32xm2\_int32xm2 (C function), 410  
 vmsnevx\_mask\_e32xm2\_uint32xm2 (C function), 410  
 vmsnevx\_mask\_e32xm4\_int32xm4 (C function), 410  
 vmsnevx\_mask\_e32xm4\_uint32xm4 (C function), 410  
 vmsnevx\_mask\_e32xm8\_int32xm8 (C function), 410  
 vmsnevx\_mask\_e32xm8\_uint32xm8 (C function), 410  
 vmsnevx\_mask\_e64xm1\_int64xm1 (C function), 410  
 vmsnevx\_mask\_e64xm1\_uint64xm1 (C function), 410  
 vmsnevx\_mask\_e64xm2\_int64xm2 (C function), 410  
 vmsnevx\_mask\_e64xm2\_uint64xm2 (C function), 410  
 vmsnevx\_mask\_e64xm4\_int64xm4 (C function), 410  
 vmsnevx\_mask\_e64xm4\_uint64xm4 (C function), 410  
 vmsnevx\_mask\_e64xm8\_int64xm8 (C function), 410  
 vmsnevx\_mask\_e64xm8\_uint64xm8 (C function), 410  
 vmsnevx\_mask\_e8xm1\_int8xm1 (C function), 410  
 vmsnevx\_mask\_e8xm1\_uint8xm1 (C function), 410  
 vmsnevx\_mask\_e8xm2\_int8xm2 (C function), 410  
 vmsnevx\_mask\_e8xm2\_uint8xm2 (C function), 410  
 vmsnevx\_mask\_e8xm4\_int8xm4 (C function), 410  
 vmsnevx\_mask\_e8xm4\_uint8xm4 (C function), 410  
 vmsnevx\_mask\_e8xm8\_int8xm8 (C function), 410  
 vmsnevx\_mask\_e8xm8\_uint8xm8 (C function), 410  
 vmsofm\_e16xm1 (C function), 811  
 vmsofm\_e16xm2 (C function), 811  
 vmsofm\_e16xm4 (C function), 811  
 vmsofm\_e16xm8 (C function), 811  
 vmsofm\_e32xm1 (C function), 812  
 vmsofm\_e32xm2 (C function), 812  
 vmsofm\_e32xm4 (C function), 812  
 vmsofm\_e32xm8 (C function), 812  
 vmsofm\_e64xm1 (C function), 812  
 vmsofm\_e64xm2 (C function), 812  
 vmsofm\_e64xm4 (C function), 812  
 vmsofm\_e64xm8 (C function), 812  
 vmsofm\_e8xm1 (C function), 812  
 vmsofm\_e8xm2 (C function), 812  
 vmsofm\_e8xm4 (C function), 812  
 vmsofm\_e8xm8 (C function), 812  
 vmsofm\_mask\_e16xm1 (C function), 812  
 vmsofm\_mask\_e16xm2 (C function), 812  
 vmsofm\_mask\_e16xm4 (C function), 812  
 vmsofm\_mask\_e16xm8 (C function), 812  
 vmsofm\_mask\_e32xm1 (C function), 812  
 vmsofm\_mask\_e32xm2 (C function), 812  
 vmsofm\_mask\_e32xm4 (C function), 812  
 vmsofm\_mask\_e32xm8 (C function), 812  
 vmsofm\_mask\_e64xm1 (C function), 812  
 vmsofm\_mask\_e64xm2 (C function), 812  
 vmsofm\_mask\_e64xm4 (C function), 812  
 vmsofm\_mask\_e64xm8 (C function), 812  
 vmsofm\_mask\_e8xm1 (C function), 812  
 vmsofm\_mask\_e8xm2 (C function), 812  
 vmsofm\_mask\_e8xm4 (C function), 812  
 vmsofm\_mask\_e8xm8 (C function), 812  
 vmulhsuvv\_int16xm1\_uint16xm1 (C function), 225  
 vmulhsuvv\_int16xm2\_uint16xm2 (C function), 225  
 vmulhsuvv\_int16xm4\_uint16xm4 (C function), 225  
 vmulhsuvv\_int16xm8\_uint16xm8 (C function), 225  
 vmulhsuvv\_int32xm1\_uint32xm1 (C function), 225  
 vmulhsuvv\_int32xm2\_uint32xm2 (C function), 225  
 vmulhsuvv\_int32xm4\_uint32xm4 (C function), 225  
 vmulhsuvv\_int32xm8\_uint32xm8 (C function), 225  
 vmulhsuvv\_int64xm1\_uint64xm1 (C function), 225  
 vmulhsuvv\_int64xm2\_uint64xm2 (C function), 225  
 vmulhsuvv\_int64xm4\_uint64xm4 (C function), 225  
 vmulhsuvv\_int64xm8\_uint64xm8 (C function), 225  
 vmulhsuvv\_int8xm1\_uint8xm1 (C function), 225  
 vmulhsuvv\_int8xm2\_uint8xm2 (C function), 225  
 vmulhsuvv\_int8xm4\_uint8xm4 (C function), 225  
 vmulhsuvv\_int8xm8\_uint8xm8 (C function), 225  
 vmulhsuvv\_mask\_int16xm1\_uint16xm1 (C function), 225  
 vmulhsuvv\_mask\_int16xm2\_uint16xm2 (C function), 225  
 vmulhsuvv\_mask\_int16xm4\_uint16xm4 (C function), 225  
 vmulhsuvv\_mask\_int16xm8\_uint16xm8 (C function), 226  
 vmulhsuvv\_mask\_int32xm1\_uint32xm1 (C function), 226  
 vmulhsuvv\_mask\_int32xm2\_uint32xm2 (C function), 226  
 vmulhsuvv\_mask\_int32xm4\_uint32xm4 (C function), 226  
 vmulhsuvv\_mask\_int32xm8\_uint32xm8 (C function), 226  
 vmulhsuvv\_mask\_int64xm1\_uint64xm1 (C function), 226  
 vmulhsuvv\_mask\_int64xm2\_uint64xm2 (C function), 226  
 vmulhsuvv\_mask\_int64xm4\_uint64xm4 (C function), 226



- vmulhsuvv\_mask\_int64xm8\_uint64xm8 (C function), 226  
 vmulhsuvv\_mask\_int8xm1\_uint8xm1 (C function), 226  
 vmulhsuvv\_mask\_int8xm2\_uint8xm2 (C function), 226  
 vmulhsuvv\_mask\_int8xm4\_uint8xm4 (C function), 226  
 vmulhsuvv\_mask\_int8xm8\_uint8xm8 (C function), 226  
 vmulhsuvx\_int16xm1 (C function), 227  
 vmulhsuvx\_int16xm2 (C function), 227  
 vmulhsuvx\_int16xm4 (C function), 227  
 vmulhsuvx\_int16xm8 (C function), 227  
 vmulhsuvx\_int32xm1 (C function), 227  
 vmulhsuvx\_int32xm2 (C function), 227  
 vmulhsuvx\_int32xm4 (C function), 227  
 vmulhsuvx\_int32xm8 (C function), 227  
 vmulhsuvx\_int64xm1 (C function), 227  
 vmulhsuvx\_int64xm2 (C function), 227  
 vmulhsuvx\_int64xm4 (C function), 227  
 vmulhsuvx\_int64xm8 (C function), 227  
 vmulhsuvx\_int8xm1 (C function), 227  
 vmulhsuvx\_int8xm2 (C function), 227  
 vmulhsuvx\_int8xm4 (C function), 227  
 vmulhsuvx\_int8xm8 (C function), 227  
 vmulhsuvx\_mask\_int16xm1 (C function), 227  
 vmulhsuvx\_mask\_int16xm2 (C function), 227  
 vmulhsuvx\_mask\_int16xm4 (C function), 227  
 vmulhsuvx\_mask\_int16xm8 (C function), 227  
 vmulhsuvx\_mask\_int32xm1 (C function), 227  
 vmulhsuvx\_mask\_int32xm2 (C function), 227  
 vmulhsuvx\_mask\_int32xm4 (C function), 227  
 vmulhsuvx\_mask\_int32xm8 (C function), 227  
 vmulhsuvx\_mask\_int64xm1 (C function), 227  
 vmulhsuvx\_mask\_int64xm2 (C function), 228  
 vmulhsuvx\_mask\_int64xm4 (C function), 228  
 vmulhsuvx\_mask\_int64xm8 (C function), 228  
 vmulhsuvx\_mask\_int8xm1 (C function), 228  
 vmulhsuvx\_mask\_int8xm2 (C function), 228  
 vmulhsuvx\_mask\_int8xm4 (C function), 228  
 vmulhsuvx\_mask\_int8xm8 (C function), 228  
 vmulhuvv\_mask\_uint16xm1 (C function), 229  
 vmulhuvv\_mask\_uint16xm2 (C function), 229  
 vmulhuvv\_mask\_uint16xm4 (C function), 229  
 vmulhuvv\_mask\_uint16xm8 (C function), 229  
 vmulhuvv\_mask\_uint32xm1 (C function), 229  
 vmulhuvv\_mask\_uint32xm2 (C function), 229  
 vmulhuvv\_mask\_uint32xm4 (C function), 229  
 vmulhuvv\_mask\_uint32xm8 (C function), 229  
 vmulhuvv\_mask\_uint64xm1 (C function), 229  
 vmulhuvv\_mask\_uint64xm2 (C function), 229  
 vmulhuvv\_mask\_uint64xm4 (C function), 229  
 vmulhuvv\_mask\_uint64xm8 (C function), 229  
 vmulhuvv\_mask\_uint8xm1 (C function), 229  
 vmulhuvv\_mask\_uint8xm2 (C function), 229  
 vmulhuvv\_mask\_uint8xm4 (C function), 229  
 vmulhuvv\_mask\_uint8xm8 (C function), 229  
 vmulhuvv\_uint16xm1 (C function), 228  
 vmulhuvv\_uint16xm2 (C function), 228  
 vmulhuvv\_uint16xm4 (C function), 228  
 vmulhuvv\_uint16xm8 (C function), 228  
 vmulhuvv\_uint32xm1 (C function), 228  
 vmulhuvv\_uint32xm2 (C function), 228  
 vmulhuvv\_uint32xm4 (C function), 228  
 vmulhuvv\_uint32xm8 (C function), 228  
 vmulhuvv\_uint64xm1 (C function), 228  
 vmulhuvv\_uint64xm2 (C function), 228  
 vmulhuvv\_uint64xm4 (C function), 228  
 vmulhuvv\_uint64xm8 (C function), 228  
 vmulhuvv\_uint8xm1 (C function), 228  
 vmulhuvv\_uint8xm2 (C function), 228  
 vmulhuvv\_uint8xm4 (C function), 228  
 vmulhuvv\_uint8xm8 (C function), 229  
 vmulhuvx\_mask\_uint16xm1 (C function), 230  
 vmulhuvx\_mask\_uint16xm2 (C function), 230  
 vmulhuvx\_mask\_uint16xm4 (C function), 230  
 vmulhuvx\_mask\_uint16xm8 (C function), 230  
 vmulhuvx\_mask\_uint32xm1 (C function), 230  
 vmulhuvx\_mask\_uint32xm2 (C function), 230  
 vmulhuvx\_mask\_uint32xm4 (C function), 230  
 vmulhuvx\_mask\_uint32xm8 (C function), 230  
 vmulhuvx\_mask\_uint64xm1 (C function), 231  
 vmulhuvx\_mask\_uint64xm2 (C function), 231  
 vmulhuvx\_mask\_uint64xm4 (C function), 231  
 vmulhuvx\_mask\_uint64xm8 (C function), 231  
 vmulhuvx\_mask\_uint8xm1 (C function), 231  
 vmulhuvx\_mask\_uint8xm2 (C function), 231  
 vmulhuvx\_mask\_uint8xm4 (C function), 231  
 vmulhuvx\_mask\_uint8xm8 (C function), 231  
 vmulhuvx\_uint16xm1 (C function), 230  
 vmulhuvx\_uint16xm2 (C function), 230  
 vmulhuvx\_uint16xm4 (C function), 230  
 vmulhuvx\_uint16xm8 (C function), 230  
 vmulhuvx\_uint32xm1 (C function), 230  
 vmulhuvx\_uint32xm2 (C function), 230  
 vmulhuvx\_uint32xm4 (C function), 230  
 vmulhuvx\_uint32xm8 (C function), 230  
 vmulhuvx\_uint64xm1 (C function), 230  
 vmulhuvx\_uint64xm2 (C function), 230  
 vmulhuvx\_uint64xm4 (C function), 230  
 vmulhuvx\_uint64xm8 (C function), 230  
 vmulhuvx\_uint8xm1 (C function), 230  
 vmulhuvx\_uint8xm2 (C function), 230  
 vmulhuvx\_uint8xm4 (C function), 230  
 vmulhuvx\_uint8xm8 (C function), 230  
 vmulhvv\_int16xm1 (C function), 222  
 vmulhvv\_int16xm2 (C function), 222  
 vmulhvv\_int16xm4 (C function), 222  
 vmulhvv\_int16xm8 (C function), 222  
 vmulhvv\_int32xm1 (C function), 222  
 vmulhvv\_int32xm2 (C function), 222

vmulhvv\_int32xm4 (C function), 222  
 vmulhvv\_int32xm8 (C function), 222  
 vmulhvv\_int64xm1 (C function), 222  
 vmulhvv\_int64xm2 (C function), 222  
 vmulhvv\_int64xm4 (C function), 222  
 vmulhvv\_int64xm8 (C function), 222  
 vmulhvv\_int8xm1 (C function), 222  
 vmulhvv\_int8xm2 (C function), 222  
 vmulhvv\_int8xm4 (C function), 222  
 vmulhvv\_int8xm8 (C function), 222  
 vmulhvv\_mask\_int16xm1 (C function), 222  
 vmulhvv\_mask\_int16xm2 (C function), 222  
 vmulhvv\_mask\_int16xm4 (C function), 222  
 vmulhvv\_mask\_int16xm8 (C function), 222  
 vmulhvv\_mask\_int32xm1 (C function), 222  
 vmulhvv\_mask\_int32xm2 (C function), 222  
 vmulhvv\_mask\_int32xm4 (C function), 222  
 vmulhvv\_mask\_int32xm8 (C function), 222  
 vmulhvv\_mask\_int64xm1 (C function), 223  
 vmulhvv\_mask\_int64xm2 (C function), 223  
 vmulhvv\_mask\_int64xm4 (C function), 223  
 vmulhvv\_mask\_int64xm8 (C function), 223  
 vmulhvv\_mask\_int8xm1 (C function), 223  
 vmulhvv\_mask\_int8xm2 (C function), 223  
 vmulhvv\_mask\_int8xm4 (C function), 223  
 vmulhvv\_mask\_int8xm8 (C function), 223  
 vmulhvx\_int16xm1 (C function), 223  
 vmulhvx\_int16xm2 (C function), 223  
 vmulhvx\_int16xm4 (C function), 223  
 vmulhvx\_int16xm8 (C function), 223  
 vmulhvx\_int32xm1 (C function), 223  
 vmulhvx\_int32xm2 (C function), 223  
 vmulhvx\_int32xm4 (C function), 223  
 vmulhvx\_int32xm8 (C function), 223  
 vmulhvx\_int64xm1 (C function), 223  
 vmulhvx\_int64xm2 (C function), 223  
 vmulhvx\_int64xm4 (C function), 223  
 vmulhvx\_int64xm8 (C function), 223  
 vmulhvx\_int8xm1 (C function), 223  
 vmulhvx\_int8xm2 (C function), 223  
 vmulhvx\_int8xm4 (C function), 224  
 vmulhvx\_int8xm8 (C function), 224  
 vmulhvx\_mask\_int16xm1 (C function), 224  
 vmulhvx\_mask\_int16xm2 (C function), 224  
 vmulhvx\_mask\_int16xm4 (C function), 224  
 vmulhvx\_mask\_int16xm8 (C function), 224  
 vmulhvx\_mask\_int32xm1 (C function), 224  
 vmulhvx\_mask\_int32xm2 (C function), 224  
 vmulhvx\_mask\_int32xm4 (C function), 224  
 vmulhvx\_mask\_int32xm8 (C function), 224  
 vmulhvx\_mask\_int64xm1 (C function), 224  
 vmulhvx\_mask\_int64xm2 (C function), 224  
 vmulhvx\_mask\_int64xm4 (C function), 224  
 vmulhvx\_mask\_int64xm8 (C function), 224  
 vmulhvx\_mask\_int8xm1 (C function), 224  
 vmulhvx\_mask\_int8xm2 (C function), 224  
 vmulhvx\_mask\_int8xm4 (C function), 224  
 vmulhvx\_mask\_int8xm8 (C function), 224  
 vmulvv\_int16xm1 (C function), 217  
 vmulvv\_int16xm2 (C function), 217  
 vmulvv\_int16xm4 (C function), 217  
 vmulvv\_int16xm8 (C function), 217  
 vmulvv\_int32xm1 (C function), 217  
 vmulvv\_int32xm2 (C function), 218  
 vmulvv\_int32xm4 (C function), 218  
 vmulvv\_int32xm8 (C function), 218  
 vmulvv\_int64xm1 (C function), 218  
 vmulvv\_int64xm2 (C function), 218  
 vmulvv\_int64xm4 (C function), 218  
 vmulvv\_int64xm8 (C function), 218  
 vmulvv\_int8xm1 (C function), 218  
 vmulvv\_int8xm2 (C function), 218  
 vmulvv\_int8xm4 (C function), 218  
 vmulvv\_int8xm8 (C function), 218  
 vmulvv\_mask\_int16xm1 (C function), 218  
 vmulvv\_mask\_int16xm2 (C function), 218  
 vmulvv\_mask\_int16xm4 (C function), 218  
 vmulvv\_mask\_int16xm8 (C function), 219  
 vmulvv\_mask\_int32xm1 (C function), 219  
 vmulvv\_mask\_int32xm2 (C function), 219  
 vmulvv\_mask\_int32xm4 (C function), 219  
 vmulvv\_mask\_int32xm8 (C function), 219  
 vmulvv\_mask\_int64xm1 (C function), 219  
 vmulvv\_mask\_int64xm2 (C function), 219  
 vmulvv\_mask\_int64xm4 (C function), 219  
 vmulvv\_mask\_int64xm8 (C function), 219  
 vmulvv\_mask\_int8xm1 (C function), 219  
 vmulvv\_mask\_int8xm2 (C function), 219  
 vmulvv\_mask\_int8xm4 (C function), 219  
 vmulvv\_mask\_int8xm8 (C function), 219  
 vmulvv\_mask\_uint16xm1 (C function), 219  
 vmulvv\_mask\_uint16xm2 (C function), 219  
 vmulvv\_mask\_uint16xm4 (C function), 219  
 vmulvv\_mask\_uint16xm8 (C function), 219  
 vmulvv\_mask\_uint32xm1 (C function), 219  
 vmulvv\_mask\_uint32xm2 (C function), 219  
 vmulvv\_mask\_uint32xm4 (C function), 219  
 vmulvv\_mask\_uint32xm8 (C function), 219  
 vmulvv\_mask\_uint64xm1 (C function), 219  
 vmulvv\_mask\_uint64xm2 (C function), 219  
 vmulvv\_mask\_uint64xm4 (C function), 220  
 vmulvv\_mask\_uint64xm8 (C function), 220  
 vmulvv\_mask\_uint8xm1 (C function), 220  
 vmulvv\_mask\_uint8xm2 (C function), 220  
 vmulvv\_mask\_uint8xm4 (C function), 220  
 vmulvv\_mask\_uint8xm8 (C function), 220  
 vmulvv\_uint16xm1 (C function), 218  
 vmulvv\_uint16xm2 (C function), 218



vmulvv\_uint16xm4 (C function), 218  
vmulvv\_uint16xm8 (C function), 218  
vmulvv\_uint32xm1 (C function), 218  
vmulvv\_uint32xm2 (C function), 218  
vmulvv\_uint32xm4 (C function), 218  
vmulvv\_uint32xm8 (C function), 218  
vmulvv\_uint64xm1 (C function), 218  
vmulvv\_uint64xm2 (C function), 218  
vmulvv\_uint64xm4 (C function), 218  
vmulvv\_uint64xm8 (C function), 218  
vmulvv\_uint8xm1 (C function), 218  
vmulvv\_uint8xm2 (C function), 218  
vmulvv\_uint8xm4 (C function), 218  
vmulvv\_uint8xm8 (C function), 218  
vmulvx\_int16xm1 (C function), 220  
vmulvx\_int16xm2 (C function), 220  
vmulvx\_int16xm4 (C function), 220  
vmulvx\_int16xm8 (C function), 220  
vmulvx\_int32xm1 (C function), 220  
vmulvx\_int32xm2 (C function), 220  
vmulvx\_int32xm4 (C function), 220  
vmulvx\_int32xm8 (C function), 220  
vmulvx\_int64xm1 (C function), 220  
vmulvx\_int64xm2 (C function), 220  
vmulvx\_int64xm4 (C function), 220  
vmulvx\_int64xm8 (C function), 220  
vmulvx\_int8xm1 (C function), 220  
vmulvx\_int8xm2 (C function), 220  
vmulvx\_int8xm4 (C function), 220  
vmulvx\_int8xm8 (C function), 220  
vmulvx\_mask\_int16xm1 (C function), 221  
vmulvx\_mask\_int16xm2 (C function), 221  
vmulvx\_mask\_int16xm4 (C function), 221  
vmulvx\_mask\_int16xm8 (C function), 221  
vmulvx\_mask\_int32xm1 (C function), 221  
vmulvx\_mask\_int32xm2 (C function), 221  
vmulvx\_mask\_int32xm4 (C function), 221  
vmulvx\_mask\_int32xm8 (C function), 221  
vmulvx\_mask\_int64xm1 (C function), 221  
vmulvx\_mask\_int64xm2 (C function), 221  
vmulvx\_mask\_int64xm4 (C function), 221  
vmulvx\_mask\_int64xm8 (C function), 221  
vmulvx\_mask\_int8xm1 (C function), 221  
vmulvx\_mask\_int8xm2 (C function), 221  
vmulvx\_mask\_int8xm4 (C function), 221  
vmulvx\_mask\_int8xm8 (C function), 221  
vmvsx\_int16xm1 (C function), 826  
vmvsx\_int16xm2 (C function), 826  
vmvsx\_int16xm4 (C function), 826  
vmvsx\_int16xm8 (C function), 826  
vmvsx\_int32xm1 (C function), 826  
vmvsx\_int32xm2 (C function), 827  
vmvsx\_int32xm4 (C function), 827  
vmvsx\_int32xm8 (C function), 827  
vmvsx\_int64xm1 (C function), 827  
vmvsx\_int64xm2 (C function), 827  
vmvsx\_int64xm4 (C function), 827  
vmvsx\_int64xm8 (C function), 827  
vmvsx\_int8xm1 (C function), 827  
vmvsx\_int8xm2 (C function), 827  
vmvsx\_int8xm4 (C function), 827  
vmvsx\_int8xm8 (C function), 827  
vmvsx\_uint16xm1 (C function), 827  
vmvsx\_uint16xm2 (C function), 827  
vmvsx\_uint16xm4 (C function), 827  
vmvsx\_uint16xm8 (C function), 827  
vmvsx\_uint32xm1 (C function), 827  
vmvsx\_uint32xm2 (C function), 827  
vmvsx\_uint32xm4 (C function), 827  
vmvsx\_uint32xm8 (C function), 827  
vmvsx\_uint64xm1 (C function), 827  
vmvsx\_uint64xm2 (C function), 827  
vmvsx\_uint64xm4 (C function), 827  
vmvsx\_uint64xm8 (C function), 827  
vmvsx\_uint8xm1 (C function), 827  
vmvsx\_uint8xm2 (C function), 827  
vmvsx\_uint8xm4 (C function), 827  
vmvsx\_uint8xm8 (C function), 827  
vmvvi\_int16xm1 (C function), 827  
vmvvi\_int16xm2 (C function), 827  
vmvvi\_int16xm4 (C function), 828  
vmvvi\_int16xm8 (C function), 828  
vmvvi\_int32xm1 (C function), 828  
vmvvi\_int32xm2 (C function), 828  
vmvvi\_int32xm4 (C function), 828  
vmvvi\_int32xm8 (C function), 828  
vmvvi\_int64xm1 (C function), 828  
vmvvi\_int64xm2 (C function), 828  
vmvvi\_int64xm4 (C function), 828  
vmvvi\_int64xm8 (C function), 828  
vmvvi\_int8xm1 (C function), 828  
vmvvi\_int8xm2 (C function), 828  
vmvvi\_int8xm4 (C function), 828  
vmvvi\_int8xm8 (C function), 828  
vmvvv\_float16xm1 (C function), 828  
vmvvv\_float16xm2 (C function), 828  
vmvvv\_float16xm4 (C function), 828  
vmvvv\_float16xm8 (C function), 828  
vmvvv\_float32xm1 (C function), 828  
vmvvv\_float32xm2 (C function), 828  
vmvvv\_float32xm4 (C function), 828  
vmvvv\_float32xm8 (C function), 828  
vmvvv\_float64xm1 (C function), 828  
vmvvv\_float64xm2 (C function), 828  
vmvvv\_float64xm4 (C function), 828  
vmvvv\_float64xm8 (C function), 828  
vmvvv\_int16xm1 (C function), 828  
vmvvv\_int16xm2 (C function), 828

vmvvy\_int16xm4 (C function), 829  
 vmvvy\_int16xm8 (C function), 829  
 vmvvy\_int32xm1 (C function), 829  
 vmvvy\_int32xm2 (C function), 829  
 vmvvy\_int32xm4 (C function), 829  
 vmvvy\_int32xm8 (C function), 829  
 vmvvy\_int64xm1 (C function), 829  
 vmvvy\_int64xm2 (C function), 829  
 vmvvy\_int64xm4 (C function), 829  
 vmvvy\_int64xm8 (C function), 829  
 vmvvy\_int8xm1 (C function), 829  
 vmvvy\_int8xm2 (C function), 829  
 vmvvy\_int8xm4 (C function), 829  
 vmvvy\_int8xm8 (C function), 829  
 vmvvy\_uint16xm1 (C function), 829  
 vmvvy\_uint16xm2 (C function), 829  
 vmvvy\_uint16xm4 (C function), 829  
 vmvvy\_uint16xm8 (C function), 829  
 vmvvy\_uint32xm1 (C function), 829  
 vmvvy\_uint32xm2 (C function), 829  
 vmvvy\_uint32xm4 (C function), 829  
 vmvvy\_uint32xm8 (C function), 829  
 vmvvy\_uint64xm1 (C function), 829  
 vmvvy\_uint64xm2 (C function), 829  
 vmvvy\_uint64xm4 (C function), 829  
 vmvvy\_uint64xm8 (C function), 829  
 vmvvy\_uint8xm1 (C function), 829  
 vmvvy\_uint8xm2 (C function), 829  
 vmvvy\_uint8xm4 (C function), 829  
 vmvvy\_uint8xm8 (C function), 829  
 vmvxx\_int16xm1 (C function), 830  
 vmvxx\_int16xm2 (C function), 830  
 vmvxx\_int16xm4 (C function), 830  
 vmvxx\_int16xm8 (C function), 830  
 vmvxx\_int32xm1 (C function), 830  
 vmvxx\_int32xm2 (C function), 830  
 vmvxx\_int32xm4 (C function), 830  
 vmvxx\_int32xm8 (C function), 830  
 vmvxx\_int64xm1 (C function), 830  
 vmvxx\_int64xm2 (C function), 830  
 vmvxx\_int64xm4 (C function), 830  
 vmvxx\_int64xm8 (C function), 830  
 vmvxx\_int8xm1 (C function), 830  
 vmvxx\_int8xm2 (C function), 830  
 vmvxx\_int8xm4 (C function), 830  
 vmvxx\_int8xm8 (C function), 830  
 vmvxx\_uint16xm1 (C function), 830  
 vmvxx\_uint16xm2 (C function), 830  
 vmvxx\_uint16xm4 (C function), 830  
 vmvxx\_uint16xm8 (C function), 830  
 vmvxx\_uint32xm1 (C function), 830  
 vmvxx\_uint32xm2 (C function), 830  
 vmvxx\_uint32xm4 (C function), 830  
 vmvxx\_uint32xm8 (C function), 830  
 vmvxx\_uint64xm1 (C function), 830  
 vmvxx\_uint64xm2 (C function), 830  
 vmvxx\_uint64xm4 (C function), 830  
 vmvxx\_uint64xm8 (C function), 830  
 vmvxx\_uint8xm1 (C function), 830  
 vmvxx\_uint8xm2 (C function), 830  
 vmvxx\_uint8xm4 (C function), 830  
 vmvxx\_uint8xm8 (C function), 830  
 vmxnorrm\_e16xm1 (C function), 813  
 vmxnorrm\_e16xm2 (C function), 813  
 vmxnorrm\_e16xm4 (C function), 813  
 vmxnorrm\_e16xm8 (C function), 813  
 vmxnorrm\_e32xm1 (C function), 813  
 vmxnorrm\_e32xm2 (C function), 813  
 vmxnorrm\_e32xm4 (C function), 813  
 vmxnorrm\_e32xm8 (C function), 813  
 vmxnorrm\_e64xm1 (C function), 813  
 vmxnorrm\_e64xm2 (C function), 813  
 vmxnorrm\_e64xm4 (C function), 813  
 vmxnorrm\_e64xm8 (C function), 813  
 vmxnorrm\_e8xm1 (C function), 813  
 vmxnorrm\_e8xm2 (C function), 813

vmxnorrm\_e8xm4 (C function), 813  
 vmxnorrm\_e8xm8 (C function), 813  
 vmxnorrm\_e16xm1 (C function), 813  
 vmxnorrm\_e16xm2 (C function), 813  
 vmxnorrm\_e16xm4 (C function), 814  
 vmxnorrm\_e16xm8 (C function), 814  
 vmxnorrm\_e32xm1 (C function), 814  
 vmxnorrm\_e32xm2 (C function), 814  
 vmxnorrm\_e32xm4 (C function), 814  
 vmxnorrm\_e32xm8 (C function), 814  
 vmxnorrm\_e64xm1 (C function), 814  
 vmxnorrm\_e64xm2 (C function), 814  
 vmxnorrm\_e64xm4 (C function), 814  
 vmxnorrm\_e64xm8 (C function), 814  
 vmxnorrm\_e8xm1 (C function), 814  
 vmxnorrm\_e8xm2 (C function), 814  
 vmxnorrm\_e8xm4 (C function), 814  
 vmxnorrm\_e8xm8 (C function), 814  
 vnclipuvi\_mask\_uint16xm1\_uint32xm2 (C function), 235  
 vnclipuvi\_mask\_uint16xm2\_uint32xm4 (C function), 235  
 vnclipuvi\_mask\_uint16xm4\_uint32xm8 (C function), 235  
 vnclipuvi\_mask\_uint32xm1\_uint64xm2 (C function), 236  
 vnclipuvi\_mask\_uint32xm2\_uint64xm4 (C function), 236  
 vnclipuvi\_mask\_uint32xm4\_uint64xm8 (C function), 236  
 vnclipuvi\_mask\_uint8xm1\_uint16xm2 (C function), 236  
 vnclipuvi\_mask\_uint8xm2\_uint16xm4 (C function), 236  
 vnclipuvi\_mask\_uint8xm4\_uint16xm8 (C function), 236  
 vnclipuvi\_uint16xm1\_uint32xm2 (C function), 235  
 vnclipuvi\_uint16xm2\_uint32xm4 (C function), 235  
 vnclipuvi\_uint16xm4\_uint32xm8 (C function), 235  
 vnclipuvi\_uint32xm1\_uint64xm2 (C function), 235  
 vnclipuvi\_uint32xm2\_uint64xm4 (C function), 235  
 vnclipuvi\_uint32xm4\_uint64xm8 (C function), 235  
 vnclipuvi\_uint8xm1\_uint16xm2 (C function), 235  
 vnclipuvi\_uint8xm2\_uint16xm4 (C function), 235  
 vnclipuvi\_uint8xm4\_uint16xm8 (C function), 235  
 vnclipuvv\_mask\_uint16xm1\_uint32xm2 (C function), 237  
 vnclipuvv\_mask\_uint16xm2\_uint32xm4 (C function), 237  
 vnclipuvv\_mask\_uint16xm4\_uint32xm8 (C function), 237  
 vnclipuvv\_mask\_uint32xm1\_uint64xm2 (C function), 237  
 vnclipuvv\_mask\_uint32xm2\_uint64xm4 (C function), 237  
 vnclipuvv\_mask\_uint32xm4\_uint64xm8 (C function), 237  
 vnclipuvv\_mask\_uint8xm1\_uint16xm2 (C function), 237  
 vnclipuvv\_mask\_uint8xm2\_uint16xm4 (C function), 237  
 vnclipuvv\_mask\_uint8xm4\_uint16xm8 (C function), 237  
 vnclipuvx\_mask\_uint16xm1\_uint32xm2 (C function), 238  
 vnclipuvx\_mask\_uint16xm2\_uint32xm4 (C function), 238  
 vnclipuvx\_mask\_uint16xm4\_uint32xm8 (C function), 238  
 vnclipuvx\_mask\_uint32xm1\_uint64xm2 (C function), 238  
 vnclipuvx\_mask\_uint32xm2\_uint64xm4 (C function), 238  
 vnclipuvx\_mask\_uint32xm4\_uint64xm8 (C function), 238  
 vnclipuvx\_mask\_uint8xm1\_uint16xm2 (C function), 238  
 vnclipuvx\_mask\_uint8xm2\_uint16xm4 (C function), 238  
 vnclipuvx\_mask\_uint8xm4\_uint16xm8 (C function), 238  
 vnclipuvx\_uint16xm1\_uint32xm2 (C function), 237  
 vnclipuvx\_uint16xm2\_uint32xm4 (C function), 238  
 vnclipuvx\_uint16xm4\_uint32xm8 (C function), 238  
 vnclipuvx\_uint32xm1\_uint64xm2 (C function), 238  
 vnclipuvx\_uint32xm2\_uint64xm4 (C function), 238  
 vnclipuvx\_uint32xm4\_uint64xm8 (C function), 238  
 vnclipuvx\_uint8xm1\_uint16xm2 (C function), 238  
 vnclipuvx\_uint8xm2\_uint16xm4 (C function), 238  
 vnclipuvx\_uint8xm4\_uint16xm8 (C function), 238  
 vnclipvi\_int16xm1\_int32xm2 (C function), 231  
 vnclipvi\_int16xm2\_int32xm4 (C function), 231  
 vnclipvi\_int16xm4\_int32xm8 (C function), 231  
 vnclipvi\_int32xm1\_int64xm2 (C function), 231  
 vnclipvi\_int32xm2\_int64xm4 (C function), 231  
 vnclipvi\_int32xm4\_int64xm8 (C function), 231  
 vnclipvi\_int8xm1\_int16xm2 (C function), 231  
 vnclipvi\_int8xm2\_int16xm4 (C function), 231  
 vnclipvi\_int8xm4\_int16xm8 (C function), 231  
 vnclipvi\_mask\_int16xm1\_int32xm2 (C function), 232  
 vnclipvi\_mask\_int16xm2\_int32xm4 (C function), 232  
 vnclipvi\_mask\_int16xm4\_int32xm8 (C function), 232  
 vnclipvi\_mask\_int32xm1\_int64xm2 (C function), 232  
 vnclipvi\_mask\_int32xm2\_int64xm4 (C function), 232  
 vnclipvi\_mask\_int32xm4\_int64xm8 (C function), 232  
 vnclipvi\_mask\_int8xm1\_int16xm2 (C function), 232  
 vnclipvi\_mask\_int8xm2\_int16xm4 (C function), 232  
 vnclipvi\_mask\_int8xm4\_int16xm8 (C function), 232

vnclipvv\_int16xm1\_int32xm2\_uint16xm1 (C function), 232  
 vnclipvv\_int16xm2\_int32xm4\_uint16xm2 (C function), 232  
 vnclipvv\_int16xm4\_int32xm8\_uint16xm4 (C function), 232  
 vnclipvv\_int32xm1\_int64xm2\_uint32xm1 (C function), 232  
 vnclipvv\_int32xm2\_int64xm4\_uint32xm2 (C function), 232  
 vnclipvv\_int32xm4\_int64xm8\_uint32xm4 (C function), 233  
 vnclipvv\_int8xm1\_int16xm2\_uint8xm1 (C function), 233  
 vnclipvv\_int8xm2\_int16xm4\_uint8xm2 (C function), 233  
 vnclipvv\_int8xm4\_int16xm8\_uint8xm4 (C function), 233  
 vnclipvv\_mask\_int16xm1\_int32xm2\_uint16xm1 (C function), 233  
 vnclipvv\_mask\_int16xm2\_int32xm4\_uint16xm2 (C function), 233  
 vnclipvv\_mask\_int16xm4\_int32xm8\_uint16xm4 (C function), 233  
 vnclipvv\_mask\_int32xm1\_int64xm2\_uint32xm1 (C function), 233  
 vnclipvv\_mask\_int32xm2\_int64xm4\_uint32xm2 (C function), 233  
 vnclipvv\_mask\_int32xm4\_int64xm8\_uint32xm4 (C function), 233  
 vnclipvv\_mask\_int8xm1\_int16xm2\_uint8xm1 (C function), 233  
 vnclipvv\_mask\_int8xm2\_int16xm4\_uint8xm2 (C function), 233  
 vnclipvv\_mask\_int8xm4\_int16xm8\_uint8xm4 (C function), 233  
 vnclipvx\_int16xm1\_int32xm2 (C function), 234  
 vnclipvx\_int16xm2\_int32xm4 (C function), 234  
 vnclipvx\_int16xm4\_int32xm8 (C function), 234  
 vnclipvx\_int32xm1\_int64xm2 (C function), 234  
 vnclipvx\_int32xm2\_int64xm4 (C function), 234  
 vnclipvx\_int32xm4\_int64xm8 (C function), 234  
 vnclipvx\_int8xm1\_int16xm2 (C function), 234  
 vnclipvx\_int8xm2\_int16xm4 (C function), 234  
 vnclipvx\_int8xm4\_int16xm8 (C function), 234  
 vnclipvx\_mask\_int16xm1\_int32xm2 (C function), 234  
 vnclipvx\_mask\_int16xm2\_int32xm4 (C function), 234  
 vnclipvx\_mask\_int16xm4\_int32xm8 (C function), 234  
 vnclipvx\_mask\_int32xm1\_int64xm2 (C function), 234  
 vnclipvx\_mask\_int32xm2\_int64xm4 (C function), 234  
 vnclipvx\_mask\_int32xm4\_int64xm8 (C function), 234  
 vnclipvx\_mask\_int8xm1\_int16xm2 (C function), 234  
 vnclipvx\_mask\_int8xm2\_int16xm4 (C function), 234  
 vnclipvx\_mask\_int8xm4\_int16xm8 (C function), 234  
 vnmsacvv\_int16xm1 (C function), 239  
 vnmsacvv\_int16xm2 (C function), 239  
 vnmsacvv\_int16xm4 (C function), 239  
 vnmsacvv\_int16xm8 (C function), 239  
 vnmsacvv\_int32xm1 (C function), 239  
 vnmsacvv\_int32xm2 (C function), 239  
 vnmsacvv\_int32xm4 (C function), 239  
 vnmsacvv\_int32xm8 (C function), 239  
 vnmsacvv\_int64xm1 (C function), 239  
 vnmsacvv\_int64xm2 (C function), 239  
 vnmsacvv\_int64xm4 (C function), 239  
 vnmsacvv\_int64xm8 (C function), 239  
 vnmsacvv\_int8xm1 (C function), 239  
 vnmsacvv\_int8xm2 (C function), 239  
 vnmsacvv\_int8xm4 (C function), 239  
 vnmsacvv\_int8xm8 (C function), 239  
 vnmsacvv\_mask\_int16xm1 (C function), 240  
 vnmsacvv\_mask\_int16xm2 (C function), 240  
 vnmsacvv\_mask\_int16xm4 (C function), 240  
 vnmsacvv\_mask\_int32xm1 (C function), 240  
 vnmsacvv\_mask\_int32xm2 (C function), 240  
 vnmsacvv\_mask\_int32xm4 (C function), 240  
 vnmsacvv\_mask\_int64xm1 (C function), 240  
 vnmsacvv\_mask\_int64xm2 (C function), 240  
 vnmsacvv\_mask\_int64xm4 (C function), 240  
 vnmsacvv\_mask\_int8xm1 (C function), 240  
 vnmsacvv\_mask\_int8xm2 (C function), 240  
 vnmsacvv\_mask\_int8xm4 (C function), 240  
 vnmsacvv\_mask\_uint16xm1 (C function), 240  
 vnmsacvv\_mask\_uint16xm2 (C function), 241  
 vnmsacvv\_mask\_uint16xm4 (C function), 241  
 vnmsacvv\_mask\_uint32xm1 (C function), 241  
 vnmsacvv\_mask\_uint32xm2 (C function), 241  
 vnmsacvv\_mask\_uint32xm4 (C function), 241  
 vnmsacvv\_mask\_uint64xm1 (C function), 241  
 vnmsacvv\_mask\_uint64xm2 (C function), 241  
 vnmsacvv\_mask\_uint64xm4 (C function), 241  
 vnmsacvv\_mask\_uint8xm1 (C function), 241  
 vnmsacvv\_mask\_uint8xm2 (C function), 241  
 vnmsacvv\_mask\_uint8xm4 (C function), 241  
 vnmsacvv\_uint16xm1 (C function), 239  
 vnmsacvv\_uint16xm2 (C function), 239  
 vnmsacvv\_uint16xm4 (C function), 239  
 vnmsacvv\_uint16xm8 (C function), 239  
 vnmsacvv\_uint32xm1 (C function), 239  
 vnmsacvv\_uint32xm2 (C function), 239  
 vnmsacvv\_uint32xm4 (C function), 239  
 vnmsacvv\_uint32xm8 (C function), 240  
 vnmsacvv\_uint64xm1 (C function), 240  
 vnmsacvv\_uint64xm2 (C function), 240  
 vnmsacvv\_uint64xm4 (C function), 240  
 vnmsacvv\_uint64xm8 (C function), 240  
 vnmsacvv\_uint8xm1 (C function), 240  
 vnmsacvv\_uint8xm2 (C function), 240





vnmsubvv\_uint64xm4 (C function), 245  
 vnmsubvv\_uint64xm8 (C function), 245  
 vnmsubvv\_uint8xm1 (C function), 245  
 vnmsubvv\_uint8xm2 (C function), 245  
 vnmsubvv\_uint8xm4 (C function), 245  
 vnmsubvv\_uint8xm8 (C function), 245  
 vnmsubvx\_int16xm1 (C function), 246  
 vnmsubvx\_int16xm2 (C function), 246  
 vnmsubvx\_int16xm4 (C function), 246  
 vnmsubvx\_int16xm8 (C function), 246  
 vnmsubvx\_int32xm1 (C function), 246  
 vnmsubvx\_int32xm2 (C function), 246  
 vnmsubvx\_int32xm4 (C function), 246  
 vnmsubvx\_int32xm8 (C function), 246  
 vnmsubvx\_int64xm1 (C function), 246  
 vnmsubvx\_int64xm2 (C function), 247  
 vnmsubvx\_int64xm4 (C function), 247  
 vnmsubvx\_int64xm8 (C function), 247  
 vnmsubvx\_int8xm1 (C function), 247  
 vnmsubvx\_int8xm2 (C function), 247  
 vnmsubvx\_int8xm4 (C function), 247  
 vnmsubvx\_int8xm8 (C function), 247  
 vnmsubvx\_mask\_int16xm1 (C function), 247  
 vnmsubvx\_mask\_int16xm2 (C function), 248  
 vnmsubvx\_mask\_int16xm4 (C function), 248  
 vnmsubvx\_mask\_int32xm1 (C function), 248  
 vnmsubvx\_mask\_int32xm2 (C function), 248  
 vnmsubvx\_mask\_int32xm4 (C function), 248  
 vnmsubvx\_mask\_int32xm8 (C function), 248  
 vnmsubvx\_mask\_int64xm1 (C function), 248  
 vnmsubvx\_mask\_int64xm2 (C function), 248  
 vnmsubvx\_mask\_int64xm4 (C function), 248  
 vnmsubvx\_mask\_int8xm1 (C function), 248  
 vnmsubvx\_mask\_int8xm2 (C function), 248  
 vnmsubvx\_mask\_int8xm4 (C function), 248  
 vnmsubvx\_mask\_uint16xm1 (C function), 248  
 vnmsubvx\_mask\_uint16xm2 (C function), 248  
 vnmsubvx\_mask\_uint16xm4 (C function), 248  
 vnmsubvx\_mask\_uint32xm1 (C function), 248  
 vnmsubvx\_mask\_uint32xm2 (C function), 248  
 vnmsubvx\_mask\_uint32xm4 (C function), 248  
 vnmsubvx\_mask\_uint32xm8 (C function), 248  
 vnmsubvx\_mask\_uint64xm1 (C function), 248  
 vnmsubvx\_mask\_uint64xm2 (C function), 248  
 vnmsubvx\_mask\_uint64xm4 (C function), 248  
 vnmsubvx\_mask\_uint8xm1 (C function), 248  
 vnmsubvx\_mask\_uint8xm2 (C function), 248  
 vnmsubvx\_mask\_uint8xm4 (C function), 248  
 vnmsubvx\_uint16xm1 (C function), 247  
 vnmsubvx\_uint16xm2 (C function), 247  
 vnmsubvx\_uint16xm4 (C function), 247  
 vnmsubvx\_uint16xm8 (C function), 247  
 vnmsubvx\_uint32xm1 (C function), 247  
 vnmsubvx\_uint32xm2 (C function), 247  
 vnmsubvx\_uint32xm4 (C function), 247  
 vnmsubvx\_uint32xm8 (C function), 247  
 vnmsubvx\_uint64xm1 (C function), 247  
 vnmsubvx\_uint64xm2 (C function), 247  
 vnmsubvx\_uint64xm4 (C function), 247  
 vnmsubvx\_uint64xm8 (C function), 247  
 vnmsubvx\_uint8xm1 (C function), 247  
 vnmsubvx\_uint8xm2 (C function), 247  
 vnmsubvx\_uint8xm4 (C function), 247  
 vnmsubvx\_uint8xm8 (C function), 247  
 vnotv\_int16xm1 (C function), 13  
 vnotv\_int16xm2 (C function), 14  
 vnotv\_int16xm4 (C function), 14  
 vnotv\_int16xm8 (C function), 14  
 vnotv\_int32xm1 (C function), 14  
 vnotv\_int32xm2 (C function), 14  
 vnotv\_int32xm4 (C function), 14  
 vnotv\_int32xm8 (C function), 14  
 vnotv\_int64xm1 (C function), 14  
 vnotv\_int64xm2 (C function), 14  
 vnotv\_int64xm4 (C function), 14  
 vnotv\_int64xm8 (C function), 14  
 vnotv\_int8xm1 (C function), 14  
 vnotv\_int8xm2 (C function), 14  
 vnotv\_int8xm4 (C function), 14  
 vnotv\_int8xm8 (C function), 14  
 vnotv\_mask\_int16xm1 (C function), 15  
 vnotv\_mask\_int16xm2 (C function), 15  
 vnotv\_mask\_int16xm4 (C function), 15  
 vnotv\_mask\_int16xm8 (C function), 15  
 vnotv\_mask\_int32xm1 (C function), 15  
 vnotv\_mask\_int32xm2 (C function), 15  
 vnotv\_mask\_int32xm4 (C function), 15  
 vnotv\_mask\_int32xm8 (C function), 15  
 vnotv\_mask\_int64xm1 (C function), 15  
 vnotv\_mask\_int64xm2 (C function), 15  
 vnotv\_mask\_int64xm4 (C function), 15  
 vnotv\_mask\_int64xm8 (C function), 15  
 vnotv\_mask\_int8xm1 (C function), 15  
 vnotv\_mask\_int8xm2 (C function), 15  
 vnotv\_mask\_int8xm4 (C function), 15  
 vnotv\_mask\_int8xm8 (C function), 15  
 vnotv\_mask\_uint16xm1 (C function), 15  
 vnotv\_mask\_uint16xm2 (C function), 15  
 vnotv\_mask\_uint16xm4 (C function), 15  
 vnotv\_mask\_uint16xm8 (C function), 15  
 vnotv\_mask\_uint32xm1 (C function), 15  
 vnotv\_mask\_uint32xm2 (C function), 15  
 vnotv\_mask\_uint32xm4 (C function), 15  
 vnotv\_mask\_uint32xm8 (C function), 15  
 vnotv\_mask\_uint64xm1 (C function), 15  
 vnotv\_mask\_uint64xm2 (C function), 16  
 vnotv\_mask\_uint64xm4 (C function), 16  
 vnotv\_mask\_uint64xm8 (C function), 16  
 vnotv\_mask\_uint8xm1 (C function), 16  
 vnotv\_mask\_uint8xm2 (C function), 16



