Vector Floating-point Processing Unit (VFPU) User's Manual

Table of Contents

1. VFPU Specifications

1.1. Introduction

The VFPU (Vector Floating-point Processing Unit) is a small, high-performance, low-power consumption, 128-bit vector floating-point coprocessor developed by Sony Corporation. It is well-suited for geometry processing and signal processing applications such as audio.

1.2. Features

- Configured as ALLEGREX™ coprocessor
- 32-bit instructions
- 128 32-bit matrix registers suitable for matrix operations
- Write buffer can write directly to memory, bypassing the CPU
- 128-bit SIMD operation
- Vector floating-point arithmetic equivalent to IEEE 754 single-precision
- Flexible, variable length pipeline
- Pipelined arithmetic function unit
- Complete pipeline control providing low-power operation
- Pseudorandom number generator

1.3. Block Diagram

Figure 1-1 is a block diagram of the VFPU.

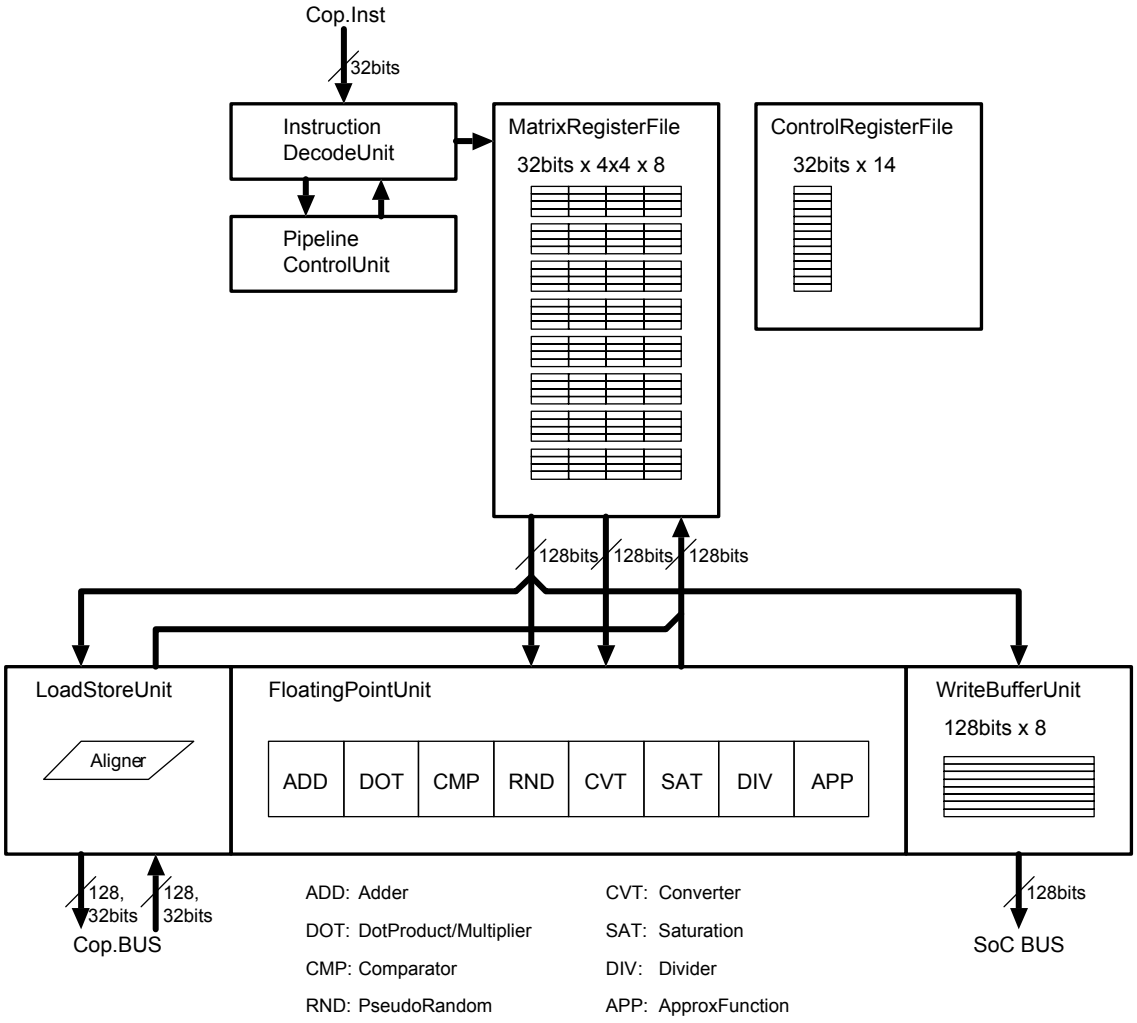


Figure 1-1: VFPU Block Diagram

2. VFPU Registers

2.1. Introduction

The VFPU's registers consist of a set of matrix registers that are used for arithmetic operations and a set of control registers that are used for control.

2.2. Matrix Registers

The VFPU has 128 matrix registers that can store 32-bit single-precision floating-point values. The matrix registers are organized as a register file. The register file allows values to be simultaneously read from an arbitrary source and target, and written to an arbitrary destination all at the same time. Here, source, target and destination can each consist of multiple registers.
The CPU accesses the matrix registers using move, load and store instructions. The VFPU uses values in the matrix registers as operands when performing operations, then stores the results from those operations back into the matrix registers.
Matrix registers are accessed by specifying a matrix register number, ranging from 0 to 127. The smallest unit of access is 32 bits.

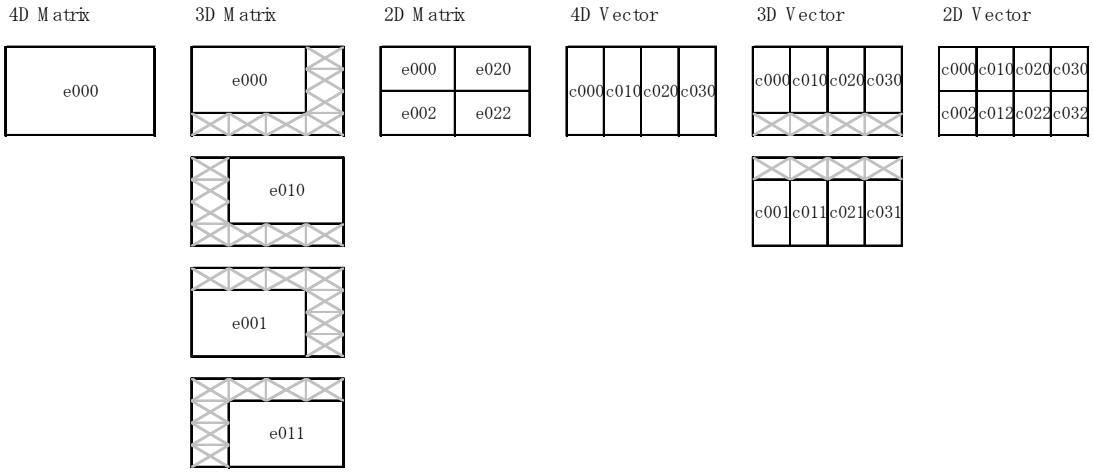
MSB 127 96	95 64	63 32	LSB 31 0
$96	$64	$32	$0
$97	$65	$33	$1
$98	$66	$34	$2
$99	$67	$35	$3
$100	$68	$36	$4
$101	$69	$37	$5
$102	$70	$38	$6
$103	$71	$39	$7
$104	$72	$40	$8
$105	$73	$41	$9
$106	$74	$42	$10
$107	$75	$43	$11
$108	$76	$44	$12
$109	$77	$45	$13
$110	$78	$46	$14
$111	$79	$47	$15
$112	$80	$48	$16
$113	$81	$49	$17
$114	$82	$50	$18
$115	$83	$51	$19
$116	$84	$52	$20
$117	$85	$53	$21
$118	$86	$54	$22
$119	$87	$55	$23
$120	$88	$56	$24
$121	$89	$57	$25
$122	$90	$58	$26
$123	$91	$59	$27
$124	$92	$60	$28
$125	$93	$61	$29
$126	$94	$62	$30
$127	$95	$63	$31

2.3. Access Model for Matrix Registers

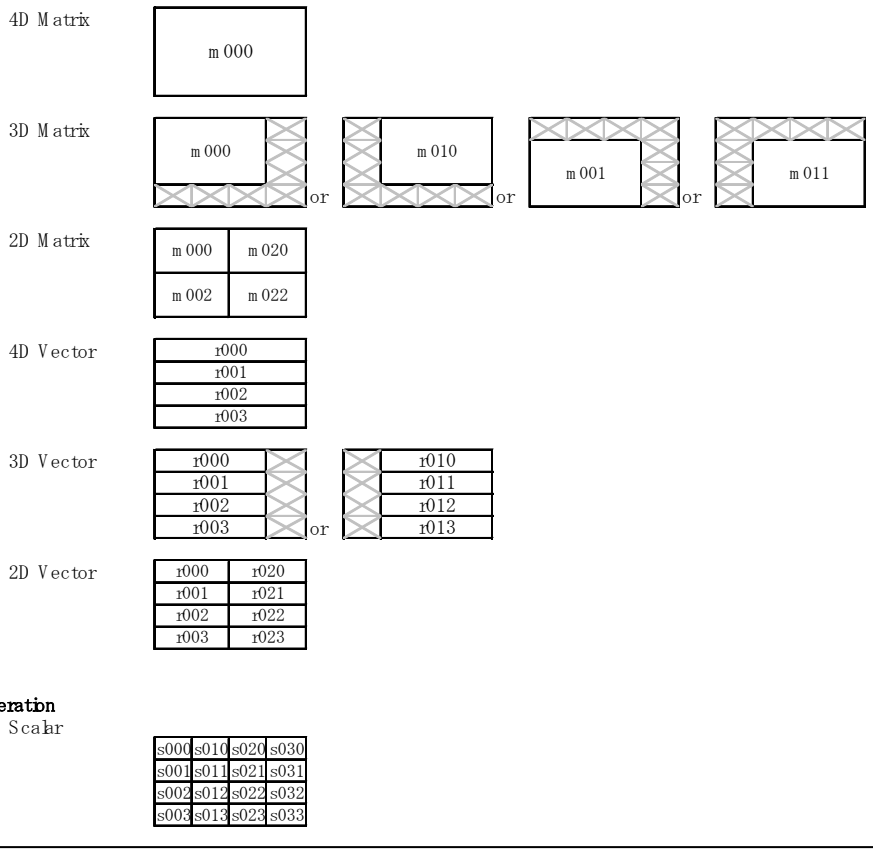
Data in the matrix register file can be read and written either in scalar format, vector format (2D, 3D, 4D), or matrix format (2 × 2, 3 × 3, 4 × 4).

The vector and matrix formats allow data to be read and written either vertically (column operation) or horizontally (row operation).

Coulmn operation



Row operation



Scalar operation

Scalar

s000	s010	s020	s030
s001	s011	s021	s031
s002	s012	s022	s032
s003	s013	s023	s033

2.3.1. Correspondence Between Scalar Format and Matrix Register File

The S*** format specification indicates that the matrix register file should be accessed using scalar format. The correspondence between the S*** format specification and matrix register number is shown below.

Scalar

s000	s010	s020	s030	s100	s110	s120	s130	s200	s210	s220	s230	s300	s310	s320	s330	s400	s410	s420	s430	s500	s510	s520	s530	s600	s610	s620	s630	s700	s710	s720	s730
s001	s011	s021	s031	s101	s111	s121	s131	s201	s211	s221	s231	s301	s311	s321	s331	s401	s411	s421	s431	s501	s511	s521	s531	s601	s611	s621	s631	s701	s711	s721	s731
s002	s012	s022	s032	s102	s112	s122	s132	s202	s212	s222	s232	s302	s312	s322	s332	s402	s412	s422	s432	s502	s512	s522	s532	s602	s612	s622	s632	s702	s712	s722	s732
s003	s013	s023	s033	s103	s113	s123	s133	s203	s213	s223	s233	s303	s313	s323	s333	s403	s413	s423	s433	s503	s513	s523	s533	s603	s613	s623	s633	s703	s713	s723	s733

Correspondence with register number

$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$10	$11	$12	$13	$14	$15	$16	$17	$18	$19	$20	$21	$22	$23	$24	$25	$26	$27	$28	$29	$30	$31
$32	$33	$34	$35	$36	$37	$38	$39	$40	$41	$42	$43	$44	$45	$46	$47	$48	$49	$50	$51	$52	$53	$54	$55	$56	$57	$58	$59	$60	$61	$62	$63
$64	$65	$66	$67	$68	$69	$70	$71	$72	$73	$74	$75	$76	$77	$78	$79	$80	$81	$82	$83	$84	$85	$86	$87	$88	$89	$90	$91	$92	$93	$94	$95
$96	$97	$98	$99	$100	$101	$102	$103	$104	$105	$106	$107	$108	$109	$110	$111	$112	$113	$114	$115	$116	$117	$118	$119	$120	$121	$122	$123	$124	$125	$126	$127

2.3.2. Correspondence Between Vector Format and Matrix Register File

The C*** and R*** format specifications indicate that the matrix register file should be accessed using vector format. Use the C*** format specification to indicate that access to a vector in the matrix register file should be in the vertical direction (column vector). Use the R*** format specification to indicate that access to a vector in the matrix register file should be in the horizontal direction (row vector).

Accessing 2-dimensional vectors in the vertical direction (2D column vector format)

The relationship between the C*** format specification, the S*** format elements of the matrix register file and the corresponding vector notation for a 2-dimensional vector is shown below.

2D Column vector

c000	c010	c020	c030	c100	c110	c120	c130	c200	c210	c220	c230	c300	c310	c320	c330	c400	c410	c420	c430	c500	c510	c520	c530	c600	c610	c620	c630	c700	c710	c720	c730
c002	c012	c022	c032	c102	c112	c122	c132	c202	c212	c222	c232	c302	c312	c322	c332	c402	c412	c422	c432	c502	c512	c522	c532	c602	c612	c622	c632	c702	c712	c722	c732

$$v = \begin{pmatrix} x \\ y \end{pmatrix}$$

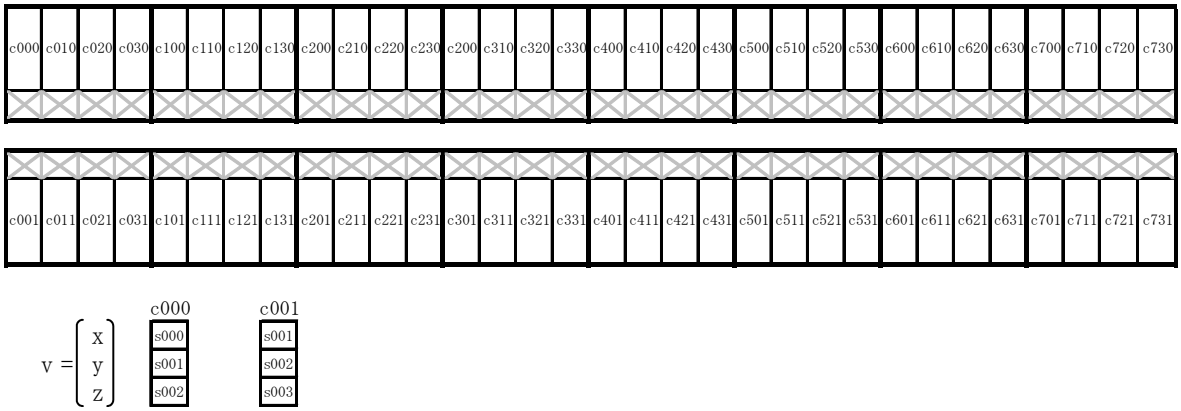
c000

s000
s001

Accessing 3-dimensional vectors in the vertical direction (3D column vector format)

The relationship between the C*** format specification, the S*** format elements of the matrix register file and the corresponding vector notation for a 3-dimensional vector is shown below.

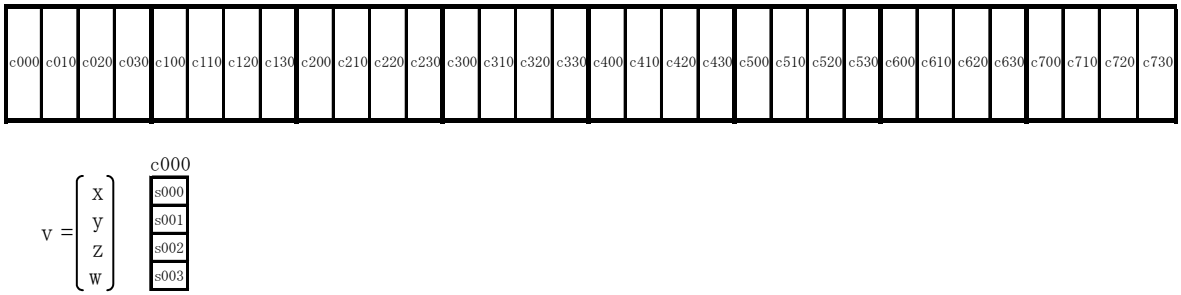
3D Column vector



Accessing 4-dimensional vectors in the vertical direction (4D column vector format)

The relationship between the C*** format specification, the S*** format elements of the matrix register file and the corresponding vector notation for a 4-dimensional vector is shown below.

4D Column vector



Accessing 2-dimensional vectors in the horizontal direction (2D row vector format)

The relationship between the R*** format specification, the S*** format elements of the matrix register file and the corresponding vector notation for a 2-dimensional vector is shown below.

2D Row vector

r000	r020	r100	r120	r200	r220	r300	r320	r400	r400	r500	r520	r600	r620	r700	r720
r001	r021	r101	r121	r201	r221	r301	r321	r401	r401	r501	r521	r601	r621	r701	r721
r002	r022	r102	r122	r202	r222	r302	r322	r402	r402	r502	r522	r602	r622	r702	r722
r003	r023	r103	r123	r203	r223	r303	r323	r403	r403	r503	r523	r603	r623	r703	r723

$v^t = (x \quad y)$ r000 [s000 | s010]

Accessing 3-dimensional vectors in the horizontal direction (3D row vector)

The relationship between the R*** format specification, the S*** format elements of the matrix register file and the corresponding vector notation for a 3-dimensional vector is shown below.

3D Row vector

r000	r100	r200	r300	r400	r500	r600	r700
r001	r101	r201	r301	r401	r501	r601	r701
r002	r102	r202	r302	r402	r502	r602	r702
r003	r103	r203	r303	r403	r503	r603	r703

r010	r110	r210	r310	r410	r510	r610	r710
r011	r111	r211	r311	r411	r511	r611	r711
r012	r112	r212	r312	r412	r512	r612	r712
r013	r113	r213	r313	r413	r513	r613	r713

$v^t = (x \quad y \quad z)$ r000 [s000 | s010 | s020], r010 [s010 | s020 | s030]

Accessing 4-dimensional vectors in the horizontal direction

The relationship between the R*** format specification, the S*** format elements of the matrix register file and the corresponding vector notation for a 4-dimensional vector is shown below.

4D Row vector

r000	r100	r200	r300	r400	r500	r600	r700
r001	r101	r201	r301	r401	r501	r601	r701
r002	r102	r202	r302	r402	r502	r602	r702
r003	r103	r203	r303	r403	r503	r603	r703

$v^t = (x \quad y \quad z \quad w)$ r000 [s000 | s010 | s020 | s030]

2.3.3. Correspondence Between Matrix Format and Matrix Register File

The E***/M*** format specifications indicate that the matrix register file is to be used in matrix format.
If a matrix register file is accessed in column major ordering format, which is constructed by combining column vectors, it is indicated in E*** format.
If a matrix register file is accessed in row major ordering format, which is constructed by combining row vectors, it is indicated in M*** format.

E*** format and M*** have a transpositional relationship to one another.
When C*** format is used as the vector format, the matrix which is constructed by combining C*** format vectors is E*** format. In this case, M*** format is the transposition matrix.
When R*** format is used as the vector format, the matrix which is constructed by combining R*** format vectors is M*** format. In this case, E*** format is the transposition matrix.

Accessing a 2 × 2 matrix in the vertical direction (2D column major ordering matrix format)

The relationship between the E*** format specification, the S*** format elements of the matrix register file and the corresponding matrix notation for a 2 × 2 matrix is shown below.

2D Column major ordering matrix

e000	e020	e100	e120	e200	e220	e300	e320	e400	e420	e500	e520	e600	e620	e700	e720
e002	e022	e102	e122	e202	e222	e302	e322	e402	e422	e502	e522	e602	e622	e702	e722

$$f = \begin{pmatrix} f11 & f12 \\ f21 & f22 \end{pmatrix}$$

e000

s000	s010
s001	s011

$$f^t = \begin{pmatrix} f11 & f21 \\ f12 & f22 \end{pmatrix}$$

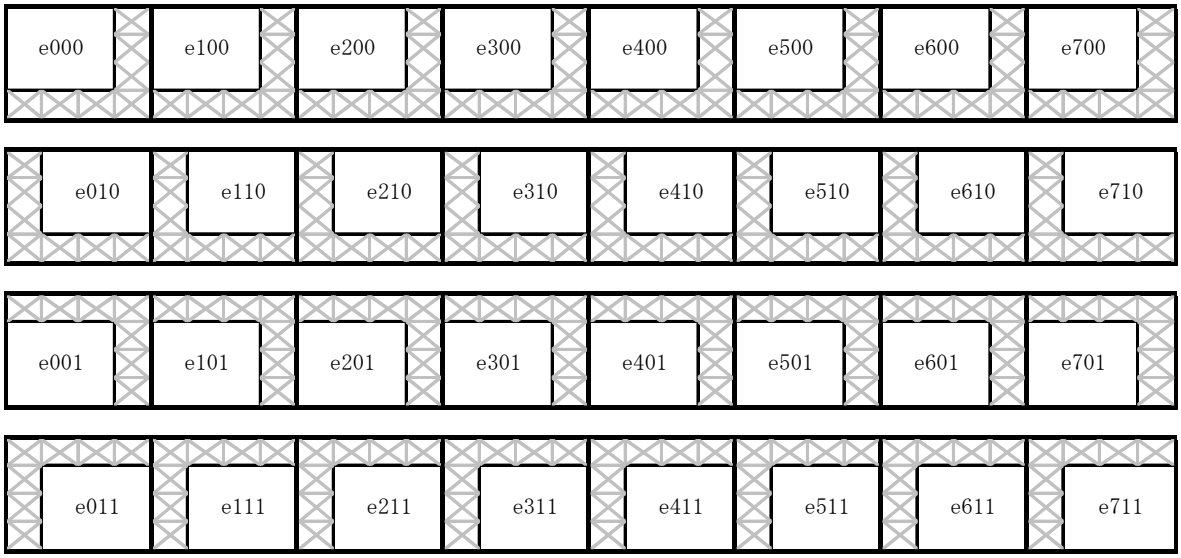
m000

s000	s001
s010	s011

Accessing a 3 × 3 matrix in the vertical direction (3D Column major ordering matrix format)

The relationship between the E*** format specification, the S*** format elements of the matrix register file and the corresponding matrix notation for a 3 × 3 matrix is shown below.

3D Column major ordering matrix



$$f = \begin{pmatrix} f11 & f12 & f13 \\ f21 & f22 & f23 \\ f31 & f31 & f33 \end{pmatrix}$$

e000		
s000	s010	s020
s001	s011	s021
s002	s012	s022

e010		
s010	s020	s030
s011	s021	s031
s012	s022	s032

e001		
s001	s011	s021
s002	s012	s022
s003	s013	s023

e011		
s011	s021	s031
s012	s022	s023
s013	s023	s033

$$f^t = \begin{pmatrix} f11 & f21 & f31 \\ f12 & f22 & f32 \\ f13 & f23 & f33 \end{pmatrix}$$

m000		
s000	s010	s020
s001	s011	s021
s002	s012	s022

m010		
s010	s020	s030
s011	s021	s031
s012	s022	s032

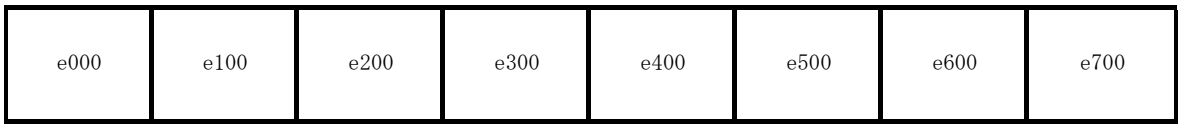
m001		
s001	s011	s021
s002	s012	s022
s003	s013	s023

m011		
s011	s021	s031
s012	s022	s023
s013	s023	s033

Accessing a 4 × 4 matrix in the vertical direction (4D Column major ordering matrix format)

The relationship between the E*** format specification, the S*** format elements of the matrix register file and the corresponding matrix notation for a 4 × 4 matrix is shown below.

4D Column major matrix



$$f = \begin{pmatrix} f11 & f12 & f13 & f14 \\ f21 & f22 & f23 & f24 \\ f31 & f32 & f33 & f34 \\ f41 & f42 & f43 & f44 \end{pmatrix}$$

e000			
s000	s010	s020	s030
s001	s011	s021	s031
s002	s012	s022	s023
s003	s013	s023	s033

$$f^t = \begin{pmatrix} f11 & f21 & f31 & f41 \\ f12 & f22 & f32 & f42 \\ f13 & f23 & f33 & f43 \\ f14 & f24 & f34 & f44 \end{pmatrix}$$

m000			
s000	s010	s020	s030
s001	s011	s021	s031
s002	s012	s022	s023
s003	s013	s023	s033

Accessing a 2 × 2 matrix in the horizontal direction (2D row major ordering matrix format)

The relationship between the M*** format specification, the S*** format elements of the matrix register file and the corresponding matrix notation for a 2 × 2 matrix is shown below.

