Vector Intrinsic Manual

Release 1.1

Apr 15, 2020

CONTENTS

CHAPTER

ONE

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	`float64xm1_t`	`float64xm2_t`	`float64xm4_t`	`float64xm8_t`
float	`float32xm1_t`	`float32xm2_t`	`float32xm4_t`	`float32xm8_t`
float16	`float16xm1_t`	`float16xm2_t`	`float16xm4_t`	`float16xm8_t`
int64	`int64xm1_t`	`int64xm2_t`	`int64xm4_t`	`int64xm8_t`
int32	`int32xm1_t`	`int32xm2_t`	`int32xm4_t`	`int32xm8_t`
int16	`int16xm1_t`	`int16xm2_t`	`int16xm4_t`	`int16xm8_t`
int8	`int8xm1_t`	`int8xm2_t`	`int8xm4_t`	`int8xm8_t`
uint64	`uint64xm1_t`	`uint64xm2_t`	`uint64xm4_t`	`uint64xm8_t`
uint32	`uint32xm1_t`	`uint32xm2_t`	`uint32xm4_t`	`uint32xm8_t`
uint16	`uint16xm1_t`	`uint16xm2_t`	`uint16xm4_t`	`uint16xm8_t`
uint8	`uint8xm1_t`	`uint8xm2_t`	`uint8xm4_t`	`uint8xm8_t`

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	`e64xm1_t`	`e64xm2_t`	`e64xm4_t`	`e64xm8_t`
32	`e32xm1_t`	`e32xm2_t`	`e32xm4_t`	`e32xm8_t`
16	`e16xm1_t`	`e16xm2_t`	`e16xm4_t`	`e16xm8_t`
8	`e8xm1_t`	`e8xm2_t`	`e8xm4_t`	`e8xm8_t`

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

CHAPTER

TWO

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: ['vxor.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: ['vfncvt.f.f.v']

Prototypes:

- *float16xm1_t* **vfncvtffv_float16xm1_float32xm2** (*float32xm2_t a*, unsigned int *gvl*)
- *float16xm2_t* **vfncvtffv_float16xm2_float32xm4** (*float32xm4_t a*, unsigned int *gvl*)
- *float16xm4_t* **vfncvtffv_float16xm4_float32xm8** (*float32xm8_t a*, unsigned int *gvl*)
- *float32xm1_t* **vfncvtffv_float32xm1_float64xm2** (*float64xm2_t a*, unsigned int *gvl*)
- *float32xm2_t* **vfncvtffv_float32xm2_float64xm4** (*float64xm4_t a*, unsigned int *gvl*)
- *float32xm4_t* **vfncvtffv_float32xm4_float64xm8** (*float64xm8_t a*, unsigned int *gvl*)

Operation:

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

Masked prototypes:

- *float16xm1_t* **vfncvtffv_mask_float16xm1_float32xm2** (*float16xm1_t merge*, *float32xm2_t a*, *e16xm1_t mask*, unsigned int *gvl*)
- *float16xm2_t* **vfncvtffv_mask_float16xm2_float32xm4** (*float16xm2_t merge*, *float32xm4_t a*, *e16xm2_t mask*, unsigned int *gvl*)
- *float16xm4_t* **vfncvtffv_mask_float16xm4_float32xm8** (*float16xm4_t merge*, *float32xm8_t a*, *e16xm4_t mask*, unsigned int *gvl*)
- *float32xm1_t* **vfncvtffv_mask_float32xm1_float64xm2** (*float32xm1_t merge*, *float64xm2_t a*, *e32xm1_t mask*, unsigned int *gvl*)
- *float32xm2_t* **vfncvtffv_mask_float32xm2_float64xm4** (*float32xm2_t merge*, *float64xm4_t a*, *e32xm2_t mask*, unsigned int *gvl*)

- *float32xm4_t* **vfncvtffv_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[elemnt] = 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 interger to floating-point

Instruction: ['vfcvt.f.x.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[elemnt] = 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[elemnt] = 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 interger

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[elemnt] = 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 interger 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[elemnt] = 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: ['vfncvt.f.xu.v']

Prototypes:

- *float16xm1_t* **vfncvtfxuv_float16xm1_uint32xm2** (*uint32xm2_t a*, unsigned int *gvl*)
- *float16xm2_t* **vfncvtfxuv_float16xm2_uint32xm4** (*uint32xm4_t a*, unsigned int *gvl*)
- *float16xm4_t* **vfncvtfxuv_float16xm4_uint32xm8** (*uint32xm8_t a*, unsigned int *gvl*)
- *float32xm1_t* **vfncvtfxuv_float32xm1_uint64xm2** (*uint64xm2_t a*, unsigned int *gvl*)
- *float32xm2_t* **vfncvtfxuv_float32xm2_uint64xm4** (*uint64xm4_t a*, unsigned int *gvl*)
- *float32xm4_t* **vfncvtfxuv_float32xm4_uint64xm8** (*uint64xm8_t a*, unsigned int *gvl*)

Operation:

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

Masked prototypes:

- *float16xm1_t* **vfncvtfxuv_mask_float16xm1_uint32xm2** (*float16xm1_t merge*, *uint32xm2_t a*, *e16xm1_t mask*, unsigned int *gvl*)
- *float16xm2_t* **vfncvtfxuv_mask_float16xm2_uint32xm4** (*float16xm2_t merge*, *uint32xm4_t a*, *e16xm2_t mask*, unsigned int *gvl*)
- *float16xm4_t* **vfncvtfxuv_mask_float16xm4_uint32xm8** (*float16xm4_t merge*, *uint32xm8_t a*, *e16xm4_t mask*, unsigned int *gvl*)
- *float32xm1_t* **vfncvtfxuv_mask_float32xm1_uint64xm2** (*float32xm1_t merge*, *uint64xm2_t a*, *e32xm1_t mask*, unsigned int *gvl*)
- *float32xm2_t* **vfncvtfxuv_mask_float32xm2_uint64xm4** (*float32xm2_t merge*, *uint64xm4_t a*, *e32xm2_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfncvtfxuv_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: ['vfncvt.x.f.v']

Prototypes:

- *int16xm1_t* **vfncvtxfv_int16xm1_float32xm2** (*float32xm2_t a*, unsigned int *gvl*)
- *int16xm2_t* **vfncvtxfv_int16xm2_float32xm4** (*float32xm4_t a*, unsigned int *gvl*)
- *int16xm4_t* **vfncvtxfv_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[elemnt] = 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 interger

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[elemnt] = 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[elemnt] = 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 interger 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[elemnt] = 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[elemnt] = 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* **vfwcvtxufv_uint64xm2_float32xm1** (*float32xm1_t a*, unsigned int *gvl*)
- *uint64xm4_t* **vfwcvtxufv_uint64xm4_float32xm2** (*float32xm2_t a*, unsigned int *gvl*)
- *uint64xm8_t* **vfwcvtxufv_uint64xm8_float32xm4** (*float32xm4_t a*, unsigned int *gvl*)

Operation:

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

Masked prototypes:

- *uint32xm2_t* **vfwcvtxufv_mask_uint32xm2_float16xm1** (*uint32xm2_t merge*, *float16xm1_t a*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vfwcvtxufv_mask_uint32xm4_float16xm2** (*uint32xm4_t merge*, *float16xm2_t a*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vfwcvtxufv_mask_uint32xm8_float16xm4** (*uint32xm8_t merge*, *float16xm4_t a*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vfwcvtxufv_mask_uint64xm2_float32xm1** (*uint64xm2_t merge*, *float32xm1_t a*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vfwcvtxufv_mask_uint64xm4_float32xm2** (*uint64xm4_t merge*, *float32xm2_t a*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vfwcvtxufv_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* **vfdivvf_mask_float16xm1** (*float16xm1_t merge*, *float16xm1_t a*, float16_t *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float16xm2_t* **vfdivvf_mask_float16xm2** (*float16xm2_t merge*, *float16xm2_t a*, float16_t *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float16xm4_t* **vfdivvf_mask_float16xm4** (*float16xm4_t merge*, *float16xm4_t a*, float16_t *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *float16xm8_t* **vfdivvf_mask_float16xm8** (*float16xm8_t merge*, *float16xm8_t a*, float16_t *b*, *e16xm8_t mask*, unsigned int *gvl*)
- *float32xm1_t* **vfdivvf_mask_float32xm1** (*float32xm1_t merge*, *float32xm1_t a*, float *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *float32xm2_t* **vfdivvf_mask_float32xm2** (*float32xm2_t merge*, *float32xm2_t a*, float *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfdivvf_mask_float32xm4** (*float32xm4_t merge*, *float32xm4_t a*, float *b*, *e32xm4_t mask*, unsigned int *gvl*)
- *float32xm8_t* **vfdivvf_mask_float32xm8** (*float32xm8_t merge*, *float32xm8_t a*, float *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *float64xm1_t* **vfdivvf_mask_float64xm1** (*float64xm1_t merge*, *float64xm1_t a*, double *b*, *e64xm1_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vfdivvf_mask_float64xm2** (*float64xm2_t merge*, *float64xm2_t a*, double *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *float64xm4_t* **vfdivvf_mask_float64xm4** (*float64xm4_t merge*, *float64xm4_t a*, double *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *float64xm8_t* **vfdivvf_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: ['vfdiv.vv']

Prototypes:

- *float16xm1_t* **vfdivvv_float16xm1** (*float16xm1_t a*, *float16xm1_t b*, unsigned int *gvl*)
- *float16xm2_t* **vfdivvv_float16xm2** (*float16xm2_t a*, *float16xm2_t b*, unsigned int *gvl*)
- *float16xm4_t* **vfdivvv_float16xm4** (*float16xm4_t a*, *float16xm4_t b*, unsigned int *gvl*)
- *float16xm8_t* **vfdivvv_float16xm8** (*float16xm8_t a*, *float16xm8_t b*, unsigned int *gvl*)

- *float32xm1_t* **vfdivvv_float32xm1** (*float32xm1_t a*, *float32xm1_t b*, unsigned int *gvl*)
- *float32xm2_t* **vfdivvv_float32xm2** (*float32xm2_t a*, *float32xm2_t b*, unsigned int *gvl*)
- *float32xm4_t* **vfdivvv_float32xm4** (*float32xm4_t a*, *float32xm4_t b*, unsigned int *gvl*)
- *float32xm8_t* **vfdivvv_float32xm8** (*float32xm8_t a*, *float32xm8_t b*, unsigned int *gvl*)
- *float64xm1_t* **vfdivvv_float64xm1** (*float64xm1_t a*, *float64xm1_t b*, unsigned int *gvl*)
- *float64xm2_t* **vfdivvv_float64xm2** (*float64xm2_t a*, *float64xm2_t b*, unsigned int *gvl*)
- *float64xm4_t* **vfdivvv_float64xm4** (*float64xm4_t a*, *float64xm4_t b*, unsigned int *gvl*)
- *float64xm8_t* **vfdivvv_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* **vfdivvv_mask_float16xm1** (*float16xm1_t merge*, *float16xm1_t a*, *float16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float16xm2_t* **vfdivvv_mask_float16xm2** (*float16xm2_t merge*, *float16xm2_t a*, *float16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float16xm4_t* **vfdivvv_mask_float16xm4** (*float16xm4_t merge*, *float16xm4_t a*, *float16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *float16xm8_t* **vfdivvv_mask_float16xm8** (*float16xm8_t merge*, *float16xm8_t a*, *float16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)
- *float32xm1_t* **vfdivvv_mask_float32xm1** (*float32xm1_t merge*, *float32xm1_t a*, *float32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *float32xm2_t* **vfdivvv_mask_float32xm2** (*float32xm2_t merge*, *float32xm2_t a*, *float32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfdivvv_mask_float32xm4** (*float32xm4_t merge*, *float32xm4_t a*, *float32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)
- *float32xm8_t* **vfdivvv_mask_float32xm8** (*float32xm8_t merge*, *float32xm8_t a*, *float32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *float64xm1_t* **vfdivvv_mask_float64xm1** (*float64xm1_t merge*, *float64xm1_t a*, *float64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vfdivvv_mask_float64xm2** (*float64xm2_t merge*, *float64xm2_t a*, *float64xm2_t b*, *e64xm2_t mask*, unsigned int *gvl*)
- *float64xm4_t* **vfdivvv_mask_float64xm4** (*float64xm4_t merge*, *float64xm4_t a*, *float64xm4_t b*, *e64xm4_t mask*, unsigned int *gvl*)
- *float64xm8_t* **vfdivvv_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 maxmum

Instruction: ['vfmax.vf']

Prototypes:

- *float16xm1_t* **vfmaxvf_float16xm1** (*float16xm1_t a*, float16_t *b*, unsigned int *gvl*)
- *float16xm2_t* **vfmaxvf_float16xm2** (*float16xm2_t a*, float16_t *b*, unsigned int *gvl*)
- *float16xm4_t* **vfmaxvf_float16xm4** (*float16xm4_t a*, float16_t *b*, unsigned int *gvl*)
- *float16xm8_t* **vfmaxvf_float16xm8** (*float16xm8_t a*, float16_t *b*, unsigned int *gvl*)
- *float32xm1_t* **vfmaxvf_float32xm1** (*float32xm1_t a*, float *b*, unsigned int *gvl*)
- *float32xm2_t* **vfmaxvf_float32xm2** (*float32xm2_t a*, float *b*, unsigned int *gvl*)
- *float32xm4_t* **vfmaxvf_float32xm4** (*float32xm4_t a*, float *b*, unsigned int *gvl*)
- *float32xm8_t* **vfmaxvf_float32xm8** (*float32xm8_t a*, float *b*, unsigned int *gvl*)
- *float64xm1_t* **vfmaxvf_float64xm1** (*float64xm1_t a*, double *b*, unsigned int *gvl*)
- *float64xm2_t* **vfmaxvf_float64xm2** (*float64xm2_t a*, double *b*, unsigned int *gvl*)
- *float64xm4_t* **vfmaxvf_float64xm4** (*float64xm4_t a*, double *b*, unsigned int *gvl*)
- *float64xm8_t* **vfmaxvf_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* **vfmaxvf_mask_float16xm1** (*float16xm1_t merge*, *float16xm1_t a*, float16_t *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float16xm2_t* **vfmaxvf_mask_float16xm2** (*float16xm2_t merge*, *float16xm2_t a*, float16_t *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float16xm4_t* **vfmaxvf_mask_float16xm4** (*float16xm4_t merge*, *float16xm4_t a*, float16_t *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *float16xm8_t* **vfmaxvf_mask_float16xm8** (*float16xm8_t merge*, *float16xm8_t a*, float16_t *b*, *e16xm8_t mask*, unsigned int *gvl*)
- *float32xm1_t* **vfmaxvf_mask_float32xm1** (*float32xm1_t merge*, *float32xm1_t a*, float *b*, *e32xm1_t mask*, unsigned int *gvl*)

- *float32xm2_t* **vfmaxvf_mask_float32xm2** (*float32xm2_t merge*, *float32xm2_t a*, float *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfmaxvf_mask_float32xm4** (*float32xm4_t merge*, *float32xm4_t a*, float *b*, *e32xm4_t mask*, unsigned int *gvl*)
- *float32xm8_t* **vfmaxvf_mask_float32xm8** (*float32xm8_t merge*, *float32xm8_t a*, float *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *float64xm1_t* **vfmaxvf_mask_float64xm1** (*float64xm1_t merge*, *float64xm1_t a*, double *b*, *e64xm1_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vfmaxvf_mask_float64xm2** (*float64xm2_t merge*, *float64xm2_t a*, double *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *float64xm4_t* **vfmaxvf_mask_float64xm4** (*float64xm4_t merge*, *float64xm4_t a*, double *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *float64xm8_t* **vfmaxvf_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 maxmum

Instruction: ['vfmax.vv']

Prototypes:

- *float16xm1_t* **vfmaxvv_float16xm1** (*float16xm1_t a*, *float16xm1_t b*, unsigned int *gvl*)
- *float16xm2_t* **vfmaxvv_float16xm2** (*float16xm2_t a*, *float16xm2_t b*, unsigned int *gvl*)
- *float16xm4_t* **vfmaxvv_float16xm4** (*float16xm4_t a*, *float16xm4_t b*, unsigned int *gvl*)
- *float16xm8_t* **vfmaxvv_float16xm8** (*float16xm8_t a*, *float16xm8_t b*, unsigned int *gvl*)
- *float32xm1_t* **vfmaxvv_float32xm1** (*float32xm1_t a*, *float32xm1_t b*, unsigned int *gvl*)
- *float32xm2_t* **vfmaxvv_float32xm2** (*float32xm2_t a*, *float32xm2_t b*, unsigned int *gvl*)
- *float32xm4_t* **vfmaxvv_float32xm4** (*float32xm4_t a*, *float32xm4_t b*, unsigned int *gvl*)
- *float32xm8_t* **vfmaxvv_float32xm8** (*float32xm8_t a*, *float32xm8_t b*, unsigned int *gvl*)
- *float64xm1_t* **vfmaxvv_float64xm1** (*float64xm1_t a*, *float64xm1_t b*, unsigned int *gvl*)
- *float64xm2_t* **vfmaxvv_float64xm2** (*float64xm2_t a*, *float64xm2_t b*, unsigned int *gvl*)
- *float64xm4_t* **vfmaxvv_float64xm4** (*float64xm4_t a*, *float64xm4_t b*, unsigned int *gvl*)
- *float64xm8_t* **vfmaxvv_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* **vfmaxvv_mask_float16xm1** (*float16xm1_t merge*, *float16xm1_t a*, *float16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float16xm2_t* **vfmaxvv_mask_float16xm2** (*float16xm2_t merge*, *float16xm2_t a*, *float16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float16xm4_t* **vfmaxvv_mask_float16xm4** (*float16xm4_t merge*, *float16xm4_t a*, *float16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *float16xm8_t* **vfmaxvv_mask_float16xm8** (*float16xm8_t merge*, *float16xm8_t a*, *float16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)
- *float32xm1_t* **vfmaxvv_mask_float32xm1** (*float32xm1_t merge*, *float32xm1_t a*, *float32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *float32xm2_t* **vfmaxvv_mask_float32xm2** (*float32xm2_t merge*, *float32xm2_t a*, *float32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfmaxvv_mask_float32xm4** (*float32xm4_t merge*, *float32xm4_t a*, *float32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)
- *float32xm8_t* **vfmaxvv_mask_float32xm8** (*float32xm8_t merge*, *float32xm8_t a*, *float32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *float64xm1_t* **vfmaxvv_mask_float64xm1** (*float64xm1_t merge*, *float64xm1_t a*, *float64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vfmaxvv_mask_float64xm2** (*float64xm2_t merge*, *float64xm2_t a*, *float64xm2_t b*, *e64xm2_t mask*, unsigned int *gvl*)
- *float64xm4_t* **vfmaxvv_mask_float64xm4** (*float64xm4_t merge*, *float64xm4_t a*, *float64xm4_t b*, *e64xm4_t mask*, unsigned int *gvl*)
- *float64xm8_t* **vfmaxvv_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: ['vfmin.vf']

Prototypes:

- *float16xm1_t* **vfminvf_float16xm1** (*float16xm1_t a*, float16_t *b*, unsigned int *gvl*)
- *float16xm2_t* **vfminvf_float16xm2** (*float16xm2_t a*, float16_t *b*, unsigned int *gvl*)
- *float16xm4_t* **vfminvf_float16xm4** (*float16xm4_t a*, float16_t *b*, unsigned int *gvl*)
- *float16xm8_t* **vfminvf_float16xm8** (*float16xm8_t a*, float16_t *b*, unsigned int *gvl*)
- *float32xm1_t* **vfminvf_float32xm1** (*float32xm1_t a*, float *b*, unsigned int *gvl*)
- *float32xm2_t* **vfminvf_float32xm2** (*float32xm2_t a*, float *b*, unsigned int *gvl*)
- *float32xm4_t* **vfminvf_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: ['vfmin.vv']

Prototypes:

- *float16xm1_t* **vfminvv_float16xm1** (*float16xm1_t a*, *float16xm1_t b*, unsigned int *gvl*)
- *float16xm2_t* **vfminvv_float16xm2** (*float16xm2_t a*, *float16xm2_t b*, unsigned int *gvl*)
- *float16xm4_t* **vfminvv_float16xm4** (*float16xm4_t a*, *float16xm4_t b*, unsigned int *gvl*)
- *float16xm8_t* **vfminvv_float16xm8** (*float16xm8_t a*, *float16xm8_t b*, unsigned int *gvl*)
- *float32xm1_t* **vfminvv_float32xm1** (*float32xm1_t a*, *float32xm1_t b*, unsigned int *gvl*)
- *float32xm2_t* **vfminvv_float32xm2** (*float32xm2_t a*, *float32xm2_t b*, unsigned int *gvl*)
- *float32xm4_t* **vfminvv_float32xm4** (*float32xm4_t a*, *float32xm4_t b*, unsigned int *gvl*)
- *float32xm8_t* **vfminvv_float32xm8** (*float32xm8_t a*, *float32xm8_t b*, unsigned int *gvl*)
- *float64xm1_t* **vfminvv_float64xm1** (*float64xm1_t a*, *float64xm1_t b*, unsigned int *gvl*)
- *float64xm2_t* **vfminvv_float64xm2** (*float64xm2_t a*, *float64xm2_t b*, unsigned int *gvl*)
- *float64xm4_t* **vfminvv_float64xm4** (*float64xm4_t a*, *float64xm4_t b*, unsigned int *gvl*)
- *float64xm8_t* **vfminvv_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* **vfminvv_mask_float16xm1** (*float16xm1_t merge*, *float16xm1_t a*, *float16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float16xm2_t* **vfminvv_mask_float16xm2** (*float16xm2_t merge*, *float16xm2_t a*, *float16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float16xm4_t* **vfminvv_mask_float16xm4** (*float16xm4_t merge*, *float16xm4_t a*, *float16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *float16xm8_t* **vfminvv_mask_float16xm8** (*float16xm8_t merge*, *float16xm8_t a*, *float16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)
- *float32xm1_t* **vfminvv_mask_float32xm1** (*float32xm1_t merge*, *float32xm1_t a*, *float32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *float32xm2_t* **vfminvv_mask_float32xm2** (*float32xm2_t merge*, *float32xm2_t a*, *float32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfminvv_mask_float32xm4** (*float32xm4_t merge*, *float32xm4_t a*, *float32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)
- *float32xm8_t* **vfminvv_mask_float32xm8** (*float32xm8_t merge*, *float32xm8_t a*, *float32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *float64xm1_t* **vfminvv_mask_float64xm1** (*float64xm1_t merge*, *float64xm1_t a*, *float64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vfminvv_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 maxmum 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 minmum 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* **vfredsumvs_mask_float16xm1** (*float16xm1_t a*, *float16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float16xm2_t* **vfredsumvs_mask_float16xm2** (*float16xm2_t a*, *float16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float16xm4_t* **vfredsumvs_mask_float16xm4** (*float16xm4_t a*, *float16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *float16xm8_t* **vfredsumvs_mask_float16xm8** (*float16xm8_t a*, *float16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)
- *float32xm1_t* **vfredsumvs_mask_float32xm1** (*float32xm1_t a*, *float32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *float32xm2_t* **vfredsumvs_mask_float32xm2** (*float32xm2_t a*, *float32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfredsumvs_mask_float32xm4** (*float32xm4_t a*, *float32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)
- *float32xm8_t* **vfredsumvs_mask_float32xm8** (*float32xm8_t a*, *float32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *float64xm1_t* **vfredsumvs_mask_float64xm1** (*float64xm1_t a*, *float64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vfredsumvs_mask_float64xm2** (*float64xm2_t a*, *float64xm2_t b*, *e64xm2_t mask*, unsigned int *gvl*)
- *float64xm4_t* **vfredsumvs_mask_float64xm4** (*float64xm4_t a*, *float64xm4_t b*, *e64xm4_t mask*, unsigned int *gvl*)
- *float64xm8_t* **vfredsumvs_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] = fsignj(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] = fsignj(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] = fsignj(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: ['vfsub.vf']

Prototypes:

- *float16xm1_t* **vfsubvf_float16xm1** (*float16xm1_t a*, float16_t *b*, unsigned int *gvl*)
- *float16xm2_t* **vfsubvf_float16xm2** (*float16xm2_t a*, float16_t *b*, unsigned int *gvl*)
- *float16xm4_t* **vfsubvf_float16xm4** (*float16xm4_t a*, float16_t *b*, unsigned int *gvl*)
- *float16xm8_t* **vfsubvf_float16xm8** (*float16xm8_t a*, float16_t *b*, unsigned int *gvl*)
- *float32xm1_t* **vfsubvf_float32xm1** (*float32xm1_t a*, float *b*, unsigned int *gvl*)
- *float32xm2_t* **vfsubvf_float32xm2** (*float32xm2_t a*, float *b*, unsigned int *gvl*)
- *float32xm4_t* **vfsubvf_float32xm4** (*float32xm4_t a*, float *b*, unsigned int *gvl*)
- *float32xm8_t* **vfsubvf_float32xm8** (*float32xm8_t a*, float *b*, unsigned int *gvl*)
- *float64xm1_t* **vfsubvf_float64xm1** (*float64xm1_t a*, double *b*, unsigned int *gvl*)
- *float64xm2_t* **vfsubvf_float64xm2** (*float64xm2_t a*, double *b*, unsigned int *gvl*)
- *float64xm4_t* **vfsubvf_float64xm4** (*float64xm4_t a*, double *b*, unsigned int *gvl*)
- *float64xm8_t* **vfsubvf_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* **vfsubvf_mask_float16xm1** (*float16xm1_t merge*, *float16xm1_t a*, float16_t *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float16xm2_t* **vfsubvf_mask_float16xm2** (*float16xm2_t merge*, *float16xm2_t a*, float16_t *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float16xm4_t* **vfsubvf_mask_float16xm4** (*float16xm4_t merge*, *float16xm4_t a*, float16_t *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *float16xm8_t* **vfsubvf_mask_float16xm8** (*float16xm8_t merge*, *float16xm8_t a*, float16_t *b*, *e16xm8_t mask*, unsigned int *gvl*)
- *float32xm1_t* **vfsubvf_mask_float32xm1** (*float32xm1_t merge*, *float32xm1_t a*, float *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *float32xm2_t* **vfsubvf_mask_float32xm2** (*float32xm2_t merge*, *float32xm2_t a*, float *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfsubvf_mask_float32xm4** (*float32xm4_t merge*, *float32xm4_t a*, float *b*, *e32xm4_t mask*, unsigned int *gvl*)
- *float32xm8_t* **vfsubvf_mask_float32xm8** (*float32xm8_t merge*, *float32xm8_t a*, float *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *float64xm1_t* **vfsubvf_mask_float64xm1** (*float64xm1_t merge*, *float64xm1_t a*, double *b*, *e64xm1_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vfsubvf_mask_float64xm2** (*float64xm2_t merge*, *float64xm2_t a*, double *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *float64xm4_t* **vfsubvf_mask_float64xm4** (*float64xm4_t merge*, *float64xm4_t a*, double *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *float64xm8_t* **vfsubvf_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: ['vfsub.vv']

Prototypes:

- *float16xm1_t* **vfsubvv_float16xm1** (*float16xm1_t a*, *float16xm1_t b*, unsigned int *gvl*)
- *float16xm2_t* **vfsubvv_float16xm2** (*float16xm2_t a*, *float16xm2_t b*, unsigned int *gvl*)
- *float16xm4_t* **vfsubvv_float16xm4** (*float16xm4_t a*, *float16xm4_t b*, unsigned int *gvl*)
- *float16xm8_t* **vfsubvv_float16xm8** (*float16xm8_t a*, *float16xm8_t b*, unsigned int *gvl*)
- *float32xm1_t* **vfsubvv_float32xm1** (*float32xm1_t a*, *float32xm1_t b*, unsigned int *gvl*)
- *float32xm2_t* **vfsubvv_float32xm2** (*float32xm2_t a*, *float32xm2_t b*, unsigned int *gvl*)
- *float32xm4_t* **vfsubvv_float32xm4** (*float32xm4_t a*, *float32xm4_t b*, unsigned int *gvl*)
- *float32xm8_t* **vfsubvv_float32xm8** (*float32xm8_t a*, *float32xm8_t b*, unsigned int *gvl*)
- *float64xm1_t* **vfsubvv_float64xm1** (*float64xm1_t a*, *float64xm1_t b*, unsigned int *gvl*)
- *float64xm2_t* **vfsubvv_float64xm2** (*float64xm2_t a*, *float64xm2_t b*, unsigned int *gvl*)
- *float64xm4_t* **vfsubvv_float64xm4** (*float64xm4_t a*, *float64xm4_t b*, unsigned int *gvl*)
- *float64xm8_t* **vfsubvv_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* **vfsubvv_mask_float16xm1** (*float16xm1_t merge*, *float16xm1_t a*, *float16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float16xm2_t* **vfsubvv_mask_float16xm2** (*float16xm2_t merge*, *float16xm2_t a*, *float16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float16xm4_t* **vfsubvv_mask_float16xm4** (*float16xm4_t merge*, *float16xm4_t a*, *float16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *float16xm8_t* **vfsubvv_mask_float16xm8** (*float16xm8_t merge*, *float16xm8_t a*, *float16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)
- *float32xm1_t* **vfsubvv_mask_float32xm1** (*float32xm1_t merge*, *float32xm1_t a*, *float32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *float32xm2_t* **vfsubvv_mask_float32xm2** (*float32xm2_t merge*, *float32xm2_t a*, *float32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfsubvv_mask_float32xm4** (*float32xm4_t merge*, *float32xm4_t a*, *float32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)

- *float32xm8_t* **vfsubvv_mask_float32xm8** (*float32xm8_t merge*, *float32xm8_t a*, *float32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *float64xm1_t* **vfsubvv_mask_float64xm1** (*float64xm1_t merge*, *float64xm1_t a*, *float64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vfsubvv_mask_float64xm2** (*float64xm2_t merge*, *float64xm2_t a*, *float64xm2_t b*, *e64xm2_t mask*, unsigned int *gvl*)
- *float64xm4_t* **vfsubvv_mask_float64xm4** (*float64xm4_t merge*, *float64xm4_t a*, *float64xm4_t b*, *e64xm4_t mask*, unsigned int *gvl*)
- *float64xm8_t* **vfsubvv_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: ['vfwmacc.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: ['vfwmacc.vv']

Prototypes:

- *float32xm2_t* **vfwmaccvv_float32xm2_float16xm1** (*float32xm2_t a*, *float16xm1_t b*, *float16xm1_t c*, unsigned int *gvl*)
- *float32xm4_t* **vfwmaccvv_float32xm4_float16xm2** (*float32xm4_t a*, *float16xm2_t b*, *float16xm2_t c*, unsigned int *gvl*)
- *float32xm8_t* **vfwmaccvv_float32xm8_float16xm4** (*float32xm8_t a*, *float16xm4_t b*, *float16xm4_t c*, unsigned int *gvl*)
- *float64xm2_t* **vfwmaccvv_float64xm2_float32xm1** (*float64xm2_t a*, *float32xm1_t b*, *float32xm1_t c*, unsigned int *gvl*)
- *float64xm4_t* **vfwmaccvv_float64xm4_float32xm2** (*float64xm4_t a*, *float32xm2_t b*, *float32xm2_t c*, unsigned int *gvl*)
- *float64xm8_t* **vfwmaccvv_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* **vfwmaccvv_mask_float32xm2_float16xm1** (*float32xm2_t merge*, *float32xm2_t a*, *float16xm1_t b*, *float16xm1_t c*, *e16xm1_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfwmaccvv_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: ['vfwnmsac.vv']

Prototypes:

- *float32xm2_t* **vfwnmsacvv_float32xm2_float16xm1** (*float32xm2_t* *a*, *float16xm1_t* *b*, *float16xm1_t* *c*, unsigned int *gvl*)
- *float32xm4_t* **vfwnmsacvv_float32xm4_float16xm2** (*float32xm4_t* *a*, *float16xm2_t* *b*, *float16xm2_t* *c*, unsigned int *gvl*)
- *float32xm8_t* **vfwnmsacvv_float32xm8_float16xm4** (*float32xm8_t* *a*, *float16xm4_t* *b*, *float16xm4_t* *c*, unsigned int *gvl*)
- *float64xm2_t* **vfwnmsacvv_float64xm2_float32xm1** (*float64xm2_t* *a*, *float32xm1_t* *b*, *float32xm1_t* *c*, unsigned int *gvl*)
- *float64xm4_t* **vfwnmsacvv_float64xm4_float32xm2** (*float64xm4_t* *a*, *float32xm2_t* *b*, *float32xm2_t* *c*, unsigned int *gvl*)
- *float64xm8_t* **vfwnmsacvv_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* **vfwnmsacvv_mask_float32xm2_float16xm1** (*float32xm2_t* *merge*, *float32xm2_t* *a*, *float16xm1_t* *b*, *float16xm1_t* *c*, *e16xm1_t* *mask*, unsigned int *gvl*)
- *float32xm4_t* **vfwnmsacvv_mask_float32xm4_float16xm2** (*float32xm4_t* *merge*, *float32xm4_t* *a*, *float16xm2_t* *b*, *float16xm2_t* *c*, *e16xm2_t* *mask*, unsigned int *gvl*)
- *float32xm8_t* **vfwnmsacvv_mask_float32xm8_float16xm4** (*float32xm8_t* *merge*, *float32xm8_t* *a*, *float16xm4_t* *b*, *float16xm4_t* *c*, *e16xm4_t* *mask*, unsigned int *gvl*)
- *float64xm2_t* **vfwnmsacvv_mask_float64xm2_float32xm1** (*float64xm2_t* *merge*, *float64xm2_t* *a*, *float32xm1_t* *b*, *float32xm1_t* *c*, *e32xm1_t* *mask*, unsigned int *gvl*)
- *float64xm4_t* **vfwnmsacvv_mask_float64xm4_float32xm2** (*float64xm4_t* *merge*, *float64xm4_t* *a*, *float32xm2_t* *b*, *float32xm2_t* *c*, *e32xm2_t* *mask*, unsigned int *gvl*)
- *float64xm8_t* **vfwnmsacvv_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 odered 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: ['vfwsub.vf']