vnotv\_mask\_uint8xm4 (C function), 16  
 vnotv\_mask\_uint8xm8 (C function), 16  
 vnotv\_uint16xm1 (C function), 14  
 vnotv\_uint16xm2 (C function), 14  
 vnotv\_uint16xm4 (C function), 14  
 vnotv\_uint16xm8 (C function), 14  
 vnotv\_uint32xm1 (C function), 14  
 vnotv\_uint32xm2 (C function), 14  
 vnotv\_uint32xm4 (C function), 14  
 vnotv\_uint32xm8 (C function), 14  
 vnotv\_uint64xm1 (C function), 14  
 vnotv\_uint64xm2 (C function), 14  
 vnotv\_uint64xm4 (C function), 14  
 vnotv\_uint64xm8 (C function), 14  
 vnotv\_uint8xm1 (C function), 14  
 vnotv\_uint8xm2 (C function), 14  
 vnotv\_uint8xm4 (C function), 14  
 vnotv\_uint8xm8 (C function), 14  
 vnsravi\_int16xm1\_int32xm2 (C function), 249  
 vnsravi\_int16xm2\_int32xm4 (C function), 249  
 vnsravi\_int16xm4\_int32xm8 (C function), 249  
 vnsravi\_int32xm1\_int64xm2 (C function), 249  
 vnsravi\_int32xm2\_int64xm4 (C function), 249  
 vnsravi\_int32xm4\_int64xm8 (C function), 249  
 vnsravi\_int8xm1\_int16xm2 (C function), 249  
 vnsravi\_int8xm2\_int16xm4 (C function), 249  
 vnsravi\_int8xm4\_int16xm8 (C function), 249  
 vnsravi\_mask\_int16xm1\_int32xm2 (C function), 249  
 vnsravi\_mask\_int16xm2\_int32xm4 (C function), 249  
 vnsravi\_mask\_int16xm4\_int32xm8 (C function), 249  
 vnsravi\_mask\_int32xm1\_int64xm2 (C function), 249  
 vnsravi\_mask\_int32xm2\_int64xm4 (C function), 249  
 vnsravi\_mask\_int32xm4\_int64xm8 (C function), 249  
 vnsravi\_mask\_int8xm1\_int16xm2 (C function), 249  
 vnsravi\_mask\_int8xm2\_int16xm4 (C function), 250  
 vnsravi\_mask\_int8xm4\_int16xm8 (C function), 250  
 vnsravv\_int16xm1\_int32xm2\_uint16xm1 (C function), 250  
 vnsravv\_int16xm2\_int32xm4\_uint16xm2 (C function), 250  
 vnsravv\_int16xm4\_int32xm8\_uint16xm4 (C function), 250  
 vnsravv\_int32xm1\_int64xm2\_uint32xm1 (C function), 250  
 vnsravv\_int32xm2\_int64xm4\_uint32xm2 (C function), 250  
 vnsravv\_int32xm4\_int64xm8\_uint32xm4 (C function), 250  
 vnsravv\_int8xm1\_int16xm2\_uint8xm1 (C function), 250  
 vnsravv\_int8xm2\_int16xm4\_uint8xm2 (C function), 250  
 vnsravv\_int8xm4\_int16xm8\_uint8xm4 (C function), 250  
 vnsravv\_mask\_int16xm1\_int32xm2\_uint16xm1 (C function), 250  
 vnsravv\_mask\_int16xm2\_int32xm4\_uint16xm2 (C function), 251  
 vnsravv\_mask\_int16xm4\_int32xm8\_uint16xm4 (C function), 251  
 vnsravv\_mask\_int32xm1\_int64xm2\_uint32xm1 (C function), 251  
 vnsravv\_mask\_int32xm2\_int64xm4\_uint32xm2 (C function), 251  
 vnsravv\_mask\_int32xm4\_int64xm8\_uint32xm4 (C function), 251  
 vnsravv\_mask\_int8xm1\_int16xm2\_uint8xm1 (C function), 251  
 vnsravv\_mask\_int8xm2\_int16xm4\_uint8xm2 (C function), 251  
 vnsravv\_mask\_int8xm4\_int16xm8\_uint8xm4 (C function), 251  
 vnsravx\_int16xm1\_int32xm2 (C function), 251  
 vnsravx\_int16xm2\_int32xm4 (C function), 251  
 vnsravx\_int16xm4\_int32xm8 (C function), 251  
 vnsravx\_int32xm1\_int64xm2 (C function), 251  
 vnsravx\_int32xm2\_int64xm4 (C function), 251  
 vnsravx\_int32xm4\_int64xm8 (C function), 252  
 vnsravx\_int8xm1\_int16xm2 (C function), 252  
 vnsravx\_int8xm2\_int16xm4 (C function), 252  
 vnsravx\_int8xm4\_int16xm8 (C function), 252  
 vnsravx\_mask\_int16xm1\_int32xm2 (C function), 252  
 vnsravx\_mask\_int16xm2\_int32xm4 (C function), 252  
 vnsravx\_mask\_int16xm4\_int32xm8 (C function), 252  
 vnsravx\_mask\_int32xm1\_int64xm2 (C function), 252  
 vnsravx\_mask\_int32xm2\_int64xm4 (C function), 252  
 vnsravx\_mask\_int32xm4\_int64xm8 (C function), 252  
 vnsravx\_mask\_int8xm1\_int16xm2 (C function), 252  
 vnsravx\_mask\_int8xm2\_int16xm4 (C function), 252  
 vnsravx\_mask\_int8xm4\_int16xm8 (C function), 252  
 vnsrlvi\_mask\_uint16xm1\_uint32xm2 (C function), 253  
 vnsrlvi\_mask\_uint16xm2\_uint32xm4 (C function), 253  
 vnsrlvi\_mask\_uint16xm4\_uint32xm8 (C function), 253  
 vnsrlvi\_mask\_uint32xm1\_uint64xm2 (C function), 253  
 vnsrlvi\_mask\_uint32xm2\_uint64xm4 (C function), 253  
 vnsrlvi\_mask\_uint32xm4\_uint64xm8 (C function), 253  
 vnsrlvi\_mask\_uint8xm1\_int16xm2 (C function), 253  
 vnsrlvi\_mask\_uint8xm2\_int16xm4 (C function), 253  
 vnsrlvi\_mask\_uint8xm4\_int16xm8 (C function), 253  
 vnsrlvi\_uint16xm1\_uint32xm2 (C function), 252  
 vnsrlvi\_uint16xm2\_uint32xm4 (C function), 252  
 vnsrlvi\_uint16xm4\_uint32xm8 (C function), 253  
 vnsrlvi\_uint32xm1\_uint64xm2 (C function), 253  
 vnsrlvi\_uint32xm2\_uint64xm4 (C function), 253  
 vnsrlvi\_uint32xm4\_uint64xm8 (C function), 253  
 vnsrlvi\_uint8xm1\_int16xm2 (C function), 253  
 vnsrlvi\_uint8xm2\_int16xm4 (C function), 253  
 vnsrlvi\_uint8xm4\_int16xm8 (C function), 253  
 vnsrlvv\_mask\_uint16xm1\_uint32xm2 (C function), 254  
 vnsrlvv\_mask\_uint16xm2\_uint32xm4 (C function), 254

vnsrlvv\_mask\_uint16xm4\_uint32xm8 (C function), 254  
 vnsrlvv\_mask\_uint32xm1\_uint64xm2 (C function), 254  
 vnsrlvv\_mask\_uint32xm2\_uint64xm4 (C function), 254  
 vnsrlvv\_mask\_uint32xm4\_uint64xm8 (C function), 254  
 vnsrlvv\_mask\_uint8xm1\_uint16xm2 (C function), 255  
 vnsrlvv\_mask\_uint8xm2\_uint16xm4 (C function), 255  
 vnsrlvv\_mask\_uint8xm4\_uint16xm8 (C function), 255  
 vnsrlvv\_uint16xm1\_uint32xm2 (C function), 254  
 vnsrlvv\_uint16xm2\_uint32xm4 (C function), 254  
 vnsrlvv\_uint16xm4\_uint32xm8 (C function), 254  
 vnsrlvv\_uint32xm1\_uint64xm2 (C function), 254  
 vnsrlvv\_uint32xm2\_uint64xm4 (C function), 254  
 vnsrlvv\_uint32xm4\_uint64xm8 (C function), 254  
 vnsrlvv\_uint8xm1\_uint16xm2 (C function), 254  
 vnsrlvv\_uint8xm2\_uint16xm4 (C function), 254  
 vnsrlvv\_uint8xm4\_uint16xm8 (C function), 254  
 vnsrlvx\_mask\_uint16xm1\_uint32xm2 (C function), 255  
 vnsrlvx\_mask\_uint16xm2\_uint32xm4 (C function), 255  
 vnsrlvx\_mask\_uint16xm4\_uint32xm8 (C function), 256  
 vnsrlvx\_mask\_uint32xm1\_uint64xm2 (C function), 256  
 vnsrlvx\_mask\_uint32xm2\_uint64xm4 (C function), 256  
 vnsrlvx\_mask\_uint32xm4\_uint64xm8 (C function), 256  
 vnsrlvx\_mask\_uint8xm1\_uint16xm2 (C function), 256  
 vnsrlvx\_mask\_uint8xm2\_uint16xm4 (C function), 256  
 vnsrlvx\_mask\_uint8xm4\_uint16xm8 (C function), 256  
 vnsrlvx\_uint16xm1\_uint32xm2 (C function), 255  
 vnsrlvx\_uint16xm2\_uint32xm4 (C function), 255  
 vnsrlvx\_uint16xm4\_uint32xm8 (C function), 255  
 vnsrlvx\_uint32xm1\_uint64xm2 (C function), 255  
 vnsrlvx\_uint32xm2\_uint64xm4 (C function), 255  
 vnsrlvx\_uint32xm4\_uint64xm8 (C function), 255  
 vnsrlvx\_uint8xm1\_uint16xm2 (C function), 255  
 vnsrlvx\_uint8xm2\_uint16xm4 (C function), 255  
 vnsrlvx\_uint8xm4\_uint16xm8 (C function), 255  
 vorvi\_int16xm1 (C function), 16  
 vorvi\_int16xm2 (C function), 16  
 vorvi\_int16xm4 (C function), 16  
 vorvi\_int16xm8 (C function), 16  
 vorvi\_int32xm1 (C function), 16  
 vorvi\_int32xm2 (C function), 16  
 vorvi\_int32xm4 (C function), 16  
 vorvi\_int32xm8 (C function), 16  
 vorvi\_int64xm1 (C function), 16  
 vorvi\_int64xm2 (C function), 16  
 vorvi\_int64xm4 (C function), 16  
 vorvi\_int64xm8 (C function), 16  
 vorvi\_int8xm1 (C function), 16  
 vorvi\_int8xm2 (C function), 16  
 vorvi\_int8xm4 (C function), 16  
 vorvi\_int8xm8 (C function), 17  
 vorvi\_mask\_int16xm1 (C function), 17  
 vorvi\_mask\_int16xm2 (C function), 17  
 vorvi\_mask\_int16xm4 (C function), 17  
 vorvi\_mask\_int16xm8 (C function), 17  
 vorvi\_mask\_int32xm1 (C function), 17  
 vorvi\_mask\_int32xm2 (C function), 17  
 vorvi\_mask\_int32xm4 (C function), 17  
 vorvi\_mask\_int32xm8 (C function), 17  
 vorvi\_mask\_int64xm1 (C function), 17  
 vorvi\_mask\_int64xm2 (C function), 18  
 vorvi\_mask\_int64xm4 (C function), 18  
 vorvi\_mask\_int64xm8 (C function), 18  
 vorvi\_mask\_int8xm1 (C function), 18  
 vorvi\_mask\_int8xm2 (C function), 18  
 vorvi\_mask\_int8xm4 (C function), 18  
 vorvi\_mask\_int8xm8 (C function), 18  
 vorvi\_mask\_uint16xm1 (C function), 18  
 vorvi\_mask\_uint16xm2 (C function), 18  
 vorvi\_mask\_uint16xm4 (C function), 18  
 vorvi\_mask\_uint16xm8 (C function), 18  
 vorvi\_mask\_uint32xm1 (C function), 18  
 vorvi\_mask\_uint32xm2 (C function), 18  
 vorvi\_mask\_uint32xm4 (C function), 18  
 vorvi\_mask\_uint32xm8 (C function), 18  
 vorvi\_mask\_uint64xm1 (C function), 18  
 vorvi\_mask\_uint64xm2 (C function), 18  
 vorvi\_mask\_uint64xm4 (C function), 18  
 vorvi\_mask\_uint64xm8 (C function), 18  
 vorvi\_mask\_uint8xm1 (C function), 18  
 vorvi\_mask\_uint8xm2 (C function), 18  
 vorvi\_mask\_uint8xm4 (C function), 18  
 vorvi\_mask\_uint8xm8 (C function), 18  
 vorvi\_uint16xm1 (C function), 17  
 vorvi\_uint16xm2 (C function), 17  
 vorvi\_uint16xm4 (C function), 17  
 vorvi\_uint16xm8 (C function), 17  
 vorvi\_uint32xm1 (C function), 17  
 vorvi\_uint32xm2 (C function), 17  
 vorvi\_uint32xm4 (C function), 17  
 vorvi\_uint32xm8 (C function), 17  
 vorvi\_uint64xm1 (C function), 17  
 vorvi\_uint64xm2 (C function), 17  
 vorvi\_uint64xm4 (C function), 17  
 vorvi\_uint64xm8 (C function), 17  
 vorvi\_uint8xm1 (C function), 17  
 vorvi\_uint8xm2 (C function), 17  
 vorvi\_uint8xm4 (C function), 17  
 vorvi\_uint8xm8 (C function), 17  
 vorvv\_int16xm1 (C function), 19  
 vorvv\_int16xm2 (C function), 19  
 vorvv\_int16xm4 (C function), 19  
 vorvv\_int16xm8 (C function), 19  
 vorvv\_int32xm1 (C function), 19  
 vorvv\_int32xm2 (C function), 19  
 vorvv\_int32xm4 (C function), 19  
 vorvv\_int32xm8 (C function), 19  
 vorvv\_int64xm1 (C function), 19  
 vorvv\_int64xm2 (C function), 19

vorvv\_int64xm4 (C function), 19  
vorvv\_int64xm8 (C function), 19  
vorvv\_int8xm1 (C function), 19  
vorvv\_int8xm2 (C function), 19  
vorvv\_int8xm4 (C function), 19  
vorvv\_int8xm8 (C function), 19  
vorvv\_mask\_int16xm1 (C function), 20  
vorvv\_mask\_int16xm2 (C function), 20  
vorvv\_mask\_int16xm4 (C function), 20  
vorvv\_mask\_int16xm8 (C function), 20  
vorvv\_mask\_int32xm1 (C function), 20  
vorvv\_mask\_int32xm2 (C function), 20  
vorvv\_mask\_int32xm4 (C function), 20  
vorvv\_mask\_int32xm8 (C function), 20  
vorvv\_mask\_int64xm1 (C function), 20  
vorvv\_mask\_int64xm2 (C function), 20  
vorvv\_mask\_int64xm4 (C function), 20  
vorvv\_mask\_int64xm8 (C function), 20  
vorvv\_mask\_int8xm1 (C function), 20  
vorvv\_mask\_int8xm2 (C function), 20  
vorvv\_mask\_int8xm4 (C function), 20  
vorvv\_mask\_int8xm8 (C function), 20  
vorvv\_mask\_uint16xm1 (C function), 21  
vorvv\_mask\_uint16xm2 (C function), 21  
vorvv\_mask\_uint16xm4 (C function), 21  
vorvv\_mask\_uint16xm8 (C function), 21  
vorvv\_mask\_uint32xm1 (C function), 21  
vorvv\_mask\_uint32xm2 (C function), 21  
vorvv\_mask\_uint32xm4 (C function), 21  
vorvv\_mask\_uint32xm8 (C function), 21  
vorvv\_mask\_uint64xm1 (C function), 21  
vorvv\_mask\_uint64xm2 (C function), 21  
vorvv\_mask\_uint64xm4 (C function), 21  
vorvv\_mask\_uint64xm8 (C function), 21  
vorvv\_mask\_uint8xm1 (C function), 21  
vorvv\_mask\_uint8xm2 (C function), 21  
vorvv\_mask\_uint8xm4 (C function), 21  
vorvv\_mask\_uint8xm8 (C function), 21  
vorvv\_uint16xm1 (C function), 19  
vorvv\_uint16xm2 (C function), 19  
vorvv\_uint16xm4 (C function), 19  
vorvv\_uint16xm8 (C function), 19  
vorvv\_uint32xm1 (C function), 19  
vorvv\_uint32xm2 (C function), 19  
vorvv\_uint32xm4 (C function), 19  
vorvv\_uint32xm8 (C function), 19  
vorvv\_uint64xm1 (C function), 19  
vorvv\_uint64xm2 (C function), 19  
vorvv\_uint64xm4 (C function), 20  
vorvv\_uint64xm8 (C function), 20  
vorvv\_uint8xm1 (C function), 20  
vorvv\_uint8xm2 (C function), 20  
vorvv\_uint8xm4 (C function), 20  
vorvv\_uint8xm8 (C function), 20

vorvx\_int16xm1 (C function), 21  
vorvx\_int16xm2 (C function), 22  
vorvx\_int16xm4 (C function), 22  
vorvx\_int16xm8 (C function), 22  
vorvx\_int32xm1 (C function), 22  
vorvx\_int32xm2 (C function), 22  
vorvx\_int32xm4 (C function), 22  
vorvx\_int32xm8 (C function), 22  
vorvx\_int64xm1 (C function), 22  
vorvx\_int64xm2 (C function), 22  
vorvx\_int64xm4 (C function), 22  
vorvx\_int64xm8 (C function), 22  
vorvx\_int8xm1 (C function), 22  
vorvx\_int8xm2 (C function), 22  
vorvx\_int8xm4 (C function), 22  
vorvx\_int8xm8 (C function), 22  
vorvx\_mask\_int16xm1 (C function), 23  
vorvx\_mask\_int16xm2 (C function), 23  
vorvx\_mask\_int16xm4 (C function), 23  
vorvx\_mask\_int16xm8 (C function), 23  
vorvx\_mask\_int32xm1 (C function), 23  
vorvx\_mask\_int32xm2 (C function), 23  
vorvx\_mask\_int32xm4 (C function), 23  
vorvx\_mask\_int32xm8 (C function), 23  
vorvx\_mask\_int64xm1 (C function), 23  
vorvx\_mask\_int64xm2 (C function), 23  
vorvx\_mask\_int64xm4 (C function), 23  
vorvx\_mask\_int64xm8 (C function), 23  
vorvx\_mask\_int8xm1 (C function), 23  
vorvx\_mask\_int8xm2 (C function), 23  
vorvx\_mask\_int8xm4 (C function), 23  
vorvx\_mask\_int8xm8 (C function), 23  
vorvx\_mask\_uint16xm1 (C function), 23  
vorvx\_mask\_uint16xm2 (C function), 23  
vorvx\_mask\_uint16xm4 (C function), 23  
vorvx\_mask\_uint16xm8 (C function), 23  
vorvx\_mask\_uint32xm1 (C function), 23  
vorvx\_mask\_uint32xm2 (C function), 23  
vorvx\_mask\_uint32xm4 (C function), 23  
vorvx\_mask\_uint32xm8 (C function), 24  
vorvx\_mask\_uint64xm1 (C function), 24  
vorvx\_mask\_uint64xm2 (C function), 24  
vorvx\_mask\_uint64xm4 (C function), 24  
vorvx\_mask\_uint64xm8 (C function), 24  
vorvx\_mask\_uint8xm1 (C function), 24  
vorvx\_mask\_uint8xm2 (C function), 24  
vorvx\_mask\_uint8xm4 (C function), 24  
vorvx\_mask\_uint8xm8 (C function), 24  
vorvx\_uint16xm1 (C function), 22  
vorvx\_uint16xm2 (C function), 22  
vorvx\_uint16xm4 (C function), 22  
vorvx\_uint16xm8 (C function), 22  
vorvx\_uint32xm1 (C function), 22  
vorvx\_uint32xm2 (C function), 22





vredmaxvs\_int16xm4 (C function), 259  
 vredmaxvs\_int16xm8 (C function), 259  
 vredmaxvs\_int32xm1 (C function), 259  
 vredmaxvs\_int32xm2 (C function), 259  
 vredmaxvs\_int32xm4 (C function), 259  
 vredmaxvs\_int32xm8 (C function), 259  
 vredmaxvs\_int64xm1 (C function), 259  
 vredmaxvs\_int64xm2 (C function), 259  
 vredmaxvs\_int64xm4 (C function), 259  
 vredmaxvs\_int64xm8 (C function), 259  
 vredmaxvs\_int8xm1 (C function), 259  
 vredmaxvs\_int8xm2 (C function), 259  
 vredmaxvs\_int8xm4 (C function), 259  
 vredmaxvs\_int8xm8 (C function), 259  
 vredmaxvs\_mask\_int16xm1 (C function), 260  
 vredmaxvs\_mask\_int16xm2 (C function), 260  
 vredmaxvs\_mask\_int16xm4 (C function), 260  
 vredmaxvs\_mask\_int16xm8 (C function), 260  
 vredmaxvs\_mask\_int32xm1 (C function), 260  
 vredmaxvs\_mask\_int32xm2 (C function), 260  
 vredmaxvs\_mask\_int32xm4 (C function), 260  
 vredmaxvs\_mask\_int32xm8 (C function), 260  
 vredmaxvs\_mask\_int64xm1 (C function), 260  
 vredmaxvs\_mask\_int64xm2 (C function), 260  
 vredmaxvs\_mask\_int64xm4 (C function), 260  
 vredmaxvs\_mask\_int64xm8 (C function), 260  
 vredmaxvs\_mask\_int8xm1 (C function), 260  
 vredmaxvs\_mask\_int8xm2 (C function), 260  
 vredmaxvs\_mask\_int8xm4 (C function), 260  
 vredmaxvs\_mask\_int8xm8 (C function), 260  
 vredminuvs\_mask\_uint16xm1 (C function), 264  
 vredminuvs\_mask\_uint16xm2 (C function), 264  
 vredminuvs\_mask\_uint16xm4 (C function), 265  
 vredminuvs\_mask\_uint16xm8 (C function), 265  
 vredminuvs\_mask\_uint32xm1 (C function), 265  
 vredminuvs\_mask\_uint32xm2 (C function), 265  
 vredminuvs\_mask\_uint32xm4 (C function), 265  
 vredminuvs\_mask\_uint32xm8 (C function), 265  
 vredminuvs\_mask\_uint64xm1 (C function), 265  
 vredminuvs\_mask\_uint64xm2 (C function), 265  
 vredminuvs\_mask\_uint64xm4 (C function), 265  
 vredminuvs\_mask\_uint64xm8 (C function), 265  
 vredminuvs\_mask\_uint8xm1 (C function), 265  
 vredminuvs\_mask\_uint8xm2 (C function), 265  
 vredminuvs\_mask\_uint8xm4 (C function), 265  
 vredminuvs\_mask\_uint8xm8 (C function), 265  
 vredminuvs\_uint16xm1 (C function), 264  
 vredminuvs\_uint16xm2 (C function), 264  
 vredminuvs\_uint16xm4 (C function), 264  
 vredminuvs\_uint16xm8 (C function), 264  
 vredminuvs\_uint32xm1 (C function), 264  
 vredminuvs\_uint32xm2 (C function), 264  
 vredminuvs\_uint32xm4 (C function), 264  
 vredminuvs\_uint32xm8 (C function), 264  
 vredminuvs\_uint64xm1 (C function), 264  
 vredminuvs\_uint64xm2 (C function), 264  
 vredminuvs\_uint64xm4 (C function), 264  
 vredminuvs\_uint64xm8 (C function), 264  
 vredminuvs\_uint8xm1 (C function), 264  
 vredminuvs\_uint8xm2 (C function), 264  
 vredminuvs\_uint8xm4 (C function), 264  
 vredminuvs\_uint8xm8 (C function), 264  
 vredorvs\_int16xm1 (C function), 265  
 vredorvs\_int16xm2 (C function), 265  
 vredorvs\_int16xm4 (C function), 265  
 vredorvs\_int16xm8 (C function), 266  
 vredorvs\_int32xm1 (C function), 266  
 vredorvs\_int32xm2 (C function), 266  
 vredorvs\_int32xm4 (C function), 266  
 vredorvs\_int32xm8 (C function), 266  
 vredorvs\_int64xm1 (C function), 266  
 vredorvs\_int64xm2 (C function), 266  
 vredorvs\_int64xm4 (C function), 266  
 vredorvs\_int64xm8 (C function), 266  
 vredorvs\_int8xm1 (C function), 266  
 vredorvs\_int8xm2 (C function), 266





vredsumvs\_uint64xm4 (C function), 269  
vredsumvs\_uint64xm8 (C function), 269  
vredsumvs\_uint8xm1 (C function), 269  
vredsumvs\_uint8xm2 (C function), 269  
vredsumvs\_uint8xm4 (C function), 269  
vredsumvs\_uint8xm8 (C function), 269  
vredxorvs\_int16xm1 (C function), 271  
vredxorvs\_int16xm2 (C function), 271  
vredxorvs\_int16xm4 (C function), 271  
vredxorvs\_int16xm8 (C function), 271  
vredxorvs\_int32xm1 (C function), 271  
vredxorvs\_int32xm2 (C function), 271  
vredxorvs\_int32xm4 (C function), 271  
vredxorvs\_int32xm8 (C function), 271  
vredxorvs\_int64xm1 (C function), 271  
vredxorvs\_int64xm2 (C function), 271  
vredxorvs\_int64xm4 (C function), 271  
vredxorvs\_int64xm8 (C function), 271  
vredxorvs\_int8xm1 (C function), 271  
vredxorvs\_int8xm2 (C function), 271  
vredxorvs\_int8xm4 (C function), 271  
vredxorvs\_int8xm8 (C function), 271  
vredxorvs\_mask\_int16xm1 (C function), 272  
vredxorvs\_mask\_int16xm2 (C function), 272  
vredxorvs\_mask\_int16xm4 (C function), 272  
vredxorvs\_mask\_int16xm8 (C function), 272  
vredxorvs\_mask\_int32xm1 (C function), 272  
vredxorvs\_mask\_int32xm2 (C function), 272  
vredxorvs\_mask\_int32xm4 (C function), 272  
vredxorvs\_mask\_int32xm8 (C function), 272  
vredxorvs\_mask\_int64xm1 (C function), 272  
vredxorvs\_mask\_int64xm2 (C function), 272  
vredxorvs\_mask\_int64xm4 (C function), 273  
vredxorvs\_mask\_int64xm8 (C function), 273  
vredxorvs\_mask\_int8xm1 (C function), 273  
vredxorvs\_mask\_int8xm2 (C function), 273  
vredxorvs\_mask\_int8xm4 (C function), 273  
vredxorvs\_mask\_int8xm8 (C function), 273  
vredxorvs\_mask\_uint16xm1 (C function), 273  
vredxorvs\_mask\_uint16xm2 (C function), 273  
vredxorvs\_mask\_uint16xm4 (C function), 273  
vredxorvs\_mask\_uint16xm8 (C function), 273  
vredxorvs\_mask\_uint32xm1 (C function), 273  
vredxorvs\_mask\_uint32xm2 (C function), 273  
vredxorvs\_mask\_uint32xm4 (C function), 273  
vredxorvs\_mask\_uint32xm8 (C function), 273  
vredxorvs\_mask\_uint64xm1 (C function), 273  
vredxorvs\_mask\_uint64xm2 (C function), 273  
vredxorvs\_mask\_uint64xm4 (C function), 273  
vredxorvs\_mask\_uint64xm8 (C function), 273  
vredxorvs\_mask\_uint8xm1 (C function), 273  
vredxorvs\_mask\_uint8xm2 (C function), 273  
vredxorvs\_mask\_uint8xm4 (C function), 273  
vredxorvs\_mask\_uint8xm8 (C function), 273  
vredxorvs\_uint16xm1 (C function), 271  
vredxorvs\_uint16xm2 (C function), 271  
vredxorvs\_uint16xm4 (C function), 272  
vredxorvs\_uint16xm8 (C function), 272  
vredxorvs\_uint32xm1 (C function), 272  
vredxorvs\_uint32xm2 (C function), 272  
vredxorvs\_uint32xm4 (C function), 272  
vredxorvs\_uint32xm8 (C function), 272  
vredxorvs\_uint64xm1 (C function), 272  
vredxorvs\_uint64xm2 (C function), 272  
vredxorvs\_uint64xm4 (C function), 272  
vredxorvs\_uint64xm8 (C function), 272  
vredxorvs\_uint8xm1 (C function), 272  
vredxorvs\_uint8xm2 (C function), 272  
vredxorvs\_uint8xm4 (C function), 272  
vredxorvs\_uint8xm8 (C function), 272  
vremuvv\_mask\_uint16xm1 (C function), 277  
vremuvv\_mask\_uint16xm2 (C function), 278  
vremuvv\_mask\_uint16xm4 (C function), 278  
vremuvv\_mask\_uint16xm8 (C function), 278  
vremuvv\_mask\_uint32xm1 (C function), 278  
vremuvv\_mask\_uint32xm2 (C function), 278  
vremuvv\_mask\_uint32xm4 (C function), 278  
vremuvv\_mask\_uint32xm8 (C function), 278  
vremuvv\_mask\_uint64xm1 (C function), 278  
vremuvv\_mask\_uint64xm2 (C function), 278  
vremuvv\_mask\_uint64xm4 (C function), 278  
vremuvv\_mask\_uint64xm8 (C function), 278  
vremuvv\_mask\_uint8xm1 (C function), 278  
vremuvv\_mask\_uint8xm2 (C function), 278  
vremuvv\_mask\_uint8xm4 (C function), 278  
vremuvv\_mask\_uint8xm8 (C function), 278  
vremuvv\_uint16xm1 (C function), 277  
vremuvv\_uint16xm2 (C function), 277  
vremuvv\_uint16xm4 (C function), 277  
vremuvv\_uint16xm8 (C function), 277  
vremuvv\_uint32xm1 (C function), 277  
vremuvv\_uint32xm2 (C function), 277  
vremuvv\_uint32xm4 (C function), 277  
vremuvv\_uint32xm8 (C function), 277  
vremuvv\_uint64xm1 (C function), 277  
vremuvv\_uint64xm2 (C function), 277  
vremuvv\_uint64xm4 (C function), 277  
vremuvv\_uint64xm8 (C function), 277  
vremuvv\_uint8xm1 (C function), 277  
vremuvv\_uint8xm2 (C function), 277  
vremuvv\_uint8xm4 (C function), 277  
vremuvv\_uint8xm8 (C function), 277  
vremuvx\_mask\_uint16xm1 (C function), 279  
vremuvx\_mask\_uint16xm2 (C function), 279  
vremuvx\_mask\_uint16xm4 (C function), 279  
vremuvx\_mask\_uint16xm8 (C function), 279  
vremuvx\_mask\_uint32xm1 (C function), 279  
vremuvx\_mask\_uint32xm2 (C function), 279

vremuvx\_mask\_uint32xm4 (C function), 279  
 vremuvx\_mask\_uint32xm8 (C function), 279  
 vremuvx\_mask\_uint64xm1 (C function), 279  
 vremuvx\_mask\_uint64xm2 (C function), 279  
 vremuvx\_mask\_uint64xm4 (C function), 279  
 vremuvx\_mask\_uint64xm8 (C function), 279  
 vremuvx\_mask\_uint8xm1 (C function), 280  
 vremuvx\_mask\_uint8xm2 (C function), 280  
 vremuvx\_mask\_uint8xm4 (C function), 280  
 vremuvx\_mask\_uint8xm8 (C function), 280  
 vremuvx\_int16xm1 (C function), 278  
 vremuvx\_int16xm2 (C function), 278  
 vremuvx\_int16xm4 (C function), 278  
 vremuvx\_int16xm8 (C function), 279  
 vremuvx\_int32xm1 (C function), 279  
 vremuvx\_int32xm2 (C function), 279  
 vremuvx\_int32xm4 (C function), 279  
 vremuvx\_int32xm8 (C function), 279  
 vremuvx\_int64xm1 (C function), 279  
 vremuvx\_int64xm2 (C function), 279  
 vremuvx\_int64xm4 (C function), 279  
 vremuvx\_int64xm8 (C function), 279  
 vremuvx\_int8xm1 (C function), 279  
 vremuvx\_int8xm2 (C function), 279  
 vremuvx\_int8xm4 (C function), 279  
 vremuvx\_int8xm8 (C function), 279  
 vremvv\_int16xm1 (C function), 274  
 vremvv\_int16xm2 (C function), 274  
 vremvv\_int16xm4 (C function), 274  
 vremvv\_int16xm8 (C function), 274  
 vremvv\_int32xm1 (C function), 274  
 vremvv\_int32xm2 (C function), 274  
 vremvv\_int32xm4 (C function), 274  
 vremvv\_int32xm8 (C function), 274  
 vremvv\_int64xm1 (C function), 274  
 vremvv\_int64xm2 (C function), 274  
 vremvv\_int64xm4 (C function), 274  
 vremvv\_int64xm8 (C function), 274  
 vremvv\_int8xm1 (C function), 274  
 vremvv\_int8xm2 (C function), 274  
 vremvv\_int8xm4 (C function), 274  
 vremvv\_int8xm8 (C function), 274  
 vremvv\_mask\_int16xm1 (C function), 274  
 vremvv\_mask\_int16xm2 (C function), 274  
 vremvv\_mask\_int16xm4 (C function), 274  
 vremvv\_mask\_int16xm8 (C function), 275  
 vremvv\_mask\_int32xm1 (C function), 275  
 vremvv\_mask\_int32xm2 (C function), 275  
 vremvv\_mask\_int32xm4 (C function), 275  
 vremvv\_mask\_int32xm8 (C function), 275  
 vremvv\_mask\_int64xm1 (C function), 275  
 vremvv\_mask\_int64xm2 (C function), 275  
 vremvv\_mask\_int64xm4 (C function), 275  
 vremvv\_mask\_int64xm8 (C function), 275  
 vremvv\_mask\_int8xm1 (C function), 275  
 vremvv\_mask\_int8xm2 (C function), 275  
 vremvv\_mask\_int8xm4 (C function), 275  
 vremvv\_mask\_int8xm8 (C function), 275  
 vrgatherv\_float16xm1 (C function), 832  
 vrgatherv\_float16xm2 (C function), 832  
 vrgatherv\_float16xm4 (C function), 832  
 vrgatherv\_float16xm8 (C function), 832  
 vrgatherv\_float32xm1 (C function), 832  
 vrgatherv\_float32xm2 (C function), 832  
 vrgatherv\_float32xm4 (C function), 832  
 vrgatherv\_float32xm8 (C function), 832  
 vrgatherv\_float64xm1 (C function), 832  
 vrgatherv\_float64xm2 (C function), 832  
 vrgatherv\_float64xm4 (C function), 832  
 vrgatherv\_float64xm8 (C function), 832  
 vrgatherv\_int16xm1 (C function), 832  
 vrgatherv\_int16xm2 (C function), 832  
 vrgatherv\_int16xm4 (C function), 832  
 vrgatherv\_int16xm8 (C function), 832  
 vrgatherv\_int32xm1 (C function), 832  
 vrgatherv\_int32xm2 (C function), 832

[illegible]

vrgathervv\_mask\_float32xm2\_uint32xm2 (C function), 837  
 vrgathervv\_mask\_float32xm4\_uint32xm4 (C function), 837  
 vrgathervv\_mask\_float32xm8\_uint32xm8 (C function), 837  
 vrgathervv\_mask\_float64xm1\_uint64xm1 (C function), 838  
 vrgathervv\_mask\_float64xm2\_uint64xm2 (C function), 838  
 vrgathervv\_mask\_float64xm4\_uint64xm4 (C function), 838  
 vrgathervv\_mask\_float64xm8\_uint64xm8 (C function), 838  
 vrgathervv\_mask\_int16xm1\_uint16xm1 (C function), 838  
 vrgathervv\_mask\_int16xm2\_uint16xm2 (C function), 838  
 vrgathervv\_mask\_int16xm4\_uint16xm4 (C function), 838  
 vrgathervv\_mask\_int16xm8\_uint16xm8 (C function), 838  
 vrgathervv\_mask\_int32xm1\_uint32xm1 (C function), 838  
 vrgathervv\_mask\_int32xm2\_uint32xm2 (C function), 838  
 vrgathervv\_mask\_int32xm4\_uint32xm4 (C function), 838  
 vrgathervv\_mask\_int32xm8\_uint32xm8 (C function), 838  
 vrgathervv\_mask\_int64xm1\_uint64xm1 (C function), 838  
 vrgathervv\_mask\_int64xm2\_uint64xm2 (C function), 838  
 vrgathervv\_mask\_int64xm4\_uint64xm4 (C function), 838  
 vrgathervv\_mask\_int64xm8\_uint64xm8 (C function), 838  
 vrgathervv\_mask\_int8xm1\_uint8xm1 (C function), 839  
 vrgathervv\_mask\_int8xm2\_uint8xm2 (C function), 839  
 vrgathervv\_mask\_int8xm4\_uint8xm4 (C function), 839  
 vrgathervv\_mask\_int8xm8\_uint8xm8 (C function), 839  
 vrgathervv\_mask\_uint16xm1 (C function), 839  
 vrgathervv\_mask\_uint16xm2 (C function), 839  
 vrgathervv\_mask\_uint16xm4 (C function), 839  
 vrgathervv\_mask\_uint16xm8 (C function), 839  
 vrgathervv\_mask\_uint32xm1 (C function), 839  
 vrgathervv\_mask\_uint32xm2 (C function), 839  
 vrgathervv\_mask\_uint32xm4 (C function), 839  
 vrgathervv\_mask\_uint32xm8 (C function), 839  
 vrgathervv\_mask\_uint64xm1 (C function), 839  
 vrgathervv\_mask\_uint64xm2 (C function), 839  
 vrgathervv\_mask\_uint64xm4 (C function), 839  
 vrgathervv\_mask\_uint64xm8 (C function), 839  
 vrgathervv\_mask\_uint8xm1 (C function), 839  
 vrgathervv\_mask\_uint8xm2 (C function), 839  
 vrgathervv\_mask\_uint8xm4 (C function), 839  
 vrgathervv\_mask\_uint8xm8 (C function), 839  
 vrgathervv\_uint16xm1 (C function), 836  
 vrgathervv\_uint16xm2 (C function), 836  
 vrgathervv\_uint16xm4 (C function), 836  
 vrgathervv\_uint16xm8 (C function), 836  
 vrgathervv\_uint32xm1 (C function), 836  
 vrgathervv\_uint32xm2 (C function), 837  
 vrgathervv\_uint32xm4 (C function), 837  
 vrgathervv\_uint32xm8 (C function), 837  
 vrgathervv\_uint64xm1 (C function), 837  
 vrgathervv\_uint64xm2 (C function), 837  
 vrgathervv\_uint64xm4 (C function), 837  
 vrgathervv\_uint64xm8 (C function), 837  
 vrgathervv\_uint8xm1 (C function), 837  
 vrgathervv\_uint8xm2 (C function), 837  
 vrgathervv\_uint8xm4 (C function), 837  
 vrgathervv\_uint8xm8 (C function), 837  
 vrgathervx\_float16xm1 (C function), 840  
 vrgathervx\_float16xm2 (C function), 840  
 vrgathervx\_float16xm4 (C function), 840  
 vrgathervx\_float16xm8 (C function), 840  
 vrgathervx\_float32xm1 (C function), 840  
 vrgathervx\_float32xm2 (C function), 840  
 vrgathervx\_float32xm4 (C function), 840  
 vrgathervx\_float32xm8 (C function), 840  
 vrgathervx\_float64xm1 (C function), 840  
 vrgathervx\_float64xm2 (C function), 840  
 vrgathervx\_float64xm4 (C function), 840  
 vrgathervx\_float64xm8 (C function), 840  
 vrgathervx\_int16xm1 (C function), 840  
 vrgathervx\_int16xm2 (C function), 840  
 vrgathervx\_int16xm4 (C function), 840  
 vrgathervx\_int16xm8 (C function), 840  
 vrgathervx\_int32xm1 (C function), 840  
 vrgathervx\_int32xm2 (C function), 840  
 vrgathervx\_int32xm4 (C function), 840  
 vrgathervx\_int32xm8 (C function), 840  
 vrgathervx\_int64xm1 (C function), 840  
 vrgathervx\_int64xm2 (C function), 840  
 vrgathervx\_int64xm4 (C function), 840  
 vrgathervx\_int64xm8 (C function), 840  
 vrgathervx\_int8xm1 (C function), 840  
 vrgathervx\_int8xm2 (C function), 840  
 vrgathervx\_int8xm4 (C function), 840  
 vrgathervx\_int8xm8 (C function), 840  
 vrgathervx\_mask\_float16xm1 (C function), 841  
 vrgathervx\_mask\_float16xm2 (C function), 841  
 vrgathervx\_mask\_float16xm4 (C function), 841  
 vrgathervx\_mask\_float16xm8 (C function), 841  
 vrgathervx\_mask\_float32xm1 (C function), 841  
 vrgathervx\_mask\_float32xm2 (C function), 841





vrsubvi\_uint32xm4 (C function), 281  
 vrsubvi\_uint32xm8 (C function), 281  
 vrsubvi\_uint64xm1 (C function), 281  
 vrsubvi\_uint64xm2 (C function), 281  
 vrsubvi\_uint64xm4 (C function), 281  
 vrsubvi\_uint64xm8 (C function), 281  
 vrsubvi\_uint8xm1 (C function), 281  
 vrsubvi\_uint8xm2 (C function), 281  
 vrsubvi\_uint8xm4 (C function), 281  
 vrsubvi\_uint8xm8 (C function), 281  
 vrsubvx\_int16xm1 (C function), 283  
 vrsubvx\_int16xm2 (C function), 283  
 vrsubvx\_int16xm4 (C function), 283  
 vrsubvx\_int16xm8 (C function), 283  
 vrsubvx\_int32xm1 (C function), 283  
 vrsubvx\_int32xm2 (C function), 283  
 vrsubvx\_int32xm4 (C function), 283  
 vrsubvx\_int32xm8 (C function), 283  
 vrsubvx\_int64xm1 (C function), 283  
 vrsubvx\_int64xm2 (C function), 283  
 vrsubvx\_int64xm4 (C function), 283  
 vrsubvx\_int64xm8 (C function), 283  
 vrsubvx\_int8xm1 (C function), 283  
 vrsubvx\_int8xm2 (C function), 283  
 vrsubvx\_int8xm4 (C function), 283  
 vrsubvx\_int8xm8 (C function), 283  
 vrsubvx\_mask\_int16xm1 (C function), 284  
 vrsubvx\_mask\_int16xm2 (C function), 284  
 vrsubvx\_mask\_int16xm4 (C function), 284  
 vrsubvx\_mask\_int16xm8 (C function), 284  
 vrsubvx\_mask\_int32xm1 (C function), 284  
 vrsubvx\_mask\_int32xm2 (C function), 284  
 vrsubvx\_mask\_int32xm4 (C function), 284  
 vrsubvx\_mask\_int32xm8 (C function), 284  
 vrsubvx\_mask\_int64xm1 (C function), 284  
 vrsubvx\_mask\_int64xm2 (C function), 284  
 vrsubvx\_mask\_int64xm4 (C function), 284  
 vrsubvx\_mask\_int64xm8 (C function), 284  
 vrsubvx\_mask\_int8xm1 (C function), 284  
 vrsubvx\_mask\_int8xm2 (C function), 284  
 vrsubvx\_mask\_int8xm4 (C function), 284  
 vrsubvx\_mask\_int8xm8 (C function), 284  
 vrsubvx\_mask\_uint16xm1 (C function), 284  
 vrsubvx\_mask\_uint16xm2 (C function), 284  
 vrsubvx\_mask\_uint16xm4 (C function), 284  
 vrsubvx\_mask\_uint16xm8 (C function), 284  
 vrsubvx\_mask\_uint32xm1 (C function), 284  
 vrsubvx\_mask\_uint32xm2 (C function), 285  
 vrsubvx\_mask\_uint32xm4 (C function), 285  
 vrsubvx\_mask\_uint32xm8 (C function), 285  
 vrsubvx\_mask\_uint64xm1 (C function), 285  
 vrsubvx\_mask\_uint64xm2 (C function), 285  
 vrsubvx\_mask\_uint64xm4 (C function), 285  
 vrsubvx\_mask\_uint64xm8 (C function), 285  
 vrsubvx\_mask\_uint8xm1 (C function), 285  
 vrsubvx\_mask\_uint8xm2 (C function), 285  
 vrsubvx\_mask\_uint8xm4 (C function), 285  
 vrsubvx\_mask\_uint8xm8 (C function), 285  
 vsadduvi\_mask\_uint16xm1 (C function), 290  
 vsadduvi\_mask\_uint16xm2 (C function), 291  
 vsadduvi\_mask\_uint16xm4 (C function), 291  
 vsadduvi\_mask\_uint16xm8 (C function), 291  
 vsadduvi\_mask\_uint32xm1 (C function), 291  
 vsadduvi\_mask\_uint32xm2 (C function), 291  
 vsadduvi\_mask\_uint32xm4 (C function), 291  
 vsadduvi\_mask\_uint32xm8 (C function), 291  
 vsadduvi\_mask\_uint64xm1 (C function), 291  
 vsadduvi\_mask\_uint64xm2 (C function), 291  
 vsadduvi\_mask\_uint64xm4 (C function), 291  
 vsadduvi\_mask\_uint64xm8 (C function), 291  
 vsadduvi\_mask\_uint8xm1 (C function), 291  
 vsadduvi\_mask\_uint8xm2 (C function), 291  
 vsadduvi\_mask\_uint8xm4 (C function), 291  
 vsadduvi\_mask\_uint8xm8 (C function), 291  
 vsadduvi\_uint16xm1 (C function), 290  
 vsadduvi\_uint16xm2 (C function), 290  
 vsadduvi\_uint16xm4 (C function), 290  
 vsadduvi\_uint16xm8 (C function), 290  
 vsadduvi\_uint32xm1 (C function), 290  
 vsadduvi\_uint32xm2 (C function), 290  
 vsadduvi\_uint32xm4 (C function), 290  
 vsadduvi\_uint32xm8 (C function), 290  
 vsadduvi\_uint64xm1 (C function), 290  
 vsadduvi\_uint64xm2 (C function), 290  
 vsadduvi\_uint64xm4 (C function), 290  
 vsadduvi\_uint64xm8 (C function), 290  
 vsadduvi\_uint8xm1 (C function), 290  
 vsadduvi\_uint8xm2 (C function), 290  
 vsadduvi\_uint8xm4 (C function), 290  
 vsadduvi\_uint8xm8 (C function), 290  
 vsadduvv\_mask\_uint16xm1 (C function), 292  
 vsadduvv\_mask\_uint16xm2 (C function), 292



`vsadduvv_mask_uint16xm4` (C function), 292  
`vsadduvv_mask_uint16xm8` (C function), 292  
`vsadduvv_mask_uint32xm1` (C function), 292  
`vsadduvv_mask_uint32xm2` (C function), 292  
`vsadduvv_mask_uint32xm4` (C function), 292  
`vsadduvv_mask_uint32xm8` (C function), 292  
`vsadduvv_mask_uint64xm1` (C function), 292  
`vsadduvv_mask_uint64xm2` (C function), 292  
`vsadduvv_mask_uint64xm4` (C function), 292  
`vsadduvv_mask_uint64xm8` (C function), 292  
`vsadduvv_mask_uint8xm1` (C function), 293  
`vsadduvv_mask_uint8xm2` (C function), 293  
`vsadduvv_mask_uint8xm4` (C function), 293  
`vsadduvv_mask_uint8xm8` (C function), 293  
`vsadduvv_uint16xm1` (C function), 291  
`vsadduvv_uint16xm2` (C function), 291  
`vsadduvv_uint16xm4` (C function), 291  
`vsadduvv_uint16xm8` (C function), 292  
`vsadduvv_uint32xm1` (C function), 292  
`vsadduvv_uint32xm2` (C function), 292  
`vsadduvv_uint32xm4` (C function), 292  
`vsadduvv_uint32xm8` (C function), 292  
`vsadduvv_uint64xm1` (C function), 292  
`vsadduvv_uint64xm2` (C function), 292  
`vsadduvv_uint64xm4` (C function), 292  
`vsadduvv_uint64xm8` (C function), 292  
`vsadduvv_uint8xm1` (C function), 292  
`vsadduvv_uint8xm2` (C function), 292  
`vsadduvv_uint8xm4` (C function), 292  
`vsadduvv_uint8xm8` (C function), 292  
`vsadduvx_mask_uint16xm1` (C function), 294  
`vsadduvx_mask_uint16xm2` (C function), 294  
`vsadduvx_mask_uint16xm4` (C function), 294  
`vsadduvx_mask_uint16xm8` (C function), 294  
`vsadduvx_mask_uint32xm1` (C function), 294  
`vsadduvx_mask_uint32xm2` (C function), 294  
`vsadduvx_mask_uint32xm4` (C function), 294  
`vsadduvx_mask_uint32xm8` (C function), 294  
`vsadduvx_mask_uint64xm1` (C function), 294  
`vsadduvx_mask_uint64xm2` (C function), 294  
`vsadduvx_mask_uint64xm4` (C function), 294  
`vsadduvx_mask_uint64xm8` (C function), 294  
`vsadduvx_mask_uint8xm1` (C function), 294  
`vsadduvx_mask_uint8xm2` (C function), 294  
`vsadduvx_mask_uint8xm4` (C function), 294  
`vsadduvx_mask_uint8xm8` (C function), 294  
`vsadduvx_uint16xm1` (C function), 293  
`vsadduvx_uint16xm2` (C function), 293  
`vsadduvx_uint16xm4` (C function), 293  
`vsadduvx_uint16xm8` (C function), 293  
`vsadduvx_uint32xm1` (C function), 293  
`vsadduvx_uint32xm2` (C function), 293  
`vsadduvx_uint32xm4` (C function), 293  
`vsadduvx_uint32xm8` (C function), 293  
`vsadduvx_uint64xm1` (C function), 293  
`vsadduvx_uint64xm2` (C function), 293  
`vsadduvx_uint64xm4` (C function), 293  
`vsadduvx_uint64xm8` (C function), 293  
`vsaddvx_uint8xm1` (C function), 293  
`vsaddvx_uint8xm2` (C function), 293  
`vsaddvx_uint8xm4` (C function), 293  
`vsaddvx_uint8xm8` (C function), 293  
`vsaddvi_int16xm1` (C function), 285  
`vsaddvi_int16xm2` (C function), 285  
`vsaddvi_int16xm4` (C function), 285  
`vsaddvi_int16xm8` (C function), 285  
`vsaddvi_int32xm1` (C function), 285  
`vsaddvi_int32xm2` (C function), 285  
`vsaddvi_int32xm4` (C function), 285  
`vsaddvi_int32xm8` (C function), 285  
`vsaddvi_int64xm1` (C function), 285  
`vsaddvi_int64xm2` (C function), 286  
`vsaddvi_int64xm4` (C function), 286  
`vsaddvi_int64xm8` (C function), 286  
`vsaddvi_int8xm1` (C function), 286  
`vsaddvi_int8xm2` (C function), 286  
`vsaddvi_int8xm4` (C function), 286  
`vsaddvi_int8xm8` (C function), 286  
`vsaddvi_mask_int16xm1` (C function), 286  
`vsaddvi_mask_int16xm2` (C function), 286  
`vsaddvi_mask_int16xm4` (C function), 286  
`vsaddvi_mask_int16xm8` (C function), 286  
`vsaddvi_mask_int32xm1` (C function), 286  
`vsaddvi_mask_int32xm2` (C function), 286  
`vsaddvi_mask_int32xm4` (C function), 286  
`vsaddvi_mask_int32xm8` (C function), 286  
`vsaddvi_mask_int64xm1` (C function), 286  
`vsaddvi_mask_int64xm2` (C function), 286  
`vsaddvi_mask_int64xm4` (C function), 286  
`vsaddvi_mask_int64xm8` (C function), 286  
`vsaddvi_mask_int8xm1` (C function), 286  
`vsaddvi_mask_int8xm2` (C function), 286  
`vsaddvi_mask_int8xm4` (C function), 286  
`vsaddvi_mask_int8xm8` (C function), 286  
`vsaddvv_int16xm1` (C function), 287  
`vsaddvv_int16xm2` (C function), 287  
`vsaddvv_int16xm4` (C function), 287  
`vsaddvv_int16xm8` (C function), 287  
`vsaddvv_int32xm1` (C function), 287  
`vsaddvv_int32xm2` (C function), 287  
`vsaddvv_int32xm4` (C function), 287  
`vsaddvv_int32xm8` (C function), 287  
`vsaddvv_int64xm1` (C function), 287  
`vsaddvv_int64xm2` (C function), 287  
`vsaddvv_int64xm4` (C function), 287  
`vsaddvv_int64xm8` (C function), 287  
`vsaddvv_int8xm1` (C function), 287  
`vsaddvv_int8xm2` (C function), 287



