APEX

Generated by Doxygen 1.6.1

Mon Aug 13 04:54:35 2018

CONTENTS i

Contents

1	Mod	Module Index							1		
	1.1	Modules							•	 •	1
2	Mod	ule Documentation									1
	2.1	Other Intrinsics									1
		2.1.1 Detailed Descr	iption								3
		2.1.2 Define Docume	entation								3
		2.1.3 Function Docu	mentation								3
	2.2	Arithmetic Intrinsics									8
		2.2.1 Detailed Descr	iption								14
		2.2.2 Function Docu	mentation								14
	2.3	Bitwise Intrinsics									38
		2.3.1 Detailed Descr	iption								40
		2.3.2 Function Docu	mentation								40
	2.4	Shift Intrinsics									49
		2.4.1 Detailed Descr	iption								50
		2.4.2 Function Docu	mentation								50
	2.5	Comparison Intrinsics									57
		2.5.1 Detailed Descr	iption								59
		2.5.2 Function Docu	mentation								59
	2.6	Element Intrinsics									68
		2.6.1 Detailed Descr	iption								70
		2.6.2 Function Docu	mentation								70
	2.7	Memory Intrinsics									79
		2.7.1 Detailed Descr	iption								82
		2.7.2 Function Docu	mentation								82
	2.8	Specialized Shift Intrin	sics								92
		2.8.1 Detailed Descr	iption								92
		2.8.2 Function Docu	mentation								92
	2.9	Specialized Multiplicat	tion Intrinsics								95
		2.9.1 Detailed Descr	iption								98
		2.9.2 Function Docu	mentation								98
	2.10	Swap Intrinsics									111

1 Module Index 1

2.10.1 Detailed Description	112
2.10.2 Function Documentation	112
2.11 Move/Rotate Intrinsics	115
2.11.1 Detailed Description	119
2.11.2 Function Documentation	119
4 M 1 1 T 1	
1 Module Index	
1.1 Modules	
Here is a list of all modules:	
Other Intrinsics	1
Arithmetic Intrinsics	8
Bitwise Intrinsics	38
Shift Intrinsics	49
Comparison Intrinsics	57
Element Intrinsics	68
Memory Intrinsics	79
Specialized Shift Intrinsics	92
Specialized Multiplication Intrinsics	95
Swap Intrinsics	111
Move/Rotate Intrinsics	115
2 Module Documentation	
2.1 Other Intrinsics	
Other intrinsics.	
Defines	
• #define wait(imm)builtin_apex_wait(imm)	
Wait for a given number of cycles.	

Functions

- int pcnt (unsigned int a)

 Pop count.
- int clz (unsigned int a)

 Count leading zero.
- int clb (unsigned int a)

 Count leading same bits.
- int select (bool c, int a, int b)

 Signed integer select.
- unsigned select (bool c, unsigned a, unsigned b)

 Unsigned integer select.
- bool vany (vbool v)

 Vector count any.
- bool vall (vbool v)

 Vector count all.
- vec16s vhaddss (vec16s a, vec16s b)

 Signed vector add and half.
- vec16u vhadduu (vec16u a, vec16u b)

 Unsigned vector add and half.
- vec16s vrhaddss (vec16s a, vec16s b)
 Signed vector add, plus one then half.
- vec16u vrhadduu (vec16u a, vec16u b)

 Unsigned vector add, plus one then half.
- vec16u vabs_diffu (vec16u a, vec16u b)

 Unsigned vector absolute difference.
- int haddss (int a, int b)

 Signed add and half.
- unsigned hadduu (unsigned a, unsigned b)

 Unsigned add and half.
- int rhaddss (int a, int b)

 Signed add, plus one then half.

- unsigned rhadduu (unsigned a, unsigned b)

 Unsigned add, plus one then half.
- void swbreak ()

2.1.1 Detailed Description

Other intrinsics.

2.1.2 Define Documentation

2.1.2.1 #define wait(imm) __builtin_apex_wait(imm)

Wait for a given number of cycles. This instruction will prevent the processor from issuing new instructions for the number of cycles described by immediate parameter. This is defined as a macro because the immediate value needs to be encoded in the instruction.

Parameters:

imm The number of cycles to wait

2.1.3 Function Documentation

2.1.3.1 int clb (unsigned int *a*)

Count leading same bits.

Parameters:

a unsigned int

Returns:

int total number of leading identical bits

2.1.3.2 int clz (unsigned int *a*)

Count leading zero.

Parameters:

a unsigned int

Returns:

int total number of leading zero bits

2.1.3.3 int haddss (int a, int b)

Signed add and half.

Parameters:

- a first input int
- **b** second input int

Returns:

int (a + b) / 2

2.1.3.4 unsigned hadduu (unsigned a, unsigned b)

Unsigned add and half.

Parameters:

- a first input int
- b second input int

Returns:

unsigned (a + b) / 2

2.1.3.5 int pcnt (unsigned int a)

Pop count.

Parameters:

a unsigned int

Returns:

int total number of one bits

2.1.3.6 int rhaddss (int a, int b)

Signed add, plus one then half.

Parameters:

- a first input int
- b second input int

Returns:

int
$$(a + b + 1) / 2$$

2.1.3.7 unsigned rhadduu (unsigned a, unsigned b)

Unsigned add, plus one then half.

Parameters:

- a first input int
- b second input int

Returns:

int
$$(a + b + 1) / 2$$

2.1.3.8 unsigned select (bool c, unsigned a, unsigned b)

Unsigned integer select.

Parameters:

- a first input unsigned int
- b second input unsigned int
- c control boolean for select

Returns:

int return a if true else b

6

2.1.3.9 int select (bool c, int a, int b)

Signed integer select.

Parameters:

- a first input signed int
- **b** second input signed int
- c control boolean for select

Returns:

int return a if true else b

2.1.3.10 void swbreak ()

2.1.3.11 vec16u vabs_diffu (vec16u a, vec16u b)

Unsigned vector absolute difference.

Parameters:

- a first input vector
- b second input vector

Returns:

vector | va - vb |

2.1.3.12 bool vall (vbool *v*)

Vector count all.

Parameters:

v Input vector

Returns:

bool True if all CU is set

2.1.3.13 bool vany (vbool ν)

Vector count any.

Parameters:

v Input vector

Returns:

bool True if any CU is set

2.1.3.14 vec16s vhaddss (vec16s a, vec16s b)

Signed vector add and half.

Parameters:

- a first input vector
- b second input vector

Returns:

vector (va + vb) / 2

2.1.3.15 vec16u vhadduu (vec16u a, vec16u b)

Unsigned vector add and half.

Parameters:

- a first input vector
- b second input vector

Returns:

vector (va + vb) / 2

2.1.3.16 vec16s vrhaddss (vec16s a, vec16s b)

Signed vector add, plus one then half.

Parameters:

- a first input vector
- b second input vector

Returns:

```
vector (va + vb + 1)/2
```

2.1.3.17 vec16u vrhadduu (vec16u *a*, vec16u *b*)

Unsigned vector add, plus one then half.

Parameters:

- a first input vector
- b second input vector

Returns:

```
vector (va + vb + 1)/2
```

2.2 Arithmetic Intrinsics

Vector arithmetic intrinsics.

Functions

- vec16s vadd (vec16s va, vec16s vb)

 Addition of two vectors.
- vec16u vadd (vec16u va, vec16u vb)

 Addition of two vectors.
- vec32s vadd (vec32s va, vec32s vb)

 Addition of two vectors.
- vec32u vadd (vec32u va, vec32u vb)

 Addition of two vectors.
- vec16s vadd (vec16s va, vec16s vb, vbool *restrict vf)
 Addition of two vectors.
- vec16u vadd (vec16u va, vec16u vb, vbool *restrict vf)

 Addition of two vectors with carry generation.

- vec32s vadd (vec32s va, vec32s vb, vbool *restrict vf)
 Addition of two vectors with carry generation.
- vec32u vadd (vec32u va, vec32u vb, vbool *restrict vf)
 Addition of two vectors with carry generation.
- vec16s vaddx (vec16s va, vec16s vb, vbool vc)

 Addition of two vectors with carry.
- vec16u vaddx (vec16u va, vec16u vb, vbool vc)

 Addition of two vectors with carry.
- vec32s vaddx (vec32s va, vec32s vb, vbool vc)
 Addition of two vectors with carry.
- vec32u vaddx (vec32u va, vec32u vb, vbool vc)

 Addition of two vectors with carry.
- vec16s vaddx (vec16s va, vec16s vb, vbool vc, vbool *restrict vf)
 Addition of two vectors with carry and carry generation.
- vec16u vaddx (vec16u va, vec16u vb, vbool vc, vbool *restrict vf)

 Addition of two vectors with carry and carry generation.
- vec32s vaddx (vec32s va, vec32s vb, vbool vc, vbool *restrict vf)
 Addition of two vectors with carry and carry generation.
- vec32u vaddx (vec32u va, vec32u vb, vbool vc, vbool *restrict vf)

 Addition of two vectors with carry and carry generation.
- vec16s vadd_sat (vec16s va, vec16s vb)
 Addition of two signed vectors, with saturation Positive results will saturate at 0x7FFF and negative results will saturate at 0x8000.
- vec16u vadd_sat (vec16u va, vec16u vb)
 Addition of two unsigned vectors, with saturation Positive results will saturate at 0xFFFE.
- vec16s vsub (vec16s va, vec16s vb)

 Subtract one vector from another.
- vec16u vsub (vec16u va, vec16u vb)

 Subtract one vector from another.
- vec32s vsub (vec32s va, vec32s vb)

Subtract one vector from another.

- vec32u vsub (vec32u va, vec32u vb)

 Subtract one vector from another.
- vec16s vsub (vec16s va, vec16s vb, vbool *restrict vf)
 Subtract one vector from another, with carry generation.
- vec16u vsub (vec16u va, vec16u vb, vbool *restrict vf)
 Subtract one vector from another, with carry generation.
- vec32s vsub (vec32s va, vec32s vb, vbool *restrict vf)
 Subtract one vector from another, with carry generation.
- vec32u vsub (vec32u va, vec32u vb, vbool *restrict vf)
 Subtract one vector from another, with carry generation.
- vec16s vsubx (vec16s va, vec16s vb, vbool vc)
 Subtract one vector and the carry from another vector.
- vec16u vsubx (vec16u va, vec16u vb, vbool vc)
 Subtract one vector and the carry from another vector.
- vec32s vsubx (vec32s va, vec32s vb, vbool vc)
 Subtract one vector and the carry from another vector.
- vec32u vsubx (vec32u va, vec32u vb, vbool vc)
 Subtract one vector and the carry from another vector.
- vec16s vsubx (vec16s va, vec16s vb, vbool vc, vbool *restrict vf)
 Subtract one vector and the carry from another vector, with carry generation.
- vec16u vsubx (vec16u va, vec16u vb, vbool vc, vbool *restrict vf)
 Subtract one vector and the carry from another vector, with carry generation.
- vec32s vsubx (vec32s va, vec32s vb, vbool vc, vbool *restrict vf)
 Subtract one vector and the carry from another vector, with carry generation.
- vec32u vsubx (vec32u va, vec32u vb, vbool vc, vbool *restrict vf)
 Subtract one vector and the carry from another vector, with carry generation.
- vec16s vsub_sat (vec16s va, vec16s vb)
 Signed vector subtraction, with saturation Positive results will saturate at 0x7FFF and negative results will saturate at 0x8000.
- vec16u vsub_sat (vec16u va, vec16u vb)

Unsigned vector subtraction, with saturation Positive results will saturate at 0xFFFF.

- vec16s vabs_diff (vec16s va, vec16s vb)
 Vector absolute difference of 2 signed vectors.
- vec16u vabs_diff (vec16u va, vec16u vb)

 Vector absolute difference of 2 unsigned vectors.
- vec16s vasb (vec16s va, vec16s vb, vbool vc)

 Vector Add/Subtract based on condition.
- vec16u vasb (vec16u a, vec16u b, vbool c)
 Vector Add/Subtract based on condition.
- vec16s vasbs (vec16s a, vec16s b)
 Vector add with absolute value of second vector The instruction will check the first bit of vb to determine it's sign.
- vec16u vasbs (vec16u a, vec16u b)
 Vector add with absolute value of second vector The instruction will check the first bit of vb to determine it's sign.
- vec16s vmul (vec16s va, vec16s vb)

 Multiplication of two vectors.
- vec16u vmul (vec16u va, vec16u vb)

 Multiplication of two vectors.
- vec32s vmul (vec32s va, vec32s vb)

 Multiplication of two vectors.
- vec32u vmul (vec32u va, vec32u vb)
 Multiplication of two vectors.
- void vacl (vec16s *restrict a, vec16u *restrict b, vec16s c) Vector accumulate low (32-bit += 16-bit, zero-extend).
- void vacl (vec16s *restrict a, vec16u *restrict b, vec16u c) Vector accumulate low (32-bit += 16-bit, zero-extend).
- void vacl (vec16u *restrict a, vec16u *restrict b, vec16u c)

 Vector accumulate low (32-bit += 16-bit, zero-extend).
- void vacl (vec32s *restrict a, vec16s c)

 Vector accumulate low (32-bit += 16-bit, sign-extend).
- void vacl (vec32u *restrict a, vec16u c)

Vector accumulate low (32-bit += 16-bit, zero-extend).

- void vacl (vec32s *restrict a, vec16u c)

 Vector accumulate low (32-bit += 16-bit, zero-extend).
- void vacl (vec32u *restrict a, vec16s c)

 Vector accumulate low (32-bit += 16-bit, zero-extend).
- void vacm (vec16s *restrict a, vec16u *restrict b, vec16s c) Vector accumulate middle (32-bit += 16-bit, sign-extend).
- void vacm (vec16s *restrict a, vec16u *restrict b, vec16u c) Vector accumulate middle (32-bit += 16-bit, zero-extend).
- void vacm (vec16u *restrict a, vec16u *restrict b, vec16u c) Vector accumulate middle (32-bit += 16-bit, zero-extend).
- void vacm (vec32s *restrict a, vec16s c)

 Vector accumulate middle (32-bit += 16-bit, sign-extend).
- void vacm (vec32u *restrict a, vec16u c)

 Vector accumulate middle (32-bit += 16-bit, zero-extend).
- void vacm (vec32s *restrict a, vec16u c)

 Vector accumulate middle (32-bit += 16-bit, zero-extend).
- void vacm (vec32u *restrict a, vec16s c)

 Vector accumulate middle (32-bit += 16-bit, zero-extend).
- void vach (vec16s *restrict a, vec16u *restrict b, vec16s c)

 Vector accumulate high (32-bit += 16-bit).
- void vach (vec16s *restrict a, vec16u *restrict b, vec16u c) Vector accumulate high (32-bit += 16-bit).
- void vach (vec32s *restrict a, vec16s c)

 Vector accumulate high (32-bit += 16-bit).
- void vach (vec32u *restrict a, vec16u c)

 Vector accumulate high (32-bit += 16-bit).
- void vach (vec32u *restrict a, vec16s c)

 Vector accumulate high (32-bit += 16-bit).
- vec08u vabs (vec08s va) Vector absolute value.

