There is no *Enable* bit for this condition; it always causes a trap. After the appropriate emulation or other operation is done in a software exception handler, the original instruction stream can be continued.

5.9 FPU Instructions

The FPU instructions comprise the following functional groups:

- "Data Transfer Instructions"
- "Arithmetic Instructions"
- "Conversion Instructions"
- "Formatted Operand-Value Move Instructions"
- "Conditional Branch Instructions"
- "Miscellaneous Instructions"

5.9.1 Data Transfer Instructions

The FPU has two separate register sets: coprocessor general registers and coprocessor control registers. The FPU has a load/store architecture; all computations are done on data held in coprocessor general registers. The control registers are used to control FPU operation. Data is transferred between registers and the rest of the system with dedicated load, store, and move instructions. The transferred data is treated as unformatted binary data; no format conversions are performed, and therefore no IEEE floating point exceptions can occur.

The supported transfer operations are listed in Table 5-12.

Table 5-12 FPU Data Transfer Instructions

Transfer Direction			Data Transferred
FPU general reg	↔	Memory	Word/doubleword load/store
FPU general reg	↔	CPU general reg	Word move
FPU control reg	↔	CPU general reg	Word move

5.9.1.1 Data Alignment in Loads, Stores, and Moves

All coprocessor loads and stores operate on naturally-aligned data items. An attempt to load or store to an address that is not naturally aligned for the data item causes an Address Error exception. Regardless of byte-ordering (the endianness), the address of a word or doubleword is the smallest byte address in the object. For a big-endian machine, this is the most-significant byte; for a little-endian machine, this is the least-significant byte (endianness is described in "Byte Ordering and Endianness" on page 21).

5.9.1.2 Addressing Used in Data Transfer Instructions

The FPU has loads and stores using the same *register+offset* addressing as that used by the CPU. Moreover, for the FPU only, there are load and store instructions using *register+register* addressing.

Tables 5-13 through 5-15 list the FPU data transfer instructions.

Table 5-13 FPU Loads and Stores Using Register+Offset Address Mode

Mnemonic	Instruction	Defined in MIPS ISA
LDC1	Load Doubleword to Floating Point	MIPS32
LWC1	Load Word to Floating Point	MIPS32
SDC1	Store Doubleword to Floating Point	MIPS32
SWC1	Store Word to Floating Point	MIPS32

Table 5-14 FPU Loads and Using Register+Register Address Mode

Mnemonic	Instruction	Defined in MIPS ISA
LDXC1	Load Doubleword Indexed to Floating Point	MIPS64 MIPS32 Release 2
LUXC1	Load Doubleword Indexed Unaligned to Floating Point	MIPS64 MIPS32 Release 2
LWXC1	Load Word Indexed to Floating Point	MIPS64 MIPS32 Release 2
SDXC1	Store Doubleword Indexed to Floating Point	MIPS64 MIPS32 Release 2
SUXC1	Store Doubleword Indexed Unaligned to Floating Point	MIPS64 MIPS32 Release 2
SWXC1	Store Word Indexed to Floating Point	MIPS64 MIPS32 Release 2

Table 5-15 FPU Move To and From Instructions

Mnemonic	Instruction	Defined in MIPS ISA
CFC1	Move Control Word From Floating Point	MIPS32
CTC1	Move Control Word To Floating Point	MIPS32
MFC1	Move Word From Floating Point	MIPS32
MFHC1	Move Word from High Half of Floating Point Register	MIPS32 Release 2
MTC1	Move Word To Floating Point	MIPS32
MTHC1	Move Word to High Half of Floating Point Register	MIPS32 Release 2

5.9.2 Arithmetic Instructions

Arithmetic instructions operate on formatted data values. The results of most floating point arithmetic operations meet the IEEE standard specification for accuracy—a result is identical to an infinite-precision result that has been rounded to the specified format, using the current rounding mode. The rounded result differs from the exact result by less than one unit in the least-significant place (ULP).