Prototypes:

- *float32xm2_t* **vfwsubvf_float32xm2_float16xm1** (*float16xm1_t a*, float16_t *b*, unsigned int *gvl*)
- *float32xm4_t* **vfwsubvf_float32xm4_float16xm2** (*float16xm2_t a*, float16_t *b*, unsigned int *gvl*)
- *float32xm8_t* **vfwsubvf_float32xm8_float16xm4** (*float16xm4_t a*, float16_t *b*, unsigned int *gvl*)
- *float64xm2_t* **vfwsubvf_float64xm2_float32xm1** (*float32xm1_t a*, float *b*, unsigned int *gvl*)
- *float64xm4_t* **vfwsubvf_float64xm4_float32xm2** (*float32xm2_t a*, float *b*, unsigned int *gvl*)
- *float64xm8_t* **vfwsubvf_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* **vfwsubvf_mask_float32xm2_float16xm1** (*float32xm2_t merge*, *float16xm1_t a*, float16_t *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfwsubvf_mask_float32xm4_float16xm2** (*float32xm4_t merge*, *float16xm2_t a*, float16_t *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float32xm8_t* **vfwsubvf_mask_float32xm8_float16xm4** (*float32xm8_t merge*, *float16xm4_t a*, float16_t *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vfwsubvf_mask_float64xm2_float32xm1** (*float64xm2_t merge*, *float32xm1_t a*, float *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *float64xm4_t* **vfwsubvf_mask_float64xm4_float32xm2** (*float64xm4_t merge*, *float32xm2_t a*, float *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float64xm8_t* **vfwsubvf_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: ['vfwsub.vv']

Prototypes:

- *float32xm2_t* **vfwsubvv_float32xm2_float16xm1** (*float16xm1_t a*, *float16xm1_t b*, unsigned int *gvl*)
- *float32xm4_t* **vfwsubvv_float32xm4_float16xm2** (*float16xm2_t a*, *float16xm2_t b*, unsigned int *gvl*)
- *float32xm8_t* **vfwsubvv_float32xm8_float16xm4** (*float16xm4_t a*, *float16xm4_t b*, unsigned int *gvl*)
- *float64xm2_t* **vfwsubvv_float64xm2_float32xm1** (*float32xm1_t a*, *float32xm1_t b*, unsigned int *gvl*)
- *float64xm4_t* **vfwsubvv_float64xm4_float32xm2** (*float32xm2_t a*, *float32xm2_t b*, unsigned int *gvl*)
- *float64xm8_t* **vfwsubvv_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* **vfwsubvv_mask_float32xm2_float16xm1** (*float32xm2_t merge*, *float16xm1_t a*, *float16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfwsubvv_mask_float32xm4_float16xm2** (*float32xm4_t merge*, *float16xm2_t a*, *float16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float32xm8_t* **vfwsubvv_mask_float32xm8_float16xm4** (*float32xm8_t merge*, *float16xm4_t a*, *float16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vfwsubvv_mask_float64xm2_float32xm1** (*float64xm2_t merge*, *float32xm1_t a*, *float32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *float64xm4_t* **vfwsubvv_mask_float64xm4_float32xm2** (*float64xm4_t merge*, *float32xm2_t a*, *float32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float64xm8_t* **vfwsubvv_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: ['vfwsub.wf']

Prototypes:

- *float32xm2_t* **vfwsubwf_float32xm2** (*float32xm2_t a*, float16_t *b*, unsigned int *gvl*)
- *float32xm4_t* **vfwsubwf_float32xm4** (*float32xm4_t a*, float16_t *b*, unsigned int *gvl*)
- *float32xm8_t* **vfwsubwf_float32xm8** (*float32xm8_t a*, float16_t *b*, unsigned int *gvl*)
- *float64xm2_t* **vfwsubwf_float64xm2** (*float64xm2_t a*, float *b*, unsigned int *gvl*)
- *float64xm4_t* **vfwsubwf_float64xm4** (*float64xm4_t a*, float *b*, unsigned int *gvl*)
- *float64xm8_t* **vfwsubwf_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* **vfwsubwf_mask_float32xm2** (*float32xm2_t merge*, *float32xm2_t a*, float16_t *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfwsubwf_mask_float32xm4** (*float32xm4_t merge*, *float32xm4_t a*, float16_t *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float32xm8_t* **vfwsubwf_mask_float32xm8** (*float32xm8_t merge*, *float32xm8_t a*, float16_t *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vfwsubwf_mask_float64xm2** (*float64xm2_t merge*, *float64xm2_t a*, float *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *float64xm4_t* **vfwsubwf_mask_float64xm4** (*float64xm4_t merge*, *float64xm4_t a*, float *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float64xm8_t* **vfwsubwf_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: ['vfwsub.wv']

Prototypes:

- *float32xm2_t* **vfwsubwv_float32xm2_float16xm1** (*float32xm2_t a*, *float16xm1_t b*, unsigned int *gvl*)
- *float32xm4_t* **vfwsubwv_float32xm4_float16xm2** (*float32xm4_t a*, *float16xm2_t b*, unsigned int *gvl*)
- *float32xm8_t* **vfwsubwv_float32xm8_float16xm4** (*float32xm8_t a*, *float16xm4_t b*, unsigned int *gvl*)
- *float64xm2_t* **vfwsubwv_float64xm2_float32xm1** (*float64xm2_t a*, *float32xm1_t b*, unsigned int *gvl*)
- *float64xm4_t* **vfwsubwv_float64xm4_float32xm2** (*float64xm4_t a*, *float32xm2_t b*, unsigned int *gvl*)
- *float64xm8_t* **vfwsubwv_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* **vfwsubwv_mask_float32xm2_float16xm1** (*float32xm2_t merge*, *float32xm2_t a*, *float16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vfwsubwv_mask_float32xm4_float16xm2** (*float32xm4_t merge*, *float32xm4_t a*, *float16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float32xm8_t* **vfwsubwv_mask_float32xm8_float16xm4** (*float32xm8_t merge*, *float32xm8_t a*, *float16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vfwsubwv_mask_float64xm2_float32xm1** (*float64xm2_t merge*, *float64xm2_t a*, *float32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *float64xm4_t* **vfwsubwv_mask_float64xm4_float32xm2** (*float64xm4_t merge*, *float64xm4_t a*, *float32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float64xm8_t* **vfwsubwv_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: ['vmfeq.vf']

Prototypes:

- *e16xm1_t* **vmfeqvf_e16xm1_float16xm1** (*float16xm1_t a*, float16_t *b*, unsigned int *gvl*)
- *e16xm2_t* **vmfeqvf_e16xm2_float16xm2** (*float16xm2_t a*, float16_t *b*, unsigned int *gvl*)
- *e16xm4_t* **vmfeqvf_e16xm4_float16xm4** (*float16xm4_t a*, float16_t *b*, unsigned int *gvl*)
- *e16xm8_t* **vmfeqvf_e16xm8_float16xm8** (*float16xm8_t a*, float16_t *b*, unsigned int *gvl*)
- *e32xm1_t* **vmfeqvf_e32xm1_float32xm1** (*float32xm1_t a*, float *b*, unsigned int *gvl*)
- *e32xm2_t* **vmfeqvf_e32xm2_float32xm2** (*float32xm2_t a*, float *b*, unsigned int *gvl*)
- *e32xm4_t* **vmfeqvf_e32xm4_float32xm4** (*float32xm4_t a*, float *b*, unsigned int *gvl*)
- *e32xm8_t* **vmfeqvf_e32xm8_float32xm8** (*float32xm8_t a*, float *b*, unsigned int *gvl*)
- *e64xm1_t* **vmfeqvf_e64xm1_float64xm1** (*float64xm1_t a*, double *b*, unsigned int *gvl*)
- *e64xm2_t* **vmfeqvf_e64xm2_float64xm2** (*float64xm2_t a*, double *b*, unsigned int *gvl*)
- *e64xm4_t* **vmfeqvf_e64xm4_float64xm4** (*float64xm4_t a*, double *b*, unsigned int *gvl*)
- *e64xm8_t* **vmfeqvf_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* **vmfeqvf_mask_e16xm1_float16xm1** (*e16xm1_t merge*, *float16xm1_t a*, float16_t *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *e16xm2_t* **vmfeqvf_mask_e16xm2_float16xm2** (*e16xm2_t merge*, *float16xm2_t a*, float16_t *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *e16xm4_t* **vmfeqvf_mask_e16xm4_float16xm4** (*e16xm4_t merge*, *float16xm4_t a*, float16_t *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *e16xm8_t* **vmfeqvf_mask_e16xm8_float16xm8** (*e16xm8_t merge*, *float16xm8_t a*, float16_t *b*, *e16xm8_t mask*, unsigned int *gvl*)
- *e32xm1_t* **vmfeqvf_mask_e32xm1_float32xm1** (*e32xm1_t merge*, *float32xm1_t a*, float *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *e32xm2_t* **vmfeqvf_mask_e32xm2_float32xm2** (*e32xm2_t merge*, *float32xm2_t a*, float *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *e32xm4_t* **vmfeqvf_mask_e32xm4_float32xm4** (*e32xm4_t merge*, *float32xm4_t a*, float *b*, *e32xm4_t mask*, unsigned int *gvl*)
- *e32xm8_t* **vmfeqvf_mask_e32xm8_float32xm8** (*e32xm8_t merge*, *float32xm8_t a*, float *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *e64xm1_t* **vmfeqvf_mask_e64xm1_float64xm1** (*e64xm1_t merge*, *float64xm1_t a*, double *b*, *e64xm1_t mask*, unsigned int *gvl*)

- *e64xm2_t* **vmfeqvf_mask_e64xm2_float64xm2** (*e64xm2_t merge*, *float64xm2_t a*, double *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *e64xm4_t* **vmfeqvf_mask_e64xm4_float64xm4** (*e64xm4_t merge*, *float64xm4_t a*, double *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *e64xm8_t* **vmfeqvf_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: ['vmfeq.vv']

Prototypes:

- *e16xm1_t* **vmfeqvv_e16xm1_float16xm1** (*float16xm1_t a*, *float16xm1_t b*, unsigned int *gvl*)
- *e16xm2_t* **vmfeqvv_e16xm2_float16xm2** (*float16xm2_t a*, *float16xm2_t b*, unsigned int *gvl*)
- *e16xm4_t* **vmfeqvv_e16xm4_float16xm4** (*float16xm4_t a*, *float16xm4_t b*, unsigned int *gvl*)
- *e16xm8_t* **vmfeqvv_e16xm8_float16xm8** (*float16xm8_t a*, *float16xm8_t b*, unsigned int *gvl*)
- *e32xm1_t* **vmfeqvv_e32xm1_float32xm1** (*float32xm1_t a*, *float32xm1_t b*, unsigned int *gvl*)
- *e32xm2_t* **vmfeqvv_e32xm2_float32xm2** (*float32xm2_t a*, *float32xm2_t b*, unsigned int *gvl*)
- *e32xm4_t* **vmfeqvv_e32xm4_float32xm4** (*float32xm4_t a*, *float32xm4_t b*, unsigned int *gvl*)
- *e32xm8_t* **vmfeqvv_e32xm8_float32xm8** (*float32xm8_t a*, *float32xm8_t b*, unsigned int *gvl*)
- *e64xm1_t* **vmfeqvv_e64xm1_float64xm1** (*float64xm1_t a*, *float64xm1_t b*, unsigned int *gvl*)
- *e64xm2_t* **vmfeqvv_e64xm2_float64xm2** (*float64xm2_t a*, *float64xm2_t b*, unsigned int *gvl*)
- *e64xm4_t* **vmfeqvv_e64xm4_float64xm4** (*float64xm4_t a*, *float64xm4_t b*, unsigned int *gvl*)
- *e64xm8_t* **vmfeqvv_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* **vmfeqvv_mask_e16xm1_float16xm1** (*e16xm1_t merge*, *float16xm1_t a*, *float16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *e16xm2_t* **vmfeqvv_mask_e16xm2_float16xm2** (*e16xm2_t merge*, *float16xm2_t a*, *float16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)

- *e16xm4_t* **vmfeqvv_mask_e16xm4_float16xm4** (*e16xm4_t merge*, *float16xm4_t a*, *float16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *e16xm8_t* **vmfeqvv_mask_e16xm8_float16xm8** (*e16xm8_t merge*, *float16xm8_t a*, *float16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)
- *e32xm1_t* **vmfeqvv_mask_e32xm1_float32xm1** (*e32xm1_t merge*, *float32xm1_t a*, *float32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *e32xm2_t* **vmfeqvv_mask_e32xm2_float32xm2** (*e32xm2_t merge*, *float32xm2_t a*, *float32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *e32xm4_t* **vmfeqvv_mask_e32xm4_float32xm4** (*e32xm4_t merge*, *float32xm4_t a*, *float32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)
- *e32xm8_t* **vmfeqvv_mask_e32xm8_float32xm8** (*e32xm8_t merge*, *float32xm8_t a*, *float32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *e64xm1_t* **vmfeqvv_mask_e64xm1_float64xm1** (*e64xm1_t merge*, *float64xm1_t a*, *float64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)
- *e64xm2_t* **vmfeqvv_mask_e64xm2_float64xm2** (*e64xm2_t merge*, *float64xm2_t a*, *float64xm2_t b*, *e64xm2_t mask*, unsigned int *gvl*)
- *e64xm4_t* **vmfeqvv_mask_e64xm4_float64xm4** (*e64xm4_t merge*, *float64xm4_t a*, *float64xm4_t b*, *e64xm4_t mask*, unsigned int *gvl*)
- *e64xm8_t* **vmfeqvv_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
    reuslt[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* **vmfgtvf_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* **vmfgtvf_mask_e16xm1_float16xm1** (*e16xm1_t merge*, *float16xm1_t a*, float16_t *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *e16xm2_t* **vmfgtvf_mask_e16xm2_float16xm2** (*e16xm2_t merge*, *float16xm2_t a*, float16_t *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *e16xm4_t* **vmfgtvf_mask_e16xm4_float16xm4** (*e16xm4_t merge*, *float16xm4_t a*, float16_t *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *e16xm8_t* **vmfgtvf_mask_e16xm8_float16xm8** (*e16xm8_t merge*, *float16xm8_t a*, float16_t *b*, *e16xm8_t mask*, unsigned int *gvl*)
- *e32xm1_t* **vmfgtvf_mask_e32xm1_float32xm1** (*e32xm1_t merge*, *float32xm1_t a*, float *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *e32xm2_t* **vmfgtvf_mask_e32xm2_float32xm2** (*e32xm2_t merge*, *float32xm2_t a*, float *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *e32xm4_t* **vmfgtvf_mask_e32xm4_float32xm4** (*e32xm4_t merge*, *float32xm4_t a*, float *b*, *e32xm4_t mask*, unsigned int *gvl*)
- *e32xm8_t* **vmfgtvf_mask_e32xm8_float32xm8** (*e32xm8_t merge*, *float32xm8_t a*, float *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *e64xm1_t* **vmfgtvf_mask_e64xm1_float64xm1** (*e64xm1_t merge*, *float64xm1_t a*, double *b*, *e64xm1_t mask*, unsigned int *gvl*)
- *e64xm2_t* **vmfgtvf_mask_e64xm2_float64xm2** (*e64xm2_t merge*, *float64xm2_t a*, double *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *e64xm4_t* **vmfgtvf_mask_e64xm4_float64xm4** (*e64xm4_t merge*, *float64xm4_t a*, double *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *e64xm8_t* **vmfgtvf_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* **vmfgtvv_e16xm1_float16xm1** (*float16xm1_t a*, *float16xm1_t b*, unsigned int *gvl*)
- *e16xm2_t* **vmfgtvv_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
    reuslt[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
    reuslt[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* **vmfltvf_e16xm1_float16xm1** (*float16xm1_t a*, float16_t *b*, unsigned int *gvl*)
- *e16xm2_t* **vmfltvf_e16xm2_float16xm2** (*float16xm2_t a*, float16_t *b*, unsigned int *gvl*)
- *e16xm4_t* **vmfltvf_e16xm4_float16xm4** (*float16xm4_t a*, float16_t *b*, unsigned int *gvl*)
- *e16xm8_t* **vmfltvf_e16xm8_float16xm8** (*float16xm8_t a*, float16_t *b*, unsigned int *gvl*)
- *e32xm1_t* **vmfltvf_e32xm1_float32xm1** (*float32xm1_t a*, float *b*, unsigned int *gvl*)
- *e32xm2_t* **vmfltvf_e32xm2_float32xm2** (*float32xm2_t a*, float *b*, unsigned int *gvl*)
- *e32xm4_t* **vmfltvf_e32xm4_float32xm4** (*float32xm4_t a*, float *b*, unsigned int *gvl*)
- *e32xm8_t* **vmfltvf_e32xm8_float32xm8** (*float32xm8_t a*, float *b*, unsigned int *gvl*)
- *e64xm1_t* **vmfltvf_e64xm1_float64xm1** (*float64xm1_t a*, double *b*, unsigned int *gvl*)
- *e64xm2_t* **vmfltvf_e64xm2_float64xm2** (*float64xm2_t a*, double *b*, unsigned int *gvl*)
- *e64xm4_t* **vmfltvf_e64xm4_float64xm4** (*float64xm4_t a*, double *b*, unsigned int *gvl*)
- *e64xm8_t* **vmfltvf_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* **vmfltvf_mask_e16xm1_float16xm1** (*e16xm1_t merge*, *float16xm1_t a*, float16_t *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *e16xm2_t* **vmfltvf_mask_e16xm2_float16xm2** (*e16xm2_t merge*, *float16xm2_t a*, float16_t *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *e16xm4_t* **vmfltvf_mask_e16xm4_float16xm4** (*e16xm4_t merge*, *float16xm4_t a*, float16_t *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *e16xm8_t* **vmfltvf_mask_e16xm8_float16xm8** (*e16xm8_t merge*, *float16xm8_t a*, float16_t *b*, *e16xm8_t mask*, unsigned int *gvl*)
- *e32xm1_t* **vmfltvf_mask_e32xm1_float32xm1** (*e32xm1_t merge*, *float32xm1_t a*, float *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *e32xm2_t* **vmfltvf_mask_e32xm2_float32xm2** (*e32xm2_t merge*, *float32xm2_t a*, float *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *e32xm4_t* **vmfltvf_mask_e32xm4_float32xm4** (*e32xm4_t merge*, *float32xm4_t a*, float *b*, *e32xm4_t mask*, unsigned int *gvl*)

- *e32xm8_t* **vmfltvf_mask_e32xm8_float32xm8** (*e32xm8_t merge*, *float32xm8_t a*, float *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *e64xm1_t* **vmfltvf_mask_e64xm1_float64xm1** (*e64xm1_t merge*, *float64xm1_t a*, double *b*, *e64xm1_t mask*, unsigned int *gvl*)
- *e64xm2_t* **vmfltvf_mask_e64xm2_float64xm2** (*e64xm2_t merge*, *float64xm2_t a*, double *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *e64xm4_t* **vmfltvf_mask_e64xm4_float64xm4** (*e64xm4_t merge*, *float64xm4_t a*, double *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *e64xm8_t* **vmfltvf_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* **vmfltvv_e16xm1_float16xm1** (*float16xm1_t a*, *float16xm1_t b*, unsigned int *gvl*)
- *e16xm2_t* **vmfltvv_e16xm2_float16xm2** (*float16xm2_t a*, *float16xm2_t b*, unsigned int *gvl*)
- *e16xm4_t* **vmfltvv_e16xm4_float16xm4** (*float16xm4_t a*, *float16xm4_t b*, unsigned int *gvl*)
- *e16xm8_t* **vmfltvv_e16xm8_float16xm8** (*float16xm8_t a*, *float16xm8_t b*, unsigned int *gvl*)
- *e32xm1_t* **vmfltvv_e32xm1_float32xm1** (*float32xm1_t a*, *float32xm1_t b*, unsigned int *gvl*)
- *e32xm2_t* **vmfltvv_e32xm2_float32xm2** (*float32xm2_t a*, *float32xm2_t b*, unsigned int *gvl*)
- *e32xm4_t* **vmfltvv_e32xm4_float32xm4** (*float32xm4_t a*, *float32xm4_t b*, unsigned int *gvl*)
- *e32xm8_t* **vmfltvv_e32xm8_float32xm8** (*float32xm8_t a*, *float32xm8_t b*, unsigned int *gvl*)
- *e64xm1_t* **vmfltvv_e64xm1_float64xm1** (*float64xm1_t a*, *float64xm1_t b*, unsigned int *gvl*)
- *e64xm2_t* **vmfltvv_e64xm2_float64xm2** (*float64xm2_t a*, *float64xm2_t b*, unsigned int *gvl*)
- *e64xm4_t* **vmfltvv_e64xm4_float64xm4** (*float64xm4_t a*, *float64xm4_t b*, unsigned int *gvl*)
- *e64xm8_t* **vmfltvv_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* **vmfltvv_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: ['vmfne.vf']

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
    reuslt[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
    reuslt[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: ['vmford.vf']

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
    reuslt[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 averge 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 averge 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 averge 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 addtion 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* **vadcvim_mask_uint64xm1** (*uint64xm1_t a*, const int *b*, *e64xm1_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vadcvim_mask_uint64xm2** (*uint64xm2_t a*, const int *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vadcvim_mask_uint64xm4** (*uint64xm4_t a*, const int *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vadcvim_mask_uint64xm8** (*uint64xm8_t a*, const int *b*, *e64xm8_t mask*, unsigned int *gvl*)
- *uint8xm1_t* **vadcvim_mask_uint8xm1** (*uint8xm1_t a*, const int *b*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint8xm2_t* **vadcvim_mask_uint8xm2** (*uint8xm2_t a*, const int *b*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint8xm4_t* **vadcvim_mask_uint8xm4** (*uint8xm4_t a*, const int *b*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint8xm8_t* **vadcvim_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 addtion with carry

Instruction: ['vadc.vvm']

Masked prototypes:

- *int16xm1_t* **vadcvvm_mask_int16xm1** (*int16xm1_t a*, *int16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *int16xm2_t* **vadcvvm_mask_int16xm2** (*int16xm2_t a*, *int16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vadcvvm_mask_int16xm4** (*int16xm4_t a*, *int16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vadcvvm_mask_int16xm8** (*int16xm8_t a*, *int16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)
- *int32xm1_t* **vadcvvm_mask_int32xm1** (*int32xm1_t a*, *int32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vadcvvm_mask_int32xm2** (*int32xm2_t a*, *int32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vadcvvm_mask_int32xm4** (*int32xm4_t a*, *int32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vadcvvm_mask_int32xm8** (*int32xm8_t a*, *int32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *int64xm1_t* **vadcvvm_mask_int64xm1** (*int64xm1_t a*, *int64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vadcvvm_mask_int64xm2** (*int64xm2_t a*, *int64xm2_t b*, *e64xm2_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vadcvvm_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 addtion 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* **vadcvxm_mask_uint8xm2** (*uint8xm2_t a*, unsigned char *b*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint8xm4_t* **vadcvxm_mask_uint8xm4** (*uint8xm4_t a*, unsigned char *b*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint8xm8_t* **vadcvxm_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 addtion

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 addtion

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 addtion

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 averge 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 averge 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 muiltiply-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 muiltiply-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 addtion 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* **vmadcvim_mask_e32xm8_uint32xm8** (*uint32xm8_t a*, const int *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *e64xm1_t* **vmadcvim_mask_e64xm1_int64xm1** (*int64xm1_t a*, const int *b*, *e64xm1_t mask*, unsigned int *gvl*)
- *e64xm1_t* **vmadcvim_mask_e64xm1_uint64xm1** (*uint64xm1_t a*, const int *b*, *e64xm1_t mask*, unsigned int *gvl*)
- *e64xm2_t* **vmadcvim_mask_e64xm2_int64xm2** (*int64xm2_t a*, const int *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *e64xm2_t* **vmadcvim_mask_e64xm2_uint64xm2** (*uint64xm2_t a*, const int *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *e64xm4_t* **vmadcvim_mask_e64xm4_int64xm4** (*int64xm4_t a*, const int *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *e64xm4_t* **vmadcvim_mask_e64xm4_uint64xm4** (*uint64xm4_t a*, const int *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *e64xm8_t* **vmadcvim_mask_e64xm8_int64xm8** (*int64xm8_t a*, const int *b*, *e64xm8_t mask*, unsigned int *gvl*)
- *e64xm8_t* **vmadcvim_mask_e64xm8_uint64xm8** (*uint64xm8_t a*, const int *b*, *e64xm8_t mask*, unsigned int *gvl*)
- *e8xm1_t* **vmadcvim_mask_e8xm1_int8xm1** (*int8xm1_t a*, const int *b*, *e8xm1_t mask*, unsigned int *gvl*)
- *e8xm1_t* **vmadcvim_mask_e8xm1_uint8xm1** (*uint8xm1_t a*, const int *b*, *e8xm1_t mask*, unsigned int *gvl*)
- *e8xm2_t* **vmadcvim_mask_e8xm2_int8xm2** (*int8xm2_t a*, const int *b*, *e8xm2_t mask*, unsigned int *gvl*)
- *e8xm2_t* **vmadcvim_mask_e8xm2_uint8xm2** (*uint8xm2_t a*, const int *b*, *e8xm2_t mask*, unsigned int *gvl*)
- *e8xm4_t* **vmadcvim_mask_e8xm4_int8xm4** (*int8xm4_t a*, const int *b*, *e8xm4_t mask*, unsigned int *gvl*)
- *e8xm4_t* **vmadcvim_mask_e8xm4_uint8xm4** (*uint8xm4_t a*, const int *b*, *e8xm4_t mask*, unsigned int *gvl*)
- *e8xm8_t* **vmadcvim_mask_e8xm8_int8xm8** (*int8xm8_t a*, const int *b*, *e8xm8_t mask*, unsigned int *gvl*)
- *e8xm8_t* **vmadcvim_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 addtion with carry in mask register format

Instruction: ['vmadc.vvm']

Masked prototypes:

- *e16xm1_t* **vmadcvvm_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 addtion 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 muiltiply-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 muiltiply-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 addtion with borrow in mask register format

Instruction: ['vmsbc.vvm']

Masked prototypes:

- *e16xm1_t* **vmsbcvvm_mask_e16xm1_int16xm1** (*int16xm1_t a*, *int16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *e16xm1_t* **vmsbcvvm_mask_e16xm1_uint16xm1** (*uint16xm1_t a*, *uint16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *e16xm2_t* **vmsbcvvm_mask_e16xm2_int16xm2** (*int16xm2_t a*, *int16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *e16xm2_t* **vmsbcvvm_mask_e16xm2_uint16xm2** (*uint16xm2_t a*, *uint16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *e16xm4_t* **vmsbcvvm_mask_e16xm4_int16xm4** (*int16xm4_t a*, *int16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *e16xm4_t* **vmsbcvvm_mask_e16xm4_uint16xm4** (*uint16xm4_t a*, *uint16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *e16xm8_t* **vmsbcvvm_mask_e16xm8_int16xm8** (*int16xm8_t a*, *int16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)
- *e16xm8_t* **vmsbcvvm_mask_e16xm8_uint16xm8** (*uint16xm8_t a*, *uint16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)

- *e32xm1_t* **vmsbcvvm_mask_e32xm1_int32xm1** (*int32xm1_t a*, *int32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *e32xm1_t* **vmsbcvvm_mask_e32xm1_uint32xm1** (*uint32xm1_t a*, *uint32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *e32xm2_t* **vmsbcvvm_mask_e32xm2_int32xm2** (*int32xm2_t a*, *int32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *e32xm2_t* **vmsbcvvm_mask_e32xm2_uint32xm2** (*uint32xm2_t a*, *uint32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *e32xm4_t* **vmsbcvvm_mask_e32xm4_int32xm4** (*int32xm4_t a*, *int32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)
- *e32xm4_t* **vmsbcvvm_mask_e32xm4_uint32xm4** (*uint32xm4_t a*, *uint32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)
- *e32xm8_t* **vmsbcvvm_mask_e32xm8_int32xm8** (*int32xm8_t a*, *int32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *e32xm8_t* **vmsbcvvm_mask_e32xm8_uint32xm8** (*uint32xm8_t a*, *uint32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *e64xm1_t* **vmsbcvvm_mask_e64xm1_int64xm1** (*int64xm1_t a*, *int64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)
- *e64xm1_t* **vmsbcvvm_mask_e64xm1_uint64xm1** (*uint64xm1_t a*, *uint64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)
- *e64xm2_t* **vmsbcvvm_mask_e64xm2_int64xm2** (*int64xm2_t a*, *int64xm2_t b*, *e64xm2_t mask*, unsigned int *gvl*)
- *e64xm2_t* **vmsbcvvm_mask_e64xm2_uint64xm2** (*uint64xm2_t a*, *uint64xm2_t b*, *e64xm2_t mask*, unsigned int *gvl*)
- *e64xm4_t* **vmsbcvvm_mask_e64xm4_int64xm4** (*int64xm4_t a*, *int64xm4_t b*, *e64xm4_t mask*, unsigned int *gvl*)
- *e64xm4_t* **vmsbcvvm_mask_e64xm4_uint64xm4** (*uint64xm4_t a*, *uint64xm4_t b*, *e64xm4_t mask*, unsigned int *gvl*)
- *e64xm8_t* **vmsbcvvm_mask_e64xm8_int64xm8** (*int64xm8_t a*, *int64xm8_t b*, *e64xm8_t mask*, unsigned int *gvl*)
- *e64xm8_t* **vmsbcvvm_mask_e64xm8_uint64xm8** (*uint64xm8_t a*, *uint64xm8_t b*, *e64xm8_t mask*, unsigned int *gvl*)
- *e8xm1_t* **vmsbcvvm_mask_e8xm1_int8xm1** (*int8xm1_t a*, *int8xm1_t b*, *e8xm1_t mask*, unsigned int *gvl*)
- *e8xm1_t* **vmsbcvvm_mask_e8xm1_uint8xm1** (*uint8xm1_t a*, *uint8xm1_t b*, *e8xm1_t mask*, unsigned int *gvl*)
- *e8xm2_t* **vmsbcvvm_mask_e8xm2_int8xm2** (*int8xm2_t a*, *int8xm2_t b*, *e8xm2_t mask*, unsigned int *gvl*)
- *e8xm2_t* **vmsbcvvm_mask_e8xm2_uint8xm2** (*uint8xm2_t a*, *uint8xm2_t b*, *e8xm2_t mask*, unsigned int *gvl*)
- *e8xm4_t* **vmsbcvvm_mask_e8xm4_int8xm4** (*int8xm4_t a*, *int8xm4_t b*, *e8xm4_t mask*, unsigned int *gvl*)
- *e8xm4_t* **vmsbcvvm_mask_e8xm4_uint8xm4** (*uint8xm4_t a*, *uint8xm4_t b*, *e8xm4_t mask*, unsigned int *gvl*)
- *e8xm8_t* **vmsbcvvm_mask_e8xm8_int8xm8** (*int8xm8_t a*, *int8xm8_t b*, *e8xm8_t mask*, unsigned int *gvl*)