- vec16u vabs (vec16s va) Vector absolute value.
- vec32u vabs (vec32s va) Vector absolute value.
- vec16s vsat (vec16s va, vec16s vb, vec16s vc) Vector saturate with lower and upper bounds.
- vec16u vsat (vec16u va, vec16u vb, vec16u vc) Vector saturate with lower and upper bounds.
- vec16s vclz (vec16s va)
 Vector count leading zero bits.
- vec16u vclz (vec16u va)

 Vector count leading zero bits.
- vec16s vclz (vec32s va)
 Vector count leading zero bits.
- vec16u vclz (vec32u va)
 Vector count leading zero bits.
- vec16s vpcnt (vec16s va)

 Vector count number of 1 bits.
- vec16u vpcnt (vec16u va)

 Vector count number of 1 bits.
- vec16s vpcnt (vec32s va)
 Vector count number of 1 bits.
- vec16u vpcnt (vec32u va)

 Vector count number of 1 bits.
- vec16s vclb (vec16s va)

 Vector count leading same bits.
- vec16u vclb (vec16u va)

 Vector count leading same bits.
- vec16s vclb (vec32s va)

 Vector count leading same bits.

• vec16u vclb (vec32u va)

Vector count leading same bits.

2.2.1 Detailed Description

Vector arithmetic intrinsics.

2.2.2 Function Documentation

2.2.2.1 vec32u vabs (vec32s va)

Vector absolute value.

Parameters:

va Input vector

Returns:

Output absolute vector

2.2.2.2 vec16u vabs (vec16s va)

Vector absolute value.

Parameters:

va Input vector

Returns:

Output absolute vector

2.2.2.3 vec08u vabs (vec08s va)

Vector absolute value.

Parameters:

va Input vector

Returns:

Output absolute vector

2.2.2.4 vec16u vabs_diff (vec16u va, vec16u vb)

Vector absolute difference of 2 unsigned vectors.

Parameters:

- va The first vector
- vb The second vector

Returns:

va - vb

2.2.2.5 vec16s vabs_diff (vec16s va, vec16s vb)

Vector absolute difference of 2 signed vectors.

Parameters:

- va The first vector
- *vb* The second vector

Returns:

va - vb

2.2.2.6 void vach (vec32u *restrict a, vec16s c)

Vector accumulate high (32-bit += 16-bit).

Parameters:

- a Output and first input vector
- c The second input vector

2.2.2.7 void vach (vec32u *restrict a, vec16u c)

Vector accumulate high (32-bit += 16-bit).

- a Output and first input vector
- c The second input vector

2.2.2.8 void vach (vec32s *restrict a, vec16s c)

Vector accumulate high (32-bit += 16-bit).

Parameters:

- a Output and first input vector
- c The second input vector

2.2.2.9 void vach (vec16s *restrict a, vec16u *restrict b, vec16u c)

Vector accumulate high (32-bit += 16-bit).

Parameters:

- a Output vector
- **b** The first input vector
- c The second input vector

2.2.2.10 void vach (vec16s *restrict a, vec16u *restrict b, vec16s c)

Vector accumulate high (32-bit += 16-bit).

Parameters:

- a Output vector
- **b** The first input vector
- c The second input vector

2.2.2.11 void vacl (vec32u *restrict a, vec16s c)

Vector accumulate low (32-bit += 16-bit, zero-extend).

- a Output and first input vector
- c The second input vector

2.2.2.12 void vacl (vec32s *restrict a, vec16u c)

Vector accumulate low (32-bit += 16-bit, zero-extend).

Parameters:

- a Output and first input vector
- c The second input vector

2.2.2.13 void vacl (vec32u *restrict a, vec16u c)

Vector accumulate low (32-bit += 16-bit, zero-extend).

Parameters:

- a Output and first input vector
- c The second input vector

2.2.2.14 void vacl (vec32s *restrict a, vec16s c)

Vector accumulate low (32-bit += 16-bit, sign-extend).

Parameters:

- a Output and first input vector
- c The second input vector

2.2.2.15 void vacl (vec16u *restrict a, vec16u *restrict b, vec16u c)

Vector accumulate low (32-bit += 16-bit, zero-extend).

- a Output vector
- **b** The first input vector
- c The second input vector

2.2.2.16 void vacl (vec16s *restrict a, vec16u *restrict b, vec16u c)

Vector accumulate low (32-bit += 16-bit, zero-extend).

Parameters:

- a Output vector
- **b** The first input vector
- c The second input vector

2.2.2.17 void vacl (vec16s *restrict a, vec16u *restrict b, vec16s c)

Vector accumulate low (32-bit += 16-bit, zero-extend).

Parameters:

- a Output vector
- **b** The first input vector
- c The second input vector

2.2.2.18 void vacm (vec32u *restrict a, vec16s c)

Vector accumulate middle (32-bit += 16-bit, zero-extend).

Parameters:

- a Output and first input vector
- c The second input vector

2.2.2.19 void vacm (vec32s *restrict a, vec16u c)

Vector accumulate middle (32-bit += 16-bit, zero-extend).

- a Output and first input vector
- c The second input vector

2.2.2.20 void vacm (vec32u *restrict a, vec16u c)

Vector accumulate middle (32-bit += 16-bit, zero-extend).

Parameters:

- a Output and first input vector
- c The second input vector

2.2.2.21 void vacm (vec32s *restrict a, vec16s c)

Vector accumulate middle (32-bit += 16-bit, sign-extend).

Parameters:

- a Output and first input vector
- c The second input vector

2.2.2.22 void vacm (vec16u *restrict a, vec16u *restrict b, vec16u c)

Vector accumulate middle (32-bit += 16-bit, zero-extend).

Parameters:

- a Output vector
- **b** The first input vector
- c The second input vector

2.2.2.23 void vacm (vec16s *restrict a, vec16u *restrict b, vec16u c)

Vector accumulate middle (32-bit += 16-bit, zero-extend).

- a Output vector
- **b** The first input vector
- c The second input vector

2.2.2.24 void vacm (vec16s *restrict a, vec16u *restrict b, vec16s c)

Vector accumulate middle (32-bit += 16-bit, sign-extend).

Parameters:

- a Output vector
- **b** The first input vector
- c The second input vector

2.2.2.25 vec32u vadd (vec32u va, vec32u vb, vbool *restrict vf)

Addition of two vectors with carry generation.

Parameters:

- va The first vector
- *vb* The second vector
- $\rightarrow vf$ Pointer to the carry

Returns:

va + vb

2.2.2.26 vec32s vadd (vec32s va, vec32s vb, vbool *restrict vf)

Addition of two vectors with carry generation.

Parameters:

- va The first vector
- vb The second vector
- $\rightarrow vf$ Pointer to the carry

Returns:

va + vb

2.2.2.27 vec16u vadd (vec16u va, vec16u vb, vbool *restrict vf)

Addition of two vectors with carry generation.

Parameters:

- va The first vector
- *vb* The second vector
- $\rightarrow vf$ Pointer to the carry

Returns:

va + vb

2.2.2.28 vec16s vadd (vec16s va, vec16s vb, vbool *restrict vf)

Addition of two vectors.

Parameters:

- va The first vector
- *vb* The second vector
- $\rightarrow vf$ Pointer to the carry

Returns:

va + vb

2.2.2.29 vec32u vadd (vec32u va, vec32u vb)

Addition of two vectors.

Parameters:

- va The first vector
- *vb* The second vector

Returns:

va + vb

2.2.2.30 vec32s vadd (vec32s va, vec32s vb)

Addition of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va + vb

2.2.2.31 vec16u vadd (vec16u va, vec16u vb)

Addition of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va + vb

2.2.2.32 vec16s vadd (vec16s va, vec16s vb)

Addition of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va + vb

2.2.2.33 vec16u vadd_sat (vec16u va, vec16u vb)

Addition of two unsigned vectors, with saturation Positive results will saturate at 0xFFFF.

Parameters:

va The first vector

vb The second vector

Returns:

va + vb

2.2.2.34 vec16s vadd_sat (vec16s va, vec16s vb)

Addition of two signed vectors, with saturation Positive results will saturate at 0x7FFF and negative results will saturate at 0x8000.

Parameters:

- va The first vector
- vb The second vector

Returns:

va + vb

2.2.2.35 vec32u vaddx (vec32u va, vec32u vb, vbool vc, vbool *restrict vf)

Addition of two vectors with carry and carry generation.

Parameters:

- va The first vector
- *vb* The second vector
- vc The input carry
- $\rightarrow vf$ Pointer to the output carry

Returns:

va + vb + vc

2.2.2.36 vec32s vaddx (vec32s va, vec32s vb, vbool vc, vbool *restrict vf)

Addition of two vectors with carry and carry generation.

Parameters:

- va The first vector
- *vb* The second vector
- vc The input carry
- $\rightarrow vf$ Pointer to the output carry

Returns:

va + vb + vc

2.2.2.37 vec16u vaddx (vec16u va, vec16u vb, vbool vc, vbool *restrict vf)

Addition of two vectors with carry and carry generation.

Parameters:

- va The first vector
- *vb* The second vector
- vc The input carry
- $\rightarrow vf$ Pointer to the output carry

Returns:

```
va + vb + vc
```

2.2.2.38 vec16s vaddx (vec16s va, vec16s vb, vbool vc, vbool *restrict vf)

Addition of two vectors with carry and carry generation.

Parameters:

- va The first vector
- *vb* The second vector
- vc The input carry
- $\rightarrow vf$ Pointer to the output carry

Returns:

```
va + vb + vc
```

2.2.2.39 vec32u vaddx (vec32u va, vec32u vb, vbool vc)

Addition of two vectors with carry.

Parameters:

- va The first vector
- *vb* The second vector
- vc The carry

Returns:

$$va + vb + vc$$

2.2.2.40 vec32s vaddx (vec32s va, vec32s vb, vbool vc)

Addition of two vectors with carry.

Parameters:

- va The first vector
- *vb* The second vector
- vc The carry

Returns:

va + vb + vc

2.2.2.41 vec16u vaddx (vec16u va, vec16u vb, vbool vc)

Addition of two vectors with carry.

Parameters:

- va The first vector
- *vb* The second vector
- vc The carry

Returns:

va + vb + vc

2.2.2.42 vec16s vaddx (vec16s va, vec16s vb, vbool vc)

Addition of two vectors with carry.

Parameters:

- va The first vector
- *vb* The second vector
- vc The carry

Returns:

va + vb + vc

2.2.2.43 vec16u vasb (vec16u a, vec16u b, vbool c)

Vector Add/Subtract based on condition.

Parameters:

- a The first vector
- **b** The second vector
- *c* The condition

Returns:

```
vc ? (va + vb) : (va - vb)
```

2.2.2.44 vec16s vasb (vec16s va, vec16s vb, vbool vc)

Vector Add/Subtract based on condition.

Parameters:

- va The first vector
- *vb* The second vector
- vc The condition

Returns:

```
vc ? (va + vb) : (va - vb)
```

2.2.2.45 vec16u vasbs (vec16u a, vec16u b)

Vector add with absolute value of second vector The instruction will check the first bit of vb to determine it's sign.

Parameters:

- a The first vector
- b The second vector

Returns:

$$va + |vb|$$

2.2.2.46 vec16s vasbs (vec16s a, vec16s b)

Vector add with absolute value of second vector The instruction will check the first bit of vb to determine it's sign.

Parameters:

- a The first vector
- b The second vector

Returns:

va + |vb|

2.2.2.47 vec16u vclb (vec32u va)

Vector count leading same bits.

Parameters:

va Input vector

Returns:

vector of leading same bits count

2.2.2.48 vec16s vclb (vec32s va)

Vector count leading same bits.

Parameters:

va Input vector

Returns:

vector of leading same bits count

2.2.2.49 vec16u vclb (vec16u va)

Vector count leading same bits.

Parameters:

va Input vector

Returns:

vector of leading same bits count

2.2.2.50 vec16s vclb (vec16s va)

Vector count leading same bits.

Parameters:

va Input vector

Returns:

vector of leading same bits count

2.2.2.51 vec16u vclz (vec32u va)

Vector count leading zero bits.

Parameters:

va Input vector

Returns:

vector of leading zero bits count

2.2.2.52 vec16s vclz (vec32s va)

Vector count leading zero bits.

Parameters:

va Input vector

Returns:

vector of leading zero bits count

2.2.2.53 vec16u vclz (vec16u va)

Vector count leading zero bits.

Parameters:

va Input vector

Returns:

vector of leading zero bits count

2.2.2.54 vec16s vclz (vec16s va)

Vector count leading zero bits.

Parameters:

va Input vector

Returns:

vector of leading zero bits count

2.2.2.55 vec32u vmul (vec32u va, vec32u vb)

Multiplication of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va * vb

2.2.2.56 vec32s vmul (vec32s va, vec32s vb)

Multiplication of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va * vb

2.2.2.57 vec16u vmul (vec16u va, vec16u vb)

Multiplication of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va * vb

2.2.2.58 vec16s vmul (vec16s *va*, vec16s *vb*)

Multiplication of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va * vb

2.2.2.59 vec16u vpcnt (vec32u va)

Vector count number of 1 bits.

Parameters:

va Input vector

Returns:

vector of 1 bits count

2.2.2.60 vec16s vpcnt (vec32s va)

Vector count number of 1 bits.

Parameters:

va Input vector

Returns:

vector of 1 bits count

2.2.2.61 vec16u vpcnt (vec16u va)

Vector count number of 1 bits.

Parameters:

va Input vector

Returns:

vector of 1 bits count

2.2.2.62 vec16s vpcnt (vec16s va)

Vector count number of 1 bits.

Parameters:

va Input vector

Returns:

vector of 1 bits count

2.2.2.63 vec16u vsat (vec16u va, vec16u vb, vec16u vc)

Vector saturate with lower and upper bounds.