FPU IEEE-approximate arithmetic operations are listed in Table 5-16.

Table 5-16 FPU IEEE Arithmetic Operations

Mnemonic	Instruction	Defined in MIPS ISA
ABS.fmt	Floating Point Absolute Value	MIPS32
ABS.fmt (PS)	Floating Point Absolute Value (Paired Single)	MIPS64 MIPS32 Release 2
ADD.fmt	Floating Point Add	MIPS32
ADD.fmt *(PS)*	Floating Point Add (Paired Single)	MIPS64 MIPS32 Release 2
C.cond.fmt	Floating Point Compare	MIPS32
C.cond.fmt *(PS)*	Floating Point Compare (Paired Single)	MIPS64 MIPS32 Release 2
DIV.fmt	Floating Point Divide	MIPS32
MUL.fmt	Floating Point Multiply	MIPS32
MUL.fmt *(PS)*	Floating Point Multiply (Paired Single)	MIPS64 MIPS32 Release 2
NEG.fmt	Floating Point Negate	MIPS32
NEG.fmt *(PS)*	Floating Point Negate (Paired Single)	MIPS64 MIPS32 Release 2
SQRT.fmt	Floating Point Square Root	MIPS32
SUB.fmt	Floating Point Subtract	MIPS32
SUB.fmt *(PS)*	Floating Point Subtract (Paired Single)	MIPS64 MIPS32 Release 2

Two operations, Reciprocal Approximation (RECIP) and Reciprocal Square Root Approximation (RSQRT), may be less accurate than the IEEE specification:

- The result of RECIP differs from the exact reciprocal by no more than one ULP.
- The result of RSQRT differs from the exact reciprocal square root by no more than two ULPs.

Within these error limits, the results of these instructions are implementation specific.

A list of FPU-approximate arithmetic operations is given in Table 5-17..

Table 5-17 FPU-Approximate Arithmetic Operations

Mnemonic	Instruction	Defined in MIPS ISA
RECIP.fmt	Floating Point Reciprocal Approximation	MIPS64 MIPS32 Release 2
RSQRT.fmt	Floating Point Reciprocal Square Root Approximation	MIPS64 MIPS32 Release 2

Four compound-operation instructions perform variations of multiply-accumulate—that is, multiply two operands, accumulate the result to a third operand, and produce a result. These instructions are listed in Table 5-18. The product is rounded according to the current rounding mode prior to the accumulation. This model meets the IEEE accuracy specification; the result is numerically identical to an equivalent computation using multiply, add, subtract, or negate instructions.

Table 5-18 lists the FPU Multiply-Accumulate arithmetic operations.

Table 5-18 FPU Multiply-Accumulate Arithmetic Operations

Mnemonic	Instruction	Defined in MIPS ISA
MADD.fmt	Floating Point Multiply Add	MIPS64 MIPS32 Release 2
MADD.fmt *(PS)*	Floating Point Multiply Add (Paired Single)	MIPS64 MIPS32 Release 2
MSUB.fmt	Floating Point Multiply Subtract	MIPS64 MIPS32 Release 2
MSUB.fmt *(PS)*	Floating Point Multiply Subtract (Paired Single)	MIPS64 MIPS32 Release 2
NMADD.fmt	Floating Point Negative Multiply Add	MIPS64 MIPS32 Release 2
NMADD.fmt *(PS)*	Floating Point Negative Multiply Add (Paired Single)	MIPS64 MIPS32 Release 2
NMSUB.fmt	Floating Point Negative Multiply Subtract	MIPS64 MIPS32 Release 2
NMSUB.fmt *(PS)*	Floating Point Negative Multiply Subtract (Paired Single)	MIPS64 MIPS32 Release 2

5.9.3 Conversion Instructions

These instructions perform conversions between floating point and fixed point data types. Each instruction converts values from a number of operand formats to a particular result format. Some conversion instructions use the rounding mode specified in the *Floating Control/Status* register (*FCSR*), while others specify the rounding mode directly. Table 5-19 and Table 5-20 list the FPU conversion instructions according to their rounding mode.

