Vector Floating-point Processing Unit (VFPU) Instruction Manual

Table of Contents

VFPU Instructions

bvf

Branch on VFPU False

31 30 29 28 27 26	25 24 23 22 21	20 19 18	17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 1 0 0 1 0	0 1 0 0 0	imm3	0 0	offset
6	5	3	2	16

VFPU

Syntax:

```
bvf  imm3, offset
```

Instruction Type:

CPU interlock instruction

Processing Time:

latency : 0 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

If the value of the VFPU_CC control register bit indicated by the imm3 field is false (0), the program branches with a one instruction delay to the branch target address. The branch target address is the sum of the PC and the 16-bit offset after it is shifted left two bits and sign-extended to a 32 bit value.

Operation:

```
I+0: condition <- (VFPU_CC[imm3] == 0);
     target_offset <- sign_extend(offset<<2)
I+1: if condition then
       PC <-PC + target_offset;
     endif
```

bvfl

Branch on VFPU False Likely

31 30 29 28 27 26	25 24 23 22 21	20 19 18	17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 1 0 0 1 0	0 1 0 0 0	imm3	1 0	offset
6	5	3	2	16

VFPU

Syntax:

```
bvfl  imm3, offset
```

Instruction Type:

CPU interlock instruction

Processing Time:

latency : 0 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

If the value of the VFPU_CC control register bit indicated by the imm3 field is false (0), the program branches with a one instruction delay to the branch target address. The branch target address is the sum of the PC and the 16-bit offset after it is shifted left two bits and sign-extended to a 32 bit value. If the branch is not taken, the instruction in the branch delay slot is discarded.

Operation:

```
I+0: condition <- (VFPU_CC[imm3] == 0);
     target_offset <- sign_extend(offset<<2)
I+1: if condition then
       PC <-PC + target_offset;
     else
       NullifyCurrentInstruction();
     endif
```

bvt

Branch on VFPU True

31 30 29 28 27 26	25 24 23 22 21	20 19 18	17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 1 0 0 1 0	0 1 0 0 0	imm3	0 1	offset
6	5	3	2	16

VFPU

Syntax:

```
bvt  imm3, offset
```

Instruction Type:

CPU interlock instruction

Processing Time:

latency : 0　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

If the value of the VFPU_CC control register bit indicated by the imm3 field is true (1), the program branches with a one instruction delay to the branch target address. The branch target address is the sum of the PC and the 16-bit offset after it is shifted left two bits and sign-extended to a 32 bit value.

Operation:

```
I+0: condition <- (VFPU_CC[imm3] == 1);
     target_offset <- sign_extend(offset<<2);
I+1: if condition then
       PC <-PC + target_offset;
     endif
```

bvtl

Branch on VFPU True Likely

31 30 29 28 27 26	25 24 23 22 21	20 19 18	17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 1 0 0 1 0	0 1 0 0 0	imm3	1 1	offset
6	5	3	2	16

VFPU

Syntax:

```
bvtl  imm3, offset
```

Instruction Type:

CPU interlock instruction

Processing Time:

latency : 0   pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

If the value of the VFPU_CC control register bit indicated by the imm3 field is true (1), the program branches with a one instruction delay to the branch target address. The branch target address is the sum of the PC and the 16-bit offset after it is shifted left two bits and sign-extended to a 32 bit value. If the branch is not taken, the instruction in the branch delay slot is discarded.

Operation:

```
I+0: condition <- (VFPU_CC[imm3] == 1 );
     target_offset <- sign_extend(offset<<2);
I+1: if condition then
       PC <-PC + target_offset;
     else
       NullifyCurrentInstruction();
     endif
```

lv.s

Load Single Word to VFPU

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2	1 0
1 1 0 0 1 0	rs	vt[4:0]	offset	vt [6:5]
6	5	5	14	2

VFPU

Syntax:

```
lv.s  vt, offset(rs)
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

An effective address is generated by ignoring the two low-order bits of the 16-bit offset, sign-extending the remaining 14 bits, and adding the result to the contents of CPU general-purpose register rs. A one-word value starting at the virtual address represented by this effective address is stored at the location in the matrix register file indicated by vt. If the address is not word aligned, the CPU generates an address error exception.

Operation:

```
vAddr <- sign_extend( {offset[15:2], 2'b0} ) + GPR[rs];
pAddr <- AddressTranslation( vAddr, DATA, LOAD );
memword <- LoadMemory( SINGLEWORD, pAddr, vAddr, DATA );
WriteMatrix( SINGLEWORD, vt, memword );
```

Exceptions:

```
Address Error exception
```

```
        Bus Error exception
```

lv.q

Load Quad Word to VFPU

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2	1	0
1 1 0 1 1 0	rs	vt[4:0]	offset	0	vt [5]
6	5	5	14	1	1

VFPU

Syntax:

```
lv.q  vt, offset(rs)
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3   pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

An effective address is generated by ignoring the two low-order bits of the 16-bit offset, sign-extending the remaining 14 bits, and adding the result to the contents of CPU general-purpose register rs. A quadword value starting at the virtual address represented by this effective address is stored at the location in the matrix register file indicated by vt. If the address is not quadword aligned, the CPU generates an address error exception.

Operation:

```
vAddr <- sign_extend( {offset[15:2], 2'b0} ) + GPR[rs];
pAddr <- AddressTranslation( vAddr, DATA, LOAD );
memword <- LoadMemory( QUADWORD, pAddr, vAddr, DATA );
WriteMatrix( QUADWORD, {1'b0, vt[5:0]}, memword );
```

Exceptions:

```
Address Error exception
Bus Error exception
```

mfv

Move Word from VFPU

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
0 1 0 0 1 0	0 0 0 1 1	rt	0 0 0 0 0 0 0 0 0	vd
6	5	5	9	7

VFPU

Syntax:

```
mfv  rt, vd
```

Instruction Type:

CPU interlock instruction

Processing Time:

latency : 0   pitch : 6

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

The contents of the matrix register indicated by vd are copied to CPU general-purpose register rt.

Operation:

```
dataword <- ReadMatrix( SINGLEWORD, vd );
GPR[rt] <- dataword;
```

mfvc

Move Word from VFPU Control

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0 1 0 0 1 0	0 0 0 1 1	rt	0 0 0 0 0 0 0 0	imm8
6	5	5	8	8

VFPU

Syntax:

```
mfvc  rt, imm8
```

Instruction Type:

CPU interlock instruction

Processing Time:

latency : 0 pitch : 6

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

The contents of the VFPU control register indicated by the imm8 field are copied to CPU general-purpose register rt.
imm8 can range from 128 to 143. In other words, bit 7 of the opcode is always 1.

Notes:

When reading control registers, the pipeline will not interlock because of a RAW hazard.
If the preceding instruction performs a rewrite to a control register, then one latency cycle's worth of vnop instructions must be inserted before the mfvc instruction.
In these situations, note that the assembler will not automatically insert the vnop instructions.

Operation:

```
dataword <- ReadControl( imm8 );
GPR[rt] <- dataword;
```

mtv

Move Word to VFPU

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
0 1 0 0 1 0	0 0 1 1 1	rt	0 0 0 0 0 0 0 0 0	vd
6	5	5	9	7

VFPU

Syntax:

```
mtv  rt, vd
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3   pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

The contents of CPU general-purpose register rt are copied to the location in the matrix register file indicated by vd.

Operation:

```
dataword <- GPR[rt];
WriteMatrix( SINGLEWORD, vd, dataword );
```

mtvc

Move Word to VFPU Control

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
0 1 0 0 1 0	0 0 1 1 1	rt	0 0 0 0 0 0 0 0	imm8
6	5	5	8	8

VFPU

Syntax:

```
mtvc  rt, imm8
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

The contents of CPU general-purpose register rt are copied to the VFPU control register indicated by the imm8 field.

imm8 can have values from 128 to 143. In other words, bit 7 of the opcode is always 1.

Operation:

```
dataword <- GPR[rt];
WriteControl( imm8, dataword );
```

sv.s

Store Single Word from VFPU

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2	1 0
1 1 1 0 1 0	rs	vt[4:0]	offset	vt [6:5]
6	5	5	14	2

VFPU

Syntax:

```
sv.s  vt, offset(rs)
```

Instruction Type:

CPU interlock instruction

Processing Time:

latency : 0 pitch : 7

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

A virtual address is generated by ignoring the two low-order bits of the 16-bit offset, sign-extending the remaining 14 bits, and adding the result to the contents of CPU general-purpose register rs. The single word from the location in the matrix register file indicated by vt is written to memory at this effective address.
If the effective address is not word aligned, the CPU generates an address error exception.

Operation:

```
vAddr <- sign_extend({offset[15:2], 2'b0}) + GPR[rs];
pAddr <- AddressTranslation(vAddr, DATA, STORE);
dataword <- ReadMatrix( SINGLEWORD, vt );
StoreMemory( SINGLEWORD, dataword, pAddr, vAddr, DATA);
```

Exceptions:

```
Address Error exception
```

sv.q

Store Quad Word from VFPU

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2	1	0
1 1 1 1 1 0	rs	vt[4:0]	offset	0	vt [5]
6	5	5	14	1	1

VFPU

Syntax:

```
sv.q  vt, offset(rs)
```

Instruction Type:

CPU interlock instruction

Processing Time:

latency : 0	pitch : 7	for cached addresses
latency : 0	pitch : 10	for non-cached addresses

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

A virtual address is generated by ignoring the two low-order bits of the 16-bit offset, sign-extending the remaining 14 bits, and adding the result to the contents of CPU general-purpose register rs. The quadword from locations in the matrix register file indicated by vt is written to memory at this effective address.
If the effective address is not quadword aligned, the CPU generates an address error exception.

Operation:

```
vAddr <- sign_extend({offset[15:2], 2'b0}) + GPR[rs];
pAddr <- AddressTranslation(vAddr, DATA, STORE);
dataword <- ReadMatrix( QUADWORD, {1'b0, vt[5:0]} );
StoreMemory( QUADWORD, dataword, pAddr, vAddr, DATA);
```

Exceptions:

```
        Address Error exception
```

sv.q

Store Quad Word to Write Buffer

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2	1	0
1 1 1 1 1 0	rs	vt[4:0]	offset	1	vt [5]
6	5	5	14	1	1

VFPU

Syntax:

```
sv.q  vt, offset(rs),  wb
```

Instruction Type:

Pipeline (non-cached) / CPU interlock (cached) instruction

Processing Time:

latency : 0	pitch : 7	for cached addresses
latency : 0	pitch : 1	for non-cached addresses

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

A virtual address is generated by ignoring the two low-order bits of the 16-bit offset, sign-extending the remaining 14 bits, and adding the result to the contents of CPU general-purpose register rs.
If this effective address is in the cached space, the quadword from locations in the matrix register file indicated by vt is written to memory at that address. At the same time, the address is converted to a physical address and the quadword is also written to physical memory via the write buffer.
If the effective address is in the non-cached space, it is converted into a physical address, and the quadword from locations in the matrix register file indicated by vt are only written to physical memory via the write buffer. In addition, if the effective address is not quadword aligned, then the CPU generates an address error exception.

Notes:

When the VFPU write buffer is full, if the next VFPU instruction is an arithmetic instruction which performs a write to the matrix register file, then a known problem occurs in which incorrect values are written to the registers as calculation results. To avoid this problem, insert a vnop instruction after the sv.q,wb instruction, before executing the arithmetic instruction in question. For psp-as 1.5.1 and later, the assembler will automatically insert a vnop when necessary.
When an lv.s or lv.q instruction is executed after an sv.q,wb instruction, there is a known error in which incorrect data may end up getting loaded. To avoid this problem, you should execute a vsync2 instruction immediately before the lv.s or lv.q instruction that comes after the sv.q,wb instruction. In psp-as 1.10.4 or later, the assembler will automatically insert a vnop when necessary.

Operation:

```
vAddr <- sign_extend({offset[15:2], 2'b0}) + GPR[rs];
pAddr <- AddressTranslation(vAddr, DATA, STORE);
dataword <- ReadMatrix( QUADWORD, {1'b0,vt[5:0]} );
if( isCacheSpace( pAddr ) )
   StoreMemory( QUADWORD, dataword, pAddr, vAddr, DATA);
StoreMemory_WriteBuffer( QUADWORD, dataword, pAddr, vAddr, DATA);
```

Exceptions:

```
Address Error exception
```

svl.q

Store Quad Word Left from VFPU

31–26	25–21	20–16	15–2	1	0
1 1 1 1 0 1	rs	vt[4:0]	offset	0	vt [5]
6	5	5	14	1	1

VFPU

Syntax:

```
svl.q  vt, offset(rs)
```

Instruction Type:

CPU interlock instruction

Processing Time:

latency : 0	pitch : 7	for cached addresses.
latency : 0	pitch : 10	for non-cached addresses.

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

A virtual address is generated by ignoring the two low-order bits of the 16-bit offset, sign-extending the remaining 14 bits, and adding the result to the contents of CPU general-purpose register rs. From one to four words are read from locations in the matrix register file indicated by vt, then stored to memory such that the high-order word from the matrix register file is stored at the effective address, and the low-order word is stored at the end of the quadword boundary. The words are stored in memory starting with the leftmost word within the quadword from the matrix register file. Any remaining words to the right in the quadword are not stored in memory and are unaffected by this instruction. If the effective address is not word aligned, the CPU generates an address error exception.

Operation:

```
vAddr    <- sign_extend({offset[15:2],2'b0}) + GPR[rs];
pAddr    <- AddressTranslation(vAddr, DATA, STORE);
offset   <- pAddr[3:2];
dataword <- LoadMemory( QUADWORD, pAddr, vAddr, DATA );
d        <- ReadMatrix( QUADWORD,{1'b0, vt[5:0]} );
switch( offset )
{
  case 0 : dataword[0] <- d[3]; break;
  case 1 : dataword[1] <- d[3];
           dataword[0] <- d[2]; break;
  case 2 : dataword[2] <- d[3];
           dataword[1] <- d[2];
           dataword[0] <- d[1]; break;
  case 3 : dataword[3] <- d[3];
           dataword[2] <- d[2];
           dataword[1] <- d[1];
           dataword[0] <- d[0]; break;
}
StoreMemory( QUADWORD, dataword, pAddr, vAddr, DATA);
```

Exceptions:

```
Address Error exception
```

svr.q

Store Quad Word Right from VFPU

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2	1	0
1 1 1 1 0 1	rs	vt[4:0]	offset	1	vt [5]
6	5	5	14	1	1

VFPU

Syntax:

```
svr.q  vt, offset(rs)
```

Instruction Type:

CPU interlock instruction

Processing Time:

latency : 0	pitch : 7	for cached addresses.
latency : 0	pitch : 10	for non-cached addresses.

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

A virtual address is generated by ignoring the two low-order bits of the 16-bit offset, sign-extending the remaining 14 bits, and adding the result to the contents of CPU general-purpose register rs. From one to four words are read from locations in the matrix register file indicated by vt, then stored to memory such that the low-order word from the matrix register file is stored at the effective address, and the high-order word is stored at the quadword boundary. The words are stored in memory starting with the rightmost word within the quadword from the matrix register file. Any remaining words to the left in the quadword are not stored in memory and are unaffected by this instruction. If the effective address is not word aligned, the CPU generates an address error exception.

Operation:

```
vAddr    <- sign_extend({offset[15:2],2'b0}) + GPR[rs];
pAddr    <- AddressTranslation(vAddr, DATA, STORE);
offset   <- pAddr[3:2];
dataword <- LoadMemory( QUADWORD, pAddr, vAddr, DATA );
d        <- ReadMatrix( QUADWORD,{1'b0, vt[5:0]} );
switch( offset )
{
  case 0 : dataword[3] <- d[3];
           dataword[2] <- d[2];
           dataword[1] <- d[1];
           dataword[0] <- d[0]; break;
  case 1 : dataword[2] <- d[3];
           dataword[1] <- d[2];
           dataword[0] <- d[1]; break;
  case 2 : dataword[1] <- d[3];
           dataword[0] <- d[2]; break;
  case 3 : dataword[0] <- d[3]; break;
}
StoreMemory( QUADWORD, dataword, pAddr, vAddr, DATA);
```

Exceptions:

```
Address Error exception
```

vabs.s

Absolute Value Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 0 1 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vabs.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Valid

Description:

The absolute value of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Special solutions are as follows.

abs(+***nan***) = +***nan***

abs(-***nan***) = +***nan***

abs(+***inf***) = +***inf***

abs(-***inf***) = +***inf***

abs(+0) = +0.0

abs(-0) = +0.0

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- abs( s[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vabs.p

Absolute Value Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 0 1 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vabs.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Valid

Description:

The absolute values of the floating-point values of two elements from the matrix registers indicated by vs are calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Special solutions are as follows.

abs(+***nan***) = +***nan***

abs(-***nan***) = +***nan***

abs(+***inf***) = +***inf***

abs(-***inf***) = +***inf***

abs(+0) = +0.0

abs(-0) = +0.0

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- abs( s[0] );
d[1] <- abs( s[1] );
```

```
        WriteMatrix( PAIRWORD, vd, d );
```

vabs.t

Absolute Value Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 0 1 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vabs.t  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Valid

Description:

The absolute values of the floating-point values of three elements from the matrix registers indicated by vs are calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Special solutions are as follows.

abs(+***nan***) = +***nan***

abs(-***nan***) = +***nan***

abs(+***inf***) = +***inf***

abs(-***inf***) = +***inf***

abs(+0) = +0.0

abs(-0) = +0.0

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- abs( s[0] );
d[1] <- abs( s[1] );
```

```
        d[2] <- abs( s[2] );
        WriteMatrix( TRIPLEWORD, vd, d );
```

vabs.q

Absolute Value Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 0 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vabs.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency：3　　　　pitch：1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Valid

Description:

The absolute values of the floating-point values of four elements from the matrix registers indicated by vs are calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.
Special solutions are as follows.

abs(+***nan***) = +***nan***

abs(-***nan***) = +***nan***

abs(+***inf***) = +***inf***

abs(-***inf***) = +***inf***

abs(+0) = +0.0

abs(-0) = +0.0

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- abs( s[0] );
d[1] <- abs( s[1] );
```

```
        d[2] <- abs( s[2] );
        d[3] <- abs( s[3] );
        WriteMatrix( QUADWORD, vd, d );
```

vadd.s

Add Single Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 0	0 0 0	vt	0	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vadd.s  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

One element from the matrix register indicated by vs is added to one element from the matrix register indicated by vt. The elements are treated as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <- s[0] + t[0];
WriteMatrix( SINGLEWORD, vd, d );
```

vadd.p

Add Pair Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 0	0 0 0	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vadd.p  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Two elements from the matrix registers indicated by vs are added to two elements from the matrix registers indicated by vt. The elements are treated as floating-point numbers. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vt );
d[0] <- s[0] + t[0];
d[1] <- s[1] + t[1];
WriteMatrix( PAIRWORD, vd, d );
```

vadd.t

Add Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 0	0 0 0	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vadd.t  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Three elements from the matrix registers indicated by vs are added to three elements from the matrix registers indicated by vt. The elements are treated as floating-point numbers. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- s[0] + t[0];
d[1] <- s[1] + t[1];
d[2] <- s[2] + t[2];
WriteMatrix( TRIPLEWORD, vd, d );
```

vadd.q

Add Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 0	0 0 0	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vadd.q  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Four elements from the matrix registers indicated by vs are added to four elements from the matrix registers indicated by vt. The elements are treated as floating-point numbers. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
d[0] <- s[0] + t[0];
d[1] <- s[1] + t[1];
d[2] <- s[2] + t[2];
d[3] <- s[3] + t[3];
WriteMatrix( QUADWORD, vd, d );
```

vasin.s

Arc Sine Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 1 1 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vasin.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The arcsine of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$|\, \mathit{approx\_asin}(x) / \mathit{M\_PI\_2} - \mathit{asin}(x) / \mathit{M\_PI\_2} \,| < 4.6 \times 10^{-7}$; $0 <= |\, x \,| < 0.75$

$|\, \mathit{approx\_asin}(x) / \mathit{M\_PI\_2} - \mathit{asin}(x) / \mathit{M\_PI\_2} \,| < 2.0 \times 10^{-2}$; $0.75 <= |\, x \,| <= 1.0$

Special solutions are as follows.

$\mathit{approx\_asin}(\mathit{nan}) = \mathit{nan}$

$\mathit{approx\_asin}(+\mathit{inf}) = \mathit{nan}$

$\mathit{approx\_asin}(-\mathit{inf}) = \mathit{nan}$

$\mathit{approx\_asin}(+0.0) = +0.0$

$\mathit{approx\_asin}(-0.0) = -0.0$

$\mathit{approx\_asin}(x) = \mathit{nan}$; $-\mathit{inf} < x < -1.0$

$\mathit{approx\_asin}(x) = \mathit{nan}$; $+1.0 < x < +\mathit{inf}$

Notes:

The value of the arcsine, which is generated from the result of the vasin.s instruction, is in units in which 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- approx_asin( s[0] ) / M_PI_2;
WriteMatrix( SINGLEWORD, vd, d );
```

vasin.p

Arc Sine Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 1 1 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vasin.p  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The arcsines of the floating-point values of two elements from the matrix registers indicated by vs are calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \textit{\textbf{approx\_asin}}(\textbf{\textit{x}}) / \textit{\textbf{M\_PI\_2}} - \textit{\textbf{asin}}(\textbf{\textit{x}}) / \textit{\textbf{M\_PI\_2}} | < 4.6 \times 10^{-7}$; $0 <= | \textbf{\textit{x}} | < 0.75$

$| \textit{\textbf{approx\_asin}}(\textbf{\textit{x}}) / \textit{\textbf{M\_PI\_2}} - \textit{\textbf{asin}}(\textbf{\textit{x}}) / \textit{\textbf{M\_PI\_2}} | < 2.0 \times 10^{-2}$; $0.75 <= | \textbf{\textit{x}} | <=1.0$

Special solutions are as follows.

approx_asin(***nan***) = ***nan***

approx_asin(+***inf***) = ***nan***

approx_asin(-***inf***) = ***nan***

approx_asin(+0.0) = +0.0

approx_asin(-0.0) = -0.0

approx_asin(***x***) = ***nan***; -***inf*** < ***x*** < -1.0

approx_asin(***x***) = ***nan***; +1.0 < ***x*** < +***inf***

Notes:

The value of the arcsine, which is generated from the result of the vasin.p instruction, is in units in which 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- approx_asin( s[0] ) / M_PI_2;
d[1] <- approx_asin( s[1] ) / M_PI_2;
WriteMatrix( PAIRWORD, vd, d );
```

vasin.t

Arc Sine Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 1 1 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vasin.t  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency：9　　　　pitch：3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The arcsines of the floating-point values of three elements from the matrix registers indicated by vs are calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$|\ \boldsymbol{approx\_asin}(\boldsymbol{x})\ /\boldsymbol{M\_PI\_2} - \boldsymbol{asin}(\boldsymbol{x})\ /\boldsymbol{M\_PI\_2}\ | < 4.6 \times 10^{-7}$; $0 <= |\ \boldsymbol{x}\ | <= 0.75$

$|\ \boldsymbol{approx\_asin}(\boldsymbol{x})\ /\boldsymbol{M\_PI\_2} - \boldsymbol{asin}(\boldsymbol{x})\ /\boldsymbol{M\_PI\_2}\ | < 2.0 \times 10^{-2}$; $0.75 <= |\ \boldsymbol{x}\ | <=1.0$

Special solutions are as follows.

$\boldsymbol{approx\_asin}(\boldsymbol{nan}) = \boldsymbol{nan}$

$\boldsymbol{approx\_asin}(+\boldsymbol{inf}) = \boldsymbol{nan}$

$\boldsymbol{approx\_asin}(-\boldsymbol{inf}) = \boldsymbol{nan}$

$\boldsymbol{approx\_asin}(+0.0) = +0.0$

$\boldsymbol{approx\_asin}(-0.0) = -0.0$

$\boldsymbol{approx\_asin}(\boldsymbol{x}) = \boldsymbol{nan}$; $-\boldsymbol{inf} < \boldsymbol{x} < -1.0$

$\boldsymbol{approx\_asin}(\boldsymbol{x}) = \boldsymbol{nan}$; $+1.0 < \boldsymbol{x} < +\boldsymbol{inf}$

Notes:

The value of the arcsine, which is generated from the result of the vasin.t instruction, is in units in which 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- approx_asin( s[0] ) / M_PI_2;
d[1] <- approx_asin( s[1] ) / M_PI_2;
d[2] <- approx_asin( s[2] ) / M_PI_2;
WriteMatrix( TRIPLEWORD, vd, d );
```

vasin.q

Arc Sine Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 1 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vasin.q  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10　　　pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The arcsines of the floating-point values of four elements from the matrix registers indicated by vs are calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \boldsymbol{approx\_asin}(\boldsymbol{x}) / \boldsymbol{M\_PI\_2} - \boldsymbol{asin}(\boldsymbol{x}) / \boldsymbol{M\_PI\_2} | < 4.6 \times 10^{-7}$; $0 <= | \boldsymbol{x} | < 0.75$

$| \boldsymbol{approx\_asin}(\boldsymbol{x}) / \boldsymbol{M\_PI\_2} - \boldsymbol{asin}(\boldsymbol{x}) / \boldsymbol{M\_PI\_2} | < 2.0 \times 10^{-2}$; $0.75 <= | \boldsymbol{x} | <=1.0$

Special solutions are as follows.

approx_asin(***nan***) = ***nan***

approx_asin(+***inf***) = ***nan***

approx_asin(-***inf***) = ***nan***

approx_asin(+0.0) = +0.0

approx_asin(-0.0) = -0.0

approx_asin(***x***) = ***nan***; -***inf*** < ***x*** < -1.0

approx_asin(***x***) = ***nan***; +1.0 < ***x*** < +***inf***

Notes:

The value of the arcsine, which is generated from the result of the vasin.q instruction, is in units in which 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- approx_asin( s[0] ) / M_PI_2;
d[1] <- approx_asin( s[1] ) / M_PI_2;
d[2] <- approx_asin( s[2] ) / M_PI_2;
d[3] <- approx_asin( s[3] ) / M_PI_2;
WriteMatrix( QUADWORD, vd, d );
```

vavg.p

Average Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 1 1 1 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vavg.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

The average of the floating-point values of two elements from the matrix registers indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- s[0]/2;
d[0] <- d[0] + s[1]/2;
WriteMatrix( SINGLEWORD, vd, d );
```

vavg.t

Average Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 1 1 1 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vavg.t  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

The average of the floating-point values of three elements from the matrix registers indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- s[0]/3;
d[0] <- d[0] + s[1]/3;
d[0] <- d[0] + s[2]/3;
WriteMatrix( SINGLEWORD, vd, d );
```

vavg.q

Average Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 1 1 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vavg.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

The average of the floating-point values of four elements from the matrix registers indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- s[0]/4;
d[0] <- d[0] + s[1]/4;
d[0] <- d[0] + s[2]/4;
d[0] <- d[0] + s[3]/4;
WriteMatrix( SINGLEWORD, vd, d );
```

vbfy1.p

Butterfly 1 Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 0 1 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vbfy1.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Valid

Description:

The butterfly of the floating-point values of two elements from the matrix registers indicated by vs is calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- s[0] + s[1];
d[1] <- s[0] - s[1];
WriteMatrix( PAIRWORD, vd, d );
```

vbfy1.q

Butterfly 1 Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 0 1 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vbfy1.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Valid

Description:

The butterfly of the floating-point values of four elements from the matrix registers indicated by vs is calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- s[0] + s[1];
d[1] <- s[0] - s[1];
d[2] <- s[2] + s[3];
d[3] <- s[2] - s[3];
WriteMatrix( QUADWORD, vd, d );
```

vbfy2.q

Butterfly 2 Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 0 1 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vbfy2.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Valid

Description:

The butterfly of the floating-point values of four elements from the matrix registers indicated by vs is calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- s[0] + s[2];
d[1] <- s[1] + s[3];
d[2] <- s[0] - s[2];
d[3] <- s[1] - s[3];
WriteMatrix( QUADWORD, vd, d );
```

vc2i.s

Convert signed char single word to integer

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 1 0 0 1 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vc2i.s  vd, vs
```

Instruction Type::

Pipeline instruction

Processing time:

latency: 3   pitch: 1

Prefixing:

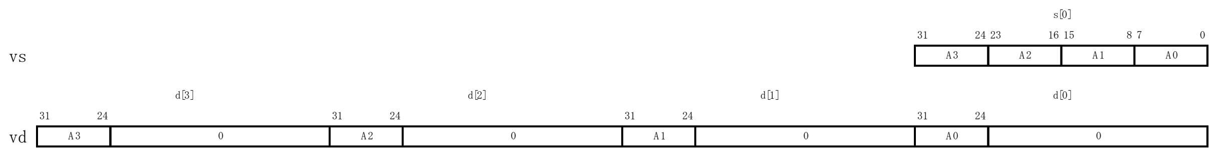
vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Only write mask is valid

Description:

Unpacks the 32-bit packed data in the matrix register specified by vs, converts the data from signed 8-bit integers to signed 32-bit integers, and stores the four integer elements at the locations in the matrix register specified by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- {s[ 7: 0], 24'b0};
d[1] <- {s[15: 8], 24'b0};
d[2] <- {s[23:16], 24'b0};
d[3] <- {s[31:24], 24'b0};
WriteMatrix( QUADWORD, vd, d );
```



vcmovf.s

Conditional Move on False Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19	18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 1 0 1 0 1	imm3	0	vs	0	vd
6	7	3	1	7	1	7

VFPU

Syntax:

```
vcmovf.s  vd, vs, imm3
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Use prohibited

Description:

If the condition of the VFPU_CC control register bit indicated by the imm3 field is false (0), one floating-point element from the matrix register indicated by vs is copied to the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vd );
c <- ReadControlBit( VFPU_CC );
d[0] <-  t[0];
if( imm3<6 )
    begin
       if( !c[imm3] )
           begin
              d[0] <-  s[0];
           end
    end
else if( imm3==6 )
    begin
       if( !c[0] )
```

```
                d[0] <-  s[0];
            end
        WriteMatrix( SINGLEWORD, vd, d );
```

vcmovf.p

Conditional Move on False Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19	18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 1 0 1 0 1	imm3	0	vs	1	vd
6	7	3	1	7	1	7

VFPU

Syntax:

```
vcmovf.p  vd, vs, imm3
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Use prohibited

Description:

If the condition of the VFPU_CC control register bit indicated by the imm3 field is false (0), two floating-point elements from the matrix registers indicated by vs are copied to locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vd );
c <- ReadControlBit( VFPU_CC );
d[0] <-  t[0];
d[1] <-  t[1];
if( imm3<6 )
    begin
       if( !c[imm3] )
          begin
             d[0] <-  s[0];
             d[1] <-  s[1];
          end
    end
else if( imm3==6 )
```

```
        begin
            if( !c[0] )
                d[0] <-  s[0];
            if( !c[1] )
                d[1] <-  s[1];
        end
    WriteMatrix( PAIRWORD, vd, d );
```

vcmovf.t

Conditional Move on False Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19	18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 1 0 1 0 1	imm3	1	vs	0	vd
6	7	3	1	7	1	7

VFPU

Syntax:

```
vcmovf.t  vd, vs, imm3
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Use prohibited

Description:

If the condition of the VFPU_CC control register bit indicated by the imm3 field is false (0), three floating-point elements from the matrix registers indicated by vs are copied to locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vd );
c <- ReadControlBit( VFPU_CC );
d[0] <-  t[0];
d[1] <-  t[1];
d[2] <-  t[2];
if( imm3<6 )
    begin
       if( !c[imm3] )
          begin
             d[0] <-  s[0];
             d[1] <-  s[1];
             d[2] <-  s[2];
          end
```

```
            end
        else if( imm3==6 )
            begin
                if( !c[0] )
                    d[0] <-  s[0];
                if( !c[1] )
                    d[1] <-  s[1];
                if( !c[2] )
                    d[2] <-  s[2];
            end
        WriteMatrix( TRIPLEWORD, vd, d );
```

vcmovf.q

Conditional Move on False Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19	18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 1 0 1 0 1	imm3	1	vs	1	vd
6	7	3	1	7	1	7

VFPU

Syntax:

```
vcmovf.q  vd, vs, imm3
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5   pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Use prohibited

Description:

If the condition of the VFPU_CC control register bit indicated by the imm3 field is false (0), four floating-point elements from the matrix registers indicated by vs are copied to locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vd );
c <- ReadControlBit( VFPU_CC );
d[0] <-  t[0];
d[1] <-  t[1];
d[2] <-  t[2];
d[3] <-  t[3];
if( imm3<6 )
    begin
        if( !c[imm3] )
            begin
                d[0] <-  s[0];
                d[1] <-  s[1];
                d[2] <-  s[2];
                d[3] <-  s[3];
            end
    end
else if( imm3==6 )
    begin
        if( !c[0] )
            d[0] <-  s[0];
        if( !c[1] )
            d[1] <-  s[1];
        if( !c[2] )
            d[2] <-  s[2];
        if( !c[3] )
            d[3] <-  s[3];
    end
WriteMatrix( QUADWORD, vd, d );
```

vcmovt.s

Conditional Move on True Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19	18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 1 0 1 0 0	imm3	0	vs	0	vd
6	7	3	1	7	1	7

VFPU

Syntax:

```
vcmovt.s  vd, vs, imm3
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Use prohibited

Description:

If the condition of the VFPU_CC control register bit indicated by the imm3 field is true (1), one floating-point element from the matrix register indicated by vs is copied to the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vd );
c <- ReadControlBit( VFPU_CC );
d[0] <-  t[0];
if( imm3<6 )
    begin
       if( c[imm3] )
          begin
             d[0] <-  s[0];
          end
    end
else if( imm3==6 )
    begin
       if( c[0] )
```

```
                d[0] <-  s[0];
            end
        WriteMatrix( SINGLEWORD, vd, d );
```

vcmovt.p

Conditional Move on True Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19	18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 1 0 1 0 0	imm3	0	vs	1	vd
6	7	3	1	7	1	7

VFPU

Syntax:

```
vcmovt.p  vd, vs, imm3
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Use prohibited

Description:

If the condition of the VFPU_CC control register bit indicated by the imm3 field is true (1), two floating-point elements from the matrix registers indicated by vs are copied to locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vd );
c <- ReadControlBit( VFPU_CC );
d[0] <-  t[0];
d[1] <-  t[1];
if( imm3<6 )
    begin
        if( c[imm3] )
            begin
                d[0] <-  s[0];
                d[1] <-  s[1];
            end
    end
else if( imm3==6 )
```

```
        begin
            if( c[0] )
                d[0] <-  s[0];
            if( c[1] )
                d[1] <-  s[1];
        end
    WriteMatrix( PAIRWORD, vd, d );
```

vcmovt.t

Conditional Move on True Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19	18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 1 0 1 0 0	imm3	1	vs	0	vd
6	7	3	1	7	1	7

VFPU

Syntax:

```
vcmovt.t  vd, vs, imm3
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Use prohibited

Description:

If the condition of the VFPU_CC control register bit indicated by the imm3 field is true (1), three floating-point elements from the matrix registers indicated by vs are copied to locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vd );
c <- ReadControlBit( VFPU_CC );
d[0] <-  t[0];
d[1] <-  t[1];
d[2] <-  t[2];
if( imm3<6 )
    begin
       if( c[imm3] )
           begin
              d[0] <-  s[0];
              d[1] <-  s[1];
              d[2] <-  s[2];
           end
```

```
        end
    else if( imm3==6 )
        begin
            if( c[0] )
                d[0] <-  s[0];
            if( c[1] )
                d[1] <-  s[1];
            if( c[2] )
                d[2] <-  s[2];
        end
    WriteMatrix( TRIPLEWORD, vd, d );
```

vcmovt.q

Conditional Move on True Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19	18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 1 0 1 0 0	imm3	1	vs	1	vd
6	7	3	1	7	1	7

VFPU

Syntax:

```
vcmovt.q  vd, vs, imm3
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5   pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Use prohibited

Description:

If the condition of the VFPU_CC control register bit indicated by the imm3 field is true (1), four floating-point elements from the matrix registers indicated by vs are copied to locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vd );
c <- ReadControlBit( VFPU_CC );
d[0] <-  t[0];
d[1] <-  t[1];
d[2] <-  t[2];
d[3] <-  t[3];
if( imm3<6 )
    begin
       if( c[imm3] )
          begin
             d[0] <-  s[0];
             d[1] <-  s[1];
             d[2] <-  s[2];
```

```
                d[3] <-  s[3];
            end
        end
    else if( imm3==6 )
        begin
            if( c[0] )
                d[0] <-  s[0];
            if( c[1] )
                d[1] <-  s[1];
            if( c[2] )
                d[2] <-  s[2];
            if( c[3] )
                d[3] <-  s[3];
        end
    WriteMatrix( QUADWORD, vd, d );
```

vcmp.s

Compare Single Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
0 1 1 0 1 1	0 0 0	vt	0	vs	0 0 0 0	cond
6	3	7	1	7	4	4

VFPU

Syntax:

```
vcmp.s  cond, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	No effect

Description:

One element from the matrix register indicated by vs is compared with one element from the matrix register indicated by vt according to the condition cond. The elements are treated as floating-point numbers. The 6-bit comparison result is stored in control register VFPU_CC. The comparison result can be referenced by the bvt, bvf, bvtl, and bvfl conditional branching instructions, or by the vcmovt and vcmovf conditional assignment instructions.
The following mnemonics can be used for cond.

Code (cond)	Mnemonic	Function
0	FL	Always false
1	EQ	Equal
2	LT	Less than
3	LE	Less than or equal
4	TR	Always true
5	NE	Not equal
6	GE	Greater than or equal

Code (cond)	Mnemonic	Function
7	GT	Greater than
8	EZ	Equal to zero
9	EN	Equal to NaN
10	EI	Absolute value equal to infinity
11	ES	Equal to infinity or NaN
12	NZ	Not equal to zero
13	NN	Not equal to NaN
14	NI	Absolute value not equal to infinity
15	NS	Not equal to infinity and not equal to NaN

Notes:

If the cond comparison condition is EZ, EN, EI, ES, NZ, NN, NI, or NS, then only the matrix register specified by vs will be targeted for comparison. In this case, vt is a dummy and can be omitted. If vt is omitted, the assembler is able to avoid generating unintended RAW hazards by specifying in vt the same value as that in vs.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
c[0] <- Compare( cond, s[0], t[0] );
c[4] <- c[0];
c[5] <- c[0];
WriteControlBit( VFPU_CC, 0, c[0] );
WriteControlBit( VFPU_CC, 4, c[4] );
WriteControlBit( VFPU_CC, 5, c[5] );
```

vcmp.p

Compare Pair Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
0 1 1 0 1 1	0 0 0	vt	0	vs	1 0 0 0	cond
6	3	7	1	7	4	4

VFPU

Syntax:

```
vcmp.p  cond, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	No effect

Description:

Two elements from the matrix registers indicated by vs are compared with two elements from the matrix registers indicated by vt according to the condition cond. The elements are treated as floating-point numbers. The 6-bit comparison result is stored in control register VFPU_CC. The comparison result can be referenced by the bvt, bvf, bvtl, and bvfl conditional branching instructions, or by the vcmovt and vcmovf conditional assignment instructions.
The following mnemonics can be used for cond.

Code (cond)	Mnemonic	Function
0	FL	Always false
1	EQ	Equal
2	LT	Less than
3	LE	Less than or equal
4	TR	Always true
5	NE	Not equal
6	GE	Greater than or equal

Code (cond)	Mnemonic	Function
7	GT	Greater than
8	EZ	Equal to zero
9	EN	Equal to NaN
10	EI	Absolute value equal to infinity
11	ES	Equal to infinity or NaN
12	NZ	Not equal to zero
13	NN	Not equal to NaN
14	NI	Absolute value not equal to infinity
15	NS	Not equal to infinity and not equal to NaN

Notes:

If the cond comparison condition is EZ, EN, EI, ES, NZ, NN, NI, or NS, then only the matrix register specified by vs will be targeted for comparison. In this case, vt is a dummy and can be omitted. If vt is omitted, the assembler is able to avoid generating unintended RAW hazards by specifying in vt the same value as that in vs.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vt );
c[0] <- Compare( cond, s[0], t[0] );
c[4] <- c[0];
c[5] <- c[0];
WriteControlBit( VFPU_CC, 0, c[0] );
c[1] <- Compare( cond, s[1], t[1] );
c[4] <- c[4] | c[1];
c[5] <- c[5] & c[1];
WriteControlBit( VFPU_CC, 1, c[1] );
WriteControlBit( VFPU_CC, 4, c[4] );
WriteControlBit( VFPU_CC, 5, c[5] );
```

vcmp.t

Compare Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
0 1 1 0 1 1	0 0 0	vt	1	vs	0 0 0 0	cond
6	3	7	1	7	4	4

VFPU

Syntax:

```
vcmp.t  cond, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	No effect

Description:

Three elements from the matrix registers indicated by vs are compared with three elements from the matrix registers indicated by vt according to the condition cond. The elements are treated as floating-point numbers. The 6-bit comparison result is stored in control register VFPU_CC. The comparison result can be referenced by the bvt, bvf, bvtl, and bvfl conditional branching instructions, or by the vcmovt and vcmovf conditional assignment instructions.

The following mnemonics can be used for cond.

Code (cond)	Mnemonic	Function
0	FL	Always false
1	EQ	Equal
2	LT	Less than
3	LE	Less than or equal
4	TR	Always true
5	NE	Not equal
6	GE	Greater than or equal

Code (cond)	Mnemonic	Function
7	GT	Greater than
8	EZ	Equal to zero
9	EN	Equal to NaN
10	EI	Absolute value equal to infinity
11	ES	Equal to infinity or NaN
12	NZ	Not equal to zero
13	NN	Not equal to NaN
14	NI	Absolute value not equal to infinity
15	NS	Not equal to infinity and not equal to NaN

Notes:

If the cond comparison condition is EZ, EN, EI, ES, NZ, NN, NI, or NS, then only the matrix register specified by vs will be targeted for comparison. In this case, vt is a dummy and can be omitted. If vt is omitted, the assembler is able to avoid generating unintended RAW hazards by specifying in vt the same value as that in vs.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
c[0] <- Compare( cond, s[0], t[0] );
c[4] <- c[0];
c[5] <- c[0];
WriteControlBit( VFPU_CC, 0, c[0] );
c[1] <- Compare( cond, s[1], t[1] );
c[4] <- c[4] | c[1];
c[5] <- c[5] & c[1];
WriteControlBit( VFPU_CC, 1, c[1] );
c[2] <- Compare( cond, s[2], t[2] );
c[4] <- c[4] | c[2];
c[5] <- c[5] & c[2];
WriteControlBit( VFPU_CC, 2, c[2] );
WriteControlBit( VFPU_CC, 4, c[4] );
WriteControlBit( VFPU_CC, 5, c[5] );
```

vcmp.q

Compare Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
0 1 1 0 1 1	0 0 0	vt	1	vs	1 0 0 0	cond
6	3	7	1	7	4	4

VFPU

Syntax:

```
vcmp.q  cond, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	No effect

Description:

Four elements from the matrix registers indicated by vs are compared with four elements from the matrix registers indicated by vt according to the condition cond. The elements are treated as floating-point numbers. The 6-bit comparison result is stored in control register VFPU_CC. The comparison result can be referenced by the bvt, bvf, bvtl, and bvfl conditional branching instructions, or by the vcmovt and vcmovf conditional assignment instructions.
The following mnemonics can be used for cond.

Code (cond)	Mnemonic	Function
0	FL	Always false
1	EQ	Equal
2	LT	Less than
3	LE	Less than or equal
4	TR	Always true
5	NE	Not equal
6	GE	Greater than or equal

Code (cond)	Mnemonic	Function
7	GT	Greater than
8	EZ	Equal to zero
9	EN	Equal to NaN
10	EI	Absolute value equal to infinity
11	ES	Equal to infinity or NaN
12	NZ	Not equal to zero
13	NN	Not equal to NaN
14	NI	Absolute value not equal to infinity
15	NS	Not equal to infinity and not equal to NaN

Notes:

If the cond comparison condition is EZ, EN, EI, ES, NZ, NN, NI, or NS, then only the matrix register specified by vs will be targeted for comparison. In this case, vt is a dummy and can be omitted. If vt is omitted, the assembler is able to avoid generating unintended RAW hazards by specifying in vt the same value as that in vs.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
c[0] <- Compare( cond, s[0], t[0] );
c[4] <- c[0];
c[5] <- c[0];
WriteControlBit( VFPU_CC, 0, c[0] );
c[1] <- Compare( cond, s[1], t[1] );
c[4] <- c[4] | c[1];
c[5] <- c[5] & c[1];
WriteControlBit( VFPU_CC, 1, c[1] );
c[2] <- Compare( cond, s[2], t[2] );
c[4] <- c[4] | c[2];
c[5] <- c[5] & c[2];
WriteControlBit( VFPU_CC, 2, c[2] );
c[3] <- Compare( cond, s[3], t[3] );
c[4] <- c[4] | c[3];
c[5] <- c[5] & c[3];
WriteControlBit( VFPU_CC, 3, c[3] );
WriteControlBit( VFPU_CC, 4, c[4] );
WriteControlBit( VFPU_CC, 5, c[5] );
```

vcos.s

Cosine Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 1 1 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vcos.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The cosine of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \mathit{approx\_cos}\,(\mathit{M\_PI\_2} \times x) - \mathit{cos}(\mathit{M\_PI\_2} \times x) | < 4.1 \times 10^{-7}$; $0.0 <= | x | < 32.0$

$| \mathit{approx\_cos}\,(\mathit{M\_PI\_2} \times x) - \mathit{cos}(\mathit{M\_PI\_2} \times x) | < 2.0$; $32.0 <= | x | < \mathit{inf}$

Special solutions are as follows.

approx_cos(+***nan***) = +***nan***

approx_cos(-***nan***) = +***nan***

approx_cos(+***inf***) = +***nan***

approx_cos(-***inf***) = +***nan***

approx_cos(+0.0) = +1.0

approx_cos(-0.0) = +1.0

Notes:

The units of the angle which is provided as input for the vcos.s instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- approx_cos( M_PI_2 * s[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vcos.p

Cosine Pair Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14–8	7	6–0
1	1	0	1	0	0	0	0	0	0	0	1	0	0	1	1	0	vs	1	vd
6						11											7	1	7

VFPU

Syntax:

```
vcos.p  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The cosines of the floating-point values of two elements from the matrix registers indicated by vs are calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \textbf{\textit{approx\_cos}}\ (\textbf{\textit{M\_PI\_2}} \times \textbf{\textit{x}}) - \textbf{\textit{cos}}(\textbf{\textit{M\_PI\_2}} \times \textbf{\textit{x}}\ ) \ | < 4.1 \times 10^{-7}\ ;\ 0.0 <= | \textbf{\textit{x}} | < 32.0$

$| \textbf{\textit{approx\_cos}}\ (\textbf{\textit{M\_PI\_2}} \times \textbf{\textit{x}}) - \textbf{\textit{cos}}(\textbf{\textit{M\_PI\_2}} \times \textbf{\textit{x}}\ ) \ | < 2.0\ ;\ 32.0 <= | \textbf{\textit{x}} | < \textbf{\textit{inf}}$

Special solutions are as follows.

approx_cos(+***nan***) = +***nan***

approx_cos(-***nan***) = +***nan***

approx_cos(+***inf***) = +***nan***

approx_cos(-***inf***) = +***nan***

approx_cos(+0.0) = +1.0

approx_cos(-0.0) = +1.0

Notes:

The units of the angle which is provided as input for the vcos.p instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- approx_cos( M_PI_2 * s[0] );
d[1] <- approx_cos( M_PI_2 * s[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vcos.t

Cosine Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 1 1 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vcos.t  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The cosines of the floating-point values of three elements from the matrix registers indicated by vs are calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \boldsymbol{approx\_cos}\ (\boldsymbol{M\_PI\_2} \times \boldsymbol{x}) - \boldsymbol{cos}(\boldsymbol{M\_PI\_2} \times \boldsymbol{x}\ ) \mid < 4.1 \times 10^{-7}$; $0.0 <= | \boldsymbol{x} | < 32.0$

$| \boldsymbol{approx\_cos}\ (\boldsymbol{M\_PI\_2} \times \boldsymbol{x}) - \boldsymbol{cos}(\boldsymbol{M\_PI\_2} \times \boldsymbol{x}\ ) \mid < 2.0$; $32.0 <= | \boldsymbol{x} | < \boldsymbol{inf}$

Special solutions are as follows.

approx_cos(+***nan***) = +***nan***

approx_cos(-***nan***) = +***nan***

approx_cos(+***inf***) = +***nan***

approx_cos(-***inf***) = +***nan***

approx_cos(+0.0) = +1.0

approx_cos(-0.0) = +1.0

Notes:

The units of the angle which is provided as input for the vcos.t instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- approx_cos( M_PI_2 * s[0] );
d[1] <- approx_cos( M_PI_2 * s[1] );
d[2] <- approx_cos( M_PI_2 * s[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vcos.q

Cosine Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 1 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vcos.q  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10 pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The cosines of the floating-point values of four elements from the matrix registers indicated by vs are calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \textit{approx\_cos}\,(M\_PI\_2 \times x) - \cos(M\_PI\_2 \times x)\,| < 4.1 \times 10^{-7}\ ;\ 0.0 <= |\,x\,| < 32.0$

$| \textit{approx\_cos}\,(M\_PI\_2 \times x) - \cos(M\_PI\_2 \times x)\,| < 2.0\ ;\ 32.0 <= |\,x\,| < \textit{inf}$

Special solutions are as follows.

approx_cos(+***nan***) = +***nan***

approx_cos(-***nan***) = +***nan***

approx_cos(+***inf***) = +***nan***

approx_cos(-***inf***) = +***nan***

approx_cos(+0.0) = +1.0

approx_cos(-0.0) = +1.0

Notes:

The units of the angle which is provided as input for the vcos.q instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- approx_cos( M_PI_2 * s[0] );
d[1] <- approx_cos( M_PI_2 * s[1] );
d[2] <- approx_cos( M_PI_2 * s[2] );
d[3] <- approx_cos( M_PI_2 * s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vcrs.t

Cross Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	1 0 1	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vcrs.t  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5   pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Valid

Description:

The first part of the cross product between three elements from the matrix registers indicated by vs and three elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point numbers. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- s[1] * t[2];
d[1] <- s[2] * t[0];
d[2] <- s[0] * t[1];
WriteMatrix( TRIPLEWORD, vd, d );
```

vcrsp.t

Cross Product Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 0 1	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vcrsp.t  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The cross product between three elements from the matrix registers indicated by vs and three elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point numbers. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- + s[1]*t[2] - s[2]*t[1];
d[1] <- + s[2]*t[0] - s[0]*t[2];
d[2] <- + s[0]*t[1] - s[1]*t[0];
WriteMatrix( TRIPLEWORD, vd, d );
```

vcst.s

Set Constant Single Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 1	imm5	0 0 0 0 0 0 0 0 0	vd
6	5	5	9	7

VFPU

Syntax:

```
vcst.s  vd, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Valid

Description:

The constant indicated by imm5 is stored as a one-element floating-point value at the location in the matrix register file indicated by vd.
The table below shows the constants that can be specified for imm5 and their corresponding mnemonics.

Code (imm5)	Mnemonic	Value	Expression
0	-	Undefined	-
1	VFPU_HUGE	3.402823e+38	Maximum value that can be represented by a 32-bit single-precision floating-point number
2	VFPU_SQRT2	1.414214e+00	$\sqrt{2}$
3	VFPU_SQRT1_2	7.071068e-01	$\sqrt{\frac{1}{2}}$
4	VFPU_2_SQRTPI	1.128379e+00	$\frac{2}{\sqrt{\pi}}$
5	VFPU_2_PI	6.366197e-01	$\frac{2}{\pi}$
6	VFPU_1_PI	3.183099e-01	$\frac{1}{\pi}$
7	VFPU_PI_4	7.853982e-01	$\frac{\pi}{4}$

Code (imm5)	Mnemonic	Value	Expression
8	VFPU_PI_2	1.570796e+00	$\frac{\pi}{2}$
9	VFPU_PI	3.141593e+00	π
10	VFPU_E	2.718282e+00	e (base of natural logarithm)
11	VFPU_LOG2E	1.442695e+00	$\log_2 e$
12	VFPU_LOG10E	4.342945e-01	$\log_{10} e$
13	VFPU_LN2	6.931472e-01	ln2
14	VFPU_LN10	2.302585e+00	ln10
15	VFPU_2PI	6.283185e+00	2π
16	VFPU_PI_6	5.235988e-01	$\frac{\pi}{6}$
17	VFPU_LOG10TWO	3.010300e-01	$\log_{10} 2$
18	VFPU_LOG2TEN	3.321928e+00	$\log_2 10$
19	VFPU_SQRT3_2	8.660254e-01	$\frac{\sqrt{3}}{2}$
20	-	Undefined	-
21	-	Undefined	-
22	-	Undefined	-
23	-	Undefined	-
24	-	Undefined	-
25	-	Undefined	-
26	-	Undefined	-
27	-	Undefined	-
28	-	Undefined	-
29	-	Undefined	-
30	-	Undefined	-
31	-	Undefined	-

Operation:

```
d[0] <- const( imm5 );
WriteMatrix( SINGLEWORD, vd, d );
```

vcst.p

Set Constant Pair Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 1	imm5	0 0 0 0 0 0 0 0 1	vd
6	5	5	9	7

VFPU

Syntax:

```
vcst.p  vd, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Valid

Description:

The constant indicated by imm5 is stored as a two-element floating-point value at locations in the matrix register file indicated by vd.

The table below shows the constants that can be specified for imm5 and their corresponding mnemonics.

Code (imm5)	Mnemonic	Value	Expression
0	-	Undefined	-
1	VFPU_HUGE	3.402823e+38	Maximum value that can be represented by a 32-bit single-precision floating-point number
2	VFPU_SQRT2	1.414214e+00	$\sqrt{2}$
3	VFPU_SQRT1_2	7.071068e-01	$\sqrt{\frac{1}{2}}$
4	VFPU_2_SQRTPI	1.128379e+00	$\frac{2}{\sqrt{\pi}}$
5	VFPU_2_PI	6.366197e-01	$\frac{2}{\pi}$
6	VFPU_1_PI	3.183099e-01	$\frac{1}{\pi}$
7	VFPU_PI_4	7.853982e-01	$\frac{\pi}{4}$

Code (imm5)	Mnemonic	Value	Expression
8	VFPU_PI_2	1.570796e+00	$\frac{\pi}{2}$
9	VFPU_PI	3.141593e+00	π
10	VFPU_E	2.718282e+00	e (base of natural logarithm)
11	VFPU_LOG2E	1.442695e+00	$\log_2 e$
12	VFPU_LOG10E	4.342945e-01	$\log_{10} e$
13	VFPU_LN2	6.931472e-01	ln2
14	VFPU_LN10	2.302585e+00	ln10
15	VFPU_2PI	6.283185e+00	2π
16	VFPU_PI_6	5.235988e-01	$\frac{\pi}{6}$
17	VFPU_LOG10TWO	3.010300e-01	$\log_{10} 2$
18	VFPU_LOG2TEN	3.321928e+00	$\log_2 10$
19	VFPU_SQRT3_2	8.660254e-01	$\frac{\sqrt{3}}{2}$
20	-	Undefined	-
21	-	Undefined	-
22	-	Undefined	-
23	-	Undefined	-
24	-	Undefined	-
25	-	Undefined	-
26	-	Undefined	-
27	-	Undefined	-
28	-	Undefined	-
29	-	Undefined	-
30	-	Undefined	-
31	-	Undefined	-

Operation:

```
d[0] <- const( imm5 );
d[1] <- const( imm5 );
WriteMatrix( PAIRWORD, vd, d );
```

vcst.t

Set Constant Triple Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 1	imm5	1 0 0 0 0 0 0 0 0	vd
6	5	5	9	7

VFPU

Syntax:

```
vcst.t  vd, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Valid

Description:

The constant indicated by imm5 is stored as a three-element floating-point value at locations in the matrix register file indicated by vd.
The table below shows the constants that can be specified for imm5 and their corresponding mnemonics.

Code (imm5)	Mnemonic	Value	Expression
0	-	Undefined	-
1	VFPU_HUGE	3.402823e+38	Maximum value that can be represented by a 32-bit single-precision floating-point number
2	VFPU_SQRT2	1.414214e+00	$\sqrt{2}$
3	VFPU_SQRT1_2	7.071068e-01	$\sqrt{\frac{1}{2}}$
4	VFPU_2_SQRTPI	1.128379e+00	$\frac{2}{\sqrt{\pi}}$
5	VFPU_2_PI	6.366197e-01	$\frac{2}{\pi}$
6	VFPU_1_PI	3.183099e-01	$\frac{1}{\pi}$
7	VFPU_PI_4	7.853982e-01	$\frac{\pi}{4}$

Code (imm5)	Mnemonic	Value	Expression
8	VFPU_PI_2	1.570796e+00	$\frac{\pi}{2}$
9	VFPU_PI	3.141593e+00	π
10	VFPU_E	2.718282e+00	e (base of natural logarithm)
11	VFPU_LOG2E	1.442695e+00	$\log_2 e$
12	VFPU_LOG10E	4.342945e-01	$\log_{10} e$
13	VFPU_LN2	6.931472e-01	ln2
14	VFPU_LN10	2.302585e+00	ln10
15	VFPU_2PI	6.283185e+00	2π
16	VFPU_PI_6	5.235988e-01	$\frac{\pi}{6}$
17	VFPU_LOG10TWO	3.010300e-01	$\log_{10} 2$
18	VFPU_LOG2TEN	3.321928e+00	$\log_2 10$
19	VFPU_SQRT3_2	8.660254e-01	$\frac{\sqrt{3}}{2}$
20	-	Undefined	-
21	-	Undefined	-
22	-	Undefined	-
23	-	Undefined	-
24	-	Undefined	-
25	-	Undefined	-
26	-	Undefined	-
27	-	Undefined	-
28	-	Undefined	-
29	-	Undefined	-
30	-	Undefined	-
31	-	Undefined	-

Operation:

```
d[0] <- const( imm5 );
d[1] <- const( imm5 );
d[2] <- const( imm5 );
WriteMatrix( TRIPLEWORD, vd, d );
```

vcst.q

Set Constant Quad Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 1	imm5	1 0 0 0 0 0 0 0 1	vd
6	5	5	9	7

VFPU

Syntax:

```
vcst.q  vd, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Valid

Description:

The constant indicated by imm5 is stored as a four-element floating-point value at locations in the matrix register file indicated by vd.

The table below shows the constants that can be specified for imm5 and their corresponding mnemonics.

