2.6 Instructions, Sorted by ISA

This section lists the instructions that are a part of the MIPS32 and MIPS64 ISAs.

2.6.1 List of MIPS32 Instructions

Table 2-1 lists of those instructions included in the MIPS32 ISA.

Table 2-1 MIPS32 Instructions

ABS.D	ABS.PS[1]	ABS.S	ADD	ADD.D	ADD.PS[1]	ADD.S	ADDI
ADDIU	ADDU	ALNV.PS[1]	AND	ANDI	BC1F	BC1FL	BC1T
BC1TL	BC2F	BC2FL	BC2T	BC2TL	BEQ	BEQL	BGEZ
BGEZAL	BGEZALL	BGEZL	BGTZ	BGTZL	BLEZ	BLEZL	BLTZ
BLTZAL	BLTZALL	BLTZL	BNE	BNEL	BREAK	C.cond.D	C.cond.PS[1]
C.cond.S	CACHE	CEIL.L.D[1]	CEIL.L.S[1]	CEIL.W.D	CEIL.W.S	CFC1	CFC2
CLO	CLZ	COP2	CTC1	CTC2	CVT.D.L[1]	CVT.D.S	CVT.D.W
CVT.L.D[1]	CVT.L.S[1]	CVT.PS.S[1]	CVT.S.D	CVT.S.L[1]	CVT.S.PL[1]	CVT.S.PU[1]	CVT.S.W
CVT.W.D	CVT.W.S	DERET	DI[2]	DIV	DIV.D	DIV.S	DIVU
EHB[2]	EI[2]	ERET	EXT[2]	FLOOR.L.D[1]	FLOOR.L.S[1]	FLOOR.W.D	FLOOR.W.S
INS[2]	J	JAL	JALR	JALR.HB[2]	JR	JR.HB[2]	LB
LBU	LDC1	LDC2	LDXC1[1]	LH	LHU	LL	LUI
LUXC1[1]	LW	LWC1	LWC2	LWL	LWR	LWXC1[1]	MADD
MADD.D[1]	MADD.PS[1]	MADD.S[1]	MADDU	MFC0	MFC1	MFC2	MFHC1[2]
MFHC2[2]	MFHI	MFLO	MOV.D	MOV.PS[1]	MOV.S	MOVF	MOVF.D
MOVF.PS[1]	MOVF.S	MOVN	MOVN.D	MOVN.PS[1]	MOVN.S	MOVT	MOVT.D
MOVT.PS[1]	MOVT.S	MOVZ	MOVZ.D	MOVZ.PS[1]	MOVZ.S	MSUB	MSUB.D[1]
MSUB.PS[1]	MSUB.S[1]	MSUBU	MTC0	MTC1	MTC2	MTHC1[2]	MTHC2[2]
MTHI	MTLO	MUL	MUL.D	MUL.PS[1]	MUL.S	MULT	MULTU
NEG.D	NEG.PS[1]	NEG.S	NMADD.D[1]	NMADD.PS[1]	NMADD.S[1]	NMSUB.D[1]	NMSUB.PS[1]
NMSUB.S[1]	NOR	OR	ORI	PLL.PS[1]	PLU.PS[1]	PREF	PREFX[1]
PUL.PS[1]	PUU.PS[1]	RDHWR[2]	RDPGPR[2]	RECIP.D[1]	RECIP.S[1]	ROTR[2]	ROTRV[2]
ROUND.L.D[1]	ROUND.L.S[1]	ROUND.W.D	ROUND.W.S	RSQRT.D[1]	RSQRT.S[1]	SB	SC
SDBBP	SDC1	SDC2	SDXC1[1]	SEB[2]	SEH[2]	SH	SLL
SLLV	SLT	SLTI	SLTIU	SLTU	SQRT.D	SQRT.S	SRA
SRAV	SRL	SRLV	SSNOP	SUB	SUB.D	SUB.PS[1]	SUB.S
SUBU	SUXC1[1]	SW	SWC1	SWC2	SWL	SWR	SWXC1[1]
SYNC	SYNCI[2]	SYSCALL	TEQ	TEQI	TGE	TGEI	TGEIU
TGEU	TLBP	TLBR	TLBWI	TLBWR	TLT	TLTI	TLTIU
TLTU	TNE	TNEI	TRUNC.L.D[1]	TRUNC.L.S[1]	TRUNC.W.D	TRUNC.W.S	WAIT
WRPGPR[2]	WSBH[2]	XOR	XORI				

1. In Release 1 of the Architecture, these instructions are legal only with a MIPS64 processor with 64-bit operations enabled (they are, in effect, actually MIPS64 instructions). In Release 2 of the Architecture, these instructions are legal with either a MIPS32 or MIPS64 processor which includes a 64-bit floating point unit.
2. These instructions are legal only in an implementation of Release 2 of the Architecture

Table 4-24 describes the fields used in these instructions.

Table 4-24 CPU Instruction Format Fields

Field	Description
opcode	6-bit primary operation code
rd	5-bit specifier for the destination register
rs	5-bit specifier for the source register
rt	5-bit specifier for the target (source/destination) register or used to specify functions within the primary *opcode* REGIMM
immediate	16-bit signed *immediate* used for logical operands, arithmetic signed operands, load/store address byte offsets, and PC-relative branch signed instruction displacement
instr_index	26-bit index shifted left two bits to supply the low-order 28 bits of the jump target address
sa	5-bit shift amount
function