- *e8xm8_t* **vmsbcvvm_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[elemet])
    result[gvl : VLMAX] = 0
```

2.7.33 Elementwise vector-scalar integer addtion with borrow in mask register format

Instruction: ['vmsbc.vxm']

Masked prototypes:

- *e16xm1_t* **vmsbcvxm_mask_e16xm1_int16xm1** (*int16xm1_t a*, short *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *e16xm1_t* **vmsbcvxm_mask_e16xm1_uint16xm1** (*uint16xm1_t a*, unsigned short *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *e16xm2_t* **vmsbcvxm_mask_e16xm2_int16xm2** (*int16xm2_t a*, short *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *e16xm2_t* **vmsbcvxm_mask_e16xm2_uint16xm2** (*uint16xm2_t a*, unsigned short *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *e16xm4_t* **vmsbcvxm_mask_e16xm4_int16xm4** (*int16xm4_t a*, short *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *e16xm4_t* **vmsbcvxm_mask_e16xm4_uint16xm4** (*uint16xm4_t a*, unsigned short *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *e16xm8_t* **vmsbcvxm_mask_e16xm8_int16xm8** (*int16xm8_t a*, short *b*, *e16xm8_t mask*, unsigned int *gvl*)
- *e16xm8_t* **vmsbcvxm_mask_e16xm8_uint16xm8** (*uint16xm8_t a*, unsigned short *b*, *e16xm8_t mask*, unsigned int *gvl*)
- *e32xm1_t* **vmsbcvxm_mask_e32xm1_int32xm1** (*int32xm1_t a*, int *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *e32xm1_t* **vmsbcvxm_mask_e32xm1_uint32xm1** (*uint32xm1_t a*, unsigned int *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *e32xm2_t* **vmsbcvxm_mask_e32xm2_int32xm2** (*int32xm2_t a*, int *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *e32xm2_t* **vmsbcvxm_mask_e32xm2_uint32xm2** (*uint32xm2_t a*, unsigned int *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *e32xm4_t* **vmsbcvxm_mask_e32xm4_int32xm4** (*int32xm4_t a*, int *b*, *e32xm4_t mask*, unsigned int *gvl*)
- *e32xm4_t* **vmsbcvxm_mask_e32xm4_uint32xm4** (*uint32xm4_t a*, unsigned int *b*, *e32xm4_t mask*, unsigned int *gvl*)
- *e32xm8_t* **vmsbcvxm_mask_e32xm8_int32xm8** (*int32xm8_t a*, int *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *e32xm8_t* **vmsbcvxm_mask_e32xm8_uint32xm8** (*uint32xm8_t a*, unsigned int *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *e64xm1_t* **vmsbcvxm_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: ['vnclipu.vi']

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*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint16xm2_t* **vnclipuvi_mask_uint16xm2_uint32xm4** (*uint16xm2_t merge*, *uint32xm4_t a*, const unsigned short *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vnclipuvi_mask_uint16xm4_uint32xm8** (*uint16xm4_t merge*, *uint32xm8_t a*, const unsigned short *b*, *e16xm4_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 muiltiply-subtarction, overwrite minuend

Instruction: ['vnmsac.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 muiltiply-subtarction, 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 muiltiply-subtarction, 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 muiltiply-subtarction, 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*, unsigned int *gvl*)
- *int8xm2_t* **vnsravv_mask_int8xm2_int16xm4_uint8xm2** (*int8xm2_t merge*, *int16xm4_t a*, *uint8xm2_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *int8xm4_t* **vnsravv_mask_int8xm4_int16xm8_uint8xm4** (*int8xm4_t merge*, *int16xm8_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 (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*, *e16xm1_t mask*, unsigned int *gvl*)
- *int16xm2_t* **vremvv_mask_int16xm2** (*int16xm2_t merge*, *int16xm2_t a*, *int16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vremvv_mask_int16xm4** (*int16xm4_t merge*, *int16xm4_t a*, *int16xm4_t b*, *e16xm4_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: ['vrsub.vi']

Prototypes:

- *int16xm1_t* **vrsubvi_int16xm1** (*int16xm1_t a*, const short *b*, unsigned int *gvl*)
- *int16xm2_t* **vrsubvi_int16xm2** (*int16xm2_t a*, const short *b*, unsigned int *gvl*)
- *int16xm4_t* **vrsubvi_int16xm4** (*int16xm4_t a*, const short *b*, unsigned int *gvl*)
- *int16xm8_t* **vrsubvi_int16xm8** (*int16xm8_t a*, const short *b*, unsigned int *gvl*)
- *int32xm1_t* **vrsubvi_int32xm1** (*int32xm1_t a*, const int *b*, unsigned int *gvl*)
- *int32xm2_t* **vrsubvi_int32xm2** (*int32xm2_t a*, const int *b*, unsigned int *gvl*)
- *int32xm4_t* **vrsubvi_int32xm4** (*int32xm4_t a*, const int *b*, unsigned int *gvl*)
- *int32xm8_t* **vrsubvi_int32xm8** (*int32xm8_t a*, const int *b*, unsigned int *gvl*)
- *int64xm1_t* **vrsubvi_int64xm1** (*int64xm1_t a*, const long *b*, unsigned int *gvl*)
- *int64xm2_t* **vrsubvi_int64xm2** (*int64xm2_t a*, const long *b*, unsigned int *gvl*)
- *int64xm4_t* **vrsubvi_int64xm4** (*int64xm4_t a*, const long *b*, unsigned int *gvl*)
- *int64xm8_t* **vrsubvi_int64xm8** (*int64xm8_t a*, const long *b*, unsigned int *gvl*)
- *int8xm1_t* **vrsubvi_int8xm1** (*int8xm1_t a*, const signed char *b*, unsigned int *gvl*)
- *int8xm2_t* **vrsubvi_int8xm2** (*int8xm2_t a*, const signed char *b*, unsigned int *gvl*)
- *int8xm4_t* **vrsubvi_int8xm4** (*int8xm4_t a*, const signed char *b*, unsigned int *gvl*)
- *int8xm8_t* **vrsubvi_int8xm8** (*int8xm8_t a*, const signed char *b*, unsigned int *gvl*)
- *uint16xm1_t* **vrsubvi_uint16xm1** (*uint16xm1_t a*, const unsigned short *b*, unsigned int *gvl*)
- *uint16xm2_t* **vrsubvi_uint16xm2** (*uint16xm2_t a*, const unsigned short *b*, unsigned int *gvl*)
- *uint16xm4_t* **vrsubvi_uint16xm4** (*uint16xm4_t a*, const unsigned short *b*, unsigned int *gvl*)
- *uint16xm8_t* **vrsubvi_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 addtion 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 addtion 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 addtion 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 addtion 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 addtion 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 addtion 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 addtion with borrow

Instruction: ['vsbc.vvm']

Masked prototypes:

- *int16xm1_t* **vsbcvvm_mask_int16xm1** (*int16xm1_t a*, *int16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *int16xm2_t* **vsbcvvm_mask_int16xm2** (*int16xm2_t a*, *int16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vsbcvvm_mask_int16xm4** (*int16xm4_t a*, *int16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vsbcvvm_mask_int16xm8** (*int16xm8_t a*, *int16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)
- *int32xm1_t* **vsbcvvm_mask_int32xm1** (*int32xm1_t a*, *int32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vsbcvvm_mask_int32xm2** (*int32xm2_t a*, *int32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vsbcvvm_mask_int32xm4** (*int32xm4_t a*, *int32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vsbcvvm_mask_int32xm8** (*int32xm8_t a*, *int32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *int64xm1_t* **vsbcvvm_mask_int64xm1** (*int64xm1_t a*, *int64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vsbcvvm_mask_int64xm2** (*int64xm2_t a*, *int64xm2_t b*, *e64xm2_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vsbcvvm_mask_int64xm4** (*int64xm4_t a*, *int64xm4_t b*, *e64xm4_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vsbcvvm_mask_int64xm8** (*int64xm8_t a*, *int64xm8_t b*, *e64xm8_t mask*, unsigned int *gvl*)
- *int8xm1_t* **vsbcvvm_mask_int8xm1** (*int8xm1_t a*, *int8xm1_t b*, *e8xm1_t mask*, unsigned int *gvl*)
- *int8xm2_t* **vsbcvvm_mask_int8xm2** (*int8xm2_t a*, *int8xm2_t b*, *e8xm2_t mask*, unsigned int *gvl*)
- *int8xm4_t* **vsbcvvm_mask_int8xm4** (*int8xm4_t a*, *int8xm4_t b*, *e8xm4_t mask*, unsigned int *gvl*)
- *int8xm8_t* **vsbcvvm_mask_int8xm8** (*int8xm8_t a*, *int8xm8_t b*, *e8xm8_t mask*, unsigned int *gvl*)
- *uint16xm1_t* **vsbcvvm_mask_uint16xm1** (*uint16xm1_t a*, *uint16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint16xm2_t* **vsbcvvm_mask_uint16xm2** (*uint16xm2_t a*, *uint16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vsbcvvm_mask_uint16xm4** (*uint16xm4_t a*, *uint16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vsbcvvm_mask_uint16xm8** (*uint16xm8_t a*, *uint16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)
- *uint32xm1_t* **vsbcvvm_mask_uint32xm1** (*uint32xm1_t a*, *uint32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vsbcvvm_mask_uint32xm2** (*uint32xm2_t a*, *uint32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vsbcvvm_mask_uint32xm4** (*uint32xm4_t a*, *uint32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vsbcvvm_mask_uint32xm8** (*uint32xm8_t a*, *uint32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *uint64xm1_t* **vsbcvvm_mask_uint64xm1** (*uint64xm1_t a*, *uint64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)

- *uint64xm2_t* **vsbcvvm_mask_uint64xm2** (*uint64xm2_t a*, *uint64xm2_t b*, *e64xm2_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vsbcvvm_mask_uint64xm4** (*uint64xm4_t a*, *uint64xm4_t b*, *e64xm4_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vsbcvvm_mask_uint64xm8** (*uint64xm8_t a*, *uint64xm8_t b*, *e64xm8_t mask*, unsigned int *gvl*)
- *uint8xm1_t* **vsbcvvm_mask_uint8xm1** (*uint8xm1_t a*, *uint8xm1_t b*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint8xm2_t* **vsbcvvm_mask_uint8xm2** (*uint8xm2_t a*, *uint8xm2_t b*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint8xm4_t* **vsbcvvm_mask_uint8xm4** (*uint8xm4_t a*, *uint8xm4_t b*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint8xm8_t* **vsbcvvm_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
```

2.7.79 Elementwise vector-scalar integer addtion with borrow

Instruction: ['vsbc.vxm']

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[elemet] + (1 << b - 1)) >> b[elemet]
    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: ['vssrl.vi']

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[elemet] + (1 << b - 1)) >> b[elemet]
    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[elemet] + (1 << b[element] - 1)) >> b[elemet]
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

2.7.87 Elementwise unsigned vector-scalar unsigned scaling shift

Instruction: ['vssrl.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[elemet] + (1 << b - 1)) >> b[elemet]
    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[elemet] + (1 << b - 1)) >> b[elemet]
      else
        result[element] = merge[element]
    result[gvl : VLMAX] = 0
```

2.7.88 Elementwise signed vector-vector substraction 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 substraction 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 substraction 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 substraction 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: ['vwmacc.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: ['vwmacc.vx']

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: ['vwmaccu.vv']

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* **vwmaccuvx_uint16xm2_uint8xm1** (unsigned char *a*, *uint8xm1_t b*, *uint16xm2_t result*, unsigned int *gvl*)
- *uint16xm4_t* **vwmaccuvx_uint16xm4_uint8xm2** (unsigned char *a*, *uint8xm2_t b*, *uint16xm4_t result*, unsigned int *gvl*)
- *uint16xm8_t* **vwmaccuvx_uint16xm8_uint8xm4** (unsigned char *a*, *uint8xm4_t b*, *uint16xm8_t result*, unsigned int *gvl*)
- *uint32xm2_t* **vwmaccuvx_uint32xm2_uint16xm1** (unsigned short *a*, *uint16xm1_t b*, *uint32xm2_t result*, unsigned int *gvl*)
- *uint32xm4_t* **vwmaccuvx_uint32xm4_uint16xm2** (unsigned short *a*, *uint16xm2_t b*, *uint32xm4_t result*, unsigned int *gvl*)
- *uint32xm8_t* **vwmaccuvx_uint32xm8_uint16xm4** (unsigned short *a*, *uint16xm4_t b*, *uint32xm8_t result*, unsigned int *gvl*)
- *uint64xm2_t* **vwmaccuvx_uint64xm2_uint32xm1** (unsigned int *a*, *uint32xm1_t b*, *uint64xm2_t result*, unsigned int *gvl*)
- *uint64xm4_t* **vwmaccuvx_uint64xm4_uint32xm2** (unsigned int *a*, *uint32xm2_t b*, *uint64xm4_t result*, unsigned int *gvl*)
- *uint64xm8_t* **vwmaccuvx_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* **vwmaccuvx_mask_uint16xm2_uint8xm1** (*uint16xm2_t merge*, unsigned char *a*, *uint8xm1_t b*, *uint16xm2_t result*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vwmaccuvx_mask_uint16xm4_uint8xm2** (*uint16xm4_t merge*, unsigned char *a*, *uint8xm2_t b*, *uint16xm4_t result*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vwmaccuvx_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 multiplition

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 multiplition

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: ['vwmulsu.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* **vwsmaccvv_int32xm4_int16xm2** (*int16xm2_t a*, *int16xm2_t b*, *int32xm4_t result*, unsigned int *gvl*)
- *int32xm8_t* **vwsmaccvv_int32xm8_int16xm4** (*int16xm4_t a*, *int16xm4_t b*, *int32xm8_t result*, unsigned int *gvl*)
- *int64xm2_t* **vwsmaccvv_int64xm2_int32xm1** (*int32xm1_t a*, *int32xm1_t b*, *int64xm2_t result*, unsigned int *gvl*)
- *int64xm4_t* **vwsmaccvv_int64xm4_int32xm2** (*int32xm2_t a*, *int32xm2_t b*, *int64xm4_t result*, unsigned int *gvl*)
- *int64xm8_t* **vwsmaccvv_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[elemet] * c[elemet] + round) >> (sew / 2)) +␣
↪result[element])
    result[gvl : VLMAX] = 0
```

Masked prototypes:

- *int16xm2_t* **vwsmaccvv_mask_int16xm2_int8xm1** (*int16xm2_t merge*, *int8xm1_t a*, *int8xm1_t b*, *int16xm2_t result*, *e8xm1_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vwsmaccvv_mask_int16xm4_int8xm2** (*int16xm4_t merge*, *int8xm2_t a*, *int8xm2_t b*, *int16xm4_t result*, *e8xm2_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vwsmaccvv_mask_int16xm8_int8xm4** (*int16xm8_t merge*, *int8xm4_t a*, *int8xm4_t b*, *int16xm8_t result*, *e8xm4_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vwsmaccvv_mask_int32xm2_int16xm1** (*int32xm2_t merge*, *int16xm1_t a*, *int16xm1_t b*, *int32xm2_t result*, *e16xm1_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vwsmaccvv_mask_int32xm4_int16xm2** (*int32xm4_t merge*, *int16xm2_t a*, *int16xm2_t b*, *int32xm4_t result*, *e16xm2_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vwsmaccvv_mask_int32xm8_int16xm4** (*int32xm8_t merge*, *int16xm4_t a*, *int16xm4_t b*, *int32xm8_t result*, *e16xm4_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vwsmaccvv_mask_int64xm2_int32xm1** (*int64xm2_t merge*, *int32xm1_t a*, *int32xm1_t b*, *int64xm2_t result*, *e32xm1_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vwsmaccvv_mask_int64xm4_int32xm2** (*int64xm4_t merge*, *int32xm2_t a*, *int32xm2_t b*, *int64xm4_t result*, *e32xm2_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vwsmaccvv_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[elemet] * c[elemet] + 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* **vwsmaccvx_mask_int32xm2_int16xm1** (*int32xm2_t merge*, short *a*, *int16xm1_t b*, *int32xm2_t result*, *e16xm1_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vwsmaccvx_mask_int32xm4_int16xm2** (*int32xm4_t merge*, short *a*, *int16xm2_t b*, *int32xm4_t result*, *e16xm2_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vwsmaccvx_mask_int32xm8_int16xm4** (*int32xm8_t merge*, short *a*, *int16xm4_t b*, *int32xm8_t result*, *e16xm4_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vwsmaccvx_mask_int64xm2_int32xm1** (*int64xm2_t merge*, int *a*, *int32xm1_t b*, *int64xm2_t result*, *e32xm1_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vwsmaccvx_mask_int64xm4_int32xm2** (*int64xm4_t merge*, int *a*, *int32xm2_t b*, *int64xm4_t result*, *e32xm2_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vwsmaccvx_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 satruation

Instruction: ['vwsmaccsu.vv']

Prototypes:

- *int16xm2_t* **vwsmaccsuvv_int16xm2_int8xm1_uint8xm1** (*int8xm1_t a*, *uint8xm1_t b*, *int16xm2_t result*, unsigned int *gvl*)
- *int16xm4_t* **vwsmaccsuvv_int16xm4_int8xm2_uint8xm2** (*int8xm2_t a*, *uint8xm2_t b*, *int16xm4_t result*, unsigned int *gvl*)
- *int16xm8_t* **vwsmaccsuvv_int16xm8_int8xm4_uint8xm4** (*int8xm4_t a*, *uint8xm4_t b*, *int16xm8_t result*, unsigned int *gvl*)
- *int32xm2_t* **vwsmaccsuvv_int32xm2_int16xm1_uint16xm1** (*int16xm1_t a*, *uint16xm1_t b*, *int32xm2_t result*, unsigned int *gvl*)
- *int32xm4_t* **vwsmaccsuvv_int32xm4_int16xm2_uint16xm2** (*int16xm2_t a*, *uint16xm2_t b*, *int32xm4_t result*, unsigned int *gvl*)
- *int32xm8_t* **vwsmaccsuvv_int32xm8_int16xm4_uint16xm4** (*int16xm4_t a*, *uint16xm4_t b*, *int32xm8_t result*, unsigned int *gvl*)

- *int64xm2_t* **vwsmaccsuvv_int64xm2_int32xm1_uint32xm1** (*int32xm1_t a*, *uint32xm1_t b*, *int64xm2_t result*, unsigned int *gvl*)
- *int64xm4_t* **vwsmaccsuvv_int64xm4_int32xm2_uint32xm2** (*int32xm2_t a*, *uint32xm2_t b*, *int64xm4_t result*, unsigned int *gvl*)
- *int64xm8_t* **vwsmaccsuvv_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* **vwsmaccsuvv_mask_int16xm2_int8xm1_uint8xm1** (*int16xm2_t merge*, *int8xm1_t a*, *uint8xm1_t b*, *int16xm2_t result*, *e8xm1_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vwsmaccsuvv_mask_int16xm4_int8xm2_uint8xm2** (*int16xm4_t merge*, *int8xm2_t a*, *uint8xm2_t b*, *int16xm4_t result*, *e8xm2_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vwsmaccsuvv_mask_int16xm8_int8xm4_uint8xm4** (*int16xm8_t merge*, *int8xm4_t a*, *uint8xm4_t b*, *int16xm8_t result*, *e8xm4_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vwsmaccsuvv_mask_int32xm2_int16xm1_uint16xm1** (*int32xm2_t merge*, *int16xm1_t a*, *uint16xm1_t b*, *int32xm2_t result*, *e16xm1_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vwsmaccsuvv_mask_int32xm4_int16xm2_uint16xm2** (*int32xm4_t merge*, *int16xm2_t a*, *uint16xm2_t b*, *int32xm4_t result*, *e16xm2_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vwsmaccsuvv_mask_int32xm8_int16xm4_uint16xm4** (*int32xm8_t merge*, *int16xm4_t a*, *uint16xm4_t b*, *int32xm8_t result*, *e16xm4_t mask*, unsigned int *gvl*)

- *int64xm2_t* **vwsmaccsuvv_mask_int64xm2_int32xm1_uint32xm1** (*int64xm2_t merge*, *int32xm1_t a*, *uint32xm1_t b*, *int64xm2_t result*, *e32xm1_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vwsmaccsuvv_mask_int64xm4_int32xm2_uint32xm2** (*int64xm4_t merge*, *int32xm2_t a*, *uint32xm2_t b*, *int64xm4_t result*, *e32xm2_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vwsmaccsuvv_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 satruation

Instruction: ['vwsmaccsu.vx']

Prototypes:

- *int16xm2_t* **vwsmaccsuvx_int16xm2_uint8xm1** (signed char *a*, *uint8xm1_t b*, *int16xm2_t result*, unsigned int *gvl*)
- *int16xm4_t* **vwsmaccsuvx_int16xm4_uint8xm2** (signed char *a*, *uint8xm2_t b*, *int16xm4_t result*, unsigned int *gvl*)
- *int16xm8_t* **vwsmaccsuvx_int16xm8_uint8xm4** (signed char *a*, *uint8xm4_t b*, *int16xm8_t result*, unsigned int *gvl*)
- *int32xm2_t* **vwsmaccsuvx_int32xm2_uint16xm1** (short *a*, *uint16xm1_t b*, *int32xm2_t result*, unsigned int *gvl*)
- *int32xm4_t* **vwsmaccsuvx_int32xm4_uint16xm2** (short *a*, *uint16xm2_t b*, *int32xm4_t result*, unsigned int *gvl*)
- *int32xm8_t* **vwsmaccsuvx_int32xm8_uint16xm4** (short *a*, *uint16xm4_t b*, *int32xm8_t result*, unsigned int *gvl*)
- *int64xm2_t* **vwsmaccsuvx_int64xm2_uint32xm1** (int *a*, *uint32xm1_t b*, *int64xm2_t result*, unsigned int *gvl*)
- *int64xm4_t* **vwsmaccsuvx_int64xm4_uint32xm2** (int *a*, *uint32xm2_t b*, *int64xm4_t result*, unsigned int *gvl*)

- *int64xm8_t* **vwsmaccsuvx_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* **vwsmaccsuvx_mask_int16xm2_uint8xm1** (*int16xm2_t merge*, signed char *a*, *uint8xm1_t b*, *int16xm2_t result*, *e8xm1_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vwsmaccsuvx_mask_int16xm4_uint8xm2** (*int16xm4_t merge*, signed char *a*, *uint8xm2_t b*, *int16xm4_t result*, *e8xm2_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vwsmaccsuvx_mask_int16xm8_uint8xm4** (*int16xm8_t merge*, signed char *a*, *uint8xm4_t b*, *int16xm8_t result*, *e8xm4_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vwsmaccsuvx_mask_int32xm2_uint16xm1** (*int32xm2_t merge*, short *a*, *uint16xm1_t b*, *int32xm2_t result*, *e16xm1_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vwsmaccsuvx_mask_int32xm4_uint16xm2** (*int32xm4_t merge*, short *a*, *uint16xm2_t b*, *int32xm4_t result*, *e16xm2_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vwsmaccsuvx_mask_int32xm8_uint16xm4** (*int32xm8_t merge*, short *a*, *uint16xm4_t b*, *int32xm8_t result*, *e16xm4_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vwsmaccsuvx_mask_int64xm2_uint32xm1** (*int64xm2_t merge*, int *a*, *uint32xm1_t b*, *int64xm2_t result*, *e32xm1_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vwsmaccsuvx_mask_int64xm4_uint32xm2** (*int64xm4_t merge*, int *a*, *uint32xm2_t b*, *int64xm4_t result*, *e32xm2_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vwsmaccsuvx_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: ['vwsmaccu.vv']

Prototypes:

- *uint16xm2_t* **vwsmaccuvv_uint16xm2_uint8xm1** (*uint8xm1_t a*, *uint8xm1_t b*, *uint16xm2_t result*, unsigned int *gvl*)
- *uint16xm4_t* **vwsmaccuvv_uint16xm4_uint8xm2** (*uint8xm2_t a*, *uint8xm2_t b*, *uint16xm4_t result*, unsigned int *gvl*)
- *uint16xm8_t* **vwsmaccuvv_uint16xm8_uint8xm4** (*uint8xm4_t a*, *uint8xm4_t b*, *uint16xm8_t result*, unsigned int *gvl*)
- *uint32xm2_t* **vwsmaccuvv_uint32xm2_uint16xm1** (*uint16xm1_t a*, *uint16xm1_t b*, *uint32xm2_t result*, unsigned int *gvl*)
- *uint32xm4_t* **vwsmaccuvv_uint32xm4_uint16xm2** (*uint16xm2_t a*, *uint16xm2_t b*, *uint32xm4_t result*, unsigned int *gvl*)
- *uint32xm8_t* **vwsmaccuvv_uint32xm8_uint16xm4** (*uint16xm4_t a*, *uint16xm4_t b*, *uint32xm8_t result*, unsigned int *gvl*)
- *uint64xm2_t* **vwsmaccuvv_uint64xm2_uint32xm1** (*uint32xm1_t a*, *uint32xm1_t b*, *uint64xm2_t result*, unsigned int *gvl*)
- *uint64xm4_t* **vwsmaccuvv_uint64xm4_uint32xm2** (*uint32xm2_t a*, *uint32xm2_t b*, *uint64xm4_t result*, unsigned int *gvl*)
- *uint64xm8_t* **vwsmaccuvv_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* **vwsmaccuvv_mask_uint16xm2_uint8xm1** (*uint16xm2_t merge*, *uint8xm1_t a*, *uint8xm1_t b*, *uint16xm2_t result*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vwsmaccuvv_mask_uint16xm4_uint8xm2** (*uint16xm4_t merge*, *uint8xm2_t a*, *uint8xm2_t b*, *uint16xm4_t result*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vwsmaccuvv_mask_uint16xm8_uint8xm4** (*uint16xm8_t merge*, *uint8xm4_t a*, *uint8xm4_t b*, *uint16xm8_t result*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vwsmaccuvv_mask_uint32xm2_uint16xm1** (*uint32xm2_t merge*, *uint16xm1_t a*, *uint16xm1_t b*, *uint32xm2_t result*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vwsmaccuvv_mask_uint32xm4_uint16xm2** (*uint32xm4_t merge*, *uint16xm2_t a*, *uint16xm2_t b*, *uint32xm4_t result*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vwsmaccuvv_mask_uint32xm8_uint16xm4** (*uint32xm8_t merge*, *uint16xm4_t a*, *uint16xm4_t b*, *uint32xm8_t result*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vwsmaccuvv_mask_uint64xm2_uint32xm1** (*uint64xm2_t merge*, *uint32xm1_t a*, *uint32xm1_t b*, *uint64xm2_t result*, *e32xm1_t mask*, unsigned int *gvl*)

- *uint64xm4_t* **vwsmaccuvv_mask_uint64xm4_uint32xm2** (*uint64xm4_t merge*, *uint32xm2_t a*, *uint32xm2_t b*, *uint64xm4_t result*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vwsmaccuvv_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: ['vwsmaccu.vx']

Prototypes:

- *uint16xm2_t* **vwsmaccuvx_uint16xm2_uint8xm1** (unsigned char *a*, *uint8xm1_t b*, *uint16xm2_t result*, unsigned int *gvl*)
- *uint16xm4_t* **vwsmaccuvx_uint16xm4_uint8xm2** (unsigned char *a*, *uint8xm2_t b*, *uint16xm4_t result*, unsigned int *gvl*)
- *uint16xm8_t* **vwsmaccuvx_uint16xm8_uint8xm4** (unsigned char *a*, *uint8xm4_t b*, *uint16xm8_t result*, unsigned int *gvl*)
- *uint32xm2_t* **vwsmaccuvx_uint32xm2_uint16xm1** (unsigned short *a*, *uint16xm1_t b*, *uint32xm2_t result*, unsigned int *gvl*)
- *uint32xm4_t* **vwsmaccuvx_uint32xm4_uint16xm2** (unsigned short *a*, *uint16xm2_t b*, *uint32xm4_t result*, unsigned int *gvl*)
- *uint32xm8_t* **vwsmaccuvx_uint32xm8_uint16xm4** (unsigned short *a*, *uint16xm4_t b*, *uint32xm8_t result*, unsigned int *gvl*)
- *uint64xm2_t* **vwsmaccuvx_uint64xm2_uint32xm1** (unsigned int *a*, *uint32xm1_t b*, *uint64xm2_t result*, unsigned int *gvl*)
- *uint64xm4_t* **vwsmaccuvx_uint64xm4_uint32xm2** (unsigned int *a*, *uint32xm2_t b*, *uint64xm4_t result*, unsigned int *gvl*)
- *uint64xm8_t* **vwsmaccuvx_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* **vwsmaccuvx_mask_uint16xm2_uint8xm1** (*uint16xm2_t merge*, unsigned char *a*, *uint8xm1_t b*, *uint16xm2_t result*, *e8xm1_t mask*, unsigned int *gvl*)

- *uint16xm4_t* **vwsmaccuvx_mask_uint16xm4_uint8xm2** (*uint16xm4_t merge*, unsigned char *a*, *uint8xm2_t b*, *uint16xm4_t result*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vwsmaccuvx_mask_uint16xm8_uint8xm4** (*uint16xm8_t merge*, unsigned char *a*, *uint8xm4_t b*, *uint16xm8_t result*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vwsmaccuvx_mask_uint32xm2_uint16xm1** (*uint32xm2_t merge*, unsigned short *a*, *uint16xm1_t b*, *uint32xm2_t result*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vwsmaccuvx_mask_uint32xm4_uint16xm2** (*uint32xm4_t merge*, unsigned short *a*, *uint16xm2_t b*, *uint32xm4_t result*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vwsmaccuvx_mask_uint32xm8_uint16xm4** (*uint32xm8_t merge*, unsigned short *a*, *uint16xm4_t b*, *uint32xm8_t result*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vwsmaccuvx_mask_uint64xm2_uint32xm1** (*uint64xm2_t merge*, unsigned int *a*, *uint32xm1_t b*, *uint64xm2_t result*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vwsmaccuvx_mask_uint64xm4_uint32xm2** (*uint64xm4_t merge*, unsigned int *a*, *uint32xm2_t b*, *uint64xm4_t result*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vwsmaccuvx_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: ['vwsmaccus.vx']

Prototypes:

- *int16xm2_t* **vwsmaccusvx_int16xm2_int8xm1** (unsigned char *a*, *int8xm1_t b*, *int16xm2_t result*, unsigned int *gvl*)
- *int16xm4_t* **vwsmaccusvx_int16xm4_int8xm2** (unsigned char *a*, *int8xm2_t b*, *int16xm4_t result*, unsigned int *gvl*)
- *int16xm8_t* **vwsmaccusvx_int16xm8_int8xm4** (unsigned char *a*, *int8xm4_t b*, *int16xm8_t result*, unsigned int *gvl*)
- *int32xm2_t* **vwsmaccusvx_int32xm2_int16xm1** (unsigned short *a*, *int16xm1_t b*, *int32xm2_t result*, unsigned int *gvl*)
- *int32xm4_t* **vwsmaccusvx_int32xm4_int16xm2** (unsigned short *a*, *int16xm2_t b*, *int32xm4_t result*, unsigned int *gvl*)

- *int32xm8_t* **vwsmaccusvx_int32xm8_int16xm4** (unsigned short *a*, *int16xm4_t b*, *int32xm8_t result*, unsigned int *gvl*)
- *int64xm2_t* **vwsmaccusvx_int64xm2_int32xm1** (unsigned int *a*, *int32xm1_t b*, *int64xm2_t result*, unsigned int *gvl*)
- *int64xm4_t* **vwsmaccusvx_int64xm4_int32xm2** (unsigned int *a*, *int32xm2_t b*, *int64xm4_t result*, unsigned int *gvl*)
- *int64xm8_t* **vwsmaccusvx_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* **vwsmaccusvx_mask_int16xm2_int8xm1** (*int16xm2_t merge*, unsigned char *a*, *int8xm1_t b*, *int16xm2_t result*, *e8xm1_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vwsmaccusvx_mask_int16xm4_int8xm2** (*int16xm4_t merge*, unsigned char *a*, *int8xm2_t b*, *int16xm4_t result*, *e8xm2_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vwsmaccusvx_mask_int16xm8_int8xm4** (*int16xm8_t merge*, unsigned char *a*, *int8xm4_t b*, *int16xm8_t result*, *e8xm4_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vwsmaccusvx_mask_int32xm2_int16xm1** (*int32xm2_t merge*, unsigned short *a*, *int16xm1_t b*, *int32xm2_t result*, *e16xm1_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vwsmaccusvx_mask_int32xm4_int16xm2** (*int32xm4_t merge*, unsigned short *a*, *int16xm2_t b*, *int32xm4_t result*, *e16xm2_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vwsmaccusvx_mask_int32xm8_int16xm4** (*int32xm8_t merge*, unsigned short *a*, *int16xm4_t b*, *int32xm8_t result*, *e16xm4_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vwsmaccusvx_mask_int64xm2_int32xm1** (*int64xm2_t merge*, unsigned int *a*, *int32xm1_t b*, *int64xm2_t result*, *e32xm1_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vwsmaccusvx_mask_int64xm4_int32xm2** (*int64xm4_t merge*, unsigned int *a*, *int32xm2_t b*, *int64xm4_t result*, *e32xm2_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vwsmaccusvx_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: ['vwsub.vv']

Prototypes:

- *int16xm2_t* **vwsubvv_int16xm2_int8xm1** (*int8xm1_t a*, *int8xm1_t b*, unsigned int *gvl*)
- *int16xm4_t* **vwsubvv_int16xm4_int8xm2** (*int8xm2_t a*, *int8xm2_t b*, unsigned int *gvl*)
- *int16xm8_t* **vwsubvv_int16xm8_int8xm4** (*int8xm4_t a*, *int8xm4_t b*, unsigned int *gvl*)
- *int32xm2_t* **vwsubvv_int32xm2_int16xm1** (*int16xm1_t a*, *int16xm1_t b*, unsigned int *gvl*)
- *int32xm4_t* **vwsubvv_int32xm4_int16xm2** (*int16xm2_t a*, *int16xm2_t b*, unsigned int *gvl*)
- *int32xm8_t* **vwsubvv_int32xm8_int16xm4** (*int16xm4_t a*, *int16xm4_t b*, unsigned int *gvl*)
- *int64xm2_t* **vwsubvv_int64xm2_int32xm1** (*int32xm1_t a*, *int32xm1_t b*, unsigned int *gvl*)
- *int64xm4_t* **vwsubvv_int64xm4_int32xm2** (*int32xm2_t a*, *int32xm2_t b*, unsigned int *gvl*)
- *int64xm8_t* **vwsubvv_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* **vwsubvv_mask_int16xm2_int8xm1** (*int16xm2_t merge*, *int8xm1_t a*, *int8xm1_t b*, *e8xm1_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vwsubvv_mask_int16xm4_int8xm2** (*int16xm4_t merge*, *int8xm2_t a*, *int8xm2_t b*, *e8xm2_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vwsubvv_mask_int16xm8_int8xm4** (*int16xm8_t merge*, *int8xm4_t a*, *int8xm4_t b*, *e8xm4_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vwsubvv_mask_int32xm2_int16xm1** (*int32xm2_t merge*, *int16xm1_t a*, *int16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vwsubvv_mask_int32xm4_int16xm2** (*int32xm4_t merge*, *int16xm2_t a*, *int16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vwsubvv_mask_int32xm8_int16xm4** (*int32xm8_t merge*, *int16xm4_t a*, *int16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vwsubvv_mask_int64xm2_int32xm1** (*int64xm2_t merge*, *int32xm1_t a*, *int32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vwsubvv_mask_int64xm4_int32xm2** (*int64xm4_t merge*, *int32xm2_t a*, *int32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vwsubvv_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: ['vwsub.vx']

Prototypes:

- *int16xm2_t* **vwsubvx_int16xm2_int8xm1** (*int8xm1_t a*, signed char *b*, unsigned int *gvl*)
- *int16xm4_t* **vwsubvx_int16xm4_int8xm2** (*int8xm2_t a*, signed char *b*, unsigned int *gvl*)
- *int16xm8_t* **vwsubvx_int16xm8_int8xm4** (*int8xm4_t a*, signed char *b*, unsigned int *gvl*)
- *int32xm2_t* **vwsubvx_int32xm2_int16xm1** (*int16xm1_t a*, short *b*, unsigned int *gvl*)
- *int32xm4_t* **vwsubvx_int32xm4_int16xm2** (*int16xm2_t a*, short *b*, unsigned int *gvl*)
- *int32xm8_t* **vwsubvx_int32xm8_int16xm4** (*int16xm4_t a*, short *b*, unsigned int *gvl*)
- *int64xm2_t* **vwsubvx_int64xm2_int32xm1** (*int32xm1_t a*, int *b*, unsigned int *gvl*)
- *int64xm4_t* **vwsubvx_int64xm4_int32xm2** (*int32xm2_t a*, int *b*, unsigned int *gvl*)
- *int64xm8_t* **vwsubvx_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* **vwsubvx_mask_int16xm2_int8xm1** (*int16xm2_t merge*, *int8xm1_t a*, signed char *b*, *e8xm1_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vwsubvx_mask_int16xm4_int8xm2** (*int16xm4_t merge*, *int8xm2_t a*, signed char *b*, *e8xm2_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vwsubvx_mask_int16xm8_int8xm4** (*int16xm8_t merge*, *int8xm4_t a*, signed char *b*, *e8xm4_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vwsubvx_mask_int32xm2_int16xm1** (*int32xm2_t merge*, *int16xm1_t a*, short *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vwsubvx_mask_int32xm4_int16xm2** (*int32xm4_t merge*, *int16xm2_t a*, short *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vwsubvx_mask_int32xm8_int16xm4** (*int32xm8_t merge*, *int16xm4_t a*, short *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vwsubvx_mask_int64xm2_int32xm1** (*int64xm2_t merge*, *int32xm1_t a*, int *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vwsubvx_mask_int64xm4_int32xm2** (*int64xm4_t merge*, *int32xm2_t a*, int *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vwsubvx_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: ['vwsub.wv']

Prototypes:

- *int16xm2_t* **vwsubwv_int16xm2_int8xm1** (*int16xm2_t a*, *int8xm1_t b*, unsigned int *gvl*)
- *int16xm4_t* **vwsubwv_int16xm4_int8xm2** (*int16xm4_t a*, *int8xm2_t b*, unsigned int *gvl*)
- *int16xm8_t* **vwsubwv_int16xm8_int8xm4** (*int16xm8_t a*, *int8xm4_t b*, unsigned int *gvl*)
- *int32xm2_t* **vwsubwv_int32xm2_int16xm1** (*int32xm2_t a*, *int16xm1_t b*, unsigned int *gvl*)
- *int32xm4_t* **vwsubwv_int32xm4_int16xm2** (*int32xm4_t a*, *int16xm2_t b*, unsigned int *gvl*)
- *int32xm8_t* **vwsubwv_int32xm8_int16xm4** (*int32xm8_t a*, *int16xm4_t b*, unsigned int *gvl*)
- *int64xm2_t* **vwsubwv_int64xm2_int32xm1** (*int64xm2_t a*, *int32xm1_t b*, unsigned int *gvl*)
- *int64xm4_t* **vwsubwv_int64xm4_int32xm2** (*int64xm4_t a*, *int32xm2_t b*, unsigned int *gvl*)
- *int64xm8_t* **vwsubwv_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* **vwsubwv_mask_int16xm2_int8xm1** (*int16xm2_t merge*, *int16xm2_t a*, *int8xm1_t b*, *e8xm1_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vwsubwv_mask_int16xm4_int8xm2** (*int16xm4_t merge*, *int16xm4_t a*, *int8xm2_t b*, *e8xm2_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vwsubwv_mask_int16xm8_int8xm4** (*int16xm8_t merge*, *int16xm8_t a*, *int8xm4_t b*, *e8xm4_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vwsubwv_mask_int32xm2_int16xm1** (*int32xm2_t merge*, *int32xm2_t a*, *int16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vwsubwv_mask_int32xm4_int16xm2** (*int32xm4_t merge*, *int32xm4_t a*, *int16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vwsubwv_mask_int32xm8_int16xm4** (*int32xm8_t merge*, *int32xm8_t a*, *int16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vwsubwv_mask_int64xm2_int32xm1** (*int64xm2_t merge*, *int64xm2_t a*, *int32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vwsubwv_mask_int64xm4_int32xm2** (*int64xm4_t merge*, *int64xm4_t a*, *int32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)

- *int64xm8_t* **vwsubwv_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: ['vwsub.wx']

Prototypes:

- *int16xm2_t* **vwsubwx_int16xm2** (*int16xm2_t a*, signed char *b*, unsigned int *gvl*)
- *int16xm4_t* **vwsubwx_int16xm4** (*int16xm4_t a*, signed char *b*, unsigned int *gvl*)
- *int16xm8_t* **vwsubwx_int16xm8** (*int16xm8_t a*, signed char *b*, unsigned int *gvl*)
- *int32xm2_t* **vwsubwx_int32xm2** (*int32xm2_t a*, short *b*, unsigned int *gvl*)
- *int32xm4_t* **vwsubwx_int32xm4** (*int32xm4_t a*, short *b*, unsigned int *gvl*)
- *int32xm8_t* **vwsubwx_int32xm8** (*int32xm8_t a*, short *b*, unsigned int *gvl*)
- *int64xm2_t* **vwsubwx_int64xm2** (*int64xm2_t a*, int *b*, unsigned int *gvl*)
- *int64xm4_t* **vwsubwx_int64xm4** (*int64xm4_t a*, int *b*, unsigned int *gvl*)
- *int64xm8_t* **vwsubwx_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* **vwsubwx_mask_int16xm2** (*int16xm2_t merge*, *int16xm2_t a*, signed char *b*, *e8xm1_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vwsubwx_mask_int16xm4** (*int16xm4_t merge*, *int16xm4_t a*, signed char *b*, *e8xm2_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vwsubwx_mask_int16xm8** (*int16xm8_t merge*, *int16xm8_t a*, signed char *b*, *e8xm4_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vwsubwx_mask_int32xm2** (*int32xm2_t merge*, *int32xm2_t a*, short *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vwsubwx_mask_int32xm4** (*int32xm4_t merge*, *int32xm4_t a*, short *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vwsubwx_mask_int32xm8** (*int32xm8_t merge*, *int32xm8_t a*, short *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vwsubwx_mask_int64xm2** (*int64xm2_t merge*, *int64xm2_t a*, int *b*, *e32xm1_t mask*, unsigned int *gvl*)

- *int64xm4_t* **vwsubwx_mask_int64xm4** (*int64xm4_t merge*, *int64xm4_t a*, int *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vwsubwx_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: ['vwsubu.vv']

Prototypes:

- *uint16xm2_t* **vwsubuvv_uint16xm2_uint8xm1** (*uint8xm1_t a*, *uint8xm1_t b*, unsigned int *gvl*)
- *uint16xm4_t* **vwsubuvv_uint16xm4_uint8xm2** (*uint8xm2_t a*, *uint8xm2_t b*, unsigned int *gvl*)
- *uint16xm8_t* **vwsubuvv_uint16xm8_uint8xm4** (*uint8xm4_t a*, *uint8xm4_t b*, unsigned int *gvl*)
- *uint32xm2_t* **vwsubuvv_uint32xm2_uint16xm1** (*uint16xm1_t a*, *uint16xm1_t b*, unsigned int *gvl*)
- *uint32xm4_t* **vwsubuvv_uint32xm4_uint16xm2** (*uint16xm2_t a*, *uint16xm2_t b*, unsigned int *gvl*)
- *uint32xm8_t* **vwsubuvv_uint32xm8_uint16xm4** (*uint16xm4_t a*, *uint16xm4_t b*, unsigned int *gvl*)
- *uint64xm2_t* **vwsubuvv_uint64xm2_uint32xm1** (*uint32xm1_t a*, *uint32xm1_t b*, unsigned int *gvl*)
- *uint64xm4_t* **vwsubuvv_uint64xm4_uint32xm2** (*uint32xm2_t a*, *uint32xm2_t b*, unsigned int *gvl*)
- *uint64xm8_t* **vwsubuvv_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* **vwsubuvv_mask_uint16xm2_uint8xm1** (*uint16xm2_t merge*, *uint8xm1_t a*, *uint8xm1_t b*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vwsubuvv_mask_uint16xm4_uint8xm2** (*uint16xm4_t merge*, *uint8xm2_t a*, *uint8xm2_t b*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vwsubuvv_mask_uint16xm8_uint8xm4** (*uint16xm8_t merge*, *uint8xm4_t a*, *uint8xm4_t b*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vwsubuvv_mask_uint32xm2_uint16xm1** (*uint32xm2_t merge*, *uint16xm1_t a*, *uint16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)

- *uint32xm4_t* **vwsubuvv_mask_uint32xm4_uint16xm2** (*uint32xm4_t merge*, *uint16xm2_t a*, *uint16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vwsubuvv_mask_uint32xm8_uint16xm4** (*uint32xm8_t merge*, *uint16xm4_t a*, *uint16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vwsubuvv_mask_uint64xm2_uint32xm1** (*uint64xm2_t merge*, *uint32xm1_t a*, *uint32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vwsubuvv_mask_uint64xm4_uint32xm2** (*uint64xm4_t merge*, *uint32xm2_t a*, *uint32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vwsubuvv_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: ['vwsubu.vx']

Prototypes:

- *uint16xm2_t* **vwsubuvx_uint16xm2_uint8xm1** (*uint8xm1_t a*, unsigned char *b*, unsigned int *gvl*)
- *uint16xm4_t* **vwsubuvx_uint16xm4_uint8xm2** (*uint8xm2_t a*, unsigned char *b*, unsigned int *gvl*)
- *uint16xm8_t* **vwsubuvx_uint16xm8_uint8xm4** (*uint8xm4_t a*, unsigned char *b*, unsigned int *gvl*)
- *uint32xm2_t* **vwsubuvx_uint32xm2_uint16xm1** (*uint16xm1_t a*, unsigned short *b*, unsigned int *gvl*)
- *uint32xm4_t* **vwsubuvx_uint32xm4_uint16xm2** (*uint16xm2_t a*, unsigned short *b*, unsigned int *gvl*)
- *uint32xm8_t* **vwsubuvx_uint32xm8_uint16xm4** (*uint16xm4_t a*, unsigned short *b*, unsigned int *gvl*)
- *uint64xm2_t* **vwsubuvx_uint64xm2_uint32xm1** (*uint32xm1_t a*, unsigned int *b*, unsigned int *gvl*)
- *uint64xm4_t* **vwsubuvx_uint64xm4_uint32xm2** (*uint32xm2_t a*, unsigned int *b*, unsigned int *gvl*)
- *uint64xm8_t* **vwsubuvx_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* **vwsubuvx_mask_uint16xm2_uint8xm1** (*uint16xm2_t merge*, *uint8xm1_t a*, unsigned char *b*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vwsubuvx_mask_uint16xm4_uint8xm2** (*uint16xm4_t merge*, *uint8xm2_t a*, unsigned char *b*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vwsubuvx_mask_uint16xm8_uint8xm4** (*uint16xm8_t merge*, *uint8xm4_t a*, unsigned char *b*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vwsubuvx_mask_uint32xm2_uint16xm1** (*uint32xm2_t merge*, *uint16xm1_t a*, unsigned short *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vwsubuvx_mask_uint32xm4_uint16xm2** (*uint32xm4_t merge*, *uint16xm2_t a*, unsigned short *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vwsubuvx_mask_uint32xm8_uint16xm4** (*uint32xm8_t merge*, *uint16xm4_t a*, unsigned short *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vwsubuvx_mask_uint64xm2_uint32xm1** (*uint64xm2_t merge*, *uint32xm1_t a*, unsigned int *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vwsubuvx_mask_uint64xm4_uint32xm2** (*uint64xm4_t merge*, *uint32xm2_t a*, unsigned int *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vwsubuvx_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: ['vwsubu.wv']

Prototypes:

- *uint16xm2_t* **vwsubuwv_uint16xm2_uint8xm1** (*uint16xm2_t a*, *uint8xm1_t b*, unsigned int *gvl*)
- *uint16xm4_t* **vwsubuwv_uint16xm4_uint8xm2** (*uint16xm4_t a*, *uint8xm2_t b*, unsigned int *gvl*)
- *uint16xm8_t* **vwsubuwv_uint16xm8_uint8xm4** (*uint16xm8_t a*, *uint8xm4_t b*, unsigned int *gvl*)
- *uint32xm2_t* **vwsubuwv_uint32xm2_uint16xm1** (*uint32xm2_t a*, *uint16xm1_t b*, unsigned int *gvl*)
- *uint32xm4_t* **vwsubuwv_uint32xm4_uint16xm2** (*uint32xm4_t a*, *uint16xm2_t b*, unsigned int *gvl*)
- *uint32xm8_t* **vwsubuwv_uint32xm8_uint16xm4** (*uint32xm8_t a*, *uint16xm4_t b*, unsigned int *gvl*)

- *uint64xm2_t* **vwsubuwv_uint64xm2_uint32xm1** (*uint64xm2_t a*, *uint32xm1_t b*, unsigned int *gvl*)
- *uint64xm4_t* **vwsubuwv_uint64xm4_uint32xm2** (*uint64xm4_t a*, *uint32xm2_t b*, unsigned int *gvl*)
- *uint64xm8_t* **vwsubuwv_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* **vwsubuwv_mask_uint16xm2_uint8xm1** (*uint16xm2_t merge*, *uint16xm2_t a*, *uint8xm1_t b*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vwsubuwv_mask_uint16xm4_uint8xm2** (*uint16xm4_t merge*, *uint16xm4_t a*, *uint8xm2_t b*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vwsubuwv_mask_uint16xm8_uint8xm4** (*uint16xm8_t merge*, *uint16xm8_t a*, *uint8xm4_t b*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vwsubuwv_mask_uint32xm2_uint16xm1** (*uint32xm2_t merge*, *uint32xm2_t a*, *uint16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vwsubuwv_mask_uint32xm4_uint16xm2** (*uint32xm4_t merge*, *uint32xm4_t a*, *uint16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vwsubuwv_mask_uint32xm8_uint16xm4** (*uint32xm8_t merge*, *uint32xm8_t a*, *uint16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vwsubuwv_mask_uint64xm2_uint32xm1** (*uint64xm2_t merge*, *uint64xm2_t a*, *uint32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vwsubuwv_mask_uint64xm4_uint32xm2** (*uint64xm4_t merge*, *uint64xm4_t a*, *uint32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vwsubuwv_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: ['vwsubu.wx']

Prototypes:

- *uint16xm2_t* **vwsubuwx_uint16xm2** (*uint16xm2_t a*, unsigned char *b*, unsigned int *gvl*)
- *uint16xm4_t* **vwsubuwx_uint16xm4** (*uint16xm4_t a*, unsigned char *b*, unsigned int *gvl*)
- *uint16xm8_t* **vwsubuwx_uint16xm8** (*uint16xm8_t a*, unsigned char *b*, unsigned int *gvl*)
- *uint32xm2_t* **vwsubuwx_uint32xm2** (*uint32xm2_t a*, unsigned short *b*, unsigned int *gvl*)
- *uint32xm4_t* **vwsubuwx_uint32xm4** (*uint32xm4_t a*, unsigned short *b*, unsigned int *gvl*)
- *uint32xm8_t* **vwsubuwx_uint32xm8** (*uint32xm8_t a*, unsigned short *b*, unsigned int *gvl*)
- *uint64xm2_t* **vwsubuwx_uint64xm2** (*uint64xm2_t a*, unsigned int *b*, unsigned int *gvl*)
- *uint64xm4_t* **vwsubuwx_uint64xm4** (*uint64xm4_t a*, unsigned int *b*, unsigned int *gvl*)
- *uint64xm8_t* **vwsubuwx_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* **vwsubuwx_mask_uint16xm2** (*uint16xm2_t merge*, *uint16xm2_t a*, unsigned char *b*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vwsubuwx_mask_uint16xm4** (*uint16xm4_t merge*, *uint16xm4_t a*, unsigned char *b*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vwsubuwx_mask_uint16xm8** (*uint16xm8_t merge*, *uint16xm8_t a*, unsigned char *b*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vwsubuwx_mask_uint32xm2** (*uint32xm2_t merge*, *uint32xm2_t a*, unsigned short *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vwsubuwx_mask_uint32xm4** (*uint32xm4_t merge*, *uint32xm4_t a*, unsigned short *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vwsubuwx_mask_uint32xm8** (*uint32xm8_t merge*, *uint32xm8_t a*, unsigned short *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vwsubuwx_mask_uint64xm2** (*uint64xm2_t merge*, *uint64xm2_t a*, unsigned int *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vwsubuwx_mask_uint64xm4** (*uint64xm4_t merge*, *uint64xm4_t a*, unsigned int *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vwsubuwx_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: ['vmsgtu.vi']

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: ['vmsgtu.vx']

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: ['vmsle.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* **vlbuv_uint16xm1** (const unsigned short **address*, unsigned int *gvl*)
- *uint16xm2_t* **vlbuv_uint16xm2** (const unsigned short **address*, unsigned int *gvl*)
- *uint16xm4_t* **vlbuv_uint16xm4** (const unsigned short **address*, unsigned int *gvl*)
- *uint16xm8_t* **vlbuv_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* **vlsbuv_uint16xm1** (const unsigned short *address, long *stride*, unsigned int *gvl*)
- *uint16xm2_t* **vlsbuv_uint16xm2** (const unsigned short *address, long *stride*, unsigned int *gvl*)
- *uint16xm4_t* **vlsbuv_uint16xm4** (const unsigned short *address, long *stride*, unsigned int *gvl*)
- *uint16xm8_t* **vlsbuv_uint16xm8** (const unsigned short *address, long *stride*, unsigned int *gvl*)
- *uint32xm1_t* **vlsbuv_uint32xm1** (const unsigned int *address, long *stride*, unsigned int *gvl*)
- *uint32xm2_t* **vlsbuv_uint32xm2** (const unsigned int *address, long *stride*, unsigned int *gvl*)
- *uint32xm4_t* **vlsbuv_uint32xm4** (const unsigned int *address, long *stride*, unsigned int *gvl*)
- *uint32xm8_t* **vlsbuv_uint32xm8** (const unsigned int *address, long *stride*, unsigned int *gvl*)
- *uint64xm1_t* **vlsbuv_uint64xm1** (const unsigned long *address, long *stride*, unsigned int *gvl*)

- *uint64xm2_t* **vlsbuv_uint64xm2** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- *uint64xm4_t* **vlsbuv_uint64xm4** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- *uint64xm8_t* **vlsbuv_uint64xm8** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- *uint8xm1_t* **vlsbuv_uint8xm1** (const unsigned char **address*, long *stride*, unsigned int *gvl*)
- *uint8xm2_t* **vlsbuv_uint8xm2** (const unsigned char **address*, long *stride*, unsigned int *gvl*)
- *uint8xm4_t* **vlsbuv_uint8xm4** (const unsigned char **address*, long *stride*, unsigned int *gvl*)
- *uint8xm8_t* **vlsbuv_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* **vlsbuv_mask_uint16xm1** (*uint16xm1_t merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint16xm2_t* **vlsbuv_mask_uint16xm2** (*uint16xm2_t merge*, const unsigned short **address*, long *stride*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vlsbuv_mask_uint16xm4** (*uint16xm4_t merge*, const unsigned short **address*, long *stride*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vlsbuv_mask_uint16xm8** (*uint16xm8_t merge*, const unsigned short **address*, long *stride*, *e16xm8_t mask*, unsigned int *gvl*)
- *uint32xm1_t* **vlsbuv_mask_uint32xm1** (*uint32xm1_t merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vlsbuv_mask_uint32xm2** (*uint32xm2_t merge*, const unsigned int **address*, long *stride*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vlsbuv_mask_uint32xm4** (*uint32xm4_t merge*, const unsigned int **address*, long *stride*, *e32xm4_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vlsbuv_mask_uint32xm8** (*uint32xm8_t merge*, const unsigned int **address*, long *stride*, *e32xm8_t mask*, unsigned int *gvl*)
- *uint64xm1_t* **vlsbuv_mask_uint64xm1** (*uint64xm1_t merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vlsbuv_mask_uint64xm2** (*uint64xm2_t merge*, const unsigned long **address*, long *stride*, *e64xm2_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vlsbuv_mask_uint64xm4** (*uint64xm4_t merge*, const unsigned long **address*, long *stride*, *e64xm4_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vlsbuv_mask_uint64xm8** (*uint64xm8_t merge*, const unsigned long **address*, long *stride*, *e64xm8_t mask*, unsigned int *gvl*)
- *uint8xm1_t* **vlsbuv_mask_uint8xm1** (*uint8xm1_t merge*, const unsigned char **address*, long *stride*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint8xm2_t* **vlsbuv_mask_uint8xm2** (*uint8xm2_t merge*, const unsigned char **address*, long *stride*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint8xm4_t* **vlsbuv_mask_uint8xm4** (*uint8xm4_t merge*, const unsigned char **address*, long *stride*, *e8xm4_t mask*, unsigned int *gvl*)

- *uint8xm8_t* **vlsbuv_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
```

2.9.8 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: ['vlsw.v']

Prototypes:

- *int16xm1_t* **vlswv_int16xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- *int16xm2_t* **vlswv_int16xm2** (const short **address*, long *stride*, unsigned int *gvl*)
- *int16xm4_t* **vlswv_int16xm4** (const short **address*, long *stride*, unsigned int *gvl*)
- *int16xm8_t* **vlswv_int16xm8** (const short **address*, long *stride*, unsigned int *gvl*)
- *int32xm1_t* **vlswv_int32xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- *int32xm2_t* **vlswv_int32xm2** (const int **address*, long *stride*, unsigned int *gvl*)
- *int32xm4_t* **vlswv_int32xm4** (const int **address*, long *stride*, unsigned int *gvl*)
- *int32xm8_t* **vlswv_int32xm8** (const int **address*, long *stride*, unsigned int *gvl*)
- *int64xm1_t* **vlswv_int64xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- *int64xm2_t* **vlswv_int64xm2** (const long **address*, long *stride*, unsigned int *gvl*)
- *int64xm4_t* **vlswv_int64xm4** (const long **address*, long *stride*, unsigned int *gvl*)
- *int64xm8_t* **vlswv_int64xm8** (const long **address*, long *stride*, unsigned int *gvl*)
- *int8xm1_t* **vlswv_int8xm1** (const signed char **address*, long *stride*, unsigned int *gvl*)
- *int8xm2_t* **vlswv_int8xm2** (const signed char **address*, long *stride*, unsigned int *gvl*)
- *int8xm4_t* **vlswv_int8xm4** (const signed char **address*, long *stride*, unsigned int *gvl*)
- *int8xm8_t* **vlswv_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* **vlswv_mask_int16xm1** (*int16xm1_t merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- *int16xm2_t* **vlswv_mask_int16xm2** (*int16xm2_t merge*, const short **address*, long *stride*, *e16xm2_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vlswv_mask_int16xm4** (*int16xm4_t merge*, const short **address*, long *stride*, *e16xm4_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vlswv_mask_int16xm8** (*int16xm8_t merge*, const short **address*, long *stride*, *e16xm8_t mask*, unsigned int *gvl*)
- *int32xm1_t* **vlswv_mask_int32xm1** (*int32xm1_t merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vlswv_mask_int32xm2** (*int32xm2_t merge*, const int **address*, long *stride*, *e32xm2_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vlswv_mask_int32xm4** (*int32xm4_t merge*, const int **address*, long *stride*, *e32xm4_t mask*, unsigned int *gvl*)

- *int32xm8_t* **vlswv_mask_int32xm8** (*int32xm8_t merge*, const int **address*, long *stride*, *e32xm8_t mask*, unsigned int *gvl*)
- *int64xm1_t* **vlswv_mask_int64xm1** (*int64xm1_t merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vlswv_mask_int64xm2** (*int64xm2_t merge*, const long **address*, long *stride*, *e64xm2_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vlswv_mask_int64xm4** (*int64xm4_t merge*, const long **address*, long *stride*, *e64xm4_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vlswv_mask_int64xm8** (*int64xm8_t merge*, const long **address*, long *stride*, *e64xm8_t mask*, unsigned int *gvl*)
- *int8xm1_t* **vlswv_mask_int8xm1** (*int8xm1_t merge*, const signed char **address*, long *stride*, *e8xm1_t mask*, unsigned int *gvl*)
- *int8xm2_t* **vlswv_mask_int8xm2** (*int8xm2_t merge*, const signed char **address*, long *stride*, *e8xm2_t mask*, unsigned int *gvl*)
- *int8xm4_t* **vlswv_mask_int8xm4** (*int8xm4_t merge*, const signed char **address*, long *stride*, *e8xm4_t mask*, unsigned int *gvl*)
- *int8xm8_t* **vlswv_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: ['vlswu.v']

Prototypes:

- *uint16xm1_t* **vlswuv_uint16xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- *uint16xm2_t* **vlswuv_uint16xm2** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- *uint16xm4_t* **vlswuv_uint16xm4** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- *uint16xm8_t* **vlswuv_uint16xm8** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- *uint32xm1_t* **vlswuv_uint32xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- *uint32xm2_t* **vlswuv_uint32xm2** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- *uint32xm4_t* **vlswuv_uint32xm4** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- *uint32xm8_t* **vlswuv_uint32xm8** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- *uint64xm1_t* **vlswuv_uint64xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- *uint64xm2_t* **vlswuv_uint64xm2** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- *uint64xm4_t* **vlswuv_uint64xm4** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- *uint64xm8_t* **vlswuv_uint64xm8** (const unsigned long **address*, long *stride*, unsigned int *gvl*)

- *uint8xm1_t* **vlswuv_uint8xm1** (const unsigned char **address*, long *stride*, unsigned int *gvl*)
- *uint8xm2_t* **vlswuv_uint8xm2** (const unsigned char **address*, long *stride*, unsigned int *gvl*)
- *uint8xm4_t* **vlswuv_uint8xm4** (const unsigned char **address*, long *stride*, unsigned int *gvl*)
- *uint8xm8_t* **vlswuv_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* **vlswuv_mask_uint16xm1** (*uint16xm1_t merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint16xm2_t* **vlswuv_mask_uint16xm2** (*uint16xm2_t merge*, const unsigned short **address*, long *stride*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vlswuv_mask_uint16xm4** (*uint16xm4_t merge*, const unsigned short **address*, long *stride*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vlswuv_mask_uint16xm8** (*uint16xm8_t merge*, const unsigned short **address*, long *stride*, *e16xm8_t mask*, unsigned int *gvl*)
- *uint32xm1_t* **vlswuv_mask_uint32xm1** (*uint32xm1_t merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vlswuv_mask_uint32xm2** (*uint32xm2_t merge*, const unsigned int **address*, long *stride*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vlswuv_mask_uint32xm4** (*uint32xm4_t merge*, const unsigned int **address*, long *stride*, *e32xm4_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vlswuv_mask_uint32xm8** (*uint32xm8_t merge*, const unsigned int **address*, long *stride*, *e32xm8_t mask*, unsigned int *gvl*)
- *uint64xm1_t* **vlswuv_mask_uint64xm1** (*uint64xm1_t merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vlswuv_mask_uint64xm2** (*uint64xm2_t merge*, const unsigned long **address*, long *stride*, *e64xm2_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vlswuv_mask_uint64xm4** (*uint64xm4_t merge*, const unsigned long **address*, long *stride*, *e64xm4_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vlswuv_mask_uint64xm8** (*uint64xm8_t merge*, const unsigned long **address*, long *stride*, *e64xm8_t mask*, unsigned int *gvl*)
- *uint8xm1_t* **vlswuv_mask_uint8xm1** (*uint8xm1_t merge*, const unsigned char **address*, long *stride*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint8xm2_t* **vlswuv_mask_uint8xm2** (*uint8xm2_t merge*, const unsigned char **address*, long *stride*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint8xm4_t* **vlswuv_mask_uint8xm4** (*uint8xm4_t merge*, const unsigned char **address*, long *stride*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint8xm8_t* **vlswuv_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* **vlwuv_mask_uint8xm2** (*uint8xm2_t merge*, const unsigned char **address*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint8xm4_t* **vlwuv_mask_uint8xm4** (*uint8xm4_t merge*, const unsigned char **address*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint8xm8_t* **vlwuv_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* **vlxbuv_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- *uint16xm2_t* **vlxbuv_uint16xm2** (const unsigned short **address*, *uint16xm2_t index*, unsigned int *gvl*)
- *uint16xm4_t* **vlxbuv_uint16xm4** (const unsigned short **address*, *uint16xm4_t index*, unsigned int *gvl*)
- *uint16xm8_t* **vlxbuv_uint16xm8** (const unsigned short **address*, *uint16xm8_t index*, unsigned int *gvl*)
- *uint32xm1_t* **vlxbuv_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- *uint32xm2_t* **vlxbuv_uint32xm2** (const unsigned int **address*, *uint32xm2_t index*, unsigned int *gvl*)
- *uint32xm4_t* **vlxbuv_uint32xm4** (const unsigned int **address*, *uint32xm4_t index*, unsigned int *gvl*)
- *uint32xm8_t* **vlxbuv_uint32xm8** (const unsigned int **address*, *uint32xm8_t index*, unsigned int *gvl*)
- *uint64xm1_t* **vlxbuv_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- *uint64xm2_t* **vlxbuv_uint64xm2** (const unsigned long **address*, *uint64xm2_t index*, unsigned int *gvl*)
- *uint64xm4_t* **vlxbuv_uint64xm4** (const unsigned long **address*, *uint64xm4_t index*, unsigned int *gvl*)
- *uint64xm8_t* **vlxbuv_uint64xm8** (const unsigned long **address*, *uint64xm8_t index*, unsigned int *gvl*)
- *uint8xm1_t* **vlxbuv_uint8xm1** (const unsigned char **address*, *uint8xm1_t index*, unsigned int *gvl*)
- *uint8xm2_t* **vlxbuv_uint8xm2** (const unsigned char **address*, *uint8xm2_t index*, unsigned int *gvl*)
- *uint8xm4_t* **vlxbuv_uint8xm4** (const unsigned char **address*, *uint8xm4_t index*, unsigned int *gvl*)
- *uint8xm8_t* **vlxbuv_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* **vlxbuv_mask_uint16xm1** (*uint16xm1_t merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint16xm2_t* **vlxbuv_mask_uint16xm2** (*uint16xm2_t merge*, const unsigned short **address*, *uint16xm2_t index*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vlxbuv_mask_uint16xm4** (*uint16xm4_t merge*, const unsigned short **address*, *uint16xm4_t index*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vlxbuv_mask_uint16xm8** (*uint16xm8_t merge*, const unsigned short **address*, *uint16xm8_t index*, *e16xm8_t mask*, unsigned int *gvl*)
- *uint32xm1_t* **vlxbuv_mask_uint32xm1** (*uint32xm1_t merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vlxbuv_mask_uint32xm2** (*uint32xm2_t merge*, const unsigned int **address*, *uint32xm2_t index*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vlxbuv_mask_uint32xm4** (*uint32xm4_t merge*, const unsigned int **address*, *uint32xm4_t index*, *e32xm4_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vlxbuv_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* **vlxev_float16xm1** (const float16_t **address*, *int16xm1_t index*, unsigned int *gvl*)
- *float16xm2_t* **vlxev_float16xm2** (const float16_t **address*, *int16xm2_t index*, unsigned int *gvl*)
- *float16xm4_t* **vlxev_float16xm4** (const float16_t **address*, *int16xm4_t index*, unsigned int *gvl*)
- *float16xm8_t* **vlxev_float16xm8** (const float16_t **address*, *int16xm8_t index*, unsigned int *gvl*)
- *float32xm1_t* **vlxev_float32xm1** (const float **address*, *int32xm1_t index*, unsigned int *gvl*)
- *float32xm2_t* **vlxev_float32xm2** (const float **address*, *int32xm2_t index*, unsigned int *gvl*)
- *float32xm4_t* **vlxev_float32xm4** (const float **address*, *int32xm4_t index*, unsigned int *gvl*)
- *float32xm8_t* **vlxev_float32xm8** (const float **address*, *int32xm8_t index*, unsigned int *gvl*)
- *float64xm1_t* **vlxev_float64xm1** (const double **address*, *int64xm1_t index*, unsigned int *gvl*)
- *float64xm2_t* **vlxev_float64xm2** (const double **address*, *int64xm2_t index*, unsigned int *gvl*)
- *float64xm4_t* **vlxev_float64xm4** (const double **address*, *int64xm4_t index*, unsigned int *gvl*)
- *float64xm8_t* **vlxev_float64xm8** (const double **address*, *int64xm8_t index*, unsigned int *gvl*)
- *int16xm1_t* **vlxev_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)

- *int16xm2_t* **vlxev_int16xm2** (const short **address*, *int16xm2_t index*, unsigned int *gvl*)
- *int16xm4_t* **vlxev_int16xm4** (const short **address*, *int16xm4_t index*, unsigned int *gvl*)
- *int16xm8_t* **vlxev_int16xm8** (const short **address*, *int16xm8_t index*, unsigned int *gvl*)
- *int32xm1_t* **vlxev_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- *int32xm2_t* **vlxev_int32xm2** (const int **address*, *int32xm2_t index*, unsigned int *gvl*)
- *int32xm4_t* **vlxev_int32xm4** (const int **address*, *int32xm4_t index*, unsigned int *gvl*)
- *int32xm8_t* **vlxev_int32xm8** (const int **address*, *int32xm8_t index*, unsigned int *gvl*)
- *int64xm1_t* **vlxev_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- *int64xm2_t* **vlxev_int64xm2** (const long **address*, *int64xm2_t index*, unsigned int *gvl*)
- *int64xm4_t* **vlxev_int64xm4** (const long **address*, *int64xm4_t index*, unsigned int *gvl*)
- *int64xm8_t* **vlxev_int64xm8** (const long **address*, *int64xm8_t index*, unsigned int *gvl*)
- *int8xm1_t* **vlxev_int8xm1** (const signed char **address*, *int8xm1_t index*, unsigned int *gvl*)
- *int8xm2_t* **vlxev_int8xm2** (const signed char **address*, *int8xm2_t index*, unsigned int *gvl*)
- *int8xm4_t* **vlxev_int8xm4** (const signed char **address*, *int8xm4_t index*, unsigned int *gvl*)
- *int8xm8_t* **vlxev_int8xm8** (const signed char **address*, *int8xm8_t index*, unsigned int *gvl*)
- *uint16xm1_t* **vlxev_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- *uint16xm2_t* **vlxev_uint16xm2** (const unsigned short **address*, *uint16xm2_t index*, unsigned int *gvl*)
- *uint16xm4_t* **vlxev_uint16xm4** (const unsigned short **address*, *uint16xm4_t index*, unsigned int *gvl*)
- *uint16xm8_t* **vlxev_uint16xm8** (const unsigned short **address*, *uint16xm8_t index*, unsigned int *gvl*)
- *uint32xm1_t* **vlxev_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- *uint32xm2_t* **vlxev_uint32xm2** (const unsigned int **address*, *uint32xm2_t index*, unsigned int *gvl*)
- *uint32xm4_t* **vlxev_uint32xm4** (const unsigned int **address*, *uint32xm4_t index*, unsigned int *gvl*)
- *uint32xm8_t* **vlxev_uint32xm8** (const unsigned int **address*, *uint32xm8_t index*, unsigned int *gvl*)
- *uint64xm1_t* **vlxev_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- *uint64xm2_t* **vlxev_uint64xm2** (const unsigned long **address*, *uint64xm2_t index*, unsigned int *gvl*)
- *uint64xm4_t* **vlxev_uint64xm4** (const unsigned long **address*, *uint64xm4_t index*, unsigned int *gvl*)
- *uint64xm8_t* **vlxev_uint64xm8** (const unsigned long **address*, *uint64xm8_t index*, unsigned int *gvl*)
- *uint8xm1_t* **vlxev_uint8xm1** (const unsigned char **address*, *uint8xm1_t index*, unsigned int *gvl*)
- *uint8xm2_t* **vlxev_uint8xm2** (const unsigned char **address*, *uint8xm2_t index*, unsigned int *gvl*)
- *uint8xm4_t* **vlxev_uint8xm4** (const unsigned char **address*, *uint8xm4_t index*, unsigned int *gvl*)
- *uint8xm8_t* **vlxev_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* **vlxev_mask_int64xm8** (*int64xm8_t merge*, const long **address*, *int64xm8_t index*, *e64xm8_t mask*, unsigned int *gvl*)
- *int8xm1_t* **vlxev_mask_int8xm1** (*int8xm1_t merge*, const signed char **address*, *int8xm1_t index*, *e8xm1_t mask*, unsigned int *gvl*)
- *int8xm2_t* **vlxev_mask_int8xm2** (*int8xm2_t merge*, const signed char **address*, *int8xm2_t index*, *e8xm2_t mask*, unsigned int *gvl*)
- *int8xm4_t* **vlxev_mask_int8xm4** (*int8xm4_t merge*, const signed char **address*, *int8xm4_t index*, *e8xm4_t mask*, unsigned int *gvl*)
- *int8xm8_t* **vlxev_mask_int8xm8** (*int8xm8_t merge*, const signed char **address*, *int8xm8_t index*, *e8xm8_t mask*, unsigned int *gvl*)
- *uint16xm1_t* **vlxev_mask_uint16xm1** (*uint16xm1_t merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint16xm2_t* **vlxev_mask_uint16xm2** (*uint16xm2_t merge*, const unsigned short **address*, *uint16xm2_t index*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vlxev_mask_uint16xm4** (*uint16xm4_t merge*, const unsigned short **address*, *uint16xm4_t index*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vlxev_mask_uint16xm8** (*uint16xm8_t merge*, const unsigned short **address*, *uint16xm8_t index*, *e16xm8_t mask*, unsigned int *gvl*)
- *uint32xm1_t* **vlxev_mask_uint32xm1** (*uint32xm1_t merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vlxev_mask_uint32xm2** (*uint32xm2_t merge*, const unsigned int **address*, *uint32xm2_t index*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vlxev_mask_uint32xm4** (*uint32xm4_t merge*, const unsigned int **address*, *uint32xm4_t index*, *e32xm4_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vlxev_mask_uint32xm8** (*uint32xm8_t merge*, const unsigned int **address*, *uint32xm8_t index*, *e32xm8_t mask*, unsigned int *gvl*)
- *uint64xm1_t* **vlxev_mask_uint64xm1** (*uint64xm1_t merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vlxev_mask_uint64xm2** (*uint64xm2_t merge*, const unsigned long **address*, *uint64xm2_t index*, *e64xm2_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vlxev_mask_uint64xm4** (*uint64xm4_t merge*, const unsigned long **address*, *uint64xm4_t index*, *e64xm4_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vlxev_mask_uint64xm8** (*uint64xm8_t merge*, const unsigned long **address*, *uint64xm8_t index*, *e64xm8_t mask*, unsigned int *gvl*)
- *uint8xm1_t* **vlxev_mask_uint8xm1** (*uint8xm1_t merge*, const unsigned char **address*, *uint8xm1_t index*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint8xm2_t* **vlxev_mask_uint8xm2** (*uint8xm2_t merge*, const unsigned char **address*, *uint8xm2_t index*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint8xm4_t* **vlxev_mask_uint8xm4** (*uint8xm4_t merge*, const unsigned char **address*, *uint8xm4_t index*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint8xm8_t* **vlxev_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 **vlseg2buv_uint16x2xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint16x2xm2_t **vlseg2buv_uint16x2xm2** (const unsigned short **address*, unsigned int *gvl*)
- uint16x2xm4_t **vlseg2buv_uint16x2xm4** (const unsigned short **address*, unsigned int *gvl*)
- uint32x2xm1_t **vlseg2buv_uint32x2xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint32x2xm2_t **vlseg2buv_uint32x2xm2** (const unsigned int **address*, unsigned int *gvl*)
- uint32x2xm4_t **vlseg2buv_uint32x2xm4** (const unsigned int **address*, unsigned int *gvl*)
- uint64x2xm1_t **vlseg2buv_uint64x2xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint64x2xm2_t **vlseg2buv_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: ['vlseg2wu.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 **vlseg3buv_uint16x3xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint16x3xm2_t **vlseg3buv_uint16x3xm2** (const unsigned short **address*, unsigned int *gvl*)
- uint32x3xm1_t **vlseg3buv_uint32x3xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint32x3xm2_t **vlseg3buv_uint32x3xm2** (const unsigned int **address*, unsigned int *gvl*)
- uint64x3xm1_t **vlseg3buv_uint64x3xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint64x3xm2_t **vlseg3buv_uint64x3xm2** (const unsigned long **address*, unsigned int *gvl*)
- uint8x3xm1_t **vlseg3buv_uint8x3xm1** (const unsigned char **address*, unsigned int *gvl*)
- uint8x3xm2_t **vlseg3buv_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 **vlseg3buv_mask_uint16x3xm1** (uint16x3xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint16x3xm2_t **vlseg3buv_mask_uint16x3xm2** (uint16x3xm2_t *merge*, const unsigned short **address*, *e16xm2_t mask*, unsigned int *gvl*)
- uint32x3xm1_t **vlseg3buv_mask_uint32x3xm1** (uint32x3xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint32x3xm2_t **vlseg3buv_mask_uint32x3xm2** (uint32x3xm2_t *merge*, const unsigned int **address*, *e32xm2_t mask*, unsigned int *gvl*)
- uint64x3xm1_t **vlseg3buv_mask_uint64x3xm1** (uint64x3xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- uint64x3xm2_t **vlseg3buv_mask_uint64x3xm2** (uint64x3xm2_t *merge*, const unsigned long **address*, *e64xm2_t mask*, unsigned int *gvl*)
- uint8x3xm1_t **vlseg3buv_mask_uint8x3xm1** (uint8x3xm1_t *merge*, const unsigned char **address*, *e8xm1_t mask*, unsigned int *gvl*)

- uint8x3xm2_t **vlseg3buv_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
```