Code (imm5)	Mnemonic	Value	Expression
0	-	Undefined	-
1	VFPU_HUGE	3.402823e+38	Maximum value that can be represented by a 32-bit single-precision floating-point number
2	VFPU_SQRT2	1.414214e+00	$\sqrt{2}$
3	VFPU_SQRT1_2	7.071068e-01	$\sqrt{\frac{1}{2}}$
4	VFPU_2_SQRTPI	1.128379e+00	$\frac{2}{\sqrt{\pi}}$
5	VFPU_2_PI	6.366197e-01	$\frac{2}{\pi}$
6	VFPU_1_PI	3.183099e-01	$\frac{1}{\pi}$
7	VFPU_PI_4	7.853982e-01	$\frac{\pi}{4}$

Code (imm5)	Mnemonic	Value	Expression
8	VFPU_PI_2	1.570796e+00	$\frac{\pi}{2}$
9	VFPU_PI	3.141593e+00	π
10	VFPU_E	2.718282e+00	e (base of natural logarithm)
11	VFPU_LOG2E	1.442695e+00	$\log_2 e$
12	VFPU_LOG10E	4.342945e-01	$\log_{10} e$
13	VFPU_LN2	6.931472e-01	ln2
14	VFPU_LN10	2.302585e+00	ln10
15	VFPU_2PI	6.283185e+00	2π
16	VFPU_PI_6	5.235988e-01	$\frac{\pi}{6}$
17	VFPU_LOG10TWO	3.010300e-01	$\log_{10} 2$
18	VFPU_LOG2TEN	3.321928e+00	$\log_2 10$
19	VFPU_SQRT3_2	8.660254e-01	$\frac{\sqrt{3}}{2}$
20	-	Undefined	-
21	-	Undefined	-
22	-	Undefined	-
23	-	Undefined	-
24	-	Undefined	-
25	-	Undefined	-
26	-	Undefined	-
27	-	Undefined	-
28	-	Undefined	-
29	-	Undefined	-
30	-	Undefined	-
31	-	Undefined	-

Operation:

```
d[0] <- const( imm5 );
d[1] <- const( imm5 );
d[2] <- const( imm5 );
d[3] <- const( imm5 );
WriteMatrix( QUADWORD, vd, d );
```

vdet.p

2X2 Matrix Determinant

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	1 1 0	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vdet.p  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

Two elements from the matrix registers indicated by vs and two elements from the matrix registers indicated by vt are treated as elements of a 2x2 matrix. The elements are treated as floating-point numbers and the determinant of the matrix is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vt );
d[0] <- s[0] * t[1] - s[1] * t[0];
WriteMatrix( SINGLEWORD, vd, d );
```

vdiv.s

Divide Single Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 0	1 1 1	vt	0	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vdiv.s  vd, vs, vt
```

Instruction Type:

Multi-cycle instruction

Processing Time:

latency : 17   pitch : 14

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

One element from the matrix register indicated by vs is divided by one element from the matrix register indicated by vt. The elements are treated as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <- s[0] / t[0];
WriteMatrix( SINGLEWORD, vd, d );
```

vdot.p

Dot Product Pair Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	0 0 1	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vdot.p  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

The dot product of two elements from the matrix registers indicated by vs and two elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vt );
d[0] <- s[0] * t[0];
d[0] <- d[0] + s[1] * t[1];
WriteMatrix( SINGLEWORD, vd, d );
```

vdot.t

Dot Product Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	0 0 1	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vdot.t  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

The dot product of three elements from the matrix registers indicated by vs and three elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- s[0] * t[0];
d[0] <- d[0] + s[1] * t[1];
d[0] <- d[0] + s[2] * t[2];
WriteMatrix( SINGLEWORD, vd, d );
```

vdot.q

Dot Product Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	0 0 1	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vdot.q  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

The dot product of four elements from the matrix registers indicated by vs and four elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
d[0] <- s[0] * t[0];
d[0] <- d[0] + s[1] * t[1];
d[0] <- d[0] + s[2] * t[2];
d[0] <- d[0] + s[3] * t[3];
WriteMatrix( SINGLEWORD, vd, d );
```

vexp2.s

Exponential base 2 Single Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	vs	0	vd
6						11											7	1	7

VFPU

Syntax:

```
vexp2.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The base 2 exponential of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

The valid input range is $\boldsymbol{-inf} < \boldsymbol{x} < \boldsymbol{128}$

The precision of the calculation is given by the following expression.

$|\,(\boldsymbol{approx\_exp2}(\boldsymbol{x}) - (2^{x}))\,/\,(2^{x})\,| < 7.2 \times 10^{7}$; $0.0 <= \boldsymbol{x} < 128.0$

$|\,(\boldsymbol{approx\_exp2}(\boldsymbol{x}) - (2^{x}))\,/\,(2^{x})\,| < 7.2 \times 10^{7}$; $-64.0 < \boldsymbol{x} <= -0.0$

$|\,(\boldsymbol{approx\_exp2}(\boldsymbol{x}) - (2^{x}))\,/\,(2^{x})\,| <= 1.0$; $\boldsymbol{-inf} < \boldsymbol{x} <= -64.0$

Special solutions are as follows.

approx_exp2(+***nan***) = +***nan***

approx_exp2(-***nan***) = +***nan***

approx_exp2(+***inf***) = +***inf***

approx_exp2(-***inf***) = +0.0

approx_exp2(+0.0) = +1.0

approx_exp2(-0.0) = +1.0

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- approx_exp2( s[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vexp2.p

Exponential base 2 Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 0 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vexp2.p  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8　　　　pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The base 2 exponentials of the floating-point values of two elements from the matrix registers indicated by vs are calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The valid input range is $\boldsymbol{-inf} < \boldsymbol{x} < \boldsymbol{128}$

The precision of the calculation is given by the following expression.

$| (\boldsymbol{approx\_exp2}(\boldsymbol{x}) - (2^{x})) / (2^{x}) | < 7.2 \times 10^{7}$; $0.0 <= \boldsymbol{x} < 128.0$

$| (\boldsymbol{approx\_exp2}(\boldsymbol{x}) - (2^{x})) / (2^{x}) | < 7.2 \times 10^{7}$; $-64.0 < \boldsymbol{x} <= -0.0$

$| (\boldsymbol{approx\_exp2}(\boldsymbol{x}) - (2^{x})) / (2^{x}) | <= 1.0$; $\boldsymbol{-inf} < \boldsymbol{x} <= -64.0$

Special solutions are as follows.

approx_exp2(+***nan***) = +***nan***

approx_exp2(-***nan***) = +***nan***

approx_exp2(+***inf***) = +***inf***

approx_exp2(-***inf***) = +0.0

approx_exp2(+0.0) = +1.0

approx_exp2(-0.0) = +1.0

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- approx_exp2( s[0] );
d[1] <- approx_exp2( s[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vexp2.t

Exponential base 2 Triple Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	1	vs							0	vd						
6						11											7							1	7						

VFPU

Syntax:

```
vexp2.t  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9　　　　pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The base 2 exponentials of the floating-point values of three elements from the matrix registers indicated by vs are calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The valid input range is $\boldsymbol{-inf} < \boldsymbol{x} < \boldsymbol{128}$

The precision of the calculation is given by the following expression.

$|\,(\boldsymbol{approx\_exp2}(\boldsymbol{x}) - (2^x))\,/\,(2^x)\,| < 7.2 \times 10^7$; $0.0 <= \boldsymbol{x} < 128.0$

$|\,(\boldsymbol{approx\_exp2}(\boldsymbol{x}) - (2^x))\,/\,(2^x)\,| < 7.2 \times 10^7$; $-64.0 < \boldsymbol{x} <= -0.0$

$|\,(\boldsymbol{approx\_exp2}(\boldsymbol{x}) - (2^x))\,/\,(2^x)\,| <= 1.0$; $\boldsymbol{-inf} < \boldsymbol{x} <= -64.0$

Special solutions are as follows.

approx_exp2(+***nan***) = +***nan***

approx_exp2(-***nan***) = +***nan***

approx_exp2(+***inf***) = +***inf***

approx_exp2(-***inf***) = +0.0

approx_exp2(+0.0) = +1.0

approx_exp2(-0.0) = +1.0

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- approx_exp2( s[0] );
d[1] <- approx_exp2( s[1] );
d[2] <- approx_exp2( s[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vexp2.q

Exponential base 2 Quad Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	1	vs							1	vd						
6						11											7							1	7						

VFPU

Syntax:

```
vexp2.q  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10　　　pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The base 2 exponentials of the floating-point values of four elements from the matrix registers indicated by vs are calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The valid input range is ***-inf*** < ***x*** < ***128***

The precision of the calculation is given by the following expression.

$| (\boldsymbol{approx\_exp2}(\boldsymbol{x}) - (2^{x})) / (2^{x}) | < 7.2 \times 10^{7}$; $0.0 <= \boldsymbol{x} < 128.0$

$| (\boldsymbol{approx\_exp2}(\boldsymbol{x}) - (2^{x})) / (2^{x}) | < 7.2 \times 10^{7}$; $-64.0 < \boldsymbol{x} <= -0.0$

$| (\boldsymbol{approx\_exp2}(\boldsymbol{x}) - (2^{x})) / (2^{x}) | <= 1.0$; $\boldsymbol{-inf} < \boldsymbol{x} <= -64.0$

Special solutions are as follows.

approx_exp2(+***nan***) = +***nan***

approx_exp2(-***nan***) = +***nan***

approx_exp2(+***inf***) = +***inf***

approx_exp2(-***inf***) = +0.0

approx_exp2(+0.0) = +1.0

approx_exp2(-0.0) = +1.0

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- approx_exp2( s[0] );
d[1] <- approx_exp2( s[1] );
d[2] <- approx_exp2( s[2] );
d[3] <- approx_exp2( s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vf2h.p

Convert float to float16 Pair Word

31 … 26	25 … 15	14 … 8	7	6 … 0
1 1 0 1 0 0	0 0 0 0 1 1 0 0 1 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vf2h.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The single-precision floating-point values of two elements from the matrix registers indicated by vs are converted to half-precision floating-point numbers and packed into 32 bits. The 32-bit result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0].l <- float_to_float16( s[0] );
d[0].u <- float_to_float16( s[1] );
WriteMatrix( SINGLEWORD, vd, d );
```

vf2h.q

Convert float to float16 Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 0 0 1 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vf2h.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The single-precision floating-point values of four elements from the matrix registers indicated by vs are converted to half-precision floating-point numbers and packed into 64 bits. The 64-bit result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0].l <- float_to_float16( s[0] );
d[0].u <- float_to_float16( s[1] );
d[1].l <- float_to_float16( s[2] );
d[1].u <- float_to_float16( s[3] );
WriteMatrix( PAIRWORD, vd, d );
```

vf2id.s

Round to largest integer from float with Scaling Single Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 1 1	imm5	0	vs	0	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2id.s  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point value of one element from the matrix register indicated by vs is multiplied by 2 raised to the imm5 power and rounded to the largest integer less than or equal to the argument. The one-element integer result is stored at the location in the matrix register file indicated by vd. Special solutions are as follows.

floor(+***nan***) = 0x7FFFFFFF

floor(-***nan***) = 0x7FFFFFFF

floor(+***inf***) = 0x7FFFFFFF

floor(-***inf***) = 0x80000000

floor(***x***) = 0x7FFFFFFF ; $+2^{31} <= x < +inf$

floor(***x***) = 0x80000000 ; $-inf < x < -2^{31}$

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- floor( s[0] * (1<<imm5) );
WriteMatrix( SINGLEWORD, vd, d );
```

vf2id.p

Round to largest integer from float with Scaling Pair Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 1 1	imm5	0	vs	1	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2id.p  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point values of two elements from the matrix registers indicated by vs are multiplied by 2 raised to the imm5 power and rounded to the largest integer less than or equal to the arguments. The two-element integer result is stored at locations in the matrix register file indicated by vd. Special solutions are as follows.

floor(+***nan***) = 0x7FFFFFFF

floor(-***nan***) = 0x7FFFFFFF

floor(+***inf***) = 0x7FFFFFFF

floor(-***inf***) = 0x80000000

floor(***x***) = 0x7FFFFFFF ; $+2^{31} <= x < +inf$

floor(***x***) = 0x80000000 ; $-inf < x < -2^{31}$

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- floor( s[0] * (1<<imm5) );
d[1] <- floor( s[1] * (1<<imm5) );
```

```
        WriteMatrix( PAIRWORD, vd, d );
```

vf2id.t

Round to largest integer from float with Scaling Triple Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 1 1	imm5	1	vs	0	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2id.t  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point values of three elements from the matrix registers indicated by vs are multiplied by 2 raised to the imm5 power and rounded to the largest integer less than or equal to the arguments. The three-element integer result is stored at locations in the matrix register file indicated by vd. Special solutions are as follows.

floor(+***nan***) = 0x7FFFFFFF

floor(-***nan***) = 0x7FFFFFFF

floor(+***inf***) = 0x7FFFFFFF

floor(-***inf***) = 0x80000000

floor(***x***) = 0x7FFFFFFF ; $+2^{31} <= \boldsymbol{x} < +\boldsymbol{inf}$

floor(***x***) = 0x80000000 ; $-\boldsymbol{inf} < \boldsymbol{x} < -2^{31}$

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- floor( s[0] * (1<<imm5) );
d[1] <- floor( s[1] * (1<<imm5) );
```

```
        d[2] <- floor( s[2] * (1<<imm5) );
        WriteMatrix( TRIPLEWORD, vd, d );
```

vf2id.q

Round to largest integer from float with Scaling Quad Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 1 1	imm5	1	vs	1	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2id.q  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point values of four elements from the matrix registers indicated by vs are multiplied by 2 raised to the imm5 power and rounded to the largest integer less than or equal to the arguments. The four-element integer result is stored at locations in the matrix register file indicated by vd. Special solutions are as follows.

floor(+***nan***) = 0x7FFFFFFF

floor(-***nan***) = 0x7FFFFFFF

floor(+***inf***) = 0x7FFFFFFF

floor(-***inf***) = 0x80000000

floor(***x***) = 0x7FFFFFFF ; $+2^{31} <= x < +inf$

floor(***x***) = 0x80000000 ; $-inf < x < -2^{31}$

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- floor( s[0] * (1<<imm5) );
d[1] <- floor( s[1] * (1<<imm5) );
```

```
        d[2] <- floor( s[2] * (1<<imm5) );
        d[3] <- floor( s[3] * (1<<imm5) );
        WriteMatrix( QUADWORD, vd, d );
```

vf2in.s

Round to nearest integer from float with Scaling Single Word

31–26	25–21	20–16	15	14–8	7	6–0
1 1 0 1 0 0	1 0 0 0 0	imm5	0	vs	0	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2in.s  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point value of one element from the matrix register indicated by vs is multiplied by 2 raised to the imm5 power and rounded to the nearest integer. The one-element integer result is stored at the location in the matrix register file indicated by vd. Special solutions are as follows.

rint(+***nan***) = 0x7FFFFFFF

rint(-***nan***) = 0x7FFFFFFF

rint(+***inf***) = 0x7FFFFFFF

rint(-***inf***) = 0x80000000

rint(***x***) = 0x7FFFFFFF ; $+2^{31} <= x < +inf$

rint(***x***) = 0x80000000 ; $-inf < x < -2^{31}$

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- rint( s[0] * (1<<imm5) );
WriteMatrix( SINGLEWORD, vd, d );
```

vf2in.p

Round to nearest integer from float with Scaling Pair Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 0 0	imm5	0	vs	1	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2in.p  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point values of two elements from the matrix registers indicated by vs are multiplied by 2 raised to the imm5 power and rounded to the nearest integer. The two-element integer result is stored at locations in the matrix register file indicated by vd.

Special solutions are as follows.

rint(+***nan***) = 0x7FFFFFFF

rint(-***nan***) = 0x7FFFFFFF

rint(+***inf***) = 0x7FFFFFFF

rint(-***inf***) = 0x80000000

rint(***x***) = 0x7FFFFFFF ; $+2^{31} <= x < +inf$

rint(***x***) = 0x80000000 ; $-inf < x < -2^{31}$

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
```

```
        d[0] <- rint( s[0] * (1<<imm5) );
        d[1] <- rint( s[1] * (1<<imm5) );
        WriteMatrix( PAIRWORD, vd, d );
```

vf2in.t

Round to nearest integer from float with Scaling Triple Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 0 0	imm5	1	vs	0	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2in.t  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5   pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point values of three elements from the matrix registers indicated by vs are multiplied by 2 raised to the imm5 power and rounded to the nearest integer. The three-element integer result is stored at locations in the matrix register file indicated by vd.

Special solutions are as follows.

rint(+***nan***) = 0x7FFFFFFF

rint(-***nan***) = 0x7FFFFFFF

rint(+***inf***) = 0x7FFFFFFF

rint(-***inf***) = 0x80000000

rint(***x***) = 0x7FFFFFFF ; $+2^{31} <= \boldsymbol{x} < +\boldsymbol{inf}$

rint(***x***) = 0x80000000 ; $-\boldsymbol{inf} < \boldsymbol{x} < -2^{31}$

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
```

```
        d[0] <- rint( s[0] * (1<<imm5) );
        d[1] <- rint( s[1] * (1<<imm5) );
        d[2] <- rint( s[2] * (1<<imm5) );
        WriteMatrix( TRIPLEWORD, vd, d );
```

vf2in.q

Round to nearest integer from float with Scaling Quad Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 0 0	imm5	1	vs	1	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2in.q  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point values of four elements from the matrix registers indicated by vs are multiplied by 2 raised to the imm5 power and rounded to the nearest integer. The four-element integer result is stored at locations in the matrix register file indicated by vd.

Special solutions are as follows.

rint(+***nan***) = 0x7FFFFFFF

rint(-***nan***) = 0x7FFFFFFF

rint(+***inf***) = 0x7FFFFFFF

rint(-***inf***) = 0x80000000

rint(***x***) = 0x7FFFFFFF ; $+2^{31} <= \boldsymbol{x} < +\boldsymbol{inf}$

rint(***x***) = 0x80000000 ; $-\boldsymbol{inf} < \boldsymbol{x} < -2^{31}$

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
```

```
        d[0] <- rint( s[0] * (1<<imm5) );
        d[1] <- rint( s[1] * (1<<imm5) );
        d[2] <- rint( s[2] * (1<<imm5) );
        d[3] <- rint( s[3] * (1<<imm5) );
        WriteMatrix( QUADWORD, vd, d );
```

vf2iu.s

Round to smallest integer from float with Scaling Single Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 1 0	imm5	0	vs	0	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2iu.s  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point value of one element from the matrix register indicated by vs is multiplied by 2 raised to the imm5 power and rounded to the smallest integer greater than or equal to the argument. The one-element integer result is stored at the location in the matrix register file indicated by vd. Special solutions are as follows.

ceil(+***nan***) = 0x7FFFFFFF

ceil(-***nan***) = 0x7FFFFFFF

ceil(+***inf***) = 0x7FFFFFFF

ceil(-***inf***) = 0x80000000

ceil(***x***) = 0x7FFFFFFF ; $+2^{31} <= \boldsymbol{x} < +\boldsymbol{inf}$

ceil(***x***) = 0x80000000 ; $-\boldsymbol{inf} < \boldsymbol{x} < -2^{31}$

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- ceil( s[0] * (1<<imm5) );
WriteMatrix( SINGLEWORD, vd, d );
```

vf2iu.p

Round to smallest integer from float with Scaling Pair Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 1 0	imm5	0	vs	1	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2iu.p  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point values of two elements from the matrix registers indicated by vs are multiplied by 2 raised to the imm5 power and rounded to the smallest integers greater than or equal to the arguments. The two-element integer result is stored at locations in the matrix register file indicated by vd. Special solutions are as follows.

ceil(+***nan***) = 0x7FFFFFFF

ceil(-***nan***) = 0x7FFFFFFF

ceil(+***inf***) = 0x7FFFFFFF

ceil(-***inf***) = 0x80000000

ceil(***x***) = 0x7FFFFFFF ; $+2^{31} <= \boldsymbol{x} < +\boldsymbol{inf}$

ceil(***x***) = 0x80000000 ; $-\boldsymbol{inf} < \boldsymbol{x} < -2^{31}$

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- ceil( s[0] * (1<<imm5) );
d[1] <- ceil( s[1] * (1<<imm5) );
```

```
        WriteMatrix( PAIRWORD, vd, d );
```

vf2iu.t

Round to smallest integer from float with Scaling Triple Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 1 0	imm5	1	vs	0	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2iu.t  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point values of three elements from the matrix registers indicated by vs are multiplied by 2 raised to the imm5 power and rounded to the smallest integers greater than or equal to the arguments. The three-element integer result is stored at locations in the matrix register file indicated by vd. Special solutions are as follows.

ceil(+***nan***) = 0x7FFFFFFF

ceil(-***nan***) = 0x7FFFFFFF

ceil(+***inf***) = 0x7FFFFFFF

ceil(-***inf***) = 0x80000000

ceil(***x***) = 0x7FFFFFFF ; $+2^{31}$<= ***x*** < +***inf***

ceil(***x***) = 0x80000000 ; -***inf*** < ***x*** < -2^{31}

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- ceil( s[0] * (1<<imm5) );
d[1] <- ceil( s[1] * (1<<imm5) );
```

```
        d[2] <- ceil( s[2] * (1<<imm5) );
        WriteMatrix( TRIPLEWORD, vd, d );
```

vf2iu.q

Round to smallest integer from float with Scaling Quad Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 1 0	imm5	1	vs	1	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2iu.q  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point values of four elements from the matrix registers indicated by vs are multiplied by 2 raised to the imm5 power and rounded to the smallest integers greater than or equal to the arguments. The four-element integer result is stored at locations in the matrix register file indicated by vd. Special solutions are as follows.

ceil(+***nan***) = 0x7FFFFFFF

ceil(-***nan***) = 0x7FFFFFFF

ceil(+***inf***) = 0x7FFFFFFF

ceil(-***inf***) = 0x80000000

ceil(***x***) = 0x7FFFFFFF ; $+2^{31} <= \boldsymbol{x} < +\boldsymbol{inf}$

ceil(***x***) = 0x80000000 ; $-\boldsymbol{inf} < \boldsymbol{x} < -2^{31}$

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- ceil( s[0] * (1<<imm5) );
d[1] <- ceil( s[1] * (1<<imm5) );
```

```
        d[2] <- ceil( s[2] * (1<<imm5) );
        d[3] <- ceil( s[3] * (1<<imm5) );
        WriteMatrix( QUADWORD, vd, d );
```

vf2iz.s

Round to zero integer from float with Scaling Single Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 0 1	imm5	0	vs	0	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2iz.s  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point value of one element from the matrix register indicated by vs is multiplied by 2 raised to the imm5 power and rounded to the integer value closest to zero. The one-element integer result is stored at the location in the matrix register file indicated by vd. Special solutions are as follows.

trunc(+***nan***) = 0x7FFFFFFF

trunc(-***nan***) = 0x7FFFFFFF

trunc(+***inf***) = 0x7FFFFFFF

trunc(-***inf***) = 0x80000000

trunc(***x***) = 0x7FFFFFFF ; $+2^{31} <= x < +inf$

trunc(***x***) = 0x80000000 ; $-inf < x < -2^{31}$

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- trunc( s[0] * (1<<imm5) );
WriteMatrix( SINGLEWORD, vd, d );
```

vf2iz.p

Round to zero integer from float with Scaling Pair Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 0 1	imm5	0	vs	1	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2iz.p  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5   pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point values of two elements from the matrix registers indicated by vs are multiplied by 2 raised to the imm5 power and rounded to the integer values closest to zero. The two-element integer result is stored at locations in the matrix register file indicated by vd. Special solutions are as follows.

trunc(+***nan***) = 0x7FFFFFFF

trunc(-***nan***) = 0x7FFFFFFF

trunc(+***inf***) = 0x7FFFFFFF

trunc(-***inf***) = 0x80000000

trunc(***x***) = 0x7FFFFFFF ; $+2^{31} <= \boldsymbol{x} < +\boldsymbol{inf}$

trunc(***x***) = 0x80000000 ; $-\boldsymbol{inf} < \boldsymbol{x} < -2^{31}$

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- trunc( s[0] * (1<<imm5) );
d[1] <- trunc( s[1] * (1<<imm5) );
```

```
        WriteMatrix( PAIRWORD, vd, d );
```

vf2iz.t

Round to zero integer from float with Scaling Triple Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 0 1	imm5	1	vs	0	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2iz.t  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point values of three elements from the matrix registers indicated by vs are multiplied by 2 raised to the imm5 power and rounded to the integer values closest to zero. The three-element integer result is stored at locations in the matrix register file indicated by vd. Special solutions are as follows.

trunc(+***nan***) = 0x7FFFFFFF

trunc(-***nan***) = 0x7FFFFFFF

trunc(+***inf***) = 0x7FFFFFFF

trunc(-***inf***) = 0x80000000

trunc(***x***) = 0x7FFFFFFF ; $+2^{31} <= x < +inf$

trunc(***x***) = 0x80000000 ; $-inf < x < -2^{31}$

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- trunc( s[0] * (1<<imm5) );
d[1] <- trunc( s[1] * (1<<imm5) );
```

```
        d[2] <- trunc( s[2] * (1<<imm5) );
        WriteMatrix( TRIPLEWORD, vd, d );
```

vf2iz.q

Round to zero integer from float with Scaling Quad Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 0 0 1	imm5	1	vs	1	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vf2iz.q  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Only write mask is valid

Description:

The floating-point values of four elements from the matrix registers indicated by vs are multiplied by 2 raised to the imm5 power and rounded to the integer values closest to zero. The four-element integer result is stored at locations in the matrix register file indicated by vd. Special solutions are as follows.

trunc(+***nan***) = 0x7FFFFFFF

trunc(-***nan***) = 0x7FFFFFFF

trunc(+***inf***) = 0x7FFFFFFF

trunc(-***inf***) = 0x80000000

trunc(***x***) = 0x7FFFFFFF ; $+2^{31} <= x < +inf$

trunc(***x***) = 0x80000000 ; $-inf < x < -2^{31}$

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- trunc( s[0] * (1<<imm5) );
d[1] <- trunc( s[1] * (1<<imm5) );
```

```
        d[2] <- trunc( s[2] * (1<<imm5) );
        d[3] <- trunc( s[3] * (1<<imm5) );
        WriteMatrix( QUADWORD, vd, d );
```

vfad.p

Funnel Add Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 1 1 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vfad.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

Two elements from the matrix registers indicated by vs are added together as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- s[0];
d[0] <- d[0] + s[1];
WriteMatrix( SINGLEWORD, vd, d );
```

vfad.t

Funnel Add Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 1 1 0 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vfad.t  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

Three elements from the matrix registers indicated by vs are added together as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- s[0];
d[0] <- d[0] + s[1];
d[0] <- d[0] + s[2];
WriteMatrix( SINGLEWORD, vd, d );
```

vfad.q

Funnel Add Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 1 1 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vfad.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

Four elements from the matrix registers indicated by vs are added together as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- s[0];
d[0] <- d[0] + s[1];
d[0] <- d[0] + s[2];
d[0] <- d[0] + s[3];
WriteMatrix( SINGLEWORD, vd, d );
```

vfim.s

Convert float16 immediate to float Single Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
1 1 0 1 1 1	1 1 1	vd	imm16
6	3	7	16

VFPU

Syntax:

```
vfim.s  vd, imm16
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Valid

Description:

The half-precision floating-point value indicated by imm16 is converted to a single-precision floating-point value and stored at the location in the matrix register file indicated by vd.

Operation:

```
f <- float16_to_float( imm16 );
WriteMatrix( SINGLEWORD, vd, f );
```

vflush

Flush Write Buffer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	1
		6														26															

VFPU

Syntax:

```
vflush
```

Instruction Type:

Synchronization instruction

Processing Time:

latency : 0 pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

The write buffer is flushed. The VFPU pipeline stalls until the write buffer has emptied. If the instruction following the vflush is not a VFPU instruction, then additional vnop instructions must be inserted after the vflush so that the CPU pipeline will also be stalled.

There is an errata in which the data loaded by an lv.s or lv.q instruction that follows a vflush instruction may be incorrect. After a vflush instruction, always execute a vsync2 instruction before executing an lv.s or lv.q instruction.

Operation:

```
Flush();
```

vh2f.s

Convert float16 to float Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 0 0 1 1 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vh2f.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

The half-precision floating-point values of two elements from the matrix registers indicated by vs are converted to single-precision floating-point numbers. The two-element floating-point result is stored in locations of the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- float16_to_float( s[0].l );
d[1] <- float16_to_float( s[0].u );
WriteMatrix( PAIRWORD, vd, d );
```

vh2f.p

Convert float16 to float Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 0 0 1 1 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vh2f.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

The half-precision floating-point values of four elements from the matrix registers indicated by vs are converted to single-precision floating-point numbers. The four-element floating-point result is stored in locations of the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- float16_to_float( s[0].l );
d[1] <- float16_to_float( s[0].u );
d[2] <- float16_to_float( s[1].l );
d[3] <- float16_to_float( s[1].u );
WriteMatrix( QUADWORD, vd, d );
```

vhdp.p

Homogeneous Dot Product Pair Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	1 0 0	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vhdp.p  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Valid	Valid

Description:

The homogeneous dot product of two elements from the matrix registers indicated by vs and two elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vt );
d[0] <-        s[0] * t[0];
d[0] <- d[0] +        t[1];
WriteMatrix( SINGLEWORD, vd, d );
```

vhdp.t

Homogeneous Dot Product Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	1 0 0	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vhdp.t  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Valid	Valid

Description:

The homogeneous dot product of three elements from the matrix registers indicated by vs and three elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <-        s[0] * t[0];
d[0] <- d[0] + s[1] * t[1];
d[0] <- d[0] +        t[2];
WriteMatrix( SINGLEWORD, vd, d );
```

vhdp.q

Homogeneous Dot Product Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	1 0 0	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vhdp.q  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Valid	Valid

Description:

The homogeneous dot product of four elements from the matrix registers indicated by vs and four elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
d[0] <-        s[0] * t[0];
d[0] <- d[0] + s[1] * t[1];
d[0] <- d[0] + s[2] * t[2];
d[0] <- d[0] +        t[3];
WriteMatrix( SINGLEWORD, vd, d );
```

vhtfm2.p

Homogeneous Transform 2 Pair Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	0 0 1	vt	0	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vhtfm2.p  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The transform of the elements of the 2x2 matrix from the matrix registers indicated by vs and two elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point numbers. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

Due to a known problem with the VFPU, there is a restriction that register numbers $64 to $127 cannot be specified for vd. With psp-as 1.6.1 and later, if the registers in question are specified, the assembler will treat this as an error.

The registers which can be specified for vd are as follows.

```
   c?00, c?10, c?20, c?30
   r?00, r?10, r?20, r?30
```

The registers which cannot be specified for vd are as follows.

```
   c?02, c?12, c?22, c?32
   r?02, r?12, r?22, r?32
```

With the vhtfm2.p instruction, vs is used to specify a matrix, and vt is used to specify a vector. In the assembler, the pseudo-instructions vchtfm2.p and vrhtfm2.p are provided in order to program, with respect to either row vectors or column vectors, with an operand order which is the same as the calculation order. vrhtfm2.p switches vs and vt and then assembles the result as vhtfm2.p. As a result, the following can be written.

```
vchtfm2.p     c000, e100, c200     (equivalent to vhtfm2.p   c000, e100, c200)
vrhtfm2.p     r000, r100, m200     (equivalent to vhtfm2.p   r000, m200,
r100)
```

Operation:

```
s <- ReadMatrix(  PAIRXPAIRWORD, vs );
t <- ReadMatrix(  PAIRWORD, vt );
d[0] <- s[0] * t[0]  + s[4];
d[1] <- s[1] * t[0]  + s[5];
WriteMatrix( PAIRWORD, vd, d );
```

vhtfm3.t

Homogeneous Transform 3 Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	0 1 0	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vhtfm3.t  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The transform of the elements of the 3x3 matrix from the matrix registers indicated by vs and three elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point numbers. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

Due to a known problem with the VFPU, there is a restriction that register numbers $64 to $127 cannot be specified for vd. With psp-as 1.6.1 and later, if the registers in question are specified, the assembler will treat this as an error.