`vsbvcvm_mask_uint64xm4` (C function), 297  
`vsbvcvm_mask_uint64xm8` (C function), 297  
`vsbvcvm_mask_uint8xm1` (C function), 297  
`vsbvcvm_mask_uint8xm2` (C function), 297  
`vsbvcvm_mask_uint8xm4` (C function), 297  
`vsbvcvm_mask_uint8xm8` (C function), 297  
`vsbv_int16xm1` (C function), 450  
`vsbv_int16xm2` (C function), 450  
`vsbv_int16xm4` (C function), 450  
`vsbv_int16xm8` (C function), 450  
`vsbv_int32xm1` (C function), 450  
`vsbv_int32xm2` (C function), 450  
`vsbv_int32xm4` (C function), 450  
`vsbv_int32xm8` (C function), 450  
`vsbv_int64xm1` (C function), 450  
`vsbv_int64xm2` (C function), 450  
`vsbv_int64xm4` (C function), 451  
`vsbv_int64xm8` (C function), 451  
`vsbv_int8xm1` (C function), 451  
`vsbv_int8xm2` (C function), 451  
`vsbv_int8xm4` (C function), 451  
`vsbv_int8xm8` (C function), 451  
`vsbv_mask_int16xm1` (C function), 451  
`vsbv_mask_int16xm2` (C function), 451  
`vsbv_mask_int16xm4` (C function), 451  
`vsbv_mask_int16xm8` (C function), 451  
`vsbv_mask_int32xm1` (C function), 451  
`vsbv_mask_int32xm2` (C function), 451  
`vsbv_mask_int32xm4` (C function), 451  
`vsbv_mask_int32xm8` (C function), 451  
`vsbv_mask_int64xm1` (C function), 451  
`vsbv_mask_int64xm2` (C function), 452  
`vsbv_mask_int64xm4` (C function), 452  
`vsbv_mask_int64xm8` (C function), 452  
`vsbv_mask_int8xm1` (C function), 452  
`vsbv_mask_int8xm2` (C function), 452  
`vsbv_mask_int8xm4` (C function), 452  
`vsbv_mask_int8xm8` (C function), 452  
`vsbv_mask_uint16xm1` (C function), 452  
`vsbv_mask_uint16xm2` (C function), 452  
`vsbv_mask_uint16xm4` (C function), 452  
`vsbv_mask_uint16xm8` (C function), 452  
`vsbv_mask_uint32xm1` (C function), 452  
`vsbv_mask_uint32xm2` (C function), 452  
`vsbv_mask_uint32xm4` (C function), 452  
`vsbv_mask_uint32xm8` (C function), 452  
`vsbv_mask_uint64xm1` (C function), 452  
`vsbv_mask_uint64xm2` (C function), 452  
`vsbv_mask_uint64xm4` (C function), 452  
`vsbv_mask_uint64xm8` (C function), 452  
`vsbv_mask_uint8xm1` (C function), 452  
`vsbv_mask_uint8xm2` (C function), 452  
`vsbv_mask_uint8xm4` (C function), 452  
`vsbv_mask_uint8xm8` (C function), 452  
`vsbv_uint16xm1` (C function), 451  
`vsbv_uint16xm2` (C function), 451  
`vsbv_uint16xm4` (C function), 451  
`vsbv_uint16xm8` (C function), 451  
`vsbv_uint32xm1` (C function), 451  
`vsbv_uint32xm2` (C function), 451  
`vsbv_uint32xm4` (C function), 451  
`vsbv_uint32xm8` (C function), 451  
`vsbv_uint64xm1` (C function), 451  
`vsbv_uint64xm2` (C function), 451  
`vsbv_uint64xm4` (C function), 451  
`vsbv_uint64xm8` (C function), 451  
`vsbv_uint8xm1` (C function), 451  
`vsbv_uint8xm2` (C function), 451  
`vsbv_uint8xm4` (C function), 451  
`vsbv_uint8xm8` (C function), 451  
`vsetvl` (C function), 5  
`vsetvli` (C function), 5  
`vsev_float16xm1` (C function), 453  
`vsev_float16xm2` (C function), 453  
`vsev_float16xm4` (C function), 453  
`vsev_float16xm8` (C function), 453  
`vsev_float32xm1` (C function), 453  
`vsev_float32xm2` (C function), 453  
`vsev_float32xm4` (C function), 453  
`vsev_float32xm8` (C function), 453  
`vsev_float64xm1` (C function), 453  
`vsev_float64xm2` (C function), 453  
`vsev_float64xm4` (C function), 453  
`vsev_float64xm8` (C function), 453  
`vsev_int16xm1` (C function), 453  
`vsev_int16xm2` (C function), 453  
`vsev_int16xm4` (C function), 453  
`vsev_int16xm8` (C function), 453  
`vsev_int32xm1` (C function), 453  
`vsev_int32xm2` (C function), 453  
`vsev_int32xm4` (C function), 453  
`vsev_int32xm8` (C function), 453  
`vsev_int64xm1` (C function), 453  
`vsev_int64xm2` (C function), 453  
`vsev_int64xm4` (C function), 453  
`vsev_int64xm8` (C function), 453  
`vsev_int8xm1` (C function), 453  
`vsev_int8xm2` (C function), 453  
`vsev_int8xm4` (C function), 453  
`vsev_int8xm8` (C function), 453  
`vsev_mask_float16xm1` (C function), 454  
`vsev_mask_float16xm2` (C function), 454  
`vsev_mask_float16xm4` (C function), 454  
`vsev_mask_float16xm8` (C function), 454  
`vsev_mask_float32xm1` (C function), 454  
`vsev_mask_float32xm2` (C function), 454  
`vsev_mask_float32xm4` (C function), 454  
`vsev_mask_float32xm8` (C function), 454

vsev\_mask\_float64xm1 (C function), 454  
 vsev\_mask\_float64xm2 (C function), 454  
 vsev\_mask\_float64xm4 (C function), 454  
 vsev\_mask\_float64xm8 (C function), 454  
 vsev\_mask\_int16xm1 (C function), 454  
 vsev\_mask\_int16xm2 (C function), 454  
 vsev\_mask\_int16xm4 (C function), 454  
 vsev\_mask\_int16xm8 (C function), 454  
 vsev\_mask\_int32xm1 (C function), 454  
 vsev\_mask\_int32xm2 (C function), 454  
 vsev\_mask\_int32xm4 (C function), 454  
 vsev\_mask\_int32xm8 (C function), 454  
 vsev\_mask\_int64xm1 (C function), 455  
 vsev\_mask\_int64xm2 (C function), 455  
 vsev\_mask\_int64xm4 (C function), 455  
 vsev\_mask\_int64xm8 (C function), 455  
 vsev\_mask\_int8xm1 (C function), 455  
 vsev\_mask\_int8xm2 (C function), 455  
 vsev\_mask\_int8xm4 (C function), 455  
 vsev\_mask\_int8xm8 (C function), 455  
 vsev\_mask\_uint16xm1 (C function), 455  
 vsev\_mask\_uint16xm2 (C function), 455  
 vsev\_mask\_uint16xm4 (C function), 455  
 vsev\_mask\_uint16xm8 (C function), 455  
 vsev\_mask\_uint32xm1 (C function), 455  
 vsev\_mask\_uint32xm2 (C function), 455  
 vsev\_mask\_uint32xm4 (C function), 455  
 vsev\_mask\_uint32xm8 (C function), 455  
 vsev\_mask\_uint64xm1 (C function), 455  
 vsev\_mask\_uint64xm2 (C function), 455  
 vsev\_mask\_uint64xm4 (C function), 455  
 vsev\_mask\_uint64xm8 (C function), 455  
 vsev\_mask\_uint8xm1 (C function), 455  
 vsev\_mask\_uint8xm2 (C function), 455  
 vsev\_mask\_uint8xm4 (C function), 455  
 vsev\_mask\_uint8xm8 (C function), 455  
 vsev\_uint16xm1 (C function), 453  
 vsev\_uint16xm2 (C function), 453  
 vsev\_uint16xm4 (C function), 453  
 vsev\_uint16xm8 (C function), 453  
 vsev\_uint32xm1 (C function), 453  
 vsev\_uint32xm2 (C function), 454  
 vsev\_uint32xm4 (C function), 454  
 vsev\_uint32xm8 (C function), 454  
 vsev\_uint64xm1 (C function), 454  
 vsev\_uint64xm2 (C function), 454  
 vsev\_uint64xm4 (C function), 454  
 vsev\_uint64xm8 (C function), 454  
 vsev\_uint8xm1 (C function), 454  
 vsev\_uint8xm2 (C function), 454  
 vsev\_uint8xm4 (C function), 454  
 vsev\_uint8xm8 (C function), 454  
 vshv\_int16xm1 (C function), 456  
 vshv\_int16xm2 (C function), 456

vshv\_int16xm4 (C function), 456  
 vshv\_int16xm8 (C function), 456  
 vshv\_int32xm1 (C function), 456  
 vshv\_int32xm2 (C function), 456  
 vshv\_int32xm4 (C function), 456  
 vshv\_int32xm8 (C function), 456  
 vshv\_int64xm1 (C function), 456  
 vshv\_int64xm2 (C function), 456  
 vshv\_int64xm4 (C function), 456  
 vshv\_int64xm8 (C function), 456  
 vshv\_int8xm1 (C function), 456  
 vshv\_int8xm2 (C function), 456  
 vshv\_int8xm4 (C function), 456  
 vshv\_int8xm8 (C function), 456  
 vshv\_mask\_int16xm1 (C function), 457  
 vshv\_mask\_int16xm2 (C function), 457  
 vshv\_mask\_int16xm4 (C function), 457  
 vshv\_mask\_int16xm8 (C function), 457  
 vshv\_mask\_int32xm1 (C function), 457  
 vshv\_mask\_int32xm2 (C function), 457  
 vshv\_mask\_int32xm4 (C function), 457  
 vshv\_mask\_int32xm8 (C function), 457  
 vshv\_mask\_int64xm1 (C function), 457  
 vshv\_mask\_int64xm2 (C function), 457  
 vshv\_mask\_int64xm4 (C function), 457  
 vshv\_mask\_int64xm8 (C function), 457  
 vshv\_mask\_int8xm1 (C function), 457  
 vshv\_mask\_int8xm2 (C function), 457  
 vshv\_mask\_int8xm4 (C function), 457  
 vshv\_mask\_int8xm8 (C function), 457  
 vshv\_mask\_uint16xm1 (C function), 457  
 vshv\_mask\_uint16xm2 (C function), 457  
 vshv\_mask\_uint16xm4 (C function), 457  
 vshv\_mask\_uint16xm8 (C function), 457  
 vshv\_mask\_uint32xm1 (C function), 457  
 vshv\_mask\_uint32xm2 (C function), 457  
 vshv\_mask\_uint32xm4 (C function), 457  
 vshv\_mask\_uint32xm8 (C function), 457  
 vshv\_mask\_uint64xm1 (C function), 457  
 vshv\_mask\_uint64xm2 (C function), 457  
 vshv\_mask\_uint64xm4 (C function), 458  
 vshv\_mask\_uint64xm8 (C function), 458  
 vshv\_mask\_uint8xm1 (C function), 458  
 vshv\_mask\_uint8xm2 (C function), 458  
 vshv\_mask\_uint8xm4 (C function), 458  
 vshv\_mask\_uint8xm8 (C function), 458  
 vshv\_uint16xm1 (C function), 456  
 vshv\_uint16xm2 (C function), 456  
 vshv\_uint16xm4 (C function), 456  
 vshv\_uint16xm8 (C function), 456  
 vshv\_uint32xm1 (C function), 456  
 vshv\_uint32xm2 (C function), 456  
 vshv\_uint32xm4 (C function), 456  
 vshv\_uint32xm8 (C function), 456



[illegible]





[illegible]



[illegible]

vsllvi\_int32xm1 (C function), 24  
 vsllvi\_int32xm2 (C function), 24  
 vsllvi\_int32xm4 (C function), 24  
 vsllvi\_int32xm8 (C function), 24  
 vsllvi\_int64xm1 (C function), 24  
 vsllvi\_int64xm2 (C function), 24  
 vsllvi\_int64xm4 (C function), 24  
 vsllvi\_int64xm8 (C function), 24  
 vsllvi\_int8xm1 (C function), 25  
 vsllvi\_int8xm2 (C function), 25  
 vsllvi\_int8xm4 (C function), 25  
 vsllvi\_int8xm8 (C function), 25  
 vsllvi\_mask\_int16xm1 (C function), 25  
 vsllvi\_mask\_int16xm2 (C function), 25  
 vsllvi\_mask\_int16xm4 (C function), 25  
 vsllvi\_mask\_int16xm8 (C function), 25  
 vsllvi\_mask\_int32xm1 (C function), 25  
 vsllvi\_mask\_int32xm2 (C function), 25  
 vsllvi\_mask\_int32xm4 (C function), 25  
 vsllvi\_mask\_int32xm8 (C function), 26  
 vsllvi\_mask\_int64xm1 (C function), 26  
 vsllvi\_mask\_int64xm2 (C function), 26  
 vsllvi\_mask\_int64xm4 (C function), 26  
 vsllvi\_mask\_int64xm8 (C function), 26  
 vsllvi\_mask\_int8xm1 (C function), 26  
 vsllvi\_mask\_int8xm2 (C function), 26  
 vsllvi\_mask\_int8xm4 (C function), 26  
 vsllvi\_mask\_int8xm8 (C function), 26  
 vsllvi\_mask\_uint16xm1 (C function), 26  
 vsllvi\_mask\_uint16xm2 (C function), 26  
 vsllvi\_mask\_uint16xm4 (C function), 26  
 vsllvi\_mask\_uint16xm8 (C function), 26  
 vsllvi\_mask\_uint32xm1 (C function), 26  
 vsllvi\_mask\_uint32xm2 (C function), 26  
 vsllvi\_mask\_uint32xm4 (C function), 26  
 vsllvi\_mask\_uint32xm8 (C function), 26  
 vsllvi\_mask\_uint64xm1 (C function), 26  
 vsllvi\_mask\_uint64xm2 (C function), 26  
 vsllvi\_mask\_uint64xm4 (C function), 26  
 vsllvi\_mask\_uint64xm8 (C function), 26  
 vsllvi\_mask\_uint8xm1 (C function), 26  
 vsllvi\_mask\_uint8xm2 (C function), 26  
 vsllvi\_mask\_uint8xm4 (C function), 27  
 vsllvi\_mask\_uint8xm8 (C function), 27  
 vsllvi\_uint16xm1 (C function), 25  
 vsllvi\_uint16xm2 (C function), 25  
 vsllvi\_uint16xm4 (C function), 25  
 vsllvi\_uint16xm8 (C function), 25  
 vsllvi\_uint32xm1 (C function), 25  
 vsllvi\_uint32xm2 (C function), 25  
 vsllvi\_uint32xm4 (C function), 25  
 vsllvi\_uint32xm8 (C function), 25  
 vsllvi\_uint64xm1 (C function), 25  
 vsllvi\_uint64xm2 (C function), 25  
 vsllvi\_uint64xm4 (C function), 25  
 vsllvi\_uint64xm8 (C function), 25  
 vsllvi\_uint8xm1 (C function), 25  
 vsllvi\_uint8xm2 (C function), 25  
 vsllvi\_uint8xm4 (C function), 25  
 vsllvi\_uint8xm8 (C function), 25  
 vsllvv\_int16xm1\_uint16xm1 (C function), 27  
 vsllvv\_int16xm2\_uint16xm2 (C function), 27  
 vsllvv\_int16xm4\_uint16xm4 (C function), 27  
 vsllvv\_int16xm8\_uint16xm8 (C function), 27  
 vsllvv\_int32xm1\_uint32xm1 (C function), 27  
 vsllvv\_int32xm2\_uint32xm2 (C function), 27  
 vsllvv\_int32xm4\_uint32xm4 (C function), 27  
 vsllvv\_int32xm8\_uint32xm8 (C function), 27  
 vsllvv\_int64xm1\_uint64xm1 (C function), 27  
 vsllvv\_int64xm2\_uint64xm2 (C function), 27  
 vsllvv\_int64xm4\_uint64xm4 (C function), 27  
 vsllvv\_int64xm8\_uint64xm8 (C function), 27  
 vsllvv\_int8xm1\_uint8xm1 (C function), 27  
 vsllvv\_int8xm2\_uint8xm2 (C function), 27  
 vsllvv\_int8xm4\_uint8xm4 (C function), 27  
 vsllvv\_int8xm8\_uint8xm8 (C function), 27  
 vsllvv\_mask\_int16xm1\_uint16xm1 (C function), 28  
 vsllvv\_mask\_int16xm2\_uint16xm2 (C function), 28  
 vsllvv\_mask\_int16xm4\_uint16xm4 (C function), 28  
 vsllvv\_mask\_int16xm8\_uint16xm8 (C function), 28  
 vsllvv\_mask\_int32xm1\_uint32xm1 (C function), 28  
 vsllvv\_mask\_int32xm2\_uint32xm2 (C function), 28  
 vsllvv\_mask\_int32xm4\_uint32xm4 (C function), 28  
 vsllvv\_mask\_int32xm8\_uint32xm8 (C function), 28  
 vsllvv\_mask\_int64xm1\_uint64xm1 (C function), 28  
 vsllvv\_mask\_int64xm2\_uint64xm2 (C function), 28  
 vsllvv\_mask\_int64xm4\_uint64xm4 (C function), 29  
 vsllvv\_mask\_int64xm8\_uint64xm8 (C function), 29  
 vsllvv\_mask\_int8xm1\_uint8xm1 (C function), 29  
 vsllvv\_mask\_int8xm2\_uint8xm2 (C function), 29  
 vsllvv\_mask\_int8xm4\_uint8xm4 (C function), 29  
 vsllvv\_mask\_int8xm8\_uint8xm8 (C function), 29  
 vsllvv\_mask\_uint16xm1 (C function), 29  
 vsllvv\_mask\_uint16xm2 (C function), 29  
 vsllvv\_mask\_uint16xm4 (C function), 29  
 vsllvv\_mask\_uint16xm8 (C function), 29  
 vsllvv\_mask\_uint32xm1 (C function), 29  
 vsllvv\_mask\_uint32xm2 (C function), 29  
 vsllvv\_mask\_uint32xm4 (C function), 29  
 vsllvv\_mask\_uint32xm8 (C function), 29  
 vsllvv\_mask\_uint64xm1 (C function), 29  
 vsllvv\_mask\_uint64xm2 (C function), 29  
 vsllvv\_mask\_uint64xm4 (C function), 29  
 vsllvv\_mask\_uint64xm8 (C function), 29  
 vsllvv\_mask\_uint8xm1 (C function), 29  
 vsllvv\_mask\_uint8xm2 (C function), 29  
 vsllvv\_mask\_uint8xm4 (C function), 29  
 vsllvv\_mask\_uint8xm8 (C function), 29



vsllvv\_uint16xm1 (C function), 27  
vsllvv\_uint16xm2 (C function), 27  
vsllvv\_uint16xm4 (C function), 27  
vsllvv\_uint16xm8 (C function), 27  
vsllvv\_uint32xm1 (C function), 27  
vsllvv\_uint32xm2 (C function), 27  
vsllvv\_uint32xm4 (C function), 27  
vsllvv\_uint32xm8 (C function), 28  
vsllvv\_uint64xm1 (C function), 28  
vsllvv\_uint64xm2 (C function), 28  
vsllvv\_uint64xm4 (C function), 28  
vsllvv\_uint64xm8 (C function), 28  
vsllvv\_uint8xm1 (C function), 28  
vsllvv\_uint8xm2 (C function), 28  
vsllvv\_uint8xm4 (C function), 28  
vsllvv\_uint8xm8 (C function), 28  
vsllvx\_int16xm1 (C function), 30  
vsllvx\_int16xm2 (C function), 30  
vsllvx\_int16xm4 (C function), 30  
vsllvx\_int16xm8 (C function), 30  
vsllvx\_int32xm1 (C function), 30  
vsllvx\_int32xm2 (C function), 30  
vsllvx\_int32xm4 (C function), 30  
vsllvx\_int32xm8 (C function), 30  
vsllvx\_int64xm1 (C function), 30  
vsllvx\_int64xm2 (C function), 30  
vsllvx\_int64xm4 (C function), 30  
vsllvx\_int64xm8 (C function), 30  
vsllvx\_int8xm1 (C function), 30  
vsllvx\_int8xm2 (C function), 30  
vsllvx\_int8xm4 (C function), 30  
vsllvx\_int8xm8 (C function), 30  
vsllvx\_mask\_int16xm1 (C function), 31  
vsllvx\_mask\_int16xm2 (C function), 31  
vsllvx\_mask\_int16xm4 (C function), 31  
vsllvx\_mask\_int16xm8 (C function), 31  
vsllvx\_mask\_int32xm1 (C function), 31  
vsllvx\_mask\_int32xm2 (C function), 31  
vsllvx\_mask\_int32xm4 (C function), 31  
vsllvx\_mask\_int32xm8 (C function), 31  
vsllvx\_mask\_int64xm1 (C function), 31  
vsllvx\_mask\_int64xm2 (C function), 31  
vsllvx\_mask\_int64xm4 (C function), 31  
vsllvx\_mask\_int64xm8 (C function), 31  
vsllvx\_mask\_int8xm1 (C function), 31  
vsllvx\_mask\_int8xm2 (C function), 31  
vsllvx\_mask\_int8xm4 (C function), 31  
vsllvx\_mask\_int8xm8 (C function), 31  
vsllvx\_mask\_uint16xm1 (C function), 31  
vsllvx\_mask\_uint16xm2 (C function), 32  
vsllvx\_mask\_uint16xm4 (C function), 32  
vsllvx\_mask\_uint16xm8 (C function), 32  
vsllvx\_mask\_uint32xm1 (C function), 32  
vsllvx\_mask\_uint32xm2 (C function), 32

vsllvx\_mask\_uint32xm4 (C function), 32  
vsllvx\_mask\_uint32xm8 (C function), 32  
vsllvx\_mask\_uint64xm1 (C function), 32  
vsllvx\_mask\_uint64xm2 (C function), 32  
vsllvx\_mask\_uint64xm4 (C function), 32  
vsllvx\_mask\_uint64xm8 (C function), 32  
vsllvx\_mask\_uint8xm1 (C function), 32  
vsllvx\_mask\_uint8xm2 (C function), 32  
vsllvx\_mask\_uint8xm4 (C function), 32  
vsllvx\_mask\_uint8xm8 (C function), 32  
vsllvx\_uint16xm1 (C function), 30  
vsllvx\_uint16xm2 (C function), 30  
vsllvx\_uint16xm4 (C function), 30  
vsllvx\_uint16xm8 (C function), 30  
vsllvx\_uint32xm1 (C function), 30  
vsllvx\_uint32xm2 (C function), 30  
vsllvx\_uint32xm4 (C function), 30  
vsllvx\_uint32xm8 (C function), 30  
vsllvx\_uint64xm1 (C function), 30  
vsllvx\_uint64xm2 (C function), 30  
vsllvx\_uint64xm4 (C function), 30  
vsllvx\_uint64xm8 (C function), 31  
vsllvx\_uint8xm1 (C function), 31  
vsllvx\_uint8xm2 (C function), 31  
vsllvx\_uint8xm4 (C function), 31  
vsllvx\_uint8xm8 (C function), 31  
vsmulvv\_int16xm1 (C function), 297  
vsmulvv\_int16xm2 (C function), 297  
vsmulvv\_int16xm4 (C function), 297  
vsmulvv\_int16xm8 (C function), 297  
vsmulvv\_int32xm1 (C function), 297  
vsmulvv\_int32xm2 (C function), 297  
vsmulvv\_int32xm4 (C function), 298  
vsmulvv\_int32xm8 (C function), 298  
vsmulvv\_int64xm1 (C function), 298  
vsmulvv\_int64xm2 (C function), 298  
vsmulvv\_int64xm4 (C function), 298  
vsmulvv\_int64xm8 (C function), 298  
vsmulvv\_int8xm1 (C function), 298  
vsmulvv\_int8xm2 (C function), 298  
vsmulvv\_int8xm4 (C function), 298  
vsmulvv\_int8xm8 (C function), 298  
vsmulvv\_mask\_int16xm1 (C function), 298  
vsmulvv\_mask\_int16xm2 (C function), 298  
vsmulvv\_mask\_int16xm4 (C function), 298  
vsmulvv\_mask\_int16xm8 (C function), 298  
vsmulvv\_mask\_int32xm1 (C function), 298  
vsmulvv\_mask\_int32xm2 (C function), 298  
vsmulvv\_mask\_int32xm4 (C function), 298  
vsmulvv\_mask\_int32xm8 (C function), 298  
vsmulvv\_mask\_int64xm1 (C function), 298  
vsmulvv\_mask\_int64xm2 (C function), 298  
vsmulvv\_mask\_int64xm4 (C function), 298  
vsmulvv\_mask\_int64xm8 (C function), 298

vsmulvv\_mask\_int8xm1 (C function), 298  
 vsmulvv\_mask\_int8xm2 (C function), 298  
 vsmulvv\_mask\_int8xm4 (C function), 299  
 vsmulvv\_mask\_int8xm8 (C function), 299  
 vsmulvx\_int16xm1 (C function), 299  
 vsmulvx\_int16xm2 (C function), 299  
 vsmulvx\_int16xm4 (C function), 299  
 vsmulvx\_int16xm8 (C function), 299  
 vsmulvx\_int32xm1 (C function), 299  
 vsmulvx\_int32xm2 (C function), 299  
 vsmulvx\_int32xm4 (C function), 299  
 vsmulvx\_int32xm8 (C function), 299  
 vsmulvx\_int64xm1 (C function), 299  
 vsmulvx\_int64xm2 (C function), 299  
 vsmulvx\_int64xm4 (C function), 299  
 vsmulvx\_int64xm8 (C function), 299  
 vsmulvx\_int8xm1 (C function), 299  
 vsmulvx\_int8xm2 (C function), 299  
 vsmulvx\_int8xm4 (C function), 299  
 vsmulvx\_int8xm8 (C function), 299  
 vsmulvx\_mask\_int16xm1 (C function), 299  
 vsmulvx\_mask\_int16xm2 (C function), 300  
 vsmulvx\_mask\_int16xm4 (C function), 300  
 vsmulvx\_mask\_int16xm8 (C function), 300  
 vsmulvx\_mask\_int32xm1 (C function), 300  
 vsmulvx\_mask\_int32xm2 (C function), 300  
 vsmulvx\_mask\_int32xm4 (C function), 300  
 vsmulvx\_mask\_int32xm8 (C function), 300  
 vsmulvx\_mask\_int64xm1 (C function), 300  
 vsmulvx\_mask\_int64xm2 (C function), 300  
 vsmulvx\_mask\_int64xm4 (C function), 300  
 vsmulvx\_mask\_int64xm8 (C function), 300  
 vsmulvx\_mask\_int8xm1 (C function), 300  
 vsmulvx\_mask\_int8xm2 (C function), 300  
 vsmulvx\_mask\_int8xm4 (C function), 300  
 vsmulvx\_mask\_int8xm8 (C function), 300  
 vsravi\_int16xm1 (C function), 32  
 vsravi\_int16xm2 (C function), 32  
 vsravi\_int16xm4 (C function), 32  
 vsravi\_int16xm8 (C function), 33  
 vsravi\_int32xm1 (C function), 33  
 vsravi\_int32xm2 (C function), 33  
 vsravi\_int32xm4 (C function), 33  
 vsravi\_int32xm8 (C function), 33  
 vsravi\_int64xm1 (C function), 33  
 vsravi\_int64xm2 (C function), 33  
 vsravi\_int64xm4 (C function), 33  
 vsravi\_int64xm8 (C function), 33  
 vsravi\_int8xm1 (C function), 33  
 vsravi\_int8xm2 (C function), 33  
 vsravi\_int8xm4 (C function), 33  
 vsravi\_int8xm8 (C function), 33  
 vsravi\_mask\_int16xm1 (C function), 33  
 vsravi\_mask\_int16xm2 (C function), 33  
 vsravi\_mask\_int16xm4 (C function), 33  
 vsravi\_mask\_int16xm8 (C function), 33  
 vsravi\_mask\_int32xm1 (C function), 33  
 vsravi\_mask\_int32xm2 (C function), 33  
 vsravi\_mask\_int32xm4 (C function), 33  
 vsravi\_mask\_int32xm8 (C function), 33  
 vsravi\_mask\_int64xm1 (C function), 33  
 vsravi\_mask\_int64xm2 (C function), 33  
 vsravi\_mask\_int64xm4 (C function), 33  
 vsravi\_mask\_int64xm8 (C function), 33  
 vsravi\_mask\_int8xm1 (C function), 34  
 vsravi\_mask\_int8xm2 (C function), 34  
 vsravi\_mask\_int8xm4 (C function), 34  
 vsravi\_mask\_int8xm8 (C function), 34  
 vsravv\_int16xm1\_uint16xm1 (C function), 34  
 vsravv\_int16xm2\_uint16xm2 (C function), 34  
 vsravv\_int16xm4\_uint16xm4 (C function), 34  
 vsravv\_int16xm8\_uint16xm8 (C function), 34  
 vsravv\_int32xm1\_uint32xm1 (C function), 34  
 vsravv\_int32xm2\_uint32xm2 (C function), 34  
 vsravv\_int32xm4\_uint32xm4 (C function), 34  
 vsravv\_int32xm8\_uint32xm8 (C function), 34  
 vsravv\_int64xm1\_uint64xm1 (C function), 34  
 vsravv\_int64xm2\_uint64xm2 (C function), 34  
 vsravv\_int64xm4\_uint64xm4 (C function), 34  
 vsravv\_int64xm8\_uint64xm8 (C function), 34  
 vsravv\_int8xm1\_uint8xm1 (C function), 34  
 vsravv\_int8xm2\_uint8xm2 (C function), 34  
 vsravv\_int8xm4\_uint8xm4 (C function), 34  
 vsravv\_int8xm8\_uint8xm8 (C function), 34  
 vsravv\_mask\_int16xm1\_uint16xm1 (C function), 35  
 vsravv\_mask\_int16xm2\_uint16xm2 (C function), 35  
 vsravv\_mask\_int16xm4\_uint16xm4 (C function), 35  
 vsravv\_mask\_int16xm8\_uint16xm8 (C function), 35  
 vsravv\_mask\_int32xm1\_uint32xm1 (C function), 35  
 vsravv\_mask\_int32xm2\_uint32xm2 (C function), 35  
 vsravv\_mask\_int32xm4\_uint32xm4 (C function), 35  
 vsravv\_mask\_int32xm8\_uint32xm8 (C function), 35  
 vsravv\_mask\_int64xm1\_uint64xm1 (C function), 35  
 vsravv\_mask\_int64xm2\_uint64xm2 (C function), 35  
 vsravv\_mask\_int64xm4\_uint64xm4 (C function), 35  
 vsravv\_mask\_int64xm8\_uint64xm8 (C function), 35  
 vsravv\_mask\_int8xm1\_uint8xm1 (C function), 35  
 vsravv\_mask\_int8xm2\_uint8xm2 (C function), 35  
 vsravv\_mask\_int8xm4\_uint8xm4 (C function), 35  
 vsravv\_mask\_int8xm8\_uint8xm8 (C function), 35  
 vsravx\_int16xm1 (C function), 36  
 vsravx\_int16xm2 (C function), 36  
 vsravx\_int16xm4 (C function), 36  
 vsravx\_int16xm8 (C function), 36  
 vsravx\_int32xm1 (C function), 36  
 vsravx\_int32xm2 (C function), 36  
 vsravx\_int32xm4 (C function), 36  
 vsravx\_int32xm8 (C function), 36



vsravx\_int64xm1 (C function), 36  
vsravx\_int64xm2 (C function), 36  
vsravx\_int64xm4 (C function), 36  
vsravx\_int64xm8 (C function), 36  
vsravx\_int8xm1 (C function), 36  
vsravx\_int8xm2 (C function), 36  
vsravx\_int8xm4 (C function), 36  
vsravx\_int8xm8 (C function), 36  
vsravx\_mask\_int16xm1 (C function), 36  
vsravx\_mask\_int16xm2 (C function), 36  
vsravx\_mask\_int16xm4 (C function), 36  
vsravx\_mask\_int16xm8 (C function), 36  
vsravx\_mask\_int32xm1 (C function), 37  
vsravx\_mask\_int32xm2 (C function), 37  
vsravx\_mask\_int32xm4 (C function), 37  
vsravx\_mask\_int32xm8 (C function), 37  
vsravx\_mask\_int64xm1 (C function), 37  
vsravx\_mask\_int64xm2 (C function), 37  
vsravx\_mask\_int64xm4 (C function), 37  
vsravx\_mask\_int64xm8 (C function), 37  
vsravx\_mask\_int8xm1 (C function), 37  
vsravx\_mask\_int8xm2 (C function), 37  
vsravx\_mask\_int8xm4 (C function), 37  
vsravx\_mask\_int8xm8 (C function), 37  
vsrlvi\_mask\_uint16xm1 (C function), 38  
vsrlvi\_mask\_uint16xm2 (C function), 38  
vsrlvi\_mask\_uint16xm4 (C function), 38  
vsrlvi\_mask\_uint16xm8 (C function), 38  
vsrlvi\_mask\_uint32xm1 (C function), 38  
vsrlvi\_mask\_uint32xm2 (C function), 38  
vsrlvi\_mask\_uint32xm4 (C function), 38  
vsrlvi\_mask\_uint32xm8 (C function), 38  
vsrlvi\_mask\_uint64xm1 (C function), 38  
vsrlvi\_mask\_uint64xm2 (C function), 38  
vsrlvi\_mask\_uint64xm4 (C function), 38  
vsrlvi\_mask\_uint64xm8 (C function), 38  
vsrlvi\_mask\_uint8xm1 (C function), 38  
vsrlvi\_mask\_uint8xm2 (C function), 38  
vsrlvi\_mask\_uint8xm4 (C function), 38  
vsrlvi\_mask\_uint8xm8 (C function), 39  
vsrlvi\_uint16xm1 (C function), 37  
vsrlvi\_uint16xm2 (C function), 37  
vsrlvi\_uint16xm4 (C function), 37  
vsrlvi\_uint16xm8 (C function), 37  
vsrlvi\_uint32xm1 (C function), 37  
vsrlvi\_uint32xm2 (C function), 37  
vsrlvi\_uint32xm4 (C function), 37  
vsrlvi\_uint32xm8 (C function), 37  
vsrlvi\_uint64xm1 (C function), 38  
vsrlvi\_uint64xm2 (C function), 38  
vsrlvi\_uint64xm4 (C function), 38  
vsrlvi\_uint64xm8 (C function), 38  
vsrlvi\_uint8xm1 (C function), 38  
vsrlvi\_uint8xm2 (C function), 38

vsrlvi\_uint8xm4 (C function), 38  
vsrlvi\_uint8xm8 (C function), 38  
vsrlvv\_mask\_uint16xm1 (C function), 39  
vsrlvv\_mask\_uint16xm2 (C function), 39  
vsrlvv\_mask\_uint16xm4 (C function), 40  
vsrlvv\_mask\_uint16xm8 (C function), 40  
vsrlvv\_mask\_uint32xm1 (C function), 40  
vsrlvv\_mask\_uint32xm2 (C function), 40  
vsrlvv\_mask\_uint32xm4 (C function), 40  
vsrlvv\_mask\_uint32xm8 (C function), 40  
vsrlvv\_mask\_uint64xm1 (C function), 40  
vsrlvv\_mask\_uint64xm2 (C function), 40  
vsrlvv\_mask\_uint64xm4 (C function), 40  
vsrlvv\_mask\_uint64xm8 (C function), 40  
vsrlvv\_mask\_uint8xm1 (C function), 40  
vsrlvv\_mask\_uint8xm2 (C function), 40  
vsrlvv\_mask\_uint8xm4 (C function), 40  
vsrlvv\_mask\_uint8xm8 (C function), 40  
vsrlvv\_uint16xm1 (C function), 39  
vsrlvv\_uint16xm2 (C function), 39  
vsrlvv\_uint16xm4 (C function), 39  
vsrlvv\_uint16xm8 (C function), 39  
vsrlvv\_uint32xm1 (C function), 39  
vsrlvv\_uint32xm2 (C function), 39  
vsrlvv\_uint32xm4 (C function), 39  
vsrlvv\_uint32xm8 (C function), 39  
vsrlvv\_uint64xm1 (C function), 39  
vsrlvv\_uint64xm2 (C function), 39  
vsrlvv\_uint64xm4 (C function), 39  
vsrlvv\_uint64xm8 (C function), 39  
vsrlvv\_uint8xm1 (C function), 39  
vsrlvv\_uint8xm2 (C function), 39  
vsrlvv\_uint8xm4 (C function), 39  
vsrlvv\_uint8xm8 (C function), 39  
vsrlvx\_mask\_uint16xm1 (C function), 41  
vsrlvx\_mask\_uint16xm2 (C function), 41  
vsrlvx\_mask\_uint16xm4 (C function), 41  
vsrlvx\_mask\_uint16xm8 (C function), 41  
vsrlvx\_mask\_uint32xm1 (C function), 41  
vsrlvx\_mask\_uint32xm2 (C function), 41  
vsrlvx\_mask\_uint32xm4 (C function), 41  
vsrlvx\_mask\_uint32xm8 (C function), 41  
vsrlvx\_mask\_uint64xm1 (C function), 41  
vsrlvx\_mask\_uint64xm2 (C function), 41  
vsrlvx\_mask\_uint64xm4 (C function), 41  
vsrlvx\_mask\_uint64xm8 (C function), 41  
vsrlvx\_mask\_uint8xm1 (C function), 41  
vsrlvx\_mask\_uint8xm2 (C function), 42  
vsrlvx\_mask\_uint8xm4 (C function), 42  
vsrlvx\_mask\_uint8xm8 (C function), 42  
vsrlvx\_uint16xm1 (C function), 40  
vsrlvx\_uint16xm2 (C function), 40  
vsrlvx\_uint16xm4 (C function), 40  
vsrlvx\_uint16xm8 (C function), 40

vsrlvx\_uint32xm1 (C function), 40  
 vsrlvx\_uint32xm2 (C function), 41  
 vsrlvx\_uint32xm4 (C function), 41  
 vsrlvx\_uint32xm8 (C function), 41  
 vsrlvx\_uint64xm1 (C function), 41  
 vsrlvx\_uint64xm2 (C function), 41  
 vsrlvx\_uint64xm4 (C function), 41  
 vsrlvx\_uint64xm8 (C function), 41  
 vsrlvx\_uint8xm1 (C function), 41  
 vsrlvx\_uint8xm2 (C function), 41  
 vsrlvx\_uint8xm4 (C function), 41  
 vsrlvx\_uint8xm8 (C function), 41  
 vssbv\_int16xm1 (C function), 458  
 vssbv\_int16xm2 (C function), 458  
 vssbv\_int16xm4 (C function), 458  
 vssbv\_int16xm8 (C function), 458  
 vssbv\_int32xm1 (C function), 458  
 vssbv\_int32xm2 (C function), 458  
 vssbv\_int32xm4 (C function), 458  
 vssbv\_int32xm8 (C function), 458  
 vssbv\_int64xm1 (C function), 458  
 vssbv\_int64xm2 (C function), 458  
 vssbv\_int64xm4 (C function), 458  
 vssbv\_int64xm8 (C function), 458  
 vssbv\_int8xm1 (C function), 458  
 vssbv\_int8xm2 (C function), 458  
 vssbv\_int8xm4 (C function), 458  
 vssbv\_int8xm8 (C function), 458  
 vssbv\_mask\_int16xm1 (C function), 459  
 vssbv\_mask\_int16xm2 (C function), 459  
 vssbv\_mask\_int16xm4 (C function), 459  
 vssbv\_mask\_int16xm8 (C function), 459  
 vssbv\_mask\_int32xm1 (C function), 459  
 vssbv\_mask\_int32xm2 (C function), 459  
 vssbv\_mask\_int32xm4 (C function), 459  
 vssbv\_mask\_int32xm8 (C function), 459  
 vssbv\_mask\_int64xm1 (C function), 459  
 vssbv\_mask\_int64xm2 (C function), 459  
 vssbv\_mask\_int64xm4 (C function), 459  
 vssbv\_mask\_int64xm8 (C function), 459  
 vssbv\_mask\_int8xm1 (C function), 460  
 vssbv\_mask\_int8xm2 (C function), 460  
 vssbv\_mask\_int8xm4 (C function), 460  
 vssbv\_mask\_int8xm8 (C function), 460  
 vssbv\_mask\_uint16xm1 (C function), 460  
 vssbv\_mask\_uint16xm2 (C function), 460  
 vssbv\_mask\_uint16xm4 (C function), 460  
 vssbv\_mask\_uint16xm8 (C function), 460  
 vssbv\_mask\_uint32xm1 (C function), 460  
 vssbv\_mask\_uint32xm2 (C function), 460  
 vssbv\_mask\_uint32xm4 (C function), 460  
 vssbv\_mask\_uint32xm8 (C function), 460  
 vssbv\_mask\_uint64xm1 (C function), 460  
 vssbv\_mask\_uint64xm2 (C function), 460  
 vssbv\_mask\_uint64xm4 (C function), 460  
 vssbv\_mask\_uint64xm8 (C function), 460  
 vssbv\_mask\_uint8xm1 (C function), 460  
 vssbv\_mask\_uint8xm2 (C function), 460  
 vssbv\_mask\_uint8xm4 (C function), 460  
 vssbv\_mask\_uint8xm8 (C function), 460  
 vsseg2bv\_int16x2xm1 (C function), 670  
 vsseg2bv\_int16x2xm2 (C function), 670  
 vsseg2bv\_int16x2xm4 (C function), 670  
 vsseg2bv\_int32x2xm1 (C function), 670  
 vsseg2bv\_int32x2xm2 (C function), 670  
 vsseg2bv\_int32x2xm4 (C function), 670  
 vsseg2bv\_int64x2xm1 (C function), 670  
 vsseg2bv\_int64x2xm2 (C function), 670  
 vsseg2bv\_int64x2xm4 (C function), 670  
 vsseg2bv\_int8x2xm1 (C function), 670  
 vsseg2bv\_int8x2xm2 (C function), 670  
 vsseg2bv\_int8x2xm4 (C function), 670  
 vsseg2bv\_mask\_int16x2xm1 (C function), 671  
 vsseg2bv\_mask\_int16x2xm2 (C function), 671  
 vsseg2bv\_mask\_int16x2xm4 (C function), 671  
 vsseg2bv\_mask\_int32x2xm1 (C function), 671  
 vsseg2bv\_mask\_int32x2xm2 (C function), 671  
 vsseg2bv\_mask\_int32x2xm4 (C function), 671  
 vsseg2bv\_mask\_int64x2xm1 (C function), 671  
 vsseg2bv\_mask\_int64x2xm2 (C function), 671  
 vsseg2bv\_mask\_int64x2xm4 (C function), 671  
 vsseg2bv\_mask\_int8x2xm1 (C function), 671  
 vsseg2bv\_mask\_int8x2xm2 (C function), 671  
 vsseg2bv\_mask\_int8x2xm4 (C function), 671  
 vsseg2bv\_mask\_uint16x2xm1 (C function), 671  
 vsseg2bv\_mask\_uint16x2xm2 (C function), 671  
 vsseg2bv\_mask\_uint16x2xm4 (C function), 671  
 vsseg2bv\_mask\_uint32x2xm1 (C function), 672  
 vsseg2bv\_mask\_uint32x2xm2 (C function), 672  
 vsseg2bv\_mask\_uint32x2xm4 (C function), 672  
 vsseg2bv\_mask\_uint64x2xm1 (C function), 672  
 vsseg2bv\_mask\_uint64x2xm2 (C function), 672

vsseg2bv\_mask\_uint64x2xm4 (C function), 672  
vsseg2bv\_mask\_uint8x2xm1 (C function), 672  
vsseg2bv\_mask\_uint8x2xm2 (C function), 672  
vsseg2bv\_mask\_uint8x2xm4 (C function), 672  
vsseg2bv\_uint16x2xm1 (C function), 670  
vsseg2bv\_uint16x2xm2 (C function), 670  
vsseg2bv\_uint16x2xm4 (C function), 670  
vsseg2bv\_uint32x2xm1 (C function), 670  
vsseg2bv\_uint32x2xm2 (C function), 671  
vsseg2bv\_uint32x2xm4 (C function), 671  
vsseg2bv\_uint64x2xm1 (C function), 671  
vsseg2bv\_uint64x2xm2 (C function), 671  
vsseg2bv\_uint64x2xm4 (C function), 671  
vsseg2bv\_uint8x2xm1 (C function), 671  
vsseg2bv\_uint8x2xm2 (C function), 671  
vsseg2bv\_uint8x2xm4 (C function), 671  
vsseg2ev\_float16x2xm1 (C function), 672  
vsseg2ev\_float16x2xm2 (C function), 672  
vsseg2ev\_float16x2xm4 (C function), 672  
vsseg2ev\_float32x2xm1 (C function), 672  
vsseg2ev\_float32x2xm2 (C function), 672  
vsseg2ev\_float32x2xm4 (C function), 672  
vsseg2ev\_float64x2xm1 (C function), 672  
vsseg2ev\_float64x2xm2 (C function), 672  
vsseg2ev\_float64x2xm4 (C function), 672  
vsseg2ev\_int16x2xm1 (C function), 672  
vsseg2ev\_int16x2xm2 (C function), 672  
vsseg2ev\_int16x2xm4 (C function), 672  
vsseg2ev\_int32x2xm1 (C function), 672  
vsseg2ev\_int32x2xm2 (C function), 673  
vsseg2ev\_int32x2xm4 (C function), 673  
vsseg2ev\_int64x2xm1 (C function), 673  
vsseg2ev\_int64x2xm2 (C function), 673  
vsseg2ev\_int64x2xm4 (C function), 673  
vsseg2ev\_int8x2xm1 (C function), 673  
vsseg2ev\_int8x2xm2 (C function), 673  
vsseg2ev\_int8x2xm4 (C function), 673  
vsseg2ev\_mask\_float16x2xm1 (C function), 673  
vsseg2ev\_mask\_float16x2xm2 (C function), 673  
vsseg2ev\_mask\_float16x2xm4 (C function), 673  
vsseg2ev\_mask\_float32x2xm1 (C function), 673  
vsseg2ev\_mask\_float32x2xm2 (C function), 673  
vsseg2ev\_mask\_float32x2xm4 (C function), 673  
vsseg2ev\_mask\_float64x2xm1 (C function), 673  
vsseg2ev\_mask\_float64x2xm2 (C function), 674  
vsseg2ev\_mask\_float64x2xm4 (C function), 674  
vsseg2ev\_mask\_int16x2xm1 (C function), 674  
vsseg2ev\_mask\_int16x2xm2 (C function), 674  
vsseg2ev\_mask\_int16x2xm4 (C function), 674  
vsseg2ev\_mask\_int32x2xm1 (C function), 674  
vsseg2ev\_mask\_int32x2xm2 (C function), 674  
vsseg2ev\_mask\_int32x2xm4 (C function), 674  
vsseg2ev\_mask\_int64x2xm1 (C function), 674  
vsseg2ev\_mask\_int64x2xm2 (C function), 674  
vsseg2ev\_mask\_int64x2xm4 (C function), 674  
vsseg2ev\_mask\_int8x2xm1 (C function), 674  
vsseg2ev\_mask\_int8x2xm2 (C function), 674  
vsseg2ev\_mask\_int8x2xm4 (C function), 674  
vsseg2ev\_mask\_uint16x2xm1 (C function), 674  
vsseg2ev\_mask\_uint16x2xm2 (C function), 674  
vsseg2ev\_mask\_uint16x2xm4 (C function), 674  
vsseg2ev\_mask\_uint32x2xm1 (C function), 674  
vsseg2ev\_mask\_uint32x2xm2 (C function), 674  
vsseg2ev\_mask\_uint32x2xm4 (C function), 674  
vsseg2ev\_mask\_uint64x2xm1 (C function), 674  
vsseg2ev\_mask\_uint64x2xm2 (C function), 674  
vsseg2ev\_mask\_uint64x2xm4 (C function), 674  
vsseg2ev\_mask\_uint8x2xm1 (C function), 675  
vsseg2ev\_mask\_uint8x2xm2 (C function), 675  
vsseg2ev\_mask\_uint8x2xm4 (C function), 675  
vsseg2ev\_uint16x2xm1 (C function), 673  
vsseg2ev\_uint16x2xm2 (C function), 673  
vsseg2ev\_uint16x2xm4 (C function), 673  
vsseg2ev\_uint32x2xm1 (C function), 673  
vsseg2ev\_uint32x2xm2 (C function), 673  
vsseg2ev\_uint32x2xm4 (C function), 673  
vsseg2ev\_uint64x2xm1 (C function), 673  
vsseg2ev\_uint64x2xm2 (C function), 673  
vsseg2ev\_uint64x2xm4 (C function), 673  
vsseg2ev\_uint8x2xm1 (C function), 673  
vsseg2ev\_uint8x2xm2 (C function), 673  
vsseg2ev\_uint8x2xm4 (C function), 673  
vsseg2hv\_int16x2xm1 (C function), 675  
vsseg2hv\_int16x2xm2 (C function), 675  
vsseg2hv\_int16x2xm4 (C function), 675  
vsseg2hv\_int32x2xm1 (C function), 675  
vsseg2hv\_int32x2xm2 (C function), 675  
vsseg2hv\_int32x2xm4 (C function), 675  
vsseg2hv\_int64x2xm1 (C function), 675  
vsseg2hv\_int64x2xm2 (C function), 675  
vsseg2hv\_int64x2xm4 (C function), 675  
vsseg2hv\_int8x2xm1 (C function), 675  
vsseg2hv\_int8x2xm2 (C function), 675  
vsseg2hv\_int8x2xm4 (C function), 675  
vsseg2hv\_mask\_int16x2xm1 (C function), 676  
vsseg2hv\_mask\_int16x2xm2 (C function), 676  
vsseg2hv\_mask\_int16x2xm4 (C function), 676  
vsseg2hv\_mask\_int32x2xm1 (C function), 676  
vsseg2hv\_mask\_int32x2xm2 (C function), 676  
vsseg2hv\_mask\_int32x2xm4 (C function), 676  
vsseg2hv\_mask\_int64x2xm1 (C function), 676  
vsseg2hv\_mask\_int64x2xm2 (C function), 676  
vsseg2hv\_mask\_int64x2xm4 (C function), 676  
vsseg2hv\_mask\_int8x2xm1 (C function), 676  
vsseg2hv\_mask\_int8x2xm2 (C function), 676  
vsseg2hv\_mask\_int8x2xm4 (C function), 676  
vsseg2hv\_mask\_uint16x2xm1 (C function), 676  
vsseg2hv\_mask\_uint16x2xm2 (C function), 676