2.10.10 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 **vlseg4buv_uint16x4xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint16x4xm2_t **vlseg4buv_uint16x4xm2** (const unsigned short **address*, unsigned int *gvl*)
- uint32x4xm1_t **vlseg4buv_uint32x4xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint32x4xm2_t **vlseg4buv_uint32x4xm2** (const unsigned int **address*, unsigned int *gvl*)
- uint64x4xm1_t **vlseg4buv_uint64x4xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint64x4xm2_t **vlseg4buv_uint64x4xm2** (const unsigned long **address*, unsigned int *gvl*)
- uint8x4xm1_t **vlseg4buv_uint8x4xm1** (const unsigned char **address*, unsigned int *gvl*)
- uint8x4xm2_t **vlseg4buv_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: ['vlseg4wu.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:

- int16x5xm1_t **vlseg5bv_mask_int16x5xm1** (int16x5xm1_t *merge*, const short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x5xm1_t **vlseg5bv_mask_int32x5xm1** (int32x5xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x5xm1_t **vlseg5bv_mask_int64x5xm1** (int64x5xm1_t *merge*, const long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x5xm1_t **vlseg5bv_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
```

2.10.23 Load 5 contiguous 8b unsigned fields in memory to consecutively numbered vector registers

Instruction: ['vlseg5bu.v']

Prototypes:

- uint16x5xm1_t **vlseg5buv_uint16x5xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint32x5xm1_t **vlseg5buv_uint32x5xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint64x5xm1_t **vlseg5buv_uint64x5xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint8x5xm1_t **vlseg5buv_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 **vlseg5buv_mask_uint16x5xm1** (uint16x5xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x5xm1_t **vlseg5buv_mask_uint32x5xm1** (uint32x5xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x5xm1_t **vlseg5buv_mask_uint64x5xm1** (uint64x5xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)

- uint8x5xm1_t **vlseg5buv_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
```

2.10.24 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*)

- int32x5xm1_t **vlseg5ev_mask_int32x5xm1** (int32x5xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x5xm1_t **vlseg5ev_mask_int64x5xm1** (int64x5xm1_t *merge*, const long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x5xm1_t **vlseg5ev_mask_int8x5xm1** (int8x5xm1_t *merge*, const signed char **address*, *e8xm1_t mask*, unsigned int *gvl*)
- uint16x5xm1_t **vlseg5ev_mask_uint16x5xm1** (uint16x5xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x5xm1_t **vlseg5ev_mask_uint32x5xm1** (uint32x5xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x5xm1_t **vlseg5ev_mask_uint64x5xm1** (uint64x5xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x5xm1_t **vlseg5ev_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
```

2.10.25 Load 5 contiguous 16b signed fields in memory to consecutively numbered vector registers

Instruction: ['vlseg5h.v']

Prototypes:

- int16x5xm1_t **vlseg5hv_int16x5xm1** (const short **address*, unsigned int *gvl*)
- int32x5xm1_t **vlseg5hv_int32x5xm1** (const int **address*, unsigned int *gvl*)
- int64x5xm1_t **vlseg5hv_int64x5xm1** (const long **address*, unsigned int *gvl*)
- int8x5xm1_t **vlseg5hv_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:

- int16x5xm1_t **vlseg5hv_mask_int16x5xm1** (int16x5xm1_t *merge*, const short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x5xm1_t **vlseg5hv_mask_int32x5xm1** (int32x5xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x5xm1_t **vlseg5hv_mask_int64x5xm1** (int64x5xm1_t *merge*, const long **address*, *e64xm1_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:

- int16x5xm1_t **vlseg5wv_int16x5xm1** (const short **address*, unsigned int *gvl*)
- int32x5xm1_t **vlseg5wv_int32x5xm1** (const int **address*, unsigned int *gvl*)
- int64x5xm1_t **vlseg5wv_int64x5xm1** (const long **address*, unsigned int *gvl*)
- int8x5xm1_t **vlseg5wv_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:

- int16x5xm1_t **vlseg5wv_mask_int16x5xm1** (int16x5xm1_t *merge*, const short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x5xm1_t **vlseg5wv_mask_int32x5xm1** (int32x5xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x5xm1_t **vlseg5wv_mask_int64x5xm1** (int64x5xm1_t *merge*, const long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x5xm1_t **vlseg5wv_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
```

2.10.28 Load 5 contiguous 32b unsigned fields in memory to consecutively numbered vector registers

Instruction: ['vlseg5wu.v']

Prototypes:

- uint16x5xm1_t **vlseg5wuv_uint16x5xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint32x5xm1_t **vlseg5wuv_uint32x5xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint64x5xm1_t **vlseg5wuv_uint64x5xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint8x5xm1_t **vlseg5wuv_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 **vlseg5wuv_mask_uint16x5xm1** (uint16x5xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x5xm1_t **vlseg5wuv_mask_uint32x5xm1** (uint32x5xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x5xm1_t **vlseg5wuv_mask_uint64x5xm1** (uint64x5xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x5xm1_t **vlseg5wuv_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
```

2.10.29 Load 6 contiguous 8b signed fields in memory to consecutively numbered vector registers

Instruction: ['vlseg6b.v']

Prototypes:

- int16x6xm1_t **vlseg6bv_int16x6xm1** (const short **address*, unsigned int *gvl*)
- int32x6xm1_t **vlseg6bv_int32x6xm1** (const int **address*, unsigned int *gvl*)
- int64x6xm1_t **vlseg6bv_int64x6xm1** (const long **address*, unsigned int *gvl*)
- int8x6xm1_t **vlseg6bv_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 **vlseg6bv_mask_int16x6xm1** (int16x6xm1_t *merge*, const short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x6xm1_t **vlseg6bv_mask_int32x6xm1** (int32x6xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x6xm1_t **vlseg6bv_mask_int64x6xm1** (int64x6xm1_t *merge*, const long **address*, *e64xm1_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:

- uint16x6xm1_t **vlseg6huv_uint16x6xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint32x6xm1_t **vlseg6huv_uint32x6xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint64x6xm1_t **vlseg6huv_uint64x6xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint8x6xm1_t **vlseg6huv_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 **vlseg6huv_mask_uint16x6xm1** (uint16x6xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x6xm1_t **vlseg6huv_mask_uint32x6xm1** (uint32x6xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x6xm1_t **vlseg6huv_mask_uint64x6xm1** (uint64x6xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x6xm1_t **vlseg6huv_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
```

2.10.34 Load 6 contiguous 32b signed fields in memory to consecutively numbered vector registers

Instruction: ['vlseg6w.v']

Prototypes:

- int16x6xm1_t **vlseg6wv_int16x6xm1** (const short **address*, unsigned int *gvl*)
- int32x6xm1_t **vlseg6wv_int32x6xm1** (const int **address*, unsigned int *gvl*)
- int64x6xm1_t **vlseg6wv_int64x6xm1** (const long **address*, unsigned int *gvl*)
- int8x6xm1_t **vlseg6wv_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 **vlseg6wv_mask_int16x6xm1** (int16x6xm1_t *merge*, const short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x6xm1_t **vlseg6wv_mask_int32x6xm1** (int32x6xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x6xm1_t **vlseg6wv_mask_int64x6xm1** (int64x6xm1_t *merge*, const long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x6xm1_t **vlseg6wv_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
```

2.10.35 Load 6 contiguous 32b unsigned fields in memory to consecutively numbered vector registers

Instruction: ['vlseg6wu.v']

Prototypes:

- uint16x6xm1_t **vlseg6wuv_uint16x6xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint32x6xm1_t **vlseg6wuv_uint32x6xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint64x6xm1_t **vlseg6wuv_uint64x6xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint8x6xm1_t **vlseg6wuv_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 **vlseg6wuv_mask_uint16x6xm1** (uint16x6xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x6xm1_t **vlseg6wuv_mask_uint32x6xm1** (uint32x6xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x6xm1_t **vlseg6wuv_mask_uint64x6xm1** (uint64x6xm1_t *merge*, const unsigned long **address*, *e64xm1_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:

- uint16x7xm1_t **vlseg7buv_uint16x7xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint32x7xm1_t **vlseg7buv_uint32x7xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint64x7xm1_t **vlseg7buv_uint64x7xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint8x7xm1_t **vlseg7buv_uint8x7xm1** (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:

- uint16x7xm1_t **vlseg7buv_mask_uint16x7xm1** (uint16x7xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x7xm1_t **vlseg7buv_mask_uint32x7xm1** (uint32x7xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x7xm1_t **vlseg7buv_mask_uint64x7xm1** (uint64x7xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x7xm1_t **vlseg7buv_mask_uint8x7xm1** (uint8x7xm1_t *merge*, const unsigned 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
```

2.10.38 Load 7 contiguous element fields in memory to consecutively numbered vector registers

Instruction: ['vlseg7e.v']

Prototypes:

- float16x7xm1_t **vlseg7ev_float16x7xm1** (const float16_t **address*, unsigned int *gvl*)
- float32x7xm1_t **vlseg7ev_float32x7xm1** (const float **address*, unsigned int *gvl*)
- float64x7xm1_t **vlseg7ev_float64x7xm1** (const double **address*, unsigned int *gvl*)
- int16x7xm1_t **vlseg7ev_int16x7xm1** (const short **address*, unsigned int *gvl*)
- int32x7xm1_t **vlseg7ev_int32x7xm1** (const int **address*, unsigned int *gvl*)

- int64x7xm1_t **vlseg7ev_int64x7xm1** (const long **address*, unsigned int *gvl*)
- int8x7xm1_t **vlseg7ev_int8x7xm1** (const signed char **address*, unsigned int *gvl*)
- uint16x7xm1_t **vlseg7ev_uint16x7xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint32x7xm1_t **vlseg7ev_uint32x7xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint64x7xm1_t **vlseg7ev_uint64x7xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint8x7xm1_t **vlseg7ev_uint8x7xm1** (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:

- float16x7xm1_t **vlseg7ev_mask_float16x7xm1** (float16x7xm1_t *merge*, const float16_t **address*, *e16xm1_t mask*, unsigned int *gvl*)
- float32x7xm1_t **vlseg7ev_mask_float32x7xm1** (float32x7xm1_t *merge*, const float **address*, *e32xm1_t mask*, unsigned int *gvl*)
- float64x7xm1_t **vlseg7ev_mask_float64x7xm1** (float64x7xm1_t *merge*, const double **address*, *e64xm1_t mask*, unsigned int *gvl*)
- int16x7xm1_t **vlseg7ev_mask_int16x7xm1** (int16x7xm1_t *merge*, const short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x7xm1_t **vlseg7ev_mask_int32x7xm1** (int32x7xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x7xm1_t **vlseg7ev_mask_int64x7xm1** (int64x7xm1_t *merge*, const long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x7xm1_t **vlseg7ev_mask_int8x7xm1** (int8x7xm1_t *merge*, const signed char **address*, *e8xm1_t mask*, unsigned int *gvl*)
- uint16x7xm1_t **vlseg7ev_mask_uint16x7xm1** (uint16x7xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x7xm1_t **vlseg7ev_mask_uint32x7xm1** (uint32x7xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x7xm1_t **vlseg7ev_mask_uint64x7xm1** (uint64x7xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x7xm1_t **vlseg7ev_mask_uint8x7xm1** (uint8x7xm1_t *merge*, const unsigned 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
```

2.10.39 Load 7 contiguous 16b signed fields in memory to consecutively numbered vector registers

Instruction: ['vlseg7h.v']

Prototypes:

- int16x7xm1_t **vlseg7hv_int16x7xm1** (const short **address*, unsigned int *gvl*)
- int32x7xm1_t **vlseg7hv_int32x7xm1** (const int **address*, unsigned int *gvl*)
- int64x7xm1_t **vlseg7hv_int64x7xm1** (const long **address*, unsigned int *gvl*)
- int8x7xm1_t **vlseg7hv_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 **vlseg7hv_mask_int16x7xm1** (int16x7xm1_t *merge*, const short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x7xm1_t **vlseg7hv_mask_int32x7xm1** (int32x7xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x7xm1_t **vlseg7hv_mask_int64x7xm1** (int64x7xm1_t *merge*, const long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x7xm1_t **vlseg7hv_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
```

2.10.40 Load 7 contiguous 16b unsigned fields in memory to consecutively numbered vector registers

Instruction: ['vlseg7hu.v']

Prototypes:

- uint16x7xm1_t **vlseg7huv_uint16x7xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint32x7xm1_t **vlseg7huv_uint32x7xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint64x7xm1_t **vlseg7huv_uint64x7xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint8x7xm1_t **vlseg7huv_uint8x7xm1** (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:

- uint16x7xm1_t **vlseg7huv_mask_uint16x7xm1** (uint16x7xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x7xm1_t **vlseg7huv_mask_uint32x7xm1** (uint32x7xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x7xm1_t **vlseg7huv_mask_uint64x7xm1** (uint64x7xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x7xm1_t **vlseg7huv_mask_uint8x7xm1** (uint8x7xm1_t *merge*, const unsigned 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
```

2.10.41 Load 7 contiguous 32b signed fields in memory to consecutively numbered vector registers

Instruction: ['vlseg7w.v']

Prototypes:

- int16x7xm1_t **vlseg7wv_int16x7xm1** (const short **address*, unsigned int *gvl*)
- int32x7xm1_t **vlseg7wv_int32x7xm1** (const int **address*, unsigned int *gvl*)
- int64x7xm1_t **vlseg7wv_int64x7xm1** (const long **address*, unsigned int *gvl*)
- int8x7xm1_t **vlseg7wv_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 **vlseg7wv_mask_int16x7xm1** (int16x7xm1_t *merge*, const short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x7xm1_t **vlseg7wv_mask_int32x7xm1** (int32x7xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x7xm1_t **vlseg7wv_mask_int64x7xm1** (int64x7xm1_t *merge*, const long **address*, *e64xm1_t mask*, unsigned int *gvl*)

- int8x7xm1_t **vlseg7wv_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
```

2.10.42 Load 7 contiguous 32b unsigned fields in memory to consecutively numbered vector registers

Instruction: ['vlseg7wu.v']

Prototypes:

- uint16x7xm1_t **vlseg7wuv_uint16x7xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint32x7xm1_t **vlseg7wuv_uint32x7xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint64x7xm1_t **vlseg7wuv_uint64x7xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint8x7xm1_t **vlseg7wuv_uint8x7xm1** (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:

- uint16x7xm1_t **vlseg7wuv_mask_uint16x7xm1** (uint16x7xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x7xm1_t **vlseg7wuv_mask_uint32x7xm1** (uint32x7xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x7xm1_t **vlseg7wuv_mask_uint64x7xm1** (uint64x7xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x7xm1_t **vlseg7wuv_mask_uint8x7xm1** (uint8x7xm1_t *merge*, const unsigned 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
```

2.10.43 Load 8 contiguous 8b signed fields in memory to consecutively numbered vector registers

Instruction: ['vlseg8b.v']

Prototypes:

- int16x8xm1_t **vlseg8bv_int16x8xm1** (const short **address*, unsigned int *gvl*)
- int32x8xm1_t **vlseg8bv_int32x8xm1** (const int **address*, unsigned int *gvl*)
- int64x8xm1_t **vlseg8bv_int64x8xm1** (const long **address*, unsigned int *gvl*)
- int8x8xm1_t **vlseg8bv_int8x8xm1** (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:

- int16x8xm1_t **vlseg8bv_mask_int16x8xm1** (int16x8xm1_t *merge*, const short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x8xm1_t **vlseg8bv_mask_int32x8xm1** (int32x8xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x8xm1_t **vlseg8bv_mask_int64x8xm1** (int64x8xm1_t *merge*, const long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x8xm1_t **vlseg8bv_mask_int8x8xm1** (int8x8xm1_t *merge*, const signed char **address*, *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)
      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:

- uint16x8xm1_t **vlseg8buv_uint16x8xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint32x8xm1_t **vlseg8buv_uint32x8xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint64x8xm1_t **vlseg8buv_uint64x8xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint8x8xm1_t **vlseg8buv_uint8x8xm1** (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:

- uint16x8xm1_t **vlseg8buv_mask_uint16x8xm1** (uint16x8xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x8xm1_t **vlseg8buv_mask_uint32x8xm1** (uint32x8xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x8xm1_t **vlseg8buv_mask_uint64x8xm1** (uint64x8xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x8xm1_t **vlseg8buv_mask_uint8x8xm1** (uint8x8xm1_t *merge*, const unsigned char **address*, *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)
      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:

- float16x8xm1_t **vlseg8ev_float16x8xm1** (const float16_t **address*, unsigned int *gvl*)
- float32x8xm1_t **vlseg8ev_float32x8xm1** (const float **address*, unsigned int *gvl*)
- float64x8xm1_t **vlseg8ev_float64x8xm1** (const double **address*, unsigned int *gvl*)
- int16x8xm1_t **vlseg8ev_int16x8xm1** (const short **address*, unsigned int *gvl*)
- int32x8xm1_t **vlseg8ev_int32x8xm1** (const int **address*, unsigned int *gvl*)
- int64x8xm1_t **vlseg8ev_int64x8xm1** (const long **address*, unsigned int *gvl*)
- int8x8xm1_t **vlseg8ev_int8x8xm1** (const signed char **address*, unsigned int *gvl*)
- uint16x8xm1_t **vlseg8ev_uint16x8xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint32x8xm1_t **vlseg8ev_uint32x8xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint64x8xm1_t **vlseg8ev_uint64x8xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint8x8xm1_t **vlseg8ev_uint8x8xm1** (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:

- float16x8xm1_t **vlseg8ev_mask_float16x8xm1** (float16x8xm1_t *merge*, const float16_t **address*, *e16xm1_t mask*, unsigned int *gvl*)
- float32x8xm1_t **vlseg8ev_mask_float32x8xm1** (float32x8xm1_t *merge*, const float **address*, *e32xm1_t mask*, unsigned int *gvl*)
- float64x8xm1_t **vlseg8ev_mask_float64x8xm1** (float64x8xm1_t *merge*, const double **address*, *e64xm1_t mask*, unsigned int *gvl*)
- int16x8xm1_t **vlseg8ev_mask_int16x8xm1** (int16x8xm1_t *merge*, const short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x8xm1_t **vlseg8ev_mask_int32x8xm1** (int32x8xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x8xm1_t **vlseg8ev_mask_int64x8xm1** (int64x8xm1_t *merge*, const long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x8xm1_t **vlseg8ev_mask_int8x8xm1** (int8x8xm1_t *merge*, const signed char **address*, *e8xm1_t mask*, unsigned int *gvl*)
- uint16x8xm1_t **vlseg8ev_mask_uint16x8xm1** (uint16x8xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x8xm1_t **vlseg8ev_mask_uint32x8xm1** (uint32x8xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x8xm1_t **vlseg8ev_mask_uint64x8xm1** (uint64x8xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x8xm1_t **vlseg8ev_mask_uint8x8xm1** (uint8x8xm1_t *merge*, const unsigned char **address*, *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)
      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:

- int16x8xm1_t **vlseg8hv_int16x8xm1** (const short **address*, unsigned int *gvl*)
- int32x8xm1_t **vlseg8hv_int32x8xm1** (const int **address*, unsigned int *gvl*)
- int64x8xm1_t **vlseg8hv_int64x8xm1** (const long **address*, unsigned int *gvl*)

- int8x8xm1_t **vlseg8hv_int8x8xm1** (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:

- int16x8xm1_t **vlseg8hv_mask_int16x8xm1** (int16x8xm1_t *merge*, const short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x8xm1_t **vlseg8hv_mask_int32x8xm1** (int32x8xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x8xm1_t **vlseg8hv_mask_int64x8xm1** (int64x8xm1_t *merge*, const long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x8xm1_t **vlseg8hv_mask_int8x8xm1** (int8x8xm1_t *merge*, const signed char **address*, *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)
      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:

- uint16x8xm1_t **vlseg8huv_uint16x8xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint32x8xm1_t **vlseg8huv_uint32x8xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint64x8xm1_t **vlseg8huv_uint64x8xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint8x8xm1_t **vlseg8huv_uint8x8xm1** (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:

- uint16x8xm1_t **vlseg8huv_mask_uint16x8xm1** (uint16x8xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x8xm1_t **vlseg8huv_mask_uint32x8xm1** (uint32x8xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)

- uint64x8xm1_t **vlseg8huv_mask_uint64x8xm1** (uint64x8xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x8xm1_t **vlseg8huv_mask_uint8x8xm1** (uint8x8xm1_t *merge*, const unsigned char **address*, *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)
      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:

- int16x8xm1_t **vlseg8wv_int16x8xm1** (const short **address*, unsigned int *gvl*)
- int32x8xm1_t **vlseg8wv_int32x8xm1** (const int **address*, unsigned int *gvl*)
- int64x8xm1_t **vlseg8wv_int64x8xm1** (const long **address*, unsigned int *gvl*)
- int8x8xm1_t **vlseg8wv_int8x8xm1** (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:

- int16x8xm1_t **vlseg8wv_mask_int16x8xm1** (int16x8xm1_t *merge*, const short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x8xm1_t **vlseg8wv_mask_int32x8xm1** (int32x8xm1_t *merge*, const int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x8xm1_t **vlseg8wv_mask_int64x8xm1** (int64x8xm1_t *merge*, const long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x8xm1_t **vlseg8wv_mask_int8x8xm1** (int8x8xm1_t *merge*, const signed char **address*, *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)
      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:

- uint16x8xm1_t **vlseg8wuv_uint16x8xm1** (const unsigned short **address*, unsigned int *gvl*)
- uint32x8xm1_t **vlseg8wuv_uint32x8xm1** (const unsigned int **address*, unsigned int *gvl*)
- uint64x8xm1_t **vlseg8wuv_uint64x8xm1** (const unsigned long **address*, unsigned int *gvl*)
- uint8x8xm1_t **vlseg8wuv_uint8x8xm1** (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:

- uint16x8xm1_t **vlseg8wuv_mask_uint16x8xm1** (uint16x8xm1_t *merge*, const unsigned short **address*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x8xm1_t **vlseg8wuv_mask_uint32x8xm1** (uint32x8xm1_t *merge*, const unsigned int **address*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x8xm1_t **vlseg8wuv_mask_uint64x8xm1** (uint64x8xm1_t *merge*, const unsigned long **address*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x8xm1_t **vlseg8wuv_mask_uint8x8xm1** (uint8x8xm1_t *merge*, const unsigned char **address*, *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)
      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:

- int16x2xm1_t **vlsseg2bv_int16x2xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int16x2xm2_t **vlsseg2bv_int16x2xm2** (const short **address*, long *stride*, unsigned int *gvl*)
- int16x2xm4_t **vlsseg2bv_int16x2xm4** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x2xm1_t **vlsseg2bv_int32x2xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int32x2xm2_t **vlsseg2bv_int32x2xm2** (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 **vlsseg2buv_uint16x2xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint16x2xm2_t **vlsseg2buv_uint16x2xm2** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint16x2xm4_t **vlsseg2buv_uint16x2xm4** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x2xm1_t **vlsseg2buv_uint32x2xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint32x2xm2_t **vlsseg2buv_uint32x2xm2** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint32x2xm4_t **vlsseg2buv_uint32x2xm4** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x2xm1_t **vlsseg2buv_uint64x2xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint64x2xm2_t **vlsseg2buv_uint64x2xm2** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint64x2xm4_t **vlsseg2buv_uint64x2xm4** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x2xm1_t **vlsseg2buv_uint8x2xm1** (const unsigned char **address*, long *stride*, unsigned int *gvl*)
- uint8x2xm2_t **vlsseg2buv_uint8x2xm2** (const unsigned char **address*, long *stride*, unsigned int *gvl*)
- uint8x2xm4_t **vlsseg2buv_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 **vlsseg2buv_mask_uint16x2xm1** (uint16x2xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint16x2xm2_t **vlsseg2buv_mask_uint16x2xm2** (uint16x2xm2_t *merge*, const unsigned short **address*, long *stride*, *e16xm2_t mask*, unsigned int *gvl*)

- uint16x2xm4_t **vlsseg2buv_mask_uint16x2xm4** (uint16x2xm4_t *merge*, const unsigned short **address*, long *stride*, *e16xm4_t mask*, unsigned int *gvl*)
- uint32x2xm1_t **vlsseg2buv_mask_uint32x2xm1** (uint32x2xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint32x2xm2_t **vlsseg2buv_mask_uint32x2xm2** (uint32x2xm2_t *merge*, const unsigned int **address*, long *stride*, *e32xm2_t mask*, unsigned int *gvl*)
- uint32x2xm4_t **vlsseg2buv_mask_uint32x2xm4** (uint32x2xm4_t *merge*, const unsigned int **address*, long *stride*, *e32xm4_t mask*, unsigned int *gvl*)
- uint64x2xm1_t **vlsseg2buv_mask_uint64x2xm1** (uint64x2xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint64x2xm2_t **vlsseg2buv_mask_uint64x2xm2** (uint64x2xm2_t *merge*, const unsigned long **address*, long *stride*, *e64xm2_t mask*, unsigned int *gvl*)
- uint64x2xm4_t **vlsseg2buv_mask_uint64x2xm4** (uint64x2xm4_t *merge*, const unsigned long **address*, long *stride*, *e64xm4_t mask*, unsigned int *gvl*)
- uint8x2xm1_t **vlsseg2buv_mask_uint8x2xm1** (uint8x2xm1_t *merge*, const unsigned char **address*, long *stride*, *e8xm1_t mask*, unsigned int *gvl*)
- uint8x2xm2_t **vlsseg2buv_mask_uint8x2xm2** (uint8x2xm2_t *merge*, const unsigned char **address*, long *stride*, *e8xm2_t mask*, unsigned int *gvl*)
- uint8x2xm4_t **vlsseg2buv_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
```