Parameters:

va Input source vector

```
vb Input vector of lower boundvc Input vector of upper bound
```

Returns:

Output saturated vector

2.2.2.64 vec16s vsat (vec16s va, vec16s vb, vec16s vc)

Vector saturate with lower and upper bounds.

Parameters:

```
va Input source vector
```

vb Input vector of lower bound

vc Input vector of upper bound

Returns:

Output saturated vector

2.2.2.65 vec32u vsub (vec32u va, vec32u vb, vbool *restrict vf)

Subtract one vector from another, with carry generation.

Parameters:

```
va The first vector
```

vb The second vector

 $\rightarrow vf$ Pointer to the carry

Returns:

va - vb

2.2.2.66 vec32s vsub (vec32s va, vec32s vb, vbool *restrict vf)

Subtract one vector from another, with carry generation.

Parameters:

va The first vector

```
vb The second vector
```

 $\rightarrow vf$ Pointer to the carry

Returns:

va - vb

2.2.2.67 vec16u vsub (vec16u va, vec16u vb, vbool *restrict vf)

Subtract one vector from another, with carry generation.

Parameters:

```
va The first vector
```

vb The second vector

 $\rightarrow vf$ Pointer to the carry

Returns:

va - vb

2.2.2.68 vec16s vsub (vec16s va, vec16s vb, vbool *restrict vf)

Subtract one vector from another, with carry generation.

Parameters:

```
va The first vector
```

vb The second vector

 $\rightarrow vf$ Pointer to the carry

Returns:

va - vb

2.2.2.69 vec32u vsub (vec32u va, vec32u vb)

Subtract one vector from another.

Parameters:

va The first vector

vb The second vector

Returns:

va - vb

2.2.2.70 vec32s vsub (vec32s va, vec32s vb)

Subtract one vector from another.

Parameters:

va The first vector

vb The second vector

Returns:

va - vb

2.2.2.71 vec16u vsub (vec16u va, vec16u vb)

Subtract one vector from another.

Parameters:

va The first vector

vb The second vector

Returns:

va - vb

2.2.2.72 vec16s vsub (vec16s va, vec16s vb)

Subtract one vector from another.

Parameters:

va The first vector

vb The second vector

Returns:

va - vb

2.2.2.73 vec16u vsub_sat (vec16u va, vec16u vb)

Unsigned vector subtraction, with saturation Positive results will saturate at 0xFFFF.

Parameters:

- va The first vector
- *vb* The second vector

Returns:

va - vb

2.2.2.74 vec16s vsub_sat (vec16s va, vec16s vb)

Signed vector subtraction, with saturation Positive results will saturate at 0x7FFF and negative results will saturate at 0x8000.

Parameters:

- va The first vector
- *vb* The second vector

Returns:

va - vb

2.2.2.75 vec32u vsubx (vec32u va, vec32u vb, vbool vc, vbool *restrict vf)

Subtract one vector and the carry from another vector, with carry generation.

Parameters:

- va The first vector
- *vb* The second vector
- vc The carry
- $\rightarrow vf$ Pointer to the output carry

Returns:

va - vb - vc

2.2.2.76 vec32s vsubx (vec32s va, vec32s vb, vbool vc, vbool *restrict vf)

Subtract one vector and the carry from another vector, with carry generation.

Parameters:

- va The first vector
- vb The second vector
- *vc* The carry
- $\rightarrow vf$ Pointer to the output carry

Returns:

va - vb - vc

2.2.2.77 vec16u vsubx (vec16u va, vec16u vb, vbool vc, vbool *restrict vf)

Subtract one vector and the carry from another vector, with carry generation.

Parameters:

- va The first vector
- *vb* The second vector
- vc The carry
- $\rightarrow vf$ Pointer to the output carry

Returns:

va - vb - vc

2.2.2.78 vec16s vsubx (vec16s va, vec16s vb, vbool vc, vbool *restrict vf)

Subtract one vector and the carry from another vector, with carry generation.

Parameters:

- va The first vector
- *vb* The second vector
- vc The carry
- $\rightarrow vf$ Pointer to the output carry

Returns:

va - vb - vc

2.2.2.79 vec32u vsubx (vec32u va, vec32u vb, vbool vc)

Subtract one vector and the carry from another vector.

Parameters:

- va The first vector
- *vb* The second vector
- vc The carry

Returns:

va - vb - vc

2.2.2.80 vec32s vsubx (vec32s va, vec32s vb, vbool vc)

Subtract one vector and the carry from another vector.

Parameters:

- va The first vector
- *vb* The second vector
- vc The carry

Returns:

va - vb - vc

2.2.2.81 vec16u vsubx (vec16u va, vec16u vb, vbool vc)

Subtract one vector and the carry from another vector.

Parameters:

- va The first vector
- *vb* The second vector
- vc The carry

Returns:

va - vb - vc

2.2.2.82 vec16s vsubx (vec16s va, vec16s vb, vbool vc)

Subtract one vector and the carry from another vector.

Parameters:

```
va The first vector
```

vb The second vector

vc The carry

Returns:

va - vb - vc

2.3 Bitwise Intrinsics

Vector bitewise intrinsics.

Functions

- vec08s vand (vec08s va, vec08s vb)

 Bitwise And of two vectors.
- vec08u vand (vec08u va, vec08u vb)

 Bitwise And of two vectors.
- vec16s vand (vec16s va, vec16s vb)

 Bitwise And of two vectors.
- vec16u vand (vec16u va, vec16u vb)

 Bitwise And of two vectors.
- vec32s vand (vec32s va, vec32s vb)

 Bitwise And of two vectors.
- vec32u vand (vec32u va, vec32u vb)

 Bitwise And of two vectors.
- vec08s vor (vec08s va, vec08s vb)

 Bitwise Or of two vectors.
- vec08u vor (vec08u va, vec08u vb)

 Bitwise Or of two vectors.
- vec16s vor (vec16s va, vec16s vb)

Bitwise Or of two vectors.

- vec16u vor (vec16u va, vec16u vb)

 Bitwise Or of two vectors.
- vec32s vor (vec32s va, vec32s vb)

 Bitwise Or of two vectors.
- vec32u vor (vec32u va, vec32u vb)

 Bitwise Or of two vectors.
- vec08s vxor (vec08s va, vec08s vb)

 Bitwise Exclusive-Or of two vectors.
- vec08u vxor (vec08u va, vec08u vb)

 Bitwise Exclusive-Or of two vectors.
- vec16s vxor (vec16s va, vec16s vb)
 Bitwise Exclusive-Or of two vectors.
- vec16u vxor (vec16u va, vec16u vb)

 Bitwise Exclusive-Or of two vectors.
- vec32s vxor (vec32s va, vec32s vb)

 Bitwise Exclusive-Or of two vectors.
- vec32u vxor (vec32u va, vec32u vb)

 Bitwise Exclusive-Or of two vectors.
- vec08s vnot (vec08s va)
 Bitwise Not (Vector Ones' Complement) of one vector.
- vec08u vnot (vec08u va)

 Bitwise Not (Vector Ones' Complement) of one vector.
- vec16s vnot (vec16s va)

 Bitwise Not (Vector Ones' Complement) of one vector.
- vec16u vnot (vec16u va)

 Bitwise Not (Vector Ones' Complement) of one vector.
- vec32s vnot (vec32s va)
 Bitwise Not (Vector Ones' Complement) of one vector.
- vec32u vnot (vec32u va)

 Bitwise Not (Vector Ones' Complement) of one vector.

- vec08s vcomplement (vec08s va) Vector Ones' Complement.
- vec08u vcomplement (vec08u va) Vector Ones' Complement.
- vec16s vcomplement (vec16s va) Vector Ones' Complement.
- vec16u vcomplement (vec16u va) Vector Ones' Complement.
- vec32s vcomplement (vec32s va) Vector Ones' Complement.
- vec32u vcomplement (vec32u va) Vector Ones' Complement.

2.3.1 Detailed Description

Vector bitewise intrinsics.

2.3.2 Function Documentation

2.3.2.1 vec32u vand (vec32u va, vec32u vb)

Bitwise And of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va & vb

2.3.2.2 vec32s vand (vec32s va, vec32s vb)

Bitwise And of two vectors.

va The first vector

vb The second vector

Returns:

va & vb

2.3.2.3 vec16u vand (vec16u va, vec16u vb)

Bitwise And of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va & vb

2.3.2.4 vec16s vand (vec16s va, vec16s vb)

Bitwise And of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va & vb

2.3.2.5 vec08u vand (vec08u va, vec08u vb)

Bitwise And of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va & vb

2.3.2.6 vec08s vand (vec08s va, vec08s vb)

Bitwise And of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va & vb

2.3.2.7 vec32u vcomplement (vec32u va)

Vector Ones' Complement.

Parameters:

va The vector

Returns:

vnot(va)

2.3.2.8 vec32s vcomplement (vec32s va)

Vector Ones' Complement.

Parameters:

va The vector

Returns:

vnot(va)

2.3.2.9 vec16u vcomplement (vec16u va)

Vector Ones' Complement.

Parameters:

va The vector

Returns: vnot(va) 2.3.2.10 vec16s vcomplement (vec16s va) Vector Ones' Complement. **Parameters:** va The vector **Returns:** vnot(va) 2.3.2.11 vec08u vcomplement (vec08u va) Vector Ones' Complement. **Parameters:** va The vector **Returns:** vnot(va) 2.3.2.12 vec08s vcomplement (vec08s va) Vector Ones' Complement. **Parameters:** va The vector

Returns:

vnot(va)

2.3.2.13 vec32u vnot (vec32u va)

Bitwise Not (Vector Ones' Complement) of one vector.

Parameters:

va The vector

Returns:

2.3.2.14 vec32s vnot (vec32s va)

Bitwise Not (Vector Ones' Complement) of one vector.

Parameters:

va The vector

Returns:

2.3.2.15 vec16u vnot (vec16u va)

Bitwise Not (Vector Ones' Complement) of one vector.

Parameters:

va The vector

Returns:

2.3.2.16 vec16s vnot (vec16s va)

Bitwise Not (Vector Ones' Complement) of one vector.

Parameters:

va The vector

Returns:

2.3.2.17 vec08u vnot (vec08u va)

Bitwise Not (Vector Ones' Complement) of one vector.

Parameters:

va The vector

Returns:

va [∧] (-1)

2.3.2.18 vec08s vnot (vec08s va)

Bitwise Not (Vector Ones' Complement) of one vector.

Parameters:

va The vector

Returns:

va [∧] (-1)

2.3.2.19 vec32u vor (vec32u va, vec32u vb)

Bitwise Or of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

 $va\mid vb$

2.3.2.20 vec32s vor (vec32s va, vec32s vb)

Bitwise Or of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va | vb

2.3.2.21 vec16u vor (vec16u va, vec16u vb)

Bitwise Or of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va | vb

2.3.2.22 vec16s vor (vec16s va, vec16s vb)

Bitwise Or of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va | vb

2.3.2.23 vec08u vor (vec08u va, vec08u vb)

Bitwise Or of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va | vb

2.3.2.24 vec08s va (vec08s va, vec08s vb)

Bitwise Or of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va | vb

2.3.2.25 vec32u vxor (vec32u va, vec32u vb)

Bitwise Exclusive-Or of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va ^ vb

2.3.2.26 vec32s vxor (vec32s va, vec32s vb)

Bitwise Exclusive-Or of two vectors.

Parameters:

va The first vector

vb The second vector

Returns:

va $^{\wedge}$ vb

2.3.2.27 vec16u vxor (vec16u va, vec16u vb)

Bitwise Exclusive-Or of two vectors.

- va The first vector
- *vb* The second vector

Returns:

 $va \ ^{\wedge} \ vb$

2.3.2.28 vec16s vxor (vec16s va, vec16s vb)

Bitwise Exclusive-Or of two vectors.

Parameters:

- va The first vector
- *vb* The second vector

Returns:

va ^ vb

2.3.2.29 vec08u vxor (vec08u va, vec08u vb)

Bitwise Exclusive-Or of two vectors.

Parameters:

- va The first vector
- *vb* The second vector

Returns:

va ^ vb

2.3.2.30 vec08s vxor (vec08s va, vec08s vb)

Bitwise Exclusive-Or of two vectors.

Parameters:

- va The first vector
- *vb* The second vector

Returns:

va ^ vb

Vector shift intrinsics.

Functions

- vec16s vsl (vec16s va, vec16s vb)

 Vector Shift Left (Logical).
- vec16u vsl (vec16u va, vec16u vb) Vector Shift Left (Logical).
- vec32s vsl (vec32s va, vec16s vb) Vector Shift Left (Logical).
- vec16s vsr (vec16s va, vec16s vb) Vector Shift Right (Arithmetic).
- vec16u vsr (vec16u va, vec16u vb) Vector Shift Right (Arithmetic).
- vec32s vsr (vec32s va, vec32s vb) Vector Shift Right (Arithmetic).
- vec32s vsr (vec32s va, vec16s vb) Vector Shift Right (Arithmetic).
- vec16s vsll (vec16s va, vec16s vb) Vector Shift Left (Logical).
- vec16u vsll (vec16u va, vec16u vb) Vector Shift Left (Logical).
- vec32s vsll (vec32s va, vec32s vb) Vector Shift Left (Logical).
- vec16s vsra (vec16s va, vec16s vb)
 Vector Shift Right (Arithmetic).
- vec16u vsra (vec16u va, vec16u vb)

 Vector Shift Right (Arithmetic).
- vec32s vsra (vec32s va, vec32s vb) Vector Shift Right (Arithmetic).
- vec32s vsra (vec32u va, int s)

Vector Shift Right (Arithmetic).

- vec16s vsrl (vec16s va, vec16s vb) Vector Shift Right (Logical).
- vec16u vsrl (vec16u va, vec16u vb) Vector Shift Right (Logical).
- vec32s vsrl (vec32s va, vec32s vb) Vector Shift Right (Logical).
- vec16s vslo (vec16s va, vbool vc, vbool *restrict vf) Vector Left Shift-in with OV (Logical).
- vec16s vslo (vec16s va, vbool vc) Vector Left Shift-in with OV (Logical).
- vec16s vslc (vec16s va, vbool vc)

 Vector left Shift-in with VC (Logical).
- vec16s vsro (vec16s va, vbool vc, vbool *restrict vf) Vector Right Shift-in with OV (Logical).
- vec16s vsro (vec16s va, vbool vc)

 Vector Right Shift-in with OV (Logical).
- vec16s vsrc (vec16s va, vbool vc)

 Vector Right Shift-in with VC (Logical).