2D Row major ordering matrix

m000	m020	m100	m120	m200	m220	m300	m320	m400	m420	m500	m520	m600	m620	m700	m720
m002	m022	m102	m122	m202	m222	m302	m322	m402	m422	m502	m522	m602	m622	m702	m722

$$f = \begin{pmatrix} f11 & f12 \\ f21 & f22 \end{pmatrix}$$

m000

s000	s010
s001	s011

$$f^t = \begin{pmatrix} f11 & f21 \\ f12 & f22 \end{pmatrix}$$

e000

s000	s001
s010	s011

Accessing a 3 × 3 matrix in the horizontal direction (3D row major ordering matrix format)

The relationship between the M*** format specification, the S*** format elements of the matrix register file and the corresponding matrix notation for a 2 × 2 matrix is shown below.

3D Row major ordering matrix

m000	m100	m200	m300	m400	m500	m600	m700
m010	m110	m210	m310	m410	m510	m610	m710
m001	m101	m201	m301	m401	m501	m601	m701
m011	m111	m211	m311	m411	m511	m611	m711

$$f = \begin{pmatrix} f11 & f12 & f13 \\ f21 & f22 & f23 \\ f31 & f32 & f33 \end{pmatrix}$$

m000

s000	s010	s020
s001	s011	s021
s002	s012	s022

m010

s010	s020	s030
s011	s021	s031
s012	s022	s032

m001

s001	s011	s021
s002	s012	s022
s003	s013	s023

m011

s011	s021	s031
s012	s022	s023
s013	s023	s033

$$f^t = \begin{pmatrix} f11 & f21 & f31 \\ f12 & f22 & f32 \\ f13 & f32 & f33 \end{pmatrix}$$

e000

s000	s010	s020
s001	s011	s021
s002	s012	s022

e010

s010	s020	s030
s011	s021	s031
s012	s022	s032

e001

s001	s011	s021
s002	s012	s022
s003	s013	s023

e011

s011	s021	s031
s012	s022	s023
s013	s023	s033

Accessing a 4 × 4 matrix in the horizontal direction (4D row major ordering matrix format)

The relationship between the M*** format specification, the S*** format elements of the matrix register file and the corresponding matrix notation for a 4 × 4 matrix is shown below.

4D Row major ordering matrix

m000	m100	m200	m300	m400	m500	m600	m700

$$f = \begin{pmatrix} f11 & f12 & f13 & f14 \\ f21 & f22 & f23 & f24 \\ f31 & f32 & f33 & f34 \\ f41 & f42 & f43 & f44 \end{pmatrix}$$

m000

s000	s010	s020	s030
s001	s011	s021	s031
s002	s012	s022	s023
s003	s013	s023	s033

$$f^t = \begin{pmatrix} f11 & f21 & f31 & f41 \\ f12 & f22 & f32 & f42 \\ f13 & f23 & f33 & f43 \\ f14 & f24 & f34 & f44 \end{pmatrix}$$

e000

s000	s010	s020	s030
s001	s011	s021	s031
s002	s012	s022	s023
s003	s013	s023	s033

2.4. CPU Data Transfer

The CPU accesses the VFPU matrix registers using move, load, and store instructions. There are no instructions to perform direct data transfers between the FPU and the VFPU.

Move instructions perform data transfers between the CPU's GPR (general-purpose registers) and the VFPU matrix registers.

- mfv, mtv

Load instructions read into the VFPU matrix registers the contents which have been read from the address specified by the CPU's GPR.

- lv.s, lv.q

Store instructions write into the address specified by the CPU's GPR the contents which have been read from the VFPU matrix registers.

- sv.s, sv.q, svl.q, svr.q, sv.q,wb

The lvl.q and lvr.q instructions, which perform unaligned load operations with respect to the VFPU, have been removed from the specifications. They do not function properly, so be sure not to use them.

2.4.1. Errata for lv.s, lv.q Cache Misses

There is an errata in which an erroneous value is written into the destination register as a

result of executing an lv.s or lv.q instruction. This happens when a D-cache refill occurs as a result of the lv.s. or lv.q instruction causing a cache miss while the VFPU is sending a busy signal to the CPU due to the previous VFPU instruction. Among the VFPU instructions, the vdiv.p, vdiv.t, vdiv.q, vflush, and sv.q,wb instructions, which can have long pipeline pitch cycle counts, may cause this problem.

In order to avoid this problem, the vdiv.p, vdiv.t, vdiv.q instructions were removed and the vsync2 instruction was added.
When executing an lv.s or lv.q instruction after executing an sv.q, wb or vflush instruction, insert a vsync2 instruction before the lv.s or lv.q instruction. Even if a CPU or FPU instruction is inserted after a sv.q,wb or vflush instruction, the subsequent lv.s or lv.q instruction may be affected by this problem; hence, a vsync2 instruction is required.

The vdiv.p, vdiv.t, vdiv.q instructions have been removed from the specifications. The same operation can be achieved by repeatedly executing the vdiv.s instruction.

2.5. Write Buffer

The VFPU's write buffer can perform writes directly to memory without going through the ALLEGREX™ load/store unit. The write buffer can be used by specifying an uncached address when issuing the sv.q, wb instructions.

Normally, the CPU and VFPU share the ALLEGREX™ load/store unit for the sv.q instruction and coherency is maintained between them. For the sq, wb instruction, however, the write buffer operates independently from the ALLEGREX™ load/store unit. Therefore, maintaining coherency is the software's responsibility. In this case, use the vflush instruction to ensure coherency. In order to reflect the data written from the write buffer in memory, a vflush instruction and a vnop instruction immediately after it must be used.

To write the result of an arithmetic operation performed by the VFPU to memory, use one of the sv.s, sv.q, svl.q, svr.q or sv.q, wb instructions. Write speed is fastest when using the write buffer with the sv.q, wb instructions. The write buffer has a FIFO structure, with 128 bits x 8 columns. As long as it is not full, the sv.q, wb instructions can be issued consecutively.

2.5.1. Errata when Write Buffer is Full

The sv.q and wb instructions for the uncached area perform bus accesses via the write buffer of the VFPU, but there are known problems in which, when the VFPU write buffer is full, if the next VFPU instruction is an arithmetic operation which performs a write to the matrix register file, an incorrect value is written to the destination register as the calculation result.

If the VFPU instruction following an sv.q or wb instruction is an instruction which performs a write to the matrix register file, insert vnop instructions before it. Even if a CPU or FPU instruction is inserted after an sv.q or wb instruction, there is a chance that the subsequent VFPU instruction will encounter a problem, so vnop instructions are necessary.
Of the VFPU instructions which follow an sv.q or wb instruction, those instructions which can be used without vnop are as follows.

- sv.s / sv.q / sv.q,wb
- mfv
- mfvc
- vnop / vsync / vflush/ vsync2
- vpfxd / vpfxs / vpfxt

VFPU instructions other than these are instructions which involve writing to the matrix register file, so vnop must be inserted.

2.6. Control Registers

The VFPU has 14 32-bit control registers that store various kinds of information. The CPU accesses these registers using a move instruction.
Control registers are numbered from 128 to 143 and their functions are shown below.

Register No.	Register name	Attributes	Function
$128	VFPU_PFXS	RW	Source prefix stack
$129	VFPU_PFXT	RW	Target prefix stack
$130	VFPU_PFXD	RW	Destination prefix stack
$131	VFPU_CC	RW	Condition information
$132	VFPU_INF4	RW	VFPU internal information 4
$133	VFPU_RSV5	-	Not used (reserved)
$134	VFPU_RSV6	-	Not used (reserved)
$135	VFPU_REV	R	VFPU revision information
$136	VFPU_RCX0	RW	Pseudorandom number generator internal information 0
$137	VFPU_RCX1	RW	Pseudorandom number generator internal information 1
$138	VFPU_RCX2	RW	Pseudorandom number generator internal information 2
$139	VFPU_RCX3	RW	Pseudorandom number generator internal information 3
$140	VFPU_RCX4	RW	Pseudorandom number generator internal information 4
$141	VFPU_RCX5	RW	Pseudorandom number generator internal information 5
$142	VFPU_RCX6	RW	Pseudorandom number generator internal information 6
$143	VFPU_RCX7	RW	Pseudorandom number generator internal information 7

2.6.1. Control Register Details

The details of the respective control registers are as follows.

$128 VFPU_PFXS

The VFP_PFXS register is used as a prefix stack for the source prefix. When performing context switches, it must be saved and restored.
When the vpfxs instruction is executed, decorator information is written to the register. When an instruction which consumes a prefix is executed, the register is returned to its initial value.

$129 VFPU_PFXT

The VFP_PFXT register is used as a prefix stack for the target prefix. When performing context switches, it must be saved and restored.
When the vpfxt instruction is executed, decorator information is written to the register. When an instruction which consumes a prefix is executed, the register is returned to its initial value.

$130 VFPU_PFXD

The VFP_PFXD register is used as a prefix stack for the destination prefix. When performing context switches, it must be saved and restored.
When the vpfxd instruction is executed, decorator information is written to the register. When an instruction which consumes a prefix is executed, the register is returned to its initial value.

$131 VFPU_CC

The VFPU_CC register is a register which stores a 6-bit condition flag.
When a vcmp.? instruction is executed, the comparison result is written to the register. The register can be referenced as conditions for the bvt, bvtl, bvf, and bvfl conditional branching instructions, and in addition can be used as conditions for the vcmovt.? and vcmovf.? conditional transfer instructions.

$132 VFPU_INF4

This register contains internal information which changes as a result of a number of VFPU calculations. When performing context switches, it must be saved and restored.

$135 VFPU_REV

This is a read-only register which contains VFPU revision data.

$136～$143 VFPU_RCX0～VFPU_RCX7

These registers contain internal information for the pseudo-random number generator of the VFPU. When performing context switches, they must be saved and restored.
These are set when the type of pseudo-random numbers is specified by the vrnds.s instruction. They are updated when a new random number is obtained from the vrndf?.? and vrndi.? instructions.

2.6.2. Initial Values and Initialization of Control Registers

Each control register is set to an initial value immediately after a reset. Be sure to use the values shown below when performing a software reset during initialization.

Register No.	Register name	Initial value
$128	VFPU_PFXS	0x000000e4
$129	VFPU_PFXT	0x000000e4
$130	VFPU_PFXD	0x00000000
$131	VFPU_CC	0x0000003f
$132	VFPU_INF4	0x00000000
$133	VFPU_RSV5	None
$134	VFPU_RSV6	None
$135	VFPU_REV	None
$136	VFPU_RCX0	0x3f800001
$137	VFPU_RCX1	0x3f800002
$138	VFPU_RCX2	0x3f800004
$139	VFPU_RCX3	0x3f800008
$140	VFPU_RCX4	0x3f800000
$141	VFPU_RCX5	0x3f800000
$142	VFPU_RCX6	0x3f800000
$143	VFPU_RCX7	0x3f800000

2.7. Saving and Restoring the Context

The VFPU keeps internal information in control registers 128 to 132 and 136 to 143. When saving the context, the internal state of the VFPU can be saved by saving matrix registers 0 to 127, control registers 128 to 132 and control registers 136 to 143. When restoring the context, the internal state of the VFPU can be restored by reloading the previously saved values of control registers 128 to 132 and control registers 136 to 143. If the state of the pseudorandom number generator does not need to be saved, it is not necessary to save and restore control registers 136 to 143.

2.7.1. Saving the Context

The following assembly code can be used to save the context.

```
// store VFPU Matrix registers
sv.q    c000,CTX_VFPU +  0
sv.q    c010,CTX_VFPU + 16
sv.q    c020,CTX_VFPU + 32
sv.q    c030,CTX_VFPU + 48
sv.q    c100,CTX_VFPU + 64
sv.q    c110,CTX_VFPU + 80
sv.q    c120,CTX_VFPU + 96
sv.q    c130,CTX_VFPU +112
sv.q    c200,CTX_VFPU +128
sv.q    c210,CTX_VFPU +144
sv.q    c220,CTX_VFPU +160
sv.q    c230,CTX_VFPU +176
sv.q    c300,CTX_VFPU +192
sv.q    c310,CTX_VFPU +208
sv.q    c320,CTX_VFPU +224
sv.q    c330,CTX_VFPU +240
```

```
sv.q    c400,CTX_VFPU +256
sv.q    c410,CTX_VFPU +272
sv.q    c420,CTX_VFPU +288
sv.q    c430,CTX_VFPU +304
sv.q    c500,CTX_VFPU +320
sv.q    c510,CTX_VFPU +336
sv.q    c520,CTX_VFPU +352
sv.q    c530,CTX_VFPU +368
sv.q    c600,CTX_VFPU +384
sv.q    c610,CTX_VFPU +400
sv.q    c620,CTX_VFPU +416
sv.q    c630,CTX_VFPU +432
sv.q    c700,CTX_VFPU +448
sv.q    c710,CTX_VFPU +464
sv.q    c720,CTX_VFPU +480
sv.q    c730,CTX_VFPU +496
// store VFPU Control registers
mfvc    t0,$128
mfvc    t1,$129
mfvc    t2,$130
mfvc    t3,$131
mfvc    t4,$132
sw      t0,CTX_VFPU +512
sw      t1,CTX_VFPU +516
sw      t2,CTX_VFPU +520
sw      t3,CTX_VFPU +524
sw      t4,CTX_VFPU +528
mfvc    t0,$136
mfvc    t1,$137
mfvc    t2,$138
mfvc    t3,$139
mfvc    t4,$140
mfvc    t5,$141
mfvc    t6,$142
mfvc    t7,$143
sw      t0,CTX_VFPU +532
sw      t1,CTX_VFPU +536
sw      t2,CTX_VFPU +540
sw      t3,CTX_VFPU +544
sw      t4,CTX_VFPU +548
sw      t5,CTX_VFPU +552
sw      t6,CTX_VFPU +556
sw      t7,CTX_VFPU +560
// sync VFPU pipeline
vsync
vnop
```

2.7.2. Restoring the Context

The following assembly code can be used to restore the context.

```
// sync VFPU pipeline
vsync
// restore VFPU Matrix registers
lv.q    c000,CTX_VFPU +  0
```

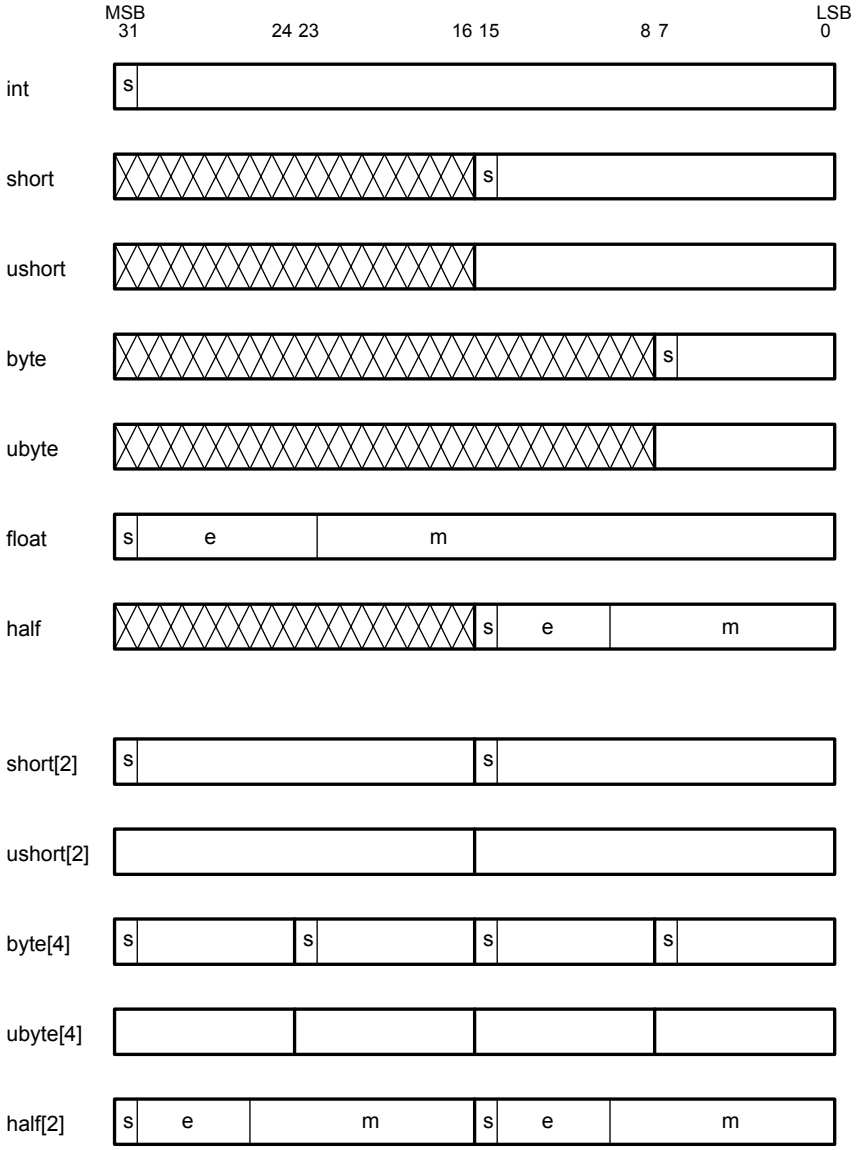
```
    lv.q    c010,CTX_VFPU + 16
    lv.q    c020,CTX_VFPU + 32
    lv.q    c030,CTX_VFPU + 48
    lv.q    c100,CTX_VFPU + 64
    lv.q    c110,CTX_VFPU + 80
    lv.q    c120,CTX_VFPU + 96
    lv.q    c130,CTX_VFPU +112
    lv.q    c200,CTX_VFPU +128
    lv.q    c210,CTX_VFPU +144
    lv.q    c220,CTX_VFPU +160
    lv.q    c230,CTX_VFPU +176
    lv.q    c300,CTX_VFPU +192
    lv.q    c310,CTX_VFPU +208
    lv.q    c320,CTX_VFPU +224
    lv.q    c330,CTX_VFPU +240
    lv.q    c400,CTX_VFPU +256
    lv.q    c410,CTX_VFPU +272
    lv.q    c420,CTX_VFPU +288
    lv.q    c430,CTX_VFPU +304
    lv.q    c500,CTX_VFPU +320
    lv.q    c510,CTX_VFPU +336
    lv.q    c520,CTX_VFPU +352
    lv.q    c530,CTX_VFPU +368
    lv.q    c600,CTX_VFPU +384
    lv.q    c610,CTX_VFPU +400
    lv.q    c620,CTX_VFPU +416
    lv.q    c630,CTX_VFPU +432
    lv.q    c700,CTX_VFPU +448
    lv.q    c710,CTX_VFPU +464
    lv.q    c720,CTX_VFPU +480
    lv.q    c730,CTX_VFPU +496
    // restore VFPU Control registers
    lw     t0,CTX_VFPU +512
    lw     t1,CTX_VFPU +516
    lw     t2,CTX_VFPU +520
    lw     t3,CTX_VFPU +524
    lw     t4,CTX_VFPU +528
    mtvc    t0,$128
    mtvc    t1,$129
    mtvc    t2,$130
    mtvc    t3,$131
    mtvc    t4,$132
    lw     t0,CTX_VFPU +532
    lw     t1,CTX_VFPU +536
    lw     t2,CTX_VFPU +540
    lw     t3,CTX_VFPU +544
    lw     t4,CTX_VFPU +548
    lw     t5,CTX_VFPU +552
    lw     t6,CTX_VFPU +556
    lw     t7,CTX_VFPU +560
    mtvc    t0,$136
    mtvc    t1,$137
    mtvc    t2,$138
    mtvc    t3,$139
    mtvc    t4,$140
```

```
    mtvc    t5,$141
    mtvc    t6,$142
    mtvc    t7,$143
```

3. VFPU Data Formats

3.1. Introduction

The VFPU supports the following data formats: 32-bit integer, 32-bit floating point, packed 8-bit integer, packed 16-bit integer and packed 16-bit floating point.
The bit arrangements for each format are shown below.



3.2. Data Formats

3.2.1. 32-bit Integer (signed)

2's complement representation.

Valid range:

$-2147483648 \leq x \leq 2147483647$

3.2.2. 16-bit Integer (signed)

2's complement representation.

Valid range:

$-32768 \leq x \leq 32767$

3.2.3. 16-bit Integer (unsigned)

Valid range:

$0 \leq x \leq 65535$

3.2.4. 8-bit Integer (signed)

2's complement representation.

Valid range:

$-128 \leq x \leq 127$

3.2.5. 8-bit Integer (unsigned)

Valid range:

$0 \leq x \leq 255$

3.2.6. 32-bit Floating-point Number

Equivalent to IEEE 754 single-precision floating-point, except that denormalized numbers are handled as zero.

Valid numbers are given by the following expressions.

$(-1)^s \times 0.0$; e = 0

$(-1)^s \times 2^{e-127} \times (1 + m/2^{23})$; 0 < e < 255

$(-1)^s \times$ inf; e = 255&m = 0

nan; e = 255&m! = 0

3.2.7. 16-bit Floating-point Number

Equivalent to s10e5, except that denormalized numbers are handled as zero.

Valid numbers are given by the following expressions.

$-1^s \times 0.0$; e = 0

$-1^s \times 2^{e-15} \times (1 + m/2^{10})$; 0 < e < 31

$-1^s \times$ inf; e = 31&m = 0

nan; e = 31&m! = 0

3.3. IEEE 754 Compatibility

The VFPU's handling of floating-point numbers basically conforms to IEEE 754 except for the following differences.

- Denormalized numbers are handled as zero.
- Non-numbers do not generate an exception and are handled as NaN.
- Rounding is always done to the nearest value. (Except for the Convert to Integer instruction.)

3.3.1. Decoding and Encoding of Floating-point Numbers

The VFPU interprets the values of floating-point numbers as shown in the table below.

input data	IEEE754	VFPU
0x7FFFFFFF ~ 0x7FC00000	QNaN	NaN
0x7FBFFFFF ~ 0x7F800001	SNaN	
0x7F800000	+Infinity	+Infinity
0x7F7FFFFF ~ 0x00800000	+Number	+Number
0x007FFFFF ~ 0x00000001	+DenormalizedNumber	+Zero
0x00000000	+Zero	
0x80000000	-Zero	-Zero
0x80000001 ~ 0x807FFFFF	-DenormalizedNumber	
0x80800000 ~ 0xFF7FFFFF	-Number	-Number
0xFF800000	-Infinity	-Infinity
0xFF800001 ~ 0xFFBFFFFF	SNaN	NaN
0xFFC00000 ~ 0xFFFFFFFF	QNaN	

The VFPU encodes the results of operations into floating-point numbers as shown in the table below.

VFPU	output data
NaN	0x7F800001
+Infinity	0x7F800000
+Number	0x7F7FFFFF ~ 0x00800000
+Zero	0x00000000
-Zero	0x80000000
-Number	0x80800000 ~ 0xFF7FFFFF
-Infinity	0xFF800000
NaN	0xFF800001

4. Precision of VFPU Operations

4.1. Introduction

The VFPU's internal floating-point arithmetic unit is compliant with the IEEE 754 specification. The precision of operations also conforms to IEEE 754 except for the handling of NaNs, denormalized numbers and rounding. This produces some incompatibility with IEEE 754.

4.2. Dot Product

The VFPU uses its dot product arithmetic unit to execute the vcrsp.t, vdot.*, vhdp.*, vdet.*, vfad.*, vavg.*, vmmul.*, vtfm*.*, vhtfm*.* and vqmul.* instructions.
The dot product arithmetic unit works by first multiplying together the corresponding elements of the operands. It then aligns the resultant products at the position of the largest exponent and sums them. Consequently, the result will be different than if the dot product were calculated using a combination of binomial operations.

4.3. Approximation Functions

The VFPU uses its approximation function arithmetic unit to execute the vrcp.*, vrsq.*, vsin.*, vcos.*, vexp2.*, vlog2.*, vsqrt.*, vasin.*, vnsin.*, vnrcp.* and vrexp2.* instructions.
See the description of these instructions for information about their precision.

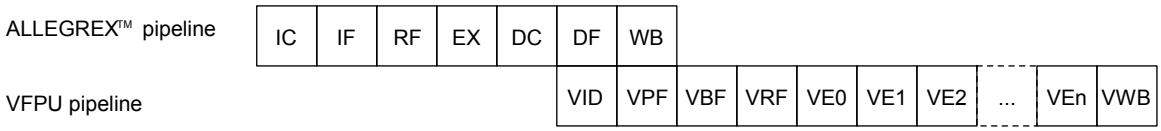
5. VFPU Pipeline

5.1. Introduction

The number of cycles required to execute a particular VFPU instruction is determined by the latency and pitch of that instruction.
Latency is the total number of cycles needed to perform register fetching and instruction execution for a given instruction. Pitch is the spacing between instruction execution stages of sequentially executing VFPU instructions of the same type.

5.2. Pipeline Overview

The VFPU pipeline starts operating from the DF stage of the ALLEGREX™ pipeline.