Table 5-19 FPU Conversion Operations Using the *FCSR* Rounding Mode

Mnemonic	Instruction	Defined in MIPS ISA
CVT.D.fmt	Floating Point Convert to Double Floating Point	MIPS32
CVT.L.fmt	Floating Point Convert to Long Fixed Point	MIPS64 MIPS32 Release 2
CVT.PS.*S*	Floating Point Convert Pair to Paired Single	MIPS64 MIPS32 Release 2
CVT.S.fmt	Floating Point Convert to Single Floating Point	MIPS32
CVT.S.fmt *(PL, PU)*	Floating Point Convert to Single Floating Point (Paired Lower, Paired Upper)	MIPS64 MIPS32 Release 2
CVT.W.fmt	Floating Point Convert to Word Fixed Point	MIPS32

Table 5-20 FPU Conversion Operations Using a Directed Rounding Mode

Mnemonic	Instruction	Defined in MIPS ISA
CEIL.L.fmt	Floating Point Ceiling to Long Fixed Point	MIPS64 MIPS32 Release 2
CEIL.W.fmt	Floating Point Ceiling to Word Fixed Point	MIPS32

Table 5-20 FPU Conversion Operations Using a Directed Rounding Mode

Mnemonic	Instruction	Defined in MIPS ISA
FLOOR.L.fmt	Floating Point Floor to Long Fixed Point	MIPS64 MIPS32 Release 2
FLOOR.W.fmt	Floating Point Floor to Word Fixed Point	MIPS32
ROUND.L.fmt	Floating Point Round to Long Fixed Point	MIPS64 MIPS32 Release 2
ROUND.W.fmt	Floating Point Round to Word Fixed Point	MIPS32
TRUNC.L.fmt	Floating Point Truncate to Long Fixed Point	MIPS64 MIPS32 Release 2
TRUNC.W.fmt	Floating Point Truncate to Word Fixed Point	MIPS32

5.9.4 Formatted Operand-Value Move Instructions

These instructions all move formatted operand values among FPU general registers. A particular operand type must be moved by the instruction that handles that type. There are three kinds of move instructions:

- Unconditional move
- Conditional move that tests an FPU true/false condition code
- Conditional move that tests a CPU general-purpose register against zero

Conditional move instructions operate in a way that may be unexpected. They always force the value in the destination register to become a value of the format specified in the instruction. If the destination register does not contain an operand of the specified format before the conditional move is executed, the contents become undefined. (For more information, see the individual descriptions of the conditional move instructions in Volume II.)

These instructions are listed in Tables Table 5-21 through Table 5-23.

Table 5-21 FPU Formatted Operand Move Instructions

Mnemonic	Instruction	Defined in MIPS ISA
MOV.fmt	Floating Point Move	MIPS32
MOV.fmt *(PS)*	Floating Point Move (Paired Single)	MIPS64 MIPS32 Release 2

Table 5-22 FPU Conditional Move on True/False Instructions

Mnemonic	Instruction	Defined in MIPS ISA
MOVF.fmt	Floating Point Move Conditional on FP False	MIPS32
MOVF.fmt *(PS)*	Floating Point Move Conditional on FP False (Paired Single)	MIPS64 MIPS32 Release 2
MOVT.fmt	Floating Point Move Conditional on FP True	MIPS32
MOVT.fmt *(PS)*	Floating Point Move Conditional on FP True (Paired Single)	MIPS64 MIPS32 Release 2

Table 5-23 FPU Conditional Move on Zero/Nonzero Instructions

Mnemonic	Instruction	Defined in MIPS ISA
MOVN.fmt	Floating Point Move Conditional on Nonzero	MIPS32
MOVN.fmt *(PS)*	Floating Point Move Conditional on Nonzero (Paired Single)	MIPS64 MIPS32 Release 2
MOVZ.fmt	Floating Point Move Conditional on Zero	MIPS32
MOVZ.fmt *(PS)*	Floating Point Move Conditional on Zero (Paired Single)	MIPS64 MIPS32 Release 2

5.9.5 Conditional Branch Instructions

The FPU has PC-relative conditional branch instructions that test condition codes set by FPU compare instructions (C.cond.fmt).