2.4.1 Detailed Description

Vector shift intrinsics.

2.4.2 Function Documentation

2.4.2.1 vec32s vsl (vec32s va, vec16s vb)

Vector Shift Left (Logical).

Parameters:

va The vector

vb The shift amount vector

Returns:

va << vb

2.4.2.2 vec16u vsl (vec16u va, vec16u vb)

Vector Shift Left (Logical).

Parameters:

va The vector

vb The shift amount vector

Returns:

va << vb

2.4.2.3 vec16s vsl (vec16s va, vec16s vb)

Vector Shift Left (Logical).

Parameters:

va The vector

vb The shift amount vector

Returns:

va << vb

2.4.2.4 vec16s vslc (vec16s va, vbool vc)

Vector left Shift-in with VC (Logical).

Parameters:

va Input vector

vc Input vector condition being shifted in to LSB

Returns:

vector va left shift by 1 bit, with vc shifted into LSB

2.4.2.5 vec32s vsll (vec32s va, vec32s vb)

Vector Shift Left (Logical).

Parameters:

va The vector

vb The shift amount vector

Returns:

va << vb

2.4.2.6 vec16u vsll (vec16u va, vec16u vb)

Vector Shift Left (Logical).

Parameters:

va The vector

vb The shift amount vector

Returns:

va << vb

2.4.2.7 vec16s vsll (vec16s va, vec16s vb)

Vector Shift Left (Logical).

Parameters:

va The vector

vb The shift amount vector

Returns:

va << vb

2.4.2.8 vec16s vslo (vec16s va, vbool vc)

Vector Left Shift-in with OV (Logical).

Parameters:

```
va Input vector
```

vc Input OV vector being shifted in to LSB

Returns:

vector va left shift by 1 bit, with vc shifted into LSB

2.4.2.9 vec16s vslo (vec16s va, vbool vc, vbool *restrict vf)

Vector Left Shift-in with OV (Logical).

Parameters:

```
va Input vector
```

vc Input OV vector being shifted in to LSB

vf Output vector of result MSB

Returns:

vector va left shift by 1 bit, with vc shifted into LSB

2.4.2.10 vec32s vsr (vec32s va, vec16s vb)

Vector Shift Right (Arithmetic).

Parameters:

va The vector

vb The shift amount vector

Returns:

va >> vb

2.4.2.11 vec32s vsr (vec32s va, vec32s vb)

Vector Shift Right (Arithmetic).

Parameters:

va The vector

vb The shift amount vector

Returns:

va >> vb

2.4.2.12 vec16u vsr (vec16u va, vec16u vb)

Vector Shift Right (Arithmetic).

Parameters:

va The vector

vb The shift amount vector

Returns:

va >> vb

2.4.2.13 vec16s vsr (vec16s va, vec16s vb)

Vector Shift Right (Arithmetic).

Parameters:

va The vector

vb The shift amount vector

Returns:

va >> vb

2.4.2.14 vec32s vsra (vec32u *va*, int *s*)

Vector Shift Right (Arithmetic).

Parameters:

va The vector

s The shift amount

Returns:

va>>vb

2.4.2.15 vec32s vsra (vec32s va, vec32s vb)

Vector Shift Right (Arithmetic).

Parameters:

va The vector

vb The shift amount vector

Returns:

va >> vb

2.4.2.16 vec16u vsra (vec16u va, vec16u vb)

Vector Shift Right (Arithmetic).

Parameters:

va The vector

vb The shift amount vector

Returns:

va >> vb

2.4.2.17 vec16s vsra (vec16s va, vec16s vb)

Vector Shift Right (Arithmetic).

Parameters:

va The vector

vb The shift amount vector

Returns:

va >> vb

2.4.2.18 vec16s vsrc (vec16s va, vbool vc)

Vector Right Shift-in with VC (Logical).

Parameters:

```
va Input vector
```

vc Input vector condition being shifted in to MSB

Returns:

vector va right shift by 1 bit, with vc shifted into MSB

2.4.2.19 vec32s vsrl (vec32s va, vec32s vb)

Vector Shift Right (Logical).

Parameters:

va The vector

vb The shift amount vector

Returns:

va >> vb

2.4.2.20 vec16u vsrl (vec16u va, vec16u vb)

Vector Shift Right (Logical).

Parameters:

va The vector

vb The shift amount vector

Returns:

va >> vb

2.4.2.21 vec16s vsrl (vec16s va, vec16s vb)

Vector Shift Right (Logical).

Parameters:

va The vector

vb The shift amount vector

Returns:

va >> vb

2.4.2.22 vec16s vsro (vec16s va, vbool vc)

Vector Right Shift-in with OV (Logical).

Parameters:

```
va Input vectorvc Input OV vector being shifted in to MSB
```

Returns:

vector va right shift by 1 bit, with vc shifted into MSB

2.4.2.23 vec16s vsro (vec16s va, vbool vc, vbool *restrict vf)

Vector Right Shift-in with OV (Logical).

Parameters:

```
va Input vectorvc Input OV vector being shifted in to MSBvf Output vector of result LSB
```

Returns:

vector va right shift by 1 bit, with vc shifted into MSB

2.5 Comparison Intrinsics

Vector comparison intrinsics.

Functions

- vbool vseq (vec16s va, vec16s vb)

 Vector Set Comparison Equal.
- vbool vsequ (vec16u va, vec16u vb)

 Vector Set Comparison Equal.
- vbool vseq (vec16u va, vec16u vb)

 Vector Set Comparison Equal.
- vbool vseq (vec32s va, vec32s vb) Vector Set Comparison Equal.

- vbool vseq (vec32u va, vec32u vb)

 Vector Set Comparison Equal.
- vbool vsne (vec16s va, vec16s vb)

 Vector Set Comparison Not Equal.
- vbool vsneu (vec16u va, vec16u vb)

 Vector Set Comparison Not Equal.
- vbool vsne (vec16u va, vec16u vb)

 Vector Set Comparison Not Equal.
- vbool vsne (vec32s va, vec32s vb)

 Vector Set Comparison Not Equal.
- vbool vsne (vec32u va, vec32u vb)

 Vector Set Comparison Not Equal.
- vbool vsge (vec16s va, vec16s vb)

 Vector Set Greater than or Equal.
- vbool vsgeu (vec16u va, vec16u vb) Vector Set Greater than or Equal.
- vbool vsge (vec16u va, vec16u vb)

 Vector Set Greater than or Equal.
- vbool vsge (vec32s va, vec32s vb)

 Vector Set Greater than or Equal.
- vbool vsge (vec32u va, vec32u vb)

 Vector Set Greater than or Equal.
- vbool vsgt (vec16s va, vec16s vb) Vector Set Greater than.
- vbool vsgtu (vec16u va, vec16u vb) Vector Set Greater than.
- vbool vsgt (vec16u va, vec16u vb)

 Vector Set Greater than.
- vbool vsgt (vec32s va, vec32s vb) Vector Set Greater than.

- vbool vsgt (vec32u va, vec32u vb) Vector Set Greater than.
- vbool vsle (vec16s va, vec16s vb) Vector Set Less than or Equal.
- vbool vsleu (vec16u va, vec16u vb)

 Vector Set Less than or Equal.
- vbool vsle (vec16u va, vec16u vb) Vector Set Less than or Equal.
- vbool vsle (vec32s va, vec32s vb) Vector Set Less than or Equal.
- vbool vsle (vec32u va, vec32u vb) Vector Set Less than or Equal.
- vbool vslt (vec16s va, vec16s vb)

 Vector Set Less than.
- vbool vsltu (vec16u va, vec16u vb) Vector Set Less than.
- vbool vslt (vec16u va, vec16u vb)

 Vector Set Less than.
- vbool vslt (vec32s va, vec32s vb) Vector Set Less than.
- vbool vslt (vec32u va, vec32u vb) Vector Set Less than.

2.5.1 Detailed Description

Vector comparison intrinsics.

2.5.2 Function Documentation

2.5.2.1 vbool vseq (vec32u *va*, vec32u *vb*)

Vector Set Comparison Equal.

```
va The first vector
```

vb The second vector

Returns:

va == vb

2.5.2.2 vbool vseq (vec32s *va*, vec32s *vb*)

Vector Set Comparison Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va == vb

2.5.2.3 vbool vseq (vec16u va, vec16u vb)

Vector Set Comparison Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va == vb

2.5.2.4 vbool vseq (vec16s va, vec16s vb)

Vector Set Comparison Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va == vb

2.5.2.5 vbool vsequ (vec16u *va*, vec16u *vb*)

Vector Set Comparison Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va == vb

2.5.2.6 vbool vsge (vec32u *va*, vec32u *vb*)

Vector Set Greater than or Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va >= vb

2.5.2.7 vbool vsge (vec32s va, vec32s vb)

Vector Set Greater than or Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va >= vb

2.5.2.8 vbool vsge (vec16u *va*, vec16u *vb*)

Vector Set Greater than or Equal.

va The first vector

vb The second vector

Returns:

va >= vb

2.5.2.9 vbool vsge (vec16s *va*, vec16s *vb*)

Vector Set Greater than or Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va >= vb

2.5.2.10 vbool vsgeu (vec16u *va*, vec16u *vb*)

Vector Set Greater than or Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va >= vb

2.5.2.11 vbool vsgt (vec32u *va*, vec32u *vb*)

Vector Set Greater than.

Parameters:

va The first vector

vb The second vector

Returns:

va > vb

2.5.2.12 vbool vsgt (vec32s *va*, vec32s *vb*)

Vector Set Greater than.

Parameters:

va The first vector

vb The second vector

Returns:

va > vb

2.5.2.13 vbool vsgt (vec16u *va*, vec16u *vb*)

Vector Set Greater than.

Parameters:

va The first vector

vb The second vector

Returns:

va > vb

2.5.2.14 vbool vsgt (vec16s *va*, vec16s *vb*)

Vector Set Greater than.

Parameters:

va The first vector

vb The second vector

Returns:

va > vb

2.5.2.15 vbool vsgtu (vec16u *va*, vec16u *vb*)

Vector Set Greater than.

```
va The first vector
```

vb The second vector

Returns:

va > vb

2.5.2.16 vbool vsle (vec32u *va*, vec32u *vb*)

Vector Set Less than or Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va <= vb

2.5.2.17 vbool vsle (vec32s *va*, vec32s *vb*)

Vector Set Less than or Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va <= vb

2.5.2.18 vbool vsle (vec16u *va*, vec16u *vb*)

Vector Set Less than or Equal.

Parameters:

va The first vector

vb The second vector

Returns:

 $va \le vb$

2.5.2.19 vbool vsle (vec16s *va*, vec16s *vb*)

Vector Set Less than or Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va <= vb

2.5.2.20 vbool vsleu (vec16u *va*, vec16u *vb*)

Vector Set Less than or Equal.

Parameters:

va The first vector

vb The second vector

Returns:

 $va \le vb$

2.5.2.21 vbool vslt (vec32u *va*, vec32u *vb*)

Vector Set Less than.

Parameters:

va The first vector

vb The second vector

Returns:

va < vb

2.5.2.22 vbool vslt (vec32s *va*, vec32s *vb*)

Vector Set Less than.

```
va The first vector
```

vb The second vector

Returns:

va < vb

2.5.2.23 vbool vslt (vec16u *va*, vec16u *vb*)

Vector Set Less than.

Parameters:

va The first vector

vb The second vector

Returns:

va < vb

2.5.2.24 vbool vslt (vec16s *va*, vec16s *vb*)

Vector Set Less than.

Parameters:

va The first vector

vb The second vector

Returns:

va < vb

2.5.2.25 vbool vsltu (vec16u va, vec16u vb)

Vector Set Less than.

Parameters:

va The first vector

vb The second vector

Returns:

va < vb

2.5.2.26 vbool vsne (vec32u *va*, vec32u *vb*)

Vector Set Comparison Not Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va!= vb

2.5.2.27 vbool vsne (vec32s *va*, vec32s *vb*)

Vector Set Comparison Not Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va!= vb

2.5.2.28 vbool vsne (vec16u va, vec16u vb)

Vector Set Comparison Not Equal.

Parameters:

va The first vector

vb The second vector

Returns:

va!= vb

2.5.2.29 vbool vsne (vec16s *va*, vec16s *vb*)

Vector Set Comparison Not Equal.

```
va The first vectorvb The second vector
```

Returns:

va!= vb

2.5.2.30 vbool vsneu (vec16u *va*, vec16u *vb*)

Vector Set Comparison Not Equal.

Parameters:

```
va The first vectorvb The second vector
```

Returns:

va!= vb

2.6 Element Intrinsics

Vector element intrinsics.

Functions

- bool vget (vbool va, int i)

 Read vector element value at CU[i].
- int08s vget (vec08s v, int i)

 Read vector element value at CU[i].
- int08u vget (vec08u v, int i)

 Read vector element value at CU[i].
- int16s vget (vec16s v, int i)

 Read vector element value at CU[i].
- int16u vget (vec16u v, int i)

 Read vector element value at CU[i].
- int32s vget (vec32s va, int c)

 Read vector element value at CU[i].

- int32u vget (vec32u va, int c)

 Read vector element value at CU[i].
- vec08s vput (vec08s s0, int08s s1, int i)
 Write vector element from vector s1 in vector s0 at CU[i].
- vec08s vput (vec08s s0, int s1, int i)
 Write vector element from vector s1 in vector s0 at CU[i].
- vec08s vput (vec08s s0, vec08s s1, int i)
 Write vector element from vector s1 in vector s0 at CU[i].
- vec08s vput (vec08s s0, vec08s s1, vec08s i)
 Write vector element from vector s1 in vector s0 at CU[i].
- vec08u vput (vec08u s0, int08u s1, int i)

 Write vector element from vector s1 in vector s0 at CU[i].
- vec08u vput (vec08u s0, int s1, int i)

 Write vector element from vector s1 in vector s0 at CU[i].
- vec08u vput (vec08u s0, vec08u s1, int i)

 Write vector element from vector s1 in vector s0 at CU[i].
- vec08u vput (vec08u s0, vec08u s1, vec08u i)

 Write vector element from vector s1 in vector s0 at CU[i].
- vec16s vput (vec16s s0, int16s s1, int i)
 Write vector element from vector s1 in vector s0 at CU[i].
- vec16s vput (vec16s s0, int s1, int i)
 Write vector element from vector s1 in vector s0 at CU[i].
- vec16s vput (vec16s s0, vec16s s1, int i)
 Write vector element from vector s1 in vector s0 at CU[i].
- vec16s vput (vec16s s0, vec16s s1, vec16s i)

 Write vector element from vector s1 in vector s0 at CU[i].
- vec16u vput (vec16u s0, int16u s1, int i)
 Write vector element from vector s1 in vector s0 at CU[i].
- vec16u vput (vec16u s0, int s1, int i)

 Write vector element from vector s1 in vector s0 at CU[i].
- vec16u vput (vec16u s0, vec16u s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

- vec16u vput (vec16u s0, vec16u s1, vec16u i)

 Write vector element from vector s1 in vector s0 at CU[i].
- vec32s vput (vec32s s0, int32s s1, int i)

 Write vector element from vector s1 in vector s0 at CU[i].
- vec32s vput (vec32s s0, vec32s s1, int i)
 Write vector element from vector s1 in vector s0 at CU[i].
- vec32u vput (vec32u s0, int32u s1, int i)

 Write vector element from vector s1 in vector s0 at CU[i].
- vec32u vput (vec32u s0, int s1, int i)

 Write vector element from vector s1 in vector s0 at CU[i].
- vec32u vput (vec32u s0, vec32u s1, int i)
 Write vector element from vector s1 in vector s0 at CU[i].