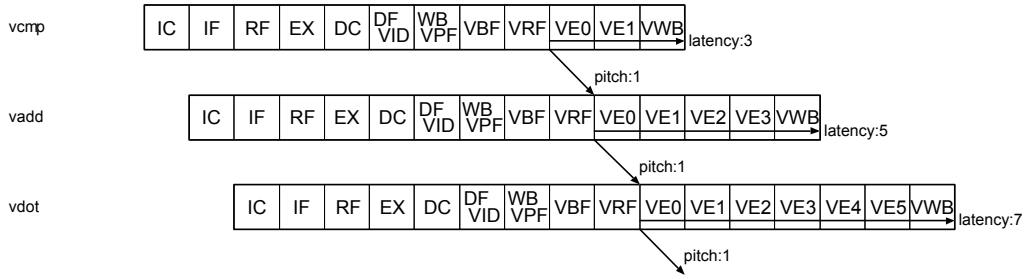


VID : VFPU Instruction Decode
VPF : VFPU Preform
VBF : VFPU Buffer Fetch
VRF : VFPU Register Fetch
VE0...n : VFPU Execut Stage0...n
VWB : VFPU Write Back

In the VFPU, instruction stage latency and pitch are different depending on the instruction.

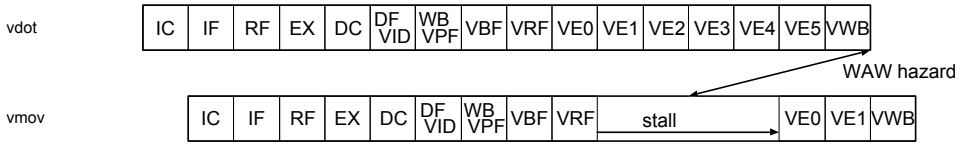
5.3. VFPU Pipeline Hazards

A sequence of pipeline instructions can execute without stalling the pipeline if each instruction in the sequence has the same or greater latency than the instruction that precedes it.



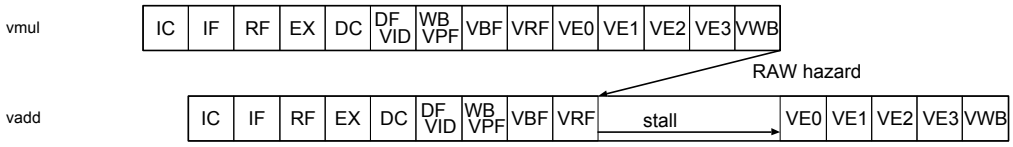
5.3.1. Write after Write Hazard

In a sequence of VFPU instructions, if one instruction has a shorter latency than the instruction which precedes it, that instruction will stall at the VRF stage so that it will not overtake the previous instruction's write operation. This happens regardless of whether or not the two instructions are writing to the same destination register.



5.3.2. Read after Write Hazard

In a sequence of VFPU instructions, if one instruction uses the result from the instruction which precedes it, that instruction will stall until the previous instruction completes its write operation. This happens even if there is just one overlapping resource between the two instructions.



When a Read after Write hazard occurs

The following example shows how a RAW hazard occurs.

```
vadd.s s003, s000, s001
vsub.q c010, c010, c000
```

If we use the actual register numbers for the resources used in the source, target and destination operands of each instruction, we obtain the following.

```
vadd.s {$96}, {$0}, {$32}
vsub.q {$1 ,$33,$65,$97}, {$1 ,$33,$65,$97}, {$0 ,$32,$64,$96}
```

Since the $96 element is used for both destination and source operands, a RAW hazard will occur resulting in a stall.

When a Read after Write hazard does not occur

The following example shows when a RAW hazard does not occur.

```
vadd.s s003, s000, s001
vsub.t c010, c010, c000
```

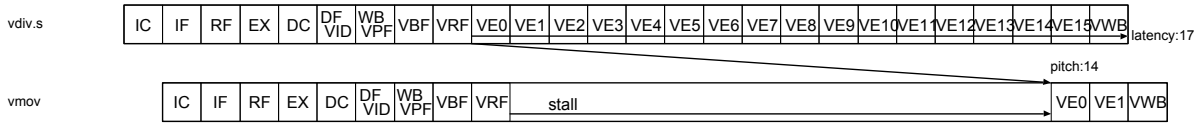
If we again use the actual register numbers for the resources used in the source, target and destination operands of each instruction, we obtain the following.

```
vadd.s {$96}, {$0}, {$32}
vsub.t {$1 ,$33,$65}, {$1 ,$33,$65}, {$0 ,$32,$64}
```

Since no dependent element is used for both destination and source operands, a RAW hazard will not occur and the pipeline will not stall.

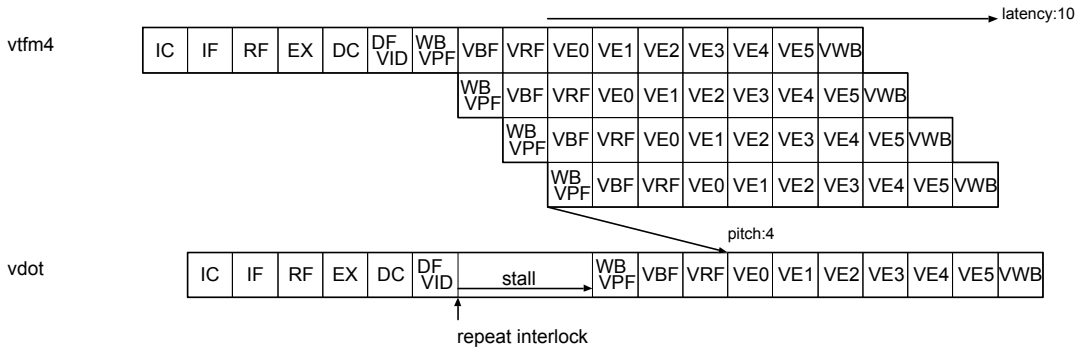
5.4. Multi-cycle Instructions

A VFPU instruction that requires multiple cycles to execute (a multi-cycle instruction) will cause a subsequent instruction to stall until the multi-cycle instruction completes execution.

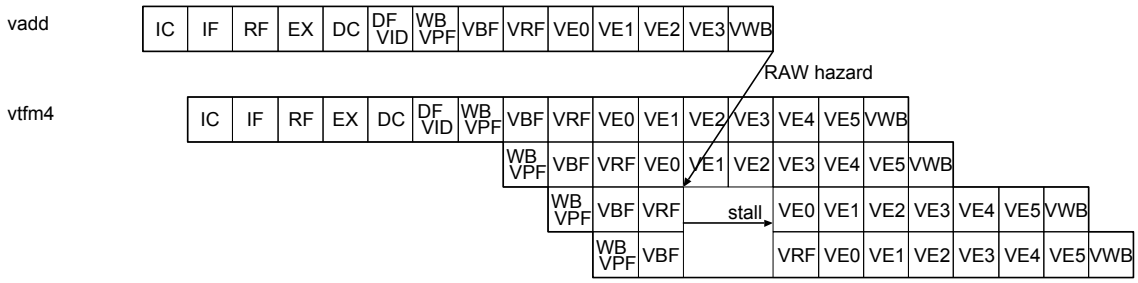


5.5. Repeat Instructions

A VFPU instruction that requires multiple internal instructions to execute (a repeat instruction) will cause a subsequent instruction to stall until the repeat instruction has completed execution.



If a hazard is generated within a repeat instruction because a dependency exists between two internal instructions, the subsequent internal instruction will stall.



5.5.1. Restrictions When Using Repeat Instructions

Repeat instructions are implemented by repeatedly executing a series of simpler instructions within the VFPU. As a result, there are some restrictions on the use of register resources. Consider the following example.

```
vsin.t c000, r002
```

This will be expanded within the VFPU and executed as:

```
vsin.s {$64}, {$66}
vsin.s {$32}, {$65}
vsin.s {$0},  {$64}
```

creating a hazard because a dependency exists between the $64 and $32 elements. It is not possible for the VFPU to avoid this hazard. Consequently, in a repeat instruction, it is not permitted to include the source or target register elements in the destination register element.

5.5.2. Exceptions to Constraints in Repeat Instructions

The following is a list of exceptional cases for which a repeat instruction can include the source or target register parameters in the destination register parameters.

- Repeat instructions other than vmmul.p, vmmul.t, vmmul.q, vtfm2.p, vtfm3.t, vtfm4.q, vhtfm2.p, vhtfm3.t, vhtfm4.q, vcrsp.t, and vqmul.q.
- If the register specified as the source or target operand is also specified by the destination operand, and the overlapping area is to be written in the same format and the same direction as for the read.

Example in which overlap does not occur

```
vsin.q      c000, c010
vmscl.q     m000, m100, s100
vtfm4.q     c000, m100, c010
vmmmul.q    m000, m100, m200
```

Example in which overlap occurs and causes an error

```
vsin.t      c000, c001              ; vd and vs overlap
vsin.t      c001, c000              ; vd and vs overlap
```

```
vsin.q      c000, r000              ; vd and vs overlap at ${0}
vmscl.q     m000, m100, s000        ; vd and vt overlap at ${0}
vmscl.q     m000, e000, s100        ; Overlap in which the direction of vd and vs
                                      do not match
vtfm4.q     c000, m000, c100        ; vd and vs areas overlap
vtfm4.q     c000, m100, c000        ; vd and vt areas overlap
vmmmul.q    m000, m000, m100        ; vd and vs areas overlap
vmmmul.q    e000, m000, m100        ; vd and vs areas overlap
```

Example in which overlap occurs but is seen as exceptional

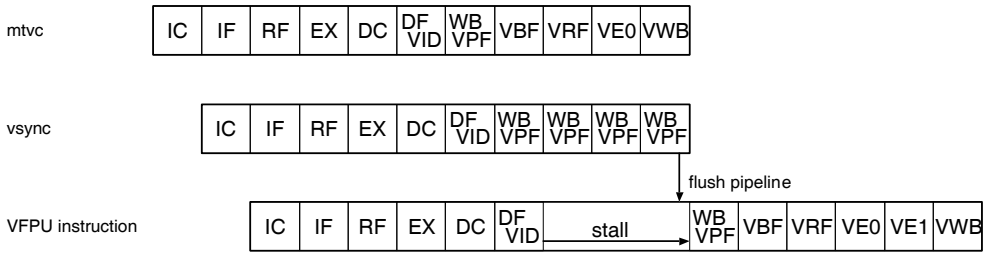
```
vsin.t      c000, c000              ; For vd and vs, the area is the same, but the
                                      direction is the same, so it is OK
vmscl.q     m000, m000, s100        ; For vd and vs, the area is the same, but the
                                      direction is the same, so it is OK
```

5.6. Synchronization Instructions

The VFPU provides 3 synchronization instructions (vsync, vnop, vflush) which are described below.

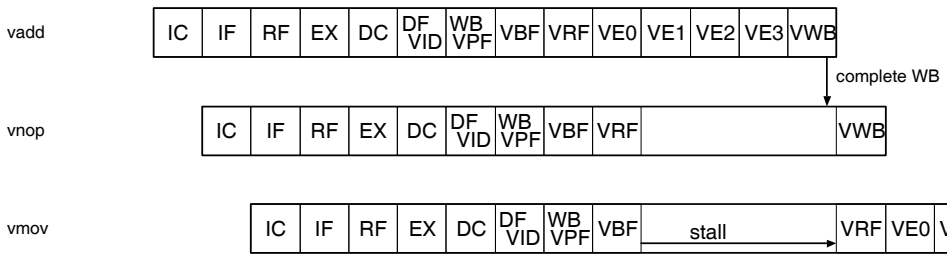
5.6.1. vsync

A vsync instruction causes subsequent VPFU instructions to stall until the execution pipeline has emptied.



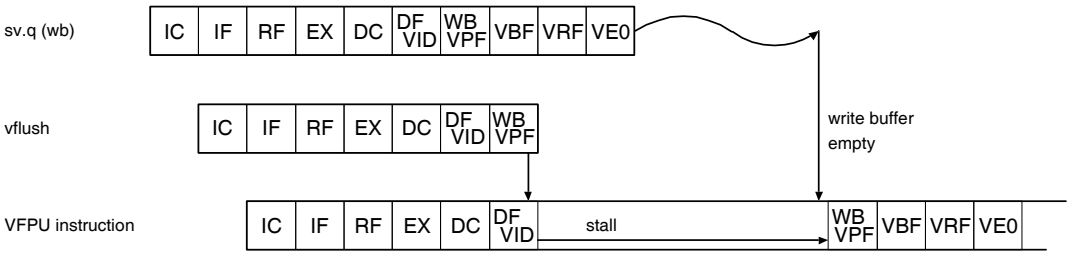
5.6.2. vnop

A vnop instruction causes an empty VFPU instruction to be executed.

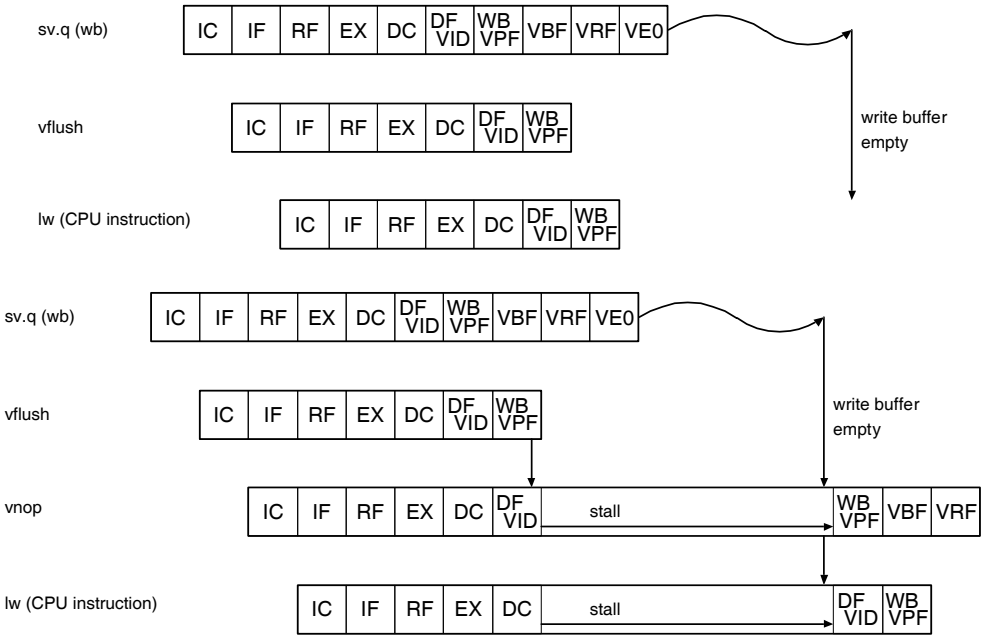


5.6.3. vflush

A vflush instruction causes subsequent VPFU instructions to stall until the VFPU write buffer has emptied.



In addition, the only instructions which can be directly stalled by the vflush instruction are subsequent VFPU instructions. CPU instructions and FPU instructions cannot be directly stalled, so it is necessary to insert vnop instructions in order to block subsequent CPU instructions.



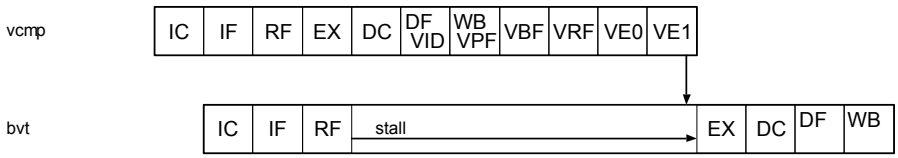
5.7. ALLEGREX™ Interlocks

This section describes situations when VFPU instructions executed by ALLEGREX™ will generate interlocks and stall the instruction pipeline.

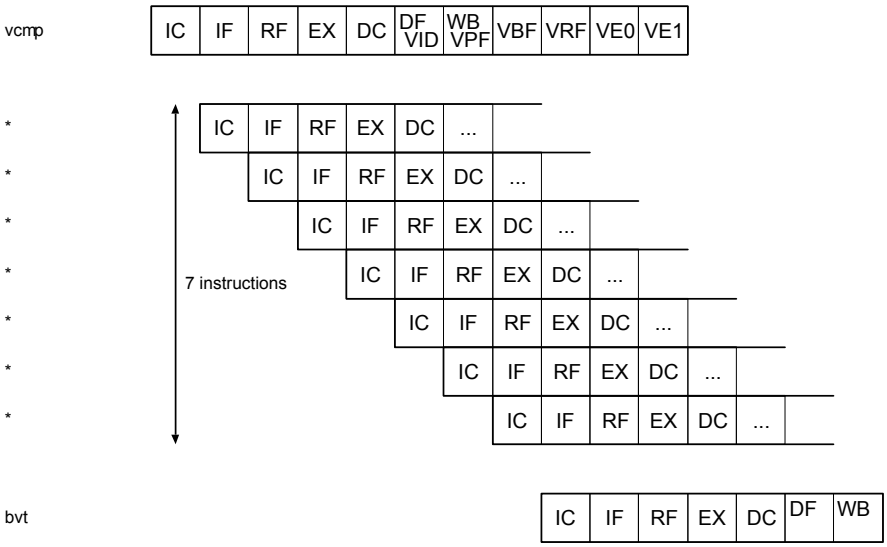
5.7.1. Branch Instructions (bv*)

A branch instruction will stall the pipeline when a bv* instruction is executed after a vcmp

instruction.



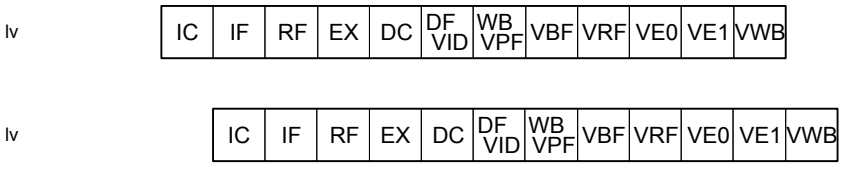
If vcmp instructions and bv* instructions are separated by seven or more instructions, and if furthermore the vcmp instructions have not stalled, then bv* instructions can be executed without stalling.



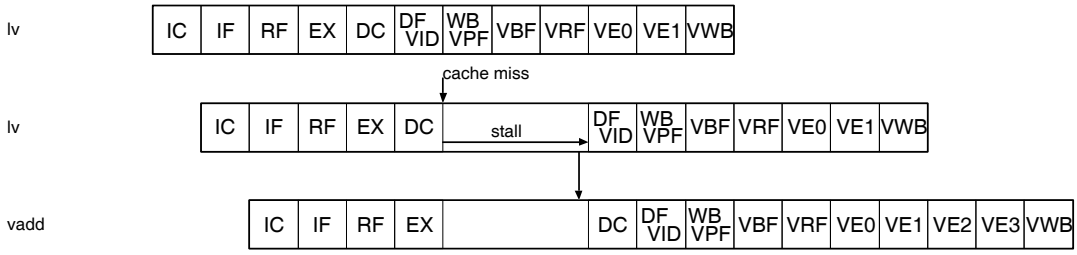
For combinations of instructions other than vcmp and bv*, interlock due to RAW hazards involving the VFPU control registers will not occur. After an instruction writes a control register, a latency cycle's worth of vnop instructions must be explicitly added by hand if the control register is to be read by the mfvc or vmfvc instruction.

5.7.2. Load Instructions (lv, …)

A load instruction will only stall the pipeline if a miss occurs in the D-cache. Otherwise, the pipeline is able to execute a sequence of continuous load instructions.



When a cache miss occurs, time is required to fill the cache.

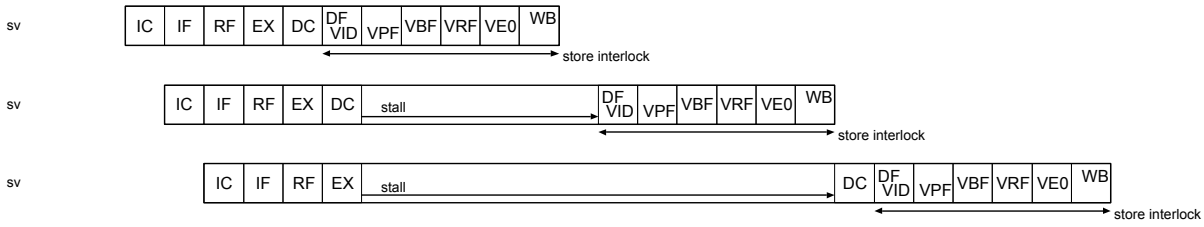


5.7.3. Store Instructions (sv,…)

Store instructions can be classified into the following three types according to the path from the VFPU to memory.

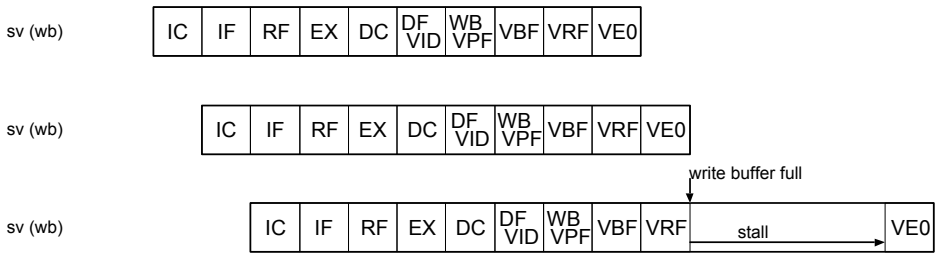
Cached / Non-cached store

Regardless of whether a store is to a cached or non-cached area, the store instruction will always stall the ALLEGREX™ CPU pipeline because of a VFPU wait interlock. For a cached store, additional cycles will also be required to fill the cache when a cache miss occurs. For a non-cached store, additional cycles will be required to flush the CPU write buffer if it has become full.



Non-cached store with VFPU write buffer write

In this case, continuous stores can be executed without stalling the pipeline as long as the VFPU write buffer does not become full.



Cached store with VFPU write buffer write

In this case, the store instruction will always stall the ALLEGREX™ CPU pipeline because of a VFPU wait interlock. Additional cycles will also be required to fill the cache when a

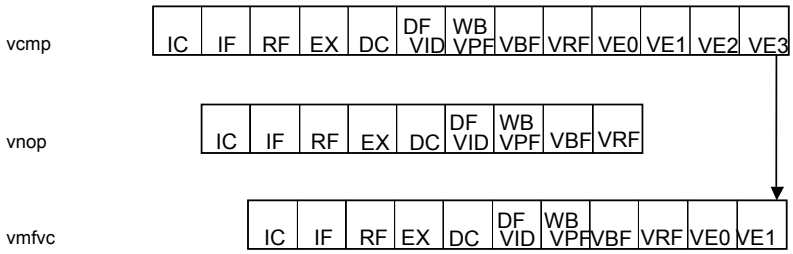
cache miss occurs and to flush the VFPU write buffer if it has become full.

5.8. Reading VFPU Control Registers

This section describes the situations when explicit delay cycles need to be inserted by software whenVFPU control registers are read. Because no automatic interlock is performed by the hardware when VFPU control registers are read, the results will be unpredictable if delay cycles are not properly inserted using the vnop instruction.

5.8.1. Reading the VFPU_CC Register Following the vcmp Instruction

When the VFPU_CC register is updated by a previous vcmp instruction, and the VFPU_CC register is subsequently read with an mfvc or vmfvc instruction, a vnop needs to be inserted as shown below.



6. VFPU Instruction Set

6.1. Operands

The VFPU has 128 separate matrix registers that can store 32-bit single-precision floating-point numbers. The matrix registers are organized as a register file. The register file allows values to be simultaneously read from an arbitrary source and target, and written to an arbitrary destination all at the same time. Here, source, target and destination can each consist of multiple registers.
The matrix register file can be read and written in scalar format, vector format (2D, 3D, or 4D) or matrix format (2 × 2, 3 × 3, or 4 × 4). The vector and matrix formats can also be specified so that reading and writing are done in the vertical or horizontal directions.
For more information on the matrix register access model, see "2.3 Access Model for Matrix Registers".

6.1.1. Reading and Writing the Matrix Register File

The matrix register file can be read and written by specifying the $*,S***,C***,R***, E***,M***formats.
The $* format is used to directly specify a matrix register from 0 to 127. The S*** format is used to specify a scalar in the matrix register file. The C*** format is used to specify a 2D, 3D or 4D vector in the matrix register file in the vertical direction. The R*** format is used to specify a 2D, 3D, or 4D vector in the matrix register file in the horizontal direction. The E*** format is used to specify a 2 × 2, 3 × 3, or 4 × 4 matrix in the matrix register file in the vertical direction. The M*** format is used to specify a 2 × 2, 3 × 3, or 4 × 4 matrix in the matrix register file in the horizontal direction.

6.1.2. Definitions of Operations

The operations of reading and writing to the matrix register file can be defined using the C language functions shown below.