vsseg3ev\_int16x3xm1 (C function), 681  
vsseg3ev\_int16x3xm2 (C function), 681  
vsseg3ev\_int32x3xm1 (C function), 681  
vsseg3ev\_int32x3xm2 (C function), 681  
vsseg3ev\_int64x3xm1 (C function), 681  
vsseg3ev\_int64x3xm2 (C function), 681  
vsseg3ev\_int8x3xm1 (C function), 681  
vsseg3ev\_int8x3xm2 (C function), 681  
vsseg3ev\_mask\_float16x3xm1 (C function), 681  
vsseg3ev\_mask\_float16x3xm2 (C function), 681  
vsseg3ev\_mask\_float32x3xm1 (C function), 681  
vsseg3ev\_mask\_float32x3xm2 (C function), 682  
vsseg3ev\_mask\_float64x3xm1 (C function), 682  
vsseg3ev\_mask\_float64x3xm2 (C function), 682  
vsseg3ev\_mask\_int16x3xm1 (C function), 682  
vsseg3ev\_mask\_int16x3xm2 (C function), 682  
vsseg3ev\_mask\_int32x3xm1 (C function), 682  
vsseg3ev\_mask\_int32x3xm2 (C function), 682  
vsseg3ev\_mask\_int64x3xm1 (C function), 682  
vsseg3ev\_mask\_int64x3xm2 (C function), 682  
vsseg3ev\_mask\_int8x3xm1 (C function), 682  
vsseg3ev\_mask\_int8x3xm2 (C function), 682  
vsseg3ev\_mask\_uint16x3xm1 (C function), 682  
vsseg3ev\_mask\_uint16x3xm2 (C function), 682  
vsseg3ev\_mask\_uint32x3xm1 (C function), 682  
vsseg3ev\_mask\_uint32x3xm2 (C function), 682  
vsseg3ev\_mask\_uint64x3xm1 (C function), 682  
vsseg3ev\_mask\_uint64x3xm2 (C function), 682  
vsseg3ev\_mask\_uint8x3xm1 (C function), 682  
vsseg3ev\_mask\_uint8x3xm2 (C function), 682  
vsseg3ev\_uint16x3xm1 (C function), 681  
vsseg3ev\_uint16x3xm2 (C function), 681  
vsseg3ev\_uint32x3xm1 (C function), 681  
vsseg3ev\_uint32x3xm2 (C function), 681  
vsseg3ev\_uint64x3xm1 (C function), 681  
vsseg3ev\_uint64x3xm2 (C function), 681  
vsseg3ev\_uint8x3xm1 (C function), 681  
vsseg3ev\_uint8x3xm2 (C function), 681  
vsseg3hv\_int16x3xm1 (C function), 683  
vsseg3hv\_int16x3xm2 (C function), 683  
vsseg3hv\_int32x3xm1 (C function), 683  
vsseg3hv\_int32x3xm2 (C function), 683  
vsseg3hv\_int64x3xm1 (C function), 683  
vsseg3hv\_int64x3xm2 (C function), 683  
vsseg3hv\_int8x3xm1 (C function), 683  
vsseg3hv\_int8x3xm2 (C function), 683  
vsseg3hv\_mask\_int16x3xm1 (C function), 683  
vsseg3hv\_mask\_int16x3xm2 (C function), 683  
vsseg3hv\_mask\_int32x3xm1 (C function), 683  
vsseg3hv\_mask\_int32x3xm2 (C function), 683  
vsseg3hv\_mask\_int64x3xm1 (C function), 683  
vsseg3hv\_mask\_int64x3xm2 (C function), 683  
vsseg3hv\_mask\_int8x3xm1 (C function), 683  
vsseg3hv\_mask\_int8x3xm2 (C function), 684  
vsseg3hv\_mask\_uint16x3xm1 (C function), 684  
vsseg3hv\_mask\_uint16x3xm2 (C function), 684  
vsseg3hv\_mask\_uint32x3xm1 (C function), 684  
vsseg3hv\_mask\_uint32x3xm2 (C function), 684  
vsseg3hv\_mask\_uint64x3xm1 (C function), 684  
vsseg3hv\_mask\_uint64x3xm2 (C function), 684  
vsseg3hv\_mask\_uint8x3xm1 (C function), 684  
vsseg3hv\_mask\_uint8x3xm2 (C function), 684  
vsseg3hv\_uint16x3xm1 (C function), 683  
vsseg3hv\_uint16x3xm2 (C function), 683  
vsseg3hv\_uint32x3xm1 (C function), 683  
vsseg3hv\_uint32x3xm2 (C function), 683  
vsseg3hv\_uint64x3xm1 (C function), 683  
vsseg3hv\_uint64x3xm2 (C function), 683  
vsseg3hv\_uint8x3xm1 (C function), 683  
vsseg3hv\_uint8x3xm2 (C function), 683  
vsseg3wv\_int16x3xm1 (C function), 684  
vsseg3wv\_int16x3xm2 (C function), 684  
vsseg3wv\_int32x3xm1 (C function), 684  
vsseg3wv\_int32x3xm2 (C function), 684  
vsseg3wv\_int64x3xm1 (C function), 684  
vsseg3wv\_int64x3xm2 (C function), 684  
vsseg3wv\_int8x3xm1 (C function), 684  
vsseg3wv\_int8x3xm2 (C function), 684  
vsseg3wv\_mask\_int16x3xm1 (C function), 685  
vsseg3wv\_mask\_int16x3xm2 (C function), 685  
vsseg3wv\_mask\_int32x3xm1 (C function), 685  
vsseg3wv\_mask\_int32x3xm2 (C function), 685  
vsseg3wv\_mask\_int64x3xm1 (C function), 685  
vsseg3wv\_mask\_int64x3xm2 (C function), 685  
vsseg3wv\_mask\_int8x3xm1 (C function), 685  
vsseg3wv\_mask\_int8x3xm2 (C function), 685  
vsseg3wv\_mask\_uint16x3xm1 (C function), 685  
vsseg3wv\_mask\_uint16x3xm2 (C function), 685  
vsseg3wv\_mask\_uint32x3xm1 (C function), 685  
vsseg3wv\_mask\_uint32x3xm2 (C function), 685  
vsseg3wv\_mask\_uint64x3xm1 (C function), 685  
vsseg3wv\_mask\_uint64x3xm2 (C function), 685  
vsseg3wv\_mask\_uint8x3xm1 (C function), 685  
vsseg3wv\_mask\_uint8x3xm2 (C function), 685  
vsseg3wv\_uint16x3xm1 (C function), 684  
vsseg3wv\_uint16x3xm2 (C function), 684  
vsseg3wv\_uint32x3xm1 (C function), 684  
vsseg3wv\_uint32x3xm2 (C function), 684  
vsseg3wv\_uint64x3xm1 (C function), 684  
vsseg3wv\_uint64x3xm2 (C function), 685  
vsseg3wv\_uint8x3xm1 (C function), 685  
vsseg3wv\_uint8x3xm2 (C function), 685  
vsseg4bv\_int16x4xm1 (C function), 686  
vsseg4bv\_int16x4xm2 (C function), 686  
vsseg4bv\_int32x4xm1 (C function), 686  
vsseg4bv\_int32x4xm2 (C function), 686  
vsseg4bv\_int64x4xm1 (C function), 686  
vsseg4bv\_int64x4xm2 (C function), 686

vsseg4bv\_int8x4xm1 (C function), 686  
 vsseg4bv\_int8x4xm2 (C function), 686  
 vsseg4bv\_mask\_int16x4xm1 (C function), 686  
 vsseg4bv\_mask\_int16x4xm2 (C function), 686  
 vsseg4bv\_mask\_int32x4xm1 (C function), 686  
 vsseg4bv\_mask\_int32x4xm2 (C function), 686  
 vsseg4bv\_mask\_int64x4xm1 (C function), 686  
 vsseg4bv\_mask\_int64x4xm2 (C function), 687  
 vsseg4bv\_mask\_int8x4xm1 (C function), 687  
 vsseg4bv\_mask\_int8x4xm2 (C function), 687  
 vsseg4bv\_mask\_uint16x4xm1 (C function), 687  
 vsseg4bv\_mask\_uint16x4xm2 (C function), 687  
 vsseg4bv\_mask\_uint32x4xm1 (C function), 687  
 vsseg4bv\_mask\_uint32x4xm2 (C function), 687  
 vsseg4bv\_mask\_uint64x4xm1 (C function), 687  
 vsseg4bv\_mask\_uint64x4xm2 (C function), 687  
 vsseg4bv\_mask\_uint8x4xm1 (C function), 687  
 vsseg4bv\_mask\_uint8x4xm2 (C function), 687  
 vsseg4bv\_uint16x4xm1 (C function), 686  
 vsseg4bv\_uint16x4xm2 (C function), 686  
 vsseg4bv\_uint32x4xm1 (C function), 686  
 vsseg4bv\_uint32x4xm2 (C function), 686  
 vsseg4bv\_uint64x4xm1 (C function), 686  
 vsseg4bv\_uint64x4xm2 (C function), 686  
 vsseg4bv\_uint8x4xm1 (C function), 686  
 vsseg4bv\_uint8x4xm2 (C function), 686  
 vsseg4ev\_float16x4xm1 (C function), 687  
 vsseg4ev\_float16x4xm2 (C function), 687  
 vsseg4ev\_float32x4xm1 (C function), 687  
 vsseg4ev\_float32x4xm2 (C function), 687  
 vsseg4ev\_float64x4xm1 (C function), 687  
 vsseg4ev\_float64x4xm2 (C function), 687  
 vsseg4ev\_int16x4xm1 (C function), 687  
 vsseg4ev\_int16x4xm2 (C function), 687  
 vsseg4ev\_int32x4xm1 (C function), 687  
 vsseg4ev\_int32x4xm2 (C function), 687  
 vsseg4ev\_int64x4xm1 (C function), 688  
 vsseg4ev\_int64x4xm2 (C function), 688  
 vsseg4ev\_int8x4xm1 (C function), 688  
 vsseg4ev\_int8x4xm2 (C function), 688  
 vsseg4ev\_mask\_float16x4xm1 (C function), 688  
 vsseg4ev\_mask\_float16x4xm2 (C function), 688  
 vsseg4ev\_mask\_float32x4xm1 (C function), 688  
 vsseg4ev\_mask\_float32x4xm2 (C function), 688  
 vsseg4ev\_mask\_float64x4xm1 (C function), 688  
 vsseg4ev\_mask\_float64x4xm2 (C function), 688  
 vsseg4ev\_mask\_int16x4xm1 (C function), 688  
 vsseg4ev\_mask\_int16x4xm2 (C function), 688  
 vsseg4ev\_mask\_int32x4xm1 (C function), 688  
 vsseg4ev\_mask\_int32x4xm2 (C function), 688  
 vsseg4ev\_mask\_int64x4xm1 (C function), 688  
 vsseg4ev\_mask\_int64x4xm2 (C function), 688  
 vsseg4ev\_mask\_int8x4xm1 (C function), 689  
 vsseg4ev\_mask\_int8x4xm2 (C function), 689  
 vsseg4ev\_mask\_uint16x4xm1 (C function), 689  
 vsseg4ev\_mask\_uint16x4xm2 (C function), 689  
 vsseg4ev\_mask\_uint32x4xm1 (C function), 689  
 vsseg4ev\_mask\_uint32x4xm2 (C function), 689  
 vsseg4ev\_mask\_uint64x4xm1 (C function), 689  
 vsseg4ev\_mask\_uint64x4xm2 (C function), 689  
 vsseg4ev\_mask\_uint8x4xm1 (C function), 689  
 vsseg4ev\_mask\_uint8x4xm2 (C function), 689  
 vsseg4ev\_uint16x4xm1 (C function), 688  
 vsseg4ev\_uint16x4xm2 (C function), 688  
 vsseg4ev\_uint32x4xm1 (C function), 688  
 vsseg4ev\_uint32x4xm2 (C function), 688  
 vsseg4ev\_uint64x4xm1 (C function), 688  
 vsseg4ev\_uint64x4xm2 (C function), 688  
 vsseg4ev\_uint8x4xm1 (C function), 688  
 vsseg4ev\_uint8x4xm2 (C function), 688  
 vsseg4hv\_int16x4xm1 (C function), 689  
 vsseg4hv\_int16x4xm2 (C function), 689  
 vsseg4hv\_int32x4xm1 (C function), 689  
 vsseg4hv\_int32x4xm2 (C function), 689  
 vsseg4hv\_int64x4xm1 (C function), 689  
 vsseg4hv\_int64x4xm2 (C function), 689  
 vsseg4hv\_int8x4xm1 (C function), 689  
 vsseg4hv\_int8x4xm2 (C function), 689  
 vsseg4hv\_mask\_int16x4xm1 (C function), 690  
 vsseg4hv\_mask\_int16x4xm2 (C function), 690  
 vsseg4hv\_mask\_int32x4xm1 (C function), 690  
 vsseg4hv\_mask\_int32x4xm2 (C function), 690  
 vsseg4hv\_mask\_int64x4xm1 (C function), 690  
 vsseg4hv\_mask\_int64x4xm2 (C function), 690  
 vsseg4hv\_mask\_int8x4xm1 (C function), 690  
 vsseg4hv\_mask\_int8x4xm2 (C function), 690  
 vsseg4hv\_mask\_uint16x4xm1 (C function), 690  
 vsseg4hv\_mask\_uint16x4xm2 (C function), 690  
 vsseg4hv\_mask\_uint32x4xm1 (C function), 690  
 vsseg4hv\_mask\_uint32x4xm2 (C function), 690  
 vsseg4hv\_mask\_uint64x4xm1 (C function), 690  
 vsseg4hv\_mask\_uint64x4xm2 (C function), 690  
 vsseg4hv\_mask\_uint8x4xm1 (C function), 690  
 vsseg4hv\_mask\_uint8x4xm2 (C function), 690  
 vsseg4hv\_uint16x4xm1 (C function), 689  
 vsseg4hv\_uint16x4xm2 (C function), 689  
 vsseg4hv\_uint32x4xm1 (C function), 689  
 vsseg4hv\_uint32x4xm2 (C function), 690  
 vsseg4hv\_uint64x4xm1 (C function), 690  
 vsseg4hv\_uint64x4xm2 (C function), 690  
 vsseg4hv\_uint8x4xm1 (C function), 690  
 vsseg4hv\_uint8x4xm2 (C function), 690  
 vsseg4wv\_int16x4xm1 (C function), 691  
 vsseg4wv\_int16x4xm2 (C function), 691  
 vsseg4wv\_int32x4xm1 (C function), 691  
 vsseg4wv\_int32x4xm2 (C function), 691  
 vsseg4wv\_int64x4xm1 (C function), 691  
 vsseg4wv\_int64x4xm2 (C function), 691



vsseg4wv\_int8x4xm1 (C function), 691  
vsseg4wv\_int8x4xm2 (C function), 691  
vsseg4wv\_mask\_int16x4xm1 (C function), 691  
vsseg4wv\_mask\_int16x4xm2 (C function), 691  
vsseg4wv\_mask\_int32x4xm1 (C function), 691  
vsseg4wv\_mask\_int32x4xm2 (C function), 691  
vsseg4wv\_mask\_int64x4xm1 (C function), 692  
vsseg4wv\_mask\_int64x4xm2 (C function), 692  
vsseg4wv\_mask\_int8x4xm1 (C function), 692  
vsseg4wv\_mask\_int8x4xm2 (C function), 692  
vsseg4wv\_mask\_uint16x4xm1 (C function), 692  
vsseg4wv\_mask\_uint16x4xm2 (C function), 692  
vsseg4wv\_mask\_uint32x4xm1 (C function), 692  
vsseg4wv\_mask\_uint32x4xm2 (C function), 692  
vsseg4wv\_mask\_uint64x4xm1 (C function), 692  
vsseg4wv\_mask\_uint64x4xm2 (C function), 692  
vsseg4wv\_mask\_uint8x4xm1 (C function), 692  
vsseg4wv\_mask\_uint8x4xm2 (C function), 692  
vsseg4wv\_uint16x4xm1 (C function), 691  
vsseg4wv\_uint16x4xm2 (C function), 691  
vsseg4wv\_uint32x4xm1 (C function), 691  
vsseg4wv\_uint32x4xm2 (C function), 691  
vsseg4wv\_uint64x4xm1 (C function), 691  
vsseg4wv\_uint64x4xm2 (C function), 691  
vsseg4wv\_uint8x4xm1 (C function), 691  
vsseg4wv\_uint8x4xm2 (C function), 691  
vsseg5bv\_int16x5xm1 (C function), 692  
vsseg5bv\_int32x5xm1 (C function), 692  
vsseg5bv\_int64x5xm1 (C function), 692  
vsseg5bv\_int8x5xm1 (C function), 692  
vsseg5bv\_mask\_int16x5xm1 (C function), 693  
vsseg5bv\_mask\_int32x5xm1 (C function), 693  
vsseg5bv\_mask\_int64x5xm1 (C function), 693  
vsseg5bv\_mask\_int8x5xm1 (C function), 693  
vsseg5bv\_mask\_uint16x5xm1 (C function), 693  
vsseg5bv\_mask\_uint32x5xm1 (C function), 693  
vsseg5bv\_mask\_uint64x5xm1 (C function), 693  
vsseg5bv\_mask\_uint8x5xm1 (C function), 693  
vsseg5bv\_uint16x5xm1 (C function), 692  
vsseg5bv\_uint32x5xm1 (C function), 692  
vsseg5bv\_uint64x5xm1 (C function), 692  
vsseg5bv\_uint8x5xm1 (C function), 692  
vsseg5ev\_float16x5xm1 (C function), 693  
vsseg5ev\_float32x5xm1 (C function), 693  
vsseg5ev\_float64x5xm1 (C function), 693  
vsseg5ev\_int16x5xm1 (C function), 693  
vsseg5ev\_int32x5xm1 (C function), 693  
vsseg5ev\_int64x5xm1 (C function), 693  
vsseg5ev\_int8x5xm1 (C function), 693  
vsseg5ev\_mask\_float16x5xm1 (C function), 694  
vsseg5ev\_mask\_float32x5xm1 (C function), 694  
vsseg5ev\_mask\_float64x5xm1 (C function), 694  
vsseg5ev\_mask\_int16x5xm1 (C function), 694  
vsseg5ev\_mask\_int32x5xm1 (C function), 694  
vsseg5ev\_mask\_int64x5xm1 (C function), 694  
vsseg5ev\_mask\_int8x5xm1 (C function), 694  
vsseg5ev\_uint16x5xm1 (C function), 693  
vsseg5ev\_uint32x5xm1 (C function), 693  
vsseg5ev\_uint64x5xm1 (C function), 693  
vsseg5ev\_uint8x5xm1 (C function), 694  
vsseg5hv\_int16x5xm1 (C function), 694  
vsseg5hv\_int32x5xm1 (C function), 694  
vsseg5hv\_int64x5xm1 (C function), 694  
vsseg5hv\_int8x5xm1 (C function), 694  
vsseg5hv\_mask\_int16x5xm1 (C function), 695  
vsseg5hv\_mask\_int32x5xm1 (C function), 695  
vsseg5hv\_mask\_int64x5xm1 (C function), 695  
vsseg5hv\_mask\_int8x5xm1 (C function), 695  
vsseg5hv\_mask\_uint16x5xm1 (C function), 695  
vsseg5hv\_mask\_uint32x5xm1 (C function), 695  
vsseg5hv\_mask\_uint64x5xm1 (C function), 695  
vsseg5hv\_mask\_uint8x5xm1 (C function), 695  
vsseg5hv\_uint16x5xm1 (C function), 695  
vsseg5hv\_uint32x5xm1 (C function), 695  
vsseg5hv\_uint64x5xm1 (C function), 695  
vsseg5hv\_uint8x5xm1 (C function), 695  
vsseg5wv\_int16x5xm1 (C function), 695  
vsseg5wv\_int32x5xm1 (C function), 695  
vsseg5wv\_int64x5xm1 (C function), 695  
vsseg5wv\_int8x5xm1 (C function), 695  
vsseg5wv\_mask\_int16x5xm1 (C function), 696  
vsseg5wv\_mask\_int32x5xm1 (C function), 696  
vsseg5wv\_mask\_int64x5xm1 (C function), 696  
vsseg5wv\_mask\_int8x5xm1 (C function), 696  
vsseg5wv\_mask\_uint16x5xm1 (C function), 696  
vsseg5wv\_mask\_uint32x5xm1 (C function), 696  
vsseg5wv\_mask\_uint64x5xm1 (C function), 696  
vsseg5wv\_mask\_uint8x5xm1 (C function), 696  
vsseg5wv\_uint16x5xm1 (C function), 695  
vsseg5wv\_uint32x5xm1 (C function), 695  
vsseg5wv\_uint64x5xm1 (C function), 696  
vsseg5wv\_uint8x5xm1 (C function), 696  
vsseg6bv\_int16x6xm1 (C function), 696  
vsseg6bv\_int32x6xm1 (C function), 696  
vsseg6bv\_int64x6xm1 (C function), 696  
vsseg6bv\_int8x6xm1 (C function), 696  
vsseg6bv\_mask\_int16x6xm1 (C function), 697  
vsseg6bv\_mask\_int32x6xm1 (C function), 697  
vsseg6bv\_mask\_int64x6xm1 (C function), 697  
vsseg6bv\_mask\_int8x6xm1 (C function), 697  
vsseg6bv\_mask\_uint16x6xm1 (C function), 697  
vsseg6bv\_mask\_uint32x6xm1 (C function), 697  
vsseg6bv\_mask\_uint64x6xm1 (C function), 697  
vsseg6bv\_mask\_uint8x6xm1 (C function), 697

vsseg6bv\_uint16x6xm1 (C function), 696  
 vsseg6bv\_uint32x6xm1 (C function), 696  
 vsseg6bv\_uint64x6xm1 (C function), 696  
 vsseg6bv\_uint8x6xm1 (C function), 696  
 vsseg6ev\_float16x6xm1 (C function), 697  
 vsseg6ev\_float32x6xm1 (C function), 697  
 vsseg6ev\_float64x6xm1 (C function), 697  
 vsseg6ev\_int16x6xm1 (C function), 697  
 vsseg6ev\_int32x6xm1 (C function), 697  
 vsseg6ev\_int64x6xm1 (C function), 697  
 vsseg6ev\_int8x6xm1 (C function), 697  
 vsseg6ev\_mask\_float16x6xm1 (C function), 698  
 vsseg6ev\_mask\_float32x6xm1 (C function), 698  
 vsseg6ev\_mask\_float64x6xm1 (C function), 698  
 vsseg6ev\_mask\_int16x6xm1 (C function), 698  
 vsseg6ev\_mask\_int32x6xm1 (C function), 698  
 vsseg6ev\_mask\_int64x6xm1 (C function), 698  
 vsseg6ev\_mask\_int8x6xm1 (C function), 698  
 vsseg6ev\_mask\_uint16x6xm1 (C function), 698  
 vsseg6ev\_mask\_uint32x6xm1 (C function), 698  
 vsseg6ev\_mask\_uint64x6xm1 (C function), 698  
 vsseg6ev\_mask\_uint8x6xm1 (C function), 698  
 vsseg6ev\_uint16x6xm1 (C function), 697  
 vsseg6ev\_uint32x6xm1 (C function), 697  
 vsseg6ev\_uint64x6xm1 (C function), 697  
 vsseg6ev\_uint8x6xm1 (C function), 698  
 vsseg6hv\_int16x6xm1 (C function), 698  
 vsseg6hv\_int32x6xm1 (C function), 698  
 vsseg6hv\_int64x6xm1 (C function), 698  
 vsseg6hv\_int8x6xm1 (C function), 698  
 vsseg6hv\_mask\_int16x6xm1 (C function), 699  
 vsseg6hv\_mask\_int32x6xm1 (C function), 699  
 vsseg6hv\_mask\_int64x6xm1 (C function), 699  
 vsseg6hv\_mask\_int8x6xm1 (C function), 699  
 vsseg6hv\_mask\_uint16x6xm1 (C function), 699  
 vsseg6hv\_mask\_uint32x6xm1 (C function), 699  
 vsseg6hv\_mask\_uint64x6xm1 (C function), 699  
 vsseg6hv\_mask\_uint8x6xm1 (C function), 699  
 vsseg6hv\_uint16x6xm1 (C function), 699  
 vsseg6hv\_uint32x6xm1 (C function), 699  
 vsseg6hv\_uint64x6xm1 (C function), 699  
 vsseg6hv\_uint8x6xm1 (C function), 699  
 vsseg6wv\_int16x6xm1 (C function), 699  
 vsseg6wv\_int32x6xm1 (C function), 699  
 vsseg6wv\_int64x6xm1 (C function), 699  
 vsseg6wv\_int8x6xm1 (C function), 699  
 vsseg6wv\_mask\_int16x6xm1 (C function), 700  
 vsseg6wv\_mask\_int32x6xm1 (C function), 700  
 vsseg6wv\_mask\_int64x6xm1 (C function), 700  
 vsseg6wv\_mask\_int8x6xm1 (C function), 700  
 vsseg6wv\_mask\_uint16x6xm1 (C function), 700  
 vsseg6wv\_mask\_uint32x6xm1 (C function), 700  
 vsseg6wv\_mask\_uint64x6xm1 (C function), 700  
 vsseg6wv\_mask\_uint8x6xm1 (C function), 700  
 vsseg6wv\_uint16x6xm1 (C function), 699  
 vsseg6wv\_uint32x6xm1 (C function), 699  
 vsseg6wv\_uint64x6xm1 (C function), 700  
 vsseg6wv\_uint8x6xm1 (C function), 700  
 vsseg7bv\_int16x7xm1 (C function), 700  
 vsseg7bv\_int32x7xm1 (C function), 700  
 vsseg7bv\_int64x7xm1 (C function), 700  
 vsseg7bv\_int8x7xm1 (C function), 700  
 vsseg7bv\_mask\_int16x7xm1 (C function), 701  
 vsseg7bv\_mask\_int32x7xm1 (C function), 701  
 vsseg7bv\_mask\_int64x7xm1 (C function), 701  
 vsseg7bv\_mask\_int8x7xm1 (C function), 701  
 vsseg7bv\_mask\_uint16x7xm1 (C function), 701  
 vsseg7bv\_mask\_uint32x7xm1 (C function), 701  
 vsseg7bv\_mask\_uint64x7xm1 (C function), 701  
 vsseg7bv\_mask\_uint8x7xm1 (C function), 701  
 vsseg7bv\_uint16x7xm1 (C function), 700  
 vsseg7bv\_uint32x7xm1 (C function), 700  
 vsseg7bv\_uint64x7xm1 (C function), 700  
 vsseg7bv\_uint8x7xm1 (C function), 700  
 vsseg7ev\_float16x7xm1 (C function), 701  
 vsseg7ev\_float32x7xm1 (C function), 701  
 vsseg7ev\_float64x7xm1 (C function), 701  
 vsseg7ev\_int16x7xm1 (C function), 701  
 vsseg7ev\_int32x7xm1 (C function), 701  
 vsseg7ev\_int64x7xm1 (C function), 701  
 vsseg7ev\_int8x7xm1 (C function), 701  
 vsseg7ev\_mask\_float16x7xm1 (C function), 702  
 vsseg7ev\_mask\_float32x7xm1 (C function), 702  
 vsseg7ev\_mask\_float64x7xm1 (C function), 702  
 vsseg7ev\_mask\_int16x7xm1 (C function), 702  
 vsseg7ev\_mask\_int32x7xm1 (C function), 702  
 vsseg7ev\_mask\_int64x7xm1 (C function), 702  
 vsseg7ev\_mask\_int8x7xm1 (C function), 702  
 vsseg7ev\_mask\_uint16x7xm1 (C function), 702  
 vsseg7ev\_mask\_uint32x7xm1 (C function), 702  
 vsseg7ev\_mask\_uint64x7xm1 (C function), 702  
 vsseg7ev\_mask\_uint8x7xm1 (C function), 702  
 vsseg7ev\_uint16x7xm1 (C function), 701  
 vsseg7ev\_uint32x7xm1 (C function), 701  
 vsseg7ev\_uint64x7xm1 (C function), 701  
 vsseg7ev\_uint8x7xm1 (C function), 702  
 vsseg7hv\_int16x7xm1 (C function), 702  
 vsseg7hv\_int32x7xm1 (C function), 702  
 vsseg7hv\_int64x7xm1 (C function), 702  
 vsseg7hv\_int8x7xm1 (C function), 702  
 vsseg7hv\_mask\_int16x7xm1 (C function), 703  
 vsseg7hv\_mask\_int32x7xm1 (C function), 703  
 vsseg7hv\_mask\_int64x7xm1 (C function), 703  
 vsseg7hv\_mask\_int8x7xm1 (C function), 703  
 vsseg7hv\_mask\_uint16x7xm1 (C function), 703  
 vsseg7hv\_mask\_uint32x7xm1 (C function), 703  
 vsseg7hv\_mask\_uint64x7xm1 (C function), 703  
 vsseg7hv\_mask\_uint8x7xm1 (C function), 703

vsseg7hv\_uint16x7xm1 (C function), 703  
vsseg7hv\_uint32x7xm1 (C function), 703  
vsseg7hv\_uint64x7xm1 (C function), 703  
vsseg7hv\_uint8x7xm1 (C function), 703  
vsseg7wv\_int16x7xm1 (C function), 703  
vsseg7wv\_int32x7xm1 (C function), 703  
vsseg7wv\_int64x7xm1 (C function), 703  
vsseg7wv\_int8x7xm1 (C function), 703  
vsseg7wv\_mask\_int16x7xm1 (C function), 704  
vsseg7wv\_mask\_int32x7xm1 (C function), 704  
vsseg7wv\_mask\_int64x7xm1 (C function), 704  
vsseg7wv\_mask\_int8x7xm1 (C function), 704  
vsseg7wv\_mask\_uint16x7xm1 (C function), 704  
vsseg7wv\_mask\_uint32x7xm1 (C function), 704  
vsseg7wv\_mask\_uint64x7xm1 (C function), 704  
vsseg7wv\_mask\_uint8x7xm1 (C function), 704  
vsseg7wv\_uint16x7xm1 (C function), 703  
vsseg7wv\_uint32x7xm1 (C function), 703  
vsseg7wv\_uint64x7xm1 (C function), 704  
vsseg7wv\_uint8x7xm1 (C function), 704  
vsseg8bv\_int16x8xm1 (C function), 704  
vsseg8bv\_int32x8xm1 (C function), 704  
vsseg8bv\_int64x8xm1 (C function), 704  
vsseg8bv\_int8x8xm1 (C function), 704  
vsseg8bv\_mask\_int16x8xm1 (C function), 705  
vsseg8bv\_mask\_int32x8xm1 (C function), 705  
vsseg8bv\_mask\_int64x8xm1 (C function), 705  
vsseg8bv\_mask\_int8x8xm1 (C function), 705  
vsseg8bv\_mask\_uint16x8xm1 (C function), 705  
vsseg8bv\_mask\_uint32x8xm1 (C function), 705  
vsseg8bv\_mask\_uint64x8xm1 (C function), 705  
vsseg8bv\_mask\_uint8x8xm1 (C function), 705  
vsseg8bv\_uint16x8xm1 (C function), 704  
vsseg8bv\_uint32x8xm1 (C function), 704  
vsseg8bv\_uint64x8xm1 (C function), 704  
vsseg8bv\_uint8x8xm1 (C function), 704  
vsseg8ev\_float16x8xm1 (C function), 705  
vsseg8ev\_float32x8xm1 (C function), 705  
vsseg8ev\_float64x8xm1 (C function), 705  
vsseg8ev\_int16x8xm1 (C function), 705  
vsseg8ev\_int32x8xm1 (C function), 705  
vsseg8ev\_int64x8xm1 (C function), 705  
vsseg8ev\_int8x8xm1 (C function), 705  
vsseg8ev\_mask\_float16x8xm1 (C function), 706  
vsseg8ev\_mask\_float32x8xm1 (C function), 706  
vsseg8ev\_mask\_float64x8xm1 (C function), 706  
vsseg8ev\_mask\_int16x8xm1 (C function), 706  
vsseg8ev\_mask\_int32x8xm1 (C function), 706  
vsseg8ev\_mask\_int64x8xm1 (C function), 706  
vsseg8ev\_mask\_int8x8xm1 (C function), 706  
vsseg8ev\_mask\_uint16x8xm1 (C function), 706  
vsseg8ev\_mask\_uint32x8xm1 (C function), 706  
vsseg8ev\_mask\_uint64x8xm1 (C function), 706  
vsseg8ev\_mask\_uint8x8xm1 (C function), 706  
vsseg8ev\_uint16x8xm1 (C function), 705  
vsseg8ev\_uint32x8xm1 (C function), 705  
vsseg8ev\_uint64x8xm1 (C function), 705  
vsseg8ev\_uint8x8xm1 (C function), 706  
vsseg8hv\_int16x8xm1 (C function), 706  
vsseg8hv\_int32x8xm1 (C function), 706  
vsseg8hv\_int64x8xm1 (C function), 706  
vsseg8hv\_int8x8xm1 (C function), 706  
vsseg8hv\_mask\_int16x8xm1 (C function), 707  
vsseg8hv\_mask\_int32x8xm1 (C function), 707  
vsseg8hv\_mask\_int64x8xm1 (C function), 707  
vsseg8hv\_mask\_int8x8xm1 (C function), 707  
vsseg8hv\_mask\_uint16x8xm1 (C function), 707  
vsseg8hv\_mask\_uint32x8xm1 (C function), 707  
vsseg8hv\_mask\_uint64x8xm1 (C function), 707  
vsseg8hv\_mask\_uint8x8xm1 (C function), 707  
vsseg8hv\_uint16x8xm1 (C function), 707  
vsseg8hv\_uint32x8xm1 (C function), 707  
vsseg8hv\_uint64x8xm1 (C function), 707  
vsseg8hv\_uint8x8xm1 (C function), 707  
vsseg8wv\_int16x8xm1 (C function), 707  
vsseg8wv\_int32x8xm1 (C function), 707  
vsseg8wv\_int64x8xm1 (C function), 707  
vsseg8wv\_int8x8xm1 (C function), 707  
vsseg8wv\_mask\_int16x8xm1 (C function), 708  
vsseg8wv\_mask\_int32x8xm1 (C function), 708  
vsseg8wv\_mask\_int64x8xm1 (C function), 708  
vsseg8wv\_mask\_int8x8xm1 (C function), 708  
vsseg8wv\_mask\_uint16x8xm1 (C function), 708  
vsseg8wv\_mask\_uint32x8xm1 (C function), 708  
vsseg8wv\_mask\_uint64x8xm1 (C function), 708  
vsseg8wv\_mask\_uint8x8xm1 (C function), 708  
vsseg8wv\_uint16x8xm1 (C function), 707  
vsseg8wv\_uint32x8xm1 (C function), 707  
vsseg8wv\_uint64x8xm1 (C function), 708  
vsseg8wv\_uint8x8xm1 (C function), 708  
vssev\_float16xm1 (C function), 461  
vssev\_float16xm2 (C function), 461  
vssev\_float16xm4 (C function), 461  
vssev\_float16xm8 (C function), 461  
vssev\_float32xm1 (C function), 461  
vssev\_float32xm2 (C function), 461  
vssev\_float32xm4 (C function), 461  
vssev\_float32xm8 (C function), 461  
vssev\_float64xm1 (C function), 461  
vssev\_float64xm2 (C function), 461  
vssev\_float64xm4 (C function), 461  
vssev\_float64xm8 (C function), 461  
vssev\_int16xm1 (C function), 461  
vssev\_int16xm2 (C function), 461  
vssev\_int16xm4 (C function), 461  
vssev\_int16xm8 (C function), 461  
vssev\_int32xm1 (C function), 461  
vssev\_int32xm2 (C function), 461

vssev\_int32xm4 (C function), 461  
 vssev\_int32xm8 (C function), 461  
 vssev\_int64xm1 (C function), 461  
 vssev\_int64xm2 (C function), 461  
 vssev\_int64xm4 (C function), 461  
 vssev\_int64xm8 (C function), 461  
 vssev\_int8xm1 (C function), 461  
 vssev\_int8xm2 (C function), 461  
 vssev\_int8xm4 (C function), 461  
 vssev\_int8xm8 (C function), 461  
 vssev\_mask\_float16xm1 (C function), 462  
 vssev\_mask\_float16xm2 (C function), 462  
 vssev\_mask\_float16xm4 (C function), 462  
 vssev\_mask\_float16xm8 (C function), 462  
 vssev\_mask\_float32xm1 (C function), 462  
 vssev\_mask\_float32xm2 (C function), 462  
 vssev\_mask\_float32xm4 (C function), 462  
 vssev\_mask\_float32xm8 (C function), 462  
 vssev\_mask\_float64xm1 (C function), 462  
 vssev\_mask\_float64xm2 (C function), 462  
 vssev\_mask\_float64xm4 (C function), 462  
 vssev\_mask\_float64xm8 (C function), 462  
 vssev\_mask\_int16xm1 (C function), 462  
 vssev\_mask\_int16xm2 (C function), 463  
 vssev\_mask\_int16xm4 (C function), 463  
 vssev\_mask\_int16xm8 (C function), 463  
 vssev\_mask\_int32xm1 (C function), 463  
 vssev\_mask\_int32xm2 (C function), 463  
 vssev\_mask\_int32xm4 (C function), 463  
 vssev\_mask\_int32xm8 (C function), 463  
 vssev\_mask\_int64xm1 (C function), 463  
 vssev\_mask\_int64xm2 (C function), 463  
 vssev\_mask\_int64xm4 (C function), 463  
 vssev\_mask\_int64xm8 (C function), 463  
 vssev\_mask\_int8xm1 (C function), 463  
 vssev\_mask\_int8xm2 (C function), 463  
 vssev\_mask\_int8xm4 (C function), 463  
 vssev\_mask\_int8xm8 (C function), 463  
 vssev\_mask\_uint16xm1 (C function), 463  
 vssev\_mask\_uint16xm2 (C function), 463  
 vssev\_mask\_uint16xm4 (C function), 463  
 vssev\_mask\_uint16xm8 (C function), 463  
 vssev\_mask\_uint32xm1 (C function), 463  
 vssev\_mask\_uint32xm2 (C function), 463  
 vssev\_mask\_uint32xm4 (C function), 463  
 vssev\_mask\_uint32xm8 (C function), 463  
 vssev\_mask\_uint64xm1 (C function), 464  
 vssev\_mask\_uint64xm2 (C function), 464  
 vssev\_mask\_uint64xm4 (C function), 464  
 vssev\_mask\_uint64xm8 (C function), 464  
 vssev\_mask\_uint8xm1 (C function), 464  
 vssev\_mask\_uint8xm2 (C function), 464  
 vssev\_mask\_uint8xm4 (C function), 464  
 vssev\_mask\_uint8xm8 (C function), 464  
 vssev\_uint16xm1 (C function), 461  
 vssev\_uint16xm2 (C function), 461  
 vssev\_uint16xm4 (C function), 461  
 vssev\_uint16xm8 (C function), 461  
 vssev\_uint32xm1 (C function), 461  
 vssev\_uint32xm2 (C function), 462  
 vssev\_uint32xm4 (C function), 462  
 vssev\_uint32xm8 (C function), 462  
 vssev\_uint64xm1 (C function), 462  
 vssev\_uint64xm2 (C function), 462  
 vssev\_uint64xm4 (C function), 462  
 vssev\_uint64xm8 (C function), 462  
 vssev\_uint8xm1 (C function), 462  
 vssev\_uint8xm2 (C function), 462  
 vssev\_uint8xm4 (C function), 462  
 vssev\_uint8xm8 (C function), 462  
 vsshv\_int16xm1 (C function), 464  
 vsshv\_int16xm2 (C function), 464  
 vsshv\_int16xm4 (C function), 464  
 vsshv\_int16xm8 (C function), 464  
 vsshv\_int32xm1 (C function), 464  
 vsshv\_int32xm2 (C function), 464  
 vsshv\_int32xm4 (C function), 464  
 vsshv\_int32xm8 (C function), 464  
 vsshv\_int64xm1 (C function), 464  
 vsshv\_int64xm2 (C function), 464  
 vsshv\_int64xm4 (C function), 464  
 vsshv\_int64xm8 (C function), 464  
 vsshv\_int8xm1 (C function), 464  
 vsshv\_int8xm2 (C function), 464  
 vsshv\_int8xm4 (C function), 464  
 vsshv\_int8xm8 (C function), 465  
 vsshv\_mask\_int16xm1 (C function), 465  
 vsshv\_mask\_int16xm2 (C function), 465  
 vsshv\_mask\_int16xm4 (C function), 465  
 vsshv\_mask\_int16xm8 (C function), 465  
 vsshv\_mask\_int32xm1 (C function), 465  
 vsshv\_mask\_int32xm2 (C function), 465  
 vsshv\_mask\_int32xm4 (C function), 465  
 vsshv\_mask\_int32xm8 (C function), 465  
 vsshv\_mask\_int64xm1 (C function), 465  
 vsshv\_mask\_int64xm2 (C function), 466  
 vsshv\_mask\_int64xm4 (C function), 466  
 vsshv\_mask\_int64xm8 (C function), 466  
 vsshv\_mask\_int8xm1 (C function), 466  
 vsshv\_mask\_int8xm2 (C function), 466  
 vsshv\_mask\_int8xm4 (C function), 466  
 vsshv\_mask\_int8xm8 (C function), 466  
 vsshv\_mask\_uint16xm1 (C function), 466  
 vsshv\_mask\_uint16xm2 (C function), 466  
 vsshv\_mask\_uint16xm4 (C function), 466  
 vsshv\_mask\_uint16xm8 (C function), 466  
 vsshv\_mask\_uint32xm1 (C function), 466  
 vsshv\_mask\_uint32xm2 (C function), 466



vsshv\_mask\_uint32xm4 (C function), 466  
vsshv\_mask\_uint32xm8 (C function), 466  
vsshv\_mask\_uint64xm1 (C function), 466  
vsshv\_mask\_uint64xm2 (C function), 466  
vsshv\_mask\_uint64xm4 (C function), 466  
vsshv\_mask\_uint64xm8 (C function), 466  
vsshv\_mask\_uint8xm1 (C function), 466  
vsshv\_mask\_uint8xm2 (C function), 466  
vsshv\_mask\_uint8xm4 (C function), 466  
vsshv\_mask\_uint8xm8 (C function), 466  
vsshv\_uint16xm1 (C function), 465  
vsshv\_uint16xm2 (C function), 465  
vsshv\_uint16xm4 (C function), 465  
vsshv\_uint16xm8 (C function), 465  
vsshv\_uint32xm1 (C function), 465  
vsshv\_uint32xm2 (C function), 465  
vsshv\_uint32xm4 (C function), 465  
vsshv\_uint32xm8 (C function), 465  
vsshv\_uint64xm1 (C function), 465  
vsshv\_uint64xm2 (C function), 465  
vsshv\_uint64xm4 (C function), 465  
vsshv\_uint64xm8 (C function), 465  
vsshv\_uint8xm1 (C function), 465  
vsshv\_uint8xm2 (C function), 465  
vsshv\_uint8xm4 (C function), 465  
vsshv\_uint8xm8 (C function), 465  
vssravi\_int16xm1 (C function), 300  
vssravi\_int16xm2 (C function), 300  
vssravi\_int16xm4 (C function), 300  
vssravi\_int16xm8 (C function), 301  
vssravi\_int32xm1 (C function), 301  
vssravi\_int32xm2 (C function), 301  
vssravi\_int32xm4 (C function), 301  
vssravi\_int32xm8 (C function), 301  
vssravi\_int64xm1 (C function), 301  
vssravi\_int64xm2 (C function), 301  
vssravi\_int64xm4 (C function), 301  
vssravi\_int64xm8 (C function), 301  
vssravi\_int8xm1 (C function), 301  
vssravi\_int8xm2 (C function), 301  
vssravi\_int8xm4 (C function), 301  
vssravi\_int8xm8 (C function), 301  
vssravi\_mask\_int16xm1 (C function), 301  
vssravi\_mask\_int16xm2 (C function), 301  
vssravi\_mask\_int16xm4 (C function), 301  
vssravi\_mask\_int16xm8 (C function), 301  
vssravi\_mask\_int32xm1 (C function), 301  
vssravi\_mask\_int32xm2 (C function), 301  
vssravi\_mask\_int32xm4 (C function), 301  
vssravi\_mask\_int32xm8 (C function), 301  
vssravi\_mask\_int64xm1 (C function), 301  
vssravi\_mask\_int64xm2 (C function), 301  
vssravi\_mask\_int64xm4 (C function), 301  
vssravi\_mask\_int64xm8 (C function), 301  
vssravi\_mask\_int8xm1 (C function), 302  
vssravi\_mask\_int8xm2 (C function), 302  
vssravi\_mask\_int8xm4 (C function), 302  
vssravi\_mask\_int8xm8 (C function), 302  
vssravv\_int16xm1\_uint16xm1 (C function), 302  
vssravv\_int16xm2\_uint16xm2 (C function), 302  
vssravv\_int16xm4\_uint16xm4 (C function), 302  
vssravv\_int16xm8\_uint16xm8 (C function), 302  
vssravv\_int32xm1\_uint32xm1 (C function), 302  
vssravv\_int32xm2\_uint32xm2 (C function), 302  
vssravv\_int32xm4\_uint32xm4 (C function), 302  
vssravv\_int32xm8\_uint32xm8 (C function), 302  
vssravv\_int64xm1\_uint64xm1 (C function), 302  
vssravv\_int64xm2\_uint64xm2 (C function), 302  
vssravv\_int64xm4\_uint64xm4 (C function), 302  
vssravv\_int64xm8\_uint64xm8 (C function), 302  
vssravv\_int8xm1\_uint8xm1 (C function), 302  
vssravv\_int8xm2\_uint8xm2 (C function), 302  
vssravv\_int8xm4\_uint8xm4 (C function), 302  
vssravv\_int8xm8\_uint8xm8 (C function), 302  
vssravv\_mask\_int16xm1\_uint16xm1 (C function), 303  
vssravv\_mask\_int16xm2\_uint16xm2 (C function), 303  
vssravv\_mask\_int16xm4\_uint16xm4 (C function), 303  
vssravv\_mask\_int16xm8\_uint16xm8 (C function), 303  
vssravv\_mask\_int32xm1\_uint32xm1 (C function), 303  
vssravv\_mask\_int32xm2\_uint32xm2 (C function), 303  
vssravv\_mask\_int32xm4\_uint32xm4 (C function), 303  
vssravv\_mask\_int32xm8\_uint32xm8 (C function), 303  
vssravv\_mask\_int64xm1\_uint64xm1 (C function), 303  
vssravv\_mask\_int64xm2\_uint64xm2 (C function), 303  
vssravv\_mask\_int64xm4\_uint64xm4 (C function), 303  
vssravv\_mask\_int64xm8\_uint64xm8 (C function), 303  
vssravv\_mask\_int8xm1\_uint8xm1 (C function), 303  
vssravv\_mask\_int8xm2\_uint8xm2 (C function), 303  
vssravv\_mask\_int8xm4\_uint8xm4 (C function), 303  
vssravv\_mask\_int8xm8\_uint8xm8 (C function), 303  
vssravx\_int16xm1 (C function), 304  
vssravx\_int16xm2 (C function), 304  
vssravx\_int16xm4 (C function), 304  
vssravx\_int16xm8 (C function), 304  
vssravx\_int32xm1 (C function), 304  
vssravx\_int32xm2 (C function), 304  
vssravx\_int32xm4 (C function), 304  
vssravx\_int32xm8 (C function), 304  
vssravx\_int64xm1 (C function), 304  
vssravx\_int64xm2 (C function), 304  
vssravx\_int64xm4 (C function), 304  
vssravx\_int64xm8 (C function), 304  
vssravx\_int8xm1 (C function), 304  
vssravx\_int8xm2 (C function), 304  
vssravx\_int8xm4 (C function), 304  
vssravx\_int8xm8 (C function), 304  
vssravx\_mask\_int16xm1 (C function), 304  
vssravx\_mask\_int16xm2 (C function), 304





vssrlvx\_uint8xm4 (C function), 309  
vssrlvx\_uint8xm8 (C function), 309  
vssseg2bv\_int16x2xm1 (C function), 708  
vssseg2bv\_int16x2xm2 (C function), 708  
vssseg2bv\_int16x2xm4 (C function), 708  
vssseg2bv\_int32x2xm1 (C function), 708  
vssseg2bv\_int32x2xm2 (C function), 708  
vssseg2bv\_int32x2xm4 (C function), 708  
vssseg2bv\_int64x2xm1 (C function), 708  
vssseg2bv\_int64x2xm2 (C function), 708  
vssseg2bv\_int64x2xm4 (C function), 709  
vssseg2bv\_int8x2xm1 (C function), 709  
vssseg2bv\_int8x2xm2 (C function), 709  
vssseg2bv\_int8x2xm4 (C function), 709  
vssseg2bv\_mask\_int16x2xm1 (C function), 709  
vssseg2bv\_mask\_int16x2xm2 (C function), 709  
vssseg2bv\_mask\_int16x2xm4 (C function), 709  
vssseg2bv\_mask\_int32x2xm1 (C function), 709  
vssseg2bv\_mask\_int32x2xm2 (C function), 709  
vssseg2bv\_mask\_int32x2xm4 (C function), 709  
vssseg2bv\_mask\_int64x2xm1 (C function), 710  
vssseg2bv\_mask\_int64x2xm2 (C function), 710  
vssseg2bv\_mask\_int64x2xm4 (C function), 710  
vssseg2bv\_mask\_int8x2xm1 (C function), 710  
vssseg2bv\_mask\_int8x2xm2 (C function), 710  
vssseg2bv\_mask\_int8x2xm4 (C function), 710  
vssseg2bv\_mask\_uint16x2xm1 (C function), 710  
vssseg2bv\_mask\_uint16x2xm2 (C function), 710  
vssseg2bv\_mask\_uint16x2xm4 (C function), 710  
vssseg2bv\_mask\_uint32x2xm1 (C function), 710  
vssseg2bv\_mask\_uint32x2xm2 (C function), 710  
vssseg2bv\_mask\_uint32x2xm4 (C function), 710  
vssseg2bv\_mask\_uint64x2xm1 (C function), 710  
vssseg2bv\_mask\_uint64x2xm2 (C function), 710  
vssseg2bv\_mask\_uint64x2xm4 (C function), 710  
vssseg2bv\_mask\_uint8x2xm1 (C function), 710  
vssseg2bv\_mask\_uint8x2xm2 (C function), 710  
vssseg2bv\_mask\_uint8x2xm4 (C function), 710  
vssseg2bv\_uint16x2xm1 (C function), 709  
vssseg2bv\_uint16x2xm2 (C function), 709  
vssseg2bv\_uint16x2xm4 (C function), 709  
vssseg2bv\_uint32x2xm1 (C function), 709  
vssseg2bv\_uint32x2xm2 (C function), 709  
vssseg2bv\_uint32x2xm4 (C function), 709  
vssseg2bv\_uint64x2xm1 (C function), 709  
vssseg2bv\_uint64x2xm2 (C function), 709  
vssseg2bv\_uint64x2xm4 (C function), 709  
vssseg2bv\_uint8x2xm1 (C function), 709  
vssseg2bv\_uint8x2xm2 (C function), 709  
vssseg2bv\_uint8x2xm4 (C function), 709  
vssseg2ev\_float16x2xm1 (C function), 711  
vssseg2ev\_float16x2xm2 (C function), 711  
vssseg2ev\_float16x2xm4 (C function), 711  
vssseg2ev\_float32x2xm1 (C function), 711  
vssseg2ev\_float32x2xm2 (C function), 711  
vssseg2ev\_float32x2xm4 (C function), 711  
vssseg2ev\_int16x2xm1 (C function), 711  
vssseg2ev\_int16x2xm2 (C function), 711  
vssseg2ev\_int16x2xm4 (C function), 711  
vssseg2ev\_int32x2xm1 (C function), 711  
vssseg2ev\_int32x2xm2 (C function), 711  
vssseg2ev\_int32x2xm4 (C function), 711  
vssseg2ev\_int64x2xm1 (C function), 711  
vssseg2ev\_int64x2xm2 (C function), 711  
vssseg2ev\_int64x2xm4 (C function), 711  
vssseg2ev\_int8x2xm1 (C function), 711  
vssseg2ev\_int8x2xm2 (C function), 711  
vssseg2ev\_int8x2xm4 (C function), 711  
vssseg2ev\_mask\_float16x2xm1 (C function), 712  
vssseg2ev\_mask\_float16x2xm2 (C function), 712  
vssseg2ev\_mask\_float16x2xm4 (C function), 712  
vssseg2ev\_mask\_float32x2xm1 (C function), 712  
vssseg2ev\_mask\_float32x2xm2 (C function), 712  
vssseg2ev\_mask\_float32x2xm4 (C function), 712  
vssseg2ev\_mask\_float64x2xm1 (C function), 712  
vssseg2ev\_mask\_float64x2xm2 (C function), 712  
vssseg2ev\_mask\_float64x2xm4 (C function), 712  
vssseg2ev\_mask\_int16x2xm1 (C function), 712  
vssseg2ev\_mask\_int16x2xm2 (C function), 712  
vssseg2ev\_mask\_int16x2xm4 (C function), 712  
vssseg2ev\_mask\_int32x2xm1 (C function), 712  
vssseg2ev\_mask\_int32x2xm2 (C function), 712  
vssseg2ev\_mask\_int32x2xm4 (C function), 712  
vssseg2ev\_mask\_int64x2xm1 (C function), 712  
vssseg2ev\_mask\_int64x2xm2 (C function), 713  
vssseg2ev\_mask\_int64x2xm4 (C function), 713  
vssseg2ev\_mask\_int8x2xm1 (C function), 713  
vssseg2ev\_mask\_int8x2xm2 (C function), 713  
vssseg2ev\_mask\_int8x2xm4 (C function), 713  
vssseg2ev\_mask\_uint16x2xm1 (C function), 713  
vssseg2ev\_mask\_uint16x2xm2 (C function), 713  
vssseg2ev\_mask\_uint16x2xm4 (C function), 713  
vssseg2ev\_mask\_uint32x2xm1 (C function), 713  
vssseg2ev\_mask\_uint32x2xm2 (C function), 713  
vssseg2ev\_mask\_uint32x2xm4 (C function), 713  
vssseg2ev\_mask\_uint64x2xm1 (C function), 713  
vssseg2ev\_mask\_uint64x2xm2 (C function), 713  
vssseg2ev\_mask\_uint64x2xm4 (C function), 713  
vssseg2ev\_mask\_uint8x2xm1 (C function), 713  
vssseg2ev\_mask\_uint8x2xm2 (C function), 713  
vssseg2ev\_mask\_uint8x2xm4 (C function), 713  
vssseg2ev\_uint16x2xm1 (C function), 711  
vssseg2ev\_uint16x2xm2 (C function), 711  
vssseg2ev\_uint16x2xm4 (C function), 711  
vssseg2ev\_uint32x2xm1 (C function), 711