2.6.1 Detailed Description

Vector element intrinsics.

2.6.2 Function Documentation

2.6.2.1 int32u vget (vec32u *va*, int *c*)

Read vector element value at CU[i].

Parameters:

va The vector

c The index

Returns:

va[c]

2.6.2.2 int32s vget (vec32s va, int c)

Read vector element value at CU[i].

va The vector

c The index

Returns:

va[c]

2.6.2.3 int16u vget (vec16u v, int i)

Read vector element value at CU[i].

Parameters:

v The vector

i The index

Returns:

va[i]

2.6.2.4 int16s vget (vec16s v, int i)

Read vector element value at CU[i].

Parameters:

v The vector

i The index

Returns:

va[i]

2.6.2.5 int08u vget (vec08u v, int i)

Read vector element value at CU[i].

Parameters:

v The vector

i The index

Returns:

va[i]

2.6.2.6 int08s vget (vec08s v, int i)

Read vector element value at CU[i].

Parameters:

- v The vector
- *i* The index

Returns:

va[i]

2.6.2.7 bool vget (vbool va, int i)

Read vector element value at CU[i].

Parameters:

va The vector

i The index

Returns:

va[i]

2.6.2.8 vec32u vput (vec32u $s\theta$, vec32u s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- i The index

Returns:

2.6.2.9 vec32u vput (vec32u $s\theta$, int s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

s0, with s1 at position i

2.6.2.10 vec32u vput (vec32u $s\theta$, int32u sI, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

s0, with s1 at position i

2.6.2.11 vec32s vput (vec32s s0, vec32s s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

2.6.2.12 vec32s vput (vec32s $s\theta$, int32s sI, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

s0, with s1 at position i

2.6.2.13 vec16u vput (vec16u $s\theta$, vec16u sI, vec16u i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

s0, with s1 at position i

2.6.2.14 vec16u vput (vec16u $s\theta$, vec16u s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

2.6.2.15 vec16u vput (vec16u $s\theta$, int s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

s0, with s1 at position i

2.6.2.16 vec16u vput (vec16u $s\theta$, int16u s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

s0, with s1 at position i

2.6.2.17 vec16s vput (vec16s s0, vec16s s1, vec16s i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

2.6.2.18 vec16s vput (vec16s s0, vec16s s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

s0 The base vector

s1 The new value

i The index

Returns:

s0, with s1 at position i

2.6.2.19 vec16s vput (vec16s s0, int s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

s0 The base vector

s1 The new value

i The index

Returns:

s0, with s1 at position i

2.6.2.20 vec16s vput (vec16s $s\theta$, int16s s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

s0 The base vector

s1 The new value

i The index

Returns:

2.6.2.21 vec08u vput (vec08u $s\theta$, vec08u sI, vec08u i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

s0, with s1 at position i

2.6.2.22 vec08u vput (vec08u $s\theta$, vec08u s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

s0, with s1 at position i

2.6.2.23 vec08u vput (vec08u $s\theta$, int s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

2.6.2.24 vec08u vput (vec08u s0, int08u s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

s0, with s1 at position i

2.6.2.25 vec08s vput (vec08s s0, vec08s s1, vec08s i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

s0, with s1 at position i

2.6.2.26 vec08s vput (vec08s s0, vec08s s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

- s0 The base vector
- s1 The new value
- *i* The index

Returns:

2.6.2.27 vec08s vput (vec08s $s\theta$, int s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

s0 The base vector

s1 The new value

i The index

Returns:

s0, with s1 at position i

2.6.2.28 vec08s vput (vec08s $s\theta$, int08s s1, int i)

Write vector element from vector s1 in vector s0 at CU[i].

Parameters:

s0 The base vector

s1 The new value

i The index

Returns:

s0, with s1 at position i

2.7 Memory Intrinsics

Vector memory intrinsics.

Functions

- vec08s vload (const vec08s *restrict ptr, vec16s vi)

 Load from vector address.
- vec08s vload (const vec08s *restrict ptr, vec08s vi)
 Load from vector address.
- vec08s vload (const vec08s *restrict ptr, vec08u vi)

 Load from vector address.
- vec08u vload (const vec08u *restrict ptr, vec16s vi)

Load from vector address.

- vec08u vload (const vec08u *restrict ptr, vec08s vi)
 Load from vector address.
- vec08u vload (const vec08u *restrict ptr, vec08u vi)

 Load from vector address.
- vec16s vload (const vec16s *restrict ptr, vec16s vi)
 Load from vector address.
- vec16s vload (const vec16s *restrict ptr, vec08s vi)
 Load from vector address.
- vec16s vload (const vec16s *restrict ptr, vec08u vi)

 Load from vector address.
- vec16u vload (const vec16u *restrict ptr, vec16s vi)

 Load from vector address.
- vec16u vload (const vec16u *restrict ptr, vec08s vi)

 Load from vector address.
- vec16u vload (const vec16u *restrict ptr, vec08u vi)

 Load from vector address.
- vec32s vload (const vec32s *restrict ptr, vec16s vi)

 Load from vector address.
- vec32s vload (const vec32s *restrict ptr, vec08s vi)

 Load from vector address.
- vec32s vload (const vec32s *restrict ptr, vec08u vi)

 Load from vector address.
- vec32u vload (const vec32u *restrict ptr, vec16s vi)

 Load from vector address.
- vec32u vload (const vec32u *restrict ptr, vec08s vi)
 Load from vector address.
- vec32u vload (const vec32u *restrict ptr, vec08u vi)

 Load from vector address.
- void vstore (vec08s *restrict ptr, vec16s vi, vec08s v)
 Store a value to a vector address.

- void vstore (vec08s *restrict ptr, vec08s vi, vec08s v)

 Store a value to a vector address.
- void vstore (vec08s *restrict ptr, vec08u vi, vec08s v)

 Store a value to a vector address.
- void vstore (vec08u *restrict ptr, vec16s vi, vec08u v)

 Store a value to a vector address.
- void vstore (vec08u *restrict ptr, vec08s vi, vec08u v)

 Store a value to a vector address.
- void vstore (vec08u *restrict ptr, vec08u vi, vec08u v)

 Store a value to a vector address.
- void vstore (vec16s *restrict ptr, vec16s vi, vec16s v)

 Store a value to a vector address.
- void vstore (vec16s *restrict ptr, vec08s vi, vec16s v)

 Store a value to a vector address.
- void vstore (vec16s *restrict ptr, vec08u vi, vec16s v)

 Store a value to a vector address.
- void vstore (vec16u *restrict ptr, vec16s vi, vec16u v)

 Store a value to a vector address.
- void vstore (vec16u *restrict ptr, vec08s vi, vec16u v)

 Store a value to a vector address.
- void vstore (vec16u *restrict ptr, vec08u vi, vec16u v)

 Store a value to a vector address.
- void vstore (vec32s *restrict ptr, vec16s vi, vec32s v)

 Store a value to a vector address.
- void vstore (vec32s *restrict ptr, vec08s vi, vec32s v)

 Store a value to a vector address.
- void vstore (vec32s *restrict ptr, vec08u vi, vec32s v)

 Store a value to a vector address.
- void vstore (vec32u *restrict ptr, vec16s vi, vec32u v)

 Store a value to a vector address.

- void vstore (vec32u *restrict ptr, vec08s vi, vec32u v) Store a value to a vector address.
- void vstore (vec32u *restrict ptr, vec08u vi, vec32u v) Store a value to a vector address.

2.7.1 Detailed Description

Vector memory intrinsics.

2.7.2 Function Documentation

2.7.2.1 vec32u vload (const vec32u *restrict ptr, vec08u vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.2 vec32u vload (const vec32u *restrict ptr, vec08s vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.3 vec32u vload (const vec32u *restrict ptr, vec16s vi)

Load from vector address.

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.4 vec32s vload (const vec32s *restrict ptr, vec08u vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.5 vec32s vload (const vec32s *restrict ptr, vec08s vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.6 vec32s vload (const vec32s *restrict ptr, vec16s vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.7 vec16u vload (const vec16u *restrict ptr, vec08u vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.8 vec16u vload (const vec16u *restrict ptr, vec08s vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.9 vec16u vload (const vec16u *restrict ptr, vec16s vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.10 vec16s vload (const vec16s *restrict ptr, vec08u vi)

Load from vector address.

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.11 vec16s vload (const vec16s *restrict ptr, vec08s vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.12 vec16s vload (const vec16s *restrict ptr, vec16s vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.13 vec08u vload (const vec08u *restrict ptr, vec08u vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.14 vec08u vload (const vec08u *restrict ptr, vec08s vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.15 vec08u vload (const vec08u *restrict ptr, vec16s vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.16 vec08s vload (const vec08s *restrict ptr, vec08u vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.17 vec08s vload (const vec08s *restrict ptr, vec08s vi)

Load from vector address.

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.18 vec08s vload (const vec08s *restrict ptr, vec16s vi)

Load from vector address.

Parameters:

```
ptr The base addressvi The vector index
```

Returns:

ptr[vi]

2.7.2.19 void vstore (vec32u *restrict ptr, vec08u vi, vec32u v)

Store a value to a vector address.

Parameters:

```
ptr The base addressvi The vector indexv The value that will be written
```

2.7.2.20 void vstore (vec32u *restrict ptr, vec08s vi, vec32u v)

Store a value to a vector address.

```
ptr The base addressvi The vector indexv The value that will be written
```

2.7.2.21 void vstore (vec32u *restrict ptr, vec16s vi, vec32u v)

Store a value to a vector address.

Parameters:

ptr The base address

vi The vector index

v The value that will be written

2.7.2.22 void vstore (vec32s *restrict ptr, vec08u vi, vec32s v)

Store a value to a vector address.

Parameters:

ptr The base address

vi The vector index

v The value that will be written

2.7.2.23 void vstore (vec32s *restrict ptr, vec08s vi, vec32s v)

Store a value to a vector address.

Parameters:

ptr The base address

vi The vector index

v The value that will be written

2.7.2.24 void vstore (vec32s *restrict ptr, vec16s vi, vec32s v)

Store a value to a vector address.

Parameters:

ptr The base address

vi The vector index

2.7.2.25 void vstore (vec16u *restrict ptr, vec08u vi, vec16u v)

Store a value to a vector address.

Parameters:

```
ptr The base address
```

vi The vector index

v The value that will be written

2.7.2.26 void vstore (vec16u *restrict ptr, vec08s vi, vec16u v)

Store a value to a vector address.

Parameters:

```
ptr The base address
```

vi The vector index

v The value that will be written

2.7.2.27 void vstore (vec16u *restrict ptr, vec16s vi, vec16u v)

Store a value to a vector address.

Parameters:

```
ptr The base address
```

vi The vector index

v The value that will be written

2.7.2.28 void vstore (vec16s *restrict ptr, vec08u vi, vec16s v)

Store a value to a vector address.

Parameters:

ptr The base address

vi The vector index

2.7.2.29 void vstore (vec16s *restrict ptr, vec08s vi, vec16s v)

Store a value to a vector address.

Parameters:

```
ptr The base address
```

vi The vector index

v The value that will be written

2.7.2.30 void vstore (vec16s *restrict ptr, vec16s vi, vec16s v)

Store a value to a vector address.

Parameters:

```
ptr The base address
```

vi The vector index

v The value that will be written

2.7.2.31 void vstore (vec08u *restrict ptr, vec08u vi, vec08u v)

Store a value to a vector address.

Parameters:

```
ptr The base address
```

vi The vector index

v The value that will be written

2.7.2.32 void vstore (vec08u *restrict ptr, vec08s vi, vec08u v)

Store a value to a vector address.

Parameters:

ptr The base address

vi The vector index

2.7.2.33 void vstore (vec08u *restrict ptr, vec16s vi, vec08u v)

Store a value to a vector address.

Parameters:

```
ptr The base address
```

vi The vector index

v The value that will be written

2.7.2.34 void vstore (vec08s *restrict ptr, vec08u vi, vec08s v)

Store a value to a vector address.

Parameters:

```
ptr The base address
```

vi The vector index

v The value that will be written

2.7.2.35 void vstore (vec08s *restrict ptr, vec08s vi, vec08s v)

Store a value to a vector address.

Parameters:

ptr The base address

vi The vector index

v The value that will be written

2.7.2.36 void vstore (vec08s *restrict ptr, vec16s vi, vec08s v)

Store a value to a vector address.

Parameters:

ptr The base address

vi The vector index

2.8 Specialized Shift Intrinsics

Vector specialized shift intrinsics.

Functions

void vsllx (vec16s *restrict vc, vec16s *restrict vd, vec16s va, vec16s vb, vec16s vs)

Vector Shift Left (Logical) of vec16 pairs.

void vsrax (vec16s *restrict vc, vec16s *restrict vd, vec16s va, vec16s vb, vec16s vs)

Vector Shift Right (Arithmetic) of vec16 pairs.

void vsrlx (vec16s *restrict vc, vec16s *restrict vd, vec16s va, vec16s vb, vec16s vs)

Vector Shift Right (Logical) of vec16 pairs.

void vsllx (vec16u *restrict vc, vec16u *restrict vd, vec16u va, vec16u vb, vec16u vs)

Vector Shift Left (Logical) of vec16 pairs.