Definition of matrix register file resource.

```
float MRF[128]; // MatrixRegisterFile

enum data_size
{
```

```
  SINGLEWORD,         // scalar
  PAIRWORD,           // 2-dimensional vector
  TRIPLEWORD,         // 3-dimensional vector
  QUADWORD,           // 4-dimensional vector
  PAIRXPAIRWORD,      // 2 × 2 matrix
  TRIPLEXTRIPLEWORD, // 3 × 3 matrix
  QUADXQUADWORD,      // 4 × 4 matrix
};
```

The following functions provide definitions of matrix register file read and write operations.

```
float* ReadMatrix( float* rd,enum data_size size,int code )
{
  int mtx;
  int idx;
  int rxc;
  int fsl;
  int r,c;

  mtx = (code>>2)&7;

  switch( size )
    {
    case SINGLEWORD :
      {
        rxc = 0;
        idx = (code>>0)&3;
        fsl = (code>>5)&3;
        r   = 1;
        c   = 1;
      }
      break;
    case PAIRWORD :
      {
        rxc = (code>>5)&1;
        idx = (code>>0)&3;
        fsl = (code>>5)&2;
        r   = (rxc==0) ? 2 : 1;
        c   = (rxc!=0) ? 2 : 1;
      }
      break;
    case TRIPLEWORD :
      {
        rxc = (code>>5)&1;
        idx = (code>>0)&3;
        fsl = (code>>6)&1;
        r   = (rxc==0) ? 3 : 1;
        c   = (rxc!=0) ? 3 : 1;
      }
      break;
    case QUADWORD :
      {
        rxc = (code>>5) & 1;
        idx = (code>>0) & 3;
```

```
        fsl = (code>>5) & 2;
        r   = (rxc==0) ? 4 : 1;
        c   = (rxc!=0) ? 4 : 1;
      }
      break;
    case PAIRXPAIRWORD :
      {
        rxc = ((code>>5) & 1) ^ 1;
        fsl = (code>>0) & 3;
        idx = (code>>5) & 2;
        r   = 2;
        c   = 2;
      }
      break;
    case TRIPLEXTRIPLEWORD :
      {
        rxc = ((code>>5) & 1) ^ 1;
        fsl = (code>>0) & 3;
        idx = (code>>6) & 1;
        r   = 3;
        c   = 3;
      }
      break;
    case QUADXQUADWORD :
      {
        rxc = ((code>>5) & 1) ^ 1;
        fsl = (code>>0) & 3;
        idx = (code>>5) & 2;
        r   = 4;
        c   = 4;
      }
      break;
  }

  {
    int i,j,k=0;
    if( rxc ) {
      for( i=0;i<r;i++ ) {
        for( j=0;j<c;j++ ) {
          rd[k+j] =MRF[mtx*4+((fsl+j)%4)+((idx+i)%4)*32];
        }
     k+=4;
      }
    } else {
      for( j=0;j<c;j++ ) {
        for( i=0;i<r;i++ ) {
          rd[k+i] =MRF[mtx*4+((idx+j)%4)+((fsl+i)%4)*32];
        }
        k+=4;
      }
    }
  }

  return rd;
}
```

```
void WriteMatrix( float *wd,int *wm,enum data_size size,int code )
{
  int mtx;
  int idx;
  int rxc;
  int fsl;
  int r,c;

  mtx = (code>>2)&7;

  switch( size )
    {
    case SINGLEWORD :
      {
        rxc = 0;
        idx = (code>>0)&3;
        fsl = (code>>5)&3;
        r   = 1;
        c   = 1;
      }
      break;
    case PAIRWORD :
      {
        rxc = (code>>5)&1;
        idx = (code>>0)&3;
        fsl = (code>>5)&2;
        r   = (rxc==0) ? 2 : 1;
        c   = (rxc!=0) ? 2 : 1;
      }
      break;
    case TRIPLEWORD :
      {
        rxc = (code>>5)&1;
        idx = (code>>0)&3;
        fsl = (code>>6)&1;
        r   = (rxc==0) ? 3 : 1;
        c   = (rxc!=0) ? 3 : 1;
      }
      break;
    case QUADWORD :
      {
        rxc = (code>>5) & 1;
        idx = (code>>0) & 3;
        fsl = (code>>5) & 2;
        r   = (rxc==0) ? 4 : 1;
        c   = (rxc!=0) ? 4 : 1;
      }
      break;
    case PAIRXPAIRWORD :
      {
        rxc = ((code>>5) & 1) ^ 1;
        fsl = (code>>0) & 3;
        idx = (code>>5) & 2;
```

```
        r   = 2;
        c   = 2;
      }
      break;
    case TRIPLEXTRIPLEWORD :
      {
        rxc = ((code>>5) & 1) ^ 1;
        fsl = (code>>0) & 3;
        idx = (code>>6) & 1;
        r   = 3;
        c   = 3;
      }
      break;
    case QUADXQUADWORD :
      {
        rxc = ((code>>5) & 1) ^ 1;
        fsl = (code>>0) & 3;
        idx = (code>>5) & 2;
        r   = 4;
        c   = 4;
      }
      break;
  }

  {
    int i,j,k=0;
    if( rxc ) {
      for( i=0;i<r;i++ ) {
        for( j=0;j<c;j++ ) {
          if( wm[mtx*4+((fsl+j)%4)+((idx+i)%4)*32]==0 ) {
            MRF[mtx*4+((fsl+j)%4)+((idx+i)%4)*32] = wd[k+j];
          }
        }
        k+=4;
      }
    } else {
      for( j=0;j<c;j++ ) {
        for( i=0;i<r;i++ ) {
          if( wm[mtx*4+((idx+j)%4)+((fsl+i)%4)*32]==0 ) {
            MRF[mtx*4+((idx+j)%4)+((fsl+i)%4)*32] = wd[k+j];
          }
        }
        k+=4;
      }
    }
  }
}
```

6.2. Source / Target / Destination Prefix

The VFPU provides the vpfxs, vpfxt and vpfxd instructions for decorating (performing

transformations on) the source, target and destination operands of instructions, respectively. These decorators, however, are only valid for a subset of VFPU instructions. Details are described later in this manual.

7. VFPU Prefixing

7.1. Introduction

The VFPU can apply the following decorators (transformation operators) to the source and target operands of a VFPU instruction.

- Swizzle
- Absolute value
- Constant insertion
- Negation

In addition, the VFPU can apply the following decorators to a destination operand.

- Saturation
- Write mask

However, these decorators can only be used with a subset of the VFPU instruction set.
See "8.5 Validity of Prefixing for Each Instruction"for a table showing which instructions can be decorated.

7.2. Prefixing Overview

Figure 7-1 shows the operation of prefixing in the VFPU.

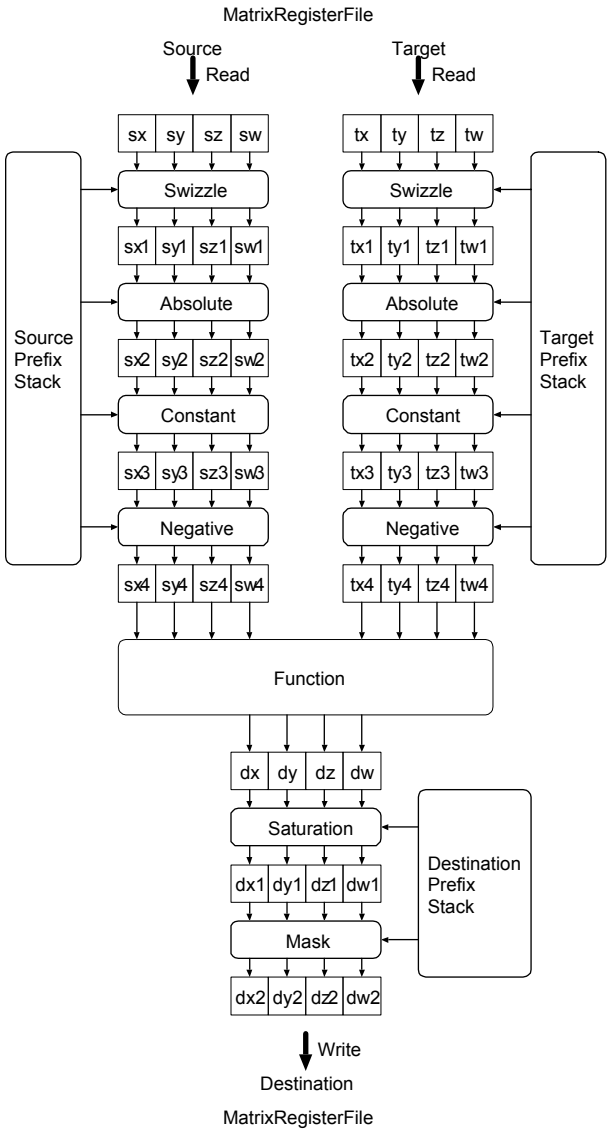


Figure 7-1: Prefixing Operation

Once decorator information is pushed onto the source and target prefix stacks, decorators will be applied to matrix register file source and target operands of subsequent VFPU instructions in the following order.

1. Swizzle
2. Absolute value
3. Constant insertion
4. Negation

After performing the desired operation, decorators are applied in the following order based on information from the destination prefix stack. The result after the decorator is applied is written to the matrix register file.

1. Saturation
2. Write mask

The vpfxs, vpfxt and vpfxd instructions are used for pushing decorator information onto the source, target and destination prefix stacks, respectively.
After a vpfx* instruction has executed, the effect is to only decorate the next instruction. However, no operation will be performed for the b*, mf*, mt*, lv*, sv*, vpfx*, vsync, vnop, vflush and vsync2 instructions because those instructions are not subject to decoration. After an instruction is decorated, the prefix stacks are initialized and no further decoration is performed until another vpfx* instruction is executed. To apply the same decoration to successive instructions, the same vpfx* instruction must be re-executed every time the decoration needs to be applied.
The modified operation can be defined in C as follows.

```
float src[4],tar[4],dst[4];

src =ReadMatrix(src, size, code);
src =PrefixSWZ(src, src_swz);
src =PrefixABS(src, src_abs);
src =PrefixCST(src, src_cst);
src =PrefixNEG(src, src_neg);

tar =ReadMatrix(tar, size, code);
tar =PrefixSWZ(tar, tar_swz);
tar =PrefixABS(tar, tar_abs);
tar =PrefixCST(tar, tar_cst);
tar =PrefixNEG(tar, tar_neg);

dst =Operation(src, tar);

dst =PrefixSAT(dst, dst_sat);
WriteMatrix(dst, dst_msk, size, code);
```

7.3. Decorators

The decorators are defined below.

7.3.1. Swizzle

The swizzle operation can be defined in C as follows.

```
void PrefixSWZ( float* x, float* y, float* z, float* w,
```

```
            float* x1,float* y1,float* z1,float* w1,
            int x_swz,int y_swz,int z_swz,int w_swz )
{
  switch( x_swz )
    {
    case 0 : *x1 =*x; break;
    case 1 : *x1 =*y; break;
    case 2 : *x1 =*z; break;
    case 3 : *x1 =*w; break;
    }

  switch( y_swz )
    {
    case 0 : *y1 =*x; break;
    case 1 : *y1 =*y; break;
    case 2 : *y1 =*z; break;
    case 3 : *y1 =*w; break;
    }

  switch( z_swz )
    {
    case 0 : *z1 =*x; break;
    case 1 : *z1 =*y; break;
    case 2 : *z1 =*z; break;
    case 3 : *z1 =*w; break;
    }

  switch( w_swz )
    {
    case 0 : *w1 =*x; break;
    case 1 : *w1 =*y; break;
    case 2 : *w1 =*z; break;
    case 3 : *w1 =*w; break;
    }
}
```

7.3.2. Absolute Value

The absolute value operation can be defined in C as follows.

```
void PrefixABS( float* x1,float* y1,float* z1,float* w1,
            float* x2,float* y2,float* z2,float* w2,
            int x_abs,int y_abs,int z_abs,int w_abs )
{
  if( x_abs==1 )
    *x2 =fabs( *x1 );
  else
    *x2 =*x1;

  if( y_abs==1 )
    *y2 =fabs( *y1 );
  else
    *y2 =*y1;

  if( z_abs==1 )
```

```
    *z2 =fabs( *z1 );
  else
    *z2 =*z1;

  if( w_abs==1 )
    *w2 =fabs( *w1 );
  else
    *w2 =*w1;
}
```

7.3.3. Constant Insertion

The constant insertion operation can be defined in C as follows.

```
void PrefixCST( float* x2,float* y2,float* z2,float* w2,
            float* x3,float* y3,float* z3,float* w3,
            int x_cst,int y_cst,int z_cst,int w_cst)
{
  static float cst[] =
    {
      0.0f,
      1.0f,
      2.0f,
      1.0f/2.0f,
      3.0f,
      1.0f/3.0f,
      1.0f/4.0f,
      1.0f/6.0f
    };

  if( (x_cst&0x8)==0x8 )
    *x3 =cst[ x_cst&0x7 ];
  else
    *x3 =*x2;

  if( (y_cst&0x8)==0x8 )
    *y3 =cst[ y_cst&0x7 ];
  else
    *y3 =*y2;

  if( (z_cst&0x8)==0x8 )
    *z3 =cst[ z_cst&0x7 ];
  else
    *z3 =*z2;

  if( (w_cst&0x8)==0x8 )
    *w3 =cst[ w_cst&0x7 ];
  else
    *w3 =*w2;
}
```

7.3.4. Negation

The negation operation can be defined in C as follows.

```
void PrefixNEG( float* x3,float* y3,float* z3,float* w3,
           float* x4,float* y4,float* z4,float* w4,
           int x_neg,int y_neg,int z_neg,int w_neg )
{
  if( x_neg==1 )
    *x4 =-(*x3);
  else
    *x4 =  *x3;

  if( y_neg==1 )
    *y4 =-(*y3);
  else
    *y4 =  *y3;

  if( z_neg==1 )
    *z4 =-(*z3);
  else
    *z4 =  *z3;

  if( w_neg==1 )
    *w4 =-(*w3);
  else
    *w4 =  *w3;
}
```

7.3.5. Saturation

The saturation operation can be defined in C as follows.

```
void PrefixSAT( float* x, float* y, float* z, float* w,
           float* x1,float* y1,float* z1,float* w1,
           int x_sat,int y_sat,int z_sat,int w_sat)
{
  switch( x_sat )
    {
    case 0 : *x1 =*x;                                              break;
    case 1 : *x1 =(*x>1.0f) ? 1.0f : (*x< 0.0f) ?  0.0f : *x; break;
    case 3 : *x1 =(*x>1.0f) ? 1.0f : (*x<-1.0f) ? -1.0f : *x; break;
    }

  switch( y_sat )
    {
    case 0 : *y1 =*y;                                              break;
    case 1 : *y1 =(*y>1.0f) ? 1.0f : (*y< 0.0f) ?  0.0f : *y; break;
    case 3 : *y1 =(*y>1.0f) ? 1.0f : (*y<-1.0f) ? -1.0f : *y; break;
    }

  switch( z_sat )
    {
    case 0 : *z1 =*z;                                              break;
    case 1 : *z1 =(*z>1.0f) ? 1.0f : (*z< 0.0f) ?  0.0f : *z; break;
    case 3 : *z1 =(*z>1.0f) ? 1.0f : (*z<-1.0f) ? -1.0f : *z; break;
    }
```

```
    switch( w_sat )
      {
      case 0 : *w1 =*w;                                                    break;
      case 1 : *w1 =(*w>1.0f) ? 1.0f : (*w< 0.0f) ?  0.0f : *w; break;
      case 3 : *w1 =(*w>1.0f) ? 1.0f : (*w<-1.0f) ? -1.0f : *w; break;
      }
  }
```

7.4. Prefix Stack

Figure 7-2 is a conceptual diagram showing how the prefix stack works. The vpfx* instructions push decorator information onto their respective prefix stack. Each prefix stack is only 1 entry deep. If a vpfx* instruction is executed when its prefix stack is already full, the decorator information on the stack will be overwritten. As a result, no stack overflow will occur. After a prefix stack has applied its decorator to a single instruction, the contents of the prefix stack are popped (deleted). However, the prefix stack is not affected by the b*, mf*, mt*, lv*, sv*, vpfx*, vsync, vnop, vflush or vsync2 instructions because these instructions are not subject to decoration. After an instruction is decorated, no further decoration is performed until another vpfx* instruction is executed and decorator information is again pushed onto the prefix stack. Note that a stack underflow will not occur.

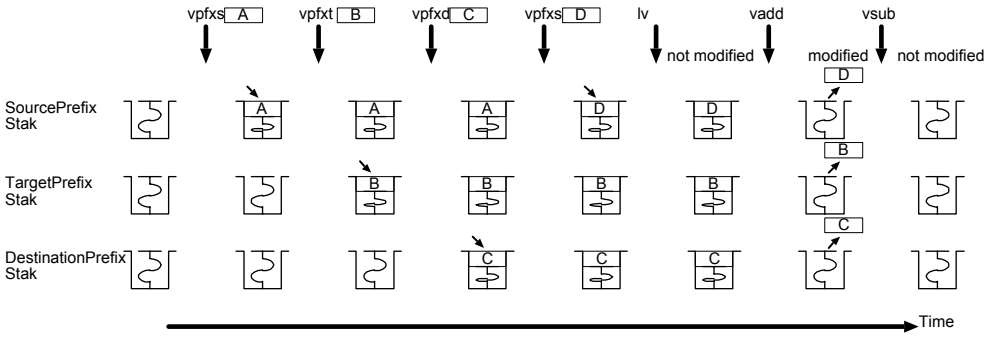


Figure 7-2: Operation of the Prefix Stack

Note that even if a VFPU instruction only uses a source operand, the target and destination prefix stacks will still be deleted. Likewise, if a VFPU instruction only uses a destination operand, the source and target prefix stacks will also be deleted.

The contents of the prefix stacks can be read and written through control registers VFPU_PFXS, VFPU_PFXT, and VFPU_PFXD. The prefix stacks must be saved when the context is switched. Figure 7-3 shows a conceptual diagram of the prefix stack during a context switch.

The mfvc instruction can be used to read the VFPU_PFXS, VFPU_PFXT, and VFPU_PFXD control registers when saving the context. When the mfvc instruction is executed, the contents

of the prefix stacks are not popped. If a prefix stack is empty when it is read, the initial value will be read without performing any decoration.

When restoring the context, the mtvc instruction can be used to write the VFPU_PFXS, VFPU_PFXT, and VFPU_PFXD control registers. If the mtvc instruction is executed when a prefix stack is already full, the decorator information will be overwritten. For this case as well, a stack overflow will not occur.

In the initial state after a reset, each prefix stack is empty.

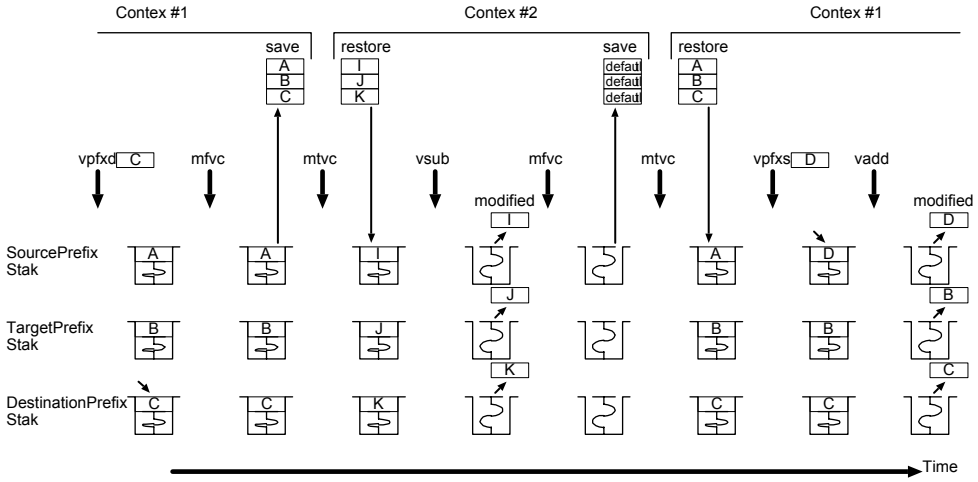


Figure 7-3: Prefix Stacks During Context Switching

7.4.1. Prefix Stack Format

The format of the prefix stacks is shown below.

Format of source and target prefix stacks

Bit field	Code type	Description
[1:0]	VFPU_SWZ or VFPU_CST[1:0]	X field swizzle or constant insertion setting
[3:2]	VFPU_SWZ or VFPU_CST[1:0]	Y field swizzle or constant insertion setting
[5:4]	VFPU_SWZ or VFPU_CST[1:0]	Z field swizzle or constant insertion setting
[7:6]	VFPU_SWZ or VFPU_CST[1:0]	W field swizzle or constant insertion setting
[8]	VFPU_ABS or VFPU_CST[2]	X field absolute value or constant insertion setting
[9]	VFPU_ABS or VFPU_CST[2]	Y field absolute value or constant insertion setting
[10]	VFPU_ABS or VFPU_CST[2]	Z field absolute value or constant insertion setting
[11]	VFPU_ABS or VFPU_CST[2]	W field absolute value or constant insertion setting
[12]	VFPU_CST[3]	X field constant insertion setting
[13]	VFPU_CST[3]	Y field constant insertion setting
[14]	VFPU_CST[3]	Z field constant insertion setting
[15]	VFPU_CST[3]	W field constant insertion setting
[16]	VFPU_NEG	X field negation setting
[17]	VFPU_NEG	Y field negation setting
[18]	VFPU_NEG	Z field negation setting
[19]	VFPU_NEG	W field negation setting

When VFPU_CST[3] = 1, VFPU_CST[2] and VFPU_CST[1:0] are used. When VFPU_CST[3] = 0, VFPU_ABS and VFPU_SWZ are used.

Format of destination prefix stack

Bit field	Code type	Description
[1:0]	VFPU_SAT	X field saturation setting
[3:2]	VFPU_SAT	Y field saturation setting
[5:4]	VFPU_SAT	Z field saturation setting
[7:6]	VFPU_SAT	W field saturation setting
[8]	VFPU_MSK	X field write mask setting
[9]	VFPU_MSK	Y field write mask setting
[10]	VFPU_MSK	Z field write mask setting
[11]	VFPU_MSK	W field write mask setting

Description of Code Types

VFPU_SWZ

Swizzle setting codes are defined as follows.

Code	Mnemonic	Function
0	X	Select X field
1	Y	Select Y field
2	Z	Select Z field
3	W	Select W field

VFPU_ABS

Absolute value setting codes are defined as follows.

Code	Mnemonic	Function
0		As is
1	| |	Absolute value

VFPU_CST

Constant insertion setting codes are defined as follows.

Code	Mnemonic	Function
8	0	0
9	1	1
10	2	2
11	1/2	$\frac{1}{2}$
12	3	3
13	1/3	$\frac{1}{3}$
14	1/4	$\frac{1}{4}$
15	1/6	$\frac{1}{6}$

VFPU_NEG

Negation setting codes are defined as follows.

Code	Mnemonic	Function
0		×1
1	–	×(-1)

VFPU_SAT

Saturation setting codes are defined as follows.

Code	Mnemonic	Function
0		No saturation
1	0	Saturate at [0:1] segment
3	1	Saturate at [-1:1] segment

VFPU_MSK

Write mask setting codes are defined as follows.

Code	Mnemonic	Function
0		Write
1	M	Do not write

7.5. Prefix Instruction Formats

7.5.1. Source and Target Prefix Instruction Formats

Source and target prefix instructions use the following formats.

```
vpfxs rpx,rpy,rpz,rpw
vpfxt rpx,rpy,rpz,rpw
```

The following list shows the mnemonics and corresponding codes for the rpx, rpy, rpz, rpw operands.

The final code is an OR of the individual codes for rpx, rpy, rpz and rpw.

Mnemonic	rpx	rpy	rpz	rpw
X	0x00000	0x00000	0x00000	0x00000
Y	0x00001	0x00004	0x00010	0x00040
Z	0x00002	0x00008	0x00020	0x00080
W	0x00003	0x0000C	0x00030	0x000C0
|X|	0x00100	0x00200	0x00400	0x00800
|Y|	0x00101	0x00204	0x00410	0x00840
|Z|	0x00102	0x00208	0x00420	0x00880
|W|	0x00103	0x0020C	0x00430	0x008C0
0	0x01000	0x02000	0x04000	0x08000
1	0x01001	0x02004	0x04010	0x08040
2	0x01002	0x02008	0x04020	0x08080
1/2	0x01003	0x0200C	0x04030	0x080C0
3	0x01100	0x02200	0x04400	0x08800
1/3	0x01101	0x02204	0x04410	0x08840
1/4	0x01102	0x02208	0x04420	0x08880
1/6	0x01103	0x0220C	0x04430	0x088C0
-X	0x10000	0x20000	0x40000	0x80000
-Y	0x10001	0x20004	0x40010	0x80040
-Z	0x10002	0x20008	0x40020	0x80080
-W	0x10003	0x2000C	0x40030	0x800C0
-|X|	0x10100	0x20200	0x40400	0x80800
-|Y|	0x10101	0x20204	0x40410	0x80840
-|Z|	0x10102	0x20208	0x40420	0x80880
-|W|	0x10103	0x2020C	0x40430	0x808C0

Mnemonic	rpx	rpy	rpz	rpw
-0	0x11000	0x22000	0x44000	0x88000
-1	0x11001	0x22004	0x44010	0x88040
-2	0x11002	0x22008	0x44020	0x88080
-1/2	0x11003	0x2200C	0x44030	0x880C0
-3	0x11100	0x22200	0x44400	0x88800
-1/3	0x11101	0x22204	0x44410	0x88840
-1/4	0x11102	0x22208	0x44420	0x88880
-1/6	0x11103	0x2220C	0x44430	0x888C0

7.5.2. Destination Prefix Instruction Format

The destination prefix instruction has the following format.

```
vpfxd wpx,wpy,wpz,wpw
```

The following list shows mnemonics and the corresponding codes for the wpx, wpy, wpz, and wpw operands.
The final code is an OR of the individual codes for wpx, wpy, wpz and wpw.

Mnemonic	wpx	wpy	wpz	wpw
	0x000	0x000	0x000	0x000
0	0x001	0x004	0x010	0x040
1	0x003	0x00C	0x030	0x0C0
M	0x100	0x200	0x400	0x800

7.6. Restrictions When Prefixing

7.6.1. Invalid Swizzle Settings

It is not permitted to specify a swizzle with an invalid field because the result of the operation is indeterminate.
For example, the code below uses the source prefix to apply a decoration after the 3-dimensional vector specified in the vadd.t instruction is read from the matrix register file.

```
vpfxs W,X,Y,
vadd.t c020,c000,c010
```

However, the W field of the swizzle has an indeterminate value. Consequently, the W field of the source will be invalid. As a result, the X field will also be invalid..