All branches have an architectural delay of one instruction. When a branch is taken, the instruction immediately following the branch instruction is said to be in the **branch delay slot**, and it is executed before the branch to the target instruction takes place. Conditional branches come in two versions, depending upon how they handle an instruction in the delay slot when the branch is not taken and execution falls through:

- **Branch** instructions execute the instruction in the delay slot.
- **Branch likely** instructions do not execute the instruction in the delay slot if the branch is not taken (they are said to *nullify* the instruction in the delay slot).

  Although the Branch Likely instructions are included in this specification, software is strongly encouraged to avoid the use of the Branch Likely instructions, as they will be removed from a future revision of the MIPS Architecture.

The MIPS32 Architecture defines eight condition codes for use in compare and branch instructions. For backward compatibility with previous revision of the ISA, condition code bit 0 and condition code bits 1 thru 7 are in discontiguous fields in *FCSR*.

Table 5-24 lists the conditional branch (branch and branch likely) FPU instructions; Table 5-25 lists the deprecated conditional branch likely instructions.

Table 5-24 FPU Conditional Branch Instructions

Mnemonic	Instruction	Defined in MIPS ISA
BC1F	Branch on FP False	MIPS32
BC1T	Branch on FP True	MIPS32

Table 5-25 Deprecated FPU Conditional Branch Likely Instructions

Mnemonic	Instruction	Defined in MIPS ISA
BC1FL	Branch on FP False Likely	MIPS32
BC1TL	Branch on FP True Likely	MIPS32

5.9.6 Miscellaneous Instructions

The MIPS ISA defines various miscellaneous instructions that conditionally move one CPU general register to another, based on an FPU condition code. It also defines an instruction to align a misaligned pair of paired-single values (ALNV.PS) and a quartet of instructions that merge a pair of paired-single values (PLL.PS, PLU.PS, PUL.PS, PUU.PS).

Table 5-26 lists these conditional move instructions.

Table 5-26 CPU Conditional Move on FPU True/False Instructions

Mnemonic	Instruction	Defined in MIPS ISA
ALNV.PS	FP Align Variable	MIPS64 MIPS32 Release 2
MOVN	Move Conditional on FP False	MIPS32
MOVZ	Move Conditional on FP True	MIPS32
PLL.PS	Pair Lower Lower	MIPS64 MIPS32 Release 2
PLU.PS	Pair Lower Upper	MIPS64 MIPS32 Release 2
PUL.PS	Pair Upper Lower	MIPS64 MIPS32 Release 2
PUU.PS	Pair Upper Upper	MIPS64 MIPS32 Release 2

5.10 Valid Operands for FPU Instructions

The floating point unit arithmetic, conversion, and operand move instructions operate on formatted values with different precision and range limits and produce formatted values for results. Each representable value in each format has a binary encoding that is read from or stored to memory. The *fmt* or *fmt3* field of the instruction encodes the operand format required for the instruction. A conversion instruction specifies the result type in the *function* field; the result of other operations is given in the same format as the operands. The encodings of the *fmt* and *fmt3* field are shown in Table 5-27.

Table 5-27 FPU Operand Format Field (*fmt*, *fmt3*) Encoding

fmt	fmt3	Instruction Mnemonic	Size		Data Type
			Name	Bits	
0-15	-	Reserved			
16	0	S	single	32	Floating point
17	1	D	double	64	Floating point
18-19	2-3	Reserved			
20	4	W	word	32	Fixed point
21	5	L	long	64	Fixed point
22	6	PS	paired single	64	Floating point
23–31	7	Reserved			