• void vsrax (vec16u *restrict vc, vec16u *restrict vd, vec16u va, vec16u vb, vec16u vs)

Vector Shift Right (Arithmetic) of vec16 pairs.

void vsrlx (vec16u *restrict vc, vec16u *restrict vd, vec16u va, vec16u vb, vec16u vs)

Vector Shift Right (Logical) of vec16 pairs.

• void vsr (vec16u *restrict vc, vec16u *restrict vd, vec16u va, vec16u vb, int s) Vector Shift Right (Arithmetic) of vec16 pairs.

2.8.1 Detailed Description

Vector specialized shift intrinsics.

2.8.2 Function Documentation

2.8.2.1 void vsllx (vec16u *restrict vc, vec16u *restrict vd, vec16u va, vec16u vb, vec16u vs)

Vector Shift Left (Logical) of vec16 pairs.

- vc [out] Pointer to the lo part
- vd [out] Pointer to the hi part
- va The lo part
- *vb* The hi part
- vs The shift amount

2.8.2.2 void vsllx (vec16s *restrict vc, vec16s *restrict vd, vec16s va, vec16s vb, vec16s vs)

Vector Shift Left (Logical) of vec16 pairs.

Parameters:

- vc [out] Pointer to the lo part
- vd [out] Pointer to the hi part
- va The lo part
- vb The hi part
- vs The shift amount

2.8.2.3 void vsr (vec16u *restrict vc, vec16u *restrict vd, vec16u va, vec16u vb, int s)

Vector Shift Right (Arithmetic) of vec16 pairs.

Parameters:

- vc [out] Pointer to the lo part
- vd [out] Pointer to the hi part
- va The lo part
- vb The hi part
- s The shift amount

2.8.2.4 void vsrax (vec16u *restrict vc, vec16u *restrict vd, vec16u va, vec16u vb, vec16u vs)

Vector Shift Right (Arithmetic) of vec16 pairs.

- vc [out] Pointer to the lo part
- vd [out] Pointer to the hi part
- va The lo part
- *vb* The hi part
- vs The shift amount

2.8.2.5 void vsrax (vec16s *restrict vc, vec16s *restrict vd, vec16s va, vec16s vb, vec16s vs)

Vector Shift Right (Arithmetic) of vec16 pairs.

Parameters:

- vc [out] Pointer to the lo part
- vd [out] Pointer to the hi part
- va The lo part
- vb The hi part
- vs The shift amount

2.8.2.6 void vsrlx (vec16u *restrict vc, vec16u *restrict vd, vec16u va, vec16u vb, vec16u vs)

Vector Shift Right (Logical) of vec16 pairs.

Parameters:

- vc [out] Pointer to the lo part
- vd [out] Pointer to the hi part
- va The lo part
- vb The hi part
- vs The shift amount

2.8.2.7 void vsrlx (vec16s *restrict vc, vec16s *restrict vd, vec16s va, vec16s vb, vec16s vs)

Vector Shift Right (Logical) of vec16 pairs.

- vc [out] Pointer to the lo part
- vd [out] Pointer to the hi part
- va The lo part
- vb The hi part
- vs The shift amount

2.9 Specialized Multiplication Intrinsics

Vector specialized multiplication intrinsics.

Functions

- vec16u vmul_ulul (vec16s a, vec16s b)
 Vector multiplication unsigned-low unsigned-low.
- vec16u vmul_ulul (vec16u a, vec16u b)
 Vector multiplication unsigned-low unsigned-low.
- vec16u vmul_uluh (vec16s a, vec16s b)
 Vector multiplication unsigned-low unsigned-high.
- vec16u vmul_uluh (vec16u a, vec16u b)

 Vector multiplication unsigned-low unsigned-high.
- vec16s vmul_ulsh (vec16s a, vec16s b)
 Vector multiplication unsigned-low signed-high.
- vec16u vmul_ulsh (vec16u a, vec16u b)
 Vector multiplication unsigned-low signed-high.
- vec16u vmul_uhul (vec16s a, vec16s b)

 Vector multiplication unsigned-high unsigned-low.
- vec16u vmul_uhul (vec16u a, vec16u b)
 Vector multiplication unsigned-high unsigned-low.
- vec16u vmul_uhuh (vec16s a, vec16s b)

 Vector multiplication unsigned-high unsigned-high.
- vec16u vmul_uhuh (vec16u a, vec16u b)
 Vector multiplication unsigned-high unsigned-high.
- vec16s vmul_uhsh (vec16s a, vec16s b)

Vector multiplication unsigned-high signed-high.

- vec16u vmul_uhsh (vec16u a, vec16u b)
 Vector multiplication unsigned-high signed-high.
- vec16s vmul_shul (vec16s a, vec16s b)
 Vector multiplication signed-high unsigned-low.
- vec16u vmul_shul (vec16u a, vec16u b)

 Vector multiplication signed-high unsigned-low.
- vec16s vmul_shuh (vec16s a, vec16s b)
 Vector multiplication signed-high unsigned-high.
- vec16u vmul_shuh (vec16u a, vec16u b)
 Vector multiplication signed-high unsigned-high.
- vec16s vmul_shsh (vec16s a, vec16s b)
 Vector multiplication signed-high signed-high.
- vec16u vmul_shsh (vec16u a, vec16u b)
 Vector multiplication signed-high signed-high.
- vec16s vmul_slul (vec16s a, vec16s b)

 Vector multiplication signed-low unsigned-low.
- vec16u vmul_slul (vec16u a, vec16u b)

 Vector multiplication signed-low unsigned-low.
- vec16s vmul_ulsl (vec16s a, vec16s b)
 Vector multiplication unsigned-low signed-low.
- vec16u vmul_ulsl (vec16u a, vec16u b)

 Vector multiplication unsigned-low signed-low.
- vec16s vmul_uhsl (vec16s a, vec16s b)

 Vector multiplication unsigned-high signed-low.
- vec16u vmul_uhsl (vec16u a, vec16u b)

 Vector multiplication unsigned-high signed-low.
- vec16s vmul_slsl (vec16s a, vec16s b)

 Vector multiplication signed-low signed-low.
- vec16u vmul_slsl (vec16u a, vec16u b)
 Vector multiplication signed-low signed-low.

- vec16s vmul_shsl (vec16s a, vec16s b)
 Vector multiplication signed-high signed-low.
- vec16u vmul_shsl (vec16u a, vec16u b)
 Vector multiplication signed-high signed-low.
- vec16s vmul_sluh (vec16s a, vec16s b)
 Vector multiplication signed-low unsigned-high.
- vec16u vmul_sluh (vec16u a, vec16u b)

 Vector multiplication signed-low unsigned-high.
- vec16s vmul_slsh (vec16s a, vec16s b)
 Vector multiplication signed-low signed-high.
- vec16u vmul_slsh (vec16u a, vec16u b)
 Vector multiplication signed-low signed-high.
- void vmul (vec16s *restrict h, vec16u *restrict l, vec16s a, vec16s b) Vector 16 x 16 -> 32-bit multiplication.
- void vmul (vec16s *restrict h, vec16u *restrict l, vec16u a, vec16s b) Vector 16 x 16 -> 32-bit multiplication.
- void vmul (vec16s *restrict h, vec16u *restrict l, vec16s a, vec16u b) Vector 16 x 16 -> 32-bit multiplication.
- void vmul (vec16u *restrict h, vec16u *restrict l, vec16u a, vec16u b) Vector 16 x 16 -> 32-bit multiplication.
- void vmac (vec16s *restrict h, vec16u *restrict l, vec16s a, vec16s b) Vector 16 x 16 -> 32-bit mult-add.
- void vmac (vec16s *restrict h, vec16u *restrict l, vec16u a, vec16s b) Vector 16 x 16 -> 32-bit mult-add.
- void vmac (vec16s *restrict h, vec16u *restrict l, vec16s a, vec16u b) Vector 16 x 16 -> 32-bit mult-add.
- void vmac (vec16u *restrict h, vec16u *restrict l, vec16u a, vec16u b)

 Vector 16 x 16 -> 32-bit mult-add.
- void vmad (vec16s *restrict h, vec16u *restrict l, vec16s a, vec16s b, vec16s ch, vec16u cl)

Vector $16 \times 16 \rightarrow 32$ -bit mult-add with src.

void vmad (vec16s *restrict h, vec16u *restrict l, vec16u a, vec16s b, vec16s ch, vec16u cl)

Vector 16 x 16 -> 32-bit mult-add with src.

void vmad (vec16s *restrict h, vec16u *restrict l, vec16s a, vec16u b, vec16s ch, vec16u cl)

Vector 16 x 16 -> 32-bit mult-add with src.

void vmad (vec16u *restrict h, vec16u *restrict l, vec16u a, vec16u b, vec16u ch, vec16u cl)

Vector 16 x 16 -> 32-bit mult-add with src.

2.9.1 Detailed Description

Vector specialized multiplication intrinsics.

2.9.2 Function Documentation

2.9.2.1 void vmac (vec16u *restrict h, vec16u *restrict l, vec16u a, vec16u b)

Vector $16 \times 16 \longrightarrow 32$ -bit mult-add.

Parameters:

- h [out] Pointer to output vector for high 16-bit result
- *l* [out] Pointer to output vector for low 16-bit result
- a First input vector
- b Second input vector

2.9.2.2 void vmac (vec16s *restrict h, vec16u *restrict l, vec16s a, vec16u b)

Vector $16 \times 16 \rightarrow 32$ -bit mult-add.

- **h** [out] Pointer to output vector for high 16-bit result
- *l* [out] Pointer to output vector for low 16-bit result
- a First input vector
- b Second input vector

2.9.2.3 void vmac (vec16s *restrict h, vec16u *restrict l, vec16u a, vec16s b)

Vector $16 \times 16 \longrightarrow 32$ -bit mult-add.

Parameters:

- h [out] Pointer to output vector for high 16-bit result
- *l* [out] Pointer to output vector for low 16-bit result
- a First input vector
- b Second input vector

2.9.2.4 void vmac (vec16s *restrict h, vec16u *restrict l, vec16s a, vec16s b)

Vector $16 \times 16 \longrightarrow 32$ -bit mult-add.

Parameters:

- **h** [out] Pointer to output vector for high 16-bit result
- *l* [out] Pointer to output vector for low 16-bit result
- a First input vector
- b Second input vector

2.9.2.5 void vmad (vec16u *restrict h, vec16u *restrict l, vec16u a, vec16u b, vec16u ch, vec16u cl)

Vector $16 \times 16 \longrightarrow 32$ -bit mult-add with src.

- h [out] Pointer to output vector for high 16-bit result
- *l* [out] Pointer to output vector for low 16-bit result
- a First input vector
- b Second input vector
- ch High 16-bit of addend input vector
- cl Low 16-bit of addend input vector

2.9.2.6 void vmad (vec16s *restrict h, vec16u *restrict l, vec16s a, vec16u b, vec16s ch, vec16u cl)

Vector $16 \times 16 \rightarrow 32$ -bit mult-add with src.

Parameters:

- h [out] Pointer to output vector for high 16-bit result
- *l* [out] Pointer to output vector for low 16-bit result
- a First input vector
- **b** Second input vector
- ch High 16-bit of addend input vector
- cl Low 16-bit of addend input vector

2.9.2.7 void vmad (vec16s *restrict h, vec16u *restrict l, vec16u a, vec16s b, vec16s ch, vec16u cl)

Vector $16 \times 16 \rightarrow 32$ -bit mult-add with src.

Parameters:

- **h** [out] Pointer to output vector for high 16-bit result
- *l* [out] Pointer to output vector for low 16-bit result
- a First input vector
- b Second input vector
- ch High 16-bit of addend input vector
- cl Low 16-bit of addend input vector

2.9.2.8 void vmad (vec16s *restrict h, vec16u *restrict l, vec16s a, vec16s b, vec16s ch, vec16u cl)

Vector 16 x 16 -> 32-bit mult-add with src.

- h [out] Pointer to output vector for high 16-bit result
- *l* [out] Pointer to output vector for low 16-bit result
- a First input vector
- b Second input vector
- ch High 16-bit of addend input vector
- cl Low 16-bit of addend input vector

2.9.2.9 void vmul (vec16u *restrict h, vec16u *restrict l, vec16u a, vec16u b)

Vector $16 \times 16 \longrightarrow 32$ -bit multiplication.

Parameters:

- h [out] Pointer to output vector for high 16-bit result
- *l* [out] Pointer to output vector for low 16-bit result
- a First input vector
- b Second input vector

2.9.2.10 void vmul (vec16s *restrict h, vec16u *restrict l, vec16s a, vec16u b)

Vector 16 x 16 -> 32-bit multiplication.

Parameters:

- h [out] Pointer to output vector for high 16-bit result
- *l* [out] Pointer to output vector for low 16-bit result
- a First input vector
- b Second input vector

2.9.2.11 void vmul (vec16s *restrict h, vec16u *restrict l, vec16u a, vec16s b)

Vector $16 \times 16 \longrightarrow 32$ -bit multiplication.

Parameters:

- **h** [out] Pointer to output vector for high 16-bit result
- *l* [out] Pointer to output vector for low 16-bit result
- a First input vector
- b Second input vector

2.9.2.12 void vmul (vec16s *restrict h, vec16u *restrict l, vec16s a, vec16s b)

Vector 16 x 16 -> 32-bit multiplication.

Parameters:

h [out] Pointer to output vector for high 16-bit result

- *l* [out] Pointer to output vector for low 16-bit result
- a First input vector
- b Second input vector

2.9.2.13 vec16u vmul_shsh (vec16u *a*, vec16u *b*)

Vector multiplication signed-high signed-high.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(signed-high) * vb(signed-high)
```

2.9.2.14 vec16s vmul_shsh (vec16s *a*, vec16s *b*)

Vector multiplication signed-high signed-high.

Parameters:

- a The first vector
- b The second vector

Returns:

```
va(signed-high) * vb(signed-high)
```

2.9.2.15 vec16u vmul_shsl (vec16u a, vec16u b)

Vector multiplication signed-high signed-low.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(signed-high) * vb(signed-low)
```

2.9.2.16 vec16s vmul_shsl (vec16s *a*, vec16s *b*)

Vector multiplication signed-high signed-low.

Parameters:

- a The first vector
- b The second vector

Returns:

```
va(signed-high) * vb(signed-low)
```

2.9.2.17 vec16u vmul_shuh (vec16u *a*, vec16u *b*)

Vector multiplication signed-high unsigned-high.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(signed-high) * vb(unsigned-high)
```

2.9.2.18 vec16s vmul_shuh (vec16s a, vec16s b)

Vector multiplication signed-high unsigned-high.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(signed-high) * vb(unsigned-high)
```

2.9.2.19 vec16u vmul_shul (vec16u a, vec16u b)

Vector multiplication signed-high unsigned-low.