2.10.52 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: ['vlsseg2wu.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 **vlsseg3buv_uint16x3xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint16x3xm2_t **vlsseg3buv_uint16x3xm2** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x3xm1_t **vlsseg3buv_uint32x3xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint32x3xm2_t **vlsseg3buv_uint32x3xm2** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x3xm1_t **vlsseg3buv_uint64x3xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint64x3xm2_t **vlsseg3buv_uint64x3xm2** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x3xm1_t **vlsseg3buv_uint8x3xm1** (const unsigned char **address*, long *stride*, unsigned int *gvl*)
- uint8x3xm2_t **vlsseg3buv_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 **vlsseg3buv_mask_uint16x3xm1** (uint16x3xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint16x3xm2_t **vlsseg3buv_mask_uint16x3xm2** (uint16x3xm2_t *merge*, const unsigned short **address*, long *stride*, *e16xm2_t mask*, unsigned int *gvl*)
- uint32x3xm1_t **vlsseg3buv_mask_uint32x3xm1** (uint32x3xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint32x3xm2_t **vlsseg3buv_mask_uint32x3xm2** (uint32x3xm2_t *merge*, const unsigned int **address*, long *stride*, *e32xm2_t mask*, unsigned int *gvl*)
- uint64x3xm1_t **vlsseg3buv_mask_uint64x3xm1** (uint64x3xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint64x3xm2_t **vlsseg3buv_mask_uint64x3xm2** (uint64x3xm2_t *merge*, const unsigned long **address*, long *stride*, *e64xm2_t mask*, unsigned int *gvl*)

- uint8x3xm1_t **vlsseg3buv_mask_uint8x3xm1** (uint8x3xm1_t *merge*, const unsigned char **address*, long *stride*, *e8xm1_t mask*, unsigned int *gvl*)
- uint8x3xm2_t **vlsseg3buv_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
```

2.10.59 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: ['vlsseg4wu.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:

- uint16x5xm1_t **vlsseg5buv_uint16x5xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x5xm1_t **vlsseg5buv_uint32x5xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x5xm1_t **vlsseg5buv_uint64x5xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x5xm1_t **vlsseg5buv_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:

- uint16x5xm1_t **vlsseg5buv_mask_uint16x5xm1** (uint16x5xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x5xm1_t **vlsseg5buv_mask_uint32x5xm1** (uint32x5xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x5xm1_t **vlsseg5buv_mask_uint64x5xm1** (uint64x5xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x5xm1_t **vlsseg5buv_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
```

2.10.73 Load 5 contiguous element fields in memory(strided) to consecutively numbered vector registers

Instruction: ['vlsseg5e.v']

Prototypes:

- float16x5xm1_t **vlsseg5ev_float16x5xm1** (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 float16_t **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:

- uint16x5xm1_t **vlsseg5huv_uint16x5xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x5xm1_t **vlsseg5huv_uint32x5xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x5xm1_t **vlsseg5huv_uint64x5xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x5xm1_t **vlsseg5huv_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:

- uint16x5xm1_t **vlsseg5huv_mask_uint16x5xm1** (uint16x5xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x5xm1_t **vlsseg5huv_mask_uint32x5xm1** (uint32x5xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x5xm1_t **vlsseg5huv_mask_uint64x5xm1** (uint64x5xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x5xm1_t **vlsseg5huv_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
```

2.10.76 Load 5 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers

Instruction: ['vlsseg5w.v']

Prototypes:

- int16x5xm1_t **vlsseg5wv_int16x5xm1** (const short **address*, long *stride*, unsigned int *gvl*)

- int32x5xm1_t **vlsseg5wv_int32x5xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x5xm1_t **vlsseg5wv_int64x5xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x5xm1_t **vlsseg5wv_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 **vlsseg5wv_mask_int16x5xm1** (int16x5xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x5xm1_t **vlsseg5wv_mask_int32x5xm1** (int32x5xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x5xm1_t **vlsseg5wv_mask_int64x5xm1** (int64x5xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x5xm1_t **vlsseg5wv_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
```

2.10.77 Load 5 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers

Instruction: ['vlsseg5wu.v']

Prototypes:

- uint16x5xm1_t **vlsseg5wuv_uint16x5xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x5xm1_t **vlsseg5wuv_uint32x5xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x5xm1_t **vlsseg5wuv_uint64x5xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x5xm1_t **vlsseg5wuv_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:

- uint16x5xm1_t **vlsseg5wuv_mask_uint16x5xm1** (uint16x5xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x5xm1_t **vlsseg5wuv_mask_uint32x5xm1** (uint32x5xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x5xm1_t **vlsseg5wuv_mask_uint64x5xm1** (uint64x5xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x5xm1_t **vlsseg5wuv_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
```

2.10.78 Load 6 contiguous 8b signed fields in memory(strided) to consecutively numbered vector registers

Instruction: ['vlsseg6b.v']

Prototypes:

- int16x6xm1_t **vlsseg6bv_int16x6xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x6xm1_t **vlsseg6bv_int32x6xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x6xm1_t **vlsseg6bv_int64x6xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x6xm1_t **vlsseg6bv_int8x6xm1** (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:

- int16x6xm1_t **vlsseg6bv_mask_int16x6xm1** (int16x6xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x6xm1_t **vlsseg6bv_mask_int32x6xm1** (int32x6xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x6xm1_t **vlsseg6bv_mask_int64x6xm1** (int64x6xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x6xm1_t **vlsseg6bv_mask_int8x6xm1** (int8x6xm1_t *merge*, const signed char **address*, long *stride*, *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) + 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:

- uint16x6xm1_t **vlsseg6buv_uint16x6xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x6xm1_t **vlsseg6buv_uint32x6xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x6xm1_t **vlsseg6buv_uint64x6xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x6xm1_t **vlsseg6buv_uint8x6xm1** (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:

- uint16x6xm1_t **vlsseg6buv_mask_uint16x6xm1** (uint16x6xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x6xm1_t **vlsseg6buv_mask_uint32x6xm1** (uint32x6xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x6xm1_t **vlsseg6buv_mask_uint64x6xm1** (uint64x6xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x6xm1_t **vlsseg6buv_mask_uint8x6xm1** (uint8x6xm1_t *merge*, const unsigned char **address*, long *stride*, *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) + 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:

- float16x6xm1_t **vlsseg6ev_float16x6xm1** (const float16_t **address*, long *stride*, unsigned int *gvl*)
- float32x6xm1_t **vlsseg6ev_float32x6xm1** (const float **address*, long *stride*, unsigned int *gvl*)
- float64x6xm1_t **vlsseg6ev_float64x6xm1** (const double **address*, long *stride*, unsigned int *gvl*)
- int16x6xm1_t **vlsseg6ev_int16x6xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x6xm1_t **vlsseg6ev_int32x6xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x6xm1_t **vlsseg6ev_int64x6xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x6xm1_t **vlsseg6ev_int8x6xm1** (const signed char **address*, long *stride*, unsigned int *gvl*)
- uint16x6xm1_t **vlsseg6ev_uint16x6xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x6xm1_t **vlsseg6ev_uint32x6xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x6xm1_t **vlsseg6ev_uint64x6xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x6xm1_t **vlsseg6ev_uint8x6xm1** (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:

- float16x6xm1_t **vlsseg6ev_mask_float16x6xm1** (float16x6xm1_t *merge*, const float16_t **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- float32x6xm1_t **vlsseg6ev_mask_float32x6xm1** (float32x6xm1_t *merge*, const float **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- float64x6xm1_t **vlsseg6ev_mask_float64x6xm1** (float64x6xm1_t *merge*, const double **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int16x6xm1_t **vlsseg6ev_mask_int16x6xm1** (int16x6xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x6xm1_t **vlsseg6ev_mask_int32x6xm1** (int32x6xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x6xm1_t **vlsseg6ev_mask_int64x6xm1** (int64x6xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)

- int8x6xm1_t **vlsseg6ev_mask_int8x6xm1** (int8x6xm1_t *merge*, const signed char **address*, long *stride*, *e8xm1_t mask*, unsigned int *gvl*)
- uint16x6xm1_t **vlsseg6ev_mask_uint16x6xm1** (uint16x6xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x6xm1_t **vlsseg6ev_mask_uint32x6xm1** (uint32x6xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x6xm1_t **vlsseg6ev_mask_uint64x6xm1** (uint64x6xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x6xm1_t **vlsseg6ev_mask_uint8x6xm1** (uint8x6xm1_t *merge*, const unsigned char **address*, long *stride*, *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) + 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:

- int16x6xm1_t **vlsseg6hv_int16x6xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x6xm1_t **vlsseg6hv_int32x6xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x6xm1_t **vlsseg6hv_int64x6xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x6xm1_t **vlsseg6hv_int8x6xm1** (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:

- int16x6xm1_t **vlsseg6hv_mask_int16x6xm1** (int16x6xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x6xm1_t **vlsseg6hv_mask_int32x6xm1** (int32x6xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x6xm1_t **vlsseg6hv_mask_int64x6xm1** (int64x6xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x6xm1_t **vlsseg6hv_mask_int8x6xm1** (int8x6xm1_t *merge*, const signed char **address*, long *stride*, *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) + 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:

- uint16x6xm1_t **vlsseg6huv_uint16x6xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x6xm1_t **vlsseg6huv_uint32x6xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x6xm1_t **vlsseg6huv_uint64x6xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x6xm1_t **vlsseg6huv_uint8x6xm1** (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:

- uint16x6xm1_t **vlsseg6huv_mask_uint16x6xm1** (uint16x6xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x6xm1_t **vlsseg6huv_mask_uint32x6xm1** (uint32x6xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x6xm1_t **vlsseg6huv_mask_uint64x6xm1** (uint64x6xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x6xm1_t **vlsseg6huv_mask_uint8x6xm1** (uint8x6xm1_t *merge*, const unsigned char **address*, long *stride*, *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) + 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:

- int16x6xm1_t **vlsseg6wv_int16x6xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x6xm1_t **vlsseg6wv_int32x6xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x6xm1_t **vlsseg6wv_int64x6xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x6xm1_t **vlsseg6wv_int8x6xm1** (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:

- int16x6xm1_t **vlsseg6wv_mask_int16x6xm1** (int16x6xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x6xm1_t **vlsseg6wv_mask_int32x6xm1** (int32x6xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x6xm1_t **vlsseg6wv_mask_int64x6xm1** (int64x6xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x6xm1_t **vlsseg6wv_mask_int8x6xm1** (int8x6xm1_t *merge*, const signed char **address*, long *stride*, *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) + 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:

- uint16x6xm1_t **vlsseg6wuv_uint16x6xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)

- uint32x6xm1_t **vlsseg6wuv_uint32x6xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x6xm1_t **vlsseg6wuv_uint64x6xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x6xm1_t **vlsseg6wuv_uint8x6xm1** (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:

- uint16x6xm1_t **vlsseg6wuv_mask_uint16x6xm1** (uint16x6xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x6xm1_t **vlsseg6wuv_mask_uint32x6xm1** (uint32x6xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x6xm1_t **vlsseg6wuv_mask_uint64x6xm1** (uint64x6xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x6xm1_t **vlsseg6wuv_mask_uint8x6xm1** (uint8x6xm1_t *merge*, const unsigned char **address*, long *stride*, *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) + 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:

- int16x7xm1_t **vlsseg7bv_int16x7xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x7xm1_t **vlsseg7bv_int32x7xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x7xm1_t **vlsseg7bv_int64x7xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x7xm1_t **vlsseg7bv_int8x7xm1** (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:

- int16x7xm1_t **vlsseg7bv_mask_int16x7xm1** (int16x7xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x7xm1_t **vlsseg7bv_mask_int32x7xm1** (int32x7xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x7xm1_t **vlsseg7bv_mask_int64x7xm1** (int64x7xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x7xm1_t **vlsseg7bv_mask_int8x7xm1** (int8x7xm1_t *merge*, const signed char **address*, long *stride*, *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) + 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:

- uint16x7xm1_t **vlsseg7buv_uint16x7xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x7xm1_t **vlsseg7buv_uint32x7xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x7xm1_t **vlsseg7buv_uint64x7xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x7xm1_t **vlsseg7buv_uint8x7xm1** (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:

- uint16x7xm1_t **vlsseg7buv_mask_uint16x7xm1** (uint16x7xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)

- uint32x7xm1_t **vlsseg7buv_mask_uint32x7xm1** (uint32x7xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x7xm1_t **vlsseg7buv_mask_uint64x7xm1** (uint64x7xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x7xm1_t **vlsseg7buv_mask_uint8x7xm1** (uint8x7xm1_t *merge*, const unsigned char **address*, long *stride*, *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) + 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:

- float16x7xm1_t **vlsseg7ev_float16x7xm1** (const float16_t **address*, long *stride*, unsigned int *gvl*)
- float32x7xm1_t **vlsseg7ev_float32x7xm1** (const float **address*, long *stride*, unsigned int *gvl*)
- float64x7xm1_t **vlsseg7ev_float64x7xm1** (const double **address*, long *stride*, unsigned int *gvl*)
- int16x7xm1_t **vlsseg7ev_int16x7xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x7xm1_t **vlsseg7ev_int32x7xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x7xm1_t **vlsseg7ev_int64x7xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x7xm1_t **vlsseg7ev_int8x7xm1** (const signed char **address*, long *stride*, unsigned int *gvl*)
- uint16x7xm1_t **vlsseg7ev_uint16x7xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x7xm1_t **vlsseg7ev_uint32x7xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x7xm1_t **vlsseg7ev_uint64x7xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x7xm1_t **vlsseg7ev_uint8x7xm1** (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:

- float16x7xm1_t **vlsseg7ev_mask_float16x7xm1** (float16x7xm1_t *merge*, const float16_t **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- float32x7xm1_t **vlsseg7ev_mask_float32x7xm1** (float32x7xm1_t *merge*, const float **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- float64x7xm1_t **vlsseg7ev_mask_float64x7xm1** (float64x7xm1_t *merge*, const double **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int16x7xm1_t **vlsseg7ev_mask_int16x7xm1** (int16x7xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x7xm1_t **vlsseg7ev_mask_int32x7xm1** (int32x7xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x7xm1_t **vlsseg7ev_mask_int64x7xm1** (int64x7xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x7xm1_t **vlsseg7ev_mask_int8x7xm1** (int8x7xm1_t *merge*, const signed char **address*, long *stride*, *e8xm1_t mask*, unsigned int *gvl*)
- uint16x7xm1_t **vlsseg7ev_mask_uint16x7xm1** (uint16x7xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x7xm1_t **vlsseg7ev_mask_uint32x7xm1** (uint32x7xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x7xm1_t **vlsseg7ev_mask_uint64x7xm1** (uint64x7xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x7xm1_t **vlsseg7ev_mask_uint8x7xm1** (uint8x7xm1_t *merge*, const unsigned char **address*, long *stride*, *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) + 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:

- int16x7xm1_t **vlsseg7hv_int16x7xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x7xm1_t **vlsseg7hv_int32x7xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x7xm1_t **vlsseg7hv_int64x7xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x7xm1_t **vlsseg7hv_int8x7xm1** (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:

- int16x7xm1_t **vlsseg7hv_mask_int16x7xm1** (int16x7xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x7xm1_t **vlsseg7hv_mask_int32x7xm1** (int32x7xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x7xm1_t **vlsseg7hv_mask_int64x7xm1** (int64x7xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x7xm1_t **vlsseg7hv_mask_int8x7xm1** (int8x7xm1_t *merge*, const signed char **address*, long *stride*, *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) + 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:

- uint16x7xm1_t **vlsseg7huv_uint16x7xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x7xm1_t **vlsseg7huv_uint32x7xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x7xm1_t **vlsseg7huv_uint64x7xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x7xm1_t **vlsseg7huv_uint8x7xm1** (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:

- uint16x7xm1_t **vlsseg7huv_mask_uint16x7xm1** (uint16x7xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)

- uint32x7xm1_t **vlsseg7huv_mask_uint32x7xm1** (uint32x7xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x7xm1_t **vlsseg7huv_mask_uint64x7xm1** (uint64x7xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x7xm1_t **vlsseg7huv_mask_uint8x7xm1** (uint8x7xm1_t *merge*, const unsigned char **address*, long *stride*, *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) + 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:

- int16x7xm1_t **vlsseg7wv_int16x7xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x7xm1_t **vlsseg7wv_int32x7xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x7xm1_t **vlsseg7wv_int64x7xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x7xm1_t **vlsseg7wv_int8x7xm1** (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:

- int16x7xm1_t **vlsseg7wv_mask_int16x7xm1** (int16x7xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x7xm1_t **vlsseg7wv_mask_int32x7xm1** (int32x7xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x7xm1_t **vlsseg7wv_mask_int64x7xm1** (int64x7xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x7xm1_t **vlsseg7wv_mask_int8x7xm1** (int8x7xm1_t *merge*, const signed char **address*, long *stride*, *e8xm1_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: ['vlsseg7wu.v']

Prototypes:

- uint16x7xm1_t **vlsseg7wuv_uint16x7xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x7xm1_t **vlsseg7wuv_uint32x7xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x7xm1_t **vlsseg7wuv_uint64x7xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x7xm1_t **vlsseg7wuv_uint8x7xm1** (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:

- uint16x7xm1_t **vlsseg7wuv_mask_uint16x7xm1** (uint16x7xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x7xm1_t **vlsseg7wuv_mask_uint32x7xm1** (uint32x7xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x7xm1_t **vlsseg7wuv_mask_uint64x7xm1** (uint64x7xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x7xm1_t **vlsseg7wuv_mask_uint8x7xm1** (uint8x7xm1_t *merge*, const unsigned char **address*, long *stride*, *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) + 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:

- int16x8xm1_t **vlsseg8bv_int16x8xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x8xm1_t **vlsseg8bv_int32x8xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x8xm1_t **vlsseg8bv_int64x8xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x8xm1_t **vlsseg8bv_int8x8xm1** (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:

- int16x8xm1_t **vlsseg8bv_mask_int16x8xm1** (int16x8xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x8xm1_t **vlsseg8bv_mask_int32x8xm1** (int32x8xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x8xm1_t **vlsseg8bv_mask_int64x8xm1** (int64x8xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x8xm1_t **vlsseg8bv_mask_int8x8xm1** (int8x8xm1_t *merge*, const signed 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
```

2.10.93 Load 8 contiguous 8b unsigned fields in memory(strided) to consecutively numbered vector registers

Instruction: ['vlsseg8bu.v']

Prototypes:

- uint16x8xm1_t **vlsseg8buv_uint16x8xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x8xm1_t **vlsseg8buv_uint32x8xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x8xm1_t **vlsseg8buv_uint64x8xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)

- uint8x8xm1_t **vlsseg8buv_uint8x8xm1** (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:

- uint16x8xm1_t **vlsseg8buv_mask_uint16x8xm1** (uint16x8xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x8xm1_t **vlsseg8buv_mask_uint32x8xm1** (uint32x8xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x8xm1_t **vlsseg8buv_mask_uint64x8xm1** (uint64x8xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x8xm1_t **vlsseg8buv_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
```

2.10.94 Load 8 contiguous element fields in memory(strided) to consecutively numbered vector registers

Instruction: ['vlsseg8e.v']

Prototypes:

- float16x8xm1_t **vlsseg8ev_float16x8xm1** (const float16_t **address*, long *stride*, unsigned int *gvl*)
- float32x8xm1_t **vlsseg8ev_float32x8xm1** (const float **address*, long *stride*, unsigned int *gvl*)
- float64x8xm1_t **vlsseg8ev_float64x8xm1** (const double **address*, long *stride*, unsigned int *gvl*)
- int16x8xm1_t **vlsseg8ev_int16x8xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x8xm1_t **vlsseg8ev_int32x8xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x8xm1_t **vlsseg8ev_int64x8xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x8xm1_t **vlsseg8ev_int8x8xm1** (const signed char **address*, long *stride*, unsigned int *gvl*)
- uint16x8xm1_t **vlsseg8ev_uint16x8xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x8xm1_t **vlsseg8ev_uint32x8xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)

- uint64x8xm1_t **vlsseg8ev_uint64x8xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x8xm1_t **vlsseg8ev_uint8x8xm1** (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:

- float16x8xm1_t **vlsseg8ev_mask_float16x8xm1** (float16x8xm1_t *merge*, const float16_t **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- float32x8xm1_t **vlsseg8ev_mask_float32x8xm1** (float32x8xm1_t *merge*, const float **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- float64x8xm1_t **vlsseg8ev_mask_float64x8xm1** (float64x8xm1_t *merge*, const double **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int16x8xm1_t **vlsseg8ev_mask_int16x8xm1** (int16x8xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x8xm1_t **vlsseg8ev_mask_int32x8xm1** (int32x8xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x8xm1_t **vlsseg8ev_mask_int64x8xm1** (int64x8xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x8xm1_t **vlsseg8ev_mask_int8x8xm1** (int8x8xm1_t *merge*, const signed char **address*, long *stride*, *e8xm1_t mask*, unsigned int *gvl*)
- uint16x8xm1_t **vlsseg8ev_mask_uint16x8xm1** (uint16x8xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x8xm1_t **vlsseg8ev_mask_uint32x8xm1** (uint32x8xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x8xm1_t **vlsseg8ev_mask_uint64x8xm1** (uint64x8xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x8xm1_t **vlsseg8ev_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
```

2.10.95 Load 8 contiguous 16b signed fields in memory(strided) to consecutively numbered vector registers

Instruction: ['vlsseg8h.v']

Prototypes:

- int16x8xm1_t **vlsseg8hv_int16x8xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x8xm1_t **vlsseg8hv_int32x8xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x8xm1_t **vlsseg8hv_int64x8xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x8xm1_t **vlsseg8hv_int8x8xm1** (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:

- int16x8xm1_t **vlsseg8hv_mask_int16x8xm1** (int16x8xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x8xm1_t **vlsseg8hv_mask_int32x8xm1** (int32x8xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x8xm1_t **vlsseg8hv_mask_int64x8xm1** (int64x8xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x8xm1_t **vlsseg8hv_mask_int8x8xm1** (int8x8xm1_t *merge*, const signed 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
```

2.10.96 Load 8 contiguous 16b unsigned fields in memory(strided) to consecutively numbered vector registers

Instruction: ['vlsseg8hu.v']

Prototypes:

- uint16x8xm1_t **vlsseg8huv_uint16x8xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x8xm1_t **vlsseg8huv_uint32x8xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x8xm1_t **vlsseg8huv_uint64x8xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)

- uint8x8xm1_t **vlsseg8huv_uint8x8xm1** (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:

- uint16x8xm1_t **vlsseg8huv_mask_uint16x8xm1** (uint16x8xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x8xm1_t **vlsseg8huv_mask_uint32x8xm1** (uint32x8xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x8xm1_t **vlsseg8huv_mask_uint64x8xm1** (uint64x8xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x8xm1_t **vlsseg8huv_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
```

2.10.97 Load 8 contiguous 32b signed fields in memory(strided) to consecutively numbered vector registers

Instruction: ['vlsseg8w.v']

Prototypes:

- int16x8xm1_t **vlsseg8wv_int16x8xm1** (const short **address*, long *stride*, unsigned int *gvl*)
- int32x8xm1_t **vlsseg8wv_int32x8xm1** (const int **address*, long *stride*, unsigned int *gvl*)
- int64x8xm1_t **vlsseg8wv_int64x8xm1** (const long **address*, long *stride*, unsigned int *gvl*)
- int8x8xm1_t **vlsseg8wv_int8x8xm1** (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:

- int16x8xm1_t **vlsseg8wv_mask_int16x8xm1** (int16x8xm1_t *merge*, const short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x8xm1_t **vlsseg8wv_mask_int32x8xm1** (int32x8xm1_t *merge*, const int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x8xm1_t **vlsseg8wv_mask_int64x8xm1** (int64x8xm1_t *merge*, const long **address*, long *stride*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x8xm1_t **vlsseg8wv_mask_int8x8xm1** (int8x8xm1_t *merge*, const signed 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
```

2.10.98 Load 8 contiguous 32b unsigned fields in memory(strided) to consecutively numbered vector registers

Instruction: ['vlsseg8wu.v']

Prototypes:

- uint16x8xm1_t **vlsseg8wuv_uint16x8xm1** (const unsigned short **address*, long *stride*, unsigned int *gvl*)
- uint32x8xm1_t **vlsseg8wuv_uint32x8xm1** (const unsigned int **address*, long *stride*, unsigned int *gvl*)
- uint64x8xm1_t **vlsseg8wuv_uint64x8xm1** (const unsigned long **address*, long *stride*, unsigned int *gvl*)
- uint8x8xm1_t **vlsseg8wuv_uint8x8xm1** (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:

- uint16x8xm1_t **vlsseg8wuv_mask_uint16x8xm1** (uint16x8xm1_t *merge*, const unsigned short **address*, long *stride*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x8xm1_t **vlsseg8wuv_mask_uint32x8xm1** (uint32x8xm1_t *merge*, const unsigned int **address*, long *stride*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x8xm1_t **vlsseg8wuv_mask_uint64x8xm1** (uint64x8xm1_t *merge*, const unsigned long **address*, long *stride*, *e64xm1_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: ['vlxseg2b.v']

Prototypes:

- int16x2xm1_t **vlxseg2bv_int16x2xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int16x2xm2_t **vlxseg2bv_int16x2xm2_int16xm2** (const short **address*, *int16xm2_t index*, unsigned int *gvl*)
- int16x2xm4_t **vlxseg2bv_int16x2xm4_int16xm4** (const short **address*, *int16xm4_t index*, unsigned int *gvl*)
- int32x2xm1_t **vlxseg2bv_int32x2xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int32x2xm2_t **vlxseg2bv_int32x2xm2_int32xm2** (const int **address*, *int32xm2_t index*, unsigned int *gvl*)
- int32x2xm4_t **vlxseg2bv_int32x2xm4_int32xm4** (const int **address*, *int32xm4_t index*, unsigned int *gvl*)
- int64x2xm1_t **vlxseg2bv_int64x2xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int64x2xm2_t **vlxseg2bv_int64x2xm2_int64xm2** (const long **address*, *int64xm2_t index*, unsigned int *gvl*)
- int64x2xm4_t **vlxseg2bv_int64x2xm4_int64xm4** (const long **address*, *int64xm4_t index*, unsigned int *gvl*)
- int8x2xm1_t **vlxseg2bv_int8x2xm1_int8xm1** (const signed char **address*, *int8xm1_t index*, unsigned int *gvl*)
- int8x2xm2_t **vlxseg2bv_int8x2xm2_int8xm2** (const signed char **address*, *int8xm2_t index*, unsigned int *gvl*)
- int8x2xm4_t **vlxseg2bv_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: ['vlxseg2bu.v']

Prototypes:

- uint16x2xm1_t **vlxseg2buv_uint16x2xm1_uint16xm1**(const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint16x2xm2_t **vlxseg2buv_uint16x2xm2_uint16xm2**(const unsigned short **address*, *uint16xm2_t index*, unsigned int *gvl*)
- uint16x2xm4_t **vlxseg2buv_uint16x2xm4_uint16xm4**(const unsigned short **address*, *uint16xm4_t index*, unsigned int *gvl*)
- uint32x2xm1_t **vlxseg2buv_uint32x2xm1_uint32xm1**(const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint32x2xm2_t **vlxseg2buv_uint32x2xm2_uint32xm2**(const unsigned int **address*, *uint32xm2_t index*, unsigned int *gvl*)
- uint32x2xm4_t **vlxseg2buv_uint32x2xm4_uint32xm4**(const unsigned int **address*, *uint32xm4_t index*, unsigned int *gvl*)
- uint64x2xm1_t **vlxseg2buv_uint64x2xm1_uint64xm1**(const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint64x2xm2_t **vlxseg2buv_uint64x2xm2_uint64xm2**(const unsigned long **address*, *uint64xm2_t index*, unsigned int *gvl*)
- uint64x2xm4_t **vlxseg2buv_uint64x2xm4_uint64xm4**(const unsigned long **address*, *uint64xm4_t index*, unsigned int *gvl*)
- uint8x2xm1_t **vlxseg2buv_uint8x2xm1_uint8xm1**(const unsigned char **address*, *uint8xm1_t index*, unsigned int *gvl*)
- uint8x2xm2_t **vlxseg2buv_uint8x2xm2_uint8xm2**(const unsigned char **address*, *uint8xm2_t index*, unsigned int *gvl*)
- uint8x2xm4_t **vlxseg2buv_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 **vlxseg2buv_mask_uint16x2xm1_uint16xm1**(uint16x2xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint16x2xm2_t **vlxseg2buv_mask_uint16x2xm2_uint16xm2**(uint16x2xm2_t *merge*, const unsigned short **address*, *uint16xm2_t index*, *e16xm2_t mask*, unsigned int *gvl*)

- uint16x2xm4_t **vlxseg2buv_mask_uint16x2xm4_uint16xm4** (uint16x2xm4_t *merge*, const unsigned short **address*, *uint16xm4_t index*, *e16xm4_t mask*, unsigned int *gvl*)
- uint32x2xm1_t **vlxseg2buv_mask_uint32x2xm1_uint32xm1** (uint32x2xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint32x2xm2_t **vlxseg2buv_mask_uint32x2xm2_uint32xm2** (uint32x2xm2_t *merge*, const unsigned int **address*, *uint32xm2_t index*, *e32xm2_t mask*, unsigned int *gvl*)
- uint32x2xm4_t **vlxseg2buv_mask_uint32x2xm4_uint32xm4** (uint32x2xm4_t *merge*, const unsigned int **address*, *uint32xm4_t index*, *e32xm4_t mask*, unsigned int *gvl*)
- uint64x2xm1_t **vlxseg2buv_mask_uint64x2xm1_uint64xm1** (uint64x2xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint64x2xm2_t **vlxseg2buv_mask_uint64x2xm2_uint64xm2** (uint64x2xm2_t *merge*, const unsigned long **address*, *uint64xm2_t index*, *e64xm2_t mask*, unsigned int *gvl*)
- uint64x2xm4_t **vlxseg2buv_mask_uint64x2xm4_uint64xm4** (uint64x2xm4_t *merge*, const unsigned long **address*, *uint64xm4_t index*, *e64xm4_t mask*, unsigned int *gvl*)
- uint8x2xm1_t **vlxseg2buv_mask_uint8x2xm1_uint8xm1** (uint8x2xm1_t *merge*, const unsigned char **address*, *uint8xm1_t index*, *e8xm1_t mask*, unsigned int *gvl*)
- uint8x2xm2_t **vlxseg2buv_mask_uint8x2xm2_uint8xm2** (uint8x2xm2_t *merge*, const unsigned char **address*, *uint8xm2_t index*, *e8xm2_t mask*, unsigned int *gvl*)
- uint8x2xm4_t **vlxseg2buv_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]
```

(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: ['vlxseg2wu.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*, *e16xm1_t mask*, unsigned int *gvl*)

- uint16x2xm2_t **vlxseg2wuv_mask_uint16x2xm2_uint16xm2** (uint16x2xm2_t *merge*, const unsigned short **address*, *uint16xm2_t index*, *e16xm2_t mask*, unsigned int *gvl*)
- uint16x2xm4_t **vlxseg2wuv_mask_uint16x2xm4_uint16xm4** (uint16x2xm4_t *merge*, const unsigned short **address*, *uint16xm4_t index*, *e16xm4_t mask*, unsigned int *gvl*)
- uint32x2xm1_t **vlxseg2wuv_mask_uint32x2xm1_uint32xm1** (uint32x2xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint32x2xm2_t **vlxseg2wuv_mask_uint32x2xm2_uint32xm2** (uint32x2xm2_t *merge*, const unsigned int **address*, *uint32xm2_t index*, *e32xm2_t mask*, unsigned int *gvl*)
- uint32x2xm4_t **vlxseg2wuv_mask_uint32x2xm4_uint32xm4** (uint32x2xm4_t *merge*, const unsigned int **address*, *uint32xm4_t index*, *e32xm4_t mask*, unsigned int *gvl*)
- uint64x2xm1_t **vlxseg2wuv_mask_uint64x2xm1_uint64xm1** (uint64x2xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint64x2xm2_t **vlxseg2wuv_mask_uint64x2xm2_uint64xm2** (uint64x2xm2_t *merge*, const unsigned long **address*, *uint64xm2_t index*, *e64xm2_t mask*, unsigned int *gvl*)
- uint64x2xm4_t **vlxseg2wuv_mask_uint64x2xm4_uint64xm4** (uint64x2xm4_t *merge*, const unsigned long **address*, *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 **vlxseg3buv_uint16x3xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint16x3xm2_t **vlxseg3buv_uint16x3xm2_uint16xm2** (const unsigned short **address*, *uint16xm2_t index*, unsigned int *gvl*)
- uint32x3xm1_t **vlxseg3buv_uint32x3xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint32x3xm2_t **vlxseg3buv_uint32x3xm2_uint32xm2** (const unsigned int **address*, *uint32xm2_t index*, unsigned int *gvl*)
- uint64x3xm1_t **vlxseg3buv_uint64x3xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint64x3xm2_t **vlxseg3buv_uint64x3xm2_uint64xm2** (const unsigned long **address*, *uint64xm2_t index*, unsigned int *gvl*)
- uint8x3xm1_t **vlxseg3buv_uint8x3xm1_uint8xm1** (const unsigned char **address*, *uint8xm1_t index*, unsigned int *gvl*)
- uint8x3xm2_t **vlxseg3buv_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: ['vlxseg3e.v']

Prototypes:

- float16x3xm1_t **vlxseg3ev_float16x3xm1_float16xm1** (const float16_t **address*, *float16xm1_t index*, unsigned int *gvl*)
- float16x3xm2_t **vlxseg3ev_float16x3xm2_float16xm2** (const float16_t **address*, *float16xm2_t index*, unsigned int *gvl*)
- float32x3xm1_t **vlxseg3ev_float32x3xm1_float32xm1** (const float **address*, *float32xm1_t index*, unsigned int *gvl*)
- float32x3xm2_t **vlxseg3ev_float32x3xm2_float32xm2** (const float **address*, *float32xm2_t index*, unsigned int *gvl*)
- float64x3xm1_t **vlxseg3ev_float64x3xm1_float64xm1** (const double **address*, *float64xm1_t index*, unsigned int *gvl*)
- float64x3xm2_t **vlxseg3ev_float64x3xm2_float64xm2** (const double **address*, *float64xm2_t index*, unsigned int *gvl*)
- int16x3xm1_t **vlxseg3ev_int16x3xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int16x3xm2_t **vlxseg3ev_int16x3xm2_int16xm2** (const short **address*, *int16xm2_t index*, unsigned int *gvl*)
- int32x3xm1_t **vlxseg3ev_int32x3xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int32x3xm2_t **vlxseg3ev_int32x3xm2_int32xm2** (const int **address*, *int32xm2_t index*, unsigned int *gvl*)
- int64x3xm1_t **vlxseg3ev_int64x3xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int64x3xm2_t **vlxseg3ev_int64x3xm2_int64xm2** (const long **address*, *int64xm2_t index*, unsigned int *gvl*)
- int8x3xm1_t **vlxseg3ev_int8x3xm1_int8xm1** (const signed char **address*, *int8xm1_t index*, unsigned int *gvl*)
- int8x3xm2_t **vlxseg3ev_int8x3xm2_int8xm2** (const signed char **address*, *int8xm2_t index*, unsigned int *gvl*)
- uint16x3xm1_t **vlxseg3ev_uint16x3xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint16x3xm2_t **vlxseg3ev_uint16x3xm2_uint16xm2** (const unsigned short **address*, *uint16xm2_t index*, unsigned int *gvl*)
- uint32x3xm1_t **vlxseg3ev_uint32x3xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint32x3xm2_t **vlxseg3ev_uint32x3xm2_uint32xm2** (const unsigned int **address*, *uint32xm2_t index*, unsigned int *gvl*)
- uint64x3xm1_t **vlxseg3ev_uint64x3xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint64x3xm2_t **vlxseg3ev_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 **vlxseg4buv_uint16x4xm1_uint16xm1**(const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint16x4xm2_t **vlxseg4buv_uint16x4xm2_uint16xm2**(const unsigned short **address*, *uint16xm2_t index*, unsigned int *gvl*)
- uint32x4xm1_t **vlxseg4buv_uint32x4xm1_uint32xm1**(const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint32x4xm2_t **vlxseg4buv_uint32x4xm2_uint32xm2**(const unsigned int **address*, *uint32xm2_t index*, unsigned int *gvl*)
- uint64x4xm1_t **vlxseg4buv_uint64x4xm1_uint64xm1**(const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint64x4xm2_t **vlxseg4buv_uint64x4xm2_uint64xm2**(const unsigned long **address*, *uint64xm2_t index*, unsigned int *gvl*)
- uint8x4xm1_t **vlxseg4buv_uint8x4xm1_uint8xm1**(const unsigned char **address*, *uint8xm1_t index*, unsigned int *gvl*)
- uint8x4xm2_t **vlxseg4buv_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 **vlxseg4buv_mask_uint16x4xm1_uint16xm1**(uint16x4xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint16x4xm2_t **vlxseg4buv_mask_uint16x4xm2_uint16xm2**(uint16x4xm2_t *merge*, const unsigned short **address*, *uint16xm2_t index*, *e16xm2_t mask*, unsigned int *gvl*)

- uint32x4xm1_t **vlxseg4buv_mask_uint32x4xm1_uint32xm1** (uint32x4xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint32x4xm2_t **vlxseg4buv_mask_uint32x4xm2_uint32xm2** (uint32x4xm2_t *merge*, const unsigned int **address*, *uint32xm2_t index*, *e32xm2_t mask*, unsigned int *gvl*)
- uint64x4xm1_t **vlxseg4buv_mask_uint64x4xm1_uint64xm1** (uint64x4xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint64x4xm2_t **vlxseg4buv_mask_uint64x4xm2_uint64xm2** (uint64x4xm2_t *merge*, const unsigned long **address*, *uint64xm2_t index*, *e64xm2_t mask*, unsigned int *gvl*)
- uint8x4xm1_t **vlxseg4buv_mask_uint8x4xm1_uint8xm1** (uint8x4xm1_t *merge*, const unsigned char **address*, *uint8xm1_t index*, *e8xm1_t mask*, unsigned int *gvl*)
- uint8x4xm2_t **vlxseg4buv_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
```

2.10.115 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: ['vlxseg4wu.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*)

- uint64x5xm1_t **vlxseg5buv_mask_uint64x5xm1_uint64xm1** (uint64x5xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x5xm1_t **vlxseg5buv_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
```

2.10.122 Load 5 contiguous element fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg5e.v']

Prototypes:

- float16x5xm1_t **vlxseg5ev_float16x5xm1_float16xm1** (const float16_t **address*, *float16xm1_t index*, unsigned int *gvl*)
- float32x5xm1_t **vlxseg5ev_float32x5xm1_float32xm1** (const float **address*, *float32xm1_t index*, unsigned int *gvl*)
- float64x5xm1_t **vlxseg5ev_float64x5xm1_float64xm1** (const double **address*, *float64xm1_t index*, unsigned int *gvl*)
- int16x5xm1_t **vlxseg5ev_int16x5xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x5xm1_t **vlxseg5ev_int32x5xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x5xm1_t **vlxseg5ev_int64x5xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x5xm1_t **vlxseg5ev_int8x5xm1_int8xm1** (const signed char **address*, *int8xm1_t index*, unsigned int *gvl*)
- uint16x5xm1_t **vlxseg5ev_uint16x5xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x5xm1_t **vlxseg5ev_uint32x5xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x5xm1_t **vlxseg5ev_uint64x5xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x5xm1_t **vlxseg5ev_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:

- float16x5xm1_t **vlxseg5ev_mask_float16x5xm1_float16xm1** (float16x5xm1_t *merge*, const float16_t **address*, *float16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- float32x5xm1_t **vlxseg5ev_mask_float32x5xm1_float32xm1** (float32x5xm1_t *merge*, const float **address*, *float32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- float64x5xm1_t **vlxseg5ev_mask_float64x5xm1_float64xm1** (float64x5xm1_t *merge*, const double **address*, *float64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int16x5xm1_t **vlxseg5ev_mask_int16x5xm1_int16xm1** (int16x5xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x5xm1_t **vlxseg5ev_mask_int32x5xm1_int32xm1** (int32x5xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x5xm1_t **vlxseg5ev_mask_int64x5xm1_int64xm1** (int64x5xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x5xm1_t **vlxseg5ev_mask_int8x5xm1_int8xm1** (int8x5xm1_t *merge*, const signed char **address*, *int8xm1_t index*, *e8xm1_t mask*, unsigned int *gvl*)
- uint16x5xm1_t **vlxseg5ev_mask_uint16x5xm1_uint16xm1** (uint16x5xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x5xm1_t **vlxseg5ev_mask_uint32x5xm1_uint32xm1** (uint32x5xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x5xm1_t **vlxseg5ev_mask_uint64x5xm1_uint64xm1** (uint64x5xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x5xm1_t **vlxseg5ev_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
```

2.10.123 Load 5 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg5h.v']

Prototypes:

- int16x5xm1_t **vlxseg5hv_int16x5xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x5xm1_t **vlxseg5hv_int32x5xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x5xm1_t **vlxseg5hv_int64x5xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x5xm1_t **vlxseg5hv_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 **vlxseg5hv_mask_int16x5xm1_int16xm1** (int16x5xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x5xm1_t **vlxseg5hv_mask_int32x5xm1_int32xm1** (int32x5xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x5xm1_t **vlxseg5hv_mask_int64x5xm1_int64xm1** (int64x5xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x5xm1_t **vlxseg5hv_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
```

(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:

- uint16x5xm1_t **vlxseg5huv_uint16x5xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x5xm1_t **vlxseg5huv_uint32x5xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x5xm1_t **vlxseg5huv_uint64x5xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x5xm1_t **vlxseg5huv_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 **vlxseg5huv_mask_uint16x5xm1_uint16xm1** (uint16x5xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x5xm1_t **vlxseg5huv_mask_uint32x5xm1_uint32xm1** (uint32x5xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x5xm1_t **vlxseg5huv_mask_uint64x5xm1_uint64xm1** (uint64x5xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x5xm1_t **vlxseg5huv_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)
```

(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:

- int16x5xm1_t **vlxseg5wv_int16x5xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x5xm1_t **vlxseg5wv_int32x5xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x5xm1_t **vlxseg5wv_int64x5xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x5xm1_t **vlxseg5wv_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 **vlxseg5wv_mask_int16x5xm1_int16xm1** (int16x5xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x5xm1_t **vlxseg5wv_mask_int32x5xm1_int32xm1** (int32x5xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x5xm1_t **vlxseg5wv_mask_int64x5xm1_int64xm1** (int64x5xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x5xm1_t **vlxseg5wv_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
```

2.10.126 Load 5 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg5wu.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:

- int16x6xm1_t **vlxseg6bv_int16x6xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x6xm1_t **vlxseg6bv_int32x6xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x6xm1_t **vlxseg6bv_int64x6xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x6xm1_t **vlxseg6bv_int8x6xm1_int8xm1** (const signed char **address*, *int8xm1_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:

- int16x6xm1_t **vlxseg6bv_mask_int16x6xm1_int16xm1** (int16x6xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x6xm1_t **vlxseg6bv_mask_int32x6xm1_int32xm1** (int32x6xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x6xm1_t **vlxseg6bv_mask_int64x6xm1_int64xm1** (int64x6xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x6xm1_t **vlxseg6bv_mask_int8x6xm1_int8xm1** (int8x6xm1_t *merge*, const signed char **address*, *int8xm1_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
```

2.10.128 Load 6 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg6bu.v']

Prototypes:

- uint16x6xm1_t **vlxseg6buv_uint16x6xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x6xm1_t **vlxseg6buv_uint32x6xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x6xm1_t **vlxseg6buv_uint64x6xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x6xm1_t **vlxseg6buv_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 **vlxseg6buv_mask_uint16x6xm1_uint16xm1** (uint16x6xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x6xm1_t **vlxseg6buv_mask_uint32x6xm1_uint32xm1** (uint32x6xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x6xm1_t **vlxseg6buv_mask_uint64x6xm1_uint64xm1** (uint64x6xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x6xm1_t **vlxseg6buv_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*)

- int32x6xm1_t **vlxseg6ev_mask_int32x6xm1_int32xm1** (int32x6xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x6xm1_t **vlxseg6ev_mask_int64x6xm1_int64xm1** (int64x6xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x6xm1_t **vlxseg6ev_mask_int8x6xm1_int8xm1** (int8x6xm1_t *merge*, const signed char **address*, *int8xm1_t index*, *e8xm1_t mask*, unsigned int *gvl*)
- uint16x6xm1_t **vlxseg6ev_mask_uint16x6xm1_uint16xm1** (uint16x6xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x6xm1_t **vlxseg6ev_mask_uint32x6xm1_uint32xm1** (uint32x6xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x6xm1_t **vlxseg6ev_mask_uint64x6xm1_uint64xm1** (uint64x6xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x6xm1_t **vlxseg6ev_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
```

2.10.130 Load 6 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg6h.v']

Prototypes:

- int16x6xm1_t **vlxseg6hv_int16x6xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x6xm1_t **vlxseg6hv_int32x6xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x6xm1_t **vlxseg6hv_int64x6xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x6xm1_t **vlxseg6hv_int8x6xm1_int8xm1** (const signed char **address*, *int8xm1_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:

- int16x6xm1_t **vlxseg6hv_mask_int16x6xm1_int16xm1** (int16x6xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x6xm1_t **vlxseg6hv_mask_int32x6xm1_int32xm1** (int32x6xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x6xm1_t **vlxseg6hv_mask_int64x6xm1_int64xm1** (int64x6xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x6xm1_t **vlxseg6hv_mask_int8x6xm1_int8xm1** (int8x6xm1_t *merge*, const signed char **address*, *int8xm1_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
```

2.10.131 Load 6 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg6hu.v']

Prototypes:

- uint16x6xm1_t **vlxseg6huv_uint16x6xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x6xm1_t **vlxseg6huv_uint32x6xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x6xm1_t **vlxseg6huv_uint64x6xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x6xm1_t **vlxseg6huv_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 **vlxseg6huv_mask_uint16x6xm1_uint16xm1** (uint16x6xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x6xm1_t **vlxseg6huv_mask_uint32x6xm1_uint32xm1** (uint32x6xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x6xm1_t **vlxseg6huv_mask_uint64x6xm1_uint64xm1** (uint64x6xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x6xm1_t **vlxseg6huv_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
```

2.10.132 Load 6 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg6w.v']

Prototypes:

- int16x6xm1_t **vlxseg6wv_int16x6xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x6xm1_t **vlxseg6wv_int32x6xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x6xm1_t **vlxseg6wv_int64x6xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x6xm1_t **vlxseg6wv_int8x6xm1_int8xm1** (const signed char **address*, *int8xm1_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:

- int16x6xm1_t **vlxseg6wv_mask_int16x6xm1_int16xm1** (int16x6xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x6xm1_t **vlxseg6wv_mask_int32x6xm1_int32xm1** (int32x6xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x6xm1_t **vlxseg6wv_mask_int64x6xm1_int64xm1** (int64x6xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x6xm1_t **vlxseg6wv_mask_int8x6xm1_int8xm1** (int8x6xm1_t *merge*, const signed char **address*, *int8xm1_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
```

2.10.133 Load 6 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg6wu.v']

Prototypes:

- uint16x6xm1_t **vlxseg6wuv_uint16x6xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x6xm1_t **vlxseg6wuv_uint32x6xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x6xm1_t **vlxseg6wuv_uint64x6xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x6xm1_t **vlxseg6wuv_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 **vlxseg6wuv_mask_uint16x6xm1_uint16xm1** (uint16x6xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)

- uint32x6xm1_t **vlxseg6wuv_mask_uint32x6xm1_uint32xm1** (uint32x6xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x6xm1_t **vlxseg6wuv_mask_uint64x6xm1_uint64xm1** (uint64x6xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x6xm1_t **vlxseg6wuv_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
```

2.10.134 Load 7 contiguous 8b signed fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg7b.v']

Prototypes:

- int16x7xm1_t **vlxseg7bv_int16x7xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x7xm1_t **vlxseg7bv_int32x7xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x7xm1_t **vlxseg7bv_int64x7xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x7xm1_t **vlxseg7bv_int8x7xm1_int8xm1** (const signed char **address*, *int8xm1_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:

- int16x7xm1_t **vlxseg7bv_mask_int16x7xm1_int16xm1** (int16x7xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x7xm1_t **vlxseg7bv_mask_int32x7xm1_int32xm1** (int32x7xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)

- int64x7xm1_t **vlxseg7bv_mask_int64x7xm1_int64xm1** (int64x7xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x7xm1_t **vlxseg7bv_mask_int8x7xm1_int8xm1** (int8x7xm1_t *merge*, const signed char **address*, *int8xm1_t index*, *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) + 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:

- uint16x7xm1_t **vlxseg7buv_uint16x7xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x7xm1_t **vlxseg7buv_uint32x7xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x7xm1_t **vlxseg7buv_uint64x7xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x7xm1_t **vlxseg7buv_uint8x7xm1_uint8xm1** (const unsigned char **address*, *uint8xm1_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:

- uint16x7xm1_t **vlxseg7buv_mask_uint16x7xm1_uint16xm1** (uint16x7xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x7xm1_t **vlxseg7buv_mask_uint32x7xm1_uint32xm1** (uint32x7xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)

- uint64x7xm1_t **vlxseg7buv_mask_uint64x7xm1_uint64xm1** (uint64x7xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x7xm1_t **vlxseg7buv_mask_uint8x7xm1_uint8xm1** (uint8x7xm1_t *merge*, const unsigned char **address*, *uint8xm1_t index*, *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) + 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:

- float16x7xm1_t **vlxseg7ev_float16x7xm1_float16xm1** (const float16_t **address*, *float16xm1_t index*, unsigned int *gvl*)
- float32x7xm1_t **vlxseg7ev_float32x7xm1_float32xm1** (const float **address*, *float32xm1_t index*, unsigned int *gvl*)
- float64x7xm1_t **vlxseg7ev_float64x7xm1_float64xm1** (const double **address*, *float64xm1_t index*, unsigned int *gvl*)
- int16x7xm1_t **vlxseg7ev_int16x7xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x7xm1_t **vlxseg7ev_int32x7xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x7xm1_t **vlxseg7ev_int64x7xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x7xm1_t **vlxseg7ev_int8x7xm1_int8xm1** (const signed char **address*, *int8xm1_t index*, unsigned int *gvl*)
- uint16x7xm1_t **vlxseg7ev_uint16x7xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x7xm1_t **vlxseg7ev_uint32x7xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x7xm1_t **vlxseg7ev_uint64x7xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x7xm1_t **vlxseg7ev_uint8x7xm1_uint8xm1** (const unsigned char **address*, *uint8xm1_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:

- float16x7xm1_t **vlxseg7ev_mask_float16x7xm1_float16xm1** (float16x7xm1_t *merge*, const float16_t **address*, *float16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- float32x7xm1_t **vlxseg7ev_mask_float32x7xm1_float32xm1** (float32x7xm1_t *merge*, const float **address*, *float32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- float64x7xm1_t **vlxseg7ev_mask_float64x7xm1_float64xm1** (float64x7xm1_t *merge*, const double **address*, *float64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int16x7xm1_t **vlxseg7ev_mask_int16x7xm1_int16xm1** (int16x7xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x7xm1_t **vlxseg7ev_mask_int32x7xm1_int32xm1** (int32x7xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x7xm1_t **vlxseg7ev_mask_int64x7xm1_int64xm1** (int64x7xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x7xm1_t **vlxseg7ev_mask_int8x7xm1_int8xm1** (int8x7xm1_t *merge*, const signed char **address*, *int8xm1_t index*, *e8xm1_t mask*, unsigned int *gvl*)
- uint16x7xm1_t **vlxseg7ev_mask_uint16x7xm1_uint16xm1** (uint16x7xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x7xm1_t **vlxseg7ev_mask_uint32x7xm1_uint32xm1** (uint32x7xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x7xm1_t **vlxseg7ev_mask_uint64x7xm1_uint64xm1** (uint64x7xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x7xm1_t **vlxseg7ev_mask_uint8x7xm1_uint8xm1** (uint8x7xm1_t *merge*, const unsigned char **address*, *uint8xm1_t index*, *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) + 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:

- int16x7xm1_t **vlxseg7hv_int16x7xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x7xm1_t **vlxseg7hv_int32x7xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x7xm1_t **vlxseg7hv_int64x7xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x7xm1_t **vlxseg7hv_int8x7xm1_int8xm1** (const signed char **address*, *int8xm1_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:

- int16x7xm1_t **vlxseg7hv_mask_int16x7xm1_int16xm1** (int16x7xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x7xm1_t **vlxseg7hv_mask_int32x7xm1_int32xm1** (int32x7xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x7xm1_t **vlxseg7hv_mask_int64x7xm1_int64xm1** (int64x7xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x7xm1_t **vlxseg7hv_mask_int8x7xm1_int8xm1** (int8x7xm1_t *merge*, const signed char **address*, *int8xm1_t index*, *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) + 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:

- uint16x7xm1_t **vlxseg7huv_uint16x7xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x7xm1_t **vlxseg7huv_uint32x7xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x7xm1_t **vlxseg7huv_uint64x7xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x7xm1_t **vlxseg7huv_uint8x7xm1_uint8xm1** (const unsigned char **address*, *uint8xm1_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:

- uint16x7xm1_t **vlxseg7huv_mask_uint16x7xm1_uint16xm1** (uint16x7xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x7xm1_t **vlxseg7huv_mask_uint32x7xm1_uint32xm1** (uint32x7xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x7xm1_t **vlxseg7huv_mask_uint64x7xm1_uint64xm1** (uint64x7xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x7xm1_t **vlxseg7huv_mask_uint8x7xm1_uint8xm1** (uint8x7xm1_t *merge*, const unsigned char **address*, *uint8xm1_t index*, *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)
```

(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:

- int16x7xm1_t **vlxseg7wv_int16x7xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x7xm1_t **vlxseg7wv_int32x7xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x7xm1_t **vlxseg7wv_int64x7xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x7xm1_t **vlxseg7wv_int8x7xm1_int8xm1** (const signed char **address*, *int8xm1_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:

- int16x7xm1_t **vlxseg7wv_mask_int16x7xm1_int16xm1** (int16x7xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x7xm1_t **vlxseg7wv_mask_int32x7xm1_int32xm1** (int32x7xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x7xm1_t **vlxseg7wv_mask_int64x7xm1_int64xm1** (int64x7xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x7xm1_t **vlxseg7wv_mask_int8x7xm1_int8xm1** (int8x7xm1_t *merge*, const signed char **address*, *int8xm1_t index*, *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) + 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: ['vlxseg7wu.v']

Prototypes:

- uint16x7xm1_t **vlxseg7wuv_uint16x7xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x7xm1_t **vlxseg7wuv_uint32x7xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x7xm1_t **vlxseg7wuv_uint64x7xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x7xm1_t **vlxseg7wuv_uint8x7xm1_uint8xm1** (const unsigned char **address*, *uint8xm1_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:

- uint16x7xm1_t **vlxseg7wuv_mask_uint16x7xm1_uint16xm1** (uint16x7xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x7xm1_t **vlxseg7wuv_mask_uint32x7xm1_uint32xm1** (uint32x7xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x7xm1_t **vlxseg7wuv_mask_uint64x7xm1_uint64xm1** (uint64x7xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x7xm1_t **vlxseg7wuv_mask_uint8x7xm1_uint8xm1** (uint8x7xm1_t *merge*, const unsigned char **address*, *uint8xm1_t index*, *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) + 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: ['vlxseg8b.v']

Prototypes:

- int16x8xm1_t **vlxseg8bv_int16x8xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x8xm1_t **vlxseg8bv_int32x8xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x8xm1_t **vlxseg8bv_int64x8xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x8xm1_t **vlxseg8bv_int8x8xm1_int8xm1** (const signed char **address*, *int8xm1_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:

- int16x8xm1_t **vlxseg8bv_mask_int16x8xm1_int16xm1** (int16x8xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x8xm1_t **vlxseg8bv_mask_int32x8xm1_int32xm1** (int32x8xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x8xm1_t **vlxseg8bv_mask_int64x8xm1_int64xm1** (int64x8xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x8xm1_t **vlxseg8bv_mask_int8x8xm1_int8xm1** (int8x8xm1_t *merge*, const signed char **address*, *int8xm1_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
```

2.10.142 Load 8 contiguous 8b unsigned fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg8bu.v']

Prototypes:

- uint16x8xm1_t **vlxseg8buv_uint16x8xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x8xm1_t **vlxseg8buv_uint32x8xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x8xm1_t **vlxseg8buv_uint64x8xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x8xm1_t **vlxseg8buv_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 **vlxseg8buv_mask_uint16x8xm1_uint16xm1** (uint16x8xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x8xm1_t **vlxseg8buv_mask_uint32x8xm1_uint32xm1** (uint32x8xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x8xm1_t **vlxseg8buv_mask_uint64x8xm1_uint64xm1** (uint64x8xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x8xm1_t **vlxseg8buv_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:

- float16x8xm1_t **vlxseg8ev_float16x8xm1_float16xm1** (const float16_t **address*, *float16xm1_t index*, unsigned int *gvl*)
- float32x8xm1_t **vlxseg8ev_float32x8xm1_float32xm1** (const float **address*, *float32xm1_t index*, unsigned int *gvl*)
- float64x8xm1_t **vlxseg8ev_float64x8xm1_float64xm1** (const double **address*, *float64xm1_t index*, unsigned int *gvl*)
- int16x8xm1_t **vlxseg8ev_int16x8xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x8xm1_t **vlxseg8ev_int32x8xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x8xm1_t **vlxseg8ev_int64x8xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x8xm1_t **vlxseg8ev_int8x8xm1_int8xm1** (const signed char **address*, *int8xm1_t index*, unsigned int *gvl*)
- uint16x8xm1_t **vlxseg8ev_uint16x8xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x8xm1_t **vlxseg8ev_uint32x8xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x8xm1_t **vlxseg8ev_uint64x8xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x8xm1_t **vlxseg8ev_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:

- float16x8xm1_t **vlxseg8ev_mask_float16x8xm1_float16xm1** (float16x8xm1_t *merge*, const float16_t **address*, *float16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- float32x8xm1_t **vlxseg8ev_mask_float32x8xm1_float32xm1** (float32x8xm1_t *merge*, const float **address*, *float32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- float64x8xm1_t **vlxseg8ev_mask_float64x8xm1_float64xm1** (float64x8xm1_t *merge*, const double **address*, *float64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int16x8xm1_t **vlxseg8ev_mask_int16x8xm1_int16xm1** (int16x8xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)

- int32x8xm1_t **vlxseg8ev_mask_int32x8xm1_int32xm1** (int32x8xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x8xm1_t **vlxseg8ev_mask_int64x8xm1_int64xm1** (int64x8xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x8xm1_t **vlxseg8ev_mask_int8x8xm1_int8xm1** (int8x8xm1_t *merge*, const signed char **address*, *int8xm1_t index*, *e8xm1_t mask*, unsigned int *gvl*)
- uint16x8xm1_t **vlxseg8ev_mask_uint16x8xm1_uint16xm1** (uint16x8xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x8xm1_t **vlxseg8ev_mask_uint32x8xm1_uint32xm1** (uint32x8xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x8xm1_t **vlxseg8ev_mask_uint64x8xm1_uint64xm1** (uint64x8xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x8xm1_t **vlxseg8ev_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
```

2.10.144 Load 8 contiguous 16b signed fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg8h.v']

Prototypes:

- int16x8xm1_t **vlxseg8hv_int16x8xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x8xm1_t **vlxseg8hv_int32x8xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x8xm1_t **vlxseg8hv_int64x8xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x8xm1_t **vlxseg8hv_int8x8xm1_int8xm1** (const signed char **address*, *int8xm1_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:

- int16x8xm1_t **vlxseg8hv_mask_int16x8xm1_int16xm1** (int16x8xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x8xm1_t **vlxseg8hv_mask_int32x8xm1_int32xm1** (int32x8xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x8xm1_t **vlxseg8hv_mask_int64x8xm1_int64xm1** (int64x8xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x8xm1_t **vlxseg8hv_mask_int8x8xm1_int8xm1** (int8x8xm1_t *merge*, const signed char **address*, *int8xm1_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
```

2.10.145 Load 8 contiguous 16b unsigned fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg8hu.v']

Prototypes:

- uint16x8xm1_t **vlxseg8huv_uint16x8xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x8xm1_t **vlxseg8huv_uint32x8xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x8xm1_t **vlxseg8huv_uint64x8xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x8xm1_t **vlxseg8huv_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 **vlxseg8huv_mask_uint16x8xm1_uint16xm1** (uint16x8xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- uint32x8xm1_t **vlxseg8huv_mask_uint32x8xm1_uint32xm1** (uint32x8xm1_t *merge*, const unsigned int **address*, *uint32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- uint64x8xm1_t **vlxseg8huv_mask_uint64x8xm1_uint64xm1** (uint64x8xm1_t *merge*, const unsigned long **address*, *uint64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- uint8x8xm1_t **vlxseg8huv_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
```

2.10.146 Load 8 contiguous 32b signed fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg8w.v']

Prototypes:

- int16x8xm1_t **vlxseg8wv_int16x8xm1_int16xm1** (const short **address*, *int16xm1_t index*, unsigned int *gvl*)
- int32x8xm1_t **vlxseg8wv_int32x8xm1_int32xm1** (const int **address*, *int32xm1_t index*, unsigned int *gvl*)
- int64x8xm1_t **vlxseg8wv_int64x8xm1_int64xm1** (const long **address*, *int64xm1_t index*, unsigned int *gvl*)
- int8x8xm1_t **vlxseg8wv_int8x8xm1_int8xm1** (const signed char **address*, *int8xm1_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:

- int16x8xm1_t **vlxseg8wv_mask_int16x8xm1_int16xm1** (int16x8xm1_t *merge*, const short **address*, *int16xm1_t index*, *e16xm1_t mask*, unsigned int *gvl*)
- int32x8xm1_t **vlxseg8wv_mask_int32x8xm1_int32xm1** (int32x8xm1_t *merge*, const int **address*, *int32xm1_t index*, *e32xm1_t mask*, unsigned int *gvl*)
- int64x8xm1_t **vlxseg8wv_mask_int64x8xm1_int64xm1** (int64x8xm1_t *merge*, const long **address*, *int64xm1_t index*, *e64xm1_t mask*, unsigned int *gvl*)
- int8x8xm1_t **vlxseg8wv_mask_int8x8xm1_int8xm1** (int8x8xm1_t *merge*, const signed char **address*, *int8xm1_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
```

2.10.147 Load 8 contiguous 32b unsigned fields in memory(indexed) to consecutively numbered vector registers

Instruction: ['vlxseg8wu.v']

Prototypes:

- uint16x8xm1_t **vlxseg8wuv_uint16x8xm1_uint16xm1** (const unsigned short **address*, *uint16xm1_t index*, unsigned int *gvl*)
- uint32x8xm1_t **vlxseg8wuv_uint32x8xm1_uint32xm1** (const unsigned int **address*, *uint32xm1_t index*, unsigned int *gvl*)
- uint64x8xm1_t **vlxseg8wuv_uint64x8xm1_uint64xm1** (const unsigned long **address*, *uint64xm1_t index*, unsigned int *gvl*)
- uint8x8xm1_t **vlxseg8wuv_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 **vlxseg8wuv_mask_uint16x8xm1_uint16xm1** (uint16x8xm1_t *merge*, const unsigned short **address*, *uint16xm1_t index*, *e16xm1_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: ['vsseg2b.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: ['vsseg2h.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: ['vsseg3b.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: ['vsseg7b.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: ['vsseg8b.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: ['vssseg2b.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: ['vsxseg2w.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: ['vmand.mm']

Prototypes:

- *e16xm1_t* **vmandmm_e16xm1** (*e16xm1_t a*, *e16xm1_t b*, unsigned int *gvl*)
- *e16xm2_t* **vmandmm_e16xm2** (*e16xm2_t a*, *e16xm2_t b*, unsigned int *gvl*)
- *e16xm4_t* **vmandmm_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* **vmandnotmm_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* **vmclrm_e16xm1** (unsigned int *gvl*)
- *e16xm2_t* **vmclrm_e16xm2** (unsigned int *gvl*)
- *e16xm4_t* **vmclrm_e16xm4** (unsigned int *gvl*)
- *e16xm8_t* **vmclrm_e16xm8** (unsigned int *gvl*)
- *e32xm1_t* **vmclrm_e32xm1** (unsigned int *gvl*)
- *e32xm2_t* **vmclrm_e32xm2** (unsigned int *gvl*)
- *e32xm4_t* **vmclrm_e32xm4** (unsigned int *gvl*)
- *e32xm8_t* **vmclrm_e32xm8** (unsigned int *gvl*)
- *e64xm1_t* **vmclrm_e64xm1** (unsigned int *gvl*)
- *e64xm2_t* **vmclrm_e64xm2** (unsigned int *gvl*)
- *e64xm4_t* **vmclrm_e64xm4** (unsigned int *gvl*)
- *e64xm8_t* **vmclrm_e64xm8** (unsigned int *gvl*)
- *e8xm1_t* **vmclrm_e8xm1** (unsigned int *gvl*)
- *e8xm2_t* **vmclrm_e8xm2** (unsigned int *gvl*)
- *e8xm4_t* **vmclrm_e8xm4** (unsigned int *gvl*)
- *e8xm8_t* **vmclrm_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* **vmcpym_e16xm1** (*e16xm1_t a*, unsigned int *gvl*)
- *e16xm2_t* **vmcpym_e16xm2** (*e16xm2_t a*, unsigned int *gvl*)
- *e16xm4_t* **vmcpym_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* **vmormm_e16xm1** (*e16xm1_t a*, *e16xm1_t b*, unsigned int *gvl*)
- *e16xm2_t* **vmormm_e16xm2** (*e16xm2_t a*, *e16xm2_t b*, unsigned int *gvl*)
- *e16xm4_t* **vmormm_e16xm4** (*e16xm4_t a*, *e16xm4_t b*, unsigned int *gvl*)
- *e16xm8_t* **vmormm_e16xm8** (*e16xm8_t a*, *e16xm8_t b*, unsigned int *gvl*)

- *e32xm1_t* **vmormm_e32xm1** (*e32xm1_t a*, *e32xm1_t b*, unsigned int *gvl*)
- *e32xm2_t* **vmormm_e32xm2** (*e32xm2_t a*, *e32xm2_t b*, unsigned int *gvl*)
- *e32xm4_t* **vmormm_e32xm4** (*e32xm4_t a*, *e32xm4_t b*, unsigned int *gvl*)
- *e32xm8_t* **vmormm_e32xm8** (*e32xm8_t a*, *e32xm8_t b*, unsigned int *gvl*)
- *e64xm1_t* **vmormm_e64xm1** (*e64xm1_t a*, *e64xm1_t b*, unsigned int *gvl*)
- *e64xm2_t* **vmormm_e64xm2** (*e64xm2_t a*, *e64xm2_t b*, unsigned int *gvl*)
- *e64xm4_t* **vmormm_e64xm4** (*e64xm4_t a*, *e64xm4_t b*, unsigned int *gvl*)
- *e64xm8_t* **vmormm_e64xm8** (*e64xm8_t a*, *e64xm8_t b*, unsigned int *gvl*)
- *e8xm1_t* **vmormm_e8xm1** (*e8xm1_t a*, *e8xm1_t b*, unsigned int *gvl*)
- *e8xm2_t* **vmormm_e8xm2** (*e8xm2_t a*, *e8xm2_t b*, unsigned int *gvl*)
- *e8xm4_t* **vmormm_e8xm4** (*e8xm4_t a*, *e8xm4_t b*, unsigned int *gvl*)
- *e8xm8_t* **vmormm_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 beforte the first one enabled