vssseg3bv\_int64x3xm1 (C function), 718  
vssseg3bv\_int64x3xm2 (C function), 718  
vssseg3bv\_int8x3xm1 (C function), 718  
vssseg3bv\_int8x3xm2 (C function), 718  
vssseg3bv\_mask\_int16x3xm1 (C function), 719  
vssseg3bv\_mask\_int16x3xm2 (C function), 719  
vssseg3bv\_mask\_int32x3xm1 (C function), 719  
vssseg3bv\_mask\_int32x3xm2 (C function), 719  
vssseg3bv\_mask\_int64x3xm1 (C function), 719  
vssseg3bv\_mask\_int64x3xm2 (C function), 719  
vssseg3bv\_mask\_int8x3xm1 (C function), 719  
vssseg3bv\_mask\_int8x3xm2 (C function), 719  
vssseg3bv\_mask\_uint16x3xm1 (C function), 719  
vssseg3bv\_mask\_uint16x3xm2 (C function), 719  
vssseg3bv\_mask\_uint32x3xm1 (C function), 719  
vssseg3bv\_mask\_uint32x3xm2 (C function), 719  
vssseg3bv\_mask\_uint64x3xm1 (C function), 719  
vssseg3bv\_mask\_uint64x3xm2 (C function), 719  
vssseg3bv\_mask\_uint8x3xm1 (C function), 719  
vssseg3bv\_mask\_uint8x3xm2 (C function), 719  
vssseg3bv\_uint16x3xm1 (C function), 718  
vssseg3bv\_uint16x3xm2 (C function), 718  
vssseg3bv\_uint32x3xm1 (C function), 718  
vssseg3bv\_uint32x3xm2 (C function), 718  
vssseg3bv\_uint64x3xm1 (C function), 719  
vssseg3bv\_uint64x3xm2 (C function), 719  
vssseg3bv\_uint8x3xm1 (C function), 719  
vssseg3bv\_uint8x3xm2 (C function), 719  
vssseg3ev\_float16x3xm1 (C function), 720  
vssseg3ev\_float16x3xm2 (C function), 720  
vssseg3ev\_float32x3xm1 (C function), 720  
vssseg3ev\_float32x3xm2 (C function), 720  
vssseg3ev\_float64x3xm1 (C function), 720  
vssseg3ev\_float64x3xm2 (C function), 720  
vssseg3ev\_int16x3xm1 (C function), 720  
vssseg3ev\_int16x3xm2 (C function), 720  
vssseg3ev\_int32x3xm1 (C function), 720  
vssseg3ev\_int32x3xm2 (C function), 720  
vssseg3ev\_int64x3xm1 (C function), 720  
vssseg3ev\_int64x3xm2 (C function), 720  
vssseg3ev\_int8x3xm1 (C function), 720  
vssseg3ev\_int8x3xm2 (C function), 720  
vssseg3ev\_mask\_float16x3xm1 (C function), 721  
vssseg3ev\_mask\_float16x3xm2 (C function), 721  
vssseg3ev\_mask\_float32x3xm1 (C function), 721  
vssseg3ev\_mask\_float32x3xm2 (C function), 721  
vssseg3ev\_mask\_float64x3xm1 (C function), 721  
vssseg3ev\_mask\_float64x3xm2 (C function), 721  
vssseg3ev\_mask\_int16x3xm1 (C function), 721  
vssseg3ev\_mask\_int16x3xm2 (C function), 721  
vssseg3ev\_mask\_int32x3xm1 (C function), 721  
vssseg3ev\_mask\_int32x3xm2 (C function), 721  
vssseg3ev\_mask\_int64x3xm1 (C function), 721  
vssseg3ev\_mask\_int64x3xm2 (C function), 721  
vssseg3ev\_mask\_int8x3xm1 (C function), 721  
vssseg3ev\_mask\_int8x3xm2 (C function), 721  
vssseg3ev\_mask\_uint16x3xm1 (C function), 721  
vssseg3ev\_mask\_uint16x3xm2 (C function), 721  
vssseg3ev\_mask\_uint32x3xm1 (C function), 721  
vssseg3ev\_mask\_uint32x3xm2 (C function), 721  
vssseg3ev\_mask\_uint64x3xm1 (C function), 721  
vssseg3ev\_mask\_uint64x3xm2 (C function), 721  
vssseg3ev\_mask\_uint8x3xm1 (C function), 721  
vssseg3ev\_mask\_uint8x3xm2 (C function), 721  
vssseg3ev\_uint16x3xm1 (C function), 720  
vssseg3ev\_uint16x3xm2 (C function), 720  
vssseg3ev\_uint32x3xm1 (C function), 720  
vssseg3ev\_uint32x3xm2 (C function), 720  
vssseg3ev\_uint64x3xm1 (C function), 720  
vssseg3ev\_uint64x3xm2 (C function), 720  
vssseg3ev\_uint8x3xm1 (C function), 720  
vssseg3ev\_uint8x3xm2 (C function), 720  
vssseg3hv\_int16x3xm1 (C function), 722  
vssseg3hv\_int16x3xm2 (C function), 722  
vssseg3hv\_int32x3xm1 (C function), 722  
vssseg3hv\_int32x3xm2 (C function), 722  
vssseg3hv\_int64x3xm1 (C function), 722  
vssseg3hv\_int64x3xm2 (C function), 722  
vssseg3hv\_int8x3xm1 (C function), 722  
vssseg3hv\_int8x3xm2 (C function), 722  
vssseg3hv\_mask\_int16x3xm1 (C function), 723  
vssseg3hv\_mask\_int16x3xm2 (C function), 723  
vssseg3hv\_mask\_int32x3xm1 (C function), 723  
vssseg3hv\_mask\_int32x3xm2 (C function), 723  
vssseg3hv\_mask\_int64x3xm1 (C function), 723  
vssseg3hv\_mask\_int64x3xm2 (C function), 723  
vssseg3hv\_mask\_int8x3xm1 (C function), 723  
vssseg3hv\_mask\_int8x3xm2 (C function), 723  
vssseg3hv\_mask\_uint16x3xm1 (C function), 723  
vssseg3hv\_mask\_uint16x3xm2 (C function), 723  
vssseg3hv\_mask\_uint32x3xm1 (C function), 723  
vssseg3hv\_mask\_uint32x3xm2 (C function), 723  
vssseg3hv\_mask\_uint64x3xm1 (C function), 723  
vssseg3hv\_mask\_uint64x3xm2 (C function), 723  
vssseg3hv\_mask\_uint8x3xm1 (C function), 723  
vssseg3hv\_mask\_uint8x3xm2 (C function), 723  
vssseg3hv\_uint16x3xm1 (C function), 722  
vssseg3hv\_uint16x3xm2 (C function), 722  
vssseg3hv\_uint32x3xm1 (C function), 722  
vssseg3hv\_uint32x3xm2 (C function), 722  
vssseg3hv\_uint64x3xm1 (C function), 722  
vssseg3hv\_uint64x3xm2 (C function), 722  
vssseg3hv\_uint8x3xm1 (C function), 722  
vssseg3hv\_uint8x3xm2 (C function), 722  
vssseg3wv\_int16x3xm1 (C function), 724  
vssseg3wv\_int16x3xm2 (C function), 724  
vssseg3wv\_int32x3xm1 (C function), 724  
vssseg3wv\_int32x3xm2 (C function), 724

vssseg3wv\_int64x3xm1 (C function), 724  
 vssseg3wv\_int64x3xm2 (C function), 724  
 vssseg3wv\_int8x3xm1 (C function), 724  
 vssseg3wv\_int8x3xm2 (C function), 724  
 vssseg3wv\_mask\_int16x3xm1 (C function), 724  
 vssseg3wv\_mask\_int16x3xm2 (C function), 724  
 vssseg3wv\_mask\_int32x3xm1 (C function), 724  
 vssseg3wv\_mask\_int32x3xm2 (C function), 724  
 vssseg3wv\_mask\_int64x3xm1 (C function), 725  
 vssseg3wv\_mask\_int64x3xm2 (C function), 725  
 vssseg3wv\_mask\_int8x3xm1 (C function), 725  
 vssseg3wv\_mask\_int8x3xm2 (C function), 725  
 vssseg3wv\_mask\_uint16x3xm1 (C function), 725  
 vssseg3wv\_mask\_uint16x3xm2 (C function), 725  
 vssseg3wv\_mask\_uint32x3xm1 (C function), 725  
 vssseg3wv\_mask\_uint32x3xm2 (C function), 725  
 vssseg3wv\_mask\_uint64x3xm1 (C function), 725  
 vssseg3wv\_mask\_uint64x3xm2 (C function), 725  
 vssseg3wv\_mask\_uint8x3xm1 (C function), 725  
 vssseg3wv\_mask\_uint8x3xm2 (C function), 725  
 vssseg3wv\_uint16x3xm1 (C function), 724  
 vssseg3wv\_uint16x3xm2 (C function), 724  
 vssseg3wv\_uint32x3xm1 (C function), 724  
 vssseg3wv\_uint32x3xm2 (C function), 724  
 vssseg3wv\_uint64x3xm1 (C function), 724  
 vssseg3wv\_uint64x3xm2 (C function), 724  
 vssseg3wv\_uint8x3xm1 (C function), 724  
 vssseg3wv\_uint8x3xm2 (C function), 724  
 vssseg4bv\_int16x4xm1 (C function), 725  
 vssseg4bv\_int16x4xm2 (C function), 725  
 vssseg4bv\_int32x4xm1 (C function), 725  
 vssseg4bv\_int32x4xm2 (C function), 725  
 vssseg4bv\_int64x4xm1 (C function), 725  
 vssseg4bv\_int64x4xm2 (C function), 725  
 vssseg4bv\_int8x4xm1 (C function), 725  
 vssseg4bv\_int8x4xm2 (C function), 725  
 vssseg4bv\_mask\_int16x4xm1 (C function), 726  
 vssseg4bv\_mask\_int16x4xm2 (C function), 726  
 vssseg4bv\_mask\_int32x4xm1 (C function), 726  
 vssseg4bv\_mask\_int32x4xm2 (C function), 726  
 vssseg4bv\_mask\_int64x4xm1 (C function), 726  
 vssseg4bv\_mask\_int64x4xm2 (C function), 726  
 vssseg4bv\_mask\_int8x4xm1 (C function), 726  
 vssseg4bv\_mask\_int8x4xm2 (C function), 726  
 vssseg4bv\_mask\_uint16x4xm1 (C function), 726  
 vssseg4bv\_mask\_uint16x4xm2 (C function), 726  
 vssseg4bv\_mask\_uint32x4xm1 (C function), 726  
 vssseg4bv\_mask\_uint32x4xm2 (C function), 726  
 vssseg4bv\_mask\_uint64x4xm1 (C function), 727  
 vssseg4bv\_mask\_uint64x4xm2 (C function), 727  
 vssseg4bv\_mask\_uint8x4xm1 (C function), 727  
 vssseg4bv\_mask\_uint8x4xm2 (C function), 727  
 vssseg4bv\_uint16x4xm1 (C function), 726  
 vssseg4bv\_uint16x4xm2 (C function), 726  
 vssseg4bv\_uint32x4xm1 (C function), 726  
 vssseg4bv\_uint32x4xm2 (C function), 726  
 vssseg4bv\_uint64x4xm1 (C function), 726  
 vssseg4bv\_uint64x4xm2 (C function), 726  
 vssseg4bv\_uint8x4xm1 (C function), 726  
 vssseg4bv\_uint8x4xm2 (C function), 726  
 vssseg4ev\_float16x4xm1 (C function), 727  
 vssseg4ev\_float16x4xm2 (C function), 727  
 vssseg4ev\_float32x4xm1 (C function), 727  
 vssseg4ev\_float32x4xm2 (C function), 727  
 vssseg4ev\_float64x4xm1 (C function), 727  
 vssseg4ev\_float64x4xm2 (C function), 727  
 vssseg4ev\_int16x4xm1 (C function), 727  
 vssseg4ev\_int16x4xm2 (C function), 727  
 vssseg4ev\_int32x4xm1 (C function), 727  
 vssseg4ev\_int32x4xm2 (C function), 727  
 vssseg4ev\_int64x4xm1 (C function), 727  
 vssseg4ev\_int64x4xm2 (C function), 727  
 vssseg4ev\_int8x4xm1 (C function), 727  
 vssseg4ev\_int8x4xm2 (C function), 727  
 vssseg4ev\_mask\_float16x4xm1 (C function), 728  
 vssseg4ev\_mask\_float16x4xm2 (C function), 728  
 vssseg4ev\_mask\_float32x4xm1 (C function), 728  
 vssseg4ev\_mask\_float32x4xm2 (C function), 728  
 vssseg4ev\_mask\_float64x4xm1 (C function), 728  
 vssseg4ev\_mask\_float64x4xm2 (C function), 728  
 vssseg4ev\_mask\_int16x4xm1 (C function), 728  
 vssseg4ev\_mask\_int16x4xm2 (C function), 728  
 vssseg4ev\_mask\_int32x4xm1 (C function), 728  
 vssseg4ev\_mask\_int32x4xm2 (C function), 728  
 vssseg4ev\_mask\_int64x4xm1 (C function), 728  
 vssseg4ev\_mask\_int64x4xm2 (C function), 728  
 vssseg4ev\_mask\_int8x4xm1 (C function), 728  
 vssseg4ev\_mask\_int8x4xm2 (C function), 728  
 vssseg4ev\_mask\_uint16x4xm1 (C function), 728  
 vssseg4ev\_mask\_uint16x4xm2 (C function), 729  
 vssseg4ev\_mask\_uint32x4xm1 (C function), 729  
 vssseg4ev\_mask\_uint32x4xm2 (C function), 729  
 vssseg4ev\_mask\_uint64x4xm1 (C function), 729  
 vssseg4ev\_mask\_uint64x4xm2 (C function), 729  
 vssseg4ev\_mask\_uint8x4xm1 (C function), 729  
 vssseg4ev\_mask\_uint8x4xm2 (C function), 729  
 vssseg4ev\_uint16x4xm1 (C function), 727  
 vssseg4ev\_uint16x4xm2 (C function), 727  
 vssseg4ev\_uint32x4xm1 (C function), 727  
 vssseg4ev\_uint32x4xm2 (C function), 728  
 vssseg4ev\_uint64x4xm1 (C function), 728  
 vssseg4ev\_uint64x4xm2 (C function), 728  
 vssseg4ev\_uint8x4xm1 (C function), 728  
 vssseg4ev\_uint8x4xm2 (C function), 728  
 vssseg4hv\_int16x4xm1 (C function), 729  
 vssseg4hv\_int16x4xm2 (C function), 729  
 vssseg4hv\_int32x4xm1 (C function), 729  
 vssseg4hv\_int32x4xm2 (C function), 729



vssseg4hv\_int64x4xm1 (C function), 729  
vssseg4hv\_int64x4xm2 (C function), 729  
vssseg4hv\_int8x4xm1 (C function), 729  
vssseg4hv\_int8x4xm2 (C function), 729  
vssseg4hv\_mask\_int16x4xm1 (C function), 730  
vssseg4hv\_mask\_int16x4xm2 (C function), 730  
vssseg4hv\_mask\_int32x4xm1 (C function), 730  
vssseg4hv\_mask\_int32x4xm2 (C function), 730  
vssseg4hv\_mask\_int64x4xm1 (C function), 730  
vssseg4hv\_mask\_int64x4xm2 (C function), 730  
vssseg4hv\_mask\_int8x4xm1 (C function), 730  
vssseg4hv\_mask\_int8x4xm2 (C function), 730  
vssseg4hv\_mask\_uint16x4xm1 (C function), 730  
vssseg4hv\_mask\_uint16x4xm2 (C function), 730  
vssseg4hv\_mask\_uint32x4xm1 (C function), 730  
vssseg4hv\_mask\_uint32x4xm2 (C function), 730  
vssseg4hv\_mask\_uint64x4xm1 (C function), 730  
vssseg4hv\_mask\_uint64x4xm2 (C function), 730  
vssseg4hv\_mask\_uint8x4xm1 (C function), 730  
vssseg4hv\_mask\_uint8x4xm2 (C function), 730  
vssseg4hv\_uint16x4xm1 (C function), 729  
vssseg4hv\_uint16x4xm2 (C function), 729  
vssseg4hv\_uint32x4xm1 (C function), 729  
vssseg4hv\_uint32x4xm2 (C function), 729  
vssseg4hv\_uint64x4xm1 (C function), 729  
vssseg4hv\_uint64x4xm2 (C function), 730  
vssseg4hv\_uint8x4xm1 (C function), 730  
vssseg4hv\_uint8x4xm2 (C function), 730  
vssseg4wv\_int16x4xm1 (C function), 731  
vssseg4wv\_int16x4xm2 (C function), 731  
vssseg4wv\_int32x4xm1 (C function), 731  
vssseg4wv\_int32x4xm2 (C function), 731  
vssseg4wv\_int64x4xm1 (C function), 731  
vssseg4wv\_int64x4xm2 (C function), 731  
vssseg4wv\_int8x4xm1 (C function), 731  
vssseg4wv\_int8x4xm2 (C function), 731  
vssseg4wv\_mask\_int16x4xm1 (C function), 731  
vssseg4wv\_mask\_int16x4xm2 (C function), 732  
vssseg4wv\_mask\_int32x4xm1 (C function), 732  
vssseg4wv\_mask\_int32x4xm2 (C function), 732  
vssseg4wv\_mask\_int64x4xm1 (C function), 732  
vssseg4wv\_mask\_int64x4xm2 (C function), 732  
vssseg4wv\_mask\_int8x4xm1 (C function), 732  
vssseg4wv\_mask\_int8x4xm2 (C function), 732  
vssseg4wv\_mask\_uint16x4xm1 (C function), 732  
vssseg4wv\_mask\_uint16x4xm2 (C function), 732  
vssseg4wv\_mask\_uint32x4xm1 (C function), 732  
vssseg4wv\_mask\_uint32x4xm2 (C function), 732  
vssseg4wv\_mask\_uint64x4xm1 (C function), 732  
vssseg4wv\_mask\_uint64x4xm2 (C function), 732  
vssseg4wv\_mask\_uint8x4xm1 (C function), 732  
vssseg4wv\_mask\_uint8x4xm2 (C function), 732  
vssseg4wv\_uint16x4xm1 (C function), 731  
vssseg4wv\_uint16x4xm2 (C function), 731  
vssseg4wv\_uint32x4xm1 (C function), 731  
vssseg4wv\_uint32x4xm2 (C function), 731  
vssseg4wv\_uint64x4xm1 (C function), 731  
vssseg4wv\_uint64x4xm2 (C function), 731  
vssseg4wv\_uint8x4xm1 (C function), 731  
vssseg4wv\_uint8x4xm2 (C function), 731  
vssseg5bv\_int16x5xm1 (C function), 732  
vssseg5bv\_int32x5xm1 (C function), 732  
vssseg5bv\_int64x5xm1 (C function), 732  
vssseg5bv\_int8x5xm1 (C function), 733  
vssseg5bv\_mask\_int16x5xm1 (C function), 733  
vssseg5bv\_mask\_int32x5xm1 (C function), 733  
vssseg5bv\_mask\_int64x5xm1 (C function), 733  
vssseg5bv\_mask\_int8x5xm1 (C function), 733  
vssseg5bv\_mask\_uint16x5xm1 (C function), 733  
vssseg5bv\_mask\_uint32x5xm1 (C function), 733  
vssseg5bv\_mask\_uint64x5xm1 (C function), 733  
vssseg5bv\_mask\_uint8x5xm1 (C function), 733  
vssseg5bv\_uint16x5xm1 (C function), 733  
vssseg5bv\_uint32x5xm1 (C function), 733  
vssseg5bv\_uint64x5xm1 (C function), 733  
vssseg5bv\_uint8x5xm1 (C function), 733  
vssseg5ev\_float16x5xm1 (C function), 733  
vssseg5ev\_float32x5xm1 (C function), 733  
vssseg5ev\_float64x5xm1 (C function), 734  
vssseg5ev\_int16x5xm1 (C function), 734  
vssseg5ev\_int32x5xm1 (C function), 734  
vssseg5ev\_int64x5xm1 (C function), 734  
vssseg5ev\_int8x5xm1 (C function), 734  
vssseg5ev\_mask\_float16x5xm1 (C function), 734  
vssseg5ev\_mask\_float32x5xm1 (C function), 734  
vssseg5ev\_mask\_float64x5xm1 (C function), 734  
vssseg5ev\_mask\_int16x5xm1 (C function), 734  
vssseg5ev\_mask\_int32x5xm1 (C function), 734  
vssseg5ev\_mask\_int64x5xm1 (C function), 734  
vssseg5ev\_mask\_int8x5xm1 (C function), 734  
vssseg5ev\_mask\_uint16x5xm1 (C function), 734  
vssseg5ev\_mask\_uint32x5xm1 (C function), 734  
vssseg5ev\_mask\_uint64x5xm1 (C function), 734  
vssseg5ev\_mask\_uint8x5xm1 (C function), 734  
vssseg5ev\_uint16x5xm1 (C function), 734  
vssseg5ev\_uint32x5xm1 (C function), 734  
vssseg5ev\_uint64x5xm1 (C function), 734  
vssseg5ev\_uint8x5xm1 (C function), 734  
vssseg5hv\_int16x5xm1 (C function), 735  
vssseg5hv\_int32x5xm1 (C function), 735  
vssseg5hv\_int64x5xm1 (C function), 735  
vssseg5hv\_int8x5xm1 (C function), 735  
vssseg5hv\_mask\_int16x5xm1 (C function), 735  
vssseg5hv\_mask\_int32x5xm1 (C function), 735  
vssseg5hv\_mask\_int64x5xm1 (C function), 735  
vssseg5hv\_mask\_int8x5xm1 (C function), 735  
vssseg5hv\_mask\_uint16x5xm1 (C function), 735  
vssseg5hv\_mask\_uint32x5xm1 (C function), 735

vssseg5hv\_mask\_uint64x5xm1 (C function), 735  
 vssseg5hv\_mask\_uint8x5xm1 (C function), 735  
 vssseg5hv\_uint16x5xm1 (C function), 735  
 vssseg5hv\_uint32x5xm1 (C function), 735  
 vssseg5hv\_uint64x5xm1 (C function), 735  
 vssseg5hv\_uint8x5xm1 (C function), 735  
 vssseg5wv\_int16x5xm1 (C function), 736  
 vssseg5wv\_int32x5xm1 (C function), 736  
 vssseg5wv\_int64x5xm1 (C function), 736  
 vssseg5wv\_int8x5xm1 (C function), 736  
 vssseg5wv\_mask\_int16x5xm1 (C function), 736  
 vssseg5wv\_mask\_int32x5xm1 (C function), 736  
 vssseg5wv\_mask\_int64x5xm1 (C function), 736  
 vssseg5wv\_mask\_int8x5xm1 (C function), 736  
 vssseg5wv\_mask\_uint16x5xm1 (C function), 736  
 vssseg5wv\_mask\_uint32x5xm1 (C function), 736  
 vssseg5wv\_mask\_uint64x5xm1 (C function), 736  
 vssseg5wv\_mask\_uint8x5xm1 (C function), 736  
 vssseg5wv\_uint16x5xm1 (C function), 736  
 vssseg5wv\_uint32x5xm1 (C function), 736  
 vssseg5wv\_uint64x5xm1 (C function), 736  
 vssseg5wv\_uint8x5xm1 (C function), 736  
 vssseg6bv\_int16x6xm1 (C function), 737  
 vssseg6bv\_int32x6xm1 (C function), 737  
 vssseg6bv\_int64x6xm1 (C function), 737  
 vssseg6bv\_int8x6xm1 (C function), 737  
 vssseg6bv\_mask\_int16x6xm1 (C function), 737  
 vssseg6bv\_mask\_int32x6xm1 (C function), 737  
 vssseg6bv\_mask\_int64x6xm1 (C function), 737  
 vssseg6bv\_mask\_int8x6xm1 (C function), 737  
 vssseg6bv\_mask\_uint16x6xm1 (C function), 737  
 vssseg6bv\_mask\_uint32x6xm1 (C function), 737  
 vssseg6bv\_mask\_uint64x6xm1 (C function), 737  
 vssseg6bv\_mask\_uint8x6xm1 (C function), 737  
 vssseg6bv\_uint16x6xm1 (C function), 737  
 vssseg6bv\_uint32x6xm1 (C function), 737  
 vssseg6bv\_uint64x6xm1 (C function), 737  
 vssseg6bv\_uint8x6xm1 (C function), 737  
 vssseg6ev\_float16x6xm1 (C function), 738  
 vssseg6ev\_float32x6xm1 (C function), 738  
 vssseg6ev\_float64x6xm1 (C function), 738  
 vssseg6ev\_int16x6xm1 (C function), 738  
 vssseg6ev\_int32x6xm1 (C function), 738  
 vssseg6ev\_int64x6xm1 (C function), 738  
 vssseg6ev\_int8x6xm1 (C function), 738  
 vssseg6ev\_mask\_float16x6xm1 (C function), 738  
 vssseg6ev\_mask\_float32x6xm1 (C function), 738  
 vssseg6ev\_mask\_float64x6xm1 (C function), 738  
 vssseg6ev\_mask\_int16x6xm1 (C function), 738  
 vssseg6ev\_mask\_int32x6xm1 (C function), 738  
 vssseg6ev\_mask\_int64x6xm1 (C function), 739  
 vssseg6ev\_mask\_int8x6xm1 (C function), 739  
 vssseg6ev\_mask\_uint16x6xm1 (C function), 739  
 vssseg6ev\_mask\_uint32x6xm1 (C function), 739  
 vssseg6ev\_mask\_uint64x6xm1 (C function), 739  
 vssseg6ev\_mask\_uint8x6xm1 (C function), 739  
 vssseg6hv\_int16x6xm1 (C function), 739  
 vssseg6hv\_int32x6xm1 (C function), 739  
 vssseg6hv\_int64x6xm1 (C function), 739  
 vssseg6hv\_int8x6xm1 (C function), 739  
 vssseg6hv\_mask\_int16x6xm1 (C function), 739  
 vssseg6hv\_mask\_int32x6xm1 (C function), 739  
 vssseg6hv\_mask\_int64x6xm1 (C function), 740  
 vssseg6hv\_mask\_int8x6xm1 (C function), 740  
 vssseg6hv\_mask\_uint16x6xm1 (C function), 740  
 vssseg6hv\_mask\_uint32x6xm1 (C function), 740  
 vssseg6hv\_mask\_uint64x6xm1 (C function), 740  
 vssseg6hv\_mask\_uint8x6xm1 (C function), 740  
 vssseg6hv\_uint16x6xm1 (C function), 739  
 vssseg6hv\_uint32x6xm1 (C function), 739  
 vssseg6hv\_uint64x6xm1 (C function), 739  
 vssseg6hv\_uint8x6xm1 (C function), 739  
 vssseg6wv\_int16x6xm1 (C function), 740  
 vssseg6wv\_int32x6xm1 (C function), 740  
 vssseg6wv\_int64x6xm1 (C function), 740  
 vssseg6wv\_int8x6xm1 (C function), 740  
 vssseg6wv\_mask\_int16x6xm1 (C function), 740  
 vssseg6wv\_mask\_int32x6xm1 (C function), 740  
 vssseg6wv\_mask\_int64x6xm1 (C function), 741  
 vssseg6wv\_mask\_int8x6xm1 (C function), 741  
 vssseg6wv\_mask\_uint16x6xm1 (C function), 741  
 vssseg6wv\_mask\_uint32x6xm1 (C function), 741  
 vssseg6wv\_mask\_uint64x6xm1 (C function), 741  
 vssseg6wv\_mask\_uint8x6xm1 (C function), 741  
 vssseg6wv\_uint16x6xm1 (C function), 740  
 vssseg6wv\_uint32x6xm1 (C function), 740  
 vssseg6wv\_uint64x6xm1 (C function), 740  
 vssseg6wv\_uint8x6xm1 (C function), 740  
 vssseg7bv\_int16x7xm1 (C function), 741  
 vssseg7bv\_int32x7xm1 (C function), 741  
 vssseg7bv\_int64x7xm1 (C function), 741  
 vssseg7bv\_int8x7xm1 (C function), 741  
 vssseg7bv\_mask\_int16x7xm1 (C function), 741  
 vssseg7bv\_mask\_int32x7xm1 (C function), 741  
 vssseg7bv\_mask\_int64x7xm1 (C function), 742  
 vssseg7bv\_mask\_int8x7xm1 (C function), 742  
 vssseg7bv\_mask\_uint16x7xm1 (C function), 742  
 vssseg7bv\_mask\_uint32x7xm1 (C function), 742  
 vssseg7bv\_mask\_uint64x7xm1 (C function), 742  
 vssseg7bv\_mask\_uint8x7xm1 (C function), 742  
 vssseg7bv\_uint16x7xm1 (C function), 741  
 vssseg7bv\_uint32x7xm1 (C function), 741  
 vssseg7bv\_uint64x7xm1 (C function), 741  
 vssseg7bv\_uint8x7xm1 (C function), 741



vssseg7ev\_float16x7xm1 (C function), 742  
vssseg7ev\_float32x7xm1 (C function), 742  
vssseg7ev\_float64x7xm1 (C function), 742  
vssseg7ev\_int16x7xm1 (C function), 742  
vssseg7ev\_int32x7xm1 (C function), 742  
vssseg7ev\_int64x7xm1 (C function), 742  
vssseg7ev\_int8x7xm1 (C function), 742  
vssseg7ev\_mask\_float16x7xm1 (C function), 743  
vssseg7ev\_mask\_float32x7xm1 (C function), 743  
vssseg7ev\_mask\_float64x7xm1 (C function), 743  
vssseg7ev\_mask\_int16x7xm1 (C function), 743  
vssseg7ev\_mask\_int32x7xm1 (C function), 743  
vssseg7ev\_mask\_int64x7xm1 (C function), 743  
vssseg7ev\_mask\_int8x7xm1 (C function), 743  
vssseg7ev\_mask\_uint16x7xm1 (C function), 743  
vssseg7ev\_mask\_uint32x7xm1 (C function), 743  
vssseg7ev\_mask\_uint64x7xm1 (C function), 743  
vssseg7ev\_mask\_uint8x7xm1 (C function), 743  
vssseg7ev\_uint16x7xm1 (C function), 742  
vssseg7ev\_uint32x7xm1 (C function), 742  
vssseg7ev\_uint64x7xm1 (C function), 742  
vssseg7ev\_uint8x7xm1 (C function), 742  
vssseg7hv\_int16x7xm1 (C function), 743  
vssseg7hv\_int32x7xm1 (C function), 743  
vssseg7hv\_int64x7xm1 (C function), 743  
vssseg7hv\_int8x7xm1 (C function), 743  
vssseg7hv\_mask\_int16x7xm1 (C function), 744  
vssseg7hv\_mask\_int32x7xm1 (C function), 744  
vssseg7hv\_mask\_int64x7xm1 (C function), 744  
vssseg7hv\_mask\_int8x7xm1 (C function), 744  
vssseg7hv\_mask\_uint16x7xm1 (C function), 744  
vssseg7hv\_mask\_uint32x7xm1 (C function), 744  
vssseg7hv\_mask\_uint64x7xm1 (C function), 744  
vssseg7hv\_mask\_uint8x7xm1 (C function), 744  
vssseg7hv\_uint16x7xm1 (C function), 743  
vssseg7hv\_uint32x7xm1 (C function), 743  
vssseg7hv\_uint64x7xm1 (C function), 743  
vssseg7hv\_uint8x7xm1 (C function), 744  
vssseg7wv\_int16x7xm1 (C function), 744  
vssseg7wv\_int32x7xm1 (C function), 744  
vssseg7wv\_int64x7xm1 (C function), 744  
vssseg7wv\_int8x7xm1 (C function), 744  
vssseg7wv\_mask\_int16x7xm1 (C function), 745  
vssseg7wv\_mask\_int32x7xm1 (C function), 745  
vssseg7wv\_mask\_int64x7xm1 (C function), 745  
vssseg7wv\_mask\_int8x7xm1 (C function), 745  
vssseg7wv\_mask\_uint16x7xm1 (C function), 745  
vssseg7wv\_mask\_uint32x7xm1 (C function), 745  
vssseg7wv\_mask\_uint64x7xm1 (C function), 745  
vssseg7wv\_mask\_uint8x7xm1 (C function), 745  
vssseg7wv\_uint16x7xm1 (C function), 744  
vssseg7wv\_uint32x7xm1 (C function), 744  
vssseg7wv\_uint64x7xm1 (C function), 744  
vssseg7wv\_uint8x7xm1 (C function), 745  
vssseg8bv\_int16x8xm1 (C function), 745  
vssseg8bv\_int32x8xm1 (C function), 745  
vssseg8bv\_int64x8xm1 (C function), 745  
vssseg8bv\_int8x8xm1 (C function), 745  
vssseg8bv\_mask\_int16x8xm1 (C function), 746  
vssseg8bv\_mask\_int32x8xm1 (C function), 746  
vssseg8bv\_mask\_int64x8xm1 (C function), 746  
vssseg8bv\_mask\_int8x8xm1 (C function), 746  
vssseg8bv\_mask\_uint16x8xm1 (C function), 746  
vssseg8bv\_mask\_uint32x8xm1 (C function), 746  
vssseg8bv\_mask\_uint64x8xm1 (C function), 746  
vssseg8bv\_mask\_uint8x8xm1 (C function), 746  
vssseg8bv\_uint16x8xm1 (C function), 745  
vssseg8bv\_uint32x8xm1 (C function), 745  
vssseg8bv\_uint64x8xm1 (C function), 745  
vssseg8bv\_uint8x8xm1 (C function), 746  
vssseg8ev\_float16x8xm1 (C function), 746  
vssseg8ev\_float32x8xm1 (C function), 746  
vssseg8ev\_float64x8xm1 (C function), 746  
vssseg8ev\_int16x8xm1 (C function), 746  
vssseg8ev\_int32x8xm1 (C function), 746  
vssseg8ev\_int64x8xm1 (C function), 746  
vssseg8ev\_int8x8xm1 (C function), 746  
vssseg8ev\_mask\_float16x8xm1 (C function), 747  
vssseg8ev\_mask\_float32x8xm1 (C function), 747  
vssseg8ev\_mask\_float64x8xm1 (C function), 747  
vssseg8ev\_mask\_int16x8xm1 (C function), 747  
vssseg8ev\_mask\_int32x8xm1 (C function), 747  
vssseg8ev\_mask\_int64x8xm1 (C function), 747  
vssseg8ev\_mask\_int8x8xm1 (C function), 747  
vssseg8ev\_mask\_uint16x8xm1 (C function), 747  
vssseg8ev\_mask\_uint32x8xm1 (C function), 747  
vssseg8ev\_mask\_uint64x8xm1 (C function), 747  
vssseg8ev\_mask\_uint8x8xm1 (C function), 747  
vssseg8ev\_uint16x8xm1 (C function), 747  
vssseg8ev\_uint32x8xm1 (C function), 747  
vssseg8ev\_uint64x8xm1 (C function), 747  
vssseg8ev\_uint8x8xm1 (C function), 747  
vssseg8hv\_int16x8xm1 (C function), 748  
vssseg8hv\_int32x8xm1 (C function), 748  
vssseg8hv\_int64x8xm1 (C function), 748  
vssseg8hv\_int8x8xm1 (C function), 748  
vssseg8hv\_mask\_int16x8xm1 (C function), 748  
vssseg8hv\_mask\_int32x8xm1 (C function), 748  
vssseg8hv\_mask\_int64x8xm1 (C function), 748  
vssseg8hv\_mask\_int8x8xm1 (C function), 748  
vssseg8hv\_mask\_uint16x8xm1 (C function), 748  
vssseg8hv\_mask\_uint32x8xm1 (C function), 748  
vssseg8hv\_mask\_uint64x8xm1 (C function), 748  
vssseg8hv\_mask\_uint8x8xm1 (C function), 748  
vssseg8hv\_uint16x8xm1 (C function), 748  
vssseg8hv\_uint32x8xm1 (C function), 748  
vssseg8hv\_uint64x8xm1 (C function), 748  
vssseg8hv\_uint8x8xm1 (C function), 748

vssseg8wv\_int16x8xm1 (C function), 749  
 vssseg8wv\_int32x8xm1 (C function), 749  
 vssseg8wv\_int64x8xm1 (C function), 749  
 vssseg8wv\_int8x8xm1 (C function), 749  
 vssseg8wv\_mask\_int16x8xm1 (C function), 749  
 vssseg8wv\_mask\_int32x8xm1 (C function), 749  
 vssseg8wv\_mask\_int64x8xm1 (C function), 749  
 vssseg8wv\_mask\_int8x8xm1 (C function), 749  
 vssseg8wv\_mask\_uint16x8xm1 (C function), 749  
 vssseg8wv\_mask\_uint32x8xm1 (C function), 749  
 vssseg8wv\_mask\_uint64x8xm1 (C function), 749  
 vssseg8wv\_mask\_uint8x8xm1 (C function), 749  
 vssseg8wv\_uint16x8xm1 (C function), 749  
 vssseg8wv\_uint32x8xm1 (C function), 749  
 vssseg8wv\_uint64x8xm1 (C function), 749  
 vssseg8wv\_uint8x8xm1 (C function), 749  
 vssubuvx\_mask\_uint16xm1 (C function), 314  
 vssubuvx\_mask\_uint16xm2 (C function), 314  
 vssubuvx\_mask\_uint16xm4 (C function), 314  
 vssubuvx\_mask\_uint16xm8 (C function), 314  
 vssubuvx\_mask\_uint32xm1 (C function), 314  
 vssubuvx\_mask\_uint32xm2 (C function), 314  
 vssubuvx\_mask\_uint32xm4 (C function), 314  
 vssubuvx\_mask\_uint32xm8 (C function), 314  
 vssubuvx\_mask\_uint64xm1 (C function), 314  
 vssubuvx\_mask\_uint64xm2 (C function), 314  
 vssubuvx\_mask\_uint64xm4 (C function), 314  
 vssubuvx\_mask\_uint64xm8 (C function), 314  
 vssubuvx\_mask\_uint8xm1 (C function), 314  
 vssubuvx\_mask\_uint8xm2 (C function), 314  
 vssubuvx\_mask\_uint8xm4 (C function), 314  
 vssubuvx\_mask\_uint8xm8 (C function), 314  
 vssubuvx\_uint16xm1 (C function), 313  
 vssubuvx\_uint16xm2 (C function), 313  
 vssubuvx\_uint16xm4 (C function), 313  
 vssubuvx\_uint16xm8 (C function), 313  
 vssubuvx\_uint32xm1 (C function), 313  
 vssubuvx\_uint32xm2 (C function), 313  
 vssubuvx\_uint32xm4 (C function), 313  
 vssubuvx\_uint32xm8 (C function), 313  
 vssubuvx\_uint64xm1 (C function), 313  
 vssubuvx\_uint64xm2 (C function), 313  
 vssubuvx\_uint64xm4 (C function), 313  
 vssubuvx\_uint64xm8 (C function), 313  
 vssubuvx\_uint8xm1 (C function), 313  
 vssubuvx\_uint8xm2 (C function), 313  
 vssubuvx\_uint8xm4 (C function), 313  
 vssubuvx\_uint8xm8 (C function), 313  
 vssubuvx\_mask\_uint16xm1 (C function), 315  
 vssubuvx\_mask\_uint16xm2 (C function), 315  
 vssubuvx\_mask\_uint16xm4 (C function), 315  
 vssubuvx\_mask\_uint16xm8 (C function), 315  
 vssubuvx\_mask\_uint32xm1 (C function), 315  
 vssubuvx\_mask\_uint32xm2 (C function), 315  
 vssubuvx\_mask\_uint32xm4 (C function), 315  
 vssubuvx\_mask\_uint32xm8 (C function), 315  
 vssubuvx\_mask\_uint64xm1 (C function), 315  
 vssubuvx\_mask\_uint64xm2 (C function), 315  
 vssubuvx\_mask\_uint64xm4 (C function), 315  
 vssubuvx\_mask\_uint64xm8 (C function), 315  
 vssubuvx\_mask\_uint8xm1 (C function), 315  
 vssubuvx\_mask\_uint8xm2 (C function), 315  
 vssubuvx\_mask\_uint8xm4 (C function), 315  
 vssubuvx\_mask\_uint8xm8 (C function), 315  
 vssubvv\_int16xm1 (C function), 310  
 vssubvv\_int16xm2 (C function), 310  
 vssubvv\_int16xm4 (C function), 310  
 vssubvv\_int16xm8 (C function), 310  
 vssubvv\_int32xm1 (C function), 310  
 vssubvv\_int32xm2 (C function), 310  
 vssubvv\_int32xm4 (C function), 310  
 vssubvv\_int32xm8 (C function), 310  
 vssubvv\_int64xm1 (C function), 310  
 vssubvv\_int64xm2 (C function), 310  
 vssubvv\_int64xm4 (C function), 310  
 vssubvv\_int64xm8 (C function), 310  
 vssubvv\_int8xm1 (C function), 310  
 vssubvv\_int8xm2 (C function), 310  
 vssubvv\_int8xm4 (C function), 310  
 vssubvv\_int8xm8 (C function), 310  
 vssubvv\_mask\_int16xm1 (C function), 311  
 vssubvv\_mask\_int16xm2 (C function), 311  
 vssubvv\_mask\_int16xm4 (C function), 311  
 vssubvv\_mask\_int16xm8 (C function), 311  
 vssubvv\_mask\_int32xm1 (C function), 311  
 vssubvv\_mask\_int32xm2 (C function), 311  
 vssubvv\_mask\_int32xm4 (C function), 311  
 vssubvv\_mask\_int32xm8 (C function), 311  
 vssubvv\_mask\_int64xm1 (C function), 311  
 vssubvv\_mask\_int64xm2 (C function), 311  
 vssubvv\_mask\_int64xm4 (C function), 311  
 vssubvv\_mask\_int64xm8 (C function), 311

vssubvv\_mask\_int8xm1 (C function), 311  
vssubvv\_mask\_int8xm2 (C function), 311  
vssubvv\_mask\_int8xm4 (C function), 311  
vssubvv\_mask\_int8xm8 (C function), 311  
vssubvx\_int16xm1 (C function), 311  
vssubvx\_int16xm2 (C function), 312  
vssubvx\_int16xm4 (C function), 312  
vssubvx\_int16xm8 (C function), 312  
vssubvx\_int32xm1 (C function), 312  
vssubvx\_int32xm2 (C function), 312  
vssubvx\_int32xm4 (C function), 312  
vssubvx\_int32xm8 (C function), 312  
vssubvx\_int64xm1 (C function), 312  
vssubvx\_int64xm2 (C function), 312  
vssubvx\_int64xm4 (C function), 312  
vssubvx\_int64xm8 (C function), 312  
vssubvx\_int8xm1 (C function), 312  
vssubvx\_int8xm2 (C function), 312  
vssubvx\_int8xm4 (C function), 312  
vssubvx\_int8xm8 (C function), 312  
vssubvx\_mask\_int16xm1 (C function), 312  
vssubvx\_mask\_int16xm2 (C function), 312  
vssubvx\_mask\_int16xm4 (C function), 312  
vssubvx\_mask\_int16xm8 (C function), 312  
vssubvx\_mask\_int32xm1 (C function), 312  
vssubvx\_mask\_int32xm2 (C function), 312  
vssubvx\_mask\_int32xm4 (C function), 312  
vssubvx\_mask\_int32xm8 (C function), 312  
vssubvx\_mask\_int64xm1 (C function), 312  
vssubvx\_mask\_int64xm2 (C function), 312  
vssubvx\_mask\_int64xm4 (C function), 313  
vssubvx\_mask\_int64xm8 (C function), 313  
vssubvx\_mask\_int8xm1 (C function), 313  
vssubvx\_mask\_int8xm2 (C function), 313  
vssubvx\_mask\_int8xm4 (C function), 313  
vssubvx\_mask\_int8xm8 (C function), 313  
vsswv\_int16xm1 (C function), 467  
vsswv\_int16xm2 (C function), 467  
vsswv\_int16xm4 (C function), 467  
vsswv\_int16xm8 (C function), 467  
vsswv\_int32xm1 (C function), 467  
vsswv\_int32xm2 (C function), 467  
vsswv\_int32xm4 (C function), 467  
vsswv\_int32xm8 (C function), 467  
vsswv\_int64xm1 (C function), 467  
vsswv\_int64xm2 (C function), 467  
vsswv\_int64xm4 (C function), 467  
vsswv\_int64xm8 (C function), 467  
vsswv\_int8xm1 (C function), 467  
vsswv\_int8xm2 (C function), 467  
vsswv\_int8xm4 (C function), 467  
vsswv\_int8xm8 (C function), 467  
vsswv\_mask\_int16xm1 (C function), 468  
vsswv\_mask\_int16xm2 (C function), 468

vsswv\_mask\_int16xm4 (C function), 468  
vsswv\_mask\_int16xm8 (C function), 468  
vsswv\_mask\_int32xm1 (C function), 468  
vsswv\_mask\_int32xm2 (C function), 468  
vsswv\_mask\_int32xm4 (C function), 468  
vsswv\_mask\_int32xm8 (C function), 468  
vsswv\_mask\_int64xm1 (C function), 468  
vsswv\_mask\_int64xm2 (C function), 468  
vsswv\_mask\_int64xm4 (C function), 468  
vsswv\_mask\_int64xm8 (C function), 468  
vsswv\_mask\_int8xm1 (C function), 468  
vsswv\_mask\_int8xm2 (C function), 468  
vsswv\_mask\_int8xm4 (C function), 468  
vsswv\_mask\_int8xm8 (C function), 468  
vsswv\_mask\_uint16xm1 (C function), 468  
vsswv\_mask\_uint16xm2 (C function), 468  
vsswv\_mask\_uint16xm4 (C function), 469  
vsswv\_mask\_uint16xm8 (C function), 469  
vsswv\_mask\_uint32xm1 (C function), 469  
vsswv\_mask\_uint32xm2 (C function), 469  
vsswv\_mask\_uint32xm4 (C function), 469  
vsswv\_mask\_uint32xm8 (C function), 469  
vsswv\_mask\_uint64xm1 (C function), 469  
vsswv\_mask\_uint64xm2 (C function), 469  
vsswv\_mask\_uint64xm4 (C function), 469  
vsswv\_mask\_uint64xm8 (C function), 469  
vsswv\_mask\_uint8xm1 (C function), 469  
vsswv\_mask\_uint8xm2 (C function), 469  
vsswv\_mask\_uint8xm4 (C function), 469  
vsswv\_mask\_uint8xm8 (C function), 469  
vsswv\_uint16xm1 (C function), 467  
vsswv\_uint16xm2 (C function), 467  
vsswv\_uint16xm4 (C function), 467  
vsswv\_uint16xm8 (C function), 467  
vsswv\_uint32xm1 (C function), 467  
vsswv\_uint32xm2 (C function), 467  
vsswv\_uint32xm4 (C function), 467  
vsswv\_uint32xm8 (C function), 467  
vsswv\_uint64xm1 (C function), 467  
vsswv\_uint64xm2 (C function), 467  
vsswv\_uint64xm4 (C function), 467  
vsswv\_uint64xm8 (C function), 467  
vsswv\_uint8xm1 (C function), 468  
vsswv\_uint8xm2 (C function), 468  
vsswv\_uint8xm4 (C function), 468  
vsswv\_uint8xm8 (C function), 468  
vsubvv\_int16xm1 (C function), 316  
vsubvv\_int16xm2 (C function), 316  
vsubvv\_int16xm4 (C function), 316  
vsubvv\_int16xm8 (C function), 316  
vsubvv\_int32xm1 (C function), 316  
vsubvv\_int32xm2 (C function), 316  
vsubvv\_int32xm4 (C function), 316  
vsubvv\_int32xm8 (C function), 316





vsubvx\_uint32xm1 (C function), 319  
vsubvx\_uint32xm2 (C function), 319  
vsubvx\_uint32xm4 (C function), 319  
vsubvx\_uint32xm8 (C function), 319  
vsubvx\_uint64xm1 (C function), 319  
vsubvx\_uint64xm2 (C function), 320  
vsubvx\_uint64xm4 (C function), 320  
vsubvx\_uint64xm8 (C function), 320  
vsubvx\_uint8xm1 (C function), 320  
vsubvx\_uint8xm2 (C function), 320  
vsubvx\_uint8xm4 (C function), 320  
vsubvx\_uint8xm8 (C function), 320  
vsuxbv\_int16xm1 (C function), 469  
vsuxbv\_int16xm2 (C function), 469  
vsuxbv\_int16xm4 (C function), 469  
vsuxbv\_int16xm8 (C function), 469  
vsuxbv\_int32xm1 (C function), 469  
vsuxbv\_int32xm2 (C function), 469  
vsuxbv\_int32xm4 (C function), 470  
vsuxbv\_int32xm8 (C function), 470  
vsuxbv\_int64xm1 (C function), 470  
vsuxbv\_int64xm2 (C function), 470  
vsuxbv\_int64xm4 (C function), 470  
vsuxbv\_int64xm8 (C function), 470  
vsuxbv\_int8xm1 (C function), 470  
vsuxbv\_int8xm2 (C function), 470  
vsuxbv\_int8xm4 (C function), 470  
vsuxbv\_int8xm8 (C function), 470  
vsuxbv\_mask\_int16xm1 (C function), 470  
vsuxbv\_mask\_int16xm2 (C function), 471  
vsuxbv\_mask\_int16xm4 (C function), 471  
vsuxbv\_mask\_int16xm8 (C function), 471  
vsuxbv\_mask\_int32xm1 (C function), 471  
vsuxbv\_mask\_int32xm2 (C function), 471  
vsuxbv\_mask\_int32xm4 (C function), 471  
vsuxbv\_mask\_int32xm8 (C function), 471  
vsuxbv\_mask\_int64xm1 (C function), 471  
vsuxbv\_mask\_int64xm2 (C function), 471  
vsuxbv\_mask\_int64xm4 (C function), 471  
vsuxbv\_mask\_int64xm8 (C function), 471  
vsuxbv\_mask\_int8xm1 (C function), 471  
vsuxbv\_mask\_int8xm2 (C function), 471  
vsuxbv\_mask\_int8xm4 (C function), 471  
vsuxbv\_mask\_int8xm8 (C function), 471  
vsuxbv\_mask\_uint16xm1 (C function), 471  
vsuxbv\_mask\_uint16xm2 (C function), 471  
vsuxbv\_mask\_uint16xm4 (C function), 471  
vsuxbv\_mask\_uint16xm8 (C function), 471  
vsuxbv\_mask\_uint32xm1 (C function), 471  
vsuxbv\_mask\_uint32xm2 (C function), 471  
vsuxbv\_mask\_uint32xm4 (C function), 471  
vsuxbv\_mask\_uint32xm8 (C function), 471  
vsuxbv\_mask\_uint64xm1 (C function), 472  
vsuxbv\_mask\_uint64xm2 (C function), 472  
vsuxbv\_mask\_uint64xm4 (C function), 472  
vsuxbv\_mask\_uint64xm8 (C function), 472  
vsuxbv\_uint16xm1 (C function), 470  
vsuxbv\_uint16xm2 (C function), 470  
vsuxbv\_uint16xm4 (C function), 470  
vsuxbv\_uint16xm8 (C function), 470  
vsuxbv\_uint32xm1 (C function), 470  
vsuxbv\_uint32xm2 (C function), 470  
vsuxbv\_uint32xm4 (C function), 470  
vsuxbv\_uint32xm8 (C function), 470  
vsuxbv\_uint64xm1 (C function), 470  
vsuxbv\_uint64xm2 (C function), 470  
vsuxbv\_uint64xm4 (C function), 470  
vsuxbv\_uint64xm8 (C function), 470  
vsuxbv\_uint8xm1 (C function), 470  
vsuxbv\_uint8xm2 (C function), 470  
vsuxbv\_uint8xm4 (C function), 470  
vsuxbv\_uint8xm8 (C function), 470  
vsuxev\_float16xm1 (C function), 472  
vsuxev\_float16xm2 (C function), 472  
vsuxev\_float16xm4 (C function), 472  
vsuxev\_float16xm8 (C function), 472  
vsuxev\_float32xm1 (C function), 472  
vsuxev\_float32xm2 (C function), 472  
vsuxev\_float32xm4 (C function), 472  
vsuxev\_float32xm8 (C function), 472  
vsuxev\_float64xm1 (C function), 472  
vsuxev\_float64xm2 (C function), 472  
vsuxev\_float64xm4 (C function), 472  
vsuxev\_float64xm8 (C function), 472  
vsuxev\_int16xm1 (C function), 472  
vsuxev\_int16xm2 (C function), 472  
vsuxev\_int16xm4 (C function), 472  
vsuxev\_int16xm8 (C function), 472  
vsuxev\_int32xm1 (C function), 473  
vsuxev\_int32xm2 (C function), 473  
vsuxev\_int32xm4 (C function), 473  
vsuxev\_int32xm8 (C function), 473  
vsuxev\_int64xm1 (C function), 473  
vsuxev\_int64xm2 (C function), 473  
vsuxev\_int64xm4 (C function), 473  
vsuxev\_int64xm8 (C function), 473  
vsuxev\_int8xm1 (C function), 473  
vsuxev\_int8xm2 (C function), 473  
vsuxev\_int8xm4 (C function), 473  
vsuxev\_int8xm8 (C function), 473  
vsuxev\_mask\_float16xm1 (C function), 474  
vsuxev\_mask\_float16xm2 (C function), 474  
vsuxev\_mask\_float16xm4 (C function), 474  
vsuxev\_mask\_float16xm8 (C function), 474