- a The first vector
- **b** The second vector

Returns:

```
va(signed-high) * vb(unsigned-low)
```

2.9.2.20 vec16s vmul_shul (vec16s a, vec16s b)

Vector multiplication signed-high unsigned-low.

Parameters:

- a The first vector
- b The second vector

Returns:

```
va(signed-high) * vb(unsigned-low)
```

2.9.2.21 vec16u vmul_slsh (vec16u a, vec16u b)

Vector multiplication signed-low signed-high.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(signed-low) * vb(signed-high)
```

2.9.2.22 vec16s vmul_slsh (vec16s a, vec16s b)

Vector multiplication signed-low signed-high.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(signed-low) * vb(signed-high)
```

2.9.2.23 vec16u vmul_slsl (vec16u a, vec16u b)

Vector multiplication signed-low signed-low.

Parameters:

- a The first vector
- b The second vector

Returns:

```
va(signed-lo) * vb(signed-lo)
```

2.9.2.24 vec16s vmul_slsl (vec16s a, vec16s b)

Vector multiplication signed-low signed-low.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(signed-lo) * vb(signed-lo)
```

2.9.2.25 vec16u vmul_sluh (vec16u a, vec16u b)

Vector multiplication signed-low unsigned-high.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(signed-low) * vb(unsigned-high)
```

2.9.2.26 vec16s vmul_sluh (vec16s a, vec16s b)

Vector multiplication signed-low unsigned-high.

- a The first vector
- **b** The second vector

Returns:

```
va(signed-low) * vb(unsigned-high)
```

2.9.2.27 vec16u vmul_slul (vec16u *a*, vec16u *b*)

Vector multiplication signed-low unsigned-low.

Parameters:

- a The first vector
- b The second vector

Returns:

```
va(signed-low) * vb(unsigned-low)
```

2.9.2.28 vec16s vmul_slul (vec16s a, vec16s b)

Vector multiplication signed-low unsigned-low.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(signed-low) * vb(unsigned-low)
```

2.9.2.29 vec16u vmul_uhsh (vec16u a, vec16u b)

Vector multiplication unsigned-high signed-high.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(unsigned-high) * vb(signed-high)
```

2.9.2.30 vec16s vmul_uhsh (vec16s a, vec16s b)

Vector multiplication unsigned-high signed-high.

Parameters:

- a The first vector
- b The second vector

Returns:

```
va(unsigned-high) * vb(signed-high)
```

2.9.2.31 vec16u vmul_uhsl (vec16u *a*, vec16u *b*)

Vector multiplication unsigned-high signed-low.

Parameters:

- a The first vector
- b The second vector

Returns:

```
va(unsigned-high) * vb(signed-low)
```

2.9.2.32 vec16s vmul_uhsl (vec16s a, vec16s b)

Vector multiplication unsigned-high signed-low.

Parameters:

- a The first vector
- b The second vector

Returns:

```
va(unsigned-high) * vb(signed-low)
```

2.9.2.33 vec16u vmul_uhuh (vec16u *a*, vec16u *b*)

Vector multiplication unsigned-high unsigned-high.

- a The first vector
- **b** The second vector

Returns:

```
va(unsigned-high) * vb(unsigned-high)
```

2.9.2.34 vec16u vmul_uhuh (vec16s *a*, vec16s *b*)

Vector multiplication unsigned-high unsigned-high.

Parameters:

- a The first vector
- b The second vector

Returns:

```
va(unsigned-high) * vb(unsigned-high)
```

2.9.2.35 vec16u vmul_uhul (vec16u *a*, vec16u *b*)

Vector multiplication unsigned-high unsigned-low.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(unsigned-high) * vb(unsigned-low)
```

2.9.2.36 vec16u vmul_uhul (vec16s a, vec16s b)

Vector multiplication unsigned-high unsigned-low.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(unsigned-high) * vb(unsigned-low)
```

2.9.2.37 vec16u vmul_ulsh (vec16u *a*, vec16u *b*)

Vector multiplication unsigned-low signed-high.

Parameters:

- a The first vector
- b The second vector

Returns:

```
va(unsigned-low) * vb(signed-high)
```

2.9.2.38 vec16s vmul_ulsh (vec16s a, vec16s b)

Vector multiplication unsigned-low signed-high.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(unsigned-low) * vb(signed-high)
```

2.9.2.39 vec16u vmul_ulsl (vec16u a, vec16u b)

Vector multiplication unsigned-low signed-low.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(unsigned-low) * vb(signed-low)
```

2.9.2.40 vec16s vmul_ulsl (vec16s a, vec16s b)

Vector multiplication unsigned-low signed-low.

- a The first vector
- **b** The second vector

Returns:

```
va(unsigned-low) * vb(signed-low)
```

2.9.2.41 vec16u vmul_uluh (vec16u *a*, vec16u *b*)

Vector multiplication unsigned-low unsigned-high.

Parameters:

- a The first vector
- b The second vector

Returns:

```
va(unsigned-low) * vb(unsigned-high)
```

2.9.2.42 vec16u vmul_uluh (vec16s *a*, vec16s *b*)

Vector multiplication unsigned-low unsigned-high.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(unsigned-low) * vb(unsigned-high)
```

2.9.2.43 vec16u vmul_ulul (vec16u a, vec16u b)

Vector multiplication unsigned-low unsigned-low.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(unsigned-low) * vb(unsigned-low)
```

2.9.2.44 vec16u vmul_ulul (vec16s *a*, vec16s *b*)

Vector multiplication unsigned-low unsigned-low.

Parameters:

- a The first vector
- **b** The second vector

Returns:

```
va(unsigned-low) * vb(unsigned-low)
```

2.10 Swap Intrinsics

Vector swap intrinsics.

Functions

- vec16s vselect (vec16s a, vec16s b, vbool c)

 Vector select.
- vec16u vselect (vec16u a, vec16u b, vbool c) Vector select.
- vec16s vselect (vec08s a, vec08s b, vbool c)

 Vector select.
- vec16u vselect (vec08u a, vec08u b, vbool c) Vector select.
- vec32s vselect (vec32s a, vec32s b, vbool c) Vector select.
- vec32u vselect (vec32u a, vec32u b, vbool c) Vector select.
- void vswap (vec16s *restrict a, vec16s *restrict b, vbool c) Vector conditional swap.
- void vswap (vec16u *restrict a, vec16u *restrict b, vbool c) Vector conditional swap.
- void vswap (vec32s *restrict a, vec32s *restrict b, vbool c) Vector conditional swap.

• void vswap (vec32u *restrict a, vec32u *restrict b, vbool c) Vector conditional swap.

2.10.1 Detailed Description

Vector swap intrinsics.

2.10.2 Function Documentation

2.10.2.1 vec32u vselect (vec32u a, vec32u b, vbool c)

Vector select.

Parameters:

- a Input first vector
- **b** Input second vector
- c Input condition vector

Returns:

vc? va: vb

2.10.2.2 vec32s vselect (vec32s a, vec32s b, vbool c)

Vector select.

Parameters:

- a Input first vector
- **b** Input second vector
- c Input condition vector

Returns:

vc? va: vb

2.10.2.3 vec16u vselect (vec08u a, vec08u b, vbool c)

Vector select.

- a Input first vector
- **b** Input second vector
- c Input condition vector

Returns:

vc? va: vb

2.10.2.4 vec16s vselect (vec08s a, vec08s b, vbool c)

Vector select.

Parameters:

- a Input first vector
- **b** Input second vector
- c Input condition vector

Returns:

vc? va: vb

2.10.2.5 vec16u vselect (vec16u a, vec16u b, vbool c)

Vector select.

Parameters:

- a Input first vector
- \boldsymbol{b} Input second vector
- c Input condition vector

Returns:

vc? va: vb

2.10.2.6 vec16s vselect (vec16s a, vec16s b, vbool c)

Vector select.

- a Input first vector
- **b** Input second vector
- c Input condition vector

Returns:

vc? va: vb

2.10.2.7 void vswap (vec32u *restrict a, vec32u *restrict b, vbool c)

Vector conditional swap.

Parameters:

- a The first input vector
- **b** The second input vector
- c Input condition vector

Returns:

if(vc) swap(va,vb)

2.10.2.8 void vswap (vec32s *restrict a, vec32s *restrict b, vbool c)

Vector conditional swap.

Parameters:

- a The first input vector
- **b** The second input vector
- c Input condition vector

Returns:

if(vc) swap(va,vb)

2.10.2.9 void vswap (vec16u *restrict a, vec16u *restrict b, vbool c)

Vector conditional swap.

- a The first input vector
- **b** The second input vector
- c Input condition vector

Returns:

if(vc) swap(va,vb)

2.10.2.10 void vswap (vec16s *restrict a, vec16s *restrict b, vbool c)

Vector conditional swap.

Parameters:

- a The first input vector
- **b** The second input vector
- c Input condition vector

Returns:

if(vc) swap(va,vb)

2.11 Move/Rotate Intrinsics

Vector move/rotate intrinsics.

Functions

- vec16s vml (vec16s a, vec16s b)

 Vector element shift left.
- vec16u vml (vec16u a, vec16u b)

 Vector element shift left.
- vec08s vml (vec08s a, vec08s b)

 Vector element shift left.
- vec08u vml (vec08u a, vec08u b)

 Vector element shift left.
- vec32s vml (vec32s a, vec32s b)

 Vector element shift left.

- vec32u vml (vec32u a, vec32u b)

 Vector element shift left.
- vec16s vmsl (vec16s a, vec16s b)

 Vector element shift left.
- vec16u vmsl (vec16u a, vec16u b)

 Vector element shift left.
- vec08s vmsl (vec08s a, vec08s b)

 Vector element shift left.
- vec08u vmsl (vec08u a, vec08u b)

 Vector element shift left.
- vec32s vmsl (vec32s a, vec32s b)

 Vector element shift left.
- vec32u vmsl (vec32u a, vec32u b) Vector element shift left.
- vec16s vmsl (vec16s a) Vector element shift left.
- vec16u vmsl (vec16u a)

 Vector element shift left.
- vec08s vmsl (vec08s a)
 Vector element shift left.
- vec08u vmsl (vec08u a)

 Vector element shift left.
- vec32s vmsl (vec32s a) Vector element shift left.
- vec32u vmsl (vec32u a)

 Vector element shift left.
- vec16s vmrl (vec16s a, vec16s b)

 Vector element rotate left.
- vec16u vmrl (vec16u a, vec16u b)

 Vector element rotate left.
- vec08s vmrl (vec08s a, vec08s b)

Vector element rotate left.

- vec08u vmrl (vec08u a, vec08u b)

 Vector element rotate left.
- vec32s vmrl (vec32s a, vec32s b)

 Vector element rotate left.
- vec32u vmrl (vec32u a, vec32u b) Vector element rotate left.
- vec16s vmrl (vec16s a) Vector element rotate left.
- vec16u vmrl (vec16u a)

 Vector element rotate left.
- vec08s vmrl (vec08s a)
 Vector element rotate left.
- vec08u vmrl (vec08u a)

 Vector element rotate left.
- vec32s vmrl (vec32s a) Vector element rotate left.
- vec32u vmrl (vec32u a)

 Vector element rotate left.
- vec16s vmr (vec16s a, vec16s b)

 Vector element shift right.
- vec16u vmr (vec16u a, vec16u b)

 Vector element shift right.
- vec08s vmr (vec08s a, vec08s b)

 Vector element shift right.
- vec08u vmr (vec08u a, vec08u b)

 Vector element shift right.
- vec32s vmr (vec32s a, vec32s b)

 Vector element shift right.
- vec32u vmr (vec32u a, vec32u b)

 Vector element shift right.

- vec16s vmsr (vec16s a, vec16s b)

 Vector element shift right.
- vec16u vmsr (vec16u a, vec16u b)

 Vector element shift right.
- vec08s vmsr (vec08s a, vec08s b)

 Vector element shift right.
- vec08u vmsr (vec08u a, vec08u b)

 Vector element shift right.
- vec32s vmsr (vec32s a, vec32s b)

 Vector element shift right.
- vec32u vmsr (vec32u a, vec32u b)

 Vector element shift right.
- vec16s vmsr (vec16s a)

 Vector element shift right.
- vec16u vmsr (vec16u a)

 Vector element shift right.
- vec08s vmsr (vec08s a)

 Vector element shift right.
- vec08u vmsr (vec08u a)

 Vector element shift right.
- vec32s vmsr (vec32s a)

 Vector element shift right.
- vec32u vmsr (vec32u a)

 Vector element shift right.
- vec16s vmrr (vec16s a, vec16s b)

 Vector element rotate right.
- vec16u vmrr (vec16u a, vec16u b)

 Vector element rotate right.
- vec08s vmrr (vec08s a, vec08s b)

 Vector element rotate right.

- vec08u vmrr (vec08u a, vec08u b)

 Vector element rotate right.
- vec32s vmrr (vec32s a, vec32s b)

 Vector element rotate right.
- vec32u vmrr (vec32u a, vec32u b)

 Vector element rotate right.
- vec16s vmrr (vec16s a)

 Vector element rotate right.
- vec16u vmrr (vec16u a)

 Vector element rotate right.
- vec08s vmrr (vec08s a)

 Vector element rotate right.
- vec08u vmrr (vec08u a)

 Vector element rotate right.
- vec32s vmrr (vec32s a)

 Vector element rotate right.
- vec32u vmrr (vec32u a)
 Vector element rotate right.
- vec16u vextract_hi (int i) Vector splate hi.
- vec16u vextract_lo (int i) Vector splate lo.

2.11.1 Detailed Description

Vector move/rotate intrinsics.

2.11.2 Function Documentation

2.11.2.1 vec16u vextract_hi (int *i*)

Vector splate hi.

i Input integer

Returns:

vector of high 16-bit of i

2.11.2.2 vec16u vextract_lo (**int** *i*)

Vector splate lo.

Parameters:

i Input integer

Returns:

vector of low 16-bit of i

2.11.2.3 vec32u vml (vec32u a, vec32u b)

Vector element shift left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.4 vec32s vml (vec32s a, vec32s b)

Vector element shift left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.5 vec08u vml (vec08u a, vec08u b)

Vector element shift left.