The registers which can be specified for vd are as follows.

```
   c?00, c?10, c?20, c?30
   r?00, r?10, r?20, r?30
```

The registers which cannot be specified for vd are as follows.

```
   c?01, c?11, c?21, c?31
   r?01, r?11, r?21, r?31
```

With the vhtfm3.p instruction, vs is used to specify a matrix, and vt is used to specify a vector. In the assembler, the pseudo-instructions vchtfm3.t and vrhtfm3.t are provided in order to program, with respect to either row vectors or column vectors, with an operand order which is the same as the calculation order. vrhtfm3.t switches vs and vt and then assembles as vhtfm3.t. As a result, the following can be written.

vchtfm3.t	c000, e100, c200	(equivalent to vhtfm3.t c000, e100, c200)
vrhtfm3.t	r000, r100, m200	(equivalent to vhtfm3.t r000, m200, r100)

Operation:

```
s <- ReadMatrix( TRIPLEXTRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- s[0] * t[0] + s[4] * t[1]  + s[8];
d[1] <- s[1] * t[0] + s[5] * t[1]  + s[9];
d[2] <- s[2] * t[0] + s[6] * t[1]  + s[10];
WriteMatrix( TRIPLEWORD, vd, d );
```

vhtfm4.q

Homogeneous Transform 4 Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	0 1 1	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vhtfm4.q  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10   pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The transform of the elements of the 4x4 matrix from the matrix registers indicated by vs and four elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point numbers. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

Due to a known problem with the VFPU, there is a restriction that register numbers $64 to $127 cannot be specified for vd. For psp-as 1.6.1 and later, if the registers in question are specified, the assembler will treat this as an error.

The registers which can be specified for vd are as follows.

```
c?00, c?10, c?20, c?30
r?00, r?10, r?20, r?30
```

The registers which cannot be specified for vd are as follows.

```
c?02, c?12, c?22, c?32
r?02, r?12, r?22, r?32
```

With the vhtfm4.q instruction, vs is used to specify a matrix, and vt is used to specify a vector. In the assembler, the pseudo-instructions vchtfm4.q and vrhtfm4.q are provided in order to program, with respect to either row vectors or column vectors, with an operand order which is the same as the calculation order. vrhtfm4.q switches vs and vt and then assembles as vhtfm4.q. As a result, the following can be written.

vchtfm4.q	c000, e100, c200	(equivalent to vhtfm4.q c000, e100, c200)
vrhtfm4.q	r000, r100, m200	(equivalent to vhtfm4.q r000, m200, r100)

Operation:

```
s <- ReadMatrix( QUADXQUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
d[0] <- s[0] * t[0] + s[4] * t[1] + s[8] * t[2]  + s[12];
d[1] <- s[1] * t[0] + s[5] * t[1] + s[9] * t[2]  + s[13];
d[2] <- s[2] * t[0] + s[6] * t[1] + s[10] * t[2]  + s[14];
d[3] <- s[3] * t[0] + s[7] * t[1] + s[11] * t[2]  + s[15];
WriteMatrix( QUADWORD, vd, d );
```

vi2c.q

Convert integer to signed char Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 1 1 0 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vi2c.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency：3   pitch：1

Prefixing:

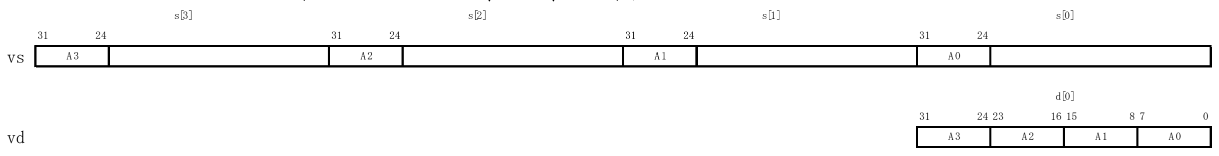
vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Only write mask is valid

Description:

The integer values of four elements from the matrix registers indicated by vs are converted to signed 8-bit integers and packed into 32 bits. The 32-bit result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
tmp[ 7: 0] <- s[0]>>24;
tmp[15: 8] <- s[1]>>24;
tmp[23:16] <- s[2]>>24;
tmp[31:24] <- s[3]>>24;
d[0] <- tmp;
WriteMatrix( SINGLEWORD, vd, d );
```



vi2f.s

Convert integer to float with Scaling Single Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 1 0 0	imm5	0	vs	0	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vi2f.s  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Valid

Description:

The integer value of one element from the matrix register indicated by vs is converted to a floating-point number and divided by 2 raised to the imm5 power. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- float( s[0] ) / (1<<imm5);
WriteMatrix( SINGLEWORD, vd, d );
```

vi2f.p

Convert integer to float with Scaling Pair Word

31–26	25–21	20–16	15	14–8	7	6–0
1 1 0 1 0 0	1 0 1 0 0	imm5	0	vs	1	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vi2f.p  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Valid

Description:

The integer values of two elements from the matrix registers indicated by vs are converted to floating-point numbers and divided by 2 raised to the imm5 power. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- float( s[0] ) / (1<<imm5);
d[1] <- float( s[1] ) / (1<<imm5);
WriteMatrix( PAIRWORD, vd, d );
```

vi2f.t

Convert integer to float with Scaling Triple Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 1 0 0	imm5	1	vs	0	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vi2f.t  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Valid

Description:

The integer values of three elements from the matrix registers indicated by vs are converted to floating-point numbers and divided by 2 raised to the imm5 power. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- float( s[0] ) / (1<<imm5);
d[1] <- float( s[1] ) / (1<<imm5);
d[2] <- float( s[2] ) / (1<<imm5);
WriteMatrix( TRIPLEWORD, vd, d );
```

vi2f.q

Convert integer to float with Scaling Quad Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	1 0 1 0 0	imm5	1	vs	1	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vi2f.q  vd, vs, imm5
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Valid

Description:

The integer values of four elements from the matrix registers indicated by vs are converted to floating-point numbers and divided by 2 raised to the imm5 power. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- float( s[0] ) / (1<<imm5);
d[1] <- float( s[1] ) / (1<<imm5);
d[2] <- float( s[2] ) / (1<<imm5);
d[3] <- float( s[3] ) / (1<<imm5);
WriteMatrix( QUADWORD, vd, d );
```

vi2s.p

Convert integer to signed short Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 1 1 1 1 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vi2s.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

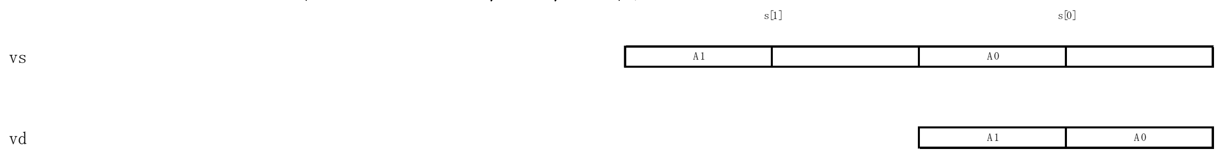
vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Only write mask is valid

Description:

The integer values of two elements from the matrix registers indicated by vs are converted to signed 16-bit integers and packed into 32 bits. The 32-bit result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
tmp0[15: 0] <- s[0]>>16;
tmp0[31:16] <- s[1]>>16;
d[0] <- tmp0;
WriteMatrix( SINGLEWORD, vd, d );
```



vi2s.q

Convert integer to signed short Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 1 1 1 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vi2s.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

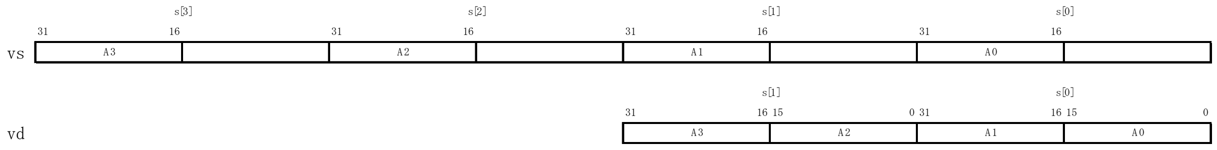
vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Only write mask is valid

Description:

The integer values of four elements from the matrix registers indicated by vs are converted to signed 16-bit integers and packed into 64 bits. The 64-bit result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
tmp0[15: 0] <- s[0]>>16;
tmp0[31:16] <- s[1]>>16;
tmp1[15: 0] <- s[2]>>16;
tmp1[31:16] <- s[3]>>16;
d[0] <- tmp0;
d[1] <- tmp1;
WriteMatrix( PAIRWORD, vd, d );
```



vi2uc.q

Convert integer to unsigned char Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 1 1 0 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vi2uc.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Only write mask is valid

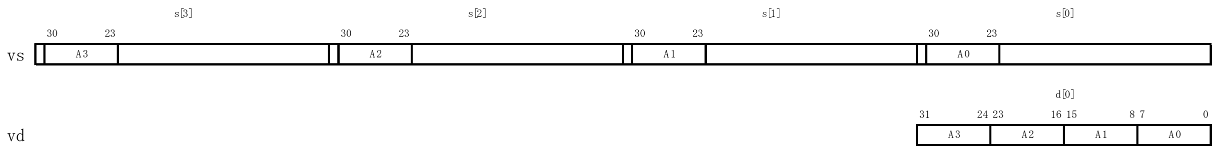
Description:

The integer values of four elements from the matrix registers indicated by vs are converted to unsigned 8-bit integers and packed into 32 bits. The 32-bit result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
if( s[0] < 0 )
    tmp[ 7: 0] <- 0;
else
    tmp[ 7: 0] <- s[0]>>23;
if( s[1] < 0 )
    tmp[15: 8] <- 0;
else
    tmp[15: 8] <- s[1]>>23;
if( s[2] < 0 )
    tmp[23:16] <- 0;
else
    tmp[23:16] <- s[2]>>23;
if( s[3] < 0 )
```

```
            tmp[31:24] <- 0;
        else
            tmp[31:24] <- s[3]>>23;
        d[0] <- tmp;
        WriteMatrix( SINGLEWORD, vd, d );
```



vi2us.p

Convert integer to unsigned short Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 1 1 1 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vi2us.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

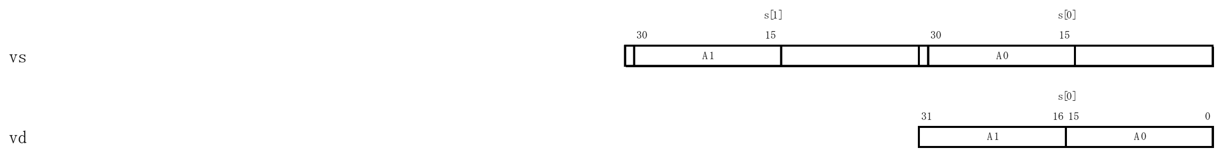
vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Only write mask is valid

Description:

The integer values of two elements from the matrix registers indicated by vs are converted to unsigned 16-bit integers and packed into 32 bits. The 32-bit result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
if( s[0] < 0 )
    tmp0[15: 0] <- 0;
else
    tmp0[15: 0] <- s[0]>>15;
if( s[1] < 0 )
    tmp0[31:16] <- 0;
else
    tmp0[31:16] <- s[1]>>15;
d[0] <- tmp0;
WriteMatrix( SINGLEWORD, vd, d );
```


s[1]
s[0]
30
15
30
15
vs
A1
A0
s[0]
31
16 15
0
vd
A1
A0

vi2us.q

Convert integer to unsigned short Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 1 1 1 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vi2us.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Only write mask is valid

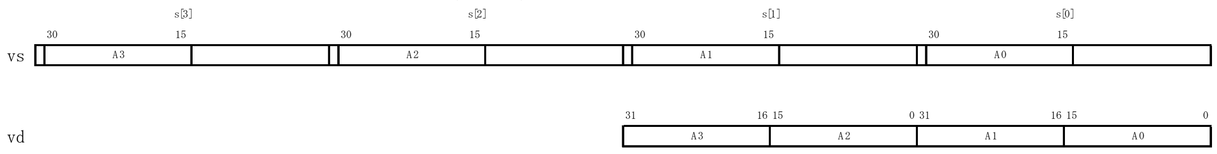
Description:

The integer values of four elements from the matrix registers indicated by vs are converted to unsigned 16-bit integers and packed into 64 bits. The 64-bit result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
if( s[0] < 0 )
    tmp0[15: 0] <- 0;
else
    tmp0[15: 0] <- s[0]>>15;
if( s[1] < 0 )
    tmp0[31:16] <- 0;
else
    tmp0[31:16] <- s[1]>>15;
if( s[2] < 0 )
    tmp1[15: 0] <- 0;
else
    tmp1[15: 0] <- s[2]>>15;
if( s[3] < 0 )
```

```
            tmp1[31:16] <- 0;
        else
            tmp1[31:16] <- s[3]>>15;
        d[0] <- tmp0;
        d[1] <- tmp1;
        WriteMatrix( PAIRWORD, vd, d );
```



vidt.p

Identity Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1	vd
6	19	7

VFPU

Syntax:

```
vidt.p  vd
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

One vector from the identity matrix is stored as a two-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
d[0] <- (vd[0]==0) ? 1.0 : 0.0;
d[1] <- (vd[0]==1) ? 1.0 : 0.0;
WriteMatrix( PAIRWORD, vd, d );
```

vidt.q

Identity Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1	vd
6	19	7

VFPU

Syntax:

```
vidt.q  vd
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

One vector from the identity matrix is stored as a four-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
d[0] <- (vd[1:0]==0) ? 1.0 : 0.0;
d[1] <- (vd[1:0]==1) ? 1.0 : 0.0;
d[2] <- (vd[1:0]==2) ? 1.0 : 0.0;
d[3] <- (vd[1:0]==3) ? 1.0 : 0.0;
WriteMatrix( QUADWORD, vd, d );
```

viim.s

Convert integer immediate to float Single Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
1 1 0 1 1 1	1 1 0	vd	imm16
6	3	7	16

VFPU

Syntax:

```
viim.s  vd, imm16
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Valid

Description:

The integer indicated by imm16 is converted to a floating-point number and stored at the location in the matrix register file indicated by vd.

Operation:

```
f <- float( imm16 );
WriteMatrix( SINGLEWORD, vd, f );
```

vlgb.s

LogB Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 0 1 1 1 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vlgb.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The logB of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

logB is defined by the following expression.

$\boldsymbol{x} = \boldsymbol{scaleBZ}(\boldsymbol{x}) * 2^{\boldsymbol{logB}(\boldsymbol{x})}$; $1 <= \boldsymbol{scaleBZ}(\boldsymbol{x}) < 2$.

Special solutions are as follows.

logB(***nan***) = ***nan***

logB(+***inf***) = +***inf***

logB(-***inf***) = -***inf***

logB(+0.0) = -***inf***

logB(-0.0) = -***inf***

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
```

```
        d[0] <- logB( |s[0]| );
        WriteMatrix( SINGLEWORD, vd, d );
```

vlog2.s

Logarithm base 2 Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 0 1 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vlog2.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The base 2 logarithm of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \boldsymbol{approx\_log2}(\boldsymbol{x}) - \boldsymbol{log2}(\boldsymbol{x}) | < 3.1 \times 10^{-5}$

Special solutions are as follows.

approx_log2(+***nan***) = +***nan***

approx_log2(-***nan***) = +***nan***

approx_log2(+***inf***) = +***inf***

approx_log2(-***inf***) = +***nan***

approx_log2(+0.0) = -***inf***

approx_log2(-0.0) = -***inf***

approx_log2(***x***) = +***nan*** ; -***inf*** < ***x*** < -0.0

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- approx_log2( s[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vlog2.p

Logarithm base 2 Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 0 1 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vlog2.p  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The base 2 logarithms of the floating-point values of two elements from the matrix registers indicated by vs are calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$|\ \boldsymbol{approx\_log2(x)} - \boldsymbol{log2(x)}\ | < 3.1 \times 10^{-5}$

Special solutions are as follows.

$\boldsymbol{approx\_log2}(+\boldsymbol{nan}) = +\boldsymbol{nan}$

$\boldsymbol{approx\_log2}(-\boldsymbol{nan}) = +\boldsymbol{nan}$

$\boldsymbol{approx\_log2}(+\boldsymbol{inf}) = +\boldsymbol{inf}$

$\boldsymbol{approx\_log2}(-\boldsymbol{inf}) = +\boldsymbol{nan}$

$\boldsymbol{approx\_log2}(+0.0) = -\boldsymbol{inf}$

$\boldsymbol{approx\_log2}(-0.0) = -\boldsymbol{inf}$

$\boldsymbol{approx\_log2}(\boldsymbol{x}) = +\boldsymbol{nan}\ ;\ -\boldsymbol{inf} < \boldsymbol{x} < -0.0$

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- approx_log2( s[0] );
d[1] <- approx_log2( s[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vlog2.t

Logarithm base 2 Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 0 1 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vlog2.t  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The base 2 logarithms of the floating-point values of three elements from the matrix registers indicated by vs are calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$|\ \boldsymbol{approx\_log2}(\boldsymbol{x}) - \boldsymbol{log2}(\boldsymbol{x})\ | < 3.1 \times 10^{-5}$

Special solutions are as follows.

$\boldsymbol{approx\_log2}(+\boldsymbol{nan}) = +\boldsymbol{nan}$

$\boldsymbol{approx\_log2}(-\boldsymbol{nan}) = +\boldsymbol{nan}$

$\boldsymbol{approx\_log2}(+\boldsymbol{inf}) = +\boldsymbol{inf}$

$\boldsymbol{approx\_log2}(-\boldsymbol{inf}) = +\boldsymbol{nan}$

$\boldsymbol{approx\_log2}(+0.0) = -\boldsymbol{inf}$

$\boldsymbol{approx\_log2}(-0.0) = -\boldsymbol{inf}$

$\boldsymbol{approx\_log2}(\boldsymbol{x}) = +\boldsymbol{nan}\ ;\ -\boldsymbol{inf} < \boldsymbol{x} < -0.0$

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- approx_log2( s[0] );
d[1] <- approx_log2( s[1] );
d[2] <- approx_log2( s[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vlog2.q

Logarithm base 2 Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 0 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vlog2.q  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10   pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The base 2 logarithms of the floating-point values of four elements from the matrix registers indicated by vs are calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \textbf{\textit{approx\_log2}}(\textbf{\textit{x}}) - \textbf{\textit{log2}}(\textbf{\textit{x}}) | < 3.1 \times 10^{-5}$

Special solutions are as follows.

approx_log2(+***nan***) = +***nan***

approx_log2(-***nan***) = +***nan***

approx_log2(+***inf***) = +***inf***

approx_log2(-***inf***) = +***nan***

approx_log2(+0.0) = -***inf***

approx_log2(-0.0) = -***inf***

approx_log2(***x***) = +***nan*** ; -***inf*** < ***x*** < -0.0

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- approx_log2( s[0] );
d[1] <- approx_log2( s[1] );
d[2] <- approx_log2( s[2] );
d[3] <- approx_log2( s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vmax.s

Maximum Single Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	0 1 1	vt	0	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmax.s  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

One element from the matrix register indicated by vs is compared with one element from the matrix register indicated by vt and the largest element is selected. The elements are treated as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Notes:

For the vmax.s instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-*nan* < -*inf* < -*num* < -0.0 = +0.0 < +*num* < +*inf* < +*nan*

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-inf	***-inf***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-1.0	***-1.0***	***-1.0***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-0.0	***-0.0***	***-0.0***	***-0.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
+0.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
+1.0	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+inf***	***+nan***
+inf	***+inf***	***+inf***	***+inf***	***+inf***	***+inf***	***+inf***	***+inf***	***+nan***
+nan	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <- max( s[0] , t[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vmax.p

Maximum Pair Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	0 1 1	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmax.p  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Two elements from the matrix registers indicated by vs are compared with the corresponding two elements from the matrix registers indicated by vt and the largest two elements from the individual comparisons are selected. The elements are treated as floating-point numbers. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vmax.p instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.

-nan < ***-inf*** < ***-num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-inf	***-inf***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-1.0	***-1.0***	***-1.0***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-0.0	***-0.0***	***-0.0***	***-0.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
+0.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
+1.0	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+inf***	***+nan***
+inf	***+inf***	***+inf***	***+inf***	***+inf***	***+inf***	***+inf***	***+inf***	***+nan***
+nan	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vt );
d[0] <- max( s[0] , t[0] );
d[1] <- max( s[1] , t[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vmax.t

Maximum Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	0 1 1	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmax.t  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Three elements from the matrix registers indicated by vs are compared with the corresponding three elements from the matrix registers indicated by vt and the largest three elements from the individual comparisons are selected. The elements are treated as floating-point numbers. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vmax.t instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-inf	***-inf***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-1.0	***-1.0***	***-1.0***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-0.0	***-0.0***	***-0.0***	***-0.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
+0.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
+1.0	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+inf***	***+nan***
+inf	***+inf***	***+inf***	***+inf***	***+inf***	***+inf***	***+inf***	***+inf***	***+nan***
+nan	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- max( s[0] , t[0] );
d[1] <- max( s[1] , t[1] );
d[2] <- max( s[2] , t[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vmax.q

Maximum Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	0 1 1	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmax.q  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Four elements from the matrix registers indicated by vs are compared with the corresponding four elements from the matrix registers indicated by vt and the largest four elements from the individual comparisons are selected. The elements are treated as floating-point numbers. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vmax.q instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-inf	***-inf***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-1.0	***-1.0***	***-1.0***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-0.0	***-0.0***	***-0.0***	***-0.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
+0.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
+1.0	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+inf***	***+nan***
+inf	***+inf***	***+inf***	***+inf***	***+inf***	***+inf***	***+inf***	***+inf***	***+nan***
+nan	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***	***+nan***

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
d[0] <- max( s[0] , t[0] );
d[1] <- max( s[1] , t[1] );
d[2] <- max( s[2] , t[2] );
d[3] <- max( s[3] , t[3] );
WriteMatrix( QUADWORD, vd, d );
```

vmfvc

VFPU Move Word from VFPU Control

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 1 0 0 0 0	imm8	0	vd
6	10	8	1	7

VFPU

Syntax:

```
vmfvc  vd, imm8
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3   pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

The contents of the VFPU control register indicated by the imm8 field are stored as a one-element floating-point value at the location in the matrix register file indicated by vd.

Notes:

When reading control registers, the pipeline will not interlock because of a RAW hazard. If the preceding instruction performs a rewrite to a control register, then one latency cycle’s worth of vnop instructions must be inserted before the vmfvc instruction.
In these situations, note that the assembler will not automatically insert the vnop instructions.

Operation:

```
dataword <- ReadControl( imm8 );
```

```
        WriteMatrix( SINGLEWORD, vd, dataword );
```

vmidt.p

Identity Pair x Pair Matrix

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1	vd
6	19	7

VFPU

Syntax:

```
vmidt.p  vd
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 4 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The 2x2 identity matrix is generated and stored at locations in the matrix register file indicated by vd. Elements are stored as floating-point values.

Operation:

```
d[0]  <- 1.0;
d[1]  <- 0.0;
d[4]  <- 0.0;
d[5]  <- 1.0;
WriteMatrix( PAIRXPAIRWORD, vd, d );
```

vmidt.t

Identity Triple x Triple Matrix

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0	vd
6	19	7

VFPU

Syntax:

```
vmidt.t  vd
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 5　　　　pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The 3x3 identity matrix is generated and stored at locations in the matrix register file indicated by vd. Elements are stored as floating-point values.

Operation:

```
d[0]  <- 1.0;
d[1]  <- 0.0;
d[2]  <- 0.0;
d[4]  <- 0.0;
d[5]  <- 1.0;
d[6]  <- 0.0;
d[8]  <- 0.0;
d[9]  <- 0.0;
d[10] <- 1.0;
WriteMatrix( TRIPLEXTRIPLEWORD, vd, d );
```

vmidt.q

Identity Quad x Quad Matrix

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	1	vd						
6						19																			7						

VFPU

Syntax:

```
vmidt.q  vd
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 6 pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The 4x4 identity matrix is generated and stored at locations in the matrix register file indicated by vd. Elements are stored as floating-point values.

Operation:

```
d[0]  <- 1.0;
d[1]  <- 0.0;
d[2]  <- 0.0;
d[3]  <- 0.0;
d[4]  <- 0.0;
d[5]  <- 1.0;
d[6]  <- 0.0;
d[7]  <- 0.0;
d[8]  <- 0.0;
d[9]  <- 0.0;
d[10] <- 1.0;
d[11] <- 0.0;
d[12] <- 0.0;
d[13] <- 0.0;
d[14] <- 0.0;
d[15] <- 1.0;
```

```
        WriteMatrix( QUADXQUADWORD, vd, d );
```

vmin.s

Minimum Single Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	0 1 0	vt	0	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmin.s  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

One element from the matrix register indicated by vs is compared with one element from the matrix register indicated by vt and the smallest element is selected. The elements are treated as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Notes:

For the vmin.s instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***
-inf	***-nan***	***-inf***	***-inf***	***-inf***	***-inf***	***-inf***	***-inf***	***-inf***
-1.0	***-nan***	***-inf***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-0.0	***-nan***	***-inf***	***-1.0***	***-0.0***	***-0.0***	***-0.0***	***-0.0***	***-0.0***
+0.0	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
+1.0	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+1.0***	***+1.0***
+inf	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+inf***
+nan	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <- min( s[0] , t[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vmin.p

Minimum Pair Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	0 1 0	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmin.p  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Two elements from the matrix registers indicated by vs are compared with the corresponding two elements from the matrix registers indicated by vt and the smallest elements from the individual comparisons are selected. The elements are treated as floating-point numbers. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vmin.p instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***
-inf	***-nan***	***-inf***	***-inf***	***-inf***	***-inf***	***-inf***	***-inf***	***-inf***
-1.0	***-nan***	***-inf***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-0.0	***-nan***	***-inf***	***-1.0***	***-0.0***	***-0.0***	***-0.0***	***-0.0***	***-0.0***
+0.0	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
+1.0	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+1.0***	***+1.0***
+inf	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+inf***
+nan	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vt );
d[0] <- min( s[0] , t[0] );
d[1] <- min( s[1] , t[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vmin.t

Minimum Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	0 1 0	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmin.t  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Three elements from the matrix registers indicated by vs are compared with the corresponding three elements from the matrix registers indicated by vt and the smallest elements from the individual comparisons are selected. The elements are treated as floating-point numbers. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vmin.t instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***
-inf	***-nan***	***-inf***	***-inf***	***-inf***	***-inf***	***-inf***	***-inf***	***-inf***
-1.0	***-nan***	***-inf***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-0.0	***-nan***	***-inf***	***-1.0***	***-0.0***	***-0.0***	***-0.0***	***-0.0***	***-0.0***
+0.0	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
+1.0	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+1.0***	***+1.0***
+inf	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+inf***
+nan	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- min( s[0] , t[0] );
d[1] <- min( s[1] , t[1] );
d[2] <- min( s[2] , t[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vmin.q

Minimum Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	0 1 0	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmin.q  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Four elements from the matrix registers indicated by vs are compared with the corresponding four elements from the matrix registers indicated by vt and the smallest elements from the individual comparisons are selected. The elements are treated as floating-point numbers. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vmin.q instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***	***-nan***
-inf	***-nan***	***-inf***	***-inf***	***-inf***	***-inf***	***-inf***	***-inf***	***-inf***
-1.0	***-nan***	***-inf***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-0.0	***-nan***	***-inf***	***-1.0***	***-0.0***	***-0.0***	***-0.0***	***-0.0***	***-0.0***
+0.0	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
+1.0	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+1.0***	***+1.0***
+inf	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+inf***
+nan	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
d[0] <- min( s[0] , t[0] );
d[1] <- min( s[1] , t[1] );
d[2] <- min( s[2] , t[2] );
d[3] <- min( s[3] , t[3] );
WriteMatrix( QUADWORD, vd, d );
```

vmmov.p

Move Pair x Pair Matrix

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 1 1 0 0 0 0 0 0 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vmmov.p  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 4 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The floating-point values of the elements of the 2x2 matrix from the matrix registers indicated by vs are copied to locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRXPAIRWORD, vs );
d[0]  <- s[0];
d[1]  <- s[1];
d[4]  <- s[4];
d[5]  <- s[5];
WriteMatrix( PAIRXPAIRWORD, vd, d );
```

vmmov.t

Move Triple x Triple Matrix

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 1 1 0 0 0 0 0 0 0 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vmmov.t  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 5 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The floating-point values of the elements of the 3x3 matrix from the matrix registers indicated by vs are copied to locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEXTRIPLEWORD, vs );
d[0]  <- s[0];
d[1]  <- s[1];
d[2]  <- s[2];
d[4]  <- s[4];
d[5]  <- s[5];
d[6]  <- s[6];
d[8]  <- s[8];
d[9]  <- s[9];
d[10] <- s[10];
WriteMatrix( TRIPLEXTRIPLEWORD, vd, d );
```

vmmov.q

Move Quad x Quad Matrix

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 1 1 0 0 0 0 0 0 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vmmov.q  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 6 pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The floating-point values of the elements of the 4x4 matrix from the matrix registers indicated by vs are copied to locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADXQUADWORD, vs );
d[0]  <- s[0];
d[1]  <- s[1];
d[2]  <- s[2];
d[3]  <- s[3];
d[4]  <- s[4];
d[5]  <- s[5];
d[6]  <- s[6];
d[7]  <- s[7];
d[8]  <- s[8];
d[9]  <- s[9];
d[10] <- s[10];
d[11] <- s[11];
d[12] <- s[12];
d[13] <- s[13];
d[14] <- s[14];
```

```
        d[15] <- s[15];
        WriteMatrix( QUADXQUADWORD, vd, d );
```

vmmul.p

Multiply Pair x Pair Matrix

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	0 0 0	vt	0	vs^t	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmmul.p  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10 pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

Matrix multiplication is performed on the 2x2 floating-point elements of the matrix register specified by vs and the 2x2 floating-point elements of the matrix register specified by vt, and the resultant 2x2 floating-point elements are stored at the location in the matrix register file specified by vd.

Notes:

In the opcode for the vmmul.p instruction, please note that the field which corresponds to the operand vs is written as vs^t which is the transpose of the matrix register. The assembler handles the vs operand of the vmmul.p instruction as an exceptional case by inverting the RXC bit (bit 13 of the opcode), and swapping the high-order bit of the IDX field (bit 9 of the opcode) with the FSL field (bit 14 of the opcode). As a result, the operation of multiplying the row-vector-format 2D matrix m100 by m200 from the right, can be written as follows.

```
vmmul.p        m000, m100, m200
```

In contrast, for column-vector-format matrices, a transposition is performed. To perform an operation in which the column-vector-format 2D matrix e100 is multiplied by e200 from the right, the following must be written.

vmmul.p e000, e200, e100

In the assembler, the pseudo-instructions vcmmul.p and vrmmul.p are provided in order to program, with respect to either row-vector-format matrices or column-vector-format matrices, with an operand order which is the same as the calculation order. vcmmul.p switches vs and vt and then assembles as vmmul.p. As a result, the following can be written.

vcmmul.p e000, e100, e200 (equivalent to vmmul.p e000, e200, e100)

vrmmul.p m000, m100, m200 (equivalent to vmmul.p m000, m100, m200)

Operation:

```
s <- ReadMatrix( PAIRXPAIRWORD,vs^t );
t <- ReadMatrix( PAIRXPAIRWORD,vt );
d[ 0] <- s[ 0]*t[ 0] + s[ 4]*t[ 4];
d[ 1] <- s[ 0]*t[ 1] + s[ 4]*t[ 5];
d[ 4] <- s[ 1]*t[ 0] + s[ 5]*t[ 4];
d[ 5] <- s[ 1]*t[ 1] + s[ 5]*t[ 5];
WriteMatrix( PAIRXPAIRWORD,vd, d );

or

s <- ReadMatrix( PAIRXPAIRWORD,vs );
t <- ReadMatrix( PAIRXPAIRWORD,vt );
d[ 0] <- s[ 0]*t[ 0] + s[ 1]*t[ 4];
d[ 1] <- s[ 0]*t[ 1] + s[ 1]*t[ 5];
d[ 4] <- s[ 4]*t[ 0] + s[ 5]*t[ 4];
d[ 5] <- s[ 4]*t[ 1] + s[ 5]*t[ 5];
WriteMatrix( PAIRXPAIRWORD,vd, d );
```

vmmul.t

Multiply Triple x Triple Matrix

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	0 0 0	vt	1	vs^t	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmmul.t  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 15 pitch : 9

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

Matrix multiplication is performed on the 3x3 floating-point elements of the matrix register specified by vs and the 3x3 floating-point elements of the matrix register specified by vt, and the resultant 3x3 floating-point elements are stored in the location of the matrix register specified by vd.

Notes:

In the opcode for the vmmul.t instruction, please note that the field which corresponds to the operand vs is written as vs^t which is an inverted expression of the matrix register. The assembler handles the vs operand of the vmmul.t instruction as an exceptional case by inverting the RXC bit (bit 13 of the opcode), and swapping the low-order bit of the IDX field (bit 8 of the opcode) with the FSL field (bit 14 of the opcode). As a result, the operation of multiplying the row-vector-format 3D matrix m100 by m200 from the right, can be written as follows.

```
vmmul.t        m000, m100, m200
```

In contrast, for column-vector-format matrices, a transposition is performed. To perform an operation in which the column-vector-format 3D matrix e100 is multiplied by e200 from the right, the following must be written.

```
vmmul.t         e000, e200, e100
```

For psp-as 1.6.1 and later, the pseudo-instructions vcmmul.t and vrmmul.t are provided in order to program, with respect to either row-vector-format matrices or column-vector-format matrices, with an operand order which is the same as the calculation order. vcmmul.t switches vs and vt and then assembles as vmmul.t. As a result, the following can be written.

```
vcmmul.t        e000, e100, e200   (equivalent to vmmul.t e000, e200, e100)
vrmmul.t        m000, m100, m200   (equivalent to vmmul.t m000, m100, m200)
```

Operation:

```
s <- ReadMatrix( TRIPLEXTRIPLEWORD,vs^t );
t <- ReadMatrix( TRIPLEXTRIPLEWORD,vt );
d[ 0] <- s[ 0]*t[ 0] + s[ 4]*t[ 4] + s[ 8]*t[ 8];
d[ 1] <- s[ 0]*t[ 1] + s[ 4]*t[ 5] + s[ 8]*t[ 9];
d[ 2] <- s[ 0]*t[ 2] + s[ 4]*t[ 6] + s[ 8]*t[10];
d[ 4] <- s[ 1]*t[ 0] + s[ 5]*t[ 4] + s[ 9]*t[ 8];
d[ 5] <- s[ 1]*t[ 1] + s[ 5]*t[ 5] + s[ 9]*t[ 9];
d[ 6] <- s[ 1]*t[ 2] + s[ 5]*t[ 6] + s[ 9]*t[10];
d[ 8] <- s[ 2]*t[ 0] + s[ 6]*t[ 4] + s[10]*t[ 8];
d[ 9] <- s[ 2]*t[ 1] + s[ 6]*t[ 5] + s[10]*t[ 9];
d[10] <- s[ 2]*t[ 2] + s[ 6]*t[ 6] + s[10]*t[10];
WriteMatrix( TRIPLEXTRIPLEWORD,vd, d );

or

s <- ReadMatrix( TRIPLEXTRIPLEWORD,vs );
t <- ReadMatrix( TRIPLEXTRIPLEWORD,vt );
d[ 0] <- s[ 0]*t[ 0] + s[ 1]*t[ 4] + s[ 2]*t[ 8];
d[ 1] <- s[ 0]*t[ 1] + s[ 1]*t[ 5] + s[ 2]*t[ 9];
d[ 2] <- s[ 0]*t[ 2] + s[ 1]*t[ 6] + s[ 2]*t[10];
d[ 4] <- s[ 4]*t[ 0] + s[ 5]*t[ 4] + s[ 6]*t[ 8];
d[ 5] <- s[ 4]*t[ 1] + s[ 5]*t[ 5] + s[ 6]*t[ 9];
d[ 6] <- s[ 4]*t[ 2] + s[ 5]*t[ 6] + s[ 6]*t[10];
d[ 8] <- s[ 8]*t[ 0] + s[ 9]*t[ 4] + s[10]*t[ 8];
d[ 9] <- s[ 8]*t[ 1] + s[ 9]*t[ 5] + s[10]*t[ 9];
d[10] <- s[ 8]*t[ 2] + s[ 9]*t[ 6] + s[10]*t[10];
WriteMatrix( TRIPLEXTRIPLEWORD,vd, d );
```

vmmul.q

Multiply Quad x Quad Matrix

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	0 0 0	vt	1	vs^t	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmmul.q  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 22   pitch : 16

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

Matrix multiplication is performed on the 4x4 floating-point elements of the matrix register specified by vs and the 4x4 floating-point elements of the matrix register specified by vt, and the resultant 4x4 floating-point elements are stored in the location of the matrix register specified by vd.

Notes:

In the opcode for the vmmul.q instruction, please note that the field which corresponds to the operand vs is written as vs^t which is an inverted expression of the matrix register. The assembler handles the vs operand of the vmmul.q instruction as an exceptional case by inverting the RXC bit (bit 13 of the opcode), and swapping the high-order bit of the IDX field (bit 9 of the opcode) with the FSL field (bit 14 of the opcode). As a result, the operation of multiplying the row-vector-format 4D matrix m100 by m200 from the right, can be written as follows.

```
vmmul.q          m000, m100, m200
```

In contrast, for column-vector-format matrices, a transposition is performed. To perform an operation in which the column-vector-format 4D matrix e100 is multiplied by e200 from the right, the following must be written.

```
vmmul.q         e000, e200, e100
```

For psp-as 1.6.1 and later, the pseudo-instructions vcmmul.q and vrmmul.q are provided in order to program, with respect to either row-vector-format matrices or column-vector-format matrices, with an operand order which is the same as the calculation order. vcmmul.q switches vs and vt and then assembles. As a result, the following can be written.

```
vcmmul.q        e000, e100, e200   (equivalent to vmmul.q e000, e200, e100)
vrmmul.q        m000, m100, m200   (equivalent to vmmul.q m000, m100, m200)
```

Operation:

```
s <- ReadMatrix( QUADXQUADWORD,vs^t );
t <- ReadMatrix( QUADXQUADWORD,vt );
d[ 0] <- s[ 0]*t[ 0] + s[ 4]*t[ 4] + s[ 8]*t[ 8] + s[12]*t[12];
d[ 1] <- s[ 0]*t[ 1] + s[ 4]*t[ 5] + s[ 8]*t[ 9] + s[12]*t[13];
d[ 2] <- s[ 0]*t[ 2] + s[ 4]*t[ 6] + s[ 8]*t[10] + s[12]*t[14];
d[ 3] <- s[ 0]*t[ 3] + s[ 4]*t[ 7] + s[ 8]*t[11] + s[12]*t[15];
d[ 4] <- s[ 1]*t[ 0] + s[ 5]*t[ 4] + s[ 9]*t[ 8] + s[13]*t[12];
d[ 5] <- s[ 1]*t[ 1] + s[ 5]*t[ 5] + s[ 9]*t[ 9] + s[13]*t[13];
d[ 6] <- s[ 1]*t[ 2] + s[ 5]*t[ 6] + s[ 9]*t[10] + s[13]*t[14];
d[ 7] <- s[ 1]*t[ 3] + s[ 5]*t[ 7] + s[ 9]*t[11] + s[13]*t[15];
d[ 8] <- s[ 2]*t[ 0] + s[ 6]*t[ 4] + s[10]*t[ 8] + s[14]*t[12];
d[ 9] <- s[ 2]*t[ 1] + s[ 6]*t[ 5] + s[10]*t[ 9] + s[14]*t[13];
d[10] <- s[ 2]*t[ 2] + s[ 6]*t[ 6] + s[10]*t[10] + s[14]*t[14];
d[11] <- s[ 2]*t[ 3] + s[ 6]*t[ 7] + s[10]*t[11] + s[14]*t[15];
d[12] <- s[ 3]*t[ 0] + s[ 7]*t[ 4] + s[11]*t[ 8] + s[15]*t[12];
d[13] <- s[ 3]*t[ 1] + s[ 7]*t[ 5] + s[11]*t[ 9] + s[15]*t[13];
d[14] <- s[ 3]*t[ 2] + s[ 7]*t[ 6] + s[11]*t[10] + s[15]*t[14];
d[15] <- s[ 3]*t[ 3] + s[ 7]*t[ 7] + s[11]*t[11] + s[15]*t[15];
WriteMatrix( QUADXQUADWORD,vd, d );

or

s <- ReadMatrix( QUADXQUADWORD,vs );
t <- ReadMatrix( QUADXQUADWORD,vt );
d[ 0] <- s[ 0]*t[ 0] + s[ 1]*t[ 4] + s[ 2]*t[ 8] + s[ 3]*t[12];
d[ 1] <- s[ 0]*t[ 1] + s[ 1]*t[ 5] + s[ 2]*t[ 9] + s[ 3]*t[13];
d[ 2] <- s[ 0]*t[ 2] + s[ 1]*t[ 6] + s[ 2]*t[10] + s[ 3]*t[14];
d[ 3] <- s[ 0]*t[ 3] + s[ 1]*t[ 7] + s[ 2]*t[11] + s[ 3]*t[15];
d[ 4] <- s[ 4]*t[ 0] + s[ 5]*t[ 4] + s[ 6]*t[ 8] + s[ 7]*t[12];
d[ 5] <- s[ 4]*t[ 1] + s[ 5]*t[ 5] + s[ 6]*t[ 9] + s[ 7]*t[13];
d[ 6] <- s[ 4]*t[ 2] + s[ 5]*t[ 6] + s[ 6]*t[10] + s[ 7]*t[14];
d[ 7] <- s[ 4]*t[ 3] + s[ 5]*t[ 7] + s[ 6]*t[11] + s[ 7]*t[15];
d[ 8] <- s[ 8]*t[ 0] + s[ 9]*t[ 4] + s[10]*t[ 8] + s[11]*t[12];
d[ 9] <- s[ 8]*t[ 1] + s[ 9]*t[ 5] + s[10]*t[ 9] + s[11]*t[13];
d[10] <- s[ 8]*t[ 2] + s[ 9]*t[ 6] + s[10]*t[10] + s[11]*t[14];
d[11] <- s[ 8]*t[ 3] + s[ 9]*t[ 7] + s[10]*t[11] + s[11]*t[15];
d[12] <- s[12]*t[ 0] + s[13]*t[ 4] + s[14]*t[ 8] + s[15]*t[12];
```

```
        d[13] <- s[12]*t[ 1] + s[13]*t[ 5] + s[14]*t[ 9] + s[15]*t[13];
        d[14] <- s[12]*t[ 2] + s[13]*t[ 6] + s[14]*t[10] + s[15]*t[14];
        d[15] <- s[12]*t[ 3] + s[13]*t[ 7] + s[14]*t[11] + s[15]*t[15];
        WriteMatrix( QUADXQUADWORD,vd, d );
```

vmone.p

One Pair x Pair Matrix

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1	vd
6	19	7

VFPU

Syntax:

```
vmone.p  vd
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 4 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

A 2x2 matrix is generated in which all elements have a floating-point value of 1.0. The generated 2x2 matrix is stored at locations in the matrix register file indicated by vd.

Operation:

```
d[0]  <- 1.0;
d[1]  <- 1.0;
d[4]  <- 1.0;
d[5]  <- 1.0;
WriteMatrix( PAIRXPAIRWORD, vd, d );
```

vmone.t

One Triple x Triple Matrix

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0				vd			
			6												19													7			

VFPU

Syntax:

```
vmone.t  vd
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 5 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

A 3x3 matrix is generated in which all elements have a floating-point value of 1.0. The generated 3x3 matrix is stored at locations in the matrix register file indicated by vd.

Operation:

```
d[0]  <- 1.0;
d[1]  <- 1.0;
d[2]  <- 1.0;
d[4]  <- 1.0;
d[5]  <- 1.0;
d[6]  <- 1.0;
d[8]  <- 1.0;
d[9]  <- 1.0;
d[10] <- 1.0;
WriteMatrix( TRIPLEXTRIPLEWORD, vd, d );
```

vmone.q

One Quad x Quad Matrix

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1	vd
6	19	7

VFPU

Syntax:

```
vmone.q  vd
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 6　　　pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

A 4x4 matrix is generated in which all elements have a floating-point value of 1.0. The generated 4x4 matrix is stored at locations in the matrix register file indicated by vd.

Operation:

```
d[0]  <- 1.0;
d[1]  <- 1.0;
d[2]  <- 1.0;
d[3]  <- 1.0;
d[4]  <- 1.0;
d[5]  <- 1.0;
d[6]  <- 1.0;
d[7]  <- 1.0;
d[8]  <- 1.0;
d[9]  <- 1.0;
d[10] <- 1.0;
d[11] <- 1.0;
d[12] <- 1.0;
d[13] <- 1.0;
d[14] <- 1.0;
d[15] <- 1.0;
```

```
        WriteMatrix( QUADXQUADWORD, vd, d );
```

vmov.s

Move Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vmov.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The floating-point value of one element from the matrix register indicated by vs is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d <- s;
WriteMatrix( SINGLEWORD, vd, d );
```

vmov.p

Move Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vmov.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The floating-point values of two elements from the matrix registers indicated by vs are stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d <- s;
WriteMatrix( PAIRWORD, vd, d );
```

vmov.t

Move Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 0 0 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vmov.t  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The floating-point values of three elements from the matrix registers indicated by vs are stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d <- s;
WriteMatrix( TRIPLEWORD, vd, d );
```

vmov.q

Move Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 0 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vmov.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The floating-point values of four elements from the matrix registers indicated by vs are stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d <- s;
WriteMatrix( QUADWORD, vd, d );
```

vmscl.p

Scale Pair x Pair Matrix

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 0 0	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmscl.p  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The elements of the 2x2 matrix from the matrix registers indicated by vs are multiplied by one element from the matrix register indicated by vt. The elements are treated as floating-point values. The 2x2 matrix floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRXPAIRWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0]  <- s[0]  * t[0];
d[1]  <- s[1]  * t[0];
d[4]  <- s[4]  * t[0];
d[5]  <- s[5]  * t[0];
WriteMatrix( PAIRXPAIRWORD, vd, d );
```

vmscl.t

Scale Triple x Triple Matrix

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 0 0	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmscl.t  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9   pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The elements of the 3x3 matrix from the matrix registers indicated by vs are multiplied by one element from the matrix register indicated by vt. The elements are treated as floating-point values. The 3x3 matrix floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEXTRIPLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0]  <- s[0]  * t[0];
d[1]  <- s[1]  * t[0];
d[2]  <- s[2]  * t[0];
d[4]  <- s[4]  * t[0];
d[5]  <- s[5]  * t[0];
d[6]  <- s[6]  * t[0];
d[8]  <- s[8]  * t[0];
d[9]  <- s[9]  * t[0];
d[10] <- s[10] * t[0];
WriteMatrix( TRIPLEXTRIPLEWORD, vd, d );
```

vmscl.q

Scale Quad x Quad Matrix

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 0 0	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmscl.q  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10 pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The elements of the 4x4 matrix from the matrix registers indicated by vs are multiplied by one element from the matrix register indicated by vt. The elements are treated as floating-point values. The 4x4 matrix floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADXQUADWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0]  <- s[0]  * t[0];
d[1]  <- s[1]  * t[0];
d[2]  <- s[2]  * t[0];
d[3]  <- s[3]  * t[0];
d[4]  <- s[4]  * t[0];
d[5]  <- s[5]  * t[0];
d[6]  <- s[6]  * t[0];
d[7]  <- s[7]  * t[0];
d[8]  <- s[8]  * t[0];
d[9]  <- s[9]  * t[0];
```

```
        d[10] <- s[10] * t[0];
        d[11] <- s[11] * t[0];
        d[12] <- s[12] * t[0];
        d[13] <- s[13] * t[0];
        d[14] <- s[14] * t[0];
        d[15] <- s[15] * t[0];
        WriteMatrix( QUADXQUADWORD, vd, d );
```

vmtvc

VFPU Move Word to VFPU Control

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 1 0 0 0 1 0	vs	imm8
6	11	7	8

VFPU

Syntax:

```
vmtvc  imm8, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

The floating-point value of one element from the matrix register indicated by vs is stored in the VFPU control register indicated by the imm8 field.

Operation:

```
dataword <- ReadMatrix( SINGLEWORD, vs );
WriteControl( imm8, dataword );
```

vmul.s

Multiply Single Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	0 0 0	vt	0	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmul.s  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

One element from the matrix register indicated by vs is multiplied by one element from the matrix register indicated by vt. The elements are treated as floating-point numbers. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <- s[0] * t[0];
WriteMatrix( SINGLEWORD, vd, d );
```

vmul.p

Multiply Pair Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	0 0 0	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmul.p  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Two elements from the matrix registers indicated by vs are multiplied by two elements from the matrix registers indicated by vt. The elements are treated as floating-point numbers. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vt );
d[0] <- s[0] * t[0];
d[1] <- s[1] * t[1];
WriteMatrix( PAIRWORD, vd, d );
```

vmul.t

Multiply Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	0 0 0	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmul.t  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Three elements from the matrix registers indicated by vs are multiplied by three elements from the matrix registers indicated by vt. The elements are treated as floating-point numbers. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- s[0] * t[0];
d[1] <- s[1] * t[1];
d[2] <- s[2] * t[2];
WriteMatrix( TRIPLEWORD, vd, d );
```

vmul.q

Multiply Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	0 0 0	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vmul.q  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Four elements from the matrix registers indicated by vs are multiplied by four elements from the matrix registers indicated by vt. The elements are treated as floating-point numbers. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
d[0] <- s[0] * t[0];
d[1] <- s[1] * t[1];
d[2] <- s[2] * t[2];
d[3] <- s[3] * t[3];
WriteMatrix( QUADWORD, vd, d );
```

vmzero.p

Zero Pair x Pair Matrix

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1	vd
6	19	7

VFPU

Syntax:

```
vmzero.p  vd
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 4 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The 2x2 zero matrix is generated and stored at locations in the matrix register file indicated by vd. Elements are stored as floating-point values.

Operation:

```
d[0]  <- 0.0;
d[1]  <- 0.0;
d[4]  <- 0.0;
d[5]  <- 0.0;
WriteMatrix( PAIRXPAIRWORD, vd, d );
```

vmzero.t

Zero Triple x Triple Matrix

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	vd						
6						19																			7						

VFPU

Syntax:

```
vmzero.t  vd
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 5 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The 3x3 zero matrix is generated and stored at locations in the matrix register file indicated by vd. Elements are stored as floating-point values.

Operation:

```
d[0]  <- 0.0;
d[1]  <- 0.0;
d[2]  <- 0.0;
d[4]  <- 0.0;
d[5]  <- 0.0;
d[6]  <- 0.0;
d[8]  <- 0.0;
d[9]  <- 0.0;
d[10] <- 0.0;
WriteMatrix( TRIPLEXTRIPLEWORD, vd, d );
```

vmzero.q

Zero Quad x Quad Matrix

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	1	1	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	vd						
6						19																			7						

VFPU

Syntax:

```
vmzero.q  vd
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 6 pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The 4x4 zero matrix is generated and stored at locations in the matrix register file indicated by vd. Elements are stored as floating-point values.

Operation:

```
d[0]  <- 0.0;
d[1]  <- 0.0;
d[2]  <- 0.0;
d[3]  <- 0.0;
d[4]  <- 0.0;
d[5]  <- 0.0;
d[6]  <- 0.0;
d[7]  <- 0.0;
d[8]  <- 0.0;
d[9]  <- 0.0;
d[10] <- 0.0;
d[11] <- 0.0;
d[12] <- 0.0;
d[13] <- 0.0;
d[14] <- 0.0;
d[15] <- 0.0;
```

```
        WriteMatrix( QUADXQUADWORD, vd, d );
```

vneg.s

Negate Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 1 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vneg.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Valid

Description:

The sign of the floating-point value of one element from the matrix register indicated by vs is inverted. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Special solutions are as follows.

neg(+***nan***) = -***nan***

neg(-***nan***) = +***nan***

neg(+***inf***) = -***inf***

neg(-***inf***) = +***inf***

neg(+0.0) = -0.0

neg(-0.0) = +0.0

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- -s[0];
WriteMatrix( SINGLEWORD, vd, d );
```

vneg.p

Negate Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 1 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vneg.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Valid

Description:

The signs of the floating-point values of two elements from the matrix registers indicated by vs are inverted. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.
Special solutions are as follows.

neg(+***nan***) = -***nan***

neg(-***nan***) = +***nan***

neg(+***inf***) = -***inf***

neg(-***inf***) = +***inf***

neg(+0.0) = -0.0

neg(-0.0) = +0.0

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
```

```
        d[0] <- -s[0];
        d[1] <- -s[1];
        WriteMatrix( PAIRWORD, vd, d );
```

vneg.t

Negate Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 1 0 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vneg.t  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency：3　　　　pitch：1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Valid

Description:

The signs of the floating-point values of three elements from the matrix registers indicated by vs are inverted. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.
Special solutions are as follows.

neg(+***nan***) = ***-nan***

neg(***-nan***) = +***nan***

neg(+***inf***) = ***-inf***

neg(***-inf***) = +***inf***

neg(+0.0) = -0.0

neg(-0.0) = +0.0

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
```

```
        d[0] <- -s[0];
        d[1] <- -s[1];
        d[2] <- -s[2];
        WriteMatrix( TRIPLEWORD, vd, d );
```

vneg.q

Negate Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 0 1 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vneg.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Valid

Description:

The signs of the floating-point values of four elements from the matrix registers indicated by vs are inverted. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Special solutions are as follows.

neg(+***nan***) = ***-nan***

neg(-***nan***) = +***nan***

neg(+***inf***) = ***-inf***

neg(-***inf***) = +***inf***

neg(+0.0) = -0.0

neg(-0.0) = +0.0

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
```

```
        d[0] <- -s[0];
        d[1] <- -s[1];
        d[2] <- -s[2];
        d[3] <- -s[3];
        WriteMatrix( QUADWORD, vd, d );
```

vnop

Nop

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6						26																									

VFPU

Syntax:

```
vnop
```

Instruction Type:

Synchronization instruction

Processing Time:

latency : 0 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

No operation is performed.

Notes:

In order to circumvent a known VFPU problem, the assembler automatically inserts vnops before VFPU instructions that follow sv.q and wb instructions.

Operation:

```
Nop();
```

vnrcp.s

Negative Reciprocal Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 1 0 0 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vnrcp.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

The negative reciprocal of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| (\boldsymbol{approx\_reciprocal}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x}) | < 6.3 \times 10^{-7} , 0.0 <= | \boldsymbol{x} | < 2^{126}$

$| (\boldsymbol{approx\_reciprocal}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x}) | <= 1.0 , 2^{126} <= | \boldsymbol{x} | < \boldsymbol{inf}$

Special solutions are as follows.

approx_reciprocal(***nan***) = ***nan***

approx_reciprocal(+***inf***) = +0.0

approx_reciprocal(-***inf***) = -0.0

approx_reciprocal(+0.0) = +***inf***

approx_reciprocal(-0.0) = -***inf***

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- -1 * approx_reciprocal( s[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vnrcp.p

Negative Reciprocal Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 1 0 0 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vnrcp.p  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8   pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The negative reciprocal of the floating-point values of two elements from the matrix registers indicated by vs is calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| (\boldsymbol{approx\_reciprocal}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x}) | < 6.3 \times 10^{-7}$, $0.0 <= | \boldsymbol{x} | < 2^{126}$

$| (\boldsymbol{approx\_reciprocal}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x}) | <= 1.0$, $2^{126} <= | \boldsymbol{x} | < \boldsymbol{inf}$

Special solutions are as follows.

approx_reciprocal*(*nan*) = *nan

approx_reciprocal(+***inf***) = +0.0

approx_reciprocal(-***inf***) = -0.0

approx_reciprocal(+0.0) = +***inf***

approx_reciprocal(-0.0) = -***inf***

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- -1 * approx_reciprocal( s[0] );
d[1] <- -1 * approx_reciprocal( s[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vnrcp.t

Negative Reciprocal Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 1 0 0 0 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vnrcp.t  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The negative reciprocal of the floating-point values of three elements from the matrix registers indicated by vs is calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| (\boldsymbol{approx\_reciprocal}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x}) | < 6.3 \times 10^{-7}$, $0.0 <= | \boldsymbol{x} | < 2^{126}$

$| (\boldsymbol{approx\_reciprocal}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x}) | <= 1.0$, $2^{126} <= | \boldsymbol{x} | < \boldsymbol{inf}$

Special solutions are as follows.

approx_reciprocal(***nan***) = ***nan***

approx_reciprocal(+***inf***) = +0.0

approx_reciprocal(-***inf***) = -0.0

approx_reciprocal(+0.0) = +***inf***

approx_reciprocal(-0.0) = -***inf***

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- -1 * approx_reciprocal( s[0] );
d[1] <- -1 * approx_reciprocal( s[1] );
d[2] <- -1 * approx_reciprocal( s[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vnrcp.q

Negative Reciprocal Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 1 0 0 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vnrcp.q  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10 pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The negative reciprocal of the floating-point values of four elements from the matrix registers indicated by vs is calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| (\textit{\textbf{approx\_reciprocal}}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x}) | < 6.3 \times 10^{-7}$, $0.0 <= | \boldsymbol{x} | < 2^{126}$

$| (\textit{\textbf{approx\_reciprocal}}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x}) | <= 1.0$, $2^{126} <= | \boldsymbol{x} | < \textit{\textbf{inf}}$

Special solutions are as follows.

approx_reciprocal(***nan***) = ***nan***

approx_reciprocal(+***inf***) = +0.0

approx_reciprocal(-***inf***) = -0.0

approx_reciprocal(+0.0) = +***inf***

approx_reciprocal(-0.0) = -***inf***

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- -1 * approx_reciprocal( s[0] );
d[1] <- -1 * approx_reciprocal( s[1] );
d[2] <- -1 * approx_reciprocal( s[2] );
d[3] <- -1 * approx_reciprocal( s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vnsin.s

Negative Sine Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 1 0 1 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vnsin.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

The negative sine of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \boldsymbol{approx\_sin}(\boldsymbol{M\_PI\_2} \times \boldsymbol{x}) - \boldsymbol{sin}(\boldsymbol{M\_PI\_2} \times \boldsymbol{x}) | < 4.8 \times 10^{-7}$; $0.0 <= | \boldsymbol{x} | < 32.0$

$| \boldsymbol{approx\_sin}(\boldsymbol{M\_PI\_2} \times \boldsymbol{x}) - \boldsymbol{sin}(\boldsymbol{M\_PI\_2} \times \boldsymbol{x}) | < 2.0$; $32.0 <= | \boldsymbol{x} | < \boldsymbol{inf}$

Special solutions are as follows.

approx_sin(+***nan***) = +***nan***

approx_sin(-***nan***) = -***nan***

approx_sin(+***inf***) = +***nan***

approx_sin(-***inf***) = -***nan***

approx_sin(+0.0) = +0.0

approx_sin(-0.0) = -0.0

Notes:

The units of the angle which is provided as input for the vnsin.s instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- -1 * approx_sin( M_PI_2 * s[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vnsin.p

Negative Sine Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 1 0 1 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vnsin.p  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8　　　　pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The negative sines of the floating-point values of two elements from the matrix register indicated by vs are calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \textit{approx\_sin}(M\_PI\_2 \times x) - \sin(M\_PI\_2 \times x) | < 4.8 \times 10^{-7}$; $0.0 <= | x | < 32.0$

$| \textit{approx\_sin}(M\_PI\_2 \times x) - \sin(M\_PI\_2 \times x) | < 2.0$; $32.0 <= | x | < \textit{inf}$

Special solutions are as follows.

approx_sin(+***nan***) = +***nan***

approx_sin(-***nan***) = -***nan***

approx_sin(+***inf***) = +***nan***

approx_sin(-***inf***) = -***nan***

approx_sin(+0.0) = +0.0

approx_sin(-0.0) = -0.0

Notes:

The units of the angle which is provided as input for the vnsin.p instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- -1 * approx_sin( M_PI_2 * s[0] );
d[1] <- -1 * approx_sin( M_PI_2 * s[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vnsin.t

Negative Sine Triple Word

31–26	25–15	14–8	7	6–0
1 1 0 1 0 0	0 0 0 0 0 1 1 0 1 0 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vnsin.t  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The negative sines of the floating-point values of three elements from the matrix register indicated by vs are calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \boldsymbol{approx\_sin}(\boldsymbol{M\_PI\_2} \times \boldsymbol{x}) - \boldsymbol{sin}(\boldsymbol{M\_PI\_2} \times \boldsymbol{x}) | < 4.8 \times 10^{-7}$; $0.0 <= | \boldsymbol{x} | < 32.0$

$| \boldsymbol{approx\_sin}(\boldsymbol{M\_PI\_2} \times \boldsymbol{x}) - \boldsymbol{sin}(\boldsymbol{M\_PI\_2} \times \boldsymbol{x}) | < 2.0$; $32.0 <= | \boldsymbol{x} | < \boldsymbol{inf}$

Special solutions are as follows.

approx_sin(+***nan***) = +***nan***

approx_sin(-***nan***) = -***nan***

approx_sin(+***inf***) = +***nan***

approx_sin(-***inf***) = -***nan***

approx_sin(+0.0) = +0.0

approx_sin(-0.0) = -0.0

Notes:

The units of the angle which is provided as input for the vnsin.t instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- -1 * approx_sin( M_PI_2 * s[0] );
d[1] <- -1 * approx_sin( M_PI_2 * s[1] );
d[2] <- -1 * approx_sin( M_PI_2 * s[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vnsin.q

Negative Sine Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 1 0 1 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vnsin.q  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10   pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The negative sines of the floating-point values of four elements from the matrix register indicated by vs are calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \textit{\textbf{approx\_sin}}(\textit{\textbf{M\_PI\_2}} \times \textit{\textbf{x}}) - \textit{\textbf{sin}}(\textit{\textbf{M\_PI\_2}} \times \textit{\textbf{x}}) | < 4.8 \times 10^{-7} \text{ ; } 0.0 <= | \textit{\textbf{x}} | < 32.0$

$| \textit{\textbf{approx\_sin}}(\textit{\textbf{M\_PI\_2}} \times \textit{\textbf{x}}) - \textit{\textbf{sin}}(\textit{\textbf{M\_PI\_2}} \times \textit{\textbf{x}}) | < 2.0 \text{ ; } 32.0 <= | \textit{\textbf{x}} | < \textit{\textbf{inf}}$

Special solutions are as follows.

approx_sin(+***nan***) = +***nan***

approx_sin(-***nan***) = -***nan***

approx_sin(+***inf***) = +***nan***

approx_sin(-***inf***) = -***nan***

approx_sin(+0.0) = +0.0

approx_sin(-0.0) = -0.0

Notes:

The units of the angle which is provided as input for the vnsin.q instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- -1 * approx_sin( M_PI_2 * s[0] );
d[1] <- -1 * approx_sin( M_PI_2 * s[1] );
d[2] <- -1 * approx_sin( M_PI_2 * s[2] );
d[3] <- -1 * approx_sin( M_PI_2 * s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vocp.s

One's Complement Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 1 0 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vocp.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Valid

Description:

The one's complement of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- 1.0 - s[0];
WriteMatrix( SINGLEWORD, vd, d );
```

vocp.p

One's Complement Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 1 0 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vocp.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Valid

Description:

The one's complements of the floating-point values of two elements from the matrix registers indicated by vs are calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- 1.0 - s[0];
d[1] <- 1.0 - s[1];
WriteMatrix( PAIRWORD, vd, d );
```

vocp.t

One's Complement Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 1 0 0 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vocp.t  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Valid

Description:

The one's complements of the floating-point values of three elements from the matrix registers indicated by vs are calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- 1.0 - s[0];
d[1] <- 1.0 - s[1];
d[2] <- 1.0 - s[2];
WriteMatrix( TRIPLEWORD, vd, d );
```

vocp.q

One's Complement Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 1 0 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vocp.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Valid

Description:

The one's complements of the floating-point values of four elements from the matrix registers indicated by vs are calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- 1.0 - s[0];
d[1] <- 1.0 - s[1];
d[2] <- 1.0 - s[2];
d[3] <- 1.0 - s[3];
WriteMatrix( QUADWORD, vd, d );
```

vone.s

Set One Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0	vd
6	19	7

VFPU

Syntax:

```
vone.s  vd
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

The value 1.0 is stored as a one-element floating-point value at the location in the matrix register file indicated by vd.

Operation:

```
d[0] <- 1.0;
WriteMatrix( SINGLEWORD, vd, d );
```

vone.p

Set One Pair Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1	vd						
			6												19													7			

VFPU

Syntax:

```
vone.p  vd
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

The value 1.0 is stored as a two-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
d[0] <- 1.0;
d[1] <- 1.0;
WriteMatrix( PAIRWORD, vd, d );
```

vone.t

Set One Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0	vd
6	19	7

VFPU

Syntax:

```
vone.t  vd
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

The value 1.0 is stored as a three-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
d[0] <- 1.0;
d[1] <- 1.0;
d[2] <- 1.0;
WriteMatrix( TRIPLEWORD, vd, d );
```

vone.q

Set One Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 1	vd
6	19	7

VFPU

Syntax:

```
vone.q  vd
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

The value 1.0 is stored as a four-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
d[0] <- 1.0;
d[1] <- 1.0;
d[2] <- 1.0;
d[3] <- 1.0;
WriteMatrix( QUADWORD, vd, d );
```

vpfxd

Destination Prefix

31 30 29 28 27 26	25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
1 1 0 1 1 1	1 0	imm24
6	2	24

VFPU

Syntax:

```
vpfxd  wpx, wpy, wpz, wpw
```

Instruction Type:

Prefix instruction

Processing Time:

latency : 0 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Overwrite

Description:

A saturation or write mask operation is applied to the destination of the following instruction, as indicated by the arguments. The prefixing operation specified by this instruction is only valid for the next VFPU instruction, excluding the b*, mf*, mt*, lv*, sv*, vmf*, vmt*, vpfx*, vsync, vnop, and vflush instructions which are not affected by prefixing.

Operation:

```
WriteControl( VFPU_PFXD, imm24[11:0] );
```

vpfxs

Source Prefix

31 30 29 28 27 26	25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
1 1 0 1 1 1	0 0	imm24
6	2	24

VFPU

Syntax:

```
vpfxs  rpx, rpy, rpz, rpw
```

Instruction Type:

Prefix instruction

Processing Time:

latency : 0 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Overwrite	No effect	No effect

Description:

A swizzle, absolute value, constant insertion or negation operation is applied to the source of the following instruction, as indicated by the arguments. The prefixing operation specified by this instruction is only valid for the next VFPU instruction, excluding the b*, mf*, mt*, lv*, sv*, vmf*, vmt*, vpfx*, vsync, vnop, and vflush instructions which are not affected by prefixing.

Operation:

```
WriteControl( VFPU_PFXS, imm24[19:0] );
```

vpfxt

Target Prefix

31 30 29 28 27 26	25 24	23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
1 1 0 1 1 1	0 1	imm24
6	2	24

VFPU

Syntax:

```
vpfxt  rpx, rpy, rpz, rpw
```

Instruction Type:

Prefix instruction

Processing Time:

latency : 0 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	Overwrite	No effect

Description:

A swizzle, absolute value, constant insertion or negation operation is applied to the target of the following instruction, as indicated by the arguments. The prefixing operation specified by this instruction is only valid for the next VFPU instruction, excluding the b*, mf*, mt*, lv*, sv*, vmf*, vmt*, vpfx*, vsync, vnop, and vflush instructions which are not affected by prefixing.

Operation:

```
WriteControl( VFPU_PFXT, imm24[19:0] );
```

vqmul.q

Quaternion Multiply Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 0 1	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vqmul.q  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10   pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The quaternion multiplication of the four elements from the matrix registers indicated by vs and the four elements from the matrix registers indicated by vt is performed. The elements are treated as floating-point values. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
d[0] <- + s[0]*t[3] + s[1]*t[2] - s[2]*t[1] + s[3]*t[0];
d[1] <- - s[0]*t[2] + s[1]*t[3] + s[2]*t[0] + s[3]*t[1];
d[2] <- + s[0]*t[1] - s[1]*t[0] + s[2]*t[3] + s[3]*t[2];
d[3] <- - s[0]*t[0] - s[1]*t[1] - s[2]*t[2] + s[3]*t[3];
WriteMatrix( QUADWORD, vd, d );
```

vrcp.s

Reciprocal Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 0 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vrcp.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7   pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The reciprocal of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

$| (\boldsymbol{approx\_reciprocal}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x}) | < 6.3 \times 10^{-7}$, $0.0 <= | \boldsymbol{x} |< 2^{126}$

$| (\boldsymbol{approx\_reciprocal}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x}) | <= 1.0$, $2^{126} <= | \boldsymbol{x} | < \boldsymbol{inf}$

Special solutions are as follows.

approx_reciprocal*(*nan*)** = ***nan

approx_reciprocal(+***inf***) = +0.0

approx_reciprocal(-***inf***) = -0.0

approx_reciprocal(+0.0) = +***inf***

approx_reciprocal(-0.0) = -***inf***

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
```

```
        d[0] <- approx_reciprocal( s[0] );
        WriteMatrix( SINGLEWORD, vd, d );
```

vrcp.p

Reciprocal Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 0 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vrcp.p  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The reciprocals of the floating-point values of two elements from the matrix registers indicated by vs are calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| (\textbf{\textit{approx\_reciprocal}}(\textbf{\textit{x}}) - (1/\textbf{\textit{x}})) / (1/\textbf{\textit{x}}) | < 6.3 \times 10^{-7}$, $0.0 <= | \textbf{\textit{x}} |< 2^{126}$

$| (\textbf{\textit{approx\_reciprocal}}(\textbf{\textit{x}}) - (1/\textbf{\textit{x}})) / (1/\textbf{\textit{x}}) | <= 1.0$, $2^{126} <= | \textbf{\textit{x}} | < \textbf{\textit{inf}}$

Special solutions are as follows.

approx_reciprocal(***nan***) = ***nan***

approx_reciprocal(+***inf***) = +0.0

approx_reciprocal(-***inf***) = -0.0

approx_reciprocal(+0.0) = +***inf***

approx_reciprocal(-0.0) = -***inf***

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- approx_reciprocal( s[0] );
d[1] <- approx_reciprocal( s[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vrcp.t

Reciprocal Triple Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	vs	0	vd
6						11											7	1	7

VFPU

Syntax:

```
vrcp.t  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The reciprocals of the floating-point values of three elements from the matrix registers indicated by vs are calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| (\textbf{\textit{approx\_reciprocal}}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x}) | < 6.3 \times 10^{-7}$, $0.0 <= | \boldsymbol{x} | < 2^{126}$

$| (\textbf{\textit{approx\_reciprocal}}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x}) | <= 1.0$, $2^{126} <= | \boldsymbol{x} | < \textbf{\textit{inf}}$

Special solutions are as follows.

approx_reciprocal(***nan***) = ***nan***

approx_reciprocal(+***inf***) = +0.0

approx_reciprocal(-***inf***) = -0.0

approx_reciprocal(+0.0) = +***inf***

approx_reciprocal(-0.0) = -***inf***

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- approx_reciprocal( s[0] );
d[1] <- approx_reciprocal( s[1] );
d[2] <- approx_reciprocal( s[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vrcp.q

Reciprocal Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 0 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vrcp.q  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10 pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The reciprocals of the floating-point values of four elements from the matrix registers indicated by vs are calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$|(\boldsymbol{approx\_reciprocal}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x})| < 6.3 \times 10^{-7}, 0.0 <= |\boldsymbol{x}| < 2^{126}$

$|(\boldsymbol{approx\_reciprocal}(\boldsymbol{x}) - (1/\boldsymbol{x})) / (1/\boldsymbol{x})| <= 1.0, 2^{126} <= |\boldsymbol{x}| < \boldsymbol{inf}$

Special solutions are as follows.

approx_reciprocal(***nan***) = ***nan***

approx_reciprocal(+***inf***) = +0.0

approx_reciprocal(-***inf***) = -0.0

approx_reciprocal(+0.0) = +***inf***

approx_reciprocal(-0.0) = -***inf***

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- approx_reciprocal( s[0] );
d[1] <- approx_reciprocal( s[1] );
d[2] <- approx_reciprocal( s[2] );
d[3] <- approx_reciprocal( s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vrexp2.s

Reciprocal Exponential base 2 Single Word

31–26	25–15	14–8	7	6–0
1 1 0 1 0 0	0 0 0 0 0 1 1 1 0 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vrexp2.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

The reciprocal of the base 2 exponential of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

The valid input range is $-128 < \boldsymbol{x} < \boldsymbol{inf}$

The precision of the calculation is given by the following expression.

$| (1 / \boldsymbol{approx\_exp2}(\boldsymbol{x}) - 1 / (2^x)) / (1 / (2^x)) | < 7.2 \times 10^7$; $-128.0 < \boldsymbol{x} <= 0.0$

$| (1 / \boldsymbol{approx\_exp2}(\boldsymbol{x}) - 1 / (2^x)) / (1 / (2^x)) | < 7.2 \times 10^7$; $0.0 <= \boldsymbol{x} < 64.0$

$| (1 / \boldsymbol{approx\_exp2}(\boldsymbol{x}) - 1 / (2^x)) / (1 / (2^x)) | <= 1.0$; $64.0 <= \boldsymbol{x} < \boldsymbol{inf}$

Special solutions are as follows.

approx_exp2(+***nan***) = +***nan***

approx_exp2(-***nan***) = +***nan***

approx_exp2(+***inf***) = +***inf***

approx_exp2(-***inf***) = +0.0

approx_exp2(+0.0) = +1.0

approx_exp2(-0.0) = +1.0

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- 1 / approx_exp2( s[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vrexp2.p

Reciprocal Exponential base 2 Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 1 1 0 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vrexp2.p  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The reciprocals of the base 2 exponentials of the floating-point values of two elements from the matrix registers indicated by vs are calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The valid input range is $-128 < \boldsymbol{x} < \boldsymbol{inf}$

The precision of the calculation is given by the following expression.

$| (1 / \boldsymbol{approx\_exp2}(\boldsymbol{x}) - 1 / (2^{x})) / (1 / (2^{x})) | < 7.2 \times 10^{7}$; $-128.0 < \boldsymbol{x} <= 0.0$

$| (1 / \boldsymbol{approx\_exp2}(\boldsymbol{x}) - 1 / (2^{x})) / (1 / (2^{x})) | < 7.2 \times 10^{7}$; $0.0 <= \boldsymbol{x} < 64.0$

$| (1 / \boldsymbol{approx\_exp2}(\boldsymbol{x}) - 1 / (2^{x})) / (1 / (2^{x})) | <= 1.0$; $64.0 <= \boldsymbol{x} < \boldsymbol{inf}$

Special solutions are as follows.

approx_exp2(+***nan***) = +***nan***

approx_exp2(-***nan***) = +***nan***

approx_exp2(+***inf***) = +***inf***

approx_exp2(-***inf***) = +0.0

approx_exp2(+0.0) = +1.0

approx_exp2(-0.0) = +1.0

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- 1 / approx_exp2( s[0] );
d[1] <- 1 / approx_exp2( s[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vrexp2.t

Reciprocal Exponential base 2 Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 1 1 0 0 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vrexp2.t  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9　　　　pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The reciprocals of the base 2 exponentials of the floating-point values of three elements from the matrix registers indicated by vs are calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The valid input range is $-128 < \boldsymbol{x} < \boldsymbol{inf}$

The precision of the calculation is given by the following expression.

$| (1 / \boldsymbol{approx\_exp2}(\boldsymbol{x}) - 1 / (2^{x})) / (1 / (2^{x})) | < 7.2 \times 10^{7}$; $-128.0 < \boldsymbol{x} <= 0.0$

$| (1 / \boldsymbol{approx\_exp2}(\boldsymbol{x}) - 1 / (2^{x})) / (1 / (2^{x})) | < 7.2 \times 10^{7}$; $0.0 <= \boldsymbol{x} < 64.0$

$| (1 / \boldsymbol{approx\_exp2}(\boldsymbol{x}) - 1 / (2^{x})) / (1 / (2^{x})) | <= 1.0$; $64.0 <= \boldsymbol{x} < \boldsymbol{inf}$

Special solutions are as follows.

approx_exp2(+***nan***) = +***nan***

approx_exp2(-***nan***) = +***nan***

approx_exp2(+***inf***) = +***inf***

approx_exp2(-***inf***) = +0.0

approx_exp2(+0.0) = +1.0

approx_exp2(-0.0) = +1.0

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- 1 / approx_exp2( s[0] );
d[1] <- 1 / approx_exp2( s[1] );
d[2] <- 1 / approx_exp2( s[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vrexp2.q

Reciprocal Exponential base 2 Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 1 1 0 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vrexp2.q  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10　　　pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The reciprocals of the base 2 exponentials of the floating-point values of four elements from the matrix registers indicated by vs are calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The valid input range is $-128 < \boldsymbol{x} < \boldsymbol{inf}$

The precision of the calculation is given by the following expression.

$| (1 / \boldsymbol{approx\_exp2}(\boldsymbol{x}) - 1 / (2^{x})) / (1 / (2^{x})) | < 7.2 \times 10^{7}$; $-128.0 < \boldsymbol{x} <= 0.0$

$| (1 / \boldsymbol{approx\_exp2}(\boldsymbol{x}) - 1 / (2^{x})) / (1 / (2^{x})) | < 7.2 \times 10^{7}$; $0.0 <= \boldsymbol{x} < 64.0$

$| (1 / \boldsymbol{approx\_exp2}(\boldsymbol{x}) - 1 / (2^{x})) / (1 / (2^{x})) | <= 1.0$; $64.0 <= \boldsymbol{x} < \boldsymbol{inf}$

Special solutions are as follows.

approx_exp2(+***nan***) = +***nan***

approx_exp2(-***nan***) = +***nan***

approx_exp2(+***inf***) = +***inf***

approx_exp2(-***inf***) = +0.0

approx_exp2(+0.0) = +1.0

approx_exp2(-0.0) = +1.0

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- 1 / approx_exp2( s[0] );
d[1] <- 1 / approx_exp2( s[1] );
d[2] <- 1 / approx_exp2( s[2] );
d[3] <- 1 / approx_exp2( s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vrndf1.s

Random Floating Single Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0				vd			
			6												19													7			

VFPU

Syntax:

```
vrndf1.s  vd
```

Instruction Type:

Multi-cycle instruction

Processing Time:

latency : 5 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Valid

Description:

One pseudorandom number is generated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

The range of random numbers is given by the following expression.

$1.0 \leq \textbf{\textit{random}}() < 2.0$

The period is $\geq 10^{38}$.

Operation:

```
d[0] <- random();
WriteMatrix( SINGLEWORD, vd, d );
```

vrndf1.p

Random Floating Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1	vd
6	19	7

VFPU

Syntax:

```
vrndf1.p  vd
```

Instruction Type:

Repeat (multi-cycle) instruction

Processing Time:

latency : 8 pitch : 6

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Use prohibited

Description:

Two pseudorandom numbers are generated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The range of random numbers is given by the following expression.

$1.0 \leq \textbf{\textit{random}}() < 2.0$

The period is $\geq 10^{38}$.

Operation:

```
d[0] <- random();
d[1] <- random();
WriteMatrix( PAIRWORD, vd, d );
```

vrndf1.t

Random Floating Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0	vd
6	19	7

VFPU

Syntax:

```
vrndf1.t  vd
```

Instruction Type:

Repeat (multi-cycle) instruction

Processing Time:

latency : 11   pitch : 9

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Use prohibited

Description:

Three pseudorandom numbers are generated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The range of random numbers is given by the following expression.

$1.0 \leq \textbf{\textit{random}}() < 2.0$

The period is $\geq 10^{38}$.

Operation:

```
d[0] <- random();
d[1] <- random();
d[2] <- random();
WriteMatrix( TRIPLEWORD, vd, d );
```

vrndf1.q

Random Floating Quad Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1	vd						
6						19																			7						

VFPU

Syntax:

```
vrndf1.q  vd
```

Instruction Type:

Repeat (multi-cycle) instruction

Processing Time:

latency : 14　　　pitch : 12

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Use prohibited

Description:

Four pseudorandom numbers are generated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The range of random numbers is given by the following expression.

$1.0 \leq \textbf{\textit{random}}() < 2.0$

The period is $\geq 10^{38}$.

Operation:

```
d[0] <- random();
d[1] <- random();
d[2] <- random();
d[3] <- random();
WriteMatrix( QUADWORD, vd, d );
```

vrndf2.s

Random Floating Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0	vd
6	19	7

VFPU

Syntax:

```
vrndf2.s  vd
```

Instruction Type:

Multi-cycle instruction

Processing Time:

latency : 5　　　pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Valid

Description:

One pseudorandom number is generated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

The range of random numbers is given by the following expression.

$2.0 \leq$ ***random***$() < 4.0$

The period is $\geq 10^{38}$.

Operation:

```
d[0] <- random();
WriteMatrix( SINGLEWORD, vd, d );
```

vrndf2.p

Random Floating Pair Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	1	vd						
6						19																			7						

VFPU

Syntax:

```
vrndf2.p  vd
```

Instruction Type:

Repeat (multi-cycle) instruction

Processing Time:

latency : 8　　　　pitch : 6

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Use prohibited

Description:

Two pseudorandom numbers are generated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The range of random numbers is given by the following expression.

$2.0 \le \boldsymbol{random}() < 4.0$

The period is $\ge 10^{38}$.

Operation:

```
d[0] <- random();
d[1] <- random();
WriteMatrix( PAIRWORD, vd, d );
```

vrndf2.t

Random Floating Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0	vd
6	19	7

VFPU

Syntax:

```
vrndf2.t  vd
```

Instruction Type:

Repeat (multi-cycle) instruction

Processing Time:

latency : 11   pitch : 9

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Use prohibited

Description:

Three pseudorandom numbers are generated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The range of random numbers is given by the following expression.

$2.0 \leq$ ***random***$() < 4.0$

The period is $\geq 10^{38}$.

Operation:

```
d[0] <- random();
d[1] <- random();
d[2] <- random();
WriteMatrix( TRIPLEWORD, vd, d );
```

vrndf2.q

Random Floating Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 1	vd
6	19	7

VFPU

Syntax:

```
vrndf2.q  vd
```

Instruction Type:

Repeat (multi-cycle) instruction

Processing Time:

latency : 14 pitch : 12

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Use prohibited

Description:

Four pseudorandom numbers are generated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The range of random numbers is given by the following expression.

$2.0 \leq \textit{\textbf{random}}() < 4.0$

The period is $\geq 10^{38}$.

Operation:

```
d[0] <- random();
d[1] <- random();
d[2] <- random();
d[3] <- random();
WriteMatrix( QUADWORD, vd, d );
```

vrndi.s

Random Integer Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0	vd
6	19	7

VFPU

Syntax:

```
vrndi.s  vd
```

Instruction Type:

Multi-cycle instruction

Processing Time:

latency : 5　　　pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Valid

Description:

One pseudorandom number is generated. The one-element integer result is stored at the location in the matrix register file indicated by vd.

The range of random numbers is given by the following expression.

0x00000000 <= ***random***() <= 0xFFFFFFFF

The period is $\geq 10^{38}$.

Operation:

```
d[0] <- random();
WriteMatrix( SINGLEWORD, vd, d );
```

vrndi.p

Random Integer Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1	vd
6	19	7

VFPU

Syntax:

```
vrndi.p  vd
```

Instruction Type:

Repeat (multi-cycle) instruction

Processing Time:

latency : 8 pitch : 6

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Use prohibited

Description:

Two pseudorandom numbers are generated. The two-element integer result is stored at locations in the matrix register file indicated by vd.

The range of random numbers is given by the following expression.

0x00000000 <= ***random***() <= 0xFFFFFFFF

The period is $\geq 10^{38}$.

Operation:

```
d[0] <- random();
d[1] <- random();
WriteMatrix( PAIRWORD, vd, d );
```

vrndi.t

Random Integer Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0	vd
6	19	7

VFPU

Syntax:

```
vrndi.t  vd
```

Instruction Type:

Repeat (multi-cycle) instruction

Processing Time:

latency : 11 pitch : 9

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Use prohibited

Description:

Three pseudorandom numbers are generated. The three-element integer result is stored at locations in the matrix register file indicated by vd.

The range of random numbers is given by the following expression.

0x00000000 <= ***random***() <= 0xFFFFFFFF

The period is $\geq 10^{38}$.

Operation:

```
d[0] <- random();
d[1] <- random();
d[2] <- random();
WriteMatrix( TRIPLEWORD, vd, d );
```

vrndi.q

Random Integer Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1	vd
6	19	7

VFPU

Syntax:

```
vrndi.q  vd
```

Instruction Type:

Repeat (multi-cycle) instruction

Processing Time:

latency : 14 pitch : 12

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	Use prohibited

Description:

Four pseudorandom numbers are generated. The four-element integer result is stored at locations in the matrix register file indicated by vd.

The range of random numbers is given by the following expression.

0x00000000 <= ***random***() <= 0xFFFFFFFF

The period is $\geq 10^{38}$.

Operation:

```
d[0] <- random();
d[1] <- random();
d[2] <- random();
d[3] <- random();
WriteMatrix( QUADWORD, vd, d );
```

vrnds.s

Random Seed Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 0 0 0 0 0 0	vs	0 0 0 0 0 0 0 0
6	11	7	8

VFPU

Syntax:

```
vrnds.s  vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

The seed of the pseudorandom number generator is set with the integer value of one element from the matrix register indicated by vs. This value must be an integer other than zero.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
srand( s[0] );
```

vrot.p

Rotator Pair Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 1 1 0 1	imm5	0	vs	1	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vrot.p  vd, vs, imm5
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The rotators indicated by the imm5 field are calculated for the floating-point values of two elements from the matrix registers indicated by vs. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.
If the write positions overlap, the cos values are overwritten.
The following mnemonics can be used for imm5.

Code (imm5)	Mnemonic
0	[C,S]
1	[S,C]
2	[S,0]
3	[S,0]
4	[C,S]
5	[S,C]
6	[0,S]
7	[0,S]
8	[C,0]
9	[0,C]

Code (imm5)	Mnemonic
10	[S,S]
11	[0,0]
12	[C,0]
13	[0,C]
14	[0,0]
15	[S,S]
16	[C,-S]
17	[-S,C]
18	[-S,0]
19	[-S,0]
20	[C,-S]
21	[-S,C]
22	[0,-S]
23	[0,-S]
24	[C,0]
25	[0,C]
26	[-S,-S]
27	[0,0]
28	[C,0]
29	[0,C]
30	[0,0]
31	[-S,-S]

Notes:

The units of the angle which is provided as input for the vrot.p instruction is such that 1.0 represents π/2 in radians and 90° in degrees.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
ts <- approx_sin( M_PI_2 * s[0] );
tc <- approx_cos( M_PI_2 * s[0] );
d[0] <- 0;
d[1] <- 0;
if( imm5[4] )
    ts <- -ts;
if( imm5[3:2]==imm5[1:0] )
    begin
        d[0] <- ts;
        d[1] <- ts;
    end
else if( imm5[3:2] <2)
    d[imm5[3:2]] <- ts;
if( imm5[1:0] <2)
    d[(imm5[1:0]] <- tc;
WriteMatrix( PAIRWORD, vd, d );
```

vrot.t

Rotator Triple Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 1 1 0 1	imm5	1	vs	0	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vrot.t  vd, vs, imm5
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The rotators indicated by the imm5 field are calculated for the floating-point values of three elements from the matrix registers indicated by vs. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.
If the write positions overlap, the cos values are overwritten.
The following mnemonics can be used for imm5.

Code (imm5)	Mnemonic
0	[C,S,S]
1	[S,C,0]
2	[S,0,C]
3	[S,0,0]
4	[C,S,0]
5	[S,C,S]
6	[0,S,C]
7	[0,S,0]
8	[C,0,S]
9	[0,C,S]

Code (imm5)	Mnemonic
10	[S,S,C]
11	[0,0,S]
12	[C,0,0]
13	[0,C,0]
14	[0,0,C]
15	[S,S,S]
16	[C,-S,-S]
17	[-S,C,0]
18	[-S,0,C]
19	[-S,0,0]
20	[C,-S,0]
21	[-S,C,-S]
22	[0,-S,C]
23	[0,-S,0]
24	[C,0,-S]
25	[0,C,-S]
26	[-S,-S,C]
27	[0,0,-S]
28	[C,0,0]
29	[0,C,0]
30	[0,0,C]
31	[-S,-S,-S]

Notes:

The units of the angle which is provided as input for the vrot.t instruction is such that 1.0 represents π/2 in radians and 90° in degrees.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
ts <- approx_sin( M_PI_2 * s[0] );
tc <- approx_cos( M_PI_2 * s[0] );
d[0] <- 0;
d[1] <- 0;
d[2] <- 0;
if( imm5[4] )
    ts <- -ts;
if( imm5[3:2]==imm5[1:0] )
    begin
        d[0] <- ts;
        d[1] <- ts;
        d[2] <- ts;
    end
else if( imm5[3:2] <3)
    d[imm5[3:2]] <- ts;
if( imm5[1:0] <3)
    d[(imm5[1:0]] <- tc;
WriteMatrix( TRIPLEWORD, vd, d );
```

vrot.q

Rotator Quad Word

31 30 29 28 27 26	25 24 23 22 21	20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	1 1 1 0 1	imm5	1	vs	1	vd
6	5	5	1	7	1	7

VFPU

Syntax:

```
vrot.q  vd, vs, imm5
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8　　　　pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The rotators indicated by the imm5 field are calculated for the floating-point values of four elements from the matrix registers indicated by vs. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.
If the write positions overlap, the cos values are overwritten.
The following mnemonics can be used for imm5.

Code (imm5)	Mnemonic
0	[C,S,S,S]
1	[S,C,0,0]
2	[S,0,C,0]
3	[S,0,0,C]
4	[C,S,0,0]
5	[S,C,S,S]
6	[0,S,C,0]
7	[0,S,0,C]
8	[C,0,S,0]
9	[0,C,S,0]

Code (imm5)	Mnemonic
10	[S,S,C,S]
11	[0,0,S,C]
12	[C,0,0,S]
13	[0,C,0,S]
14	[0,0,C,S]
15	[S,S,S,C]
16	[C,-S,-S,-S]
17	[-S,C,0,0]
18	[-S,0,C,0]
19	[-S,0,0,C]
20	[C,-S,0,0]
21	[-S,C,-S,-S]
22	[0,-S,C,0]
23	[0,-S,0,C]
24	[C,0,-S,0]
25	[0,C,-S,0]
26	[-S,-S,C,-S]
27	[0,0,-S,C]
28	[C,0,0,-S]
29	[0,C,0,-S]
30	[0,0,C,-S]
31	[-S,-S,-S,C]

Notes:

The units of the angle which is provided as input for the vrot.q instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
ts <- approx_sin( M_PI_2 * s[0] );
tc <- approx_cos( M_PI_2 * s[0] );
d[0] <- 0;
d[1] <- 0;
d[2] <- 0;
d[3] <- 0;
if( imm5[4] )
    ts <- -ts;
if( imm5[3:2]==imm5[1:0] )
    begin
       d[0] <- ts;
       d[1] <- ts;
       d[2] <- ts;
       d[3] <- ts;
    end
else if( imm5[3:2] <4)
    d[imm5[3:2]] <- ts;
if( imm5[1:0] <4)
    d[(imm5[1:0]] <- tc;
WriteMatrix( QUADWORD, vd, d );
```

vrsq.s

Reciprocal Square Root Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 0 1 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vrsq.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The reciprocal of the square root of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$|\ (\mathit{approx\_reciprocal\_sqrt}(x) - (1/\sqrt{x}))\ /\ (1/\sqrt{x})\ | < 7.3 \times 10\text{-}7$, $0.0 <= |\ x\ | < \mathit{inf}$

Special solutions are as follows.

approx_reciprocal_sqrt(+***nan***) = +***nan***

approx_reciprocal_sqrt(-***nan***) = -***nan***

approx_reciprocal_sqrt(+***inf***) = +0.0

approx_reciprocal_sqrt(-***inf***) = +***nan***

approx_reciprocal_sqrt(+0.0) = +***inf***

approx_reciprocal_sqrt(-0.0) = -***inf***

approx_reciprocal_sqrt(***x***) = +***nan***; -***inf*** < ***x*** < -0.0

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- approx_reciprocal_sqrt( s[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vrsq.p

Reciprocal Square Root Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 0 1 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vrsq.p  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8   pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The reciprocals of the square roots of the floating-point values of two elements from the matrix registers indicated by vs are calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$|\ (\textit{approx\_reciprocal\_sqrt}(x) - (1/\sqrt{x})) / (1/\sqrt{x})\ | < 7.3 \times 10\text{-}7,\ 0.0 <= |\ x\ | < \textit{inf}$

Special solutions are as follows.

approx_reciprocal_sqrt(+***nan***) = +***nan***

approx_reciprocal_sqrt(-***nan***) = -***nan***

approx_reciprocal_sqrt(+***inf***) = +0.0

approx_reciprocal_sqrt(-***inf***) = +***nan***

approx_reciprocal_sqrt(+0.0) = +***inf***

approx_reciprocal_sqrt(-0.0) = -***inf***

approx_reciprocal_sqrt(***x***) = +***nan***; -***inf*** < ***x*** < -0.0

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- approx_reciprocal_sqrt( s[0] );
d[1] <- approx_reciprocal_sqrt( s[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vrsq.t

Reciprocal Square Root Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 0 1 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vrsq.t  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The reciprocals of the square roots of the floating-point values of three elements from the matrix registers indicated by vs are calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| (\boldsymbol{approx\_reciprocal\_sqrt}(\boldsymbol{x}) - (1/\sqrt{\boldsymbol{x}})) / (1/\sqrt{\boldsymbol{x}}) | < 7.3 \times 10\text{-}7$, $0.0 <= | \boldsymbol{x} | < \boldsymbol{inf}$

Special solutions are as follows.

approx_reciprocal_sqrt(+***nan***) = +***nan***

approx_reciprocal_sqrt(-***nan***) = -***nan***

approx_reciprocal_sqrt(+***inf***) = +0.0

approx_reciprocal_sqrt(-***inf***) = +***nan***

approx_reciprocal_sqrt(+0.0) = +***inf***

approx_reciprocal_sqrt(-0.0) = -***inf***

approx_reciprocal_sqrt(***x***) = +***nan***; -***inf*** < ***x*** < -0.0

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- approx_reciprocal_sqrt( s[0] );
d[1] <- approx_reciprocal_sqrt( s[1] );
d[2] <- approx_reciprocal_sqrt( s[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vrsq.q

Reciprocal Square Root Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 0 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vrsq.q  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10　　　pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The reciprocals of the square roots of the floating-point values of four elements from the matrix registers indicated by vs are calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$|\,(\boldsymbol{approx\_reciprocal\_sqrt}(\boldsymbol{x}) - (1/\sqrt{\boldsymbol{x}}))\,/\,(1/\sqrt{\boldsymbol{x}})\,| < 7.3 \times 10\text{-}7$, $0.0 <= |\,\boldsymbol{x}\,| < \boldsymbol{inf}$

Special solutions are as follows.

approx_reciprocal_sqrt(+***nan***) = +***nan***

approx_reciprocal_sqrt(-***nan***) = -***nan***

approx_reciprocal_sqrt(+***inf***) = +0.0

approx_reciprocal_sqrt(-***inf***) = +***nan***

approx_reciprocal_sqrt(+0.0) = +***inf***

approx_reciprocal_sqrt(-0.0) = -***inf***

approx_reciprocal_sqrt(***x***) = +***nan***; -***inf*** < ***x*** < -0.0

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- approx_reciprocal_sqrt( s[0] );
d[1] <- approx_reciprocal_sqrt( s[1] );
d[2] <- approx_reciprocal_sqrt( s[2] );
d[3] <- approx_reciprocal_sqrt( s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vs2i.s

Convert signed short to integer Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 1 0 1 1 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vs2i.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

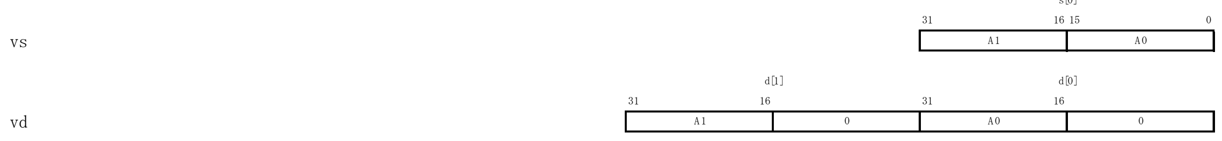
vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Only write mask is valid

Description:

The 32-bit packed data from the matrix register indicated by vs is unpacked and converted from two signed 16-bit integers to two signed 32-bit integers. The two-element integer result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- {s[0][15: 0], 16'b0};
d[1] <- {s[0][31:16], 16'b0};
WriteMatrix( PAIRWORD, vd, d );
```



vs2i.p

Convert signed short to integer Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 1 0 1 1 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vs2i.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　pitch : 1

Prefixing:

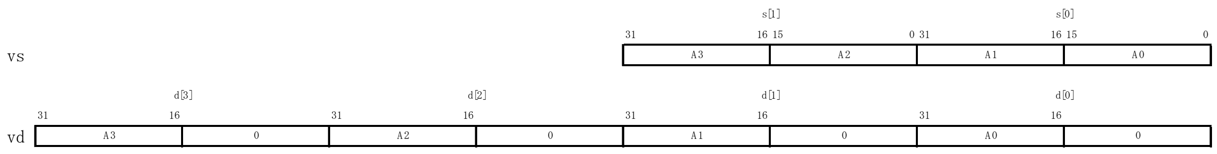
vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Only write mask is valid

Description:

The 64-bit packed data from the matrix registers indicated by vs is unpacked and converted from four signed 16-bit integers to four signed 32-bit integers. The four-element integer result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- {s[0][15: 0], 16'b0};
d[1] <- {s[0][31:16], 16'b0};
d[2] <- {s[1][15: 0], 16'b0};
d[3] <- {s[1][31:16], 16'b0};
WriteMatrix( QUADWORD, vd, d );
```



vsat0.s

Saturate to [0.0:1.0] Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 0 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vsat0.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Use prohibited

Description:

If the floating-point value of one element from the matrix register indicated by vs is less than 0.0, it is saturated to 0.0. If the value is greater than 1.0, it is saturated to 1.0. Values between 0.0 and 1.0 are not changed. The saturated result is stored as a one-element floating-point value at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- (s[0] < 0.0) ? 0.0 : ( (s[0] > 1.0) ? 1.0 : s[0]);
WriteMatrix( SINGLEWORD, vd, d );
```

vsat0.p

Saturate to [0.0:1.0] Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 0 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsat0.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Use prohibited

Description:

If any of the floating-point values of two elements from the matrix registers indicated by vs is less than 0.0, it is saturated to 0.0. If any of the values is greater than 1.0, it is saturated to 1.0. Values between 0.0 and 1.0 are not changed. The saturated result is stored as a two-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- (s[0] < 0.0) ? 0.0 : ( (s[0] > 1.0) ? 1.0 : s[0]);
d[1] <- (s[1] < 0.0) ? 0.0 : ( (s[1] > 1.0) ? 1.0 : s[1]);
WriteMatrix( PAIRWORD, vd, d );
```

vsat0.t

Saturate to [0.0:1.0] Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 0 0 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vsat0.t  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3   pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Use prohibited

Description:

If any of the floating-point values of three elements from the matrix registers indicated by vs is less than 0.0, it is saturated to 0.0. If any of the values is greater than 1.0, it is saturated to 1.0. Values between 0.0 and 1.0 are not changed. The saturated result is stored as a three-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- (s[0] < 0.0) ? 0.0 : ( (s[0] > 1.0) ? 1.0 : s[0]);
d[1] <- (s[1] < 0.0) ? 0.0 : ( (s[1] > 1.0) ? 1.0 : s[1]);
d[2] <- (s[2] < 0.0) ? 0.0 : ( (s[2] > 1.0) ? 1.0 : s[2]);
WriteMatrix( TRIPLEWORD, vd, d );
```

vsat0.q

Saturate to [0.0:1.0] Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 0 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsat0.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Use prohibited

Description:

If any of the floating-point values of four elements from the matrix registers indicated by vs is less than 0.0, it is saturated to 0.0. If any of the values is greater than 1.0, it is saturated to 1.0. Values between 0.0 and 1.0 are not changed. The saturated result is stored as a four-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- (s[0] < 0.0) ? 0.0 : ( (s[0] > 1.0) ? 1.0 : s[0]);
d[1] <- (s[1] < 0.0) ? 0.0 : ( (s[1] > 1.0) ? 1.0 : s[1]);
d[2] <- (s[2] < 0.0) ? 0.0 : ( (s[2] > 1.0) ? 1.0 : s[2]);
d[3] <- (s[3] < 0.0) ? 0.0 : ( (s[3] > 1.0) ? 1.0 : s[3]);
WriteMatrix( QUADWORD, vd, d );
```

vsat1.s

Saturate to [-1.0:1.0] Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 0 1 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vsat1.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Use prohibited

Description:

If the floating-point value of one element from the matrix register indicated by vs is less than -1.0, it is saturated to -1.0. If the value is greater than 1.0, it is saturated to 1.0. Values between -1.0 and 1.0 are not changed. The saturated result is stored as a one-element floating-point value at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- (s[0] < -1.0) ? -1.0 : ( (s[0] > 1.0) ? 1.0 : s[0]);
WriteMatrix( SINGLEWORD, vd, d );
```

vsat1.p

Saturate to [-1.0:1.0] Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 0 1 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsat1.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Use prohibited

Description:

If any of the floating-point values of two elements from the matrix registers indicated by vs is less than -1.0, it is saturated to -1.0. If any of the values is greater than 1.0, it is saturated to 1.0. Values between -1.0 and 1.0 are not changed. The saturated result is stored as a two-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- (s[0] < -1.0) ? -1.0 : ( (s[0] > 1.0) ? 1.0 : s[0]);
d[1] <- (s[1] < -1.0) ? -1.0 : ( (s[1] > 1.0) ? 1.0 : s[1]);
WriteMatrix( PAIRWORD, vd, d );
```

vsat1.t

Saturate to [-1.0:1.0] Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 0 1 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vsat1.t  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Use prohibited

Description:

If any of the floating-point values of three elements from the matrix registers indicated by vs is less than -1.0, it is saturated to -1.0. If any of the values is greater than 1.0, it is saturated to 1.0. Values between -1.0 and 1.0 are not changed. The saturated result is stored as a three-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- (s[0] < -1.0) ? -1.0 : ( (s[0] > 1.0) ? 1.0 : s[0]);
d[1] <- (s[1] < -1.0) ? -1.0 : ( (s[1] > 1.0) ? 1.0 : s[1]);
d[2] <- (s[2] < -1.0) ? -1.0 : ( (s[2] > 1.0) ? 1.0 : s[2]);
WriteMatrix( TRIPLEWORD, vd, d );
```

vsat1.q

Saturate to [-1.0:1.0] Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 0 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsat1.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Use prohibited

Description:

If any of the floating-point values of four elements from the matrix registers indicated by vs is less than -1.0, it is saturated to -1.0. If any of the values is greater than 1.0, it is saturated to 1.0. Values between -1.0 and 1.0 are not changed. The saturated result is stored as a four-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- (s[0] < -1.0) ? -1.0 : ( (s[0] > 1.0) ? 1.0 : s[0]);
d[1] <- (s[1] < -1.0) ? -1.0 : ( (s[1] > 1.0) ? 1.0 : s[1]);
d[2] <- (s[2] < -1.0) ? -1.0 : ( (s[2] > 1.0) ? 1.0 : s[2]);
d[3] <- (s[3] < -1.0) ? -1.0 : ( (s[3] > 1.0) ? 1.0 : s[3]);
WriteMatrix( QUADWORD, vd, d );
```

vsbn.s

ScaleBN Single Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 0	0 1 0	vt	0	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vsbn.s  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

The scaleBN calculation (change the binary logarithmic scale) is performed on the floating-point value of one element from the matrix register indicated by vs using the integer value of one element of the matrix register indicated by vt. The one-element floating-point result is stored at the location in the matrix register file indicated by vd. The input range of vt using two's complement representation is given by the following expression. Moreover, ***scaleBN***() is defined as

```
scaleBN(s, t) = ((*((int *)&s) & 0x807FFFFF) | ((((*(int *)(&t)) + 127) & 0xFF)<<23)
```

-127 <= ***x*** <= 127

Special solutions are as follows.

scaleBN(***nan***, ***t***) = ***nan***

scaleBN(+***inf***, ***t***) = +***inf***

scaleBN(-***inf***, ***t***) = -***inf***

scaleBN(+0.0, ***t***) = +0.0

scaleBN(-0.0, ***t***) = -0.0

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <- scaleBN( s[0], t[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vsbz.s

ScaleBZ Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 0 1 1 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vsbz.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The remainder from the [1.0 ... 2.0] interval of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

scaleBZ is defined as follows: $\boldsymbol{x} = \boldsymbol{scaleBZ}(\boldsymbol{x}) * 2^{\boldsymbol{logB}(\boldsymbol{x})}$; $1 <= \boldsymbol{scaleBZ}(\boldsymbol{x}) < 2$.

Special solutions are as follows.

scaleBZ(***nan***) = ***nan***

scaleBZ(+***inf***) = +1.0

scaleBZ(-***inf***) = -1.0

scaleBZ(+0.0) = +0.0

scaleBZ(-0.0) = -0.0

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- scaleBZ( s[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vscl.p

Scale Pair Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	0 1 0	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vscl.p  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

Two elements from the matrix registers indicated by vs are each multiplied by one element from the matrix register indicated by vt. The elements are treated as floating-point numbers. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <- s[0] * t[0];
d[1] <- s[1] * t[0];
WriteMatrix( PAIRWORD, vd, d );
```

vscl.t

Scale Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	0 1 0	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vscl.t  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

Three elements from the matrix registers indicated by vs are each multiplied by one element from the matrix register indicated by vt. The elements are treated as floating-point numbers. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <- s[0] * t[0];
d[1] <- s[1] * t[0];
d[2] <- s[2] * t[0];
WriteMatrix( TRIPLEWORD, vd, d );
```

vscl.q

Scale Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 1	0 1 0	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vscl.q  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

Four elements from the matrix registers indicated by vs are each multiplied by one element from the matrix register indicated by vt. The elements are treated as floating-point numbers. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <- s[0] * t[0];
d[1] <- s[1] * t[0];
d[2] <- s[2] * t[0];
d[3] <- s[3] * t[0];
WriteMatrix( QUADWORD, vd, d );
```

vscmp.s

Set Compare Single Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	1 0 1	vt	0	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vscmp.s  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

One element from the matrix register indicated by vs is compared with one element from the matrix register indicated by vt, as floating-point numbers. If vs is equal to vt, the result is set to 0.0. If vs is greater than vt, the result is set to 1.0. If vs is less than vt, the result is set to -1.0. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Notes:

For the vscmp.s instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-inf	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-1.0	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-0.0	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
+0.0	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***
+1.0	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***
+inf	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***
+nan	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <- (s[0]>t[0]) ? 1.0 : ((s[0]<t[0]) ? -1.0 : 0.0);
WriteMatrix( SINGLEWORD, vd, d );
```

vscmp.p

Set Compare Pair Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	1 0 1	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vscmp.p  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Two elements from the matrix registers indicated by vs are compared with two elements from the matrix registers indicated by vt, as floating-point numbers. For each pair of elements, if vs is equal to vt, the result is set to 0.0. If vs is greater than vt, the result is set to 1.0. If vs is less than vt, the result is set to -1.0. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vscmp.p instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-inf	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-1.0	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-0.0	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
+0.0	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***
+1.0	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***
+inf	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***
+nan	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vt );
d[0] <- (s[0]>t[0]) ? 1.0 : ((s[0]<t[0]) ? -1.0 : 0.0);
d[1] <- (s[1]>t[1]) ? 1.0 : ((s[1]<t[1]) ? -1.0 : 0.0);
WriteMatrix( PAIRWORD, vd, d );
```

vscmp.t

Set Compare Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	1 0 1	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vscmp.t  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Three elements from the matrix registers indicated by vs are compared with three elements from the matrix registers indicated by vt, as floating-point numbers. For each pair of elements, if vs is equal to vt, the result is set to 0.0. If vs is greater than vt, the result is set to 1.0. If vs is less than vt, the result is set to -1.0. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vscmp.t instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-inf	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-1.0	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-0.0	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
+0.0	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***
+1.0	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***
+inf	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***
+nan	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- (s[0]>t[0]) ? 1.0 : ((s[0]<t[0]) ? -1.0 : 0.0);
d[1] <- (s[1]>t[1]) ? 1.0 : ((s[1]<t[1]) ? -1.0 : 0.0);
d[2] <- (s[2]>t[2]) ? 1.0 : ((s[2]<t[2]) ? -1.0 : 0.0);
WriteMatrix( TRIPLEWORD, vd, d );
```

vscmp.q

Set Compare Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	1 0 1	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vscmp.q  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Four elements from the matrix registers indicated by vs are compared with four elements from the matrix registers indicated by vt, as floating-point numbers. For each pair of elements, if vs is equal to vt, the result is set to 0.0. If vs is greater than vt, the result is set to 1.0. If vs is less than vt, the result is set to -1.0. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vscmp.q instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-inf	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-1.0	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
-0.0	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***	***-1.0***
+0.0	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***	***-1.0***
+1.0	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***	***-1.0***
+inf	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***-1.0***
+nan	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
s[0]<t[0]) ? -1.0 : 0.0);
d[1] <- (s[1]>t[1]) ? 1.0 : ((s[1]<t[1]) ? -1.0 : 0.0);
d[2] <- (s[2]>t[2]) ? 1.0 : ((s[2]<t[2]) ? -1.0 : 0.0);
d[3] <- (s[3]>t[3]) ? 1.0 : ((s[3]<t[3]) ? -1.0 : 0.0);
WriteMatrix( QUADWORD, vd, d );
```

vsge.s

Greater Equal Single Word to Value

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	1 1 0	vt	0	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vsge.s  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3   pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

One element from the matrix register indicated by vs is compared with one element from the matrix register indicated by vt, as floating-point numbers. If vs is greater than or equal to vt, the result is set to 1.0. If vs is less than vt, the result is set to 0.0. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Notes:

For the vsge.s instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-inf	***+0.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-1.0	***+0.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-0.0	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***
+0.0	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***
+1.0	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***
+inf	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***
+nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <- ( s[0] >= t[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vsge.p

Greater Equal Pair Word to Value

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	1 1 0	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vsge.p  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Two elements from the matrix registers indicated by vs are compared with two elements from the matrix registers indicated by vt, as floating-point numbers. For each pair of elements, if vs is greater than or equal to vt, the result is set to 1.0. If vs is less than vt, the result is set to 0.0. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vsge.p instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-inf	***+0.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-1.0	***+0.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-0.0	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***
+0.0	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***
+1.0	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***
+inf	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***
+nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vt );
d[0] <- ( s[0] >= t[0] );
d[1] <- ( s[1] >= t[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vsge.t

Greater Equal Triple Word to Value

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	1 1 0	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vsge.t  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Three elements from the matrix registers indicated by vs are compared with three elements from the matrix registers indicated by vt, as floating-point numbers. For each pair of elements, if vs is greater than or equal to vt, the result is set to 1.0. If vs is less than vt, the result is set to 0.0. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vsge.t instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-inf	***+0.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-1.0	***+0.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-0.0	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***
+0.0	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***
+1.0	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***
+inf	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***
+nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- ( s[0] >= t[0] );
d[1] <- ( s[1] >= t[1] );
d[2] <- ( s[2] >= t[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vsge.q

Greater Equal Quad Word to Value

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	1 1 0	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vsge.q  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Four elements from the matrix registers indicated by vs are compared with four elements from the matrix registers indicated by vt, as floating-point numbers. For each pair of elements, if vs is greater than or equal to vt, the result is set to 1.0. If vs is less than vt, the result is set to 0.0. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vsge.q instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-inf	***+0.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-1.0	***+0.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-0.0	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***
+0.0	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***	***+0.0***
+1.0	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***	***+0.0***
+inf	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***
+nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
d[0] <- ( s[0] >= t[0] );
d[1] <- ( s[1] >= t[1] );
d[2] <- ( s[2] >= t[2] );
d[3] <- ( s[3] >= t[3] );
WriteMatrix( QUADWORD, vd, d );
```

vsgn.s

Sign Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 1 0 1 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vsgn.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

One element from the matrix register indicated by vs is examined. If the floating-point value of the element is less than 0.0, equal to 0.0, or greater than 0.0, the result is set to -1.0, 0.0, or 1.0, respectively. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Special solutions are as follows.

sgn(+***nan***) = +1.0

sgn(-***nan***) = -1.0

sgn(+***inf***) = +1.0

sgn(-***inf***) = -1.0

sgn(+0.0) = +0.0

sgn(-0.0) = +0.0

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- (s[0]>0.0) ? 1.0 : ((s[0]<0.0) -1.0 : 0.0);
WriteMatrix( SINGLEWORD, vd, d );
```

vsgn.p

Sign Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 1 0 1 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsgn.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

Two elements from the matrix registers indicated by vs are examined. If the floating-point value of each of the elements is less than 0.0, equal to 0.0, or greater than 0.0, the result is set to -1.0, 0.0, or 1.0, respectively, in the corresponding element. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Special solutions are as follows.

sgn(+***nan***) = +1.0

sgn(-***nan***) = -1.0

sgn(+***inf***) = +1.0

sgn(-***inf***) = -1.0

sgn(+0.0) = +0.0

sgn(-0.0) = +0.0

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- (s[0]>0.0) ? 1.0 : ((s[0]<0.0) -1.0 : 0.0);
d[1] <- (s[1]>0.0) ? 1.0 : ((s[1]<0.0) -1.0 : 0.0);
WriteMatrix( PAIRWORD, vd, d );
```

vsgn.t

Sign Triple Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	0	0	1	0	0	1	0	1	0	1				vs				0				vd			
			6								11									7				1				7			

VFPU

Syntax:

```
vsgn.t  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

Three elements from the matrix registers indicated by vs are examined. If the floating-point value of each of the elements is less than 0.0, equal to 0.0, or greater than 0.0, the result is set to -1.0, 0.0, or 1.0, respectively, in the corresponding element. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Special solutions are as follows.

sgn(+***nan***) = +1.0

sgn(-***nan***) = -1.0

sgn(+***inf***) = +1.0

sgn(-***inf***) = -1.0

sgn(+0.0) = +0.0

sgn(-0.0) = +0.0

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- (s[0]>0.0) ? 1.0 : ((s[0]<0.0) -1.0 : 0.0);
d[1] <- (s[1]>0.0) ? 1.0 : ((s[1]<0.0) -1.0 : 0.0);
d[2] <- (s[2]>0.0) ? 1.0 : ((s[2]<0.0) -1.0 : 0.0);
WriteMatrix( TRIPLEWORD, vd, d );
```

vsgn.q

Sign Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 1 0 1 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsgn.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Use prohibited	Valid

Description:

Four elements from the matrix registers indicated by vs are examined. If the floating-point value of each of the elements is less than 0.0, equal to 0.0, or greater than 0.0, the result is set to -1.0, 0.0, or 1.0, respectively, in the corresponding element. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Special solutions are as follows.

sgn(+***nan***) = +1.0

sgn(-***nan***) = -1.0

sgn(+***inf***) = +1.0

sgn(-***inf***) = -1.0

sgn(+0.0) = +0.0

sgn(-0.0) = +0.0

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- (s[0]>0.0) ? 1.0 : ((s[0]<0.0) -1.0 : 0.0);
d[1] <- (s[1]>0.0) ? 1.0 : ((s[1]<0.0) -1.0 : 0.0);
d[2] <- (s[2]>0.0) ? 1.0 : ((s[2]<0.0) -1.0 : 0.0);
d[3] <- (s[3]>0.0) ? 1.0 : ((s[3]<0.0) -1.0 : 0.0);
WriteMatrix( QUADWORD, vd, d );
```

vsin.s

Sine Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 1 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vsin.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 7 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The sine of the floating-point value of one element from the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \boldsymbol{approx\_sin}(\boldsymbol{M\_PI\_}2 \times \boldsymbol{x}) - \boldsymbol{sin}(\boldsymbol{M\_PI\_}2 \times \boldsymbol{x}) | < 4.8 \times 10^{-7}$; $0.0 <= | \boldsymbol{x} | < 32.0$

$| \boldsymbol{approx\_sin}(\boldsymbol{M\_PI\_}2 \times \boldsymbol{x}) - \boldsymbol{sin}(\boldsymbol{M\_PI\_}2 \times \boldsymbol{x}) | < 2.0$; $32.0 <= | \boldsymbol{x} | < \boldsymbol{inf}$

Special solutions are as follows.

approx_sin(+***nan***) = +***nan***

approx_sin(-***nan***) = -***nan***

approx_sin(+***inf***) = +***nan***

approx_sin(-***inf***) = -***nan***

approx_sin(+0.0) = +0.0

approx_sin(-0.0) = -0.0

Notes:

The units of the angle which is provided as input for the vsin.s instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- approx_sin( M_PI_2 * s[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vsin.p

Sine Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 1 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsin.p  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The sines of the floating-point values of two elements from the matrix registers indicated by vs are calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \boldsymbol{approx\_sin}(\boldsymbol{M\_PI}\_2 \times \boldsymbol{x}) - \boldsymbol{sin}(\boldsymbol{M\_PI}\_2 \times \boldsymbol{x}) | < 4.8 \times 10^{-7}$; $0.0 <= | \boldsymbol{x} | < 32.0$

$| \boldsymbol{approx\_sin}(\boldsymbol{M\_PI}\_2 \times \boldsymbol{x}) - \boldsymbol{sin}(\boldsymbol{M\_PI}\_2 \times \boldsymbol{x}) | < 2.0$; $32.0 <= | \boldsymbol{x} | < \boldsymbol{inf}$

Special solutions are as follows.

approx_sin(+***nan***) = +***nan***

approx_sin(-***nan***) = -***nan***

approx_sin(+***inf***) = +***nan***

approx_sin(-***inf***) = -***nan***

approx_sin(+0.0) = +0.0

approx_sin(-0.0) = -0.0

Notes:

The units of the angle which is provided as input for the vsin.p instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- approx_sin( M_PI_2 * s[0] );
d[1] <- approx_sin( M_PI_2 * s[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vsin.t

Sine Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 1 0 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vsin.t  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The sines of the floating-point values of three elements from the matrix registers indicated by vs are calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \mathit{approx\_sin}(\mathit{M\_PI\_}2 \times x) - \mathit{sin}(\mathit{M\_PI\_}2 \times x) | < 4.8 \times 10^{-7}$; $0.0 <= | x | < 32.0$

$| \mathit{approx\_sin}(\mathit{M\_PI\_}2 \times x) - \mathit{sin}(\mathit{M\_PI\_}2 \times x) | < 2.0$; $32.0 <= | x | < \mathit{inf}$

Special solutions are as follows.

approx_sin(+***nan***) = +***nan***

approx_sin(-***nan***) = -***nan***

approx_sin(+***inf***) = +***nan***

approx_sin(-***inf***) = -***nan***

approx_sin(+0.0) = +0.0

approx_sin(-0.0) = -0.0

Notes:

The units of the angle which is provided as input for the vsin.t instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- approx_sin( M_PI_2 * s[0] );
d[1] <- approx_sin( M_PI_2 * s[1] );
d[2] <- approx_sin( M_PI_2 * s[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vsin.q

Sine Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 0 1 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsin.q  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10 pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The sines of the floating-point values of four elements from the matrix registers indicated by vs are calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| \textit{\textbf{approx\_sin}}(\textit{\textbf{M\_PI\_}}2 \times \boldsymbol{x}) - \textit{\textbf{sin}}(\textit{\textbf{M\_PI\_}}2 \times \boldsymbol{x}) | < 4.8 \times 10^{-7}$; $0.0 <= | \boldsymbol{x} | < 32.0$

$| \textit{\textbf{approx\_sin}}(\textit{\textbf{M\_PI\_}}2 \times \boldsymbol{x}) - \textit{\textbf{sin}}(\textit{\textbf{M\_PI\_}}2 \times \boldsymbol{x}) | < 2.0$; $32.0 <= | \boldsymbol{x} | < \textit{\textbf{inf}}$

Special solutions are as follows.

approx_sin(+***nan***) = +***nan***

approx_sin(-***nan***) = -***nan***

approx_sin(+***inf***) = +***nan***

approx_sin(-***inf***) = -***nan***

approx_sin(+0.0) = +0.0

approx_sin(-0.0) = -0.0

Notes:

The units of the angle which is provided as input for the vsin.q instruction is such that 1.0 represents π/2 in radians and 90º in degrees.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- approx_sin( M_PI_2 * s[0] );
d[1] <- approx_sin( M_PI_2 * s[1] );
d[2] <- approx_sin( M_PI_2 * s[2] );
d[3] <- approx_sin( M_PI_2 * s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vslt.s

Less Than Single Word to Value

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	1 1 1	vt	0	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vslt.s  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

One element from the matrix register indicated by vs is compared with one element from the matrix register indicated by vt, as floating-point numbers. If vs is less than vt, the result is set to 1.0. If vs is greater than or equal to vt, the result is set to 0.0. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Notes:

For the vslt.s instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-inf	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***
-1.0	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***
-0.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+0.0***
+0.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+0.0***
+1.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+0.0***
+inf	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+0.0***
+nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <-( s[0] < t[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vslt.p

Less Than Pair Word to Value

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	1 1 1	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vslt.p  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Two elements from the matrix registers indicated by vs are compared with two elements from the matrix registers indicated by vt, as floating-point numbers. For each pair of elements, if vs is less than vt, the result is set to 1.0. If vs is greater than or equal to vt, the result is set to 0.0. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vslt.p instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-inf	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***
-1.0	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***
-0.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+0.0***
+0.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+0.0***
+1.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+0.0***
+inf	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+0.0***
+nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vt );
d[0] <- ( s[0] < t[0] );
d[1] <- ( s[1] < t[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vslt.t

Less Than Triple Word to Value

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	1 1 1	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vslt.t  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Three elements from the matrix registers indicated by vs are compared with three elements from the matrix registers indicated by vt, as floating-point numbers. For each pair of elements, if vs is less than vt, the result is set to 1.0. If vs is greater than or equal to vt, the result is set to 0.0. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vslt.t instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-inf	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***
-1.0	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***
-0.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+0.0***
+0.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+0.0***
+1.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+0.0***
+inf	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+0.0***
+nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- ( s[0] < t[0] );
d[1] <- ( s[1] < t[1] );
d[2] <- ( s[2] < t[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vslt.q

Less Than Quad Word to Value

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 1 1	1 1 1	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vslt.q  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Four elements from the matrix registers indicated by vs are compared with four elements from the matrix registers indicated by vt, as floating-point numbers. For each pair of elements, if vs is less than vt, the result is set to 1.0. If vs is greater than or equal to vt, the result is set to 0.0. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

For the vslt.q instruction, a comparison which includes the sign bit of a NaN is performed. The relative magnitudes used in the comparison are as follows.
-***nan*** < -***inf*** < -***num*** < -0.0 = +0.0 < +***num*** < +***inf*** < +***nan***

The relationship between the input value and the output value is as follows.

src \tar	***-nan***	***-inf***	***-1.0***	***-0.0***	***+0.0***	***+1.0***	***+inf***	***+nan***
-nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***
-inf	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***
-1.0	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+1.0***	***+1.0***	***+0.0***
-0.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+0.0***
+0.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+0.0***
+1.0	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+1.0***	***+0.0***
+inf	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+1.0***	***+0.0***
+nan	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***	***+0.0***

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
d[0] <- ( s[0] < t[0] );
d[1] <- ( s[1] < t[1] );
d[2] <- ( s[2] < t[2] );
d[3] <- ( s[3] < t[3] );
WriteMatrix( QUADWORD, vd, d );
```

vsocp.s

Saturate and One's Complement Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 1 0 1 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vsocp.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The one's complement of the floating-point value of one element from the matrix register indicated by vs is calculated. The original value of the element and its one's complement are each saturated to [0.0:1.0] and stored as a two-element floating-point result at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- 1.0 - s[0];
d[0] <- (d[0] < 0.0) ? 0.0 : ( (d[0] > 1.0) ? 1.0 : d[0]);
d[1] <- s[0];
d[1] <- (d[1] < 0.0) ? 0.0 : ( (d[1] > 1.0) ? 1.0 : d[1]);
WriteMatrix( PAIRWORD, vd, d );
```

vsocp.p

Saturate and One's Complement Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 1 0 1 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsocp.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The one's complements of the floating-point values of two elements from the matrix registers indicated by vs are calculated. The original values of the elements and their one's complements are each saturated to [0.0:1.0] and stored as a four-element floating-point result at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- 1.0 - s[0];
d[0] <- (d[0] < 0.0) ? 0.0 : ( (d[0] > 1.0) ? 1.0 : d[0]);
d[1] <- s[0];
d[1] <- (d[1] < 0.0) ? 0.0 : ( (d[1] > 1.0) ? 1.0 : d[1]);
d[2] <- 1.0 - s[1];
d[2] <- (d[2] < 0.0) ? 0.0 : ( (d[2] > 1.0) ? 1.0 : d[2]);
d[3] <- s[1];
d[3] <- (d[3] < 0.0) ? 0.0 : ( (d[3] > 1.0) ? 1.0 : d[3]);
WriteMatrix( QUADWORD, vd, d );
```

vsqrt.s

Square Root Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 1 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vsqrt.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency：7　　　　pitch：1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The square root of the floating-point value of one element of the matrix register indicated by vs is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| (\boldsymbol{approx\_sqrt}(\boldsymbol{x}) - \boldsymbol{sqrt}(\boldsymbol{x})) / \boldsymbol{sqrt}(\boldsymbol{x}) | < 7.1 \times 10^{-7}$;

Special solutions are as follows.

approx_sqrt(+***nan***) = +***nan***

approx_sqrt(-***nan***) = +***nan***

approx_sqrt(+***inf***) = +***inf***

approx_sqrt(-***inf***) = +***nan***

approx_sqrt(+0.0) = +0.0

approx_sqrt(-0.0) = +0.0

approx_sqrt(***x***) = +***nan***; -***inf*** < ***x*** < -0.0

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- approx_sqrt( s[0] );
WriteMatrix( SINGLEWORD, vd, d );
```

vsqrt.p

Square Root Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 1 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsqrt.p  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The square roots of the floating-point values of two elements of the matrix registers indicated by vs are calculated. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| (\boldsymbol{approx\_sqrt}(\boldsymbol{x}) - \boldsymbol{sqrt}(\boldsymbol{x})) / \boldsymbol{sqrt}(\boldsymbol{x}) | < 7.1 \times 10^{-7}$;

Special solutions are as follows.

$\boldsymbol{approx\_sqrt}(+\boldsymbol{nan}) = +\boldsymbol{nan}$

$\boldsymbol{approx\_sqrt}(-\boldsymbol{nan}) = +\boldsymbol{nan}$

$\boldsymbol{approx\_sqrt}(+\boldsymbol{inf}) = +\boldsymbol{inf}$

$\boldsymbol{approx\_sqrt}(-\boldsymbol{inf}) = +\boldsymbol{nan}$

$\boldsymbol{approx\_sqrt}(+0.0) = +0.0$

$\boldsymbol{approx\_sqrt}(-0.0) = +0.0$

$\boldsymbol{approx\_sqrt}(\boldsymbol{x}) = +\boldsymbol{nan}$; $-\boldsymbol{inf} < \boldsymbol{x} < -0.0$

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- approx_sqrt( s[0] );
d[1] <- approx_sqrt( s[1] );
WriteMatrix( PAIRWORD, vd, d );
```

vsqrt.t

Square Root Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 1 0 1	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vsqrt.t  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9　　　　pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The square roots of the floating-point values of three elements of the matrix registers indicated by vs are calculated. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$| (\boldsymbol{approx\_sqrt}(\boldsymbol{x}) - \boldsymbol{sqrt}(\boldsymbol{x})) / \boldsymbol{sqrt}(\boldsymbol{x}) | < 7.1 \times 10^{-7}$;

Special solutions are as follows.

approx_sqrt(+***nan***) = +***nan***

approx_sqrt(-***nan***) = +***nan***

approx_sqrt(+***inf***) = +***inf***

approx_sqrt(-***inf***) = +***nan***

approx_sqrt(+0.0) = +0.0

approx_sqrt(-0.0) = +0.0

approx_sqrt(***x***) = +***nan***; -***inf*** < ***x*** < -0.0

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
d[0] <- approx_sqrt( s[0] );
d[1] <- approx_sqrt( s[1] );
d[2] <- approx_sqrt( s[2] );
WriteMatrix( TRIPLEWORD, vd, d );
```

vsqrt.q

Square Root Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 1 0 1 1 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsqrt.q  vd, vs
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10　　　pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Use prohibited

Description:

The square roots of the floating-point values of four elements of the matrix registers indicated by vs are calculated. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

The precision of the calculation is given by the following expression.

$|(\boldsymbol{approx\_sqrt}(\boldsymbol{x}) - \boldsymbol{sqrt}(\boldsymbol{x})) / \boldsymbol{sqrt}(\boldsymbol{x})| < 7.1 \times 10^{-7}$;

Special solutions are as follows.

approx_sqrt(+***nan***) = +***nan***

approx_sqrt(-***nan***) = +***nan***

approx_sqrt(+***inf***) = +***inf***

approx_sqrt(-***inf***) = +***nan***

approx_sqrt(+0.0) = +0.0

approx_sqrt(-0.0) = +0.0

approx_sqrt(***x***) = +***nan***; -***inf*** < ***x*** < -0.0

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- approx_sqrt( s[0] );
d[1] <- approx_sqrt( s[1] );
d[2] <- approx_sqrt( s[2] );
d[3] <- approx_sqrt( s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vsrt1.q

Sort 1 Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 0 0 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsrt1.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency：5　　　　pitch：1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Valid

Description:

The floating-point values of four elements from the matrix registers indicated by vs are sorted. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- min( s[0] , s[1] );
d[1] <- max( s[0] , s[1] );
d[2] <- min( s[2] , s[3] );
d[3] <- max( s[2] , s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vsrt2.q

Sort 2 Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 0 0 0 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsrt2.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Valid

Description:

The floating-point values of four elements from the matrix registers indicated by vs are sorted. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- min( s[0] , s[3] );
d[1] <- min( s[1] , s[2] );
d[2] <- max( s[1] , s[2] );
d[3] <- max( s[0] , s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vsrt3.q

Sort 3 Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 1 0 0 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsrt3.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Valid

Description:

The floating-point values of four elements from the matrix registers indicated by vs are sorted. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- max( s[0] , s[1] );
d[1] <- min( s[0] , s[1] );
d[2] <- max( s[2] , s[3] );
d[3] <- min( s[2] , s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vsrt4.q

Sort 4 Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 0 1 0 0 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vsrt4.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Valid

Description:

The floating-point values of four elements from the matrix registers indicated by vs are sorted. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
d[0] <- max( s[0] , s[3] );
d[1] <- max( s[1] , s[2] );
d[2] <- min( s[1] , s[2] );
d[3] <- min( s[0] , s[3] );
WriteMatrix( QUADWORD, vd, d );
```

vsub.s

Subtract Single Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 0	0 0 1	vt	0	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vsub.s  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

One element from the matrix register indicated by vt is subtracted from one element from the matrix register indicated by vs. The elements are treated as floating-point values. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
t <- ReadMatrix( SINGLEWORD, vt );
d[0] <- s[0] - t[0];
WriteMatrix( SINGLEWORD, vd, d );
```

vsub.p

Subtract Pair Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 0	0 0 1	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vsub.p  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Two elements from the matrix registers indicated by vt are subtracted from two elements from the matrix registers indicated by vs. The elements are treated as floating-point values. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
t <- ReadMatrix( PAIRWORD, vt );
d[0] <- s[0] - t[0];
d[1] <- s[1] - t[1];
WriteMatrix( PAIRWORD, vd, d );
```

vsub.t

Subtract Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 0	0 0 1	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vsub.t  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Three elements from the matrix registers indicated by vt are subtracted from three elements from the matrix registers indicated by vs. The elements are treated as floating-point values. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( TRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- s[0] - t[0];
d[1] <- s[1] - t[1];
d[2] <- s[2] - t[2];
WriteMatrix( TRIPLEWORD, vd, d );
```

vsub.q

Subtract Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
0 1 1 0 0 0	0 0 1	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vsub.q  vd, vs, vt
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	Valid	Valid

Description:

Four elements from the matrix registers indicated by vt are subtracted from four elements from the matrix registers indicated by vs. The elements are treated as floating-point values. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
d[0] <- s[0] - t[0];
d[1] <- s[1] - t[1];
d[2] <- s[2] - t[2];
d[3] <- s[3] - t[3];
WriteMatrix( QUADWORD, vd, d );
```

vsync

Synchronize

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0
6						26																									

VFPU

Syntax:

```
vsync
```

Instruction Type:

Synchronization instruction

Processing Time:

latency : 0 pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

The VFPU's internal state is synchronized. Subsequent VFPU instructions stall until the pipeline has emptied. If the instruction following the vsync is not a VFPU instruction, a vnop should be inserted.

Operation:

```
Sync();
```

vsync2

Synchronize2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
			6															26													

VFPU

Syntax:

```
vsync2
```

Instruction Type:

Synchronization instruction

Processing Time:

latency : 0 pitch : 6

Prefixing:

vpfxs	vpfxt	vpfxd
No effect	No effect	No effect

Description:

Synchronizes the internal state of the VFPU. Stalls the subsequent VFPU instruction until the pipeline becomes empty, and blocks the execution of the subsequent CPU instruction.
The vsync2 instruction is used as a workaround for the errata in which an lv.s or lv.q instruction that follows an sv.q,wb or vflush instruction operates incorrectly. This instruction cannot be used for any other purpose.

Operation:

```
Sync2();
```

vt4444.q

Convert to color4444 from packed unsigned chars Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 1 1 0 0 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vt4444.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Use prohibited

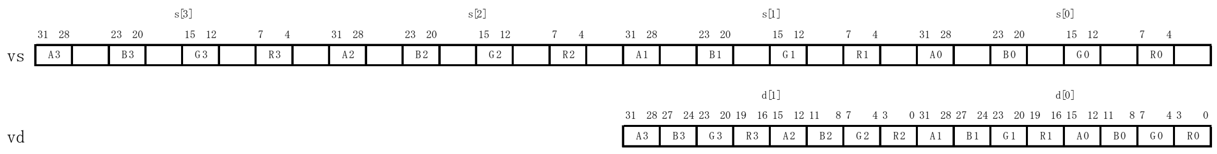
Description:

The packed, unsigned 8-bit data of four elements from the matrix registers indicated by vs is converted to packed 4444 color data and converted to 64 bits. This 64 bits is stored as two elements (32bits each) at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
tmp0[ 3: 0] <- (s[0]>> 4)&15;
tmp0[ 7: 4] <- (s[0]>>12)&15;
tmp0[11: 8] <- (s[0]>>20)&15;
tmp0[15:12] <- (s[0]>>28)&15;
tmp0[19:16] <- (s[1]>> 4)&15;
tmp0[23:20] <- (s[1]>>12)&15;
tmp0[27:24] <- (s[1]>>20)&15;
tmp0[31:28] <- (s[1]>>28)&15;
tmp1[ 3: 0] <- (s[2]>> 4)&15;
tmp1[ 7: 4] <- (s[2]>>12)&15;
tmp1[11: 8] <- (s[2]>>20)&15;
tmp1[15:12] <- (s[2]>>28)&15;
tmp1[19:16] <- (s[3]>> 4)&15;
```

```
        tmp1[23:20] <- (s[3]>>12)&15;
        tmp1[27:24] <- (s[3]>>20)&15;
        tmp1[31:28] <- (s[3]>>28)&15;
        d[0] <- tmp0;
        d[1] <- tmp1;
        WriteMatrix( PAIRWORD, vd, d );
```



vt5551.q

Convert to color5551 from packed unsigned chars Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 1 1 0 1 0 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vt5551.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency：3 pitch：1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Use prohibited

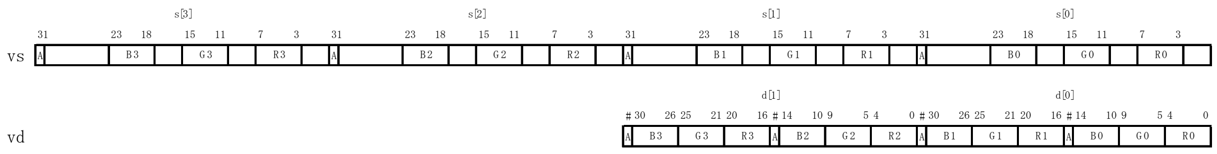
Description:

The packed, unsigned 8-bit data of four elements from the matrix registers indicated by vs is converted to packed 5551 color data and converted to 64 bits. This 64 bits is stored as two elements (32bits each) at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
tmp0[ 4: 0] <- (s[0]>> 3)&31;
tmp0[ 9: 5] <- (s[0]>>11)&31;
tmp0[14:10] <- (s[0]>>19)&31;
tmp0[15]    <- (s[0]>>31)&1 ;
tmp0[20:16] <- (s[1]>> 3)&31;
tmp0[25:21] <- (s[1]>>11)&31;
tmp0[30:26] <- (s[1]>>19)&31;
tmp0[31]    <- (s[1]>>31)&1 ;
tmp1[ 4: 0] <- (s[2]>> 3)&31;
tmp1[ 9: 5] <- (s[2]>>11)&31;
tmp1[14:10] <- (s[2]>>19)&31;
tmp1[15」   <- (s[2]>>31)&1 ;
tmp1[20:16] <- (s[3]>> 3)&31;
```

```
        tmp1[25:21] <- (s[3]>>11)&31;
        tmp1[30:26] <- (s[3]>>19)&31;
        tmp1[31]    <- (s[3]>>31)&1 ;
        d[0] <- tmp0;
        d[1] <- tmp1;
        WriteMatrix( PAIRWORD, vd, d );
```



vt5650.q

Convert to color5650 from packed unsigned chars Quad Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 1 0 1 1 0 1 1 1	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vt5650.q  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Only swizzle is valid	No effect	Use prohibited

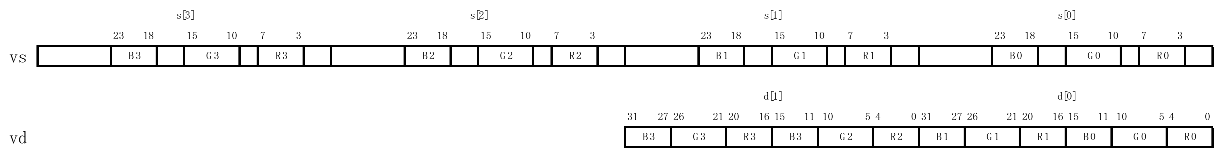
Description:

The packed, unsigned 8-bit data of four elements from the matrix registers indicated by vs is converted to packed 5650 color data and converted to 64 bits. This 64 bits is stored as two elements (32bits each) at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( QUADWORD, vs );
tmp0[ 4: 0] <- (s[0]>> 3)&31;
tmp0[10: 5] <- (s[0]>>10)&63;
tmp0[15:11] <- (s[0]>>19)&31;
tmp0[20:16] <- (s[1]>> 3)&31;
tmp0[26:21] <- (s[1]>>10)&63;
tmp0[31:27] <- (s[1]>>19)&31;
tmp1[ 4: 0] <- (s[2]>> 3)&31;
tmp1[10: 5] <- (s[2]>>10)&63;
tmp1[15:11] <- (s[2]>>19)&31;
tmp1[20:16] <- (s[3]>> 3)&31;
tmp1[26:21] <- (s[3]>>10)&63;
tmp1[31:27] <- (s[3]>>19)&31;
d[0] <- tmp0;
```

```
        d[1] <- tmp1;
        WriteMatrix( PAIRWORD, vd, d );
```



vtfm2.p

Transform 2 Pair Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	0 0 1	vt	0	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vtfm2.p  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 8 pitch : 2

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The transform of the elements of the 2x2 matrix from the matrix registers indicated by vs with the two elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point values. The two-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

With the vtfm2.p instruction, vs is used to specify a matrix, and vt is used to specify a vector. In the assembler, the pseudo-instructions vctfm2.p and vrtfm2.p are provided in order to program, with respect to either row vectors or column vectors, with an operand order which is the same as the calculation order. vrtfm2.p switches vs and vt and then assembles as vtfm2.p. As a result,

vctfm2.p	c000, e100, c200	(equivalent to vtfm2.p c000, e100, c200)
vrtfm2.p	r000, r100, m200	(equivalent to vtfm2.p r000, m200, r100)

can be written.

Operation:

```
s <- ReadMatrix(  PAIRXPAIRWORD, vs );
t <- ReadMatrix(  PAIRWORD, vt );
d[0] <- s[0] * t[0] + s[4] * t[1];
d[1] <- s[1] * t[0] + s[5] * t[1];
WriteMatrix( PAIRWORD, vd, d );
```

vtfm3.t

Transform 3 Triple Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	0 1 0	vt	1	vs	0	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vtfm3.t  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 9 pitch : 3

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The transform of the elements of the 3x3 matrix from the matrix registers indicated by vs with the three elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point values. The three-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

With the vtfm3.t instruction, vs is used to specify a matrix, and vt is used to specify a vector. In the assembler, the pseudo-instructions vctfm3.p and vrtfm3.p are provided in order to program, with respect to either row vectors or column vectors, with an operand order which is the same as the calculation order. vrtfm3.t switches vs and vt and then assembles as vtfm3.t. As a result,

vctfm3.t	c000, e100, c200	(equivalent to vtfm3.t c000, e100, c200)
vrtfm3.t	r000, r100, m200	(equivalent to vtfm3.t r000, m200, r100)

can be written.

Operation:

```
s <- ReadMatrix( TRIPLEXTRIPLEWORD, vs );
t <- ReadMatrix( TRIPLEWORD, vt );
d[0] <- s[0] * t[0] + s[4] * t[1] + s[8] * t[2];
d[1] <- s[1] * t[0] + s[5] * t[1] + s[9] * t[2];
d[2] <- s[2] * t[0] + s[6] * t[1] + s[10] * t[2];
WriteMatrix( TRIPLEWORD, vd, d );
```

vtfm4.q

Transform 4 Quad Word

31 30 29 28 27 26	25 24 23	22 21 20 19 18 17 16	15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 1 1 0 0	0 1 1	vt	1	vs	1	vd
6	3	7	1	7	1	7

VFPU

Syntax:

```
vtfm4.q  vd, vs, vt
```

Instruction Type:

Repeat (pipeline) instruction

Processing Time:

latency : 10   pitch : 4

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	Use prohibited	Use prohibited

Description:

The transform of the elements of the 4x4 matrix from the matrix registers indicated by vs with the four elements from the matrix registers indicated by vt is calculated. The elements are treated as floating-point values. The four-element floating-point result is stored at locations in the matrix register file indicated by vd.

Notes:

With the vtfm4.q instruction, vs is used to specify a matrix, and vt is used to specify a vector. In the assembler, the pseudo-instructions vctfm4.q and vrtfm4.q are provided in order to program, with respect to either row vectors or column vectors, with an operand order which is the same as the calculation order. vrtfm4.q switches vs and vt and then assembles as vtfm4.q. As a result,

vctfm4.q	c000, e100, c200	(equivalent to vtfm4.q c000, e100, c200)
vrtfm4.q	r000, r100, m200	(equivalent to vtfm4.q r000, m200, r100)

can be written.

Operation:

```
s <- ReadMatrix( QUADXQUADWORD, vs );
t <- ReadMatrix( QUADWORD, vt );
d[0] <- s[0] * t[0] + s[4] * t[1] + s[8] * t[2] + s[12] * t[3];
d[1] <- s[1] * t[0] + s[5] * t[1] + s[9] * t[2] + s[13] * t[3];
d[2] <- s[2] * t[0] + s[6] * t[1] + s[10] * t[2] + s[14] * t[3];
d[3] <- s[3] * t[0] + s[7] * t[1] + s[11] * t[2] + s[15] * t[3];
WriteMatrix( QUADWORD, vd, d );
```

vuc2ifs.s

Convert unsigned char single word to integer

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 1 0 0 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vuc2ifs.s  vd, vs
```

Instruction Type::

Pipeline instruction

Processing Time:

latency: 3   pitch: 1

Prefixing:

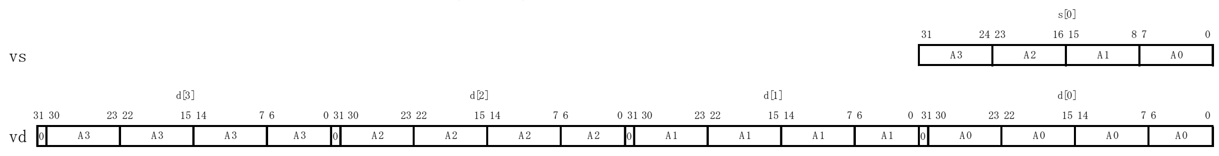
vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Only write mask is valid

Description:

Unpacks the 32-bit packed data in the matrix register specified by vs, performs pseudo-full-scale conversion from unsigned 8-bit integers to signed 32-bit integers, and stores the four integer elements at the location in the matrix register file specified by vd.

Operation:

```
s <- ReadMatrix( SIGLEWORD, vs );
d[0] <- {s[ 7: 0], s[ 7: 0], s[ 7: 0], s[ 7: 1]};
d[1] <- {s[15: 8], s[15: 8], s[15: 8], s[15: 9]};
d[2] <- {s[23:16], s[23:16], s[23:16], s[23:17]};
d[3] <- {s[31:24], s[31:24], s[31:24], s[31:25]};
WriteMatrix( QUADWORD, vd, d );
```



vus2i.s

Convert unsigned short to integer Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 1 0 1 0 0	vs	0	vd
6	11	7	1	7

VFPU

Syntax:

```
vus2i.s  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3   pitch : 1

Prefixing:

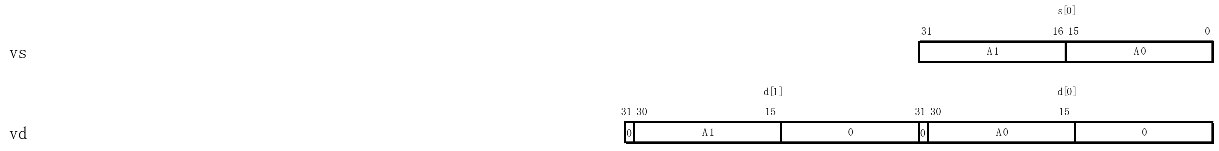
vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Only write mask is valid

Description:

32-bit packed data from the matrix register indicated by vs is unpacked and converted from unsigned 16-bit integers to signed 32-bit integers. The two-element integer result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- {s[0][15: 0], 15'b0};
d[1] <- {s[0][31:16], 15'b0};
WriteMatrix( PAIRWORD, vd, d );
```



vus2i.p

Convert unsigned short to integer Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15	14 13 12 11 10 9 8	7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 1 1 1 0 1 0 0	vs	1	vd
6	11	7	1	7

VFPU

Syntax:

```
vus2i.p  vd, vs
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

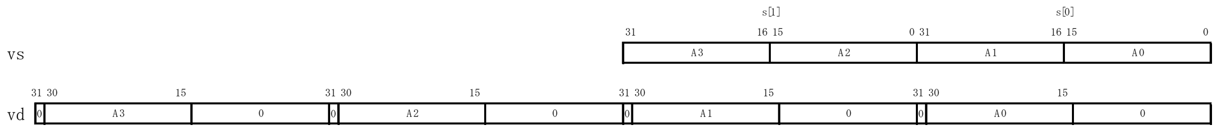
vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Only write mask is valid

Description:

64-bit packed data from the matrix registers indicated by vs is unpacked and converted from unsigned 16-bit integers to signed 32-bit integers. The four-element integer result is stored at locations in the matrix register file indicated by vd.

Operation:

```
s <- ReadMatrix( PAIRWORD, vs );
d[0] <- {s[0][15: 0], 15'b0};
d[1] <- {s[0][31:16], 15'b0};
d[2] <- {s[1][15: 0], 15'b0};
d[3] <- {s[1][31:16], 15'b0};
WriteMatrix( QUADWORD, vd, d );
```



vwbn.s

WrapBN Single Word

31–26	25–24	23–16	15	14–8	7	6–0
1 1 0 1 0 0	1 1	imm8	0	vs	0	vd
6	2	8	1	7	1	7

VFPU

Syntax:

```
vwbn.s  vd, vs, imm8
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 5 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Valid	No effect	Valid

Description:

The modulus of the floating-point value of one element from the matrix register indicated by vs with the exponent indicated by the imm8 field is calculated. The one-element floating-point result is stored at the location in the matrix register file indicated by vd.

wrapBN is defined by the following expression.

$\boldsymbol{wrapBN(s)} = \boldsymbol{fmod(s}, 2^{\boldsymbol{imm8}-127}) + (\boldsymbol{s} < 0.0 \,?\, -2^{\boldsymbol{imm8}-127} : 2^{\boldsymbol{imm8}-127})$

The input range of imm8 is as follows.

$1 <= \boldsymbol{imm8} <= 254$

Operation:

```
s <- ReadMatrix( SINGLEWORD, vs );
d[0] <- wrapBN( s[0], imm8 );
WriteMatrix( SINGLEWORD, vd, d );
```

vzero.s

Set Zero Single Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0	vd
6	19	7

VFPU

Syntax:

```
vzero.s  vd
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

0.0 is stored as a one-element floating-point value at the location in the matrix register file indicated by vd.

Operation:

```
d[0] <- 0.0;
WriteMatrix( SINGLEWORD, vd, d );
```

vzero.p

Set Zero Pair Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1	vd
6	19	7

VFPU

Syntax:

```
vzero.p  vd
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3 pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

0.0 is stored as a two-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
d[0] <- 0.0;
d[1] <- 0.0;
WriteMatrix( PAIRWORD, vd, d );
```

vzero.t

Set Zero Triple Word

31 30 29 28 27 26	25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7	6 5 4 3 2 1 0
1 1 0 1 0 0	0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0	vd
6	19	7

VFPU

Syntax:

```
vzero.t  vd
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

0.0 is stored as a three-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
d[0] <- 0.0;
d[1] <- 0.0;
d[2] <- 0.0;
WriteMatrix( TRIPLEWORD, vd, d );
```

vzero.q

Set Zero Quad Word

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	vd						
6						19																			7						

VFPU

Syntax:

```
vzero.q  vd
```

Instruction Type:

Pipeline instruction

Processing Time:

latency : 3　　　　pitch : 1

Prefixing:

vpfxs	vpfxt	vpfxd
Use prohibited	No effect	Valid

Description:

0.0 is stored as a four-element floating-point value at locations in the matrix register file indicated by vd.

Operation:

```
d[0] <- 0.0;
d[1] <- 0.0;
d[2] <- 0.0;
d[3] <- 0.0;
WriteMatrix( QUADWORD, vd, d );
```