Figure 7-4 shows this example of a swizzle with an invalid specification.
To achieve the desired result, the following code should be used instead.

```
vpfxd ,,,M
vpfxs W,X,Y,
vadd.q c020,c000,c010
```

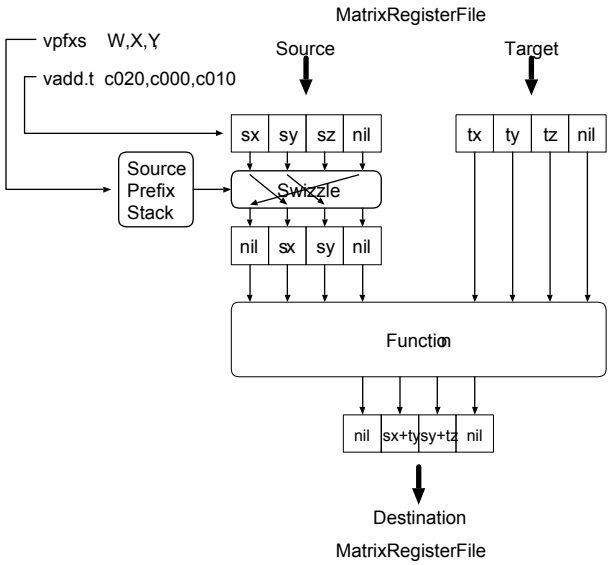


Figure 7-4: Example of Invalid Swizzle Setting

7.6.2. Pipeline Control

The prefix function was designed to apply decorations during normal pipeline operations. Consequently, when a hazard such as contention for a shared resource occurs during pipeline processing, a stall cannot be avoided even in cases where the prefix operation would seemingly remove the contention.
For example, the following code will generate a RAW hazard for the s022 element causing the vsub.s instruction to stall, even though the masking operation would appear to remove the hazard.

```
vpfxd ,,M,
vadd.q c020,c000,c010
vsub.s s033,s030,s022
```

8. APPENDIX

8.1. Correspondence Between Formats and Matrix Register Numbers

The tables below show the correspondence between the notation used to represent groups of elements in scalar (S***), vector (C***, R***) and matrix (E***, M***) formats and the matrix registers that contain those elements.

Operand	Scalar
S000	[$0]
S001	[$32]
S002	[$64]
S003	[$96]
S010	[$1]
S011	[$33]
S012	[$65]
S013	[$97]
S020	[$2]
S021	[$34]
S022	[$66]
S023	[$98]
S030	[$3]
S031	[$35]
S032	[$67]
S033	[$99]
S100	[$4]
S101	[$36]
S102	[$68]
S103	[$100]
S110	[$5]
S111	[$37]
S112	[$69]
S113	[$101]
S120	[$6]
S121	[$38]
S122	[$70]
S123	[$102]
S130	[$7]
S131	[$39]
S132	[$71]
S133	[$103]

Operand	Scalar
S200	[$8]
S201	[$40]
S202	[$72]
S203	[$104]
S210	[$9]
S211	[$41]
S212	[$73]
S213	[$105]
S220	[$10]
S221	[$42]
S222	[$74]
S223	[$106]
S230	[$11]
S231	[$43]
S232	[$75]
S233	[$107]
S300	[$12]
S301	[$44]
S302	[$76]
S303	[$108]
S310	[$13]
S311	[$45]
S312	[$77]
S313	[$109]
S320	[$14]
S321	[$46]
S322	[$78]
S323	[$110]
S330	[$15]
S331	[$47]
S332	[$79]
S333	[$111]
S400	[$16]
S401	[$48]
S402	[$80]
S403	[$112]
S410	[$17]
S411	[$49]
S412	[$81]
S413	[$113]
S420	[$18]
S421	[$50]
S422	[$82]
S423	[$114]
S430	[$19]
S431	[$51]
S432	[$83]
S433	[$115]

Operand	Scalar
S500	[$20]
S501	[$52]
S502	[$84]
S503	[$116]
S510	[$21]
S511	[$53]
S512	[$85]
S513	[$117]
S520	[$22]
S521	[$54]
S522	[$86]
S523	[$118]
S530	[$23]
S531	[$55]
S532	[$87]
S533	[$119]
S600	[$24]
S601	[$56]
S602	[$88]
S603	[$120]
S610	[$25]
S611	[$57]
S612	[$89]
S613	[$121]
S620	[$26]
S621	[$58]
S622	[$90]
S623	[$122]
S630	[$27]
S631	[$59]
S632	[$91]
S633	[$123]
S700	[$28]
S701	[$60]
S702	[$92]
S703	[$124]
S710	[$29]
S711	[$61]
S712	[$93]
S713	[$125]
S720	[$30]
S721	[$62]
S722	[$94]
S723	[$126]
S730	[$31]
S731	[$63]
S732	[$95]
S733	[$127]

Operand	2-dimensional vector	3-dimensional vector	4-dimensional vector
C000	[$0,$32]	[$0,$32,$64]	[$0,$32,$64,$96]
C001	-	[$32,$64,$96]	-
C002	[$64,$96]	-	[$64,$96,$0,$32]
C003	-	-	-
C010	[$1,$33]	[$1,$33,$65]	[$1,$33,$65,$97]
C011	-	[$33,$65,$97]	-
C012	[$65,$97]	-	[$65,$97,$1,$33]
C013	-	-	-
C020	[$2,$34]	[$2,$34,$66]	[$2,$34,$66,$98]
C021	-	[$34,$66,$98]	-
C022	[$66,$98]	-	[$66,$98,$2,$34]
C023	-	-	-
C030	[$3,$35]	[$3,$35,$67]	[$3,$35,$67,$99]
C031	-	[$35,$67,$99]	-
C032	[$67,$99]	-	[$67,$99,$3,$35]
C033	-	-	-
C100	[$4,$36]	[$4,$36,$68]	[$4,$36,$68,$100]
C101	-	[$36,$68,$100]	-
C102	[$68,$100]	-	[$68,$100,$4,$36]
C103	-	-	-
C110	[$5,$37]	[$5,$37,$69]	[$5,$37,$69,$101]
C111	-	[$37,$69,$101]	-
C112	[$69,$101]	-	[$69,$101,$5,$37]
C113	-	-	-
C120	[$6,$38]	[$6,$38,$70]	[$6,$38,$70,$102]
C121	-	[$38,$70,$102]	-
C122	[$70,$102]	-	[$70,$102,$6,$38]
C123	-	-	-
C130	[$7,$39]	[$7,$39,$71]	[$7,$39,$71,$103]
C131	-	[$39,$71,$103]	-
C132	[$71,$103]	-	[$71,$103,$7,$39]
C133	-	-	-
C200	[$8,$40]	[$8,$40,$72]	[$8,$40,$72,$104]
C201	-	[$40,$72,$104]	-
C202	[$72,$104]	-	[$72,$104,$8,$40]
C203	-	-	-
C210	[$9,$41]	[$9,$41,$73]	[$9,$41,$73,$105]
C211	-	[$41,$73,$105]	-
C212	[$73,$105]	-	[$73,$105,$9,$41]
C213	-	-	-
C220	[$10,$42]	[$10,$42,$74]	[$10,$42,$74,$106]
C221	-	[$42,$74,$106]	-
C222	[$74,$106]	-	[$74,$106,$10,$42]
C223	-	-	-
C230	[$11,$43]	[$11,$43,$75]	[$11,$43,$75,$107]
C231	-	[$43,$75,$107]	-
C232	[$75,$107]	-	[$75,$107,$11,$43]
C233	-	-	-

Operand	2-dimensional vector	3-dimensional vector	4-dimensional vector
C300	[$12,$44]	[$12,$44,$76]	[$12,$44,$76,$108]
C301	-	[$44,$76,$108]	-
C302	[$76,$108]	-	[$76,$108,$12,$44]
C303	-	-	-
C310	[$13,$45]	[$13,$45,$77]	[$13,$45,$77,$109]
C311	-	[$45,$77,$109]	-
C312	[$77,$109]	-	[$77,$109,$13,$45]
C313	-	-	-
C320	[$14,$46]	[$14,$46,$78]	[$14,$46,$78,$110]
C321	-	[$46,$78,$110]	-
C322	[$78,$110]	-	[$78,$110,$14,$46]
C323	-	-	-
C330	[$15,$47]	[$15,$47,$79]	[$15,$47,$79,$111]
C331	-	[$47,$79,$111]	-
C332	[$79,$111]	-	[$79,$111,$15,$47]
C333	-	-	-
C400	[$16,$48]	[$16,$48,$80]	[$16,$48,$80,$112]
C401	-	[$48,$80,$112]	-
C402	[$80,$112]	-	[$80,$112,$16,$48]
C403	-	-	-
C410	[$17,$49]	[$17,$49,$81]	[$17,$49,$81,$113]
C411	-	[$49,$81,$113]	-
C412	[$81,$113]	-	[$81,$113,$17,$49]
C413	-	-	-
C420	[$18,$50]	[$18,$50,$82]	[$18,$50,$82,$114]
C421	-	[$50,$82,$114]	-
C422	[$82,$114]	-	[$82,$114,$18,$50]
C423	-	-	-
C430	[$19,$51]	[$19,$51,$83]	[$19,$51,$83,$115]
C431	-	[$51,$83,$115]	-
C432	[$83,$115]	-	[$83,$115,$19,$51]
C433	-	-	-
C500	[$20,$52]	[$20,$52,$84]	[$20,$52,$84,$116]
C501	-	[$52,$84,$116]	-
C502	[$84,$116]	-	[$84,$116,$20,$52]
C503	-	-	-
C510	[$21,$53]	[$21,$53,$85]	[$21,$53,$85,$117]
C511	-	[$53,$85,$117]	-
C512	[$85,$117]	-	[$85,$117,$21,$53]
C513	-	-	-
C520	[$22,$54]	[$22,$54,$86]	[$22,$54,$86,$118]
C521	-	[$54,$86,$118]	-
C522	[$86,$118]	-	[$86,$118,$22,$54]
C523	-	-	-
C530	[$23,$55]	[$23,$55,$87]	[$23,$55,$87,$119]
C531	-	[$55,$87,$119]	-
C532	[$87,$119]	-	[$87,$119,$23,$55]
C533	-	-	-

Operand	2-dimensional vector	3-dimensional vector	4-dimensional vector
C600	[$24,$56]	[$24,$56,$88]	[$24,$56,$88,$120]
C601	-	[$56,$88,$120]	-
C602	[$88,$120]	-	[$88,$120,$24,$56]
C603	-	-	-
C610	[$25,$57]	[$25,$57,$89]	[$25,$57,$89,$121]
C611	-	[$57,$89,$121]	-
C612	[$89,$121]	-	[$89,$121,$25,$57]
C613	-	-	-
C620	[$26,$58]	[$26,$58,$90]	[$26,$58,$90,$122]
C621	-	[$58,$90,$122]	-
C622	[$90,$122]	-	[$90,$122,$26,$58]
C623	-	-	-
C630	[$27,$59]	[$27,$59,$91]	[$27,$59,$91,$123]
C631	-	[$59,$91,$123]	-
C632	[$91,$123]	-	[$91,$123,$27,$59]
C633	-	-	-
C700	[$28,$60]	[$28,$60,$92]	[$28,$60,$92,$124]
C701	-	[$60,$92,$124]	-
C702	[$92,$124]	-	[$92,$124,$28,$60]
C703	-	-	-
C710	[$29,$61]	[$29,$61,$93]	[$29,$61,$93,$125]
C711	-	[$61,$93,$125]	-
C712	[$93,$125]	-	[$93,$125,$29,$61]
C713	-	-	-
C720	[$30,$62]	[$30,$62,$94]	[$30,$62,$94,$126]
C721	-	[$62,$94,$126]	-
C722	[$94,$126]	-	[$94,$126,$30,$62]
C723	-	-	-
C730	[$31,$63]	[$31,$63,$95]	[$31,$63,$95,$127]
C731	-	[$63,$95,$127]	-
C732	[$95,$127]	-	[$95,$127,$31,$63]
C733	-	-	-
R000	[$0,$1]	[$0,$1,$2]	[$0,$1,$2,$3]
R001	[$32,$33]	[$32,$33,$34]	[$32,$33,$34,$35]
R002	[$64,$65]	[$64,$65,$66]	[$64,$65,$66,$67]
R003	[$96,$97]	[$96,$97,$98]	[$96,$97,$98,$99]
R010	-	[$1,$2,$3]	-
R011	-	[$33,$34,$35]	-
R012	-	[$65,$66,$67]	-
R013	-	[$97,$98,$99]	-
R020	[$2,$3]	-	[$2,$3,$0,$1]
R021	[$34,$35]	-	[$34,$35,$32,$33]
R022	[$66,$67]	-	[$66,$67,$64,$65]
R023	[$98,$99]	-	[$98,$99,$96,$97]
R030	-	-	-
R031	-	-	-
R032	-	-	-
R033	-	-	-

Operand	2-dimensional vector	3-dimensional vector	4-dimensional vector
R100	[$4,$5]	[$4,$5,$6]	[$4,$5,$6,$7]
R101	[$36,$37]	[$36,$37,$38]	[$36,$37,$38,$39]
R102	[$68,$69]	[$68,$69,$70]	[$68,$69,$70,$71]
R103	[$100,$101]	[$100,$101,$102]	[$100,$101,$102,$103]
R110	-	[$5,$6,$7]	-
R111	-	[$37,$38,$39]	-
R112	-	[$69,$70,$71]	-
R113	-	[$101,$102,$103]	-
R120	[$6,$7]	-	[$6,$7,$4,$5]
R121	[$38,$39]	-	[$38,$39,$36,$37]
R122	[$70,$71]	-	[$70,$71,$68,$69]
R123	[$102,$103]	-	[$102,$103,$100,$101]
R130	-	-	-
R131	-	-	-
R132	-	-	-
R133	-	-	-
R200	[$8,$9]	[$8,$9,$10]	[$8,$9,$10,$11]
R201	[$40,$41]	[$40,$41,$42]	[$40,$41,$42,$43]
R202	[$72,$73]	[$72,$73,$74]	[$72,$73,$74,$75]
R203	[$104,$105]	[$104,$105,$106]	[$104,$105,$106,$107]
R210	-	[$9,$10,$11]	-
R211	-	[$41,$42,$43]	-
R212	-	[$73,$74,$75]	-
R213	-	[$105,$106,$107]	-
R220	[$10,$11]	-	[$10,$11,$8,$9]
R221	[$42,$43]	-	[$42,$43,$40,$41]
R222	[$74,$75]	-	[$74,$75,$72,$73]
R223	[$106,$107]	-	[$106,$107,$104,$105]
R230	-	-	-
R231	-	-	-
R232	-	-	-
R233	-	-	-
R300	[$12,$13]	[$12,$13,$14]	[$12,$13,$14,$15]
R301	[$44,$45]	[$44,$45,$46]	[$44,$45,$46,$47]
R302	[$76,$77]	[$76,$77,$78]	[$76,$77,$78,$79]
R303	[$108,$109]	[$108,$109,$110]	[$108,$109,$110,$111]
R310	-	[$13,$14,$15]	-
R311	-	[$45,$46,$47]	-
R312	-	[$77,$78,$79]	-
R313	-	[$109,$110,$111]	-
R320	[$14,$15]	-	[$14,$15,$12,$13]
R321	[$46,$47]	-	[$46,$47,$44,$45]
R322	[$78,$79]	-	[$78,$79,$76,$77]
R323	[$110,$111]	-	[$110,$111,$108,$109]
R330	-	-	-
R331	-	-	-
R332	-	-	-
R333	-	-	-

Operand	2-dimensional vector	3-dimensional vector	4-dimensional vector
R400	[$16,$17]	[$16,$17,$18]	[$16,$17,$18,$19]
R401	[$48,$49]	[$48,$49,$50]	[$48,$49,$50,$51]
R402	[$80,$81]	[$80,$81,$82]	[$80,$81,$82,$83]
R403	[$112,$113]	[$112,$113,$114]	[$112,$113,$114,$115]
R410	-	[$17,$18,$19]	-
R411	-	[$49,$50,$51]	
R412		[$81,$82,$83]	-
R413		[$113,$114,$115]	-
R420	[$18,$19]	-	[$18,$19,$16,$17]
R421	[$50,$51]	-	[$50,$51,$48,$49]
R422	[$82,$83]	-	[$82,$83,$80,$81]
R423	[$114,$115]	-	[$114,$115,$112,$113]
R430	-	-	-
R431	-	-	-
R432	-	-	-
R433	-	-	-
R500	[$20,$21]	[$20,$21,$22]	[$20,$21,$22,$23]
R501	[$52,$53]	[$52,$53,$54]	[$52,$53,$54,$55]
R502	[$84,$85]	[$84,$85,$86]	[$84,$85,$86,$87]
R503	[$116,$117]	[$116,$117,$118]	[$116,$117,$118,$119]
R510	-	[$21,$22,$23]	-
R511	-	[$53,$54,$55]	-
R512	-	[$84,$85,$87]	-
R513	-	[$117,$118,$119]	-
R520	[$22,$23]	-	[$22,$23,$20,$21]
R521	[$54,$55]	-	[$54,$55,$52,$53]
R522	[$86,$87]	-	[$86,$87,$84,$85]
R523	[$118,$119]	-	[$118,$119,$116,$117]
R530	-	-	-
R531	-	-	-
R532	-	-	-
R533	-	-	-
R600	[$24,$25]	[$24,$25,$26]	[$24,$25,$26,$27]
R601	[$56,$57]	[$56,$57,$58]	[$56,$57,$58,$59]
R602	[$88,$89]	[$88,$89,$90]	[$88,$89,$90,$91]
R603	[$120,$121]	[$120,$121,$122]	[$120,$121,$122,$123]
R610	-	[$25,$26,$27]	-
R611	-	[$57,$58,$59]	-
R612	-	[$89,$90,$91]	-
R613	-	[$121,$122,$123]	-
R620	[$26,$27]	-	[$26,$27,$24,$25]
R621	[$58,$59]	-	[$58,$59,$56,$57]
R622	[$90,$91]	-	[$90,$91,$88,$89]
R623	[$122,$123]	-	[$122,$123,$120,$121]
R630	-	-	-
R631	-	-	-
R632	-	-	-
R633	-	-	-

Operand	2-dimensional vector	3-dimensional vector	4-dimensional vector
R700	[$28,$29]	[$28,$29,$30]	[$28,$29,$30,$31]
R701	[$60,$61]	[$60,$61,$62]	[$60,$61,$62,$63]
R702	[$92,$93]	[$92,$93,$94]	[$92,$93,$94,$95]
R703	[$124,$125]	[$124,$125,$126]	[$124,$125,$126.$127]
R710	-	[$29,$30,$31]	-
R711	-	[$61,$62,$63]	-
R712	-	[$93,$94,$95]	-
R713	-	[125,$126.$127]	-
R720	[$30,$31]	-	[$30,$31,$28,$29]
R721	[$62,$63]	-	[$62,$63,$60,$61]
R722	[$94,$95]	-	[$94,$95,$92,$93]
R723	[$126,$127]	-	[$126.$127,$124,$125]
R730	-	-	-
R731	-	-	-
R732	-	-	-
R733	-	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E000	[[$0, $32], [$1, $33]]	[[$0, $32, $64], [$1, $33, $65], [$2, $34, $66]]	[[$0, $32, $64, $96], [$1, $33, $65, $97], [$2, $34, $66, $98], [$3, $35, $67, $99]]
E001	-	[[$32, $64, $96], [$33, $65, $97], [$34, $66, $98]]	-
E002	[[$64, $96], [$65, $97]]	-	-
E003	-	-	-
E010	-	[[$1, $33, $65], [$2, $34, $66], [$3, $35, $67]]	-
E011	-	[[$33, $65, $97], [$34, $66, $98], [$35, $67, $99]]	-
E012	-	-	-
E013	-	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E020	[[$2, $34], [$3, $35]]	-	-
E021	-	-	-
E022	[[$66, $98], [$67, $99]]	-	-
E023	-	-	-
E030	-	-	-
E031	-	-	-
E032	-	-	-
E033	-	-	-
E100	[[$4, $36], [$5, $37]]	[[$4, $36, $68], [$5, $37, $69], [$6, $38, $70]]	[[$4, $36, $68,$100], [$5, $37, $69,$101], [$6, $38, $70,$102], [$7, $39, $71,$103]]
E101	-	[[$36, $68,$100], [$37, $69,$101], [$38, $70,$102]]	-
E102	[[$68,$100], [$69,$101]]	-	-
E103	-	-	-
E110	-	[[$5, $37, $69], [$6, $38, $70], [$7, $39, $71]]	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E111	-	[[$37, $69,$101], [$38, $70,$102], [$39, $71,$103]]	-
E112	-	-	-
E113	-	-	-
E120	[[$6,$38], [$7,$39]]	-	-
E121	-	-	-
E122	[[$70,$102], [$71,$103]]	-	-
E123	-	-	-
E130	-	-	-
E131	-	-	-
E132	-	-	-
E133	-	-	-
E200	[[$8, $40], [$9, $41]]	[[$8, $40, $72], [$9, $41, $73], [$10, $42, $74]]	[[$8, $40, $72,$104], [$9, $41, $73,$105], [$10, $42, $74,$106], [$11, $43, $75,$107]]
E201	-	[[$40, $72,$104], [$41, $73,$105], [$42, $74,$106]]	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E202	[[$72,$104], [$73,$105]]	-	-
E203	-	-	-
E210	-	[[$9, $41, $73], [$10, $42, $74], [$11, $43, $75]]	-
E211	-	[[$41, $73,$105], [$42, $74,$106], [$43, $75,$107]]	-
E212	-	-	-
E213	-	-	-
E220	[[$10, $42], [$11, $43]]	-	-
E221	-	-	-
E222	[[$74,$106], [$75,$107]]	-	-
E223	-	-	-
E230	-	-	-
E231	-	-	-
E232	-	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E233	-	-	-
E300	[[$12, $44], [$13, $45]]	[[$12, $44, $76], [$13, $45, $77], [$14, $46, $78]]	[[$12, $44, $76,$108], [$13, $45, $77,$109], [$14, $46, $78,$110], [$15, $47, $79,$111]]
E301	-	[[$44, $76,$108], [$45, $77,$109], [$46, $78,$110]]	-
E302	[[$76,$108], [$77,$109]]	-	-
E303	-	-	-
E310	-	[[$13, $45, $77], [$14, $46, $78], [$15, $47, $79]]	-
E311	-	[[$45, $77,$109], [$46, $78,$110], [$47, $79,$111]]	-
E312	-	-	-
E313	-	-	-
E320	[[$14, $46], [$15, $47]]	-	-
E321	-	-	-
E322	[[$78,$110], [$79,$111]]	-	-
E323	-	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E330	-	-	-
E331	-	-	-
E332	-	-	-
E333	-	-	-
E400	[[$16, $48], [$17, $49]]	[[$16, $48, $80], [$17, $49, $81], [$18, $50, $82]]	[[$16, $48, $80,$112], [$17, $49, $81,$113], [$18, $50, $82,$114], [$19, $51, $83,$115]]
E401	-	[[$48, $80,$112], [$49, $81,$113], [$50, $82,$114]]	-
E402	[[$80,$112], [$81,$113]]	-	-
E403	-	-	-
E410	-	[[$17, $49, $81], [$18, $50, $82], [$19, $51, $83]]	-
E411	-	[[$49, $81,$113], [$50, $82,$114], [$51, $83,$115]]	-
E412	-	-	-
E413	-	-	-
E420	[[$18, $50], [$19, $51]]	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E421	-	-	-
E422	[[$82,$114], [$83,$115]]	-	-
E423	-	-	-
E430	-	-	-
E431	-	-	-
E432	-	-	-
E433	-	-	-
E500	[[$20, $52], [$21, $53]]	[[$20, $52, $84], [$21, $53, $85], [$22, $54, $86]]	[[$20, $52, $84,$116], [$21, $53, $85,$117], [$22, $54, $86,$118], [$23, $55, $87,$119]]
E501	-	[[$52, $84,$116], [$53, $85,$117], [$54, $86,$118]]	-
E502	[[$84,$116], [$85,$117]]	-	-
E503	-	-	-
E510	-	[[$21, $53, $85], [$22, $54, $86], [$23, $55, $87]]	-
E511	-	[[$53, $85,$117], [$54, $86,$118], [$55, $87,$119]]	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E512	-	-	-
E513	-	-	-
E520	[[$22, $54], [$23, $55]]	-	-
E521	-	-	-
E522	[[$86,$118], [$87,$119]]	-	-
E523	-	-	-
E530	-	-	-
E531	-	-	-
E532	-	-	-
E533	-	-	-
E600	[[$24, $56], [$25, $57]]	[[$24, $56, $88], [$25, $57, $89], [$26, $58, $90]]	[[$24, $56, $88,$120], [$25, $57, $89,$121], [$26, $58, $90,$122], [$27, $59, $91,$123]]
E601	-	[[$56, $88,$120], [$57, $89,$121], [$58, $90,$122]]	-
E602	[[$88,$120], [$89,$121]]	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E603	-	-	-
E610	-	[[$25, $57, $89], [$26, $58, $90], [$27, $59, $91]]	-
E611	-	[[$57, $89,$121], [$58, $90,$122], [$59, $91,$123]]	-
E612	-	-	-
E613	-	-	-
E620	[[$26, $58], [$27, $59]]	-	-
E621	-	-	-
E622	[[$90,$122], [$91,$123]]	-	-
E623	-	-	-
E630	-	-	-
E631	-	-	-
E632	-	-	-
E633	-	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E700	[[$28, $60], [$29, $61]]	[[$28, $60, $92], [$29, $61, $93], [$30, $62, $94]]	[[$28, $60, $92,$124], [$29, $61, $93,$125], [$30, $62, $94,$126], [$31, $63, $95,$127]]
E701	-	[[$60, $92,$124], [$61, $93,$125], [$62, $94,$126]]	-
E702	[[$92,$124], [$93,$125]]	-	-
E703	-	-	-
E710	-	[[$29, $61, $93], [$30, $62, $94], [$31, $63, $95]]	-
E711	-	[[$61, $93,$125], [$62, $94,$126], [$63, $95,$127]]	-
E712	-	-	-
E713	-	-	-
E720	[[$30, $62], [$31, $63]]	-	-
E721	-	-	-
E722	[[$94,$126], [$95,$127]]	-	-
E723	-	-	-
E730	-	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E731	-	-	-
E732	-	-	-
E733	-	-	-
M000	[[$0, $1], [$32, $33]]	[[$0, $1, $2], [$32, $33, $34], [$64, $65, $66]]	[[$0, $1, $2, $3], [$32, $33, $34, $35], [$64, $65, $66, $67], [$96, $97, $98, $99]]
M001	-	[[$32, $33, $34], [$64, $65, $66], [$96, $97, $98]]	-
M002	[[$64, $65], [$96, $97]]	-	-
M003	-	-	-
M010	-	[[$1, $2, $3], [$33, $34, $35], [$65, $66, $67]]	-
M011	-	[[$33, $34, $35], [$65, $66, $67], [$97, $98, $99]]	-
M012	-	-	-
M013	-	-	-
M020	[[$2, $3], [$34, $35]]	-	-
M021	-	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
M022	[[$66, $67], [$98, $99]]	-	-
M023	-	-	-
M030	-	-	-
M031	-	-	-
M032	-	-	-
M033	-	-	-
M100	[[$4, $5], [$36, $37]]	[[$4, $5, $6], [$36, $37, $38], [$68, $69, $70]]	[[$4, $5, $6, $7], [$36, $37, $38, $39], [$68, $69, $70, $71], [$100,$101,$102,$103]]
M101	-	[[$36, $37, $38], [$68, $69, $70], [$100,$101,$102]]	-
M102	[[$68, $69], [$100,$101]]	-	-
M103	-	-	-
M110	-	[[$5, $6, $7], [$37, $38, $39], [$69, $70, $71]]	-
M111	-	[[$37, $38, $39], [$69, $70, $71], [$101,$102,$103]]	-
M112	-	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
M113	-	-	-
M120	[[$6, $7], [$38, $39]]	-	-
M121	-	-	-
M122	[[$70, $71], [$102,$103]]	-	-
M123	-	-	-
M130		-	-
M131		-	-
M132		-	-
M133	-	-	-
M200	[[$8, $9], [$40, $41]]	[[$8, $9, $10], [$40, $41, $42], [$72, $73, $74]]	[[$8, $9, $10, $11], [$40, $41, $42, $43], [$72, $73, $74, $75], [$104,$105,$106,$107]]
M201	-	[[$40, $41, $42], [$72, $73, $74], [$104,$105,$106]]	-
M202	[[$72, $73], [$104,$105]]	-	-
M203	-	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
M210	-	[[$9, $10, $11], [$41, $42, $43], [$73, $74, $75]]	-
M211	-	[[$41, $42, $43], [$73, $74, $75], [$105,$106,$107]]	-
M212	-	-	-
M213	-	-	-
M220	[[$10, $11], [$42, $43]]	-	-
M221	-	-	-
M222	[[$74, $75], [$106,$107]]	-	-
M223	-	-	-
M230	-	-	-
M231	-	-	-
M232	-	-	-
M233	-	-	-
M300	[[$12, $13], [$44, $45]]	[[$12, $13, $14], [$44, $45, $46], [$76, $77, $78]]	[[$12, $13, $14, $15], [$44, $45, $46, $47], [$76, $77, $78, $79], [$108,$109,$110,$111]]