Parameters:

- a Output vector and first input vector
- b The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.6 vec08s vml (vec08s a, vec08s b)

Vector element shift left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.7 vec16u vml (vec16u a, vec16u b)

Vector element shift left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.8 vec16s vml (vec16s a, vec16s b)

Vector element shift left.

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.9 vec32u vmr (vec32u a, vec32u b)

Vector element shift right.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.10 vec32s vmr (vec32s a, vec32s b)

Vector element shift right.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.11 vec
08u vmr (vec
08u a, vec
08u b)

Vector element shift right.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.12 vec08s vmr (vec08s a, vec08s b)

Vector element shift right.

Parameters:

- a Output vector and first input vector
- b The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.13 vec16u vmr (vec16u a, vec16u b)

Vector element shift right.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.14 vec16s vmr (vec16s a, vec16s b)

Vector element shift right.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.15 vec32u vmrl (vec32u a)

Vector element rotate left.

a Output vector and first input vector

Returns:

va shift left by one CU; va[31]=va[0]

2.11.2.16 vec32s vmrl (vec32s a)

Vector element rotate left.

Parameters:

a Output vector and first input vector

Returns:

va shift left by one CU; va[31]=va[0]

2.11.2.17 vec08u vmrl (vec08u a)

Vector element rotate left.

Parameters:

a Output vector and first input vector

Returns:

va shift left by one CU; va[31]=va[0]

2.11.2.18 vec08s vmrl (vec08s a)

Vector element rotate left.

Parameters:

a Output vector and first input vector

Returns:

va shift left by one CU; va[31]=va[0]

2.11.2.19 vec16u vmrl (vec16u a)

Vector element rotate left.

Parameters:

a Output vector and first input vector

Returns:

va shift left by one CU; va[31]=va[0]

2.11.2.20 vec16s vmrl (vec16s a)

Vector element rotate left.

Parameters:

a Output vector and first input vector

Returns:

va shift left by one CU; va[31]=va[0]

2.11.2.21 vec32u vmrl (vec32u a, vec32u b)

Vector element rotate left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.22 vec32s vmrl (vec32s a, vec32s b)

Vector element rotate left.

Parameters:

a Output vector and first input vector

b The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.23 vec08u vmrl (vec08u *a*, vec08u *b*)

Vector element rotate left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.24 vec08s vmrl (vec08s a, vec08s b)

Vector element rotate left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.25 vec16u vmrl (vec16u a, vec16u b)

Vector element rotate left.

Parameters:

- a Output vector and first input vector
- b The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.26 vec16s vmrl (vec16s a, vec16s b)

Vector element rotate left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.27 vec32u vmrr (vec32u a)

Vector element rotate right.

Parameters:

a Output vector and first input vector

Returns:

va shift right by one CU; va[0]=va[31]

2.11.2.28 vec32s vmrr (vec32s a)

Vector element rotate right.

Parameters:

a Output vector and first input vector

Returns:

va shift right by one CU; va[0]=va[31]

2.11.2.29 vec08u vmrr (vec08u a)

Vector element rotate right.

Parameters:

a Output vector and first input vector

Returns:

va shift right by one CU; va[0]=va[31]

2.11.2.30 vec08s vmrr (vec08s a)

Vector element rotate right.

Parameters:

a Output vector and first input vector

Returns:

va shift right by one CU; va[0]=va[31]

2.11.2.31 vec16u vmrr (vec16u a)

Vector element rotate right.

Parameters:

a Output vector and first input vector

Returns:

va shift right by one CU; va[0]=va[31]

2.11.2.32 vec16s vmrr (vec16s a)

Vector element rotate right.

Parameters:

a Output vector and first input vector

Returns:

va shift right by one CU; va[0]=va[31]

2.11.2.33 vec32u vmrr (vec32u a, vec32u b)

Vector element rotate right.

Parameters:

- a Output vector and first input vector
- b The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.34 vec32s vmrr (vec32s a, vec32s b)

Vector element rotate right.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.35 vec
08u vmrr (vec
08u $\boldsymbol{a}, \text{ vec}$ 08u $\boldsymbol{b})$

Vector element rotate right.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.36 vec08s vmrr (vec08s a, vec08s b)

Vector element rotate right.

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.37 vec16u vmrr (vec16u a, vec16u b)

Vector element rotate right.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.38 vec16s vmrr (vec16s a, vec16s b)

Vector element rotate right.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.39 vec32u vmsl (vec32u a)

Vector element shift left.

Parameters:

a Output vector and first input vector

Returns:

va shift left by one CU; va[31]=0

2.11.2.40 vec32s vmsl (vec32s a)

Vector element shift left.

Parameters:

a Output vector and first input vector

Returns:

va shift left by one CU; va[31]=0

2.11.2.41 vec08u vmsl (vec08u a)

Vector element shift left.

Parameters:

a Output vector and first input vector

Returns:

va shift left by one CU; va[31]=0

2.11.2.42 vec08s vmsl (vec08s *a*)

Vector element shift left.

Parameters:

a Output vector and first input vector

Returns:

va shift left by one CU; va[31]=0

2.11.2.43 vec16u vmsl (vec16u a)

Vector element shift left.

Parameters:

a Output vector and first input vector

Returns:

va shift left by one CU; va[31]=0

2.11.2.44 vec16s vmsl (vec16s a)

Vector element shift left.

Parameters:

a Output vector and first input vector

Returns:

va shift left by one CU; va[31]=0

2.11.2.45 vec32u vmsl (vec32u a, vec32u b)

Vector element shift left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.46 vec32s vmsl (vec32s a, vec32s b)

Vector element shift left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.47 vec08u vmsl (vec08u *a*, vec08u *b*)

Vector element shift left.

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.48 vec08s vmsl (vec08s a, vec08s b)

Vector element shift left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.49 vec16u vmsl (vec16u a, vec16u b)

Vector element shift left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.50 vec16s vmsl (vec16s a, vec16s b)

Vector element shift left.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift left by one CU; va[31]=vb[0]

2.11.2.51 vec32u vmsr (vec32u a)

Vector element shift right.

Parameters:

a Output vector and first input vector

Returns:

va shift right by one CU; va[0]=0

2.11.2.52 vec32s vmsr (vec32s a)

Vector element shift right.

Parameters:

a Output vector and first input vector

Returns:

va shift right by one CU; va[0]=0

2.11.2.53 vec08u vmsr (vec08u a)

Vector element shift right.

Parameters:

a Output vector and first input vector

Returns:

va shift right by one CU; va[0]=0

2.11.2.54 vec08s vmsr (vec08s a)

Vector element shift right.

Parameters:

a Output vector and first input vector

Returns:

va shift right by one CU; va[0]=0

2.11.2.55 vec16u vmsr (vec16u a)

Vector element shift right.

Parameters:

a Output vector and first input vector

Returns:

va shift right by one CU; va[0]=0

2.11.2.56 vec16s vmsr (vec16s a)

Vector element shift right.

Parameters:

a Output vector and first input vector

Returns:

va shift right by one CU; va[0]=0

2.11.2.57 vec32u vmsr (vec32u *a*, vec32u *b*)

Vector element shift right.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.58 vec32s vmsr (vec32s a, vec32s b)

Vector element shift right.

Parameters:

- a Output vector and first input vector
- b The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.59 vec08u vmsr (vec08u a, vec08u b)

Vector element shift right.

Parameters:

- a Output vector and first input vector
- b The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.60 vec08s vmsr (vec08s a, vec08s b)

Vector element shift right.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.61 vec16u vmsr (vec16u a, vec16u b)

Vector element shift right.

- a Output vector and first input vector
- \boldsymbol{b} The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

2.11.2.62 vec16s vmsr (vec16s a, vec16s b)

Vector element shift right.

Parameters:

- a Output vector and first input vector
- **b** The second vector

Returns:

va shift right by one CU; va[0]=vb[31]

Index

Arithmetic Intrinsics, 8	other, 5 Shift Intrinsics, 49
Bitwise Intrinsics, 38	Specialized Multiplication Intrinsics, 95
clb	Specialized Shift Intrinsics, 92 Swap Intrinsics, 111
other, 3	swbreak
other, 3	other, 6
Comparison Intrinsics, 57	vabs varithInst, 14
Element Intrinsics, 68	valumist, 14 vabs_diff
haddss	varithInst, 14, 15
other, 4 hadduu	vabs_diffu other, 6
other, 4	vach
Memory Intrinsics, 79	varithInst, 15, 16
Move/Rotate Intrinsics, 115	varithInst, 16–18
other	vacm
clb, 3	varithInst, 18, 19 vadd
clz, 3	varithInst, 20-22
haddss, 4 hadduu, 4	vadd_sat
pent, 4	varithInst, 22 vaddx
rhaddss, 4	varithInst, 23–25
rhadduu, 5	vall
select, 5 swbreak, 6	other, 6
vabs_diffu, 6	vand vbitewiseInst, 40, 41
vall, 6	vany
vany, 6	other, 6
vhaddss, 7 vhadduu, 7	varithInst
vrhaddss, 7	vabs, 14 vabs_diff, 14, 15
vrhadduu, 8	vach, 15, 16
wait, 3	vacl, 16–18
Other Intrinsics, 1	vacm, 18, 19
pent	vadd, 20–22 vadd_sat, 22
other, 4	vadd_sat, 22 vaddx, 23–25
rhaddss	vasb, 25, 26
other, 4	vasbs, 26
rhadduu	vclb, 27, 28
other, 5	vclz, 28, 29
select	vmul, 29, 30 vpcnt, 30, 31
SCICCI	· r, ,

INDEX 139

vsat, 31, 32	vmad
vsub, 32–34	vspecmulInst, 99, 100
vsub_sat, 34, 35	vmemInst
vsubx, 35–37	vload, 82–87
vasb	vstore, 87–91
varithInst, 25, 26	vml
vasbs	vrotInst, 120, 121
varithInst, 26	vmr
vbitewiseInst	vrotInst, 122, 123
vand, 40, 41	vmrl
vcomplement, 42, 43	vrotInst, 123–126
vnot, 43–45	vmrr
vor, 45, 46	vrotInst, 127–130
vxor, 47, 48	vmsl
vclb	vrotInst, 130–133
varithInst, 27, 28	vmsr
vclz	vrotInst, 133–137
varithInst, 28, 29	vmul
vcmpInst	varithInst, 29, 30
vseq, 59, 60	vspecmulInst, 100, 101
vsequ, 60	vmul_shsh
vsge, 61, 62	vspecmulInst, 102
vsgeu, 62	vmul_shsl
vsgt, 62, 63	vspecmulInst, 102
vsgtu, 63	vmul_shuh
vsle, 64	vspecmulInst, 103
vsleu, 65	vmul_shul
vslt, 65, 66	vspecmulInst, 103, 104
vsltu, 66	vmul_slsh
vsne, 66, 67	vspecmulInst, 104
vsneu, 68	vmul_slsl
vcomplement	vspecmulInst, 104, 105
vbitewiseInst, 42, 43	vmul_sluh
velemInst	vspecmulInst, 105
vget, 70–72	vmul_slul
vput, 72–79	vspecmulInst, 106
vextract_hi	vmul_uhsh
vrotInst, 119	vspecmulInst, 106
vextract_lo	vmul_uhsl
vrotInst, 120	vspecmulInst, 107
vget	vmul_uhuh
velemInst, 70–72	vspecmulInst, 107, 108
vhaddss	vmul_uhul
other, 7	vspecmulInst, 108
vhadduu	vmul_ulsh
other, 7	vspecmulInst, 108, 109
vload	vmul_ulsl
vmemInst, 82–87	vspecmulInst, 109
vmac	vmul_uluh
vspecmulInst, 98, 99	vspecmulInst, 110
•	•

INDEX 140

vmul_ulul	vshiftInst, 50, 51
vspecmulInst, 110	vslc
vnot	vshiftInst, 51
vbitewiseInst, 43–45	vsle
vor	vcmpInst, 64
vbitewiseInst, 45, 46	vsleu
vpcnt	vempInst, 65
varithInst, 30, 31	vsll
vput	vshiftInst, 51, 52
velemInst, 72–79	vsllx
vrhaddss	vspecshiftInst, 92, 93
other, 7	vslo
vrhadduu	vshiftInst, 52, 53
other, 8	vslt
vrotInst	vempInst, 65, 66
vextract_hi, 119	vsltu
vextract_lo, 120	vcmpInst, 66
vml, 120, 121	vsne
vmr, 122, 123	vempInst, 66, 67
vmrl, 123–126	vsneu
vmrr, 127–130	vcmpInst, 68
vmsl, 130–133	vspecmulInst
vmsr, 133–137	vmac, 98, 99
vsat	vmad, 99, 100
varithInst, 31, 32	vmul, 100, 101
vselect	vmul_shsh, 102
vswapInst, 112, 113	vmul_shsl, 102
vseq	vmul_shuh, 103
vcmpInst, 59, 60	vmul_shul, 103, 104
vsequ	vmul_slsh, 104
vcmpInst, 60	vmul_slsl, 104, 105
vsge	vmul_sluh, 105
vcmpInst, 61, 62	vmul_slul, 106
vsgeu	vmul_uhsh, 106
vcmpInst, 62	vmul_uhsl, 107
vsgt	vmul_uhuh, 107, 108
vempInst, 62, 63	vmul_uhul, 108
vsgtu	vmul_ulsh, 108, 109
vempInst, 63	vmul_ulsl, 109
vshiftInst	vmul_uluh, 110
vsl, 50, 51	vmul_ulul, 110
vslc, 51	vspecshiftInst
vsll, 51, 52	vsllx, 92, 93
vslo, 52, 53	vsr, 93
vsr, 53, 54	vsrax, 93, 94
vsra, 54, 55	vsrlx, 94
vsrc, 55	vsr
vsrl, 56	vshiftInst, 53, 54
vsro, 56, 57	vspecshiftInst, 93
vsl	vsra

INDEX 141

```
vshiftInst, 54, 55
vsrax
     vspecshiftInst, 93, 94
vsrc
     vshiftInst, 55
vsrl
     vshiftInst, 56
vsrlx
     vspecshiftInst, 94
vsro
     vshiftInst, 56, 57
vstore
     vmemInst, 87–91
vsub
     varithInst, 32-34
vsub_sat
    varithInst, 34, 35
vsubx
     varithInst, 35–37
vswap
    vswapInst, 114, 115
vswapInst
    vselect, 112, 113
     vswap, 114, 115
vxor
     vbitewiseInst, 47, 48
wait
    other, 3
```