Instruction: ['vmsbf.m']

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: ['vmsif.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* **vmxnormm_e16xm1** (*e16xm1_t a*, *e16xm1_t b*, unsigned int *gvl*)
- *e16xm2_t* **vmxnormm_e16xm2** (*e16xm2_t a*, *e16xm2_t b*, unsigned int *gvl*)
- *e16xm4_t* **vmxnormm_e16xm4** (*e16xm4_t a*, *e16xm4_t b*, unsigned int *gvl*)
- *e16xm8_t* **vmxnormm_e16xm8** (*e16xm8_t a*, *e16xm8_t b*, unsigned int *gvl*)
- *e32xm1_t* **vmxnormm_e32xm1** (*e32xm1_t a*, *e32xm1_t b*, unsigned int *gvl*)
- *e32xm2_t* **vmxnormm_e32xm2** (*e32xm2_t a*, *e32xm2_t b*, unsigned int *gvl*)
- *e32xm4_t* **vmxnormm_e32xm4** (*e32xm4_t a*, *e32xm4_t b*, unsigned int *gvl*)
- *e32xm8_t* **vmxnormm_e32xm8** (*e32xm8_t a*, *e32xm8_t b*, unsigned int *gvl*)
- *e64xm1_t* **vmxnormm_e64xm1** (*e64xm1_t a*, *e64xm1_t b*, unsigned int *gvl*)
- *e64xm2_t* **vmxnormm_e64xm2** (*e64xm2_t a*, *e64xm2_t b*, unsigned int *gvl*)
- *e64xm4_t* **vmxnormm_e64xm4** (*e64xm4_t a*, *e64xm4_t b*, unsigned int *gvl*)
- *e64xm8_t* **vmxnormm_e64xm8** (*e64xm8_t a*, *e64xm8_t b*, unsigned int *gvl*)
- *e8xm1_t* **vmxnormm_e8xm1** (*e8xm1_t a*, *e8xm1_t b*, unsigned int *gvl*)
- *e8xm2_t* **vmxnormm_e8xm2** (*e8xm2_t a*, *e8xm2_t b*, unsigned int *gvl*)
- *e8xm4_t* **vmxnormm_e8xm4** (*e8xm4_t a*, *e8xm4_t b*, unsigned int *gvl*)
- *e8xm8_t* **vmxnormm_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* **vmxormm_e16xm1** (*e16xm1_t a*, *e16xm1_t b*, unsigned int *gvl*)
- *e16xm2_t* **vmxormm_e16xm2** (*e16xm2_t a*, *e16xm2_t b*, unsigned int *gvl*)

- *e16xm4_t* **vmxormm_e16xm4** (*e16xm4_t a*, *e16xm4_t b*, unsigned int *gvl*)
- *e16xm8_t* **vmxormm_e16xm8** (*e16xm8_t a*, *e16xm8_t b*, unsigned int *gvl*)
- *e32xm1_t* **vmxormm_e32xm1** (*e32xm1_t a*, *e32xm1_t b*, unsigned int *gvl*)
- *e32xm2_t* **vmxormm_e32xm2** (*e32xm2_t a*, *e32xm2_t b*, unsigned int *gvl*)
- *e32xm4_t* **vmxormm_e32xm4** (*e32xm4_t a*, *e32xm4_t b*, unsigned int *gvl*)
- *e32xm8_t* **vmxormm_e32xm8** (*e32xm8_t a*, *e32xm8_t b*, unsigned int *gvl*)
- *e64xm1_t* **vmxormm_e64xm1** (*e64xm1_t a*, *e64xm1_t b*, unsigned int *gvl*)
- *e64xm2_t* **vmxormm_e64xm2** (*e64xm2_t a*, *e64xm2_t b*, unsigned int *gvl*)
- *e64xm4_t* **vmxormm_e64xm4** (*e64xm4_t a*, *e64xm4_t b*, unsigned int *gvl*)
- *e64xm8_t* **vmxormm_e64xm8** (*e64xm8_t a*, *e64xm8_t b*, unsigned int *gvl*)
- *e8xm1_t* **vmxormm_e8xm1** (*e8xm1_t a*, *e8xm1_t b*, unsigned int *gvl*)
- *e8xm2_t* **vmxormm_e8xm2** (*e8xm2_t a*, *e8xm2_t b*, unsigned int *gvl*)
- *e8xm4_t* **vmxormm_e8xm4** (*e8xm4_t a*, *e8xm4_t b*, unsigned int *gvl*)
- *e8xm8_t* **vmxormm_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.vm']

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 **vextxv_int32xm4** (*int32xm4_t* *a*, unsigned int *b*, unsigned int *gvl*)
- int **vextxv_int32xm8** (*int32xm8_t* *a*, unsigned int *b*, unsigned int *gvl*)
- long **vextxv_int64xm1** (*int64xm1_t* *a*, unsigned int *b*, unsigned int *gvl*)
- long **vextxv_int64xm2** (*int64xm2_t* *a*, unsigned int *b*, unsigned int *gvl*)
- long **vextxv_int64xm4** (*int64xm4_t* *a*, unsigned int *b*, unsigned int *gvl*)
- long **vextxv_int64xm8** (*int64xm8_t* *a*, unsigned int *b*, unsigned int *gvl*)
- signed char **vextxv_int8xm1** (*int8xm1_t* *a*, unsigned int *b*, unsigned int *gvl*)
- signed char **vextxv_int8xm2** (*int8xm2_t* *a*, unsigned int *b*, unsigned int *gvl*)
- signed char **vextxv_int8xm4** (*int8xm4_t* *a*, unsigned int *b*, unsigned int *gvl*)
- signed char **vextxv_int8xm8** (*int8xm8_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned short **vextxv_uint16xm1** (*uint16xm1_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned short **vextxv_uint16xm2** (*uint16xm2_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned short **vextxv_uint16xm4** (*uint16xm4_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned short **vextxv_uint16xm8** (*uint16xm8_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned int **vextxv_uint32xm1** (*uint32xm1_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned int **vextxv_uint32xm2** (*uint32xm2_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned int **vextxv_uint32xm4** (*uint32xm4_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned int **vextxv_uint32xm8** (*uint32xm8_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned long **vextxv_uint64xm1** (*uint64xm1_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned long **vextxv_uint64xm2** (*uint64xm2_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned long **vextxv_uint64xm4** (*uint64xm4_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned long **vextxv_uint64xm8** (*uint64xm8_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned char **vextxv_uint8xm1** (*uint8xm1_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned char **vextxv_uint8xm2** (*uint8xm2_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned char **vextxv_uint8xm4** (*uint8xm4_t* *a*, unsigned int *b*, unsigned int *gvl*)
- unsigned char **vextxv_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* **vfmvsf_float16xm1** (float16_t *a*, unsigned int *gvl*)
- *float16xm2_t* **vfmvsf_float16xm2** (float16_t *a*, unsigned int *gvl*)
- *float16xm4_t* **vfmvsf_float16xm4** (float16_t *a*, unsigned int *gvl*)
- *float16xm8_t* **vfmvsf_float16xm8** (float16_t *a*, unsigned int *gvl*)
- *float32xm1_t* **vfmvsf_float32xm1** (float *a*, unsigned int *gvl*)
- *float32xm2_t* **vfmvsf_float32xm2** (float *a*, unsigned int *gvl*)
- *float32xm4_t* **vfmvsf_float32xm4** (float *a*, unsigned int *gvl*)
- *float32xm8_t* **vfmvsf_float32xm8** (float *a*, unsigned int *gvl*)
- *float64xm1_t* **vfmvsf_float64xm1** (double *a*, unsigned int *gvl*)
- *float64xm2_t* **vfmvsf_float64xm2** (double *a*, unsigned int *gvl*)
- *float64xm4_t* **vfmvsf_float64xm4** (double *a*, unsigned int *gvl*)
- *float64xm8_t* **vfmvsf_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* **vfmvvf_float64xm4** (double *a*, unsigned int *gvl*)
- *float64xm8_t* **vfmvvf_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: ['vmerge.vvm']

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* **vmergevvm_mask_int16xm4** (*int16xm4_t a*, *int16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vmergevvm_mask_int16xm8** (*int16xm8_t a*, *int16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)
- *int32xm1_t* **vmergevvm_mask_int32xm1** (*int32xm1_t a*, *int32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vmergevvm_mask_int32xm2** (*int32xm2_t a*, *int32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vmergevvm_mask_int32xm4** (*int32xm4_t a*, *int32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vmergevvm_mask_int32xm8** (*int32xm8_t a*, *int32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *int64xm1_t* **vmergevvm_mask_int64xm1** (*int64xm1_t a*, *int64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vmergevvm_mask_int64xm2** (*int64xm2_t a*, *int64xm2_t b*, *e64xm2_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vmergevvm_mask_int64xm4** (*int64xm4_t a*, *int64xm4_t b*, *e64xm4_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vmergevvm_mask_int64xm8** (*int64xm8_t a*, *int64xm8_t b*, *e64xm8_t mask*, unsigned int *gvl*)
- *int8xm1_t* **vmergevvm_mask_int8xm1** (*int8xm1_t a*, *int8xm1_t b*, *e8xm1_t mask*, unsigned int *gvl*)
- *int8xm2_t* **vmergevvm_mask_int8xm2** (*int8xm2_t a*, *int8xm2_t b*, *e8xm2_t mask*, unsigned int *gvl*)
- *int8xm4_t* **vmergevvm_mask_int8xm4** (*int8xm4_t a*, *int8xm4_t b*, *e8xm4_t mask*, unsigned int *gvl*)
- *int8xm8_t* **vmergevvm_mask_int8xm8** (*int8xm8_t a*, *int8xm8_t b*, *e8xm8_t mask*, unsigned int *gvl*)
- *uint16xm1_t* **vmergevvm_mask_uint16xm1** (*uint16xm1_t a*, *uint16xm1_t b*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint16xm2_t* **vmergevvm_mask_uint16xm2** (*uint16xm2_t a*, *uint16xm2_t b*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vmergevvm_mask_uint16xm4** (*uint16xm4_t a*, *uint16xm4_t b*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vmergevvm_mask_uint16xm8** (*uint16xm8_t a*, *uint16xm8_t b*, *e16xm8_t mask*, unsigned int *gvl*)
- *uint32xm1_t* **vmergevvm_mask_uint32xm1** (*uint32xm1_t a*, *uint32xm1_t b*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vmergevvm_mask_uint32xm2** (*uint32xm2_t a*, *uint32xm2_t b*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vmergevvm_mask_uint32xm4** (*uint32xm4_t a*, *uint32xm4_t b*, *e32xm4_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vmergevvm_mask_uint32xm8** (*uint32xm8_t a*, *uint32xm8_t b*, *e32xm8_t mask*, unsigned int *gvl*)
- *uint64xm1_t* **vmergevvm_mask_uint64xm1** (*uint64xm1_t a*, *uint64xm1_t b*, *e64xm1_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vmergevvm_mask_uint64xm2** (*uint64xm2_t a*, *uint64xm2_t b*, *e64xm2_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vmergevvm_mask_uint64xm4** (*uint64xm4_t a*, *uint64xm4_t b*, *e64xm4_t mask*, unsigned int *gvl*)

- *uint64xm8_t* **vmergevvm_mask_uint64xm8** (*uint64xm8_t a*, *uint64xm8_t b*, *e64xm8_t mask*, unsigned int *gvl*)
- *uint8xm1_t* **vmergevvm_mask_uint8xm1** (*uint8xm1_t a*, *uint8xm1_t b*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint8xm2_t* **vmergevvm_mask_uint8xm2** (*uint8xm2_t a*, *uint8xm2_t b*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint8xm4_t* **vmergevvm_mask_uint8xm4** (*uint8xm4_t a*, *uint8xm4_t b*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint8xm8_t* **vmergevvm_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* **vmergevxm_mask_int16xm1** (*int16xm1_t a*, short *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *int16xm2_t* **vmergevxm_mask_int16xm2** (*int16xm2_t a*, short *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vmergevxm_mask_int16xm4** (*int16xm4_t a*, short *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vmergevxm_mask_int16xm8** (*int16xm8_t a*, short *b*, *e16xm8_t mask*, unsigned int *gvl*)
- *int32xm1_t* **vmergevxm_mask_int32xm1** (*int32xm1_t a*, int *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vmergevxm_mask_int32xm2** (*int32xm2_t a*, int *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vmergevxm_mask_int32xm4** (*int32xm4_t a*, int *b*, *e32xm4_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vmergevxm_mask_int32xm8** (*int32xm8_t a*, int *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *int64xm1_t* **vmergevxm_mask_int64xm1** (*int64xm1_t a*, long *b*, *e64xm1_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vmergevxm_mask_int64xm2** (*int64xm2_t a*, long *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vmergevxm_mask_int64xm4** (*int64xm4_t a*, long *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vmergevxm_mask_int64xm8** (*int64xm8_t a*, long *b*, *e64xm8_t mask*, unsigned int *gvl*)
- *int8xm1_t* **vmergevxm_mask_int8xm1** (*int8xm1_t a*, signed char *b*, *e8xm1_t mask*, unsigned int *gvl*)
- *int8xm2_t* **vmergevxm_mask_int8xm2** (*int8xm2_t a*, signed char *b*, *e8xm2_t mask*, unsigned int *gvl*)
- *int8xm4_t* **vmergevxm_mask_int8xm4** (*int8xm4_t a*, signed char *b*, *e8xm4_t mask*, unsigned int *gvl*)
- *int8xm8_t* **vmergevxm_mask_int8xm8** (*int8xm8_t a*, signed char *b*, *e8xm8_t mask*, unsigned int *gvl*)
- *uint16xm1_t* **vmergevxm_mask_uint16xm1** (*uint16xm1_t a*, unsigned short *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint16xm2_t* **vmergevxm_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* **vrgathervi_float16xm1** (*float16xm1_t a*, const unsigned short *b*, unsigned int *gvl*)
- *float16xm2_t* **vrgathervi_float16xm2** (*float16xm2_t a*, const unsigned short *b*, unsigned int *gvl*)
- *float16xm4_t* **vrgathervi_float16xm4** (*float16xm4_t a*, const unsigned short *b*, unsigned int *gvl*)
- *float16xm8_t* **vrgathervi_float16xm8** (*float16xm8_t a*, const unsigned short *b*, unsigned int *gvl*)
- *float32xm1_t* **vrgathervi_float32xm1** (*float32xm1_t a*, const unsigned int *b*, unsigned int *gvl*)
- *float32xm2_t* **vrgathervi_float32xm2** (*float32xm2_t a*, const unsigned int *b*, unsigned int *gvl*)
- *float32xm4_t* **vrgathervi_float32xm4** (*float32xm4_t a*, const unsigned int *b*, unsigned int *gvl*)
- *float32xm8_t* **vrgathervi_float32xm8** (*float32xm8_t a*, const unsigned int *b*, unsigned int *gvl*)
- *float64xm1_t* **vrgathervi_float64xm1** (*float64xm1_t a*, const unsigned long *b*, unsigned int *gvl*)
- *float64xm2_t* **vrgathervi_float64xm2** (*float64xm2_t a*, const unsigned long *b*, unsigned int *gvl*)
- *float64xm4_t* **vrgathervi_float64xm4** (*float64xm4_t a*, const unsigned long *b*, unsigned int *gvl*)
- *float64xm8_t* **vrgathervi_float64xm8** (*float64xm8_t a*, const unsigned long *b*, unsigned int *gvl*)
- *int16xm1_t* **vrgathervi_int16xm1** (*int16xm1_t a*, const unsigned short *b*, unsigned int *gvl*)
- *int16xm2_t* **vrgathervi_int16xm2** (*int16xm2_t a*, const unsigned short *b*, unsigned int *gvl*)
- *int16xm4_t* **vrgathervi_int16xm4** (*int16xm4_t a*, const unsigned short *b*, unsigned int *gvl*)
- *int16xm8_t* **vrgathervi_int16xm8** (*int16xm8_t a*, const unsigned short *b*, unsigned int *gvl*)
- *int32xm1_t* **vrgathervi_int32xm1** (*int32xm1_t a*, const unsigned int *b*, unsigned int *gvl*)
- *int32xm2_t* **vrgathervi_int32xm2** (*int32xm2_t a*, const unsigned int *b*, unsigned int *gvl*)
- *int32xm4_t* **vrgathervi_int32xm4** (*int32xm4_t a*, const unsigned int *b*, unsigned int *gvl*)
- *int32xm8_t* **vrgathervi_int32xm8** (*int32xm8_t a*, const unsigned int *b*, unsigned int *gvl*)
- *int64xm1_t* **vrgathervi_int64xm1** (*int64xm1_t a*, const unsigned long *b*, unsigned int *gvl*)
- *int64xm2_t* **vrgathervi_int64xm2** (*int64xm2_t a*, const unsigned long *b*, unsigned int *gvl*)
- *int64xm4_t* **vrgathervi_int64xm4** (*int64xm4_t a*, const unsigned long *b*, unsigned int *gvl*)
- *int64xm8_t* **vrgathervi_int64xm8** (*int64xm8_t a*, const unsigned long *b*, unsigned int *gvl*)
- *int8xm1_t* **vrgathervi_int8xm1** (*int8xm1_t a*, const unsigned char *b*, unsigned int *gvl*)
- *int8xm2_t* **vrgathervi_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[elemnt] = 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[elemnt] = 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[elemnt] = 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* **vslide1downvx_float32xm4** (*float32xm4_t a*, long *b*, unsigned int *gvl*)
- *float32xm8_t* **vslide1downvx_float32xm8** (*float32xm8_t a*, long *b*, unsigned int *gvl*)
- *float64xm1_t* **vslide1downvx_float64xm1** (*float64xm1_t a*, long *b*, unsigned int *gvl*)
- *float64xm2_t* **vslide1downvx_float64xm2** (*float64xm2_t a*, long *b*, unsigned int *gvl*)
- *float64xm4_t* **vslide1downvx_float64xm4** (*float64xm4_t a*, long *b*, unsigned int *gvl*)
- *float64xm8_t* **vslide1downvx_float64xm8** (*float64xm8_t a*, long *b*, unsigned int *gvl*)
- *int16xm1_t* **vslide1downvx_int16xm1** (*int16xm1_t a*, long *b*, unsigned int *gvl*)
- *int16xm2_t* **vslide1downvx_int16xm2** (*int16xm2_t a*, long *b*, unsigned int *gvl*)
- *int16xm4_t* **vslide1downvx_int16xm4** (*int16xm4_t a*, long *b*, unsigned int *gvl*)
- *int16xm8_t* **vslide1downvx_int16xm8** (*int16xm8_t a*, long *b*, unsigned int *gvl*)
- *int32xm1_t* **vslide1downvx_int32xm1** (*int32xm1_t a*, long *b*, unsigned int *gvl*)
- *int32xm2_t* **vslide1downvx_int32xm2** (*int32xm2_t a*, long *b*, unsigned int *gvl*)
- *int32xm4_t* **vslide1downvx_int32xm4** (*int32xm4_t a*, long *b*, unsigned int *gvl*)
- *int32xm8_t* **vslide1downvx_int32xm8** (*int32xm8_t a*, long *b*, unsigned int *gvl*)
- *int64xm1_t* **vslide1downvx_int64xm1** (*int64xm1_t a*, long *b*, unsigned int *gvl*)
- *int64xm2_t* **vslide1downvx_int64xm2** (*int64xm2_t a*, long *b*, unsigned int *gvl*)
- *int64xm4_t* **vslide1downvx_int64xm4** (*int64xm4_t a*, long *b*, unsigned int *gvl*)
- *int64xm8_t* **vslide1downvx_int64xm8** (*int64xm8_t a*, long *b*, unsigned int *gvl*)
- *int8xm1_t* **vslide1downvx_int8xm1** (*int8xm1_t a*, long *b*, unsigned int *gvl*)
- *int8xm2_t* **vslide1downvx_int8xm2** (*int8xm2_t a*, long *b*, unsigned int *gvl*)
- *int8xm4_t* **vslide1downvx_int8xm4** (*int8xm4_t a*, long *b*, unsigned int *gvl*)
- *int8xm8_t* **vslide1downvx_int8xm8** (*int8xm8_t a*, long *b*, unsigned int *gvl*)
- *uint16xm1_t* **vslide1downvx_uint16xm1** (*uint16xm1_t a*, long *b*, unsigned int *gvl*)
- *uint16xm2_t* **vslide1downvx_uint16xm2** (*uint16xm2_t a*, long *b*, unsigned int *gvl*)
- *uint16xm4_t* **vslide1downvx_uint16xm4** (*uint16xm4_t a*, long *b*, unsigned int *gvl*)
- *uint16xm8_t* **vslide1downvx_uint16xm8** (*uint16xm8_t a*, long *b*, unsigned int *gvl*)
- *uint32xm1_t* **vslide1downvx_uint32xm1** (*uint32xm1_t a*, long *b*, unsigned int *gvl*)
- *uint32xm2_t* **vslide1downvx_uint32xm2** (*uint32xm2_t a*, long *b*, unsigned int *gvl*)
- *uint32xm4_t* **vslide1downvx_uint32xm4** (*uint32xm4_t a*, long *b*, unsigned int *gvl*)
- *uint32xm8_t* **vslide1downvx_uint32xm8** (*uint32xm8_t a*, long *b*, unsigned int *gvl*)
- *uint64xm1_t* **vslide1downvx_uint64xm1** (*uint64xm1_t a*, long *b*, unsigned int *gvl*)
- *uint64xm2_t* **vslide1downvx_uint64xm2** (*uint64xm2_t a*, long *b*, unsigned int *gvl*)
- *uint64xm4_t* **vslide1downvx_uint64xm4** (*uint64xm4_t a*, long *b*, unsigned int *gvl*)
- *uint64xm8_t* **vslide1downvx_uint64xm8** (*uint64xm8_t a*, long *b*, unsigned int *gvl*)
- *uint8xm1_t* **vslide1downvx_uint8xm1** (*uint8xm1_t a*, long *b*, unsigned int *gvl*)
- *uint8xm2_t* **vslide1downvx_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* **vslide1upvx_int32xm8** (*int32xm8_t a*, long *b*, unsigned int *gvl*)
- *int64xm1_t* **vslide1upvx_int64xm1** (*int64xm1_t a*, long *b*, unsigned int *gvl*)
- *int64xm2_t* **vslide1upvx_int64xm2** (*int64xm2_t a*, long *b*, unsigned int *gvl*)
- *int64xm4_t* **vslide1upvx_int64xm4** (*int64xm4_t a*, long *b*, unsigned int *gvl*)
- *int64xm8_t* **vslide1upvx_int64xm8** (*int64xm8_t a*, long *b*, unsigned int *gvl*)
- *int8xm1_t* **vslide1upvx_int8xm1** (*int8xm1_t a*, long *b*, unsigned int *gvl*)
- *int8xm2_t* **vslide1upvx_int8xm2** (*int8xm2_t a*, long *b*, unsigned int *gvl*)
- *int8xm4_t* **vslide1upvx_int8xm4** (*int8xm4_t a*, long *b*, unsigned int *gvl*)
- *int8xm8_t* **vslide1upvx_int8xm8** (*int8xm8_t a*, long *b*, unsigned int *gvl*)
- *uint16xm1_t* **vslide1upvx_uint16xm1** (*uint16xm1_t a*, long *b*, unsigned int *gvl*)
- *uint16xm2_t* **vslide1upvx_uint16xm2** (*uint16xm2_t a*, long *b*, unsigned int *gvl*)
- *uint16xm4_t* **vslide1upvx_uint16xm4** (*uint16xm4_t a*, long *b*, unsigned int *gvl*)
- *uint16xm8_t* **vslide1upvx_uint16xm8** (*uint16xm8_t a*, long *b*, unsigned int *gvl*)
- *uint32xm1_t* **vslide1upvx_uint32xm1** (*uint32xm1_t a*, long *b*, unsigned int *gvl*)
- *uint32xm2_t* **vslide1upvx_uint32xm2** (*uint32xm2_t a*, long *b*, unsigned int *gvl*)
- *uint32xm4_t* **vslide1upvx_uint32xm4** (*uint32xm4_t a*, long *b*, unsigned int *gvl*)
- *uint32xm8_t* **vslide1upvx_uint32xm8** (*uint32xm8_t a*, long *b*, unsigned int *gvl*)
- *uint64xm1_t* **vslide1upvx_uint64xm1** (*uint64xm1_t a*, long *b*, unsigned int *gvl*)
- *uint64xm2_t* **vslide1upvx_uint64xm2** (*uint64xm2_t a*, long *b*, unsigned int *gvl*)
- *uint64xm4_t* **vslide1upvx_uint64xm4** (*uint64xm4_t a*, long *b*, unsigned int *gvl*)
- *uint64xm8_t* **vslide1upvx_uint64xm8** (*uint64xm8_t a*, long *b*, unsigned int *gvl*)
- *uint8xm1_t* **vslide1upvx_uint8xm1** (*uint8xm1_t a*, long *b*, unsigned int *gvl*)
- *uint8xm2_t* **vslide1upvx_uint8xm2** (*uint8xm2_t a*, long *b*, unsigned int *gvl*)
- *uint8xm4_t* **vslide1upvx_uint8xm4** (*uint8xm4_t a*, long *b*, unsigned int *gvl*)
- *uint8xm8_t* **vslide1upvx_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* **vslide1upvx_mask_float16xm1** (*float16xm1_t merge*, *float16xm1_t a*, long *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *float16xm2_t* **vslide1upvx_mask_float16xm2** (*float16xm2_t merge*, *float16xm2_t a*, long *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *float16xm4_t* **vslide1upvx_mask_float16xm4** (*float16xm4_t merge*, *float16xm4_t a*, long *b*, *e16xm4_t mask*, unsigned int *gvl*)

- *float16xm8_t* **vslide1upvx_mask_float16xm8** (*float16xm8_t merge*, *float16xm8_t a*, long *b*, *e16xm8_t mask*, unsigned int *gvl*)
- *float32xm1_t* **vslide1upvx_mask_float32xm1** (*float32xm1_t merge*, *float32xm1_t a*, long *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *float32xm2_t* **vslide1upvx_mask_float32xm2** (*float32xm2_t merge*, *float32xm2_t a*, long *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *float32xm4_t* **vslide1upvx_mask_float32xm4** (*float32xm4_t merge*, *float32xm4_t a*, long *b*, *e32xm4_t mask*, unsigned int *gvl*)
- *float32xm8_t* **vslide1upvx_mask_float32xm8** (*float32xm8_t merge*, *float32xm8_t a*, long *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *float64xm1_t* **vslide1upvx_mask_float64xm1** (*float64xm1_t merge*, *float64xm1_t a*, long *b*, *e64xm1_t mask*, unsigned int *gvl*)
- *float64xm2_t* **vslide1upvx_mask_float64xm2** (*float64xm2_t merge*, *float64xm2_t a*, long *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *float64xm4_t* **vslide1upvx_mask_float64xm4** (*float64xm4_t merge*, *float64xm4_t a*, long *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *float64xm8_t* **vslide1upvx_mask_float64xm8** (*float64xm8_t merge*, *float64xm8_t a*, long *b*, *e64xm8_t mask*, unsigned int *gvl*)
- *int16xm1_t* **vslide1upvx_mask_int16xm1** (*int16xm1_t merge*, *int16xm1_t a*, long *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *int16xm2_t* **vslide1upvx_mask_int16xm2** (*int16xm2_t merge*, *int16xm2_t a*, long *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *int16xm4_t* **vslide1upvx_mask_int16xm4** (*int16xm4_t merge*, *int16xm4_t a*, long *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *int16xm8_t* **vslide1upvx_mask_int16xm8** (*int16xm8_t merge*, *int16xm8_t a*, long *b*, *e16xm8_t mask*, unsigned int *gvl*)
- *int32xm1_t* **vslide1upvx_mask_int32xm1** (*int32xm1_t merge*, *int32xm1_t a*, long *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *int32xm2_t* **vslide1upvx_mask_int32xm2** (*int32xm2_t merge*, *int32xm2_t a*, long *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *int32xm4_t* **vslide1upvx_mask_int32xm4** (*int32xm4_t merge*, *int32xm4_t a*, long *b*, *e32xm4_t mask*, unsigned int *gvl*)
- *int32xm8_t* **vslide1upvx_mask_int32xm8** (*int32xm8_t merge*, *int32xm8_t a*, long *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *int64xm1_t* **vslide1upvx_mask_int64xm1** (*int64xm1_t merge*, *int64xm1_t a*, long *b*, *e64xm1_t mask*, unsigned int *gvl*)
- *int64xm2_t* **vslide1upvx_mask_int64xm2** (*int64xm2_t merge*, *int64xm2_t a*, long *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *int64xm4_t* **vslide1upvx_mask_int64xm4** (*int64xm4_t merge*, *int64xm4_t a*, long *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *int64xm8_t* **vslide1upvx_mask_int64xm8** (*int64xm8_t merge*, *int64xm8_t a*, long *b*, *e64xm8_t mask*, unsigned int *gvl*)
- *int8xm1_t* **vslide1upvx_mask_int8xm1** (*int8xm1_t merge*, *int8xm1_t a*, long *b*, *e8xm1_t mask*, unsigned int *gvl*)
- *int8xm2_t* **vslide1upvx_mask_int8xm2** (*int8xm2_t merge*, *int8xm2_t a*, long *b*, *e8xm2_t mask*, unsigned int *gvl*)

- *int8xm4_t* **vslide1upvx_mask_int8xm4** (*int8xm4_t merge*, *int8xm4_t a*, long *b*, *e8xm4_t mask*, unsigned int *gvl*)
- *int8xm8_t* **vslide1upvx_mask_int8xm8** (*int8xm8_t merge*, *int8xm8_t a*, long *b*, *e8xm8_t mask*, unsigned int *gvl*)
- *uint16xm1_t* **vslide1upvx_mask_uint16xm1** (*uint16xm1_t merge*, *uint16xm1_t a*, long *b*, *e16xm1_t mask*, unsigned int *gvl*)
- *uint16xm2_t* **vslide1upvx_mask_uint16xm2** (*uint16xm2_t merge*, *uint16xm2_t a*, long *b*, *e16xm2_t mask*, unsigned int *gvl*)
- *uint16xm4_t* **vslide1upvx_mask_uint16xm4** (*uint16xm4_t merge*, *uint16xm4_t a*, long *b*, *e16xm4_t mask*, unsigned int *gvl*)
- *uint16xm8_t* **vslide1upvx_mask_uint16xm8** (*uint16xm8_t merge*, *uint16xm8_t a*, long *b*, *e16xm8_t mask*, unsigned int *gvl*)
- *uint32xm1_t* **vslide1upvx_mask_uint32xm1** (*uint32xm1_t merge*, *uint32xm1_t a*, long *b*, *e32xm1_t mask*, unsigned int *gvl*)
- *uint32xm2_t* **vslide1upvx_mask_uint32xm2** (*uint32xm2_t merge*, *uint32xm2_t a*, long *b*, *e32xm2_t mask*, unsigned int *gvl*)
- *uint32xm4_t* **vslide1upvx_mask_uint32xm4** (*uint32xm4_t merge*, *uint32xm4_t a*, long *b*, *e32xm4_t mask*, unsigned int *gvl*)
- *uint32xm8_t* **vslide1upvx_mask_uint32xm8** (*uint32xm8_t merge*, *uint32xm8_t a*, long *b*, *e32xm8_t mask*, unsigned int *gvl*)
- *uint64xm1_t* **vslide1upvx_mask_uint64xm1** (*uint64xm1_t merge*, *uint64xm1_t a*, long *b*, *e64xm1_t mask*, unsigned int *gvl*)
- *uint64xm2_t* **vslide1upvx_mask_uint64xm2** (*uint64xm2_t merge*, *uint64xm2_t a*, long *b*, *e64xm2_t mask*, unsigned int *gvl*)
- *uint64xm4_t* **vslide1upvx_mask_uint64xm4** (*uint64xm4_t merge*, *uint64xm4_t a*, long *b*, *e64xm4_t mask*, unsigned int *gvl*)
- *uint64xm8_t* **vslide1upvx_mask_uint64xm8** (*uint64xm8_t merge*, *uint64xm8_t a*, long *b*, *e64xm8_t mask*, unsigned int *gvl*)
- *uint8xm1_t* **vslide1upvx_mask_uint8xm1** (*uint8xm1_t merge*, *uint8xm1_t a*, long *b*, *e8xm1_t mask*, unsigned int *gvl*)
- *uint8xm2_t* **vslide1upvx_mask_uint8xm2** (*uint8xm2_t merge*, *uint8xm2_t a*, long *b*, *e8xm2_t mask*, unsigned int *gvl*)
- *uint8xm4_t* **vslide1upvx_mask_uint8xm4** (*uint8xm4_t merge*, *uint8xm4_t a*, long *b*, *e8xm4_t mask*, unsigned int *gvl*)
- *uint8xm8_t* **vslide1upvx_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
```

CHAPTER

THREE

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];
}
```

CHAPTER

FOUR

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 >> 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]!=0 | 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]!=0

The rounding function:

roundoff(v, d) = (v >> d) + r

INDEX