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
M301	-	[[$44, $45, $46], [$76, $77, $78], [$108,$109,$110]]	-
M302	[[$76, $77], [$108,$109]]	-	-
M303	-	-	-
M310	-	[[$13, $14, $15], [$45, $46, $47], [$77, $78, $79]]	-
M311	-	[[$45, $46, $47], [$77, $78, $79], [$109,$110,$111]]	-
M312	-	-	-
M313	-	-	-
M320	[[$14, $15], [$46, $47]]	-	-
M321	-	-	-
M322	[[$78, $79], [$110,$111]]	-	-
M323	-	-	-
M330	-	-	-
M331	-	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
M332	-	-	
M333	-	-	-
M400	[[$16, $17], [$48, $49]]	[[$16, $17, $18], [$48, $49, $50], [$80, $81, $82]]	[[$16, $17, $18, $19], [$48, $49, $50, $51], [$80, $81, $82, $83], [$112,$113,$114,$115]]
M401	-	[[$48, $49, $50], [$80, $81, $82], [$112,$113,$114]]	-
M402	[[$80, $81], [$112,$113]]	-	-
M403	-	-	-
M410	-	[[$17, $18, $19], [$49, $50, $51], [$81, $82, $83]]	-
M411	-	[[$49, $50, $51], [$81, $82, $83], [$113,$114,$115]]	-
M412	-	-	-
M413	-	-	-
M420	[[$18, $19], [$50, $51]]	-	-
M421	-	-	-
M422	[[$82, $83], [$114,$115]]	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
M423	-	-	-
M430	-	-	-
M431	-	-	-
M432	-	-	-
M433	-	-	-
M500	[[$20, $21], [$52, $53]]	[[$20, $21, $22], [$52, $53, $54], [$84, $85, $86]]	[[$20, $21, $22, $23], [$52, $53, $54, $55], [$84, $85, $86, $87], [$116,$117,$118,$119]]
M501	-	[[$52, $53, $54], [$84, $85, $86], [$116,$117,$118]]	-
M502	[[$84, $85], [$116,$117]]	-	-
M503	-	-	-
M510	-	[[$21, $22, $23], [$53, $54, $55], [$85, $86, $87]]	-
M511	-	[[$53, $54, $55], [$85, $86, $87], [$117,$118,$119]]	-
M512	-	-	-
M513	-	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
M520	[[$22, $23], [$54, $55]]	-	-
M521	-	-	-
M522	[[$86, $87], [$118,$119]]	-	-
M523	-	-	-
M530	-	-	-
M531	-	-	-
M532	-	-	-
M533	-	-	-
M600	[[$24, $25], [$56, $57]]	[[$24, $25, $26], [$56, $57, $58], [$88, $89, $90]]	[[$24, $25, $26, $27], [$56, $57, $58, $59], [$88, $89, $90, $91], [$120,$121,$122,$123]]
M601	-	[[$56, $57, $58], [$88, $89, $90], [$120,$121,$122]]	-
M602	[[$88, $89], [$120,$121]]	-	-
M603	-	-	-
M610	-	[[$25, $26, $27], [$57, $58, $59], [$89, $90, $91]]	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
M611	-	[[$57, $58, $59], [$89, $90, $91], [$121,$122,$123]]	-
M612	-	-	-
M613	-	-	-
M620	[[$26, $27], [$58, $59]]	-	-
M621	-	-	-
M622	[[$90, $91], [$122,$123]]	-	-
M623	-	-	-
M630	-	-	-
M631	-	-	-
M632	-	-	-
M633	-	-	-
M700	[[$28, $29], [$60, $61]]	[[$28, $29, $30], [$60, $61, $62], [$92, $93, $94]]	[[$28, $29, $30, $31], [$60, $61, $62, $63], [$92, $93, $94, $95], [$124,$125,$126,$127]]
M701	-	[[$60, $61, $62], [$92, $93, $94], [$124,$125,$126]]	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
M702	[[$92, $93], [$124,$125]]	-	-
M703	-	-	-
M710	-	[[$29, $30, $31], [$61, $62, $63], [$93, $94, $95]]	-
M711	-	[[$61, $62, $63], [$93, $94, $95], [$125,$126,$127]]	-
M712	-	-	-
M713	-	-	-
M720	[[$30, $31], [$62, $63]]	-	-
M721	-	-	-
M722	[[$94, $95], [$126,$127]]	-	-
M723	-	-	-
M730	-	-	-
M731	-	-	-
M732	-	-	-

Operand	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
M733	-	-	-

8.2. Number of Execution Cycles Required for Each Instruction

The number of execution cycles required for each instruction is defined by the latency and pitch, as shown below.

Instruction	Latency	Pitch	Type
bvf	0	1	CPU interlock instruction
bvfl	0	1	CPU interlock instruction
bvt	0	1	CPU interlock instruction
bvtl	0	1	CPU interlock instruction
lv.s	3	1	Pipeline instruction
lv.q	3	1	Pipeline instruction
mfv	0	6	CPU interlock instruction
mfvc	0	6	CPU interlock instruction
mtv	3	1	Pipeline instruction
mtvc	3	1	Pipeline instruction
sv.s	0	7	CPU interlock instruction
sv.q	0	7	CPU interlock instruction
sv.q (non-cached and write buffer)	0	1	Pipeline instruction
svl.q	0	7	CPU interlock instruction
svr.q	0	7	CPU interlock instruction
vabs.s	3	1	Pipeline instruction
vabs.p	3	1	Pipeline instruction
vabs.t	3	1	Pipeline instruction
vabs.q	3	1	Pipeline instruction
vadd.s	5	1	Pipeline instruction
vadd.p	5	1	Pipeline instruction
vadd.t	5	1	Pipeline instruction
vadd.q	5	1	Pipeline instruction
vasin.s	7	1	Pipeline instruction
vasin.p	8	2	Repeat (pipeline) instruction
vasin.t	9	3	Repeat (pipeline) instruction
vasin.q	10	4	Repeat (pipeline) instruction
vavg.p	7	1	Pipeline instruction
vavg.t	7	1	Pipeline instruction
vavg.q	7	1	Pipeline instruction
vbfy1.p	5	1	Pipeline instruction
vbfy1.q	5	1	Pipeline instruction
vbfy2.q	5	1	Pipeline instruction
vc2i.s	3	1	Pipeline instruction

Instruction	Latency	Pitch	Type
vcmovf.s	5	1	Pipeline instruction
vcmovf.p	5	1	Pipeline instruction
vcmovf.t	5	1	Pipeline instruction
vcmovf.q	5	1	Pipeline instruction
vcmovt.s	5	1	Pipeline instruction
vcmovt.p	5	1	Pipeline instruction
vcmovt.t	5	1	Pipeline instruction
vcmovt.q	5	1	Pipeline instruction
vcmp.s	3	1	Pipeline instruction
vcmp.p	3	1	Pipeline instruction
vcmp.t	3	1	Pipeline instruction
vcmp.q	3	1	Pipeline instruction
vcos.s	7	1	Pipeline instruction
vcos.p	8	2	Repeat (pipeline) instruction
vcos.t	9	3	Repeat (pipeline) instruction
vcos.q	10	4	Repeat (pipeline) instruction
vcrs.t	5	1	Pipeline instruction
vcrsp.t	9	3	Repeat (pipeline) instruction
vcst.s	3	1	Pipeline instruction
vcst.p	3	1	Pipeline instruction
vcst.t	3	1	Pipeline instruction
vcst.q	3	1	Pipeline instruction
vdet.p	7	1	Pipeline instruction
vdiv.s	17	14	Multi-cycle instruction
vdot.p	7	1	Pipeline instruction
vdot.t	7	1	Pipeline instruction
vdot.q	7	1	Pipeline instruction
vexp2.s	7	1	Pipeline instruction
vexp2.p	8	2	Repeat (pipeline) instruction
vexp2.t	9	3	Repeat (pipeline) instruction
vexp2.q	10	4	Repeat (pipeline) instruction
vf2h.p	5	1	Pipeline instruction
vf2h.q	5	1	Pipeline instruction
vf2id.s	5	1	Pipeline instruction
vf2id.p	5	1	Pipeline instruction
vf2id.t	5	1	Pipeline instruction
vf2id.q	5	1	Pipeline instruction
vf2in.s	5	1	Pipeline instruction
vf2in.p	5	1	Pipeline instruction
vf2in.t	5	1	Pipeline instruction
vf2in.q	5	1	Pipeline instruction
vf2iu.s	5	1	Pipeline instruction
vf2iu.p	5	1	Pipeline instruction
vf2iu.t	5	1	Pipeline instruction
vf2iu.q	5	1	Pipeline instruction
vf2iz.s	5	1	Pipeline instruction
vf2iz.p	5	1	Pipeline instruction
vf2iz.t	5	1	Pipeline instruction
vf2iz.q	5	1	Pipeline instruction
vfad.p	7	1	Pipeline instruction
vfad.t	7	1	Pipeline instruction
vfad.q	7	1	Pipeline instruction

Instruction	Latency	Pitch	Type
vfim.s	3	1	Pipeline instruction
vflush	0	4	Synchronization instruction
vh2f.s	5	1	Pipeline instruction
vh2f.p	5	1	Pipeline instruction
vhdp.p	7	1	Pipeline instruction
vhdp.t	7	1	Pipeline instruction
vhdp.q	7	1	Pipeline instruction
vhtfm2.p	8	2	Repeat (pipeline) instruction
vhtfm3.t	9	3	Repeat (pipeline) instruction
vhtfm4.q	10	4	Repeat (pipeline) instruction
vi2c.q	3	1	Pipeline instruction
vi2f.s	5	1	Pipeline instruction
vi2f.p	5	1	Pipeline instruction
vi2f.t	5	1	Pipeline instruction
vi2f.q	5	1	Pipeline instruction
vi2s.p	3	1	Pipeline instruction
vi2s.q	3	1	Pipeline instruction
vi2uc.q	3	1	Pipeline instruction
vi2us.p	3	1	Pipeline instruction
vi2us.q	3	1	Pipeline instruction
vidt.p	3	1	Pipeline instruction
vidt.q	3	1	Pipeline instruction
viim.s	3	1	Pipeline instruction
vlgb.s	5	1	Pipeline instruction
vlog2.s	7	1	Pipeline instruction
vlog2.p	8	2	Repeat (pipeline) instruction
vlog2.t	9	3	Repeat (pipeline) instruction
vlog2.q	10	4	Repeat (pipeline) instruction
vmax.s	3	1	Pipeline instruction
vmax.p	3	1	Pipeline instruction
vmax.t	3	1	Pipeline instruction
vmax.q	3	1	Pipeline instruction
vmfvc	3	1	Pipeline instruction
vmidt.p	4	2	Repeat (pipeline) instruction
vmidt.t	5	3	Repeat (pipeline) instruction
vmidt.q	6	4	Repeat (pipeline) instruction
vmin.s	3	1	Pipeline instruction
vmin.p	3	1	Pipeline instruction
vmin.t	3	1	Pipeline instruction
vmin.q	3	1	Pipeline instruction
vmmov.p	4	2	Repeat (pipeline) instruction
vmmov.t	5	3	Repeat (pipeline) instruction
vmmov.q	6	4	Repeat (pipeline) instruction
vmmul.p	10	4	Repeat (pipeline) instruction
vmmul.t	15	9	Repeat (pipeline) instruction
vmmul.q	22	16	Repeat (pipeline) instruction
vmone.p	4	2	Repeat (pipeline) instruction
vmone.t	5	3	Repeat (pipeline) instruction
vmone.q	6	4	Repeat (pipeline) instruction
vmov.s	3	1	Pipeline instruction
vmov.p	3	1	Pipeline instruction
vmov.t	3	1	Pipeline instruction

Instruction	Latency	Pitch	Type
vmov.q	3	1	Pipeline instruction
vmscl.p	8	2	Repeat (pipeline) instruction
vmscl.t	9	3	Repeat (pipeline) instruction
vmscl.q	10	4	Repeat (pipeline) instruction
vmtvc	3	1	Pipeline instruction
vmul.s	5	1	Pipeline instruction
vmul.p	5	1	Pipeline instruction
vmul.t	5	1	Pipeline instruction
vmul.q	5	1	Pipeline instruction
vmzero.p	4	2	Repeat (pipeline) instruction
vmzero.t	5	3	Repeat (pipeline) instruction
vmzero.q	6	4	Repeat (pipeline) instruction
vneg.s	3	1	Pipeline instruction
vneg.p	3	1	Pipeline instruction
vneg.t	3	1	Pipeline instruction
vneg.q	3	1	Pipeline instruction
vnop	0	1	Synchronization instruction
vnrcp.s	7	1	Pipeline instruction
vnrcp.p	8	2	Repeat (pipeline) instruction
vnrcp.t	9	3	Repeat (pipeline) instruction
vnrcp.q	10	4	Repeat (pipeline) instruction
vnsin.s	7	1	Pipeline instruction
vnsin.p	8	2	Repeat (pipeline) instruction
vnsin.t	9	3	Repeat (pipeline) instruction
vnsin.q	10	4	Repeat (pipeline) instruction
vocp.s	5	1	Pipeline instruction
vocp.p	5	1	Pipeline instruction
vocp.t	5	1	Pipeline instruction
vocp.q	5	1	Pipeline instruction
vone.s	3	1	Pipeline instruction
vone.p	3	1	Pipeline instruction
vone.t	3	1	Pipeline instruction
vone.q	3	1	Pipeline instruction
vpfxd	0	1	Prefix instruction
vpfxs	0	1	Prefix instruction
vpfxt	0	1	Prefix instruction
vqmul.q	10	4	Repeat (pipeline) instruction
vrcp.s	7	1	Pipeline instruction
vrcp.p	8	2	Repeat (pipeline) instruction
vrcp.t	9	3	Repeat (pipeline) instruction
vrcp.q	10	4	Repeat (pipeline) instruction
vrexp2.s	7	1	Pipeline instruction
vrexp2.p	8	2	Repeat (pipeline) instruction
vrexp2.t	9	3	Repeat (pipeline) instruction
vrexp2.q	10	4	Repeat (pipeline) instruction
vrndf1.s	5	3	Multi-cycle instruction

Instruction	Latency	Pitch	Type
vrndf1.p	8	6	Repeat (multi-cycle) instruction
vrndf1.t	11	9	Repeat (multi-cycle) instruction
vrndf1.q	14	12	Repeat (multi-cycle) instruction
vrndf2.s	5	3	Multi-cycle instruction
vrndf2.p	8	6	Repeat (multi-cycle) instruction
vrndf2.t	11	9	Repeat (multi-cycle) instruction
vrndf2.q	14	12	Repeat (multi-cycle) instruction
vrndi.s	5	3	Multi-cycle instruction
vrndi.p	8	6	Repeat (multi-cycle) instruction
vrndi.t	11	9	Repeat (multi-cycle) instruction
vrndi.q	14	12	Repeat (multi-cycle) instruction
vrnds.s	3	1	Pipeline instruction
vrot.p	8	2	Repeat (pipeline) instruction
vrot.t	8	2	Repeat (pipeline) instruction
vrot.q	8	2	Repeat (pipeline) instruction
vrsq.s	7	1	Pipeline instruction
vrsq.p	8	2	Repeat (pipeline) instruction
vrsq.t	9	3	Repeat (pipeline) instruction
vrsq.q	10	4	Repeat (pipeline) instruction
vs2i.s	3	1	Pipeline instruction
vs2i.p	3	1	Pipeline instruction
vsat0.s	3	1	Pipeline instruction
vsat0.p	3	1	Pipeline instruction
vsat0.t	3	1	Pipeline instruction
vsat0.q	3	1	Pipeline instruction
vsat1.s	3	1	Pipeline instruction
vsat1.p	3	1	Pipeline instruction
vsat1.t	3	1	Pipeline instruction
vsat1.q	3	1	Pipeline instruction
vsbn.s	5	1	Pipeline instruction
vsbz.s	5	1	Pipeline instruction
vscl.p	5	1	Pipeline instruction
vscl.t	5	1	Pipeline instruction
vscl.q	5	1	Pipeline instruction
vscmp.s	3	1	Pipeline instruction
vscmp.p	3	1	Pipeline instruction
vscmp.t	3	1	Pipeline instruction
vscmp.q	3	1	Pipeline instruction
vsge.s	3	1	Pipeline instruction
vsge.p	3	1	Pipeline instruction
vsge.t	3	1	Pipeline instruction
vsge.q	3	1	Pipeline instruction
vsgn.s	3	1	Pipeline instruction

Instruction	Latency	Pitch	Type
vsgn.p	3	1	Pipeline instruction
vsgn.t	3	1	Pipeline instruction
vsgn.q	3	1	Pipeline instruction
vsin.s	7	1	Pipeline instruction
vsin.p	8	2	Repeat (pipeline) instruction
vsin.t	9	3	Repeat (pipeline) instruction
vsin.q	10	4	Repeat (pipeline) instruction
vslt.s	3	1	Pipeline instruction
vslt.p	3	1	Pipeline instruction
vslt.t	3	1	Pipeline instruction
vslt.q	3	1	Pipeline instruction
vsocp.s	5	1	Pipeline instruction
vsocp.p	5	1	Pipeline instruction
vsqrt.s	7	1	Pipeline instruction
vsqrt.p	8	2	Repeat (pipeline) instruction
vsqrt.t	9	3	Repeat (pipeline) instruction
vsqrt.q	10	4	Repeat (pipeline) instruction
vsrt1.q	5	1	Pipeline instruction
vsrt2.q	5	1	Pipeline instruction
vsrt3.q	5	1	Pipeline instruction
vsrt4.q	5	1	Pipeline instruction
vsub.s	5	1	Pipeline instruction
vsub.p	5	1	Pipeline instruction
vsub.t	5	1	Pipeline instruction
vsub.q	5	1	Pipeline instruction
vsync	0	4	Synchronization instruction
vsync2	0	6	Synchronization instruction
vt4444.q	3	1	Pipeline instruction
vt5551.q	3	1	Pipeline instruction
vt5650.q	3	1	Pipeline instruction
vtfm2.p	8	2	Repeat (pipeline) instruction
vtfm3.t	9	3	Repeat (pipeline) instruction
vtfm4.q	10	4	Repeat (pipeline) instruction
vuc2ifs.s	3	1	Pipeline instruction
vus2i.s	3	1	Pipeline instruction
vus2i.p	3	1	Pipeline instruction
vwbn.s	5	1	Pipeline instruction
vzero.s	3	1	Pipeline instruction
vzero.p	3	1	Pipeline instruction
vzero.t	3	1	Pipeline instruction
vzero.q	3	1	Pipeline instruction

8.3. Encoding Pattern for Each Format

8.3.1. Operand→Encoding

The S***, C***, R***, M*** and E*** formats are encoded as shown below.