vsuxhv\_uint32xm1 (C function), 476  
vsuxhv\_uint32xm2 (C function), 476  
vsuxhv\_uint32xm4 (C function), 476  
vsuxhv\_uint32xm8 (C function), 476  
vsuxhv\_uint64xm1 (C function), 476  
vsuxhv\_uint64xm2 (C function), 476  
vsuxhv\_uint64xm4 (C function), 476  
vsuxhv\_uint64xm8 (C function), 476  
vsuxhv\_uint8xm1 (C function), 477  
vsuxhv\_uint8xm2 (C function), 477  
vsuxhv\_uint8xm4 (C function), 477  
vsuxhv\_uint8xm8 (C function), 477  
vsuxwv\_int16xm1 (C function), 478  
vsuxwv\_int16xm2 (C function), 478  
vsuxwv\_int16xm4 (C function), 478  
vsuxwv\_int16xm8 (C function), 478  
vsuxwv\_int32xm1 (C function), 478  
vsuxwv\_int32xm2 (C function), 478  
vsuxwv\_int32xm4 (C function), 479  
vsuxwv\_int32xm8 (C function), 479  
vsuxwv\_int64xm1 (C function), 479  
vsuxwv\_int64xm2 (C function), 479  
vsuxwv\_int64xm4 (C function), 479  
vsuxwv\_int64xm8 (C function), 479  
vsuxwv\_int8xm1 (C function), 479  
vsuxwv\_int8xm2 (C function), 479  
vsuxwv\_int8xm4 (C function), 479  
vsuxwv\_int8xm8 (C function), 479  
vsuxwv\_mask\_int16xm1 (C function), 479  
vsuxwv\_mask\_int16xm2 (C function), 480  
vsuxwv\_mask\_int16xm4 (C function), 480  
vsuxwv\_mask\_int16xm8 (C function), 480  
vsuxwv\_mask\_int32xm1 (C function), 480  
vsuxwv\_mask\_int32xm2 (C function), 480  
vsuxwv\_mask\_int32xm4 (C function), 480  
vsuxwv\_mask\_int32xm8 (C function), 480  
vsuxwv\_mask\_int64xm1 (C function), 480  
vsuxwv\_mask\_int64xm2 (C function), 480  
vsuxwv\_mask\_int64xm4 (C function), 480  
vsuxwv\_mask\_int64xm8 (C function), 480  
vsuxwv\_mask\_int8xm1 (C function), 480  
vsuxwv\_mask\_int8xm2 (C function), 480  
vsuxwv\_mask\_int8xm4 (C function), 480  
vsuxwv\_mask\_int8xm8 (C function), 480  
vsuxwv\_mask\_uint16xm1 (C function), 480  
vsuxwv\_mask\_uint16xm2 (C function), 480  
vsuxwv\_mask\_uint16xm4 (C function), 480  
vsuxwv\_mask\_uint16xm8 (C function), 480  
vsuxwv\_mask\_uint32xm1 (C function), 480  
vsuxwv\_mask\_uint32xm2 (C function), 480  
vsuxwv\_mask\_uint32xm4 (C function), 480  
vsuxwv\_mask\_uint32xm8 (C function), 480  
vsuxwv\_mask\_uint64xm1 (C function), 481  
vsuxwv\_mask\_uint64xm2 (C function), 481

vsuxwv\_mask\_uint64xm4 (C function), 481  
vsuxwv\_mask\_uint64xm8 (C function), 481  
vsuxwv\_mask\_uint8xm1 (C function), 481  
vsuxwv\_mask\_uint8xm2 (C function), 481  
vsuxwv\_mask\_uint8xm4 (C function), 481  
vsuxwv\_mask\_uint8xm8 (C function), 481  
vsuxwv\_uint16xm1 (C function), 479  
vsuxwv\_uint16xm2 (C function), 479  
vsuxwv\_uint16xm4 (C function), 479  
vsuxwv\_uint16xm8 (C function), 479  
vsuxwv\_uint32xm1 (C function), 479  
vsuxwv\_uint32xm2 (C function), 479  
vsuxwv\_uint32xm4 (C function), 479  
vsuxwv\_uint32xm8 (C function), 479  
vsuxwv\_uint64xm1 (C function), 479  
vsuxwv\_uint64xm2 (C function), 479  
vsuxwv\_uint64xm4 (C function), 479  
vsuxwv\_uint64xm8 (C function), 479  
vsuxwv\_uint8xm1 (C function), 479  
vsuxwv\_uint8xm2 (C function), 479  
vsuxwv\_uint8xm4 (C function), 479  
vsuxwv\_uint8xm8 (C function), 479  
vswv\_int16xm1 (C function), 481  
vswv\_int16xm2 (C function), 481  
vswv\_int16xm4 (C function), 481  
vswv\_int16xm8 (C function), 481  
vswv\_int32xm1 (C function), 481  
vswv\_int32xm2 (C function), 481  
vswv\_int32xm4 (C function), 481  
vswv\_int32xm8 (C function), 481  
vswv\_int64xm1 (C function), 481  
vswv\_int64xm2 (C function), 481  
vswv\_int64xm4 (C function), 481  
vswv\_int64xm8 (C function), 481  
vswv\_int8xm1 (C function), 481  
vswv\_int8xm2 (C function), 481  
vswv\_int8xm4 (C function), 481  
vswv\_int8xm8 (C function), 481  
vswv\_mask\_int16xm1 (C function), 482  
vswv\_mask\_int16xm2 (C function), 482  
vswv\_mask\_int16xm4 (C function), 482  
vswv\_mask\_int16xm8 (C function), 482  
vswv\_mask\_int32xm1 (C function), 482  
vswv\_mask\_int32xm2 (C function), 482  
vswv\_mask\_int32xm4 (C function), 482  
vswv\_mask\_int32xm8 (C function), 482  
vswv\_mask\_int64xm1 (C function), 482  
vswv\_mask\_int64xm2 (C function), 482  
vswv\_mask\_int64xm4 (C function), 482  
vswv\_mask\_int64xm8 (C function), 482  
vswv\_mask\_int8xm1 (C function), 482  
vswv\_mask\_int8xm2 (C function), 482  
vswv\_mask\_int8xm4 (C function), 482  
vswv\_mask\_int8xm8 (C function), 483

vswv\_mask\_uint16xm1 (C function), 483  
 vswv\_mask\_uint16xm2 (C function), 483  
 vswv\_mask\_uint16xm4 (C function), 483  
 vswv\_mask\_uint16xm8 (C function), 483  
 vswv\_mask\_uint32xm1 (C function), 483  
 vswv\_mask\_uint32xm2 (C function), 483  
 vswv\_mask\_uint32xm4 (C function), 483  
 vswv\_mask\_uint32xm8 (C function), 483  
 vswv\_mask\_uint64xm1 (C function), 483  
 vswv\_mask\_uint64xm2 (C function), 483  
 vswv\_mask\_uint64xm4 (C function), 483  
 vswv\_mask\_uint64xm8 (C function), 483  
 vswv\_mask\_uint8xm1 (C function), 483  
 vswv\_mask\_uint8xm2 (C function), 483  
 vswv\_mask\_uint8xm4 (C function), 483  
 vswv\_mask\_uint8xm8 (C function), 483  
 vswv\_uint16xm1 (C function), 482  
 vswv\_uint16xm2 (C function), 482  
 vswv\_uint16xm4 (C function), 482  
 vswv\_uint16xm8 (C function), 482  
 vswv\_uint32xm1 (C function), 482  
 vswv\_uint32xm2 (C function), 482  
 vswv\_uint32xm4 (C function), 482  
 vswv\_uint32xm8 (C function), 482  
 vswv\_uint64xm1 (C function), 482  
 vswv\_uint64xm2 (C function), 482  
 vswv\_uint64xm4 (C function), 482  
 vswv\_uint64xm8 (C function), 482  
 vswv\_uint8xm1 (C function), 482  
 vswv\_uint8xm2 (C function), 482  
 vswv\_uint8xm4 (C function), 482  
 vswv\_uint8xm8 (C function), 482  
 vsxbv\_int16xm1 (C function), 483  
 vsxbv\_int16xm2 (C function), 483  
 vsxbv\_int16xm4 (C function), 483  
 vsxbv\_int16xm8 (C function), 483  
 vsxbv\_int32xm1 (C function), 484  
 vsxbv\_int32xm2 (C function), 484  
 vsxbv\_int32xm4 (C function), 484  
 vsxbv\_int32xm8 (C function), 484  
 vsxbv\_int64xm1 (C function), 484  
 vsxbv\_int64xm2 (C function), 484  
 vsxbv\_int64xm4 (C function), 484  
 vsxbv\_int64xm8 (C function), 484  
 vsxbv\_int8xm1 (C function), 484  
 vsxbv\_int8xm2 (C function), 484  
 vsxbv\_int8xm4 (C function), 484  
 vsxbv\_int8xm8 (C function), 484  
 vsxbv\_mask\_int16xm1 (C function), 484  
 vsxbv\_mask\_int16xm2 (C function), 485  
 vsxbv\_mask\_int16xm4 (C function), 485  
 vsxbv\_mask\_int16xm8 (C function), 485  
 vsxbv\_mask\_int32xm1 (C function), 485  
 vsxbv\_mask\_int32xm2 (C function), 485  
 vsxbv\_mask\_int32xm4 (C function), 485  
 vsxbv\_mask\_int32xm8 (C function), 485  
 vsxbv\_mask\_int64xm1 (C function), 485  
 vsxbv\_mask\_int64xm2 (C function), 485  
 vsxbv\_mask\_int64xm4 (C function), 485  
 vsxbv\_mask\_int64xm8 (C function), 485  
 vsxbv\_mask\_int8xm1 (C function), 485  
 vsxbv\_mask\_int8xm2 (C function), 485  
 vsxbv\_mask\_int8xm4 (C function), 485  
 vsxbv\_mask\_int8xm8 (C function), 485  
 vsxbv\_mask\_uint16xm1 (C function), 485  
 vsxbv\_mask\_uint16xm2 (C function), 485  
 vsxbv\_mask\_uint16xm4 (C function), 485  
 vsxbv\_mask\_uint16xm8 (C function), 485  
 vsxbv\_mask\_uint32xm1 (C function), 485  
 vsxbv\_mask\_uint32xm2 (C function), 485  
 vsxbv\_mask\_uint32xm4 (C function), 485  
 vsxbv\_mask\_uint32xm8 (C function), 485  
 vsxbv\_mask\_uint64xm1 (C function), 486  
 vsxbv\_mask\_uint64xm2 (C function), 486  
 vsxbv\_mask\_uint64xm4 (C function), 486  
 vsxbv\_mask\_uint64xm8 (C function), 486  
 vsxbv\_mask\_uint8xm1 (C function), 486  
 vsxbv\_mask\_uint8xm2 (C function), 486  
 vsxbv\_mask\_uint8xm4 (C function), 486  
 vsxbv\_mask\_uint8xm8 (C function), 486  
 vsxbv\_uint16xm1 (C function), 484  
 vsxbv\_uint16xm2 (C function), 484  
 vsxbv\_uint16xm4 (C function), 484  
 vsxbv\_uint16xm8 (C function), 484  
 vsxbv\_uint32xm1 (C function), 484  
 vsxbv\_uint32xm2 (C function), 484  
 vsxbv\_uint32xm4 (C function), 484  
 vsxbv\_uint32xm8 (C function), 484  
 vsxbv\_uint64xm1 (C function), 484  
 vsxbv\_uint64xm2 (C function), 484  
 vsxbv\_uint64xm4 (C function), 484  
 vsxbv\_uint64xm8 (C function), 484  
 vsxbv\_uint8xm1 (C function), 484  
 vsxbv\_uint8xm2 (C function), 484  
 vsxbv\_uint8xm4 (C function), 484  
 vsxbv\_uint8xm8 (C function), 484  
 vsxev\_float16xm1 (C function), 486  
 vsxev\_float16xm2 (C function), 486  
 vsxev\_float16xm4 (C function), 486  
 vsxev\_float16xm8 (C function), 486  
 vsxev\_float32xm1 (C function), 486  
 vsxev\_float32xm2 (C function), 486  
 vsxev\_float32xm4 (C function), 486  
 vsxev\_float32xm8 (C function), 486  
 vsxev\_float64xm1 (C function), 486  
 vsxev\_float64xm2 (C function), 486  
 vsxev\_float64xm4 (C function), 486  
 vsxev\_float64xm8 (C function), 486

vsxev\_int16xm1 (C function), 486  
vsxev\_int16xm2 (C function), 486  
vsxev\_int16xm4 (C function), 486  
vsxev\_int16xm8 (C function), 486  
vsxev\_int32xm1 (C function), 487  
vsxev\_int32xm2 (C function), 487  
vsxev\_int32xm4 (C function), 487  
vsxev\_int32xm8 (C function), 487  
vsxev\_int64xm1 (C function), 487  
vsxev\_int64xm2 (C function), 487  
vsxev\_int64xm4 (C function), 487  
vsxev\_int64xm8 (C function), 487  
vsxev\_int8xm1 (C function), 487  
vsxev\_int8xm2 (C function), 487  
vsxev\_int8xm4 (C function), 487  
vsxev\_int8xm8 (C function), 487  
vsxev\_mask\_float16xm1 (C function), 487  
vsxev\_mask\_float16xm2 (C function), 488  
vsxev\_mask\_float16xm4 (C function), 488  
vsxev\_mask\_float16xm8 (C function), 488  
vsxev\_mask\_float32xm1 (C function), 488  
vsxev\_mask\_float32xm2 (C function), 488  
vsxev\_mask\_float32xm4 (C function), 488  
vsxev\_mask\_float32xm8 (C function), 488  
vsxev\_mask\_float64xm1 (C function), 488  
vsxev\_mask\_float64xm2 (C function), 488  
vsxev\_mask\_float64xm4 (C function), 488  
vsxev\_mask\_float64xm8 (C function), 488  
vsxev\_mask\_int16xm1 (C function), 488  
vsxev\_mask\_int16xm2 (C function), 488  
vsxev\_mask\_int16xm4 (C function), 488  
vsxev\_mask\_int16xm8 (C function), 488  
vsxev\_mask\_int32xm1 (C function), 488  
vsxev\_mask\_int32xm2 (C function), 488  
vsxev\_mask\_int32xm4 (C function), 488  
vsxev\_mask\_int32xm8 (C function), 488  
vsxev\_mask\_int64xm1 (C function), 488  
vsxev\_mask\_int64xm2 (C function), 488  
vsxev\_mask\_int64xm4 (C function), 488  
vsxev\_mask\_int64xm8 (C function), 488  
vsxev\_mask\_int8xm1 (C function), 489  
vsxev\_mask\_int8xm2 (C function), 489  
vsxev\_mask\_int8xm4 (C function), 489  
vsxev\_mask\_int8xm8 (C function), 489  
vsxev\_mask\_uint16xm1 (C function), 489  
vsxev\_mask\_uint16xm2 (C function), 489  
vsxev\_mask\_uint16xm4 (C function), 489  
vsxev\_mask\_uint16xm8 (C function), 489  
vsxev\_mask\_uint32xm1 (C function), 489  
vsxev\_mask\_uint32xm2 (C function), 489  
vsxev\_mask\_uint32xm4 (C function), 489  
vsxev\_mask\_uint32xm8 (C function), 489  
vsxev\_mask\_uint64xm1 (C function), 489  
vsxev\_mask\_uint64xm2 (C function), 489  
vsxev\_mask\_uint64xm4 (C function), 489  
vsxev\_mask\_uint64xm8 (C function), 489  
vsxev\_mask\_uint8xm1 (C function), 489  
vsxev\_mask\_uint8xm2 (C function), 489  
vsxev\_mask\_uint8xm4 (C function), 489  
vsxev\_mask\_uint8xm8 (C function), 489  
vsxev\_uint16xm1 (C function), 487  
vsxev\_uint16xm2 (C function), 487  
vsxev\_uint16xm4 (C function), 487  
vsxev\_uint16xm8 (C function), 487  
vsxev\_uint32xm1 (C function), 487  
vsxev\_uint32xm2 (C function), 487  
vsxev\_uint32xm4 (C function), 487  
vsxev\_uint32xm8 (C function), 487  
vsxev\_uint64xm1 (C function), 487  
vsxev\_uint64xm2 (C function), 487  
vsxev\_uint64xm4 (C function), 487  
vsxev\_uint64xm8 (C function), 487  
vsxev\_uint8xm1 (C function), 487  
vsxev\_uint8xm2 (C function), 487  
vsxev\_uint8xm4 (C function), 487  
vsxev\_uint8xm8 (C function), 487  
vsxhv\_int16xm1 (C function), 490  
vsxhv\_int16xm2 (C function), 490  
vsxhv\_int16xm4 (C function), 490  
vsxhv\_int16xm8 (C function), 490  
vsxhv\_int32xm1 (C function), 490  
vsxhv\_int32xm2 (C function), 490  
vsxhv\_int32xm4 (C function), 490  
vsxhv\_int32xm8 (C function), 490  
vsxhv\_int64xm1 (C function), 490  
vsxhv\_int64xm2 (C function), 490  
vsxhv\_int64xm4 (C function), 490  
vsxhv\_int64xm8 (C function), 490  
vsxhv\_int8xm1 (C function), 490  
vsxhv\_int8xm2 (C function), 490  
vsxhv\_int8xm4 (C function), 490  
vsxhv\_int8xm8 (C function), 490  
vsxhv\_mask\_int16xm1 (C function), 491  
vsxhv\_mask\_int16xm2 (C function), 491  
vsxhv\_mask\_int16xm4 (C function), 491  
vsxhv\_mask\_int16xm8 (C function), 491  
vsxhv\_mask\_int32xm1 (C function), 491  
vsxhv\_mask\_int32xm2 (C function), 491  
vsxhv\_mask\_int32xm4 (C function), 491  
vsxhv\_mask\_int32xm8 (C function), 491  
vsxhv\_mask\_int64xm1 (C function), 491  
vsxhv\_mask\_int64xm2 (C function), 491  
vsxhv\_mask\_int64xm4 (C function), 491  
vsxhv\_mask\_int64xm8 (C function), 491  
vsxhv\_mask\_int8xm1 (C function), 491  
vsxhv\_mask\_int8xm2 (C function), 491  
vsxhv\_mask\_int8xm4 (C function), 491  
vsxhv\_mask\_int8xm8 (C function), 491

vsxhv\_mask\_uint16xm1 (C function), 491  
 vsxhv\_mask\_uint16xm2 (C function), 491  
 vsxhv\_mask\_uint16xm4 (C function), 491  
 vsxhv\_mask\_uint16xm8 (C function), 491  
 vsxhv\_mask\_uint32xm1 (C function), 492  
 vsxhv\_mask\_uint32xm2 (C function), 492  
 vsxhv\_mask\_uint32xm4 (C function), 492  
 vsxhv\_mask\_uint32xm8 (C function), 492  
 vsxhv\_mask\_uint64xm1 (C function), 492  
 vsxhv\_mask\_uint64xm2 (C function), 492  
 vsxhv\_mask\_uint64xm4 (C function), 492  
 vsxhv\_mask\_uint64xm8 (C function), 492  
 vsxhv\_mask\_uint8xm1 (C function), 492  
 vsxhv\_mask\_uint8xm2 (C function), 492  
 vsxhv\_mask\_uint8xm4 (C function), 492  
 vsxhv\_mask\_uint8xm8 (C function), 492  
 vsxhv\_uint16xm1 (C function), 490  
 vsxhv\_uint16xm2 (C function), 490  
 vsxhv\_uint16xm4 (C function), 490  
 vsxhv\_uint16xm8 (C function), 490  
 vsxhv\_uint32xm1 (C function), 490  
 vsxhv\_uint32xm2 (C function), 490  
 vsxhv\_uint32xm4 (C function), 490  
 vsxhv\_uint32xm8 (C function), 490  
 vsxhv\_uint64xm1 (C function), 490  
 vsxhv\_uint64xm2 (C function), 490  
 vsxhv\_uint64xm4 (C function), 490  
 vsxhv\_uint64xm8 (C function), 490  
 vsxhv\_uint8xm1 (C function), 490  
 vsxhv\_uint8xm2 (C function), 490  
 vsxhv\_uint8xm4 (C function), 490  
 vsxhv\_uint8xm8 (C function), 491  
 vsxseg2bv\_int16xm1\_int16x2xm1 (C function), 750  
 vsxseg2bv\_int16xm2\_int16x2xm2 (C function), 750  
 vsxseg2bv\_int16xm4\_int16x2xm4 (C function), 750  
 vsxseg2bv\_int32xm1\_int32x2xm1 (C function), 750  
 vsxseg2bv\_int32xm2\_int32x2xm2 (C function), 750  
 vsxseg2bv\_int32xm4\_int32x2xm4 (C function), 750  
 vsxseg2bv\_int64xm1\_int64x2xm1 (C function), 750  
 vsxseg2bv\_int64xm2\_int64x2xm2 (C function), 750  
 vsxseg2bv\_int64xm4\_int64x2xm4 (C function), 750  
 vsxseg2bv\_int8xm1\_int8x2xm1 (C function), 750  
 vsxseg2bv\_int8xm2\_int8x2xm2 (C function), 750  
 vsxseg2bv\_int8xm4\_int8x2xm4 (C function), 750  
 vsxseg2bv\_mask\_int16xm1\_int16x2xm1 (C function), 751  
 vsxseg2bv\_mask\_int16xm2\_int16x2xm2 (C function), 751  
 vsxseg2bv\_mask\_int16xm4\_int16x2xm4 (C function), 751  
 vsxseg2bv\_mask\_int32xm1\_int32x2xm1 (C function), 751  
 vsxseg2bv\_mask\_int32xm2\_int32x2xm2 (C function), 751  
 vsxseg2bv\_mask\_int32xm4\_int32x2xm4 (C function), 751  
 vsxseg2bv\_mask\_int64xm1\_int64x2xm1 (C function), 751  
 vsxseg2bv\_mask\_int64xm2\_int64x2xm2 (C function), 751  
 vsxseg2bv\_mask\_int64xm4\_int64x2xm4 (C function), 751  
 vsxseg2bv\_mask\_int8xm1\_int8x2xm1 (C function), 751  
 vsxseg2bv\_mask\_int8xm2\_int8x2xm2 (C function), 751  
 vsxseg2bv\_mask\_int8xm4\_int8x2xm4 (C function), 751  
 vsxseg2bv\_mask\_uint16xm1\_uint16x2xm1 (C function), 751  
 vsxseg2bv\_mask\_uint16xm2\_uint16x2xm2 (C function), 751  
 vsxseg2bv\_mask\_uint16xm4\_uint16x2xm4 (C function), 752  
 vsxseg2bv\_mask\_uint32xm1\_uint32x2xm1 (C function), 752  
 vsxseg2bv\_mask\_uint32xm2\_uint32x2xm2 (C function), 752  
 vsxseg2bv\_mask\_uint32xm4\_uint32x2xm4 (C function), 752  
 vsxseg2bv\_mask\_uint64xm1\_uint64x2xm1 (C function), 752  
 vsxseg2bv\_mask\_uint64xm2\_uint64x2xm2 (C function), 752  
 vsxseg2bv\_mask\_uint64xm4\_uint64x2xm4 (C function), 752  
 vsxseg2bv\_mask\_uint8xm1\_uint8x2xm1 (C function), 752  
 vsxseg2bv\_mask\_uint8xm2\_uint8x2xm2 (C function), 752  
 vsxseg2bv\_mask\_uint8xm4\_uint8x2xm4 (C function), 752  
 vsxseg2bv\_uint16xm1\_uint16x2xm1 (C function), 750  
 vsxseg2bv\_uint16xm2\_uint16x2xm2 (C function), 750  
 vsxseg2bv\_uint16xm4\_uint16x2xm4 (C function), 750  
 vsxseg2bv\_uint32xm1\_uint32x2xm1 (C function), 750  
 vsxseg2bv\_uint32xm2\_uint32x2xm2 (C function), 750  
 vsxseg2bv\_uint32xm4\_uint32x2xm4 (C function), 750  
 vsxseg2bv\_uint64xm1\_uint64x2xm1 (C function), 750  
 vsxseg2bv\_uint64xm2\_uint64x2xm2 (C function), 750  
 vsxseg2bv\_uint64xm4\_uint64x2xm4 (C function), 750  
 vsxseg2bv\_uint8xm1\_uint8x2xm1 (C function), 750  
 vsxseg2bv\_uint8xm2\_uint8x2xm2 (C function), 751  
 vsxseg2bv\_uint8xm4\_uint8x2xm4 (C function), 751  
 vsxseg2ev\_float16xm1\_float16x2xm1 (C function), 752  
 vsxseg2ev\_float16xm2\_float16x2xm2 (C function), 752  
 vsxseg2ev\_float16xm4\_float16x2xm4 (C function), 752  
 vsxseg2ev\_float32xm1\_float32x2xm1 (C function), 753  
 vsxseg2ev\_float32xm2\_float32x2xm2 (C function), 753  
 vsxseg2ev\_float32xm4\_float32x2xm4 (C function), 753  
 vsxseg2ev\_float64xm1\_float64x2xm1 (C function), 753



vsxseg2ev\_float64xm2\_float64x2xm2 (C function), 753  
 vsxseg2ev\_float64xm4\_float64x2xm4 (C function), 753  
 vsxseg2ev\_int16xm1\_int16x2xm1 (C function), 753  
 vsxseg2ev\_int16xm2\_int16x2xm2 (C function), 753  
 vsxseg2ev\_int16xm4\_int16x2xm4 (C function), 753  
 vsxseg2ev\_int32xm1\_int32x2xm1 (C function), 753  
 vsxseg2ev\_int32xm2\_int32x2xm2 (C function), 753  
 vsxseg2ev\_int32xm4\_int32x2xm4 (C function), 753  
 vsxseg2ev\_int64xm1\_int64x2xm1 (C function), 753  
 vsxseg2ev\_int64xm2\_int64x2xm2 (C function), 753  
 vsxseg2ev\_int64xm4\_int64x2xm4 (C function), 753  
 vsxseg2ev\_int8xm1\_int8x2xm1 (C function), 753  
 vsxseg2ev\_int8xm2\_int8x2xm2 (C function), 753  
 vsxseg2ev\_int8xm4\_int8x2xm4 (C function), 753  
 vsxseg2ev\_mask\_float16xm1\_float16x2xm1 (C function), 754  
 vsxseg2ev\_mask\_float16xm2\_float16x2xm2 (C function), 754  
 vsxseg2ev\_mask\_float16xm4\_float16x2xm4 (C function), 754  
 vsxseg2ev\_mask\_float32xm1\_float32x2xm1 (C function), 754  
 vsxseg2ev\_mask\_float32xm2\_float32x2xm2 (C function), 754  
 vsxseg2ev\_mask\_float32xm4\_float32x2xm4 (C function), 754  
 vsxseg2ev\_mask\_float64xm1\_float64x2xm1 (C function), 754  
 vsxseg2ev\_mask\_float64xm2\_float64x2xm2 (C function), 754  
 vsxseg2ev\_mask\_float64xm4\_float64x2xm4 (C function), 754  
 vsxseg2ev\_mask\_int16xm1\_int16x2xm1 (C function), 755  
 vsxseg2ev\_mask\_int16xm2\_int16x2xm2 (C function), 755  
 vsxseg2ev\_mask\_int16xm4\_int16x2xm4 (C function), 755  
 vsxseg2ev\_mask\_int32xm1\_int32x2xm1 (C function), 755  
 vsxseg2ev\_mask\_int32xm2\_int32x2xm2 (C function), 755  
 vsxseg2ev\_mask\_int32xm4\_int32x2xm4 (C function), 755  
 vsxseg2ev\_mask\_int64xm1\_int64x2xm1 (C function), 755  
 vsxseg2ev\_mask\_int64xm2\_int64x2xm2 (C function), 755  
 vsxseg2ev\_mask\_int64xm4\_int64x2xm4 (C function), 755  
 vsxseg2ev\_mask\_int8xm1\_int8x2xm1 (C function), 755  
 vsxseg2ev\_mask\_int8xm2\_int8x2xm2 (C function), 755  
 vsxseg2ev\_mask\_int8xm4\_int8x2xm4 (C function), 755  
 vsxseg2ev\_mask\_uint16xm1\_uint16x2xm1 (C function), 755  
 vsxseg2ev\_mask\_uint16xm2\_uint16x2xm2 (C function), 755  
 vsxseg2ev\_mask\_uint16xm4\_uint16x2xm4 (C function), 755  
 vsxseg2ev\_mask\_uint32xm1\_uint32x2xm1 (C function), 755  
 vsxseg2ev\_mask\_uint32xm2\_uint32x2xm2 (C function), 755  
 vsxseg2ev\_mask\_uint32xm4\_uint32x2xm4 (C function), 755  
 vsxseg2ev\_mask\_uint64xm1\_uint64x2xm1 (C function), 756  
 vsxseg2ev\_mask\_uint64xm2\_uint64x2xm2 (C function), 756  
 vsxseg2ev\_mask\_uint64xm4\_uint64x2xm4 (C function), 756  
 vsxseg2ev\_mask\_uint8xm1\_uint8x2xm1 (C function), 756  
 vsxseg2ev\_mask\_uint8xm2\_uint8x2xm2 (C function), 756  
 vsxseg2ev\_mask\_uint8xm4\_uint8x2xm4 (C function), 756  
 vsxseg2ev\_uint16xm1\_uint16x2xm1 (C function), 753  
 vsxseg2ev\_uint16xm2\_uint16x2xm2 (C function), 753  
 vsxseg2ev\_uint16xm4\_uint16x2xm4 (C function), 753  
 vsxseg2ev\_uint32xm1\_uint32x2xm1 (C function), 753  
 vsxseg2ev\_uint32xm2\_uint32x2xm2 (C function), 753  
 vsxseg2ev\_uint32xm4\_uint32x2xm4 (C function), 754  
 vsxseg2ev\_uint64xm1\_uint64x2xm1 (C function), 754  
 vsxseg2ev\_uint64xm2\_uint64x2xm2 (C function), 754  
 vsxseg2ev\_uint64xm4\_uint64x2xm4 (C function), 754  
 vsxseg2ev\_uint8xm1\_uint8x2xm1 (C function), 754  
 vsxseg2ev\_uint8xm2\_uint8x2xm2 (C function), 754  
 vsxseg2ev\_uint8xm4\_uint8x2xm4 (C function), 754  
 vsxseg2hvf\_int16xm1\_int16x2xm1 (C function), 756  
 vsxseg2hvf\_int16xm2\_int16x2xm2 (C function), 756  
 vsxseg2hvf\_int16xm4\_int16x2xm4 (C function), 756  
 vsxseg2hvf\_int32xm1\_int32x2xm1 (C function), 756  
 vsxseg2hvf\_int32xm2\_int32x2xm2 (C function), 756  
 vsxseg2hvf\_int32xm4\_int32x2xm4 (C function), 756  
 vsxseg2hvf\_int64xm1\_int64x2xm1 (C function), 756  
 vsxseg2hvf\_int64xm2\_int64x2xm2 (C function), 756  
 vsxseg2hvf\_int64xm4\_int64x2xm4 (C function), 756  
 vsxseg2hvf\_int8xm1\_int8x2xm1 (C function), 757  
 vsxseg2hvf\_int8xm2\_int8x2xm2 (C function), 757  
 vsxseg2hvf\_int8xm4\_int8x2xm4 (C function), 757  
 vsxseg2hvf\_mask\_int16xm1\_int16x2xm1 (C function), 757  
 vsxseg2hvf\_mask\_int16xm2\_int16x2xm2 (C function), 757  
 vsxseg2hvf\_mask\_int16xm4\_int16x2xm4 (C function), 757

vsxseg2hvv\_mask\_int32xm1\_int32x2xm1 (C function), 757  
 vsxseg2hvv\_mask\_int32xm2\_int32x2xm2 (C function), 758  
 vsxseg2hvv\_mask\_int32xm4\_int32x2xm4 (C function), 758  
 vsxseg2hvv\_mask\_int64xm1\_int64x2xm1 (C function), 758  
 vsxseg2hvv\_mask\_int64xm2\_int64x2xm2 (C function), 758  
 vsxseg2hvv\_mask\_int64xm4\_int64x2xm4 (C function), 758  
 vsxseg2hvv\_mask\_int8xm1\_int8x2xm1 (C function), 758  
 vsxseg2hvv\_mask\_int8xm2\_int8x2xm2 (C function), 758  
 vsxseg2hvv\_mask\_int8xm4\_int8x2xm4 (C function), 758  
 vsxseg2hvv\_mask\_uint16xm1\_uint16x2xm1 (C function), 758  
 vsxseg2hvv\_mask\_uint16xm2\_uint16x2xm2 (C function), 758  
 vsxseg2hvv\_mask\_uint16xm4\_uint16x2xm4 (C function), 758  
 vsxseg2hvv\_mask\_uint32xm1\_uint32x2xm1 (C function), 758  
 vsxseg2hvv\_mask\_uint32xm2\_uint32x2xm2 (C function), 758  
 vsxseg2hvv\_mask\_uint32xm4\_uint32x2xm4 (C function), 758  
 vsxseg2hvv\_mask\_uint64xm1\_uint64x2xm1 (C function), 758  
 vsxseg2hvv\_mask\_uint64xm2\_uint64x2xm2 (C function), 758  
 vsxseg2hvv\_mask\_uint64xm4\_uint64x2xm4 (C function), 758  
 vsxseg2hvv\_mask\_uint8xm1\_uint8x2xm1 (C function), 758  
 vsxseg2hvv\_mask\_uint8xm2\_uint8x2xm2 (C function), 759  
 vsxseg2hvv\_mask\_uint8xm4\_uint8x2xm4 (C function), 759  
 vsxseg2hvv\_uint16xm1\_uint16x2xm1 (C function), 757  
 vsxseg2hvv\_uint16xm2\_uint16x2xm2 (C function), 757  
 vsxseg2hvv\_uint16xm4\_uint16x2xm4 (C function), 757  
 vsxseg2hvv\_uint32xm1\_uint32x2xm1 (C function), 757  
 vsxseg2hvv\_uint32xm2\_uint32x2xm2 (C function), 757  
 vsxseg2hvv\_uint32xm4\_uint32x2xm4 (C function), 757  
 vsxseg2hvv\_uint64xm1\_uint64x2xm1 (C function), 757  
 vsxseg2hvv\_uint64xm2\_uint64x2xm2 (C function), 757  
 vsxseg2hvv\_uint64xm4\_uint64x2xm4 (C function), 757  
 vsxseg2hvv\_uint8xm1\_uint8x2xm1 (C function), 757  
 vsxseg2hvv\_uint8xm2\_uint8x2xm2 (C function), 757  
 vsxseg2hvv\_uint8xm4\_uint8x2xm4 (C function), 757  
 vsxseg2wvv\_int16xm1\_int16x2xm1 (C function), 759  
 vsxseg2wvv\_int16xm2\_int16x2xm2 (C function), 759  
 vsxseg2wvv\_int16xm4\_int16x2xm4 (C function), 759  
 vsxseg2wvv\_int32xm1\_int32x2xm1 (C function), 759  
 vsxseg2wvv\_int32xm2\_int32x2xm2 (C function), 759  
 vsxseg2wvv\_int32xm4\_int32x2xm4 (C function), 759  
 vsxseg2wvv\_int64xm1\_int64x2xm1 (C function), 759  
 vsxseg2wvv\_int64xm2\_int64x2xm2 (C function), 759  
 vsxseg2wvv\_int64xm4\_int64x2xm4 (C function), 759  
 vsxseg2wvv\_int8xm1\_int8x2xm1 (C function), 759  
 vsxseg2wvv\_int8xm2\_int8x2xm2 (C function), 759  
 vsxseg2wvv\_int8xm4\_int8x2xm4 (C function), 759  
 vsxseg2wvv\_mask\_int16xm1\_int16x2xm1 (C function), 760  
 vsxseg2wvv\_mask\_int16xm2\_int16x2xm2 (C function), 760  
 vsxseg2wvv\_mask\_int16xm4\_int16x2xm4 (C function), 760  
 vsxseg2wvv\_mask\_int32xm1\_int32x2xm1 (C function), 760  
 vsxseg2wvv\_mask\_int32xm2\_int32x2xm2 (C function), 760  
 vsxseg2wvv\_mask\_int32xm4\_int32x2xm4 (C function), 760  
 vsxseg2wvv\_mask\_int64xm1\_int64x2xm1 (C function), 760  
 vsxseg2wvv\_mask\_int64xm2\_int64x2xm2 (C function), 760  
 vsxseg2wvv\_mask\_int64xm4\_int64x2xm4 (C function), 761  
 vsxseg2wvv\_mask\_int8xm1\_int8x2xm1 (C function), 761  
 vsxseg2wvv\_mask\_int8xm2\_int8x2xm2 (C function), 761  
 vsxseg2wvv\_mask\_int8xm4\_int8x2xm4 (C function), 761  
 vsxseg2wvv\_mask\_uint16xm1\_uint16x2xm1 (C function), 761  
 vsxseg2wvv\_mask\_uint16xm2\_uint16x2xm2 (C function), 761  
 vsxseg2wvv\_mask\_uint16xm4\_uint16x2xm4 (C function), 761  
 vsxseg2wvv\_mask\_uint32xm1\_uint32x2xm1 (C function), 761  
 vsxseg2wvv\_mask\_uint32xm2\_uint32x2xm2 (C function), 761  
 vsxseg2wvv\_mask\_uint32xm4\_uint32x2xm4 (C function), 761  
 vsxseg2wvv\_mask\_uint64xm1\_uint64x2xm1 (C function), 761  
 vsxseg2wvv\_mask\_uint64xm2\_uint64x2xm2 (C function), 761  
 vsxseg2wvv\_mask\_uint64xm4\_uint64x2xm4 (C function), 761  
 vsxseg2wvv\_mask\_uint8xm1\_uint8x2xm1 (C function), 761  
 vsxseg2wvv\_mask\_uint8xm2\_uint8x2xm2 (C function), 761  
 vsxseg2wvv\_mask\_uint8xm4\_uint8x2xm4 (C function), 761



vsxseg2wv\_uint16xm1\_uint16x2xm1 (C function), 759  
 vsxseg2wv\_uint16xm2\_uint16x2xm2 (C function), 759  
 vsxseg2wv\_uint16xm4\_uint16x2xm4 (C function), 760  
 vsxseg2wv\_uint32xm1\_uint32x2xm1 (C function), 760  
 vsxseg2wv\_uint32xm2\_uint32x2xm2 (C function), 760  
 vsxseg2wv\_uint32xm4\_uint32x2xm4 (C function), 760  
 vsxseg2wv\_uint64xm1\_uint64x2xm1 (C function), 760  
 vsxseg2wv\_uint64xm2\_uint64x2xm2 (C function), 760  
 vsxseg2wv\_uint64xm4\_uint64x2xm4 (C function), 760  
 vsxseg2wv\_uint8xm1\_uint8x2xm1 (C function), 760  
 vsxseg2wv\_uint8xm2\_uint8x2xm2 (C function), 760  
 vsxseg2wv\_uint8xm4\_uint8x2xm4 (C function), 760  
 vsxseg3bv\_int16xm1\_int16x3xm1 (C function), 762  
 vsxseg3bv\_int16xm2\_int16x3xm2 (C function), 762  
 vsxseg3bv\_int32xm1\_int32x3xm1 (C function), 762  
 vsxseg3bv\_int32xm2\_int32x3xm2 (C function), 762  
 vsxseg3bv\_int64xm1\_int64x3xm1 (C function), 762  
 vsxseg3bv\_int64xm2\_int64x3xm2 (C function), 762  
 vsxseg3bv\_int8xm1\_int8x3xm1 (C function), 762  
 vsxseg3bv\_int8xm2\_int8x3xm2 (C function), 762  
 vsxseg3bv\_mask\_int16xm1\_int16x3xm1 (C function), 763  
 vsxseg3bv\_mask\_int16xm2\_int16x3xm2 (C function), 763  
 vsxseg3bv\_mask\_int32xm1\_int32x3xm1 (C function), 763  
 vsxseg3bv\_mask\_int32xm2\_int32x3xm2 (C function), 763  
 vsxseg3bv\_mask\_int64xm1\_int64x3xm1 (C function), 763  
 vsxseg3bv\_mask\_int64xm2\_int64x3xm2 (C function), 763  
 vsxseg3bv\_mask\_int8xm1\_int8x3xm1 (C function), 763  
 vsxseg3bv\_mask\_int8xm2\_int8x3xm2 (C function), 763  
 vsxseg3bv\_mask\_uint16xm1\_uint16x3xm1 (C function), 763  
 vsxseg3bv\_mask\_uint16xm2\_uint16x3xm2 (C function), 763  
 vsxseg3bv\_mask\_uint32xm1\_uint32x3xm1 (C function), 763  
 vsxseg3bv\_mask\_uint32xm2\_uint32x3xm2 (C function), 763  
 vsxseg3bv\_mask\_uint64xm1\_uint64x3xm1 (C function), 763  
 vsxseg3bv\_mask\_uint64xm2\_uint64x3xm2 (C function), 763  
 vsxseg3bv\_mask\_uint8xm1\_uint8x3xm1 (C function), 763  
 vsxseg3bv\_mask\_uint8xm2\_uint8x3xm2 (C function), 763  
 vsxseg3bv\_uint16xm1\_uint16x3xm1 (C function), 762  
 vsxseg3bv\_uint16xm2\_uint16x3xm2 (C function), 762  
 vsxseg3bv\_uint32xm1\_uint32x3xm1 (C function), 762  
 vsxseg3bv\_uint32xm2\_uint32x3xm2 (C function), 762  
 vsxseg3bv\_uint64xm1\_uint64x3xm1 (C function), 762  
 vsxseg3bv\_uint64xm2\_uint64x3xm2 (C function), 762  
 vsxseg3bv\_uint8xm1\_uint8x3xm1 (C function), 762  
 vsxseg3bv\_uint8xm2\_uint8x3xm2 (C function), 762  
 vsxseg3ev\_float16xm1\_float16x3xm1 (C function), 764  
 vsxseg3ev\_float16xm2\_float16x3xm2 (C function), 764  
 vsxseg3ev\_float32xm1\_float32x3xm1 (C function), 764  
 vsxseg3ev\_float32xm2\_float32x3xm2 (C function), 764  
 vsxseg3ev\_float64xm1\_float64x3xm1 (C function), 764  
 vsxseg3ev\_float64xm2\_float64x3xm2 (C function), 764  
 vsxseg3ev\_int16xm1\_int16x3xm1 (C function), 764  
 vsxseg3ev\_int16xm2\_int16x3xm2 (C function), 764  
 vsxseg3ev\_int32xm1\_int32x3xm1 (C function), 764  
 vsxseg3ev\_int32xm2\_int32x3xm2 (C function), 764  
 vsxseg3ev\_int64xm1\_int64x3xm1 (C function), 764  
 vsxseg3ev\_int64xm2\_int64x3xm2 (C function), 764  
 vsxseg3ev\_int8xm1\_int8x3xm1 (C function), 764  
 vsxseg3ev\_int8xm2\_int8x3xm2 (C function), 764  
 vsxseg3ev\_mask\_float16xm1\_float16x3xm1 (C function), 765  
 vsxseg3ev\_mask\_float16xm2\_float16x3xm2 (C function), 765  
 vsxseg3ev\_mask\_float32xm1\_float32x3xm1 (C function), 765  
 vsxseg3ev\_mask\_float32xm2\_float32x3xm2 (C function), 765  
 vsxseg3ev\_mask\_float64xm1\_float64x3xm1 (C function), 765  
 vsxseg3ev\_mask\_float64xm2\_float64x3xm2 (C function), 765  
 vsxseg3ev\_mask\_int16xm1\_int16x3xm1 (C function), 765  
 vsxseg3ev\_mask\_int16xm2\_int16x3xm2 (C function), 765  
 vsxseg3ev\_mask\_int32xm1\_int32x3xm1 (C function), 765  
 vsxseg3ev\_mask\_int32xm2\_int32x3xm2 (C function), 765  
 vsxseg3ev\_mask\_int64xm1\_int64x3xm1 (C function), 766  
 vsxseg3ev\_mask\_int64xm2\_int64x3xm2 (C function), 766  
 vsxseg3ev\_mask\_int8xm1\_int8x3xm1 (C function), 766  
 vsxseg3ev\_mask\_int8xm2\_int8x3xm2 (C function), 766  
 vsxseg3ev\_mask\_uint16xm1\_uint16x3xm1 (C function), 766  
 vsxseg3ev\_mask\_uint16xm2\_uint16x3xm2 (C function), 766  
 vsxseg3ev\_mask\_uint32xm1\_uint32x3xm1 (C function), 766  
 vsxseg3ev\_mask\_uint32xm2\_uint32x3xm2 (C function), 766  
 vsxseg3ev\_mask\_uint64xm1\_uint64x3xm1 (C function), 766

vsxseg3ev\_mask\_uint64xm2\_uint64x3xm2 (C function), 766  
 vsxseg3ev\_mask\_uint8xm1\_uint8x3xm1 (C function), 766  
 vsxseg3ev\_mask\_uint8xm2\_uint8x3xm2 (C function), 766  
 vsxseg3ev\_uint16xm1\_uint16x3xm1 (C function), 764  
 vsxseg3ev\_uint16xm2\_uint16x3xm2 (C function), 764  
 vsxseg3ev\_uint32xm1\_uint32x3xm1 (C function), 765  
 vsxseg3ev\_uint32xm2\_uint32x3xm2 (C function), 765  
 vsxseg3ev\_uint64xm1\_uint64x3xm1 (C function), 765  
 vsxseg3ev\_uint64xm2\_uint64x3xm2 (C function), 765  
 vsxseg3ev\_uint8xm1\_uint8x3xm1 (C function), 765  
 vsxseg3ev\_uint8xm2\_uint8x3xm2 (C function), 765  
 vsxseg3hv\_int16xm1\_int16x3xm1 (C function), 766  
 vsxseg3hv\_int16xm2\_int16x3xm2 (C function), 767  
 vsxseg3hv\_int32xm1\_int32x3xm1 (C function), 767  
 vsxseg3hv\_int32xm2\_int32x3xm2 (C function), 767  
 vsxseg3hv\_int64xm1\_int64x3xm1 (C function), 767  
 vsxseg3hv\_int64xm2\_int64x3xm2 (C function), 767  
 vsxseg3hv\_int8xm1\_int8x3xm1 (C function), 767  
 vsxseg3hv\_int8xm2\_int8x3xm2 (C function), 767  
 vsxseg3hv\_mask\_int16xm1\_int16x3xm1 (C function), 767  
 vsxseg3hv\_mask\_int16xm2\_int16x3xm2 (C function), 767  
 vsxseg3hv\_mask\_int32xm1\_int32x3xm1 (C function), 767  
 vsxseg3hv\_mask\_int32xm2\_int32x3xm2 (C function), 767  
 vsxseg3hv\_mask\_int64xm1\_int64x3xm1 (C function), 768  
 vsxseg3hv\_mask\_int64xm2\_int64x3xm2 (C function), 768  
 vsxseg3hv\_mask\_int8xm1\_int8x3xm1 (C function), 768  
 vsxseg3hv\_mask\_int8xm2\_int8x3xm2 (C function), 768  
 vsxseg3hv\_mask\_uint16xm1\_uint16x3xm1 (C function), 768  
 vsxseg3hv\_mask\_uint16xm2\_uint16x3xm2 (C function), 768  
 vsxseg3hv\_mask\_uint32xm1\_uint32x3xm1 (C function), 768  
 vsxseg3hv\_mask\_uint32xm2\_uint32x3xm2 (C function), 768  
 vsxseg3hv\_mask\_uint64xm1\_uint64x3xm1 (C function), 768  
 vsxseg3hv\_mask\_uint64xm2\_uint64x3xm2 (C function), 768  
 vsxseg3hv\_mask\_uint8xm1\_uint8x3xm1 (C function), 768  
 vsxseg3hv\_mask\_uint8xm2\_uint8x3xm2 (C function), 768  
 vsxseg3hv\_uint16xm1\_uint16x3xm1 (C function), 767  
 vsxseg3hv\_uint16xm2\_uint16x3xm2 (C function), 767  
 vsxseg3hv\_uint32xm1\_uint32x3xm1 (C function), 767  
 vsxseg3hv\_uint32xm2\_uint32x3xm2 (C function), 767  
 vsxseg3hv\_uint64xm1\_uint64x3xm1 (C function), 767  
 vsxseg3hv\_uint64xm2\_uint64x3xm2 (C function), 767  
 vsxseg3hv\_uint8xm1\_uint8x3xm1 (C function), 767  
 vsxseg3hv\_uint8xm2\_uint8x3xm2 (C function), 767  
 vsxseg3wv\_int16xm1\_int16x3xm1 (C function), 768  
 vsxseg3wv\_int16xm2\_int16x3xm2 (C function), 769  
 vsxseg3wv\_int32xm1\_int32x3xm1 (C function), 769  
 vsxseg3wv\_int32xm2\_int32x3xm2 (C function), 769  
 vsxseg3wv\_int64xm1\_int64x3xm1 (C function), 769  
 vsxseg3wv\_int64xm2\_int64x3xm2 (C function), 769  
 vsxseg3wv\_int8xm1\_int8x3xm1 (C function), 769  
 vsxseg3wv\_int8xm2\_int8x3xm2 (C function), 769  
 vsxseg3wv\_mask\_int16xm1\_int16x3xm1 (C function), 769  
 vsxseg3wv\_mask\_int16xm2\_int16x3xm2 (C function), 769  
 vsxseg3wv\_mask\_int32xm1\_int32x3xm1 (C function), 769  
 vsxseg3wv\_mask\_int32xm2\_int32x3xm2 (C function), 769  
 vsxseg3wv\_mask\_int64xm1\_int64x3xm1 (C function), 770  
 vsxseg3wv\_mask\_int64xm2\_int64x3xm2 (C function), 770  
 vsxseg3wv\_mask\_int8xm1\_int8x3xm1 (C function), 770  
 vsxseg3wv\_mask\_int8xm2\_int8x3xm2 (C function), 770  
 vsxseg3wv\_mask\_uint16xm1\_uint16x3xm1 (C function), 770  
 vsxseg3wv\_mask\_uint16xm2\_uint16x3xm2 (C function), 770  
 vsxseg3wv\_mask\_uint32xm1\_uint32x3xm1 (C function), 770  
 vsxseg3wv\_mask\_uint32xm2\_uint32x3xm2 (C function), 770  
 vsxseg3wv\_mask\_uint64xm1\_uint64x3xm1 (C function), 770  
 vsxseg3wv\_mask\_uint64xm2\_uint64x3xm2 (C function), 770  
 vsxseg3wv\_mask\_uint8xm1\_uint8x3xm1 (C function), 770  
 vsxseg3wv\_mask\_uint8xm2\_uint8x3xm2 (C function), 770  
 vsxseg3wv\_uint16xm1\_uint16x3xm1 (C function), 769  
 vsxseg3wv\_uint16xm2\_uint16x3xm2 (C function), 769  
 vsxseg3wv\_uint32xm1\_uint32x3xm1 (C function), 769  
 vsxseg3wv\_uint32xm2\_uint32x3xm2 (C function), 769  
 vsxseg3wv\_uint64xm1\_uint64x3xm1 (C function), 769  
 vsxseg3wv\_uint64xm2\_uint64x3xm2 (C function), 769  
 vsxseg3wv\_uint8xm1\_uint8x3xm1 (C function), 769  
 vsxseg3wv\_uint8xm2\_uint8x3xm2 (C function), 769  
 vsxseg4bv\_int16xm1\_int16x4xm1 (C function), 770  
 vsxseg4bv\_int16xm2\_int16x4xm2 (C function), 771

vsxseg4bv\_int32xm1\_int32x4xm1 (C function), 771  
 vsxseg4bv\_int32xm2\_int32x4xm2 (C function), 771  
 vsxseg4bv\_int64xm1\_int64x4xm1 (C function), 771  
 vsxseg4bv\_int64xm2\_int64x4xm2 (C function), 771  
 vsxseg4bv\_int8xm1\_int8x4xm1 (C function), 771  
 vsxseg4bv\_int8xm2\_int8x4xm2 (C function), 771  
 vsxseg4bv\_mask\_int16xm1\_int16x4xm1 (C function), 771  
 vsxseg4bv\_mask\_int16xm2\_int16x4xm2 (C function), 771  
 vsxseg4bv\_mask\_int32xm1\_int32x4xm1 (C function), 771  
 vsxseg4bv\_mask\_int32xm2\_int32x4xm2 (C function), 771  
 vsxseg4bv\_mask\_int64xm1\_int64x4xm1 (C function), 772  
 vsxseg4bv\_mask\_int64xm2\_int64x4xm2 (C function), 772  
 vsxseg4bv\_mask\_int8xm1\_int8x4xm1 (C function), 772  
 vsxseg4bv\_mask\_int8xm2\_int8x4xm2 (C function), 772  
 vsxseg4bv\_mask\_uint16xm1\_uint16x4xm1 (C function), 772  
 vsxseg4bv\_mask\_uint16xm2\_uint16x4xm2 (C function), 772  
 vsxseg4bv\_mask\_uint32xm1\_uint32x4xm1 (C function), 772  
 vsxseg4bv\_mask\_uint32xm2\_uint32x4xm2 (C function), 772  
 vsxseg4bv\_mask\_uint64xm1\_uint64x4xm1 (C function), 772  
 vsxseg4bv\_mask\_uint64xm2\_uint64x4xm2 (C function), 772  
 vsxseg4bv\_mask\_uint8xm1\_uint8x4xm1 (C function), 772  
 vsxseg4bv\_mask\_uint8xm2\_uint8x4xm2 (C function), 772  
 vsxseg4bv\_uint16xm1\_uint16x4xm1 (C function), 771  
 vsxseg4bv\_uint16xm2\_uint16x4xm2 (C function), 771  
 vsxseg4bv\_uint32xm1\_uint32x4xm1 (C function), 771  
 vsxseg4bv\_uint32xm2\_uint32x4xm2 (C function), 771  
 vsxseg4bv\_uint64xm1\_uint64x4xm1 (C function), 771  
 vsxseg4bv\_uint64xm2\_uint64x4xm2 (C function), 771  
 vsxseg4bv\_uint8xm1\_uint8x4xm1 (C function), 771  
 vsxseg4bv\_uint8xm2\_uint8x4xm2 (C function), 771  
 vsxseg4ev\_float16xm1\_float16x4xm1 (C function), 772  
 vsxseg4ev\_float16xm2\_float16x4xm2 (C function), 773  
 vsxseg4ev\_float32xm1\_float32x4xm1 (C function), 773  
 vsxseg4ev\_float32xm2\_float32x4xm2 (C function), 773  
 vsxseg4ev\_float64xm1\_float64x4xm1 (C function), 773  
 vsxseg4ev\_float64xm2\_float64x4xm2 (C function), 773  
 vsxseg4ev\_int16xm1\_int16x4xm1 (C function), 773  
 vsxseg4ev\_int16xm2\_int16x4xm2 (C function), 773  
 vsxseg4ev\_int32xm1\_int32x4xm1 (C function), 773  
 vsxseg4ev\_int32xm2\_int32x4xm2 (C function), 773  
 vsxseg4ev\_int64xm1\_int64x4xm1 (C function), 773  
 vsxseg4ev\_int64xm2\_int64x4xm2 (C function), 773  
 vsxseg4ev\_int8xm1\_int8x4xm1 (C function), 773  
 vsxseg4ev\_int8xm2\_int8x4xm2 (C function), 773  
 vsxseg4ev\_mask\_float16xm1\_float16x4xm1 (C function), 774  
 vsxseg4ev\_mask\_float16xm2\_float16x4xm2 (C function), 774  
 vsxseg4ev\_mask\_float32xm1\_float32x4xm1 (C function), 774  
 vsxseg4ev\_mask\_float32xm2\_float32x4xm2 (C function), 774  
 vsxseg4ev\_mask\_float64xm1\_float64x4xm1 (C function), 774  
 vsxseg4ev\_mask\_float64xm2\_float64x4xm2 (C function), 774  
 vsxseg4ev\_mask\_int16xm1\_int16x4xm1 (C function), 774  
 vsxseg4ev\_mask\_int16xm2\_int16x4xm2 (C function), 774  
 vsxseg4ev\_mask\_int32xm1\_int32x4xm1 (C function), 774  
 vsxseg4ev\_mask\_int32xm2\_int32x4xm2 (C function), 774  
 vsxseg4ev\_mask\_int64xm1\_int64x4xm1 (C function), 774  
 vsxseg4ev\_mask\_int64xm2\_int64x4xm2 (C function), 774  
 vsxseg4ev\_mask\_int8xm1\_int8x4xm1 (C function), 774  
 vsxseg4ev\_mask\_int8xm2\_int8x4xm2 (C function), 774  
 vsxseg4ev\_mask\_uint16xm1\_uint16x4xm1 (C function), 774  
 vsxseg4ev\_mask\_uint16xm2\_uint16x4xm2 (C function), 774  
 vsxseg4ev\_mask\_uint32xm1\_uint32x4xm1 (C function), 774  
 vsxseg4ev\_mask\_uint32xm2\_uint32x4xm2 (C function), 775  
 vsxseg4ev\_mask\_uint64xm1\_uint64x4xm1 (C function), 775  
 vsxseg4ev\_mask\_uint64xm2\_uint64x4xm2 (C function), 775  
 vsxseg4ev\_mask\_uint8xm1\_uint8x4xm1 (C function), 775  
 vsxseg4ev\_mask\_uint8xm2\_uint8x4xm2 (C function), 775  
 vsxseg4ev\_uint16xm1\_uint16x4xm1 (C function), 773  
 vsxseg4ev\_uint16xm2\_uint16x4xm2 (C function), 773  
 vsxseg4ev\_uint32xm1\_uint32x4xm1 (C function), 773  
 vsxseg4ev\_uint32xm2\_uint32x4xm2 (C function), 773  
 vsxseg4ev\_uint64xm1\_uint64x4xm1 (C function), 773  
 vsxseg4ev\_uint64xm2\_uint64x4xm2 (C function), 773  
 vsxseg4ev\_uint8xm1\_uint8x4xm1 (C function), 773  
 vsxseg4ev\_uint8xm2\_uint8x4xm2 (C function), 773

vsxseg4hv\_int16xm1\_int16x4xm1 (C function), 775  
 vsxseg4hv\_int16xm2\_int16x4xm2 (C function), 775  
 vsxseg4hv\_int32xm1\_int32x4xm1 (C function), 775  
 vsxseg4hv\_int32xm2\_int32x4xm2 (C function), 775  
 vsxseg4hv\_int64xm1\_int64x4xm1 (C function), 775  
 vsxseg4hv\_int64xm2\_int64x4xm2 (C function), 775  
 vsxseg4hv\_int8xm1\_int8x4xm1 (C function), 775  
 vsxseg4hv\_int8xm2\_int8x4xm2 (C function), 775  
 vsxseg4hv\_mask\_int16xm1\_int16x4xm1 (C function), 776  
 vsxseg4hv\_mask\_int16xm2\_int16x4xm2 (C function), 776  
 vsxseg4hv\_mask\_int32xm1\_int32x4xm1 (C function), 776  
 vsxseg4hv\_mask\_int32xm2\_int32x4xm2 (C function), 776  
 vsxseg4hv\_mask\_int64xm1\_int64x4xm1 (C function), 776  
 vsxseg4hv\_mask\_int64xm2\_int64x4xm2 (C function), 776  
 vsxseg4hv\_mask\_int8xm1\_int8x4xm1 (C function), 776  
 vsxseg4hv\_mask\_int8xm2\_int8x4xm2 (C function), 776  
 vsxseg4hv\_mask\_uint16xm1\_uint16x4xm1 (C function), 776  
 vsxseg4hv\_mask\_uint16xm2\_uint16x4xm2 (C function), 776  
 vsxseg4hv\_mask\_uint32xm1\_uint32x4xm1 (C function), 776  
 vsxseg4hv\_mask\_uint32xm2\_uint32x4xm2 (C function), 777  
 vsxseg4hv\_mask\_uint64xm1\_uint64x4xm1 (C function), 777  
 vsxseg4hv\_mask\_uint64xm2\_uint64x4xm2 (C function), 777  
 vsxseg4hv\_mask\_uint8xm1\_uint8x4xm1 (C function), 777  
 vsxseg4hv\_mask\_uint8xm2\_uint8x4xm2 (C function), 777  
 vsxseg4hv\_uint16xm1\_uint16x4xm1 (C function), 775  
 vsxseg4hv\_uint16xm2\_uint16x4xm2 (C function), 775  
 vsxseg4hv\_uint32xm1\_uint32x4xm1 (C function), 776  
 vsxseg4hv\_uint32xm2\_uint32x4xm2 (C function), 776  
 vsxseg4hv\_uint64xm1\_uint64x4xm1 (C function), 776  
 vsxseg4hv\_uint64xm2\_uint64x4xm2 (C function), 776  
 vsxseg4hv\_uint8xm1\_uint8x4xm1 (C function), 776  
 vsxseg4hv\_uint8xm2\_uint8x4xm2 (C function), 776  
 vsxseg4wv\_int16xm1\_int16x4xm1 (C function), 777  
 vsxseg4wv\_int16xm2\_int16x4xm2 (C function), 777  
 vsxseg4wv\_int32xm1\_int32x4xm1 (C function), 777  
 vsxseg4wv\_int32xm2\_int32x4xm2 (C function), 777  
 vsxseg4wv\_int64xm1\_int64x4xm1 (C function), 777  
 vsxseg4wv\_int64xm2\_int64x4xm2 (C function), 777  
 vsxseg4wv\_int8xm1\_int8x4xm1 (C function), 777  
 vsxseg4wv\_int8xm2\_int8x4xm2 (C function), 777  
 vsxseg4wv\_mask\_int16xm1\_int16x4xm1 (C function), 778  
 vsxseg4wv\_mask\_int16xm2\_int16x4xm2 (C function), 778  
 vsxseg4wv\_mask\_int32xm1\_int32x4xm1 (C function), 778  
 vsxseg4wv\_mask\_int32xm2\_int32x4xm2 (C function), 778  
 vsxseg4wv\_mask\_int64xm1\_int64x4xm1 (C function), 778  
 vsxseg4wv\_mask\_int64xm2\_int64x4xm2 (C function), 778  
 vsxseg4wv\_mask\_int8xm1\_int8x4xm1 (C function), 778  
 vsxseg4wv\_mask\_int8xm2\_int8x4xm2 (C function), 778  
 vsxseg4wv\_mask\_uint16xm1\_uint16x4xm1 (C function), 778  
 vsxseg4wv\_mask\_uint16xm2\_uint16x4xm2 (C function), 778  
 vsxseg4wv\_mask\_uint32xm1\_uint32x4xm1 (C function), 778  
 vsxseg4wv\_mask\_uint32xm2\_uint32x4xm2 (C function), 779  
 vsxseg4wv\_mask\_uint64xm1\_uint64x4xm1 (C function), 779  
 vsxseg4wv\_mask\_uint64xm2\_uint64x4xm2 (C function), 779  
 vsxseg4wv\_mask\_uint8xm1\_uint8x4xm1 (C function), 779  
 vsxseg4wv\_mask\_uint8xm2\_uint8x4xm2 (C function), 779  
 vsxseg4wv\_uint16xm1\_uint16x4xm1 (C function), 777  
 vsxseg4wv\_uint16xm2\_uint16x4xm2 (C function), 777  
 vsxseg4wv\_uint32xm1\_uint32x4xm1 (C function), 778  
 vsxseg4wv\_uint32xm2\_uint32x4xm2 (C function), 778  
 vsxseg4wv\_uint64xm1\_uint64x4xm1 (C function), 778  
 vsxseg4wv\_uint64xm2\_uint64x4xm2 (C function), 778  
 vsxseg4wv\_uint8xm1\_uint8x4xm1 (C function), 778  
 vsxseg4wv\_uint8xm2\_uint8x4xm2 (C function), 778  
 vsxseg5bv\_int16xm1\_int16x5xm1 (C function), 779  
 vsxseg5bv\_int32xm1\_int32x5xm1 (C function), 779  
 vsxseg5bv\_int64xm1\_int64x5xm1 (C function), 779  
 vsxseg5bv\_int8xm1\_int8x5xm1 (C function), 779  
 vsxseg5bv\_mask\_int16xm1\_int16x5xm1 (C function), 780  
 vsxseg5bv\_mask\_int32xm1\_int32x5xm1 (C function), 780  
 vsxseg5bv\_mask\_int64xm1\_int64x5xm1 (C function), 780  
 vsxseg5bv\_mask\_int8xm1\_int8x5xm1 (C function), 780  
 vsxseg5bv\_mask\_uint16xm1\_uint16x5xm1 (C function), 780  
 vsxseg5bv\_mask\_uint32xm1\_uint32x5xm1 (C function), 780  
 vsxseg5bv\_mask\_uint64xm1\_uint64x5xm1 (C function),

[780](#)  
[vsxseg5bv\\_mask\\_uint8xm1\\_uint8x5xm1 \(C function\),](#)  
[780](#)  
[vsxseg5bv\\_uint16xm1\\_uint16x5xm1 \(C function\),](#) [779](#)  
[vsxseg5bv\\_uint32xm1\\_uint32x5xm1 \(C function\),](#) [779](#)  
[vsxseg5bv\\_uint64xm1\\_uint64x5xm1 \(C function\),](#) [779](#)  
[vsxseg5bv\\_uint8xm1\\_uint8x5xm1 \(C function\),](#) [779](#)  
[vsxseg5ev\\_float16xm1\\_float16x5xm1 \(C function\),](#) [780](#)  
[vsxseg5ev\\_float32xm1\\_float32x5xm1 \(C function\),](#) [780](#)  
[vsxseg5ev\\_float64xm1\\_float64x5xm1 \(C function\),](#) [780](#)  
[vsxseg5ev\\_int16xm1\\_int16x5xm1 \(C function\),](#) [780](#)  
[vsxseg5ev\\_int32xm1\\_int32x5xm1 \(C function\),](#) [780](#)  
[vsxseg5ev\\_int64xm1\\_int64x5xm1 \(C function\),](#) [780](#)  
[vsxseg5ev\\_int8xm1\\_int8x5xm1 \(C function\),](#) [781](#)  
[vsxseg5ev\\_mask\\_float16xm1\\_float16x5xm1 \(C func-](#)  
[tion\),](#) [781](#)  
[vsxseg5ev\\_mask\\_float32xm1\\_float32x5xm1 \(C func-](#)  
[tion\),](#) [781](#)  
[vsxseg5ev\\_mask\\_float64xm1\\_float64x5xm1 \(C func-](#)  
[tion\),](#) [781](#)  
[vsxseg5ev\\_mask\\_int16xm1\\_int16x5xm1 \(C function\),](#)  
[781](#)  
[vsxseg5ev\\_mask\\_int32xm1\\_int32x5xm1 \(C function\),](#)  
[781](#)  
[vsxseg5ev\\_mask\\_int64xm1\\_int64x5xm1 \(C function\),](#)  
[781](#)  
[vsxseg5ev\\_mask\\_int8xm1\\_int8x5xm1 \(C function\),](#) [781](#)  
[vsxseg5ev\\_mask\\_uint16xm1\\_uint16x5xm1 \(C function\),](#)  
[781](#)  
[vsxseg5ev\\_mask\\_uint32xm1\\_uint32x5xm1 \(C function\),](#)  
[781](#)  
[vsxseg5ev\\_mask\\_uint64xm1\\_uint64x5xm1 \(C function\),](#)  
[781](#)  
[vsxseg5ev\\_mask\\_uint8xm1\\_uint8x5xm1 \(C function\),](#)  
[781](#)  
[vsxseg5ev\\_uint16xm1\\_uint16x5xm1 \(C function\),](#) [781](#)  
[vsxseg5ev\\_uint32xm1\\_uint32x5xm1 \(C function\),](#) [781](#)  
[vsxseg5ev\\_uint64xm1\\_uint64x5xm1 \(C function\),](#) [781](#)  
[vsxseg5ev\\_uint8xm1\\_uint8x5xm1 \(C function\),](#) [781](#)  
[vsxseg5hv\\_int16xm1\\_int16x5xm1 \(C function\),](#) [782](#)  
[vsxseg5hv\\_int32xm1\\_int32x5xm1 \(C function\),](#) [782](#)  
[vsxseg5hv\\_int64xm1\\_int64x5xm1 \(C function\),](#) [782](#)  
[vsxseg5hv\\_int8xm1\\_int8x5xm1 \(C function\),](#) [782](#)  
[vsxseg5hv\\_mask\\_int16xm1\\_int16x5xm1 \(C function\),](#)  
[782](#)  
[vsxseg5hv\\_mask\\_int32xm1\\_int32x5xm1 \(C function\),](#)  
[782](#)  
[vsxseg5hv\\_mask\\_int64xm1\\_int64x5xm1 \(C function\),](#)  
[782](#)  
[vsxseg5hv\\_mask\\_int8xm1\\_int8x5xm1 \(C function\),](#) [782](#)  
[vsxseg5hv\\_mask\\_uint16xm1\\_uint16x5xm1 \(C function\),](#)  
[782](#)  
[vsxseg5hv\\_mask\\_uint32xm1\\_uint32x5xm1 \(C function\),](#)  
[783](#)  
[vsxseg5hv\\_mask\\_uint64xm1\\_uint64x5xm1 \(C function\),](#)  
[783](#)  
[vsxseg5hv\\_mask\\_uint8xm1\\_uint8x5xm1 \(C function\),](#)  
[783](#)  
[vsxseg5hv\\_uint16xm1\\_uint16x5xm1 \(C function\),](#) [782](#)  
[vsxseg5hv\\_uint32xm1\\_uint32x5xm1 \(C function\),](#) [782](#)  
[vsxseg5hv\\_uint64xm1\\_uint64x5xm1 \(C function\),](#) [782](#)  
[vsxseg5hv\\_uint8xm1\\_uint8x5xm1 \(C function\),](#) [782](#)  
[vsxseg5wv\\_int16xm1\\_int16x5xm1 \(C function\),](#) [783](#)  
[vsxseg5wv\\_int32xm1\\_int32x5xm1 \(C function\),](#) [783](#)  
[vsxseg5wv\\_int64xm1\\_int64x5xm1 \(C function\),](#) [783](#)  
[vsxseg5wv\\_int8xm1\\_int8x5xm1 \(C function\),](#) [783](#)  
[vsxseg5wv\\_mask\\_int16xm1\\_int16x5xm1 \(C function\),](#)  
[783](#)  
[vsxseg5wv\\_mask\\_int32xm1\\_int32x5xm1 \(C function\),](#)  
[784](#)  
[vsxseg5wv\\_mask\\_int64xm1\\_int64x5xm1 \(C function\),](#)  
[784](#)  
[vsxseg5wv\\_mask\\_int8xm1\\_int8x5xm1 \(C function\),](#) [784](#)  
[vsxseg5wv\\_mask\\_uint16xm1\\_uint16x5xm1 \(C func-](#)  
[tion\),](#) [784](#)  
[vsxseg5wv\\_mask\\_uint32xm1\\_uint32x5xm1 \(C func-](#)  
[tion\),](#) [784](#)  
[vsxseg5wv\\_mask\\_uint64xm1\\_uint64x5xm1 \(C func-](#)  
[tion\),](#) [784](#)  
[vsxseg5wv\\_mask\\_uint8xm1\\_uint8x5xm1 \(C function\),](#)  
[784](#)  
[vsxseg5wv\\_uint16xm1\\_uint16x5xm1 \(C function\),](#) [783](#)  
[vsxseg5wv\\_uint32xm1\\_uint32x5xm1 \(C function\),](#) [783](#)  
[vsxseg5wv\\_uint64xm1\\_uint64x5xm1 \(C function\),](#) [783](#)  
[vsxseg5wv\\_uint8xm1\\_uint8x5xm1 \(C function\),](#) [783](#)  
[vsxseg6bv\\_int16xm1\\_int16x6xm1 \(C function\),](#) [784](#)  
[vsxseg6bv\\_int32xm1\\_int32x6xm1 \(C function\),](#) [784](#)  
[vsxseg6bv\\_int64xm1\\_int64x6xm1 \(C function\),](#) [784](#)  
[vsxseg6bv\\_int8xm1\\_int8x6xm1 \(C function\),](#) [784](#)  
[vsxseg6bv\\_mask\\_int16xm1\\_int16x6xm1 \(C function\),](#)  
[785](#)  
[vsxseg6bv\\_mask\\_int32xm1\\_int32x6xm1 \(C function\),](#)  
[785](#)  
[vsxseg6bv\\_mask\\_int64xm1\\_int64x6xm1 \(C function\),](#)  
[785](#)  
[vsxseg6bv\\_mask\\_int8xm1\\_int8x6xm1 \(C function\),](#) [785](#)  
[vsxseg6bv\\_mask\\_uint16xm1\\_uint16x6xm1 \(C function\),](#)  
[785](#)  
[vsxseg6bv\\_mask\\_uint32xm1\\_uint32x6xm1 \(C function\),](#)  
[785](#)  
[vsxseg6bv\\_mask\\_uint64xm1\\_uint64x6xm1 \(C function\),](#)  
[785](#)  
[vsxseg6bv\\_mask\\_uint8xm1\\_uint8x6xm1 \(C function\),](#)  
[785](#)  
[vsxseg6bv\\_uint16xm1\\_uint16x6xm1 \(C function\),](#) [784](#)  
[vsxseg6bv\\_uint32xm1\\_uint32x6xm1 \(C function\),](#) [784](#)  
[vsxseg6bv\\_uint64xm1\\_uint64x6xm1 \(C function\),](#) [784](#)  
[vsxseg6bv\\_uint8xm1\\_uint8x6xm1 \(C function\),](#) [784](#)



vsxseg6ev\_float16xm1\_float16x6xm1 (C function), 785  
 vsxseg6ev\_float32xm1\_float32x6xm1 (C function), 785  
 vsxseg6ev\_float64xm1\_float64x6xm1 (C function), 785  
 vsxseg6ev\_int16xm1\_int16x6xm1 (C function), 785  
 vsxseg6ev\_int32xm1\_int32x6xm1 (C function), 786  
 vsxseg6ev\_int64xm1\_int64x6xm1 (C function), 786  
 vsxseg6ev\_int8xm1\_int8x6xm1 (C function), 786  
 vsxseg6ev\_mask\_float16xm1\_float16x6xm1 (C function), 786  
 vsxseg6ev\_mask\_float32xm1\_float32x6xm1 (C function), 786  
 vsxseg6ev\_mask\_float64xm1\_float64x6xm1 (C function), 786  
 vsxseg6ev\_mask\_int16xm1\_int16x6xm1 (C function), 786  
 vsxseg6ev\_mask\_int32xm1\_int32x6xm1 (C function), 786  
 vsxseg6ev\_mask\_int64xm1\_int64x6xm1 (C function), 786  
 vsxseg6ev\_mask\_int8xm1\_int8x6xm1 (C function), 786  
 vsxseg6ev\_mask\_uint16xm1\_uint16x6xm1 (C function), 786  
 vsxseg6ev\_mask\_uint32xm1\_uint32x6xm1 (C function), 786  
 vsxseg6ev\_mask\_uint64xm1\_uint64x6xm1 (C function), 786  
 vsxseg6ev\_mask\_uint8xm1\_uint8x6xm1 (C function), 787  
 vsxseg6ev\_uint16xm1\_uint16x6xm1 (C function), 786  
 vsxseg6ev\_uint32xm1\_uint32x6xm1 (C function), 786  
 vsxseg6ev\_uint64xm1\_uint64x6xm1 (C function), 786  
 vsxseg6ev\_uint8xm1\_uint8x6xm1 (C function), 786  
 vsxseg6hv\_int16xm1\_int16x6xm1 (C function), 787  
 vsxseg6hv\_int32xm1\_int32x6xm1 (C function), 787  
 vsxseg6hv\_int64xm1\_int64x6xm1 (C function), 787  
 vsxseg6hv\_int8xm1\_int8x6xm1 (C function), 787  
 vsxseg6hv\_mask\_int16xm1\_int16x6xm1 (C function), 787  
 vsxseg6hv\_mask\_int32xm1\_int32x6xm1 (C function), 787  
 vsxseg6hv\_mask\_int64xm1\_int64x6xm1 (C function), 787  
 vsxseg6hv\_mask\_int8xm1\_int8x6xm1 (C function), 787  
 vsxseg6hv\_mask\_uint16xm1\_uint16x6xm1 (C function), 788  
 vsxseg6hv\_mask\_uint32xm1\_uint32x6xm1 (C function), 788  
 vsxseg6hv\_mask\_uint64xm1\_uint64x6xm1 (C function), 788  
 vsxseg6hv\_mask\_uint8xm1\_uint8x6xm1 (C function), 788  
 vsxseg6hv\_uint16xm1\_uint16x6xm1 (C function), 787  
 vsxseg6hv\_uint32xm1\_uint32x6xm1 (C function), 787  
 vsxseg6hv\_uint64xm1\_uint64x6xm1 (C function), 787  
 vsxseg6hv\_uint8xm1\_uint8x6xm1 (C function), 787  
 vsxseg6wv\_int16xm1\_int16x6xm1 (C function), 788  
 vsxseg6wv\_int32xm1\_int32x6xm1 (C function), 788  
 vsxseg6wv\_int64xm1\_int64x6xm1 (C function), 788  
 vsxseg6wv\_int8xm1\_int8x6xm1 (C function), 788  
 vsxseg6wv\_mask\_int16xm1\_int16x6xm1 (C function), 789  
 vsxseg6wv\_mask\_int32xm1\_int32x6xm1 (C function), 789  
 vsxseg6wv\_mask\_int64xm1\_int64x6xm1 (C function), 789  
 vsxseg6wv\_mask\_int8xm1\_int8x6xm1 (C function), 789  
 vsxseg6wv\_mask\_uint16xm1\_uint16x6xm1 (C function), 789  
 vsxseg6wv\_mask\_uint32xm1\_uint32x6xm1 (C function), 789  
 vsxseg6wv\_mask\_uint64xm1\_uint64x6xm1 (C function), 789  
 vsxseg6wv\_mask\_uint8xm1\_uint8x6xm1 (C function), 789  
 vsxseg6wv\_uint16xm1\_uint16x6xm1 (C function), 788  
 vsxseg6wv\_uint32xm1\_uint32x6xm1 (C function), 788  
 vsxseg6wv\_uint64xm1\_uint64x6xm1 (C function), 788  
 vsxseg6wv\_uint8xm1\_uint8x6xm1 (C function), 788  
 vsxseg7bv\_int16xm1\_int16x7xm1 (C function), 789  
 vsxseg7bv\_int32xm1\_int32x7xm1 (C function), 789  
 vsxseg7bv\_int64xm1\_int64x7xm1 (C function), 789  
 vsxseg7bv\_int8xm1\_int8x7xm1 (C function), 789  
 vsxseg7bv\_mask\_int16xm1\_int16x7xm1 (C function), 790  
 vsxseg7bv\_mask\_int32xm1\_int32x7xm1 (C function), 790  
 vsxseg7bv\_mask\_int64xm1\_int64x7xm1 (C function), 790  
 vsxseg7bv\_mask\_int8xm1\_int8x7xm1 (C function), 790  
 vsxseg7bv\_mask\_uint16xm1\_uint16x7xm1 (C function), 790  
 vsxseg7bv\_mask\_uint32xm1\_uint32x7xm1 (C function), 790  
 vsxseg7bv\_mask\_uint64xm1\_uint64x7xm1 (C function), 790  
 vsxseg7bv\_mask\_uint8xm1\_uint8x7xm1 (C function), 790  
 vsxseg7bv\_uint16xm1\_uint16x7xm1 (C function), 789  
 vsxseg7bv\_uint32xm1\_uint32x7xm1 (C function), 789  
 vsxseg7bv\_uint64xm1\_uint64x7xm1 (C function), 789  
 vsxseg7bv\_uint8xm1\_uint8x7xm1 (C function), 790  
 vsxseg7ev\_float16xm1\_float16x7xm1 (C function), 790  
 vsxseg7ev\_float32xm1\_float32x7xm1 (C function), 790  
 vsxseg7ev\_float64xm1\_float64x7xm1 (C function), 790  
 vsxseg7ev\_int16xm1\_int16x7xm1 (C function), 791  
 vsxseg7ev\_int32xm1\_int32x7xm1 (C function), 791  
 vsxseg7ev\_int64xm1\_int64x7xm1 (C function), 791  
 vsxseg7ev\_int8xm1\_int8x7xm1 (C function), 791



vsxseg7ev\_mask\_float16xm1\_float16x7xm1 (C function), 791  
 vsxseg7ev\_mask\_float32xm1\_float32x7xm1 (C function), 791  
 vsxseg7ev\_mask\_float64xm1\_float64x7xm1 (C function), 791  
 vsxseg7ev\_mask\_int16xm1\_int16x7xm1 (C function), 791  
 vsxseg7ev\_mask\_int32xm1\_int32x7xm1 (C function), 791  
 vsxseg7ev\_mask\_int64xm1\_int64x7xm1 (C function), 791  
 vsxseg7ev\_mask\_int8xm1\_int8x7xm1 (C function), 791  
 vsxseg7ev\_mask\_uint16xm1\_uint16x7xm1 (C function), 791  
 vsxseg7ev\_mask\_uint32xm1\_uint32x7xm1 (C function), 791  
 vsxseg7ev\_mask\_uint64xm1\_uint64x7xm1 (C function), 792  
 vsxseg7ev\_mask\_uint8xm1\_uint8x7xm1 (C function), 792  
 vsxseg7ev\_uint16xm1\_uint16x7xm1 (C function), 791  
 vsxseg7ev\_uint32xm1\_uint32x7xm1 (C function), 791  
 vsxseg7ev\_uint64xm1\_uint64x7xm1 (C function), 791  
 vsxseg7ev\_uint8xm1\_uint8x7xm1 (C function), 791  
 vsxseg7hv\_int16xm1\_int16x7xm1 (C function), 792  
 vsxseg7hv\_int32xm1\_int32x7xm1 (C function), 792  
 vsxseg7hv\_int64xm1\_int64x7xm1 (C function), 792  
 vsxseg7hv\_int8xm1\_int8x7xm1 (C function), 792  
 vsxseg7hv\_mask\_int16xm1\_int16x7xm1 (C function), 792  
 vsxseg7hv\_mask\_int32xm1\_int32x7xm1 (C function), 792  
 vsxseg7hv\_mask\_int64xm1\_int64x7xm1 (C function), 793  
 vsxseg7hv\_mask\_int8xm1\_int8x7xm1 (C function), 793  
 vsxseg7hv\_mask\_uint16xm1\_uint16x7xm1 (C function), 793  
 vsxseg7hv\_mask\_uint32xm1\_uint32x7xm1 (C function), 793  
 vsxseg7hv\_mask\_uint64xm1\_uint64x7xm1 (C function), 793  
 vsxseg7hv\_mask\_uint8xm1\_uint8x7xm1 (C function), 793  
 vsxseg7hv\_uint16xm1\_uint16x7xm1 (C function), 792  
 vsxseg7hv\_uint32xm1\_uint32x7xm1 (C function), 792  
 vsxseg7hv\_uint64xm1\_uint64x7xm1 (C function), 792  
 vsxseg7hv\_uint8xm1\_uint8x7xm1 (C function), 792  
 vsxseg7wv\_int16xm1\_int16x7xm1 (C function), 793  
 vsxseg7wv\_int32xm1\_int32x7xm1 (C function), 793  
 vsxseg7wv\_int64xm1\_int64x7xm1 (C function), 793  
 vsxseg7wv\_int8xm1\_int8x7xm1 (C function), 793  
 vsxseg7wv\_mask\_int16xm1\_int16x7xm1 (C function), 794  
 vsxseg7wv\_mask\_int32xm1\_int32x7xm1 (C function), 794  
 vsxseg7wv\_mask\_int64xm1\_int64x7xm1 (C function), 794  
 vsxseg7wv\_mask\_int8xm1\_int8x7xm1 (C function), 794  
 vsxseg7wv\_mask\_uint16xm1\_uint16x7xm1 (C function), 794  
 vsxseg7wv\_mask\_uint32xm1\_uint32x7xm1 (C function), 794  
 vsxseg7wv\_mask\_uint64xm1\_uint64x7xm1 (C function), 794  
 vsxseg7wv\_mask\_uint8xm1\_uint8x7xm1 (C function), 794  
 vsxseg7wv\_uint16xm1\_uint16x7xm1 (C function), 793  
 vsxseg7wv\_uint32xm1\_uint32x7xm1 (C function), 793  
 vsxseg7wv\_uint64xm1\_uint64x7xm1 (C function), 793  
 vsxseg7wv\_uint8xm1\_uint8x7xm1 (C function), 793  
 vsxseg8bv\_int16xm1\_int16x8xm1 (C function), 794  
 vsxseg8bv\_int32xm1\_int32x8xm1 (C function), 794  
 vsxseg8bv\_int64xm1\_int64x8xm1 (C function), 794  
 vsxseg8bv\_int8xm1\_int8x8xm1 (C function), 794  
 vsxseg8bv\_mask\_int16xm1\_int16x8xm1 (C function), 795  
 vsxseg8bv\_mask\_int32xm1\_int32x8xm1 (C function), 795  
 vsxseg8bv\_mask\_int64xm1\_int64x8xm1 (C function), 795  
 vsxseg8bv\_mask\_int8xm1\_int8x8xm1 (C function), 795  
 vsxseg8bv\_mask\_uint16xm1\_uint16x8xm1 (C function), 795  
 vsxseg8bv\_mask\_uint32xm1\_uint32x8xm1 (C function), 795  
 vsxseg8bv\_mask\_uint64xm1\_uint64x8xm1 (C function), 795  
 vsxseg8bv\_mask\_uint8xm1\_uint8x8xm1 (C function), 795  
 vsxseg8bv\_uint16xm1\_uint16x8xm1 (C function), 795  
 vsxseg8bv\_uint32xm1\_uint32x8xm1 (C function), 795  
 vsxseg8bv\_uint64xm1\_uint64x8xm1 (C function), 795  
 vsxseg8bv\_uint8xm1\_uint8x8xm1 (C function), 795  
 vsxseg8ev\_float16xm1\_float16x8xm1 (C function), 796  
 vsxseg8ev\_float32xm1\_float32x8xm1 (C function), 796  
 vsxseg8ev\_float64xm1\_float64x8xm1 (C function), 796  
 vsxseg8ev\_int16xm1\_int16x8xm1 (C function), 796  
 vsxseg8ev\_int32xm1\_int32x8xm1 (C function), 796  
 vsxseg8ev\_int64xm1\_int64x8xm1 (C function), 796  
 vsxseg8ev\_int8xm1\_int8x8xm1 (C function), 796  
 vsxseg8ev\_mask\_float16xm1\_float16x8xm1 (C function), 796  
 vsxseg8ev\_mask\_float32xm1\_float32x8xm1 (C function), 796  
 vsxseg8ev\_mask\_float64xm1\_float64x8xm1 (C function), 796  
 vsxseg8ev\_mask\_int16xm1\_int16x8xm1 (C function), 796

- 796  
vsxseg8ev\_mask\_int32xm1\_int32x8xm1 (C function),  
796  
vsxseg8ev\_mask\_int64xm1\_int64x8xm1 (C function),  
796  
vsxseg8ev\_mask\_int8xm1\_int8x8xm1 (C function), 796  
vsxseg8ev\_mask\_uint16xm1\_uint16x8xm1 (C function),  
797  
vsxseg8ev\_mask\_uint32xm1\_uint32x8xm1 (C function),  
797  
vsxseg8ev\_mask\_uint64xm1\_uint64x8xm1 (C function),  
797  
vsxseg8ev\_mask\_uint8xm1\_uint8x8xm1 (C function),  
797  
vsxseg8ev\_uint16xm1\_uint16x8xm1 (C function), 796  
vsxseg8ev\_uint32xm1\_uint32x8xm1 (C function), 796  
vsxseg8ev\_uint64xm1\_uint64x8xm1 (C function), 796  
vsxseg8ev\_uint8xm1\_uint8x8xm1 (C function), 796  
vsxseg8hv\_int16xm1\_int16x8xm1 (C function), 797  
vsxseg8hv\_int32xm1\_int32x8xm1 (C function), 797  
vsxseg8hv\_int64xm1\_int64x8xm1 (C function), 797  
vsxseg8hv\_int8xm1\_int8x8xm1 (C function), 797  
vsxseg8hv\_mask\_int16xm1\_int16x8xm1 (C function),  
798  
vsxseg8hv\_mask\_int32xm1\_int32x8xm1 (C function),  
798  
vsxseg8hv\_mask\_int64xm1\_int64x8xm1 (C function),  
798  
vsxseg8hv\_mask\_int8xm1\_int8x8xm1 (C function), 798  
vsxseg8hv\_mask\_uint16xm1\_uint16x8xm1 (C function),  
798  
vsxseg8hv\_mask\_uint32xm1\_uint32x8xm1 (C function),  
798  
vsxseg8hv\_mask\_uint64xm1\_uint64x8xm1 (C function),  
798  
vsxseg8hv\_mask\_uint8xm1\_uint8x8xm1 (C function),  
798  
vsxseg8hv\_uint16xm1\_uint16x8xm1 (C function), 797  
vsxseg8hv\_uint32xm1\_uint32x8xm1 (C function), 797  
vsxseg8hv\_uint64xm1\_uint64x8xm1 (C function), 797  
vsxseg8hv\_uint8xm1\_uint8x8xm1 (C function), 797  
vsxseg8wv\_int16xm1\_int16x8xm1 (C function), 798  
vsxseg8wv\_int32xm1\_int32x8xm1 (C function), 798  
vsxseg8wv\_int64xm1\_int64x8xm1 (C function), 798  
vsxseg8wv\_int8xm1\_int8x8xm1 (C function), 798  
vsxseg8wv\_mask\_int16xm1\_int16x8xm1 (C function),  
799  
vsxseg8wv\_mask\_int32xm1\_int32x8xm1 (C function),  
799  
vsxseg8wv\_mask\_int64xm1\_int64x8xm1 (C function),  
799  
vsxseg8wv\_mask\_int8xm1\_int8x8xm1 (C function), 799  
vsxseg8wv\_mask\_uint16xm1\_uint16x8xm1 (C func-  
tion), 799  
vsxseg8wv\_mask\_uint32xm1\_uint32x8xm1 (C func-  
tion), 799  
vsxseg8wv\_mask\_uint64xm1\_uint64x8xm1 (C func-  
tion), 799  
vsxseg8wv\_mask\_uint8xm1\_uint8x8xm1 (C function),  
799  
vsxseg8wv\_uint16xm1\_uint16x8xm1 (C function), 798  
vsxseg8wv\_uint32xm1\_uint32x8xm1 (C function), 798  
vsxseg8wv\_uint64xm1\_uint64x8xm1 (C function), 798  
vsxseg8wv\_uint8xm1\_uint8x8xm1 (C function), 799  
vsxwv\_int16xm1 (C function), 492  
vsxwv\_int16xm2 (C function), 492  
vsxwv\_int16xm4 (C function), 492  
vsxwv\_int16xm8 (C function), 492  
vsxwv\_int32xm1 (C function), 492  
vsxwv\_int32xm2 (C function), 492  
vsxwv\_int32xm4 (C function), 492  
vsxwv\_int32xm8 (C function), 492  
vsxwv\_int64xm1 (C function), 492  
vsxwv\_int64xm2 (C function), 492  
vsxwv\_int64xm4 (C function), 493  
vsxwv\_int64xm8 (C function), 493  
vsxwv\_int8xm1 (C function), 493  
vsxwv\_int8xm2 (C function), 493  
vsxwv\_int8xm4 (C function), 493  
vsxwv\_int8xm8 (C function), 493  
vsxwv\_mask\_int16xm1 (C function), 493  
vsxwv\_mask\_int16xm2 (C function), 493  
vsxwv\_mask\_int16xm4 (C function), 493  
vsxwv\_mask\_int16xm8 (C function), 493  
vsxwv\_mask\_int32xm1 (C function), 493  
vsxwv\_mask\_int32xm2 (C function), 494  
vsxwv\_mask\_int32xm4 (C function), 494  
vsxwv\_mask\_int32xm8 (C function), 494  
vsxwv\_mask\_int64xm1 (C function), 494  
vsxwv\_mask\_int64xm2 (C function), 494  
vsxwv\_mask\_int64xm4 (C function), 494  
vsxwv\_mask\_int64xm8 (C function), 494  
vsxwv\_mask\_int8xm1 (C function), 494  
vsxwv\_mask\_int8xm2 (C function), 494  
vsxwv\_mask\_int8xm4 (C function), 494  
vsxwv\_mask\_int8xm8 (C function), 494  
vsxwv\_mask\_uint16xm1 (C function), 494  
vsxwv\_mask\_uint16xm2 (C function), 494  
vsxwv\_mask\_uint16xm4 (C function), 494  
vsxwv\_mask\_uint16xm8 (C function), 494  
vsxwv\_mask\_uint32xm1 (C function), 494  
vsxwv\_mask\_uint32xm2 (C function), 494  
vsxwv\_mask\_uint32xm4 (C function), 494  
vsxwv\_mask\_uint32xm8 (C function), 494  
vsxwv\_mask\_uint64xm1 (C function), 494  
vsxwv\_mask\_uint64xm2 (C function), 494  
vsxwv\_mask\_uint64xm4 (C function), 494  
vsxwv\_mask\_uint64xm8 (C function), 494

vsxwv\_mask\_uint8xm1 (C function), 495  
 vsxwv\_mask\_uint8xm2 (C function), 495  
 vsxwv\_mask\_uint8xm4 (C function), 495  
 vsxwv\_mask\_uint8xm8 (C function), 495  
 vsxwv\_uint16xm1 (C function), 493  
 vsxwv\_uint16xm2 (C function), 493  
 vsxwv\_uint16xm4 (C function), 493  
 vsxwv\_uint16xm8 (C function), 493  
 vsxwv\_uint32xm1 (C function), 493  
 vsxwv\_uint32xm2 (C function), 493  
 vsxwv\_uint32xm4 (C function), 493  
 vsxwv\_uint32xm8 (C function), 493  
 vsxwv\_uint64xm1 (C function), 493  
 vsxwv\_uint64xm2 (C function), 493  
 vsxwv\_uint64xm4 (C function), 493  
 vsxwv\_uint64xm8 (C function), 493  
 vsxwv\_uint8xm1 (C function), 493  
 vsxwv\_uint8xm2 (C function), 493  
 vsxwv\_uint8xm4 (C function), 493  
 vsxwv\_uint8xm8 (C function), 493  
 vwadduvv\_mask\_uint16xm2\_uint8xm1 (C function), 326  
 vwadduvv\_mask\_uint16xm4\_uint8xm2 (C function), 326  
 vwadduvv\_mask\_uint16xm8\_uint8xm4 (C function), 326  
 vwadduvv\_mask\_uint32xm2\_uint16xm1 (C function), 326  
 vwadduvv\_mask\_uint32xm4\_uint16xm2 (C function), 326  
 vwadduvv\_mask\_uint32xm8\_uint16xm4 (C function), 326  
 vwadduvv\_mask\_uint64xm2\_uint32xm1 (C function), 326  
 vwadduvv\_mask\_uint64xm4\_uint32xm2 (C function), 326  
 vwadduvv\_mask\_uint64xm8\_uint32xm4 (C function), 327  
 vwadduvv\_uint16xm2\_uint8xm1 (C function), 326  
 vwadduvv\_uint16xm4\_uint8xm2 (C function), 326  
 vwadduvv\_uint16xm8\_uint8xm4 (C function), 326  
 vwadduvv\_uint32xm2\_uint16xm1 (C function), 326  
 vwadduvv\_uint32xm4\_uint16xm2 (C function), 326  
 vwadduvv\_uint32xm8\_uint16xm4 (C function), 326  
 vwadduvv\_uint64xm2\_uint32xm1 (C function), 326  
 vwadduvv\_uint64xm4\_uint32xm2 (C function), 326  
 vwadduvv\_uint64xm8\_uint32xm4 (C function), 326  
 vwadduvx\_mask\_uint16xm2\_uint8xm1 (C function), 327  
 vwadduvx\_mask\_uint16xm4\_uint8xm2 (C function), 327  
 vwadduvx\_mask\_uint16xm8\_uint8xm4 (C function), 327  
 vwadduvx\_mask\_uint32xm2\_uint16xm1 (C function), 327  
 vwadduvx\_mask\_uint32xm4\_uint16xm2 (C function), 328  
 vwadduvx\_mask\_uint32xm8\_uint16xm4 (C function), 328  
 vwadduvx\_mask\_uint64xm2\_uint32xm1 (C function), 328  
 vwadduvx\_mask\_uint64xm4\_uint32xm2 (C function), 328  
 vwadduvx\_mask\_uint64xm8\_uint32xm4 (C function), 328  
 vwadduvx\_mask\_uint16xm2\_uint8xm1 (C function), 327  
 vwadduvx\_mask\_uint16xm4\_uint8xm2 (C function), 327  
 vwadduvx\_mask\_uint16xm8\_uint8xm4 (C function), 327  
 vwadduvx\_mask\_uint32xm2\_uint16xm1 (C function), 327  
 vwadduvx\_mask\_uint32xm4\_uint16xm2 (C function), 327  
 vwadduvx\_mask\_uint32xm8\_uint16xm4 (C function), 327  
 vwadduvx\_mask\_uint64xm2\_uint32xm1 (C function), 327  
 vwadduvx\_mask\_uint64xm4\_uint32xm2 (C function), 327  
 vwadduvx\_mask\_uint64xm8\_uint32xm4 (C function), 327  
 vwadduwv\_mask\_uint16xm2\_uint8xm1 (C function), 329  
 vwadduwv\_mask\_uint16xm4\_uint8xm2 (C function), 329  
 vwadduwv\_mask\_uint16xm8\_uint8xm4 (C function), 329  
 vwadduwv\_mask\_uint32xm2\_uint16xm1 (C function), 329  
 vwadduwv\_mask\_uint32xm4\_uint16xm2 (C function), 329  
 vwadduwv\_mask\_uint32xm8\_uint16xm4 (C function), 329  
 vwadduwv\_mask\_uint64xm2\_uint32xm1 (C function), 329  
 vwadduwv\_mask\_uint64xm4\_uint32xm2 (C function), 329  
 vwadduwv\_mask\_uint64xm8\_uint32xm4 (C function), 329  
 vwadduwv\_uint16xm2\_uint8xm1 (C function), 328  
 vwadduwv\_uint16xm4\_uint8xm2 (C function), 328  
 vwadduwv\_uint16xm8\_uint8xm4 (C function), 328  
 vwadduwv\_uint32xm2\_uint16xm1 (C function), 328  
 vwadduwv\_uint32xm4\_uint16xm2 (C function), 328  
 vwadduwv\_uint32xm8\_uint16xm4 (C function), 328  
 vwadduwv\_uint64xm2\_uint32xm1 (C function), 328  
 vwadduwv\_uint64xm4\_uint32xm2 (C function), 328  
 vwadduwv\_uint64xm8\_uint32xm4 (C function), 328  
 vwadduwx\_mask\_uint16xm2 (C function), 330  
 vwadduwx\_mask\_uint16xm4 (C function), 330  
 vwadduwx\_mask\_uint16xm8 (C function), 330  
 vwadduwx\_mask\_uint32xm2 (C function), 330  
 vwadduwx\_mask\_uint32xm4 (C function), 330  
 vwadduwx\_mask\_uint32xm8 (C function), 330  
 vwadduwx\_mask\_uint64xm2 (C function), 330  
 vwadduwx\_mask\_uint64xm4 (C function), 330  
 vwadduwx\_mask\_uint64xm8 (C function), 330  
 vwadduwx\_uint16xm2 (C function), 329  
 vwadduwx\_uint16xm4 (C function), 329  
 vwadduwx\_uint16xm8 (C function), 329

- vwadduw\_uint32xm2 (C function), 329  
 vwadduw\_uint32xm4 (C function), 329  
 vwadduw\_uint32xm8 (C function), 329  
 vwadduw\_uint64xm2 (C function), 330  
 vwadduw\_uint64xm4 (C function), 330  
 vwadduw\_uint64xm8 (C function), 330  
 vwaddvv\_int16xm2\_int8xm1 (C function), 321  
 vwaddvv\_int16xm4\_int8xm2 (C function), 322  
 vwaddvv\_int16xm8\_int8xm4 (C function), 322  
 vwaddvv\_int32xm2\_int16xm1 (C function), 322  
 vwaddvv\_int32xm4\_int16xm2 (C function), 322  
 vwaddvv\_int32xm8\_int16xm4 (C function), 322  
 vwaddvv\_int64xm2\_int32xm1 (C function), 322  
 vwaddvv\_int64xm4\_int32xm2 (C function), 322  
 vwaddvv\_int64xm8\_int32xm4 (C function), 322  
 vwaddvv\_mask\_int16xm2\_int8xm1 (C function), 322  
 vwaddvv\_mask\_int16xm4\_int8xm2 (C function), 322  
 vwaddvv\_mask\_int16xm8\_int8xm4 (C function), 322  
 vwaddvv\_mask\_int32xm2\_int16xm1 (C function), 322  
 vwaddvv\_mask\_int32xm4\_int16xm2 (C function), 322  
 vwaddvv\_mask\_int32xm8\_int16xm4 (C function), 322  
 vwaddvv\_mask\_int64xm2\_int32xm1 (C function), 322  
 vwaddvv\_mask\_int64xm4\_int32xm2 (C function), 322  
 vwaddvv\_mask\_int64xm8\_int32xm4 (C function), 322  
 vwaddvx\_int16xm2\_int8xm1 (C function), 323  
 vwaddvx\_int16xm4\_int8xm2 (C function), 323  
 vwaddvx\_int16xm8\_int8xm4 (C function), 323  
 vwaddvx\_int32xm2\_int16xm1 (C function), 323  
 vwaddvx\_int32xm4\_int16xm2 (C function), 323  
 vwaddvx\_int32xm8\_int16xm4 (C function), 323  
 vwaddvx\_int64xm2\_int32xm1 (C function), 323  
 vwaddvx\_int64xm4\_int32xm2 (C function), 323  
 vwaddvx\_int64xm8\_int32xm4 (C function), 323  
 vwaddvx\_mask\_int16xm2\_int8xm1 (C function), 323  
 vwaddvx\_mask\_int16xm4\_int8xm2 (C function), 323  
 vwaddvx\_mask\_int16xm8\_int8xm4 (C function), 323  
 vwaddvx\_mask\_int32xm2\_int16xm1 (C function), 323  
 vwaddvx\_mask\_int32xm4\_int16xm2 (C function), 323  
 vwaddvx\_mask\_int32xm8\_int16xm4 (C function), 323  
 vwaddvx\_mask\_int64xm2\_int32xm1 (C function), 323  
 vwaddvx\_mask\_int64xm4\_int32xm2 (C function), 323  
 vwaddvx\_mask\_int64xm8\_int32xm4 (C function), 323  
 vwaddwv\_int16xm2\_int8xm1 (C function), 324  
 vwaddwv\_int16xm4\_int8xm2 (C function), 324  
 vwaddwv\_int16xm8\_int8xm4 (C function), 324  
 vwaddwv\_int32xm2\_int16xm1 (C function), 324  
 vwaddwv\_int32xm4\_int16xm2 (C function), 324  
 vwaddwv\_int32xm8\_int16xm4 (C function), 324  
 vwaddwv\_int64xm2\_int32xm1 (C function), 324  
 vwaddwv\_int64xm4\_int32xm2 (C function), 324  
 vwaddwv\_int64xm8\_int32xm4 (C function), 324  
 vwaddwv\_mask\_int16xm2\_int8xm1 (C function), 324  
 vwaddwv\_mask\_int16xm4\_int8xm2 (C function), 324  
 vwaddwv\_mask\_int16xm8\_int8xm4 (C function), 324  
 vwaddwv\_mask\_int32xm2\_int16xm1 (C function), 324  
 vwaddwv\_mask\_int32xm4\_int16xm2 (C function), 324  
 vwaddwv\_mask\_int32xm8\_int16xm4 (C function), 324  
 vwaddwv\_mask\_int64xm2\_int32xm1 (C function), 324  
 vwaddwv\_mask\_int64xm4\_int32xm2 (C function), 324  
 vwaddwv\_mask\_int64xm8\_int32xm4 (C function), 324  
 vwaddwx\_int16xm2 (C function), 325  
 vwaddwx\_int16xm4 (C function), 325  
 vwaddwx\_int16xm8 (C function), 325  
 vwaddwx\_int32xm2 (C function), 325  
 vwaddwx\_int32xm4 (C function), 325  
 vwaddwx\_int32xm8 (C function), 325  
 vwaddwx\_int64xm2 (C function), 325  
 vwaddwx\_int64xm4 (C function), 325  
 vwaddwx\_int64xm8 (C function), 325  
 vwaddwx\_mask\_int16xm2 (C function), 325  
 vwaddwx\_mask\_int16xm4 (C function), 325  
 vwaddwx\_mask\_int16xm8 (C function), 325  
 vwaddwx\_mask\_int32xm2 (C function), 325  
 vwaddwx\_mask\_int32xm4 (C function), 325  
 vwaddwx\_mask\_int32xm8 (C function), 325  
 vwaddwx\_mask\_int64xm2 (C function), 325  
 vwaddwx\_mask\_int64xm4 (C function), 325  
 vwaddwx\_mask\_int64xm8 (C function), 325  
 vwmaccsuvv\_int16xm2\_int8xm1\_uint8xm1 (C function), 333  
 vwmaccsuvv\_int16xm4\_int8xm2\_uint8xm2 (C function), 333  
 vwmaccsuvv\_int16xm8\_int8xm4\_uint8xm4 (C function), 333  
 vwmaccsuvv\_int32xm2\_int16xm1\_uint16xm1 (C function), 333  
 vwmaccsuvv\_int32xm4\_int16xm2\_uint16xm2 (C function), 333  
 vwmaccsuvv\_int32xm8\_int16xm4\_uint16xm4 (C function), 333  
 vwmaccsuvv\_int64xm2\_int32xm1\_uint32xm1 (C function), 333  
 vwmaccsuvv\_int64xm4\_int32xm2\_uint32xm2 (C function), 334  
 vwmaccsuvv\_int64xm8\_int32xm4\_uint32xm4 (C function), 334  
 vwmaccsuvv\_mask\_int16xm2\_int8xm1\_uint8xm1 (C function), 334  
 vwmaccsuvv\_mask\_int16xm4\_int8xm2\_uint8xm2 (C function), 334  
 vwmaccsuvv\_mask\_int16xm8\_int8xm4\_uint8xm4 (C function), 334  
 vwmaccsuvv\_mask\_int32xm2\_int16xm1\_uint16xm1 (C function), 334  
 vwmaccsuvv\_mask\_int32xm4\_int16xm2\_uint16xm2 (C function), 334  
 vwmaccsuvv\_mask\_int32xm8\_int16xm4\_uint16xm4 (C function), 334



vwmaccsuvv\_mask\_int64xm2\_int32xm1\_uint32xm1 (C function), 334  
 vwmaccsuvv\_mask\_int64xm4\_int32xm2\_uint32xm2 (C function), 335  
 vwmaccsuvv\_mask\_int64xm8\_int32xm4\_uint32xm4 (C function), 335  
 vwmaccsuvx\_int16xm2\_uint8xm1 (C function), 335  
 vwmaccsuvx\_int16xm4\_uint8xm2 (C function), 335  
 vwmaccsuvx\_int16xm8\_uint8xm4 (C function), 335  
 vwmaccsuvx\_int32xm2\_uint16xm1 (C function), 335  
 vwmaccsuvx\_int32xm4\_uint16xm2 (C function), 335  
 vwmaccsuvx\_int32xm8\_uint16xm4 (C function), 335  
 vwmaccsuvx\_int64xm2\_uint32xm1 (C function), 335  
 vwmaccsuvx\_int64xm4\_uint32xm2 (C function), 335  
 vwmaccsuvx\_int64xm8\_uint32xm4 (C function), 335  
 vwmaccsuvx\_mask\_int16xm2\_uint8xm1 (C function), 336  
 vwmaccsuvx\_mask\_int16xm4\_uint8xm2 (C function), 336  
 vwmaccsuvx\_mask\_int16xm8\_uint8xm4 (C function), 336  
 vwmaccsuvx\_mask\_int32xm2\_uint16xm1 (C function), 336  
 vwmaccsuvx\_mask\_int32xm4\_uint16xm2 (C function), 336  
 vwmaccsuvx\_mask\_int32xm8\_uint16xm4 (C function), 336  
 vwmaccsuvx\_mask\_int64xm2\_uint32xm1 (C function), 336  
 vwmaccsuvx\_mask\_int64xm4\_uint32xm2 (C function), 336  
 vwmaccsuvx\_mask\_int64xm8\_uint32xm4 (C function), 336  
 vwmaccsvx\_int16xm2\_int8xm1 (C function), 339  
 vwmaccsvx\_int16xm4\_int8xm2 (C function), 339  
 vwmaccsvx\_int16xm8\_int8xm4 (C function), 339  
 vwmaccsvx\_int32xm2\_int16xm1 (C function), 339  
 vwmaccsvx\_int32xm4\_int16xm2 (C function), 339  
 vwmaccsvx\_int32xm8\_int16xm4 (C function), 339  
 vwmaccsvx\_int64xm2\_int32xm1 (C function), 339  
 vwmaccsvx\_int64xm4\_int32xm2 (C function), 339  
 vwmaccsvx\_int64xm8\_int32xm4 (C function), 340  
 vwmaccsvx\_mask\_int16xm2\_int8xm1 (C function), 340  
 vwmaccsvx\_mask\_int16xm4\_int8xm2 (C function), 340  
 vwmaccsvx\_mask\_int16xm8\_int8xm4 (C function), 340  
 vwmaccsvx\_mask\_int32xm2\_int16xm1 (C function), 340  
 vwmaccsvx\_mask\_int32xm4\_int16xm2 (C function), 340  
 vwmaccsvx\_mask\_int32xm8\_int16xm4 (C function), 340  
 vwmaccsvx\_mask\_int64xm2\_int32xm1 (C function), 340  
 vwmaccsvx\_mask\_int64xm4\_int32xm2 (C function), 340  
 vwmaccsvx\_mask\_int64xm8\_int32xm4 (C function), 340  
 vwmaccuvv\_mask\_uint16xm2\_uint8xm1 (C function), 337  
 vwmaccuvv\_mask\_uint16xm4\_uint8xm2 (C function), 337  
 vwmaccuvv\_mask\_uint16xm8\_uint8xm4 (C function), 337  
 vwmaccuvv\_mask\_uint32xm2\_uint16xm1 (C function), 337  
 vwmaccuvv\_mask\_uint32xm4\_uint16xm2 (C function), 337  
 vwmaccuvv\_mask\_uint32xm8\_uint16xm4 (C function), 337  
 vwmaccuvv\_mask\_uint64xm2\_uint32xm1 (C function), 337  
 vwmaccuvv\_mask\_uint64xm4\_uint32xm2 (C function), 337  
 vwmaccuvv\_mask\_uint64xm8\_uint32xm4 (C function), 337  
 vwmaccuvv\_uint16xm2\_uint8xm1 (C function), 336  
 vwmaccuvv\_uint16xm4\_uint8xm2 (C function), 336  
 vwmaccuvv\_uint16xm8\_uint8xm4 (C function), 336  
 vwmaccuvv\_uint32xm2\_uint16xm1 (C function), 337  
 vwmaccuvv\_uint32xm4\_uint16xm2 (C function), 337  
 vwmaccuvv\_uint32xm8\_uint16xm4 (C function), 337  
 vwmaccuvv\_uint64xm2\_uint32xm1 (C function), 337  
 vwmaccuvv\_uint64xm4\_uint32xm2 (C function), 337  
 vwmaccuvv\_uint64xm8\_uint32xm4 (C function), 337  
 vwmaccuvx\_mask\_int16xm2\_int8xm1 (C function), 338  
 vwmaccuvx\_mask\_int16xm4\_int8xm2 (C function), 338  
 vwmaccuvx\_mask\_int16xm8\_int8xm4 (C function), 338  
 vwmaccuvx\_mask\_int32xm2\_int16xm1 (C function), 339  
 vwmaccuvx\_mask\_int32xm4\_int16xm2 (C function), 339  
 vwmaccuvx\_mask\_int32xm8\_int16xm4 (C function), 339  
 vwmaccuvx\_mask\_int64xm2\_int32xm1 (C function), 339  
 vwmaccuvx\_mask\_int64xm4\_int32xm2 (C function), 339  
 vwmaccuvx\_mask\_int64xm8\_int32xm4 (C function), 339  
 vwmaccuvx\_uint16xm2\_uint8xm1 (C function), 338  
 vwmaccuvx\_uint16xm4\_uint8xm2 (C function), 338  
 vwmaccuvx\_uint16xm8\_uint8xm4 (C function), 338

- vwmaaccvux\_uint32xm2\_uint16xm1 (C function), 338
- vwmaaccvux\_uint32xm4\_uint16xm2 (C function), 338
- vwmaaccvux\_uint32xm8\_uint16xm4 (C function), 338
- vwmaaccvux\_uint64xm2\_uint32xm1 (C function), 338
- vwmaaccvux\_uint64xm4\_uint32xm2 (C function), 338
- vwmaaccvux\_uint64xm8\_uint32xm4 (C function), 338
- vwmaaccvv\_int16xm2\_int8xm1 (C function), 330
- vwmaaccvv\_int16xm4\_int8xm2 (C function), 330
- vwmaaccvv\_int16xm8\_int8xm4 (C function), 331
- vwmaaccvv\_int32xm2\_int16xm1 (C function), 331
- vwmaaccvv\_int32xm4\_int16xm2 (C function), 331
- vwmaaccvv\_int32xm8\_int16xm4 (C function), 331
- vwmaaccvv\_int64xm2\_int32xm1 (C function), 331
- vwmaaccvv\_int64xm4\_int32xm2 (C function), 331
- vwmaaccvv\_int64xm8\_int32xm4 (C function), 331
- vwmaaccvv\_mask\_int16xm2\_int8xm1 (C function), 331
- vwmaaccvv\_mask\_int16xm4\_int8xm2 (C function), 331
- vwmaaccvv\_mask\_int16xm8\_int8xm4 (C function), 331
- vwmaaccvv\_mask\_int32xm2\_int16xm1 (C function), 331
- vwmaaccvv\_mask\_int32xm4\_int16xm2 (C function), 331
- vwmaaccvv\_mask\_int32xm8\_int16xm4 (C function), 331
- vwmaaccvv\_mask\_int64xm2\_int32xm1 (C function), 331
- vwmaaccvv\_mask\_int64xm4\_int32xm2 (C function), 331
- vwmaaccvv\_mask\_int64xm8\_int32xm4 (C function), 331
- vwmaaccvx\_int16xm2\_int8xm1 (C function), 332
- vwmaaccvx\_int16xm4\_int8xm2 (C function), 332
- vwmaaccvx\_int16xm8\_int8xm4 (C function), 332
- vwmaaccvx\_int32xm2\_int16xm1 (C function), 332
- vwmaaccvx\_int32xm4\_int16xm2 (C function), 332
- vwmaaccvx\_int32xm8\_int16xm4 (C function), 332
- vwmaaccvx\_int64xm2\_int32xm1 (C function), 332
- vwmaaccvx\_int64xm4\_int32xm2 (C function), 332
- vwmaaccvx\_int64xm8\_int32xm4 (C function), 332
- vwmaaccvx\_mask\_int16xm2\_int8xm1 (C function), 332
- vwmaaccvx\_mask\_int16xm4\_int8xm2 (C function), 332
- vwmaaccvx\_mask\_int16xm8\_int8xm4 (C function), 332
- vwmaaccvx\_mask\_int32xm2\_int16xm1 (C function), 332
- vwmaaccvx\_mask\_int32xm4\_int16xm2 (C function), 333
- vwmaaccvx\_mask\_int32xm8\_int16xm4 (C function), 333
- vwmaaccvx\_mask\_int64xm2\_int32xm1 (C function), 333
- vwmaaccvx\_mask\_int64xm4\_int32xm2 (C function), 333
- vwmaaccvx\_mask\_int64xm8\_int32xm4 (C function), 333
- vwmulsubv\_int16xm2\_int8xm1\_uint8xm1 (C function), 343
- vwmulsubv\_int16xm4\_int8xm2\_uint8xm2 (C function), 343
- vwmulsubv\_int16xm8\_int8xm4\_uint8xm4 (C function), 343
- vwmulsubv\_int32xm2\_int16xm1\_uint16xm1 (C function), 343
- vwmulsubv\_int32xm4\_int16xm2\_uint16xm2 (C function), 343
- vwmulsubv\_int32xm8\_int16xm4\_uint16xm4 (C function), 343
- vwmulsubv\_int64xm2\_int32xm1\_uint32xm1 (C function), 343
- vwmulsubv\_int64xm4\_int32xm2\_uint32xm2 (C function), 343
- vwmulsubv\_int64xm8\_int32xm4\_uint32xm4 (C function), 343
- vwmulsubv\_mask\_int16xm2\_int8xm1\_uint8xm1 (C function), 343
- vwmulsubv\_mask\_int16xm4\_int8xm2\_uint8xm2 (C function), 343
- vwmulsubv\_mask\_int16xm8\_int8xm4\_uint8xm4 (C function), 343
- vwmulsubv\_mask\_int32xm2\_int16xm1\_uint16xm1 (C function), 343
- vwmulsubv\_mask\_int32xm4\_int16xm2\_uint16xm2 (C function), 343
- vwmulsubv\_mask\_int32xm8\_int16xm4\_uint16xm4 (C function), 343
- vwmulsubv\_mask\_int64xm2\_int32xm1\_uint32xm1 (C function), 343
- vwmulsubv\_mask\_int64xm4\_int32xm2\_uint32xm2 (C function), 343
- vwmulsubv\_mask\_int64xm8\_int32xm4\_uint32xm4 (C function), 343
- vwmulsubv\_mask\_int16xm2\_int8xm1\_uint8xm1 (C function), 344
- vwmulsubv\_mask\_int16xm4\_int8xm2\_uint8xm2 (C function), 344
- vwmulsubv\_mask\_int16xm8\_int8xm4\_uint8xm4 (C function), 344
- vwmulsubv\_mask\_int32xm2\_int16xm1\_uint16xm1 (C function), 344
- vwmulsubv\_mask\_int32xm4\_int16xm2\_uint16xm2 (C function), 344
- vwmulsubv\_mask\_int32xm8\_int16xm4\_uint16xm4 (C function), 344
- vwmulsubv\_mask\_int64xm2\_int32xm1\_uint32xm1 (C function), 344
- vwmulsubv\_mask\_int64xm4\_int32xm2\_uint32xm2 (C function), 344
- vwmulsubv\_mask\_int64xm8\_int32xm4\_uint32xm4 (C function), 344
- vwmulsubv\_mask\_int16xm2\_int8xm1 (C function), 345
- vwmulsubv\_mask\_int16xm4\_int8xm2 (C function), 345
- vwmulsubv\_mask\_int16xm8\_int8xm4 (C function), 345
- vwmulsubv\_mask\_int32xm2\_int16xm1 (C function), 345
- vwmulsubv\_mask\_int32xm4\_int16xm2 (C function), 345
- vwmulsubv\_mask\_int32xm8\_int16xm4 (C function), 345
- vwmulsubv\_mask\_int64xm2\_int32xm1 (C function), 345
- vwmulsubv\_mask\_int64xm4\_int32xm2 (C function), 345
- vwmulsubv\_mask\_int64xm8\_int32xm4 (C function), 345
- vwmulsubv\_mask\_uint16xm2\_uint8xm1 (C function), 346
- vwmulsubv\_mask\_uint16xm4\_uint8xm2 (C function), 346
- vwmulsubv\_mask\_uint16xm8\_uint8xm4 (C function), 346
- vwmulsubv\_mask\_uint32xm2\_uint16xm1 (C function), 346
- vwmulsubv\_mask\_uint32xm4\_uint16xm2 (C function), 346
- vwmulsubv\_mask\_uint32xm8\_uint16xm4 (C function), 346



- vwmuluvv\_mask\_uint64xm2\_uint32xm1 (C function), 346
- vwmuluvv\_mask\_uint64xm4\_uint32xm2 (C function), 346
- vwmuluvv\_mask\_uint64xm8\_uint32xm4 (C function), 346
- vwmuluvv\_uint16xm2\_uint8xm1 (C function), 345
- vwmuluvv\_uint16xm4\_uint8xm2 (C function), 345
- vwmuluvv\_uint16xm8\_uint8xm4 (C function), 345
- vwmuluvv\_uint32xm2\_uint16xm1 (C function), 345
- vwmuluvv\_uint32xm4\_uint16xm2 (C function), 345
- vwmuluvv\_uint32xm8\_uint16xm4 (C function), 345
- vwmuluvv\_uint64xm2\_uint32xm1 (C function), 345
- vwmuluvv\_uint64xm4\_uint32xm2 (C function), 345
- vwmuluvv\_uint64xm8\_uint32xm4 (C function), 345
- vwmuluvx\_mask\_uint16xm2\_uint8xm1 (C function), 347
- vwmuluvx\_mask\_uint16xm4\_uint8xm2 (C function), 347
- vwmuluvx\_mask\_uint16xm8\_uint8xm4 (C function), 347
- vwmuluvx\_mask\_uint32xm2\_uint16xm1 (C function), 347
- vwmuluvx\_mask\_uint32xm4\_uint16xm2 (C function), 347
- vwmuluvx\_mask\_uint32xm8\_uint16xm4 (C function), 347
- vwmuluvx\_mask\_uint64xm2\_uint32xm1 (C function), 347
- vwmuluvx\_mask\_uint64xm4\_uint32xm2 (C function), 347
- vwmuluvx\_mask\_uint64xm8\_uint32xm4 (C function), 347
- vwmuluvx\_uint16xm2\_uint8xm1 (C function), 346
- vwmuluvx\_uint16xm4\_uint8xm2 (C function), 346
- vwmuluvx\_uint16xm8\_uint8xm4 (C function), 346
- vwmuluvx\_uint32xm2\_uint16xm1 (C function), 346
- vwmuluvx\_uint32xm4\_uint16xm2 (C function), 347
- vwmuluvx\_uint32xm8\_uint16xm4 (C function), 347
- vwmuluvx\_uint64xm2\_uint32xm1 (C function), 347
- vwmuluvx\_uint64xm4\_uint32xm2 (C function), 347
- vwmuluvx\_uint64xm8\_uint32xm4 (C function), 347
- vwmulvv\_int16xm2\_int8xm1 (C function), 341
- vwmulvv\_int16xm4\_int8xm2 (C function), 341
- vwmulvv\_int16xm8\_int8xm4 (C function), 341
- vwmulvv\_int32xm2\_int16xm1 (C function), 341
- vwmulvv\_int32xm4\_int16xm2 (C function), 341
- vwmulvv\_int32xm8\_int16xm4 (C function), 341
- vwmulvv\_int64xm2\_int32xm1 (C function), 341
- vwmulvv\_int64xm4\_int32xm2 (C function), 341
- vwmulvv\_int64xm8\_int32xm4 (C function), 341
- vwmulvv\_mask\_int16xm2\_int8xm1 (C function), 341
- vwmulvv\_mask\_int16xm4\_int8xm2 (C function), 341
- vwmulvv\_mask\_int16xm8\_int8xm4 (C function), 341
- vwmulvv\_mask\_int32xm2\_int16xm1 (C function), 341
- vwmulvv\_mask\_int32xm4\_int16xm2 (C function), 341
- vwmulvv\_mask\_int32xm8\_int16xm4 (C function), 341
- vwmulvv\_mask\_int64xm2\_int32xm1 (C function), 341
- vwmulvv\_mask\_int64xm4\_int32xm2 (C function), 341
- vwmulvv\_mask\_int64xm8\_int32xm4 (C function), 341
- vwmulvx\_int16xm2\_int8xm1 (C function), 342
- vwmulvx\_int16xm4\_int8xm2 (C function), 342
- vwmulvx\_int16xm8\_int8xm4 (C function), 342
- vwmulvx\_int32xm2\_int16xm1 (C function), 342
- vwmulvx\_int32xm4\_int16xm2 (C function), 342
- vwmulvx\_int32xm8\_int16xm4 (C function), 342
- vwmulvx\_int64xm2\_int32xm1 (C function), 342
- vwmulvx\_int64xm4\_int32xm2 (C function), 342
- vwmulvx\_int64xm8\_int32xm4 (C function), 342
- vwmulvx\_mask\_int16xm2\_int8xm1 (C function), 342
- vwmulvx\_mask\_int16xm4\_int8xm2 (C function), 342
- vwmulvx\_mask\_int16xm8\_int8xm4 (C function), 342
- vwmulvx\_mask\_int32xm2\_int16xm1 (C function), 342
- vwmulvx\_mask\_int32xm4\_int16xm2 (C function), 342
- vwmulvx\_mask\_int32xm8\_int16xm4 (C function), 342
- vwmulvx\_mask\_int64xm2\_int32xm1 (C function), 342
- vwmulvx\_mask\_int64xm4\_int32xm2 (C function), 342
- vwmulvx\_mask\_int64xm8\_int32xm4 (C function), 342
- vwredsumuvs\_mask\_uint16xm2\_uint8xm1 (C function), 349
- vwredsumuvs\_mask\_uint16xm4\_uint8xm2 (C function), 350
- vwredsumuvs\_mask\_uint16xm8\_uint8xm4 (C function), 350
- vwredsumuvs\_mask\_uint32xm2\_uint16xm1 (C function), 350
- vwredsumuvs\_mask\_uint32xm4\_uint16xm2 (C function), 350
- vwredsumuvs\_mask\_uint32xm8\_uint16xm4 (C function), 350
- vwredsumuvs\_mask\_uint64xm2\_uint32xm1 (C function), 350
- vwredsumuvs\_mask\_uint64xm4\_uint32xm2 (C function), 350
- vwredsumuvs\_mask\_uint64xm8\_uint32xm4 (C function), 350
- vwredsumuvs\_uint16xm2\_uint8xm1 (C function), 349
- vwredsumuvs\_uint16xm4\_uint8xm2 (C function), 349
- vwredsumuvs\_uint16xm8\_uint8xm4 (C function), 349
- vwredsumuvs\_uint32xm2\_uint16xm1 (C function), 349
- vwredsumuvs\_uint32xm4\_uint16xm2 (C function), 349
- vwredsumuvs\_uint32xm8\_uint16xm4 (C function), 349
- vwredsumuvs\_uint64xm2\_uint32xm1 (C function), 349
- vwredsumuvs\_uint64xm4\_uint32xm2 (C function), 349
- vwredsumuvs\_uint64xm8\_uint32xm4 (C function), 349
- vwredsumvs\_int16xm2\_int8xm1 (C function), 348
- vwredsumvs\_int16xm4\_int8xm2 (C function), 348
- vwredsumvs\_int16xm8\_int8xm4 (C function), 348

- vwredsumvs\_int32xm2\_int16xm1 (C function), 348  
 vwredsumvs\_int32xm4\_int16xm2 (C function), 348  
 vwredsumvs\_int32xm8\_int16xm4 (C function), 348  
 vwredsumvs\_int64xm2\_int32xm1 (C function), 348  
 vwredsumvs\_int64xm4\_int32xm2 (C function), 348  
 vwredsumvs\_int64xm8\_int32xm4 (C function), 348  
 vwredsumvs\_mask\_int16xm2\_int8xm1 (C function), 348  
 vwredsumvs\_mask\_int16xm4\_int8xm2 (C function), 348  
 vwredsumvs\_mask\_int16xm8\_int8xm4 (C function), 348  
 vwredsumvs\_mask\_int32xm2\_int16xm1 (C function), 348  
 vwredsumvs\_mask\_int32xm4\_int16xm2 (C function), 348  
 vwredsumvs\_mask\_int32xm8\_int16xm4 (C function), 348  
 vwredsumvs\_mask\_int64xm2\_int32xm1 (C function), 348  
 vwredsumvs\_mask\_int64xm4\_int32xm2 (C function), 348  
 vwredsumvs\_mask\_int64xm8\_int32xm4 (C function), 349  
 vwsmaccsuvv\_int16xm2\_int8xm1\_uint8xm1 (C function), 353  
 vwsmaccsuvv\_int16xm4\_int8xm2\_uint8xm2 (C function), 353  
 vwsmaccsuvv\_int16xm8\_int8xm4\_uint8xm4 (C function), 353  
 vwsmaccsuvv\_int32xm2\_int16xm1\_uint16xm1 (C function), 353  
 vwsmaccsuvv\_int32xm4\_int16xm2\_uint16xm2 (C function), 353  
 vwsmaccsuvv\_int32xm8\_int16xm4\_uint16xm4 (C function), 353  
 vwsmaccsuvv\_int64xm2\_int32xm1\_uint32xm1 (C function), 354  
 vwsmaccsuvv\_int64xm4\_int32xm2\_uint32xm2 (C function), 354  
 vwsmaccsuvv\_int64xm8\_int32xm4\_uint32xm4 (C function), 354  
 vwsmaccsuvv\_mask\_int16xm2\_int8xm1\_uint8xm1 (C function), 354  
 vwsmaccsuvv\_mask\_int16xm4\_int8xm2\_uint8xm2 (C function), 354  
 vwsmaccsuvv\_mask\_int16xm8\_int8xm4\_uint8xm4 (C function), 354  
 vwsmaccsuvv\_mask\_int32xm2\_int16xm1\_uint16xm1 (C function), 354  
 vwsmaccsuvv\_mask\_int32xm4\_int16xm2\_uint16xm2 (C function), 354  
 vwsmaccsuvv\_mask\_int32xm8\_int16xm4\_uint16xm4 (C function), 354  
 vwsmaccsuvv\_mask\_int64xm2\_int32xm1\_uint32xm1 (C function), 354  
 vwsmaccsuvv\_mask\_int64xm4\_int32xm2\_uint32xm2 (C function), 355  
 vwsmaccsuvx\_int16xm2\_uint8xm1 (C function), 355  
 vwsmaccsuvx\_int16xm4\_uint8xm2 (C function), 355  
 vwsmaccsuvx\_int16xm8\_uint8xm4 (C function), 355  
 vwsmaccsuvx\_int32xm2\_uint16xm1 (C function), 355  
 vwsmaccsuvx\_int32xm4\_uint16xm2 (C function), 355  
 vwsmaccsuvx\_int32xm8\_uint16xm4 (C function), 355  
 vwsmaccsuvx\_int64xm2\_uint32xm1 (C function), 355  
 vwsmaccsuvx\_int64xm4\_uint32xm2 (C function), 355  
 vwsmaccsuvx\_int64xm8\_uint32xm4 (C function), 356  
 vwsmaccsuvx\_mask\_int16xm2\_uint8xm1 (C function), 356  
 vwsmaccsuvx\_mask\_int16xm4\_uint8xm2 (C function), 356  
 vwsmaccsuvx\_mask\_int16xm8\_uint8xm4 (C function), 356  
 vwsmaccsuvx\_mask\_int32xm2\_uint16xm1 (C function), 356  
 vwsmaccsuvx\_mask\_int32xm4\_uint16xm2 (C function), 356  
 vwsmaccsuvx\_mask\_int32xm8\_uint16xm4 (C function), 356  
 vwsmaccsuvx\_mask\_int64xm2\_uint32xm1 (C function), 356  
 vwsmaccsuvx\_mask\_int64xm4\_uint32xm2 (C function), 356  
 vwsmaccsuvx\_mask\_int64xm8\_uint32xm4 (C function), 356  
 vwsmaccusvx\_int16xm2\_int8xm1 (C function), 359  
 vwsmaccusvx\_int16xm4\_int8xm2 (C function), 359  
 vwsmaccusvx\_int16xm8\_int8xm4 (C function), 359  
 vwsmaccusvx\_int32xm2\_int16xm1 (C function), 359  
 vwsmaccusvx\_int32xm4\_int16xm2 (C function), 359  
 vwsmaccusvx\_int32xm8\_int16xm4 (C function), 360  
 vwsmaccusvx\_int64xm2\_int32xm1 (C function), 360  
 vwsmaccusvx\_int64xm4\_int32xm2 (C function), 360  
 vwsmaccusvx\_int64xm8\_int32xm4 (C function), 360  
 vwsmaccusvx\_mask\_int16xm2\_int8xm1 (C function), 360  
 vwsmaccusvx\_mask\_int16xm4\_int8xm2 (C function), 360  
 vwsmaccusvx\_mask\_int16xm8\_int8xm4 (C function), 360  
 vwsmaccusvx\_mask\_int32xm2\_int16xm1 (C function), 360  
 vwsmaccusvx\_mask\_int32xm4\_int16xm2 (C function), 360  
 vwsmaccusvx\_mask\_int32xm8\_int16xm4 (C function), 360  
 vwsmaccusvx\_mask\_int64xm2\_int32xm1 (C function), 360  
 vwsmaccusvx\_mask\_int64xm4\_int32xm2 (C function),

[illegible]

- vwsbuvv\_mask\_uint32xm2\_uint16xm1 (C function), 365  
 vwsbuvv\_mask\_uint32xm4\_uint16xm2 (C function), 366  
 vwsbuvv\_mask\_uint32xm8\_uint16xm4 (C function), 366  
 vwsbuvv\_mask\_uint64xm2\_uint32xm1 (C function), 366  
 vwsbuvv\_mask\_uint64xm4\_uint32xm2 (C function), 366  
 vwsbuvv\_mask\_uint64xm8\_uint32xm4 (C function), 366  
 vwsbuvv\_uint16xm2\_uint8xm1 (C function), 365  
 vwsbuvv\_uint16xm4\_uint8xm2 (C function), 365  
 vwsbuvv\_uint16xm8\_uint8xm4 (C function), 365  
 vwsbuvv\_uint32xm2\_uint16xm1 (C function), 365  
 vwsbuvv\_uint32xm4\_uint16xm2 (C function), 365  
 vwsbuvv\_uint32xm8\_uint16xm4 (C function), 365  
 vwsbuvv\_uint64xm2\_uint32xm1 (C function), 365  
 vwsbuvv\_uint64xm4\_uint32xm2 (C function), 365  
 vwsbuvv\_uint64xm8\_uint32xm4 (C function), 365  
 vwsbuvx\_mask\_uint16xm2\_uint8xm1 (C function), 367  
 vwsbuvx\_mask\_uint16xm4\_uint8xm2 (C function), 367  
 vwsbuvx\_mask\_uint16xm8\_uint8xm4 (C function), 367  
 vwsbuvx\_mask\_uint32xm2\_uint16xm1 (C function), 367  
 vwsbuvx\_mask\_uint32xm4\_uint16xm2 (C function), 367  
 vwsbuvx\_mask\_uint32xm8\_uint16xm4 (C function), 367  
 vwsbuvx\_mask\_uint64xm2\_uint32xm1 (C function), 367  
 vwsbuvx\_mask\_uint64xm4\_uint32xm2 (C function), 367  
 vwsbuvx\_mask\_uint64xm8\_uint32xm4 (C function), 367  
 vwsbuvx\_uint16xm2\_uint8xm1 (C function), 366  
 vwsbuvx\_uint16xm4\_uint8xm2 (C function), 366  
 vwsbuvx\_uint16xm8\_uint8xm4 (C function), 366  
 vwsbuvx\_uint32xm2\_uint16xm1 (C function), 366  
 vwsbuvx\_uint32xm4\_uint16xm2 (C function), 366  
 vwsbuvx\_uint32xm8\_uint16xm4 (C function), 366  
 vwsbuvx\_uint64xm2\_uint32xm1 (C function), 366  
 vwsbuvx\_uint64xm4\_uint32xm2 (C function), 366  
 vwsbuvx\_uint64xm8\_uint32xm4 (C function), 366  
 vwsbuwv\_mask\_uint16xm2\_uint8xm1 (C function), 368  
 vwsbuwv\_mask\_uint16xm4\_uint8xm2 (C function), 368  
 vwsbuwv\_mask\_uint16xm8\_uint8xm4 (C function), 368  
 vwsbuwv\_mask\_uint32xm2\_uint16xm1 (C function), 368  
 vwsbuwv\_mask\_uint32xm4\_uint16xm2 (C function), 368  
 vwsbuwv\_mask\_uint32xm8\_uint16xm4 (C function), 368  
 vwsbuwv\_mask\_uint64xm2\_uint32xm1 (C function), 368  
 vwsbuwv\_mask\_uint64xm4\_uint32xm2 (C function), 368  
 vwsbuwv\_mask\_uint64xm8\_uint32xm4 (C function), 368  
 vwsbuwv\_uint16xm2\_uint8xm1 (C function), 367  
 vwsbuwv\_uint16xm4\_uint8xm2 (C function), 367  
 vwsbuwv\_uint16xm8\_uint8xm4 (C function), 367  
 vwsbuwv\_uint32xm2\_uint16xm1 (C function), 367  
 vwsbuwv\_uint32xm4\_uint16xm2 (C function), 367  
 vwsbuwv\_uint32xm8\_uint16xm4 (C function), 367  
 vwsbuwv\_uint64xm2\_uint32xm1 (C function), 368  
 vwsbuwv\_uint64xm4\_uint32xm2 (C function), 368  
 vwsbuwv\_uint64xm8\_uint32xm4 (C function), 368  
 vwsbuwx\_mask\_uint16xm2 (C function), 369  
 vwsbuwx\_mask\_uint16xm4 (C function), 369  
 vwsbuwx\_mask\_uint16xm8 (C function), 369  
 vwsbuwx\_mask\_uint32xm2 (C function), 369  
 vwsbuwx\_mask\_uint32xm4 (C function), 369  
 vwsbuwx\_mask\_uint32xm8 (C function), 369  
 vwsbuwx\_mask\_uint64xm2 (C function), 369  
 vwsbuwx\_mask\_uint64xm4 (C function), 369  
 vwsbuwx\_mask\_uint64xm8 (C function), 369  
 vwsbuwx\_uint16xm2 (C function), 369  
 vwsbuwx\_uint16xm4 (C function), 369  
 vwsbuwx\_uint16xm8 (C function), 369  
 vwsbuwx\_uint32xm2 (C function), 369  
 vwsbuwx\_uint32xm4 (C function), 369  
 vwsbuwx\_uint32xm8 (C function), 369  
 vwsbuwx\_uint64xm2 (C function), 369  
 vwsbuwx\_uint64xm4 (C function), 369  
 vwsbuwx\_uint64xm8 (C function), 369  
 vwsbvv\_int16xm2\_int8xm1 (C function), 361  
 vwsbvv\_int16xm4\_int8xm2 (C function), 361  
 vwsbvv\_int16xm8\_int8xm4 (C function), 361  
 vwsbvv\_int32xm2\_int16xm1 (C function), 361  
 vwsbvv\_int32xm4\_int16xm2 (C function), 361  
 vwsbvv\_int32xm8\_int16xm4 (C function), 361  
 vwsbvv\_int64xm2\_int32xm1 (C function), 361  
 vwsbvv\_int64xm4\_int32xm2 (C function), 361  
 vwsbvv\_int64xm8\_int32xm4 (C function), 361  
 vwsbvv\_mask\_int16xm2\_int8xm1 (C function), 361  
 vwsbvv\_mask\_int16xm4\_int8xm2 (C function), 361  
 vwsbvv\_mask\_int16xm8\_int8xm4 (C function), 361  
 vwsbvv\_mask\_int32xm2\_int16xm1 (C function), 361  
 vwsbvv\_mask\_int32xm4\_int16xm2 (C function), 361  
 vwsbvv\_mask\_int32xm8\_int16xm4 (C function), 361  
 vwsbvv\_mask\_int64xm2\_int32xm1 (C function), 361  
 vwsbvv\_mask\_int64xm4\_int32xm2 (C function), 361  
 vwsbvv\_mask\_int64xm8\_int32xm4 (C function), 361



vwsbvx\_int16xm2\_int8xm1 (C function), 362  
vwsbvx\_int16xm4\_int8xm2 (C function), 362  
vwsbvx\_int16xm8\_int8xm4 (C function), 362  
vwsbvx\_int32xm2\_int16xm1 (C function), 362  
vwsbvx\_int32xm4\_int16xm2 (C function), 362  
vwsbvx\_int32xm8\_int16xm4 (C function), 362  
vwsbvx\_int64xm2\_int32xm1 (C function), 362  
vwsbvx\_int64xm4\_int32xm2 (C function), 362  
vwsbvx\_int64xm8\_int32xm4 (C function), 362  
vwsbvx\_mask\_int16xm2\_int8xm1 (C function), 362  
vwsbvx\_mask\_int16xm4\_int8xm2 (C function), 362  
vwsbvx\_mask\_int16xm8\_int8xm4 (C function), 362  
vwsbvx\_mask\_int32xm2\_int16xm1 (C function), 362  
vwsbvx\_mask\_int32xm4\_int16xm2 (C function), 362  
vwsbvx\_mask\_int32xm8\_int16xm4 (C function), 362  
vwsbvx\_mask\_int64xm2\_int32xm1 (C function), 362  
vwsbvx\_mask\_int64xm4\_int32xm2 (C function), 362  
vwsbvx\_mask\_int64xm8\_int32xm4 (C function), 362  
vwsbwv\_int16xm2\_int8xm1 (C function), 363  
vwsbwv\_int16xm4\_int8xm2 (C function), 363  
vwsbwv\_int16xm8\_int8xm4 (C function), 363  
vwsbwv\_int32xm2\_int16xm1 (C function), 363  
vwsbwv\_int32xm4\_int16xm2 (C function), 363  
vwsbwv\_int32xm8\_int16xm4 (C function), 363  
vwsbwv\_int64xm2\_int32xm1 (C function), 363  
vwsbwv\_int64xm4\_int32xm2 (C function), 363  
vwsbwv\_int64xm8\_int32xm4 (C function), 363  
vwsbwv\_mask\_int16xm2\_int8xm1 (C function), 363  
vwsbwv\_mask\_int16xm4\_int8xm2 (C function), 363  
vwsbwv\_mask\_int16xm8\_int8xm4 (C function), 363  
vwsbwv\_mask\_int32xm2\_int16xm1 (C function), 363  
vwsbwv\_mask\_int32xm4\_int16xm2 (C function), 363  
vwsbwv\_mask\_int32xm8\_int16xm4 (C function), 363  
vwsbwv\_mask\_int64xm2\_int32xm1 (C function), 363  
vwsbwv\_mask\_int64xm4\_int32xm2 (C function), 363  
vwsbwv\_mask\_int64xm8\_int32xm4 (C function), 364  
vwsbwx\_int16xm2 (C function), 364  
vwsbwx\_int16xm4 (C function), 364  
vwsbwx\_int16xm8 (C function), 364  
vwsbwx\_int32xm2 (C function), 364  
vwsbwx\_int32xm4 (C function), 364  
vwsbwx\_int32xm8 (C function), 364  
vwsbwx\_int64xm2 (C function), 364  
vwsbwx\_int64xm4 (C function), 364  
vwsbwx\_int64xm8 (C function), 364  
vwsbwx\_mask\_int16xm2 (C function), 364  
vwsbwx\_mask\_int16xm4 (C function), 364  
vwsbwx\_mask\_int16xm8 (C function), 364  
vwsbwx\_mask\_int32xm2 (C function), 364  
vwsbwx\_mask\_int32xm4 (C function), 364  
vwsbwx\_mask\_int32xm8 (C function), 364  
vwsbwx\_mask\_int64xm2 (C function), 364  
vwsbwx\_mask\_int64xm4 (C function), 365  
vwsbwx\_mask\_int64xm8 (C function), 365  
vxorvi\_int16xm1 (C function), 42  
vxorvi\_int16xm2 (C function), 42  
vxorvi\_int16xm4 (C function), 42  
vxorvi\_int16xm8 (C function), 42  
vxorvi\_int32xm1 (C function), 42  
vxorvi\_int32xm2 (C function), 42  
vxorvi\_int32xm4 (C function), 42  
vxorvi\_int32xm8 (C function), 42  
vxorvi\_int64xm1 (C function), 42  
vxorvi\_int64xm2 (C function), 42  
vxorvi\_int64xm4 (C function), 42  
vxorvi\_int64xm8 (C function), 42  
vxorvi\_int8xm1 (C function), 42  
vxorvi\_int8xm2 (C function), 42  
vxorvi\_int8xm4 (C function), 42  
vxorvi\_int8xm8 (C function), 42  
vxorvi\_mask\_int16xm1 (C function), 43  
vxorvi\_mask\_int16xm2 (C function), 43  
vxorvi\_mask\_int16xm4 (C function), 43  
vxorvi\_mask\_int16xm8 (C function), 43  
vxorvi\_mask\_int32xm1 (C function), 43  
vxorvi\_mask\_int32xm2 (C function), 43  
vxorvi\_mask\_int32xm4 (C function), 43  
vxorvi\_mask\_int32xm8 (C function), 43  
vxorvi\_mask\_int64xm1 (C function), 43  
vxorvi\_mask\_int64xm2 (C function), 43  
vxorvi\_mask\_int64xm4 (C function), 43  
vxorvi\_mask\_int64xm8 (C function), 43  
vxorvi\_mask\_int8xm1 (C function), 43  
vxorvi\_mask\_int8xm2 (C function), 43  
vxorvi\_mask\_int8xm4 (C function), 44  
vxorvi\_mask\_int8xm8 (C function), 44  
vxorvi\_mask\_uint16xm1 (C function), 44  
vxorvi\_mask\_uint16xm2 (C function), 44  
vxorvi\_mask\_uint16xm4 (C function), 44  
vxorvi\_mask\_uint16xm8 (C function), 44  
vxorvi\_mask\_uint32xm1 (C function), 44  
vxorvi\_mask\_uint32xm2 (C function), 44  
vxorvi\_mask\_uint32xm4 (C function), 44  
vxorvi\_mask\_uint32xm8 (C function), 44  
vxorvi\_mask\_uint64xm1 (C function), 44  
vxorvi\_mask\_uint64xm2 (C function), 44  
vxorvi\_mask\_uint64xm4 (C function), 44  
vxorvi\_mask\_uint64xm8 (C function), 44  
vxorvi\_mask\_uint8xm1 (C function), 44  
vxorvi\_mask\_uint8xm2 (C function), 44  
vxorvi\_mask\_uint8xm4 (C function), 44  
vxorvi\_mask\_uint8xm8 (C function), 44  
vxorvi\_uint16xm1 (C function), 42  
vxorvi\_uint16xm2 (C function), 42  
vxorvi\_uint16xm4 (C function), 42  
vxorvi\_uint16xm8 (C function), 42  
vxorvi\_uint32xm1 (C function), 42  
vxorvi\_uint32xm2 (C function), 42

vxorvi\_uint32xm4 (C function), 43  
 vxorvi\_uint32xm8 (C function), 43  
 vxorvi\_uint64xm1 (C function), 43  
 vxorvi\_uint64xm2 (C function), 43  
 vxorvi\_uint64xm4 (C function), 43  
 vxorvi\_uint64xm8 (C function), 43  
 vxorvi\_uint8xm1 (C function), 43  
 vxorvi\_uint8xm2 (C function), 43  
 vxorvi\_uint8xm4 (C function), 43  
 vxorvi\_uint8xm8 (C function), 43  
 vxorvv\_int16xm1 (C function), 45  
 vxorvv\_int16xm2 (C function), 45  
 vxorvv\_int16xm4 (C function), 45  
 vxorvv\_int16xm8 (C function), 45  
 vxorvv\_int32xm1 (C function), 45  
 vxorvv\_int32xm2 (C function), 45  
 vxorvv\_int32xm4 (C function), 45  
 vxorvv\_int32xm8 (C function), 45  
 vxorvv\_int64xm1 (C function), 45  
 vxorvv\_int64xm2 (C function), 45  
 vxorvv\_int64xm4 (C function), 45  
 vxorvv\_int64xm8 (C function), 45  
 vxorvv\_int8xm1 (C function), 45  
 vxorvv\_int8xm2 (C function), 45  
 vxorvv\_int8xm4 (C function), 45  
 vxorvv\_int8xm8 (C function), 45  
 vxorvv\_mask\_int16xm1 (C function), 46  
 vxorvv\_mask\_int16xm2 (C function), 46  
 vxorvv\_mask\_int16xm4 (C function), 46  
 vxorvv\_mask\_int16xm8 (C function), 46  
 vxorvv\_mask\_int32xm1 (C function), 46  
 vxorvv\_mask\_int32xm2 (C function), 46  
 vxorvv\_mask\_int32xm4 (C function), 46  
 vxorvv\_mask\_int32xm8 (C function), 46  
 vxorvv\_mask\_int64xm1 (C function), 46  
 vxorvv\_mask\_int64xm2 (C function), 46  
 vxorvv\_mask\_int64xm4 (C function), 46  
 vxorvv\_mask\_int64xm8 (C function), 46  
 vxorvv\_mask\_int8xm1 (C function), 46  
 vxorvv\_mask\_int8xm2 (C function), 46  
 vxorvv\_mask\_int8xm4 (C function), 46  
 vxorvv\_mask\_int8xm8 (C function), 46  
 vxorvv\_mask\_uint16xm1 (C function), 46  
 vxorvv\_mask\_uint16xm2 (C function), 46  
 vxorvv\_mask\_uint16xm4 (C function), 46  
 vxorvv\_mask\_uint16xm8 (C function), 46  
 vxorvv\_mask\_uint32xm1 (C function), 46  
 vxorvv\_mask\_uint32xm2 (C function), 47  
 vxorvv\_mask\_uint32xm4 (C function), 47  
 vxorvv\_mask\_uint32xm8 (C function), 47  
 vxorvv\_mask\_uint64xm1 (C function), 47  
 vxorvv\_mask\_uint64xm2 (C function), 47  
 vxorvv\_mask\_uint64xm4 (C function), 47  
 vxorvv\_mask\_uint64xm8 (C function), 47  
 vxorvv\_mask\_uint8xm1 (C function), 47  
 vxorvv\_mask\_uint8xm2 (C function), 47  
 vxorvv\_mask\_uint8xm4 (C function), 47  
 vxorvv\_mask\_uint8xm8 (C function), 47  
 vxorvv\_uint16xm1 (C function), 45  
 vxorvv\_uint16xm2 (C function), 45  
 vxorvv\_uint16xm4 (C function), 45  
 vxorvv\_uint16xm8 (C function), 45  
 vxorvv\_uint32xm1 (C function), 45  
 vxorvv\_uint32xm2 (C function), 45  
 vxorvv\_uint32xm4 (C function), 45  
 vxorvv\_uint32xm8 (C function), 45  
 vxorvv\_uint64xm1 (C function), 45  
 vxorvv\_uint64xm2 (C function), 45  
 vxorvv\_uint64xm4 (C function), 45  
 vxorvv\_uint64xm8 (C function), 45  
 vxorvv\_uint8xm1 (C function), 45  
 vxorvv\_uint8xm2 (C function), 45  
 vxorvv\_uint8xm4 (C function), 45  
 vxorvv\_uint8xm8 (C function), 45  
 vxorvx\_int16xm1 (C function), 47  
 vxorvx\_int16xm2 (C function), 47  
 vxorvx\_int16xm4 (C function), 47  
 vxorvx\_int16xm8 (C function), 47  
 vxorvx\_int32xm1 (C function), 47  
 vxorvx\_int32xm2 (C function), 47  
 vxorvx\_int32xm4 (C function), 47  
 vxorvx\_int32xm8 (C function), 47  
 vxorvx\_int64xm1 (C function), 47  
 vxorvx\_int64xm2 (C function), 48  
 vxorvx\_int64xm4 (C function), 48  
 vxorvx\_int64xm8 (C function), 48  
 vxorvx\_int8xm1 (C function), 48  
 vxorvx\_int8xm2 (C function), 48  
 vxorvx\_int8xm4 (C function), 48  
 vxorvx\_int8xm8 (C function), 48  
 vxorvx\_mask\_int16xm1 (C function), 48  
 vxorvx\_mask\_int16xm2 (C function), 48  
 vxorvx\_mask\_int16xm4 (C function), 48  
 vxorvx\_mask\_int16xm8 (C function), 48  
 vxorvx\_mask\_int32xm1 (C function), 48  
 vxorvx\_mask\_int32xm2 (C function), 49  
 vxorvx\_mask\_int32xm4 (C function), 49  
 vxorvx\_mask\_int32xm8 (C function), 49  
 vxorvx\_mask\_int64xm1 (C function), 49  
 vxorvx\_mask\_int64xm2 (C function), 49  
 vxorvx\_mask\_int64xm4 (C function), 49  
 vxorvx\_mask\_int64xm8 (C function), 49  
 vxorvx\_mask\_int8xm1 (C function), 49  
 vxorvx\_mask\_int8xm2 (C function), 49  
 vxorvx\_mask\_int8xm4 (C function), 49  
 vxorvx\_mask\_int8xm8 (C function), 49  
 vxorvx\_mask\_uint16xm1 (C function), 49  
 vxorvx\_mask\_uint16xm2 (C function), 49



`vxorvx_mask_uint16xm4` (C function), 49  
`vxorvx_mask_uint16xm8` (C function), 49  
`vxorvx_mask_uint32xm1` (C function), 49  
`vxorvx_mask_uint32xm2` (C function), 49  
`vxorvx_mask_uint32xm4` (C function), 49  
`vxorvx_mask_uint32xm8` (C function), 49  
`vxorvx_mask_uint64xm1` (C function), 49  
`vxorvx_mask_uint64xm2` (C function), 49  
`vxorvx_mask_uint64xm4` (C function), 49  
`vxorvx_mask_uint64xm8` (C function), 49  
`vxorvx_mask_uint8xm1` (C function), 50  
`vxorvx_mask_uint8xm2` (C function), 50  
`vxorvx_mask_uint8xm4` (C function), 50  
`vxorvx_mask_uint8xm8` (C function), 50  
`vxorvx_uint16xm1` (C function), 48  
`vxorvx_uint16xm2` (C function), 48  
`vxorvx_uint16xm4` (C function), 48  
`vxorvx_uint16xm8` (C function), 48  
`vxorvx_uint32xm1` (C function), 48  
`vxorvx_uint32xm2` (C function), 48  
`vxorvx_uint32xm4` (C function), 48  
`vxorvx_uint32xm8` (C function), 48  
`vxorvx_uint64xm1` (C function), 48  
`vxorvx_uint64xm2` (C function), 48  
`vxorvx_uint64xm4` (C function), 48  
`vxorvx_uint64xm8` (C function), 48  
`vxorvx_uint8xm1` (C function), 48  
`vxorvx_uint8xm2` (C function), 48  
`vxorvx_uint8xm4` (C function), 48  
`vxorvx_uint8xm8` (C function), 48