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
S000	0	-	-	-	-	-	-
S001	32	-	-	-	-	-	-
S002	64	-	-	-	-	-	-
S003	96	-	-	-	-	-	-
S010	1	-	-	-	-	-	-
S011	33	-	-	-	-	-	-
S012	65	-	-	-	-	-	-
S013	97	-	-	-	-	-	-
S020	2	-	-	-	-	-	-
S021	34	-	-	-	-	-	-
S022	66	-	-	-	-	-	-
S023	98	-	-	-	-	-	-
S030	3	-	-	-	-	-	-
S031	35	-	-	-	-	-	-
S032	67	-	-	-	-	-	-
S033	99	-	-	-	-	-	-
S100	4	-	-	-	-	-	-
S101	36	-	-	-	-	-	-
S102	68	-	-	-	-	-	-
S103	100	-	-	-	-	-	-
S110	5	-	-	-	-	-	-
S111	37	-	-	-	-	-	-
S112	69	-	-	-	-	-	-
S113	101	-	-	-	-	-	-
S120	6	-	-	-	-	-	-
S121	38	-	-	-	-	-	-
S122	70	-	-	-	-	-	-
S123	102	-	-	-	-	-	-
S130	7	-	-	-	-	-	-
S131	39	-	-	-	-	-	-
S132	71	-	-	-	-	-	-
S133	103	-	-	-	-	-	-
S200	8	-	-	-	-	-	-
S201	40	-	-	-	-	-	-
S202	72	-	-	-	-	-	-
S203	104	-	-	-	-	-	-
S210	9	-	-	-	-	-	-
S211	41	-	-	-	-	-	-
S212	73	-	-	-	-	-	-
S213	105	-	-	-	-	-	-
S220	10	-	-	-	-	-	-
S221	42	-	-	-	-	-	-

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
S222	74	-	-	-	-	-	-
S223	106	-	-	-	-	-	-
S230	11	-	-	-	-	-	-
S231	43	-	-	-	-	-	-
S232	75	-	-	-	-	-	-
S233	107	-	-	-	-	-	-
S300	12	-	-	-	-	-	-
S301	44	-	-	-	-	-	-
S302	76	-	-	-	-	-	-
S303	108	-	-	-	-	-	-
S310	13	-	-	-	-	-	-
S311	45	-	-	-	-	-	-
S312	77	-	-	-	-	-	-
S313	109	-	-	-	-	-	-
S320	14	-	-	-	-	-	-
S321	46	-	-	-	-	-	-
S322	78	-	-	-	-	-	-
S323	110	-	-	-	-	-	-
S330	15	-	-	-	-	-	-
S331	47	-	-	-	-	-	-
S332	79	-	-	-	-	-	-
S333	111	-	-	-	-	-	-
S400	16	-	-	-	-	-	-
S401	48	-	-	-	-	-	-
S402	80	-	-	-	-	-	-
S403	112	-	-	-	-	-	-
S410	17	-	-	-	-	-	-
S411	49	-	-	-	-	-	-
S412	81	-	-	-	-	-	-
S413	113	-	-	-	-	-	-
S420	18	-	-	-	-	-	-
S421	50	-	-	-	-	-	-
S422	82	-	-	-	-	-	-
S423	114	-	-	-	-	-	-
S430	19	-	-	-	-	-	-
S431	51	-	-	-	-	-	-
S432	83	-	-	-	-	-	-
S433	115	-	-	-	-	-	-
S500	20	-	-	-	-	-	-
S501	52	-	-	-	-	-	-
S502	84	-	-	-	-	-	-
S503	116	-	-	-	-	-	-
S510	21	-	-	-	-	-	-
S511	53	-	-	-	-	-	-
S512	85	-	-	-	-	-	-
S513	117	-	-	-	-	-	-
S520	22	-	-	-	-	-	-
S521	54	-	-	-	-	-	-
S522	86	-	-	-	-	-	-
S523	118	-	-	-	-	-	-
S530	23	-	-	-	-	-	-
S531	55	-	-	-	-	-	-
S532	87	-	-	-	-	-	-

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
S533	119	-	-	-	-	-	-
S600	24	-	-	-	-	-	-
S601	56	-	-	-	-	-	-
S602	88	-	-	-	-	-	-
S603	120	-	-	-	-	-	-
S610	25	-	-	-	-	-	-
S611	57	-	-	-	-	-	-
S612	89	-	-	-	-	-	-
S613	121	-	-	-	-	-	-
S620	26	-	-	-	-	-	-
S621	58	-	-	-	-	-	-
S622	90	-	-	-	-	-	-
S623	122	-	-	-	-	-	-
S630	27	-	-	-	-	-	-
S631	59	-	-	-	-	-	-
S632	91	-	-	-	-	-	-
S633	123	-	-	-	-	-	-
S700	28	-	-	-	-	-	-
S701	60	-	-	-	-	-	-
S702	92	-	-	-	-	-	-
S703	124	-	-	-	-	-	-
S710	29	-	-	-	-	-	-
S711	61	-	-	-	-	-	-
S712	93	-	-	-	-	-	-
S713	125	-	-	-	-	-	-
S720	30	-	-	-	-	-	-
S721	62	-	-	-	-	-	-
S722	94	-	-	-	-	-	-
S723	126	-	-	-	-	-	-
S730	31	-	-	-	-	-	-
S731	63	-	-	-	-	-	-
S732	95	-	-	-	-	-	-
S733	127	-	-	-	-	-	-
C000	-	0	0	0	-	-	-
C001	-	-	64	-	-	-	-
C002	-	64	-	64	-	-	-
C003	-	-	-	-	-	-	-
C010	-	1	1	1	-	-	-
C011	-	-	65	-	-	-	-
C012	-	65	-	65	-	-	-
C013	-	-	-	-	-	-	-
C020	-	2	2	2	-	-	-
C021	-	-	66	-	-	-	-
C022	-	66	-	66	-	-	-
C023	-	-	-	-	-	-	-
C030	-	3	3	3	-	-	-
C031	-	-	67	-	-	-	-
C032	-	67	-	67	-	-	-
C033	-	-	-	-	-	-	-
C100	-	4	4	4	-	-	-
C101	-	-	36	-	-	-	-
C102	-	68	-	68	-	-	-
C103	-	-	-	-	-	-	-

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
C110	-	5	5	5	-	-	-
C111	-	-	69	-	-	-	-
C112	-	69	-	69	-	-	-
C113	-	-	-	-	-	-	-
C120	-	6	6	6	-	-	-
C121	-	-	70	-	-	-	-
C122	-	70	-	70	-	-	-
C123	-	-	-	-	-	-	-
C130	-	7	7	7	-	-	-
C131	-	-	71	-	-	-	-
C132	-	71	-	71	-	-	-
C133	-	-	-	-	-	-	-
C200	-	8	8	8	-	-	-
C201	-	-	72	-	-	-	-
C202	-	72	-	72	-	-	-
C203	-	-	-	-	-	-	-
C210	-	9	9	9	-	-	-
C211	-	-	73	-	-	-	-
C212	-	73	-	73	-	-	-
C213	-	-	-	-	-	-	-
C220	-	10	10	10	-	-	-
C221	-	-	74	-	-	-	-
C222	-	74	-	74	-	-	-
C223	-	-	-	-	-	-	-
C230	-	11	11	11	-	-	-
C231	-	-	75	-	-	-	-
C232	-	75	-	75	-	-	-
C233	-	-	-	-	-	-	-
C300	-	12	12	12	-	-	-
C301	-	-	76	-	-	-	-
C302	-	76	-	76	-	-	-
C303	-	-	-	-	-	-	-
C310	-	13	13	13	-	-	-
C311	-	-	77	-	-	-	-
C312	-	77	-	77	-	-	-
C313	-	-	-	-	-	-	-
C320	-	14	14	14	-	-	-
C321	-	-	78	-	-	-	-
C322	-	78	-	78	-	-	-
C323	-	-	-	-	-	-	-
C330	-	15	15	15	-	-	-
C331	-	-	79	-	-	-	-
C332	-	79	-	79	-	-	-
C333	-	-	-	-	-	-	-
C400	-	16	16	16	-	-	-
C401	-	-	80	-	-	-	-
C402	-	80	-	80	-	-	-
C403	-	-	-	-	-	-	-
C410	-	17	17	17	-	-	-
C411	-	-	81	-	-	-	-
C412	-	81	-	81	-	-	-
C413	-	-	-	-	-	-	-
C420	-	18	18	18	-	-	-

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
C421	-	-	82	-	-	-	-
C422	-	82	-	82	-	-	-
C423	-	-	-	-	-	-	-
C430	-	19	19	19	-	-	-
C431	-	-	83	-	-	-	-
C432	-	83	-	83	-	-	-
C433	-	-	-	-	-	-	-
C500	-	20	20	20	-	-	-
C501	-	-	84	-	-	-	-
C502	-	84	-	84	-	-	-
C503	-	-	-	-	-	-	-
C510	-	21	21	21	-	-	-
C511	-	-	85	-	-	-	-
C512	-	85	-	85	-	-	-
C513	-	-	-	-	-	-	-
C520	-	22	22	22	-	-	-
C521	-	-	86	-	-	-	-
C522	-	86	-	86	-	-	-
C523	-	-	-	-	-	-	-
C530	-	23	23	23	-	-	-
C531	-	-	87	-	-	-	-
C532	-	87	-	87	-	-	-
C533	-	-	-	-	-	-	-
C600	-	24	24	24	-	-	-
C601	-	-	88	-	-	-	-
C602	-	88	-	88	-	-	-
C603	-	-	-	-	-	-	-
C610	-	25	25	25	-	-	-
C611	-	-	89	-	-	-	-
C612	-	89	-	89	-	-	-
C613	-	-	-	-	-	-	-
C620	-	26	26	26	-	-	-
C621	-	-	90	-	-	-	-
C622	-	90	-	90	-	-	-
C623	-	-	-	-	-	-	-
C630	-	27	27	27	-	-	-
C631	-	-	91	-	-	-	-
C632	-	91	-	91	-	-	-
C633	-	-	-	-	-	-	-
C700	-	28	28	28	-	-	-
C701	-	-	92	-	-	-	-
C702	-	92	-	92	-	-	-
C703	-	-	-	-	-	-	-
C710	-	29	29	29	-	-	-
C711	-	-	93	-	-	-	-
C712	-	93	-	93	-	-	-
C713	-	-	-	-	-	-	-
C720	-	30	30	30	-	-	-
C721	-	-	94	-	-	-	-
C722	-	94	-	94	-	-	-
C723	-	-	-	-	-	-	-
C730	-	31	31	31	-	-	-
C731	-	-	95	-	-	-	-

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
C732	-	95	-	95	-	-	-
C733	-	-	-	-	-	-	-
R000	-	32	32	32	-	-	-
R001	-	33	33	33	-	-	-
R002	-	34	34	34	-	-	-
R003	-	35	35	35	-	-	-
R010	-	-	96	-	-	-	-
R011	-	-	97	-	-	-	-
R012	-	-	98	-	-	-	-
R013	-	-	99	-	-	-	-
R020	-	96	-	96	-	-	-
R021	-	97	-	97	-	-	-
R022	-	98	-	98	-	-	-
R023	-	99	-	99	-	-	-
R030	-	-	-	-	-	-	-
R031	-	-	-	-	-	-	-
R032	-	-	-	-	-	-	-
R033	-	-	-	-	-	-	-
R100	-	36	36	36	-	-	-
R101	-	37	37	37	-	-	-
R102	-	38	37	38	-	-	-
R103	-	39	39	39	-	-	-
R110	-	-	100	-	-	-	-
R111	-	-	101	-	-	-	-
R112	-	-	102	-	-	-	-
R113	-	-	103	-	-	-	-
R120	-	100	-	100	-	-	-
R121	-	102	-	102	-	-	-
R122	-	102	-	102	-	-	-
R123	-	103	-	103	-	-	-
R130	-	-	-	-	-	-	-
R131	-	-	-	-	-	-	-
R132	-	-	-	-	-	-	-
R133	-	-	-	-	-	-	-
R200	-	40	40	40	-	-	-
R201	-	41	41	41	-	-	-
R202	-	42	42	42	-	-	-
R203	-	43	43	43	-	-	-
R210	-	-	104	-	-	-	-
R211	-	-	105	-	-	-	-
R212	-	-	106	-	-	-	-
R213	-	-	107	-	-	-	-
R220	-	104	-	104	-	-	-
R221	-	105	-	105	-	-	-
R222	-	106	-	106	-	-	-
R223	-	-	-	-	-	-	-
R230	-	-	-	-	-	-	-
R231	-	-	43	-	-	-	-
R232	-	-	-	-	-	-	-
R233	-	-	-	-	-	-	-
R300	-	44	44	44	-	-	-
R301	-	45	45	45	-	-	-
R302	-	46	46	46	-	-	-

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
R303	-	47	47	47	-	-	-
R310	-	-	108	-	-	-	-
R311	-	-	109	-	-	-	-
R312	-	-	110	-	-	-	-
R313	-	-	111	-	-	-	-
R320	-	-	108	-	-	-	-
R321	-	-	109	-	-	-	-
R322	-	-	110	-	-	-	-
R323	-	-	111	-	-	-	-
R330	-	-	-	-	-	-	-
R331	-	-	-	-	-	-	-
R332	-	-	-	-	-	-	-
R333	-	-	-	-	-	-	-
R400	-	48	48	48	-	-	-
R401	-	49	49	49	-	-	-
R402	-	50	50	50	-	-	-
R403	-	51	51	51	-	-	-
R410	-	-	112	-	-	-	-
R411	-	-	113	-	-	-	-
R412	-	-	114	-	-	-	-
R413	-	-	115	-	-	-	-
R420	-	112	-	112	-	-	-
R421	-	113	-	113	-	-	-
R422	-	114	-	114	-	-	-
R423	-	-	-	-	-	-	-
R430	-	-	-	-	-	-	-
R431	-	-	-	-	-	-	-
R432	-	-	-	-	-	-	-
R433	-	-	-	-	-	-	-
R500	-	52	52	52	-	-	-
R501	-	53	53	53	-	-	-
R502	-	54	54	54	-	-	-
R503	-	55	55	55	-	-	-
R510	-	-	116	-	-	-	-
R511	-	-	117	-	-	-	-
R512	-	-	118	-	-	-	-
R513	-	-	119	-	-	-	-
R520	-	116	-	116	-	-	-
R521	-	117	-	117	-	-	-
R522	-	118	-	118	-	-	-
R523	-	119	-	119	-	-	-
R530	-	-	-	-	-	-	-
R531	-	-	-	-	-	-	-
R532	-	-	-	-	-	-	-
R533	-	-	-	-	-	-	-
R600	-	56	56	56	-	-	-
R601	-	57	57	57	-	-	-
R602	-	58	58	58	-	-	-
R603	-	59	59	59	-	-	-
R610	-	-	120	-	-	-	-
R611	-	-	121	-	-	-	-
R612	-	-	122	-	-	-	-
R613	-	-	123	-	-	-	-

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
R620	-	120	-	120	-	-	-
R621	-	121	-	121	-	-	-
R622	-	122	-	122	-	-	-
R623	-	123	-	123	-	-	-
R630	-	-	-	-	-	-	-
R631	-	-	-	-	-	-	-
R632	-	-	-	-	-	-	-
R633	-	-	-	-	-	-	-
R700	-	60	60	60	-	-	-
R701	-	61	61	61	-	-	-
R702	-	62	62	62	-	-	-
R703	-	63	63	63	-	-	-
R710	-	-	124	-	-	-	-
R711	-	-	125	-	-	-	-
R712	-	-	126	-	-	-	-
R713	-	-	127	-	-	-	-
R720	-	124	-	124	-	-	-
R721	-	125	-	125	-	-	-
R722	-	126	-	126	-	-	-
R723	-	127	-	127	-	-	-
R730	-	63	63	63	-	-	-
R731	-	-	-	-	-	-	-
R732	-	-	-	-	-	-	-
R733	-	-	-	-	-	-	-
M000	-	-	-	-	0	0	0
M001	-	-	-	-	-	64	-
M002	-	-	-	-	64	-	-
M003	-	-	-	-	-	-	-
M010	-	-	-	-	-	1	-
M011	-	-	-	-	-	65	-
M012	-	-	-	-	-	-	-
M013	-	-	-	-	-	-	-
M020	-	-	-	-	2	-	-
M021	-	-	-	-	-	-	-
M022	-	-	-	-	66	-	-
M023	-	-	-	-	-	-	-
M030	-	-	-	-	-	-	-
M031	-	-	-	-	-	-	-
M032	-	-	-	-	-	-	-
M033	-	-	-	-	-	-	-
M100	-	-	-	-	4	4	4
M101	-	-	-	-	-	68	-
M102	-	-	-	-	68	-	-
M103	-	-	-	-	-	-	-
M110	-	-	-	-	-	5	-
M111	-	-	-	-	-	69	-
M112	-	-	-	-	-	-	-
M113	-	-	-	-	-	-	-
M120	-	-	-	-	6	-	-
M121	-	-	-	-	-	-	-
M122	-	-	-	-	70	-	-
M123	-	-	-	-	-	-	-
M130	-	-	-	-	-	-	-

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
M131	-	-	-	-	-	-	-
M132	-	-	-	-	-	-	-
M133	-	-	-	-	-	-	-
M200	-	-	-	-	8	8	8
M201	-	-	-	-	-	72	-
M202	-	-	-	-	72	-	-
M203	-	-	-	-	-	-	-
M210	-	-	-	-	-	9	-
M211	-	-	-	-	-	73	-
M212	-	-	-	-	-	-	-
M213	-	-	-	-	-	-	-
M220	-	-	-	-	10	-	-
M221	-	-	-	-	-	-	-
M222	-	-	-	-	74	-	-
M223	-	-	-	-	-	-	-
M230	-	-	-	-	-	-	-
M231	-	-	-	-	-	-	-
M232	-	-	-	-	-	-	-
M233	-	-	-	-	-	-	-
M300	-	-	-	-	12	12	12
M301	-	-	-	-	-	76	-
M302	-	-	-	-	76	-	-
M303	-	-	-	-	-	-	-
M310	-	-	-	-	-	13	-
M311	-	-	-	-	-	77	-
M312	-	-	-	-	-	-	-
M313	-	-	-	-	-	-	-
M320	-	-	-	-	14	-	-
M321	-	-	-	-	-	-	-
M322	-	-	-	-	78	-	-
M323	-	-	-	-	-	-	-
M330	-	-	-	-	-	-	-
M331	-	-	-	-	-	-	-
M332	-	-	-	-	-	-	-
M333	-	-	-	-	-	-	-
M400	-	-	-	-	16	16	16
M401	-	-	-	-	-	80	-
M402	-	-	-	-	80	-	-
M403	-	-	-	-	-	-	-
M410	-	-	-	-	-	17	-
M411	-	-	-	-	-	81	-
M412	-	-	-	-	-	-	-
M413	-	-	-	-	-	-	-
M420	-	-	-	-	18	-	-
M421	-	-	-	-	-	-	-
M422	-	-	-	-	82	-	-
M423	-	-	-	-	-	-	-
M430	-	-	-	-	-	-	-
M431	-	-	-	-	-	-	-
M432	-	-	-	-	-	-	-
M433	-	-	-	-	-	-	-
M500	-	-	-	-	20	20	20
M501	-	-	-	-	-	84	-

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
M502	-	-	-	-	84	-	-
M503	-	-	-	-	-	-	-
M510	-	-	-	-	-	21	-
M511	-	-	-	-	-	85	-
M512	-	-	-	-	-	-	-
M513	-	-	-	-	-	-	-
M520	-	-	-	-	22	-	-
M521	-	-	-	-	-	-	-
M522	-	-	-	-	86	-	-
M523	-	-	-	-	-	-	-
M530	-	-	-	-	-	-	-
M531	-	-	-	-	-	-	-
M532	-	-	-	-	-	-	-
M533	-	-	-	-	-	-	-
M600	-	-	-	-	24	24	24
M601	-	-	-	-	-	88	-
M602	-	-	-	-	88	-	-
M603	-	-	-	-	-	-	-
M610	-	-	-	-	-	25	-
M611	-	-	-	-	-	89	-
M612	-	-	-	-	-	-	-
M613	-	-	-	-	-	-	-
M620	-	-	-	-	26	-	-
M621	-	-	-	-	-	-	-
M622	-	-	-	-	90	-	-
M623	-	-	-	-	-	-	-
M630	-	-	-	-	-	-	-
M631	-	-	-	-	-	-	-
M632	-	-	-	-	-	-	-
M633	-	-	-	-	-	-	-
M700	-	-	-	-	28	28	28
M701	-	-	-	-	-	92	-
M702	-	-	-	-	92	-	-
M703	-	-	-	-	-	-	-
M710	-	-	-	-	-	29	-
M711	-	-	-	-	-	93	-
M712	-	-	-	-	-	-	-
M713	-	-	-	-	-	-	-
M720	-	-	-	-	30	-	-
M721	-	-	-	-	-	-	-
M722	-	-	-	-	94	-	-
M723	-	-	-	-	-	-	-
M730	-	-	-	-	-	-	-
M731	-	-	-	-	-	-	-
M732	-	-	-	-	-	-	-
M733	-	-	-	-	-	-	-
E000	-	-	-	-	32	32	32
E001	-	-	-	-	-	33	-
E002	-	-	-	-	34	-	-
E003	-	-	-	-	-	-	-
E010	-	-	-	-	-	96	-
E011	-	-	-	-	-	97	-
E012	-	-	-	-	-	-	-

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E013	-	-	-	-	-	-	-
E020	-	-	-	-	96	-	-
E021	-	-	-	-	-	-	-
E022	-	-	-	-	98	-	-
E023	-	-	-	-	-	-	-
E030	-	-	-	-	-	-	-
E031	-	-	-	-	-	-	-
E032	-	-	-	-	-	-	-
E033	-	-	-	-	-	-	-
E100	-	-	-	-	36	36	36
E101	-	-	-	-	-	37	-
E102	-	-	-	-	38	-	-
E103	-	-	-	-	-	-	-
E110	-	-	-	-	-	100	-
E111	-	-	-	-	-	101	-
E112	-	-	-	-	-	-	-
E113	-	-	-	-	-	-	-
E120	-	-	-	-	100	-	-
E121	-	-	-	-	-	38	-
E122	-	-	-	-	102	-	-
E123	-	-	-	-	-	-	-
E130	-	-	-	-	-	-	-
E131	-	-	-	-	-	-	-
E132	-	-	-	-	-	-	-
E133	-	-	-	-	-	-	-
E200	-	-	-	-	40	40	40
E201	-	-	-	-	-	41	-
E202	-	-	-	-	42	-	-
E203	-	-	-	-	-	-	-
E210	-	-	-	-	-	104	-
E211	-	-	-	-	-	105	-
E212	-	-	-	-	-	-	-
E213	-	-	-	-	-	-	-
E220	-	-	-	-	104	-	-
E221	-	-	-	-	-	-	-
E222	-	-	-	-	106	-	-
E223	-	-	-	-	-	-	-
E230	-	-	-	-	-	-	-
E231	-	-	-	-	-	-	-
E232	-	-	-	-	-	-	-
E233	-	-	-	-	-	-	-
E300	-	-	-	-	44	44	44
E301	-	-	-	-	-	45	-
E302	-	-	-	-	46	-	-
E303	-	-	-	-	-	-	-
E310	-	-	-	-	-	108	-
E311	-	-	-	-	-	109	-
E312	-	-	-	-	-	-	-
E313	-	-	-	-	-	-	-
E320	-	-	-	-	108	-	-
E321	-	-	-	-	-	-	-
E322	-	-	-	-	110	-	-
E323	-	-	-	-	-	-	-

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E330	-	-	-	-	-	-	-
E331	-	-	-	-	-	-	-
E332	-	-	-	-	-	-	-
E333	-	-	-	-	-	-	-
E400	-	-	-	-	48	48	48
E401	-	-	-	-	-	49	-
E402	-	-	-	-	50	-	-
E403	-	-	-	-	-	-	-
E410	-	-	-	-	-	112	-
E411	-	-	-	-	-	113	-
E412	-	-	-	-	-	-	-
E413	-	-	-	-	-	-	-
E420	-	-	-	-	112	-	-
E421	-	-	-	-	-	-	-
E422	-	-	-	-	114	-	-
E423	-	-	-	-	-	-	-
E430	-	-	-	-	-	-	-
E431	-	-	-	-	-	-	-
E432	-	-	-	-	-	-	-
E433	-	-	-	-	-	-	-
E500	-	-	-	-	52	52	52
E501	-	-	-	-	-	53	-
E502	-	-	-	-	54	-	-
E503	-	-	-	-	-	-	-
E510	-	-	-	-	-	116	-
E511	-	-	-	-	-	117	-
E512	-	-	-	-	-	-	-
E513	-	-	-	-	-	-	-
E520	-	-	-	-	116	-	-
E521	-	-	-	-	-	-	-
E522	-	-	-	-	118	-	-
E523	-	-	-	-	-	-	-
E530	-	-	-	-	-	-	-
E531	-	-	-	-	-	-	-
E532	-	-	-	-	-	-	-
E533	-	-	-	-	-	-	-
E600	-	-	-	-	56	56	56
E601	-	-	-	-	-	57	-
E602	-	-	-	-	58	-	-
E603	-	-	-	-	-	-	-
E610	-	-	-	-	-	120	-
E611	-	-	-	-	-	121	-
E612	-	-	-	-	-	-	-
E613	-	-	-	-	-	-	-
E620	-	-	-	-	120	-	-
E621	-	-	-	-	-	-	-
E622	-	-	-	-	122	-	-
E623	-	-	-	-	-	-	-
E630	-	-	-	-	-	-	-
E631	-	-	-	-	-	-	-
E632	-	-	-	-	-	-	-
E633	-	-	-	-	-	-	-
E700	-	-	-	-	60	60	60

Operand	Scalar	2D vector	3D vector	4D vector	2 x 2 matrix	3 x 3 matrix	4 x 4 matrix
E701	-	-	-	-	-	61	-
E702	-	-	-	-	62	-	-
E703	-	-	-	-	-	-	-
E710	-	-	-	-	-	124	-
E711	-	-	-	-	-	125	-
E712	-	-	-	-	-	-	-
E713	-	-	-	-	-	-	-
E720	-	-	-	-	124	-	-
E721	-	-	-	-	-	62	-
E722	-	-	-	-	126	-	-
E723	-	-	-	-	-	-	-
E730	-	-	-	-	-	-	-
E731	-	-	-	-	-	-	-
E732	-	-	-	-	-	-	-
E733	-	-	-	-	-	-	-

8.3.2. Encoding→Operand

The operands for the S***,C***,R***,M***,E*** formats will be decoded as shown below.

Operand	Scalar	2D vector	3D vector	4D vector	2x2 matrix	3x3 matrix	4x4 matrix
0	S000	C000	C000	C000	M000	M000	M000
1	S010	C010	C010	C010		M010	
2	S020	C020	C020	C020	M020		
3	S030	C030	C030	C030			
4	S100	C100	C100	C100	M100	M100	M100
5	S110	C110	C110	C110		M110	
6	S120	C120	C120	C120	M120		
7	S130	C130	C130	C130			
8	S200	C200	C200	C200	M200	M200	M200
9	S210	C210	C210	C210		M210	
10	S220	C220	C220	C220	M220		
11	S230	C230	C230	C230			
12	S300	C300	C300	C300	M300	M300	M300
13	S310	C310	C310	C310		M310	
14	S320	C320	C320	C320	M320		
15	S330	C330	C330	C330			
16	S400	C400	C400	C400	M400	M400	M400
17	S410	C410	C410	C410		M410	
18	S420	C420	C420	C420	M420		
19	S430	C430	C430	C430			
20	S500	C500	C500	C500	M500	M500	M500
21	S510	C510	C510	C510		M510	
22	S520	C520	C520	C520	M520		
23	S530	C530	C530	C530			
24	S600	C600	C600	C600	M600	M600	M600
25	S610	C610	C610	C610		M610	
26	S620	C620	C620	C620	M620		
27	S630	C630	C630	C630			
28	S700	C700	C700	C700	M700	M700	M700
29	S710	C710	C710	C710		M710	
30	S720	C720	C720	C720	M720		
31	S730	C730	C730	C730			

Operand	Scalar	2D vector	3D vector	4D vector	2x2 matrix	3x3 matrix	4x4 matrix
32	S001	R000	R000	R000	E000	E000	E000
33	S011	R001	R001	R001		E710	
34	S021	R002	R002	R002	E002		
35	S031	R003	R003	R003			
36	S101	R100	R100	R100	E100	E100	E100
37	S111	R101	R101	R101		E101	
38	S121	R102	R102	R102	E102		
39	S131	R103	R103	R103			
40	S201	R200	R200	R200	E200	E200	E200
41	S211	R201	R201	R201		E201	
42	S221	R202	R202	R202	E202		
43	S231	R203	R203	R203			
44	S301	R300	R300	R300	E300	E300	E300
45	S311	R301	R301	R301		E301	
46	S321	R302	R302	R302	E302		
47	S331	R303	R303	R303			
48	S401	R400	R400	R400	E400	E400	E400
49	S411	R401	R401	R401		E401	
50	S421	R402	R402	R402	E402		
51	S431	R403	R403	R403			
52	S501	R500	R500	R500	E500	E500	E500
53	S511	R501	R501	R501		E501	
54	S521	R502	R502	R502	E502		
55	S531	R503	R503	R503			
56	S601	R600	R600	R600	E600	E600	E600
57	S611	R601	R601	R601		E601	
58	S621	R602	R602	R602	E602		
59	S631	R603	R603	R603			
60	S701	R700	R700	R700	E700	E700	E700
61	S711	R701	R701	R701		E701	
62	S721	R702	R702	R702	E702		
63	S731	R703	R703	R703			
64	S002	C002	C001	C002	M002	M001	
65	S012	C012	C011	C012		M011	
66	S022	C022	C021	C022	M022		
67	S032	C032	C031	C032			
68	S102	C102	C101	C102	M102	M101	
69	S112	C112	C111	C112		M111	
70	S122	C122	C121	C122	M122		
71	S132	C132	C131	C132			
72	S202	C202	C201	C202	M202	M201	
73	S212	C212	C211	C212		M211	
74	S222	C222	C221	C222	M222		
75	S232	C232	C231	C232			
76	S302	C302	C301	C302	M302	M301	
77	S312	C312	C311	C312		M311	
78	S322	C322	C321	C322	M322		
79	S332	C332	C331	C332			
80	S402	C402	C401	C402	M402	M401	
81	S412	C412	C411	C412		M411	
82	S422	C422	C421	C422	M422		
83	S432	C432	C431	C432			
84	S502	C502	C501	C502	M502	M501	

Operand	Scalar	2D vector	3D vector	4D vector	2x2 matrix	3x3 matrix	4x4 matrix
85	S512	C512	C511	C512		M511	
86	S522	C522	C521	C522	M522		
87	S532	C532	C531	C532			
88	S602	C602	C601	C602	M602	M601	
89	S612	C612	C611	C612		M611	
90	S622	C622	C621	C622	M622		
91	S632	C632	C631	C632			
92	S702	C702	C701	C702	M702	M701	
93	S712	C712	C711	C712		M711	
94	S722	C722	C721	C722	M722		
95	S732	C732	C731	C732			
96	S003	R020	R010	R020	E020	E010	
97	S013	R021	R011	R021		E011	
98	S023	R022	R012	R022	E022		
99	S033	R023	R013	R023			
100	S103	R120	R110	R120	E120	E110	
101	S113	R121	R111	R121		E111	
102	S123	R122	R112	R122	E122		
103	S133	R123	R113	R123			
104	S203	R220	R210	R220	E220	E210	
105	S213	R221	R211	R221		E211	
106	S223	R222	R212	R222	E222		
107	S233	R223	R213	R223			
108	S303	R320	R310	R320	E320	E310	
109	S313	R321	R311	R321		E311	
110	S323	R322	R312	R322	E322		
111	S333	R333	R313	R333			
112	S403	R420	R410	R420	E420	E410	
113	S413	R421	R411	R421		E411	
114	S423	R422	R412	R422	E422		
115	S433	R423	R413	R423			
116	S503	R520	R510	R520	E520	E510	
117	S513	R521	R511	R521		E511	
118	S523	R522	R512	R522	E522		
119	S533	R523	R513	R523			
120	S603	R620	R610	R620	E620	E610	
121	S613	R621	R611	R621		E611	
122	S623	R622	R612	R622	E622		
123	S633	R623	R613	R623			
124	S703	R720	R710	R720	E720	E710	
125	S713	R721	R711	R721		E711	
126	S723	R722	R712	R722	E722		
127	S733	R723	R713	R723			

8.4. Comparison Instructions

8.4.1. Condition Format

The low-order 4 bits of the condition format (cond) of a VFPU instruction are decoded as shown below.

Code	Mnemonic	Function
0	FL	Always false
1	EQ	Equal
2	LT	Less than
3	LE	Less than or equal to
4	TR	Always true
5	NE	Not equal
6	GE	Greater than or equal to
7	GT	Greater than
8	EZ	Equal to 0
9	EN	Equal to NaN
10	EI	Absolute value equal to infinity
11	ES	Infinity or NaN
12	NZ	Not equal to 0
13	NN	Not equal to NaN
14	NI	Absolute value not equal to infinity
15	NS	Not infinity and not NaN

8.4.2. Comparison Operation

For each of the above condition codes, the following truth tables show comparison results for typical values.

FL(Always false)

src＼tar	-NaN	-Inf	-1.0	-0.0	+0.0	+1.0	+Inf	+NaN
-NaN	0	0	0	0	0	0	0	0
-Inf	0	0	0	0	0	0	0	0
-1.0	0	0	0	0	0	0	0	0
-0.0	0	0	0	0	0	0	0	0
+0.0	0	0	0	0	0	0	0	0
+1.0	0	0	0	0	0	0	0	0
+Inf	0	0	0	0	0	0	0	0
+NaN	0	0	0	0	0	0	0	0

EQ(Equal)

src＼tar	-NaN	-Inf	-1.0	-0.0	+0.0	+1.0	+Inf	+NaN
-NaN	0	0	0	0	0	0	0	0
-Inf	0	1	0	0	0	0	0	0
-1.0	0	0	1	0	0	0	0	0
-0.0	0	0	0	1	1	0	0	0
+0.0	0	0	0	1	1	0	0	0
+1.0	0	0	0	0	0	1	0	0
+Inf	0	0	0	0	0	0	1	0
+NaN	0	0	0	0	0	0	0	0

LT(Less than)

src＼tar	-NaN	-Inf	-1.0	-0.0	+0.0	+1.0	+Inf	+NaN
-NaN	0	0	0	0	0	0	0	0
-Inf	0	0	1	1	1	1	1	0
-1.0	0	0	0	1	1	1	1	0
-0.0	0	0	0	0	0	1	1	0
+0.0	0	0	0	0	0	1	1	0
+1.0	0	0	0	0	0	0	1	0
+Inf	0	0	0	0	0	0	0	0
+NaN	0	0	0	0	0	0	0	0

LE(Less than or equal to)

src＼tar	-NaN	-Inf	-1.0	-0.0	+0.0	+1.0	+Inf	+NaN
-NaN	0	0	0	0	0	0	0	0
-Inf	0	1	1	1	1	1	1	0
-1.0	0	0	1	1	1	1	1	0
-0.0	0	0	0	1	1	1	1	0
+0.0	0	0	0	1	1	1	1	0
+1.0	0	0	0	0	0	1	1	0
+Inf	0	0	0	0	0	0	1	0
+NaN	0	0	0	0	0	0	0	0

TR(Always true)

src＼tar	-NaN	-Inf	-1.0	-0.0	+0.0	+1.0	+Inf	+NaN
-NaN	1	1	1	1	1	1	1	1
-Inf	1	1	1	1	1	1	1	1
-1.0	1	1	1	1	1	1	1	1
-0.0	1	1	1	1	1	1	1	1
+0.0	1	1	1	1	1	1	1	1
+1.0	1	1	1	1	1	1	1	1
+Inf	1	1	1	1	1	1	1	1
+NaN	1	1	1	1	1	1	1	1

NE(Not equal)

src＼tar	-NaN	-Inf	-1.0	-0.0	+0.0	+1.0	+Inf	+NaN
-NaN	1	1	1	1	1	1	1	1
-Inf	1	0	1	1	1	1	1	1
-1.0	1	1	0	1	1	1	1	1
-0.0	1	1	1	0	0	1	1	1
+0.0	1	1	1	0	0	1	1	1
+1.0	1	1	1	1	1	0	1	1
+Inf	1	1	1	1	1	1	0	1
+NaN	1	1	1	1	1	1	1	1

GE(Greater than or equal to)

src＼tar	-NaN	-Inf	-1.0	-0.0	+0.0	+1.0	+Inf	+NaN
-NaN	0	0	0	0	0	0	0	0
-Inf	0	1	0	0	0	0	0	0
-1.0	0	1	1	0	0	0	0	0
-0.0	0	1	1	1	1	0	0	0
+0.0	0	1	1	1	1	0	0	0
+1.0	0	1	1	1	1	1	0	0
+Inf	0	1	1	1	1	1	1	0
+NaN	0	0	0	0	0	0	0	0

GT(Greater than)

src＼tar	-NaN	-Inf	-1.0	-0.0	+0.0	+1.0	+Inf	+NaN
-NaN	0	0	0	0	0	0	0	0
-Inf	0	0	0	0	0	0	0	0
-1.0	0	1	0	0	0	0	0	0
-0.0	0	1	1	0	0	0	0	0
+0.0	0	1	1	0	0	0	0	0
+1.0	0	1	1	1	1	0	0	0
+Inf	0	1	1	1	1	1	0	0
+NaN	0	0	0	0	0	0	0	0

EZ(Equal to 0)

src	
-NaN	0
-Inf	0
-1.0	0
-0.0	1
+0.0	1
+1.0	0
+Inf	0
+NaN	0

EN(Equal to NaN)

src	
-NaN	1
-Inf	0
-1.0	0
-0.0	0
+0.0	0
+1.0	0
+Inf	0
+NaN	1

EI(Absolute value equal to infinity)

src	
-NaN	0
-Inf	1
-1.0	0
-0.0	0
+0.0	0
+1.0	0
+Inf	1
+NaN	0

ES(Infinity or NaN)

src	
-NaN	1
-Inf	1
-1.0	0
-0.0	0
+0.0	0
+1.0	0
+Inf	1
+NaN	1

NZ(Not equal to 0)

src	
-NaN	1
-Inf	1
-1.0	1
-0.0	0
+0.0	0
+1.0	1
+Inf	1
+NaN	1

NN(Not equal to NaN)

src	
-NaN	0
-Inf	1
-1.0	1
-0.0	1
+0.0	1
+1.0	1
+Inf	1
+NaN	0

NI(Absolute value not equal to infinity)

src	
-NaN	1
-Inf	0
-1.0	1
-0.0	1
+0.0	1
+1.0	1
+Inf	0
+NaN	1

NS(Not infinity or not NaN)

src	
-NaN	0
-Inf	0
-1.0	1
-0.0	1
+0.0	1
+1.0	1
+Inf	0
+NaN	0

8.5. Validity of Prefixing for Each Instruction

Prefixing is only valid for a subset of VFPU instructions. The following table shows the effect of prefixing on each VFPU instruction. Please do not use prefixing with instructions marked "Use prohibited" as the results are undefined.

Instruction	vpfxs	vpfxt	vpfxd
bvf	No effect	No effect	No effect
bvfl	No effect	No effect	No effect
bvt	No effect	No effect	No effect
bvtl	No effect	No effect	No effect
lv.s	No effect	No effect	No effect
lv.q	No effect	No effect	No effect
mfv	No effect	No effect	No effect
mfvc	No effect	No effect	No effect
mtv	No effect	No effect	No effect
mtvc	No effect	No effect	No effect
sv.s	No effect	No effect	No effect
sv.q	No effect	No effect	No effect
sv.q (non-cached and write buffer)	No effect	No effect	No effect
svl.q	No effect	No effect	No effect
svr.q	No effect	No effect	No effect
vabs.s	Only swizzle is valid	No effect	Valid
vabs.p	Only swizzle is valid	No effect	Valid
vabs.t	Only swizzle is valid	No effect	Valid
vabs.q	Only swizzle is valid	No effect	Valid
vadd.s	Valid	Valid	Valid
vadd.p	Valid	Valid	Valid
vadd.t	Valid	Valid	Valid
vadd.q	Valid	Valid	Valid
vasin.s	Valid	No effect	Valid
vasin.p	Use prohibited	No effect	Use prohibited
vasin.t	Use prohibited	No effect	Use prohibited
vasin.q	Use prohibited	No effect	Use prohibited
vavg.p	Valid	Use prohibited	Valid
vavg.t	Valid	Use prohibited	Valid
vavg.q	Valid	Use prohibited	Valid
vbfy1.p	Use prohibited	Use prohibited	Valid
vbfy1.q	Use prohibited	Use prohibited	Valid
vbfy2.q	Use prohibited	Use prohibited	Valid
vc2i.s	Use prohibited	No effect	Only write mask is valid
vcmovf.s	Valid	Use prohibited	Use prohibited
vcmovf.p	Valid	Use prohibited	Use prohibited
vcmovf.t	Valid	Use prohibited	Use prohibited
vcmovf.q	Valid	Use prohibited	Use prohibited
vcmovt.s	Valid	Use prohibited	Use prohibited
vcmovt.p	Valid	Use prohibited	Use prohibited
vcmovt.t	Valid	Use prohibited	Use prohibited
vcmovt.q	Valid	Use prohibited	Use prohibited

Instruction	vpfxs	vpfxt	vpfxd
vcmp.s	Valid	Valid	No effect
vcmp.p	Valid	Valid	No effect
vcmp.t	Valid	Valid	No effect
vcmp.q	Valid	Valid	No effect
vcos.s	Valid	No effect	Valid
vcos.p	Use prohibited	No effect	Use prohibited
vcos.t	Use prohibited	No effect	Use prohibited
vcos.q	Use prohibited	No effect	Use prohibited
vcrs.t	Use prohibited	Use prohibited	Valid
vcrsp.t	Use prohibited	Use prohibited	Use prohibited
vcst.s	No effect	No effect	Valid
vcst.p	No effect	No effect	Valid
vcst.t	No effect	No effect	Valid
vcst.q	No effect	No effect	Valid
vdet.p	Valid	Use prohibited	Valid
vdiv.s	Valid	Valid	Valid
vdot.p	Valid	Valid	Valid
vdot.t	Valid	Valid	Valid
vdot.q	Valid	Valid	Valid
vexp2.s	Valid	No effect	Valid
vexp2.p	Use prohibited	No effect	Use prohibited
vexp2.t	Use prohibited	No effect	Use prohibited
vexp2.q	Use prohibited	No effect	Use prohibited
vf2h.p	Valid	No effect	Only write mask is valid
vf2h.q	Valid	No effect	Only write mask is valid
vf2id.s	Valid	No effect	Only write mask is valid
vf2id.p	Valid	No effect	Only write mask is valid
vf2id.t	Valid	No effect	Only write mask is valid
vf2id.q	Valid	No effect	Only write mask is valid
vf2in.s	Valid	No effect	Only write mask is valid
vf2in.p	Valid	No effect	Only write mask is valid
vf2in.t	Valid	No effect	Only write mask is valid
vf2in.q	Valid	No effect	Only write mask is valid
vf2iu.s	Valid	No effect	Only write mask is valid
vf2iu.p	Valid	No effect	Only write mask is valid
vf2iu.t	Valid	No effect	Only write mask is valid
vf2iu.q	Valid	No effect	Only write mask is valid
vf2iz.s	Valid	No effect	Only write mask is valid
vf2iz.p	Valid	No effect	Only write mask is valid
vf2iz.t	Valid	No effect	Only write mask is valid
vf2iz.q	Valid	No effect	Only write mask is valid
vfad.p	Valid	Use prohibited	Valid
vfad.t	Valid	Use prohibited	Valid
vfad.q	Valid	Use prohibited	Valid
vfim.s	No effect	No effect	Valid
vflush	No effect	No effect	No effect
vh2f.s	Use prohibited	No effect	Valid
vh2f.p	Use prohibited	No effect	Valid
vhdp.p	Use prohibited	Valid	Valid
vhdp.t	Use prohibited	Valid	Valid
vhdp.q	Use prohibited	Valid	Valid
vhtfm2.p	Use prohibited	Use prohibited	Use prohibited

Instruction	vpfxs	vpfxt	vpfxd
vhtfm3.t	Use prohibited	Use prohibited	Use prohibited
vhtfm4.q	Use prohibited	Use prohibited	Use prohibited
vi2c.q	Only swizzle is valid	No effect	Only write mask is valid
vi2f.s	Only swizzle is valid	No effect	Valid
vi2f.p	Only swizzle is valid	No effect	Valid
vi2f.t	Only swizzle is valid	No effect	Valid
vi2f.q	Only swizzle is valid	No effect	Valid
vi2s.p	Only swizzle is valid	No effect	Only write mask is valid
vi2s.q	Only swizzle is valid	No effect	Only write mask is valid
vi2uc.q	Only swizzle is valid	No effect	Only write mask is valid
vi2us.p	Only swizzle is valid	No effect	Only write mask is valid
vi2us.q	Only swizzle is valid	No effect	Only write mask is valid
vidt.p	Use prohibited	No effect	Valid
vidt.q	Use prohibited	No effect	Valid
viim.s	No effect	No effect	Valid
vlgb.s	Valid	No effect	Valid
vlog2.s	Valid	No effect	Valid
vlog2.p	Use prohibited	No effect	Use prohibited
vlog2.t	Use prohibited	No effect	Use prohibited
vlog2.q	Use prohibited	No effect	Use prohibited
vmax.s	Valid	Valid	Valid
vmax.p	Valid	Valid	Valid
vmax.t	Valid	Valid	Valid
vmax.q	Valid	Valid	Valid
vmfvc	No effect	No effect	No effect
vmidt.p	Use prohibited	No effect	Use prohibited
vmidt.t	Use prohibited	No effect	Use prohibited
vmidt.q	Use prohibited	No effect	Use prohibited
vmin.s	Valid	Valid	Valid
vmin.p	Valid	Valid	Valid
vmin.t	Valid	Valid	Valid
vmin.q	Valid	Valid	Valid
vmmov.p	Use prohibited	No effect	Use prohibited
vmmov.t	Use prohibited	No effect	Use prohibited
vmmov.q	Use prohibited	No effect	Use prohibited
vmmul.p	Use prohibited	Use prohibited	Use prohibited
vmmul.t	Use prohibited	Use prohibited	Use prohibited
vmmul.q	Use prohibited	Use prohibited	Use prohibited
vmone.p	Use prohibited	No effect	Use prohibited
vmone.t	Use prohibited	No effect	Use prohibited
vmone.q	Use prohibited	No effect	Use prohibited
vmov.s	Valid	No effect	Valid
vmov.p	Valid	No effect	Valid
vmov.t	Valid	No effect	Valid
vmov.q	Valid	No effect	Valid
vmscl.p	Use prohibited	Use prohibited	Use prohibited
vmscl.t	Use prohibited	Use prohibited	Use prohibited
vmscl.q	Use prohibited	Use prohibited	Use prohibited
vmtvc	No effect	No effect	No effect
vmul.s	Valid	Valid	Valid
vmul.p	Valid	Valid	Valid
vmul.t	Valid	Valid	Valid

Instruction	vpfxs	vpfxt	vpfxd
vmul.q	Valid	Valid	Valid
vmzero.p	Use prohibited	No effect	Use prohibited
vmzero.t	Use prohibited	No effect	Use prohibited
vmzero.q	Use prohibited	No effect	Use prohibited
vneg.s	Only swizzle is valid	No effect	Valid
vneg.p	Only swizzle is valid	No effect	Valid
vneg.t	Only swizzle is valid	No effect	Valid
vneg.q	Only swizzle is valid	No effect	Valid
vnop	No effect	No effect	No effect
vnrcp.s	Use prohibited	No effect	Valid
vnrcp.p	Use prohibited	No effect	Use prohibited
vnrcp.t	Use prohibited	No effect	Use prohibited
vnrcp.q	Use prohibited	No effect	Use prohibited
vnsin.s	Use prohibited	No effect	Valid
vnsin.p	Use prohibited	No effect	Use prohibited
vnsin.t	Use prohibited	No effect	Use prohibited
vnsin.q	Use prohibited	No effect	Use prohibited
vocp.s	Use prohibited	Use prohibited	Valid
vocp.p	Use prohibited	Use prohibited	Valid
vocp.t	Use prohibited	Use prohibited	Valid
vocp.q	Use prohibited	Use prohibited	Valid
vone.s	Use prohibited	No effect	Valid
vone.p	Use prohibited	No effect	Valid
vone.t	Use prohibited	No effect	Valid
vone.q	Use prohibited	No effect	Valid
vpfxd	No effect	No effect	Overwrite
vpfxs	Overwrite	No effect	No effect
vpfxt	No effect	Overwrite	No effect
vqmul.q	Use prohibited	Use prohibited	Use prohibited
vrcp.s	Valid	No effect	Valid
vrcp.p	Use prohibited	No effect	Use prohibited
vrcp.t	Use prohibited	No effect	Use prohibited
vrcp.q	Use prohibited	No effect	Use prohibited
vrexp2.s	Use prohibited	No effect	Valid
vrexp2.p	Use prohibited	No effect	Use prohibited
vrexp2.t	Use prohibited	No effect	Use prohibited
vrexp2.q	Use prohibited	No effect	Use prohibited
vrndf1.s	No effect	No effect	Valid
vrndf1.p	No effect	No effect	Use prohibited
vrndf1.t	No effect	No effect	Use prohibited
vrndf1.q	No effect	No effect	Use prohibited
vrndf2.s	No effect	No effect	Valid
vrndf2.p	No effect	No effect	Use prohibited
vrndf2.t	No effect	No effect	Use prohibited
vrndf2.q	No effect	No effect	Use prohibited
vrndi.s	No effect	No effect	Valid
vrndi.p	No effect	No effect	Use prohibited
vrndi.t	No effect	No effect	Use prohibited
vrndi.q	No effect	No effect	Use prohibited
vrnds.s	No effect	No effect	No effect
vrot.p	Use prohibited	No effect	Use prohibited
vrot.t	Use prohibited	No effect	Use prohibited

Instruction	vpfxs	vpfxt	vpfxd
vrot.q	Use prohibited	No effect	Use prohibited
vrsq.s	Valid	No effect	Valid
vrsq.p	Use prohibited	No effect	Use prohibited
vrsq.t	Use prohibited	No effect	Use prohibited
vrsq.q	Use prohibited	No effect	Use prohibited
vs2i.s	Use prohibited	No effect	Only write mask is valid
vs2i.p	Use prohibited	No effect	Only write mask is valid
vsat0.s	Valid	No effect	Use prohibited
vsat0.p	Valid	No effect	Use prohibited
vsat0.t	Valid	No effect	Use prohibited
vsat0.q	Valid	No effect	Use prohibited
vsat1.s	Valid	No effect	Use prohibited
vsat1.p	Valid	No effect	Use prohibited
vsat1.t	Valid	No effect	Use prohibited
vsat1.q	Valid	No effect	Use prohibited
vsbn.s	Valid	Valid	Valid
vsbz.s	Valid	No effect	Valid
vscl.p	Valid	Use prohibited	Valid
vscl.t	Valid	Use prohibited	Valid
vscl.q	Valid	Use prohibited	Valid
vscmp.s	Valid	Valid	Valid
vscmp.p	Valid	Valid	Valid
vscmp.t	Valid	Valid	Valid
vscmp.q	Valid	Valid	Valid
vsge.s	Valid	Valid	Valid
vsge.p	Valid	Valid	Valid
vsge.t	Valid	Valid	Valid
vsge.q	Valid	Valid	Valid
vsgn.s	Valid	Use prohibited	Valid
vsgn.p	Valid	Use prohibited	Valid
vsgn.t	Valid	Use prohibited	Valid
vsgn.q	Valid	Use prohibited	Valid
vsin.s	Valid	No effect	Valid
vsin.p	Use prohibited	No effect	Use prohibited
vsin.t	Use prohibited	No effect	Use prohibited
vsin.q	Use prohibited	No effect	Use prohibited
vslt.s	Valid	Valid	Valid
vslt.p	Valid	Valid	Valid
vslt.t	Valid	Valid	Valid
vslt.q	Valid	Valid	Valid
vsocp.s	Use prohibited	Use prohibited	Use prohibited
vsocp.p	Use prohibited	Use prohibited	Use prohibited
vsqrt.s	Valid	No effect	Valid
vsqrt.p	Use prohibited	No effect	Use prohibited
vsqrt.t	Use prohibited	No effect	Use prohibited
vsqrt.q	Use prohibited	No effect	Use prohibited
vsrt1.q	Use prohibited	Use prohibited	Valid
vsrt2.q	Use prohibited	Use prohibited	Valid
vsrt3.q	Use prohibited	Use prohibited	Valid
vsrt4.q	Use prohibited	Use prohibited	Valid
vsub.s	Valid	Valid	Valid
vsub.p	Valid	Valid	Valid

Instruction	vpfxs	vpfxt	vpfxd
vsub.t	Valid	Valid	Valid
vsub.q	Valid	Valid	Valid
vsync	No effect	No effect	No effect
vsync2	No effect	No effect	No effect
vt4444.q	Only swizzle is valid	No effect	Use prohibited
vt5551.q	Only swizzle is valid	No effect	Use prohibited
vt5650.q	Only swizzle is valid	No effect	Use prohibited
vtfm2.p	Use prohibited	Use prohibited	Use prohibited
vtfm3.t	Use prohibited	Use prohibited	Use prohibited
vtfm4.q	Use prohibited	Use prohibited	Use prohibited
vuc2ifs.s	Use prohibited	No effect	Only write mask is valid
vus2i.s	Use prohibited	No effect	Only write mask is valid
vus2i.p	Use prohibited	No effect	Only write mask is valid
vwbn.s	Valid	No effect	Valid
vzero.s	Use prohibited	No effect	Valid
vzero.p	Use prohibited	No effect	Valid
vzero.t	Use prohibited	No effect	Valid
vzero.q	Use prohibited	No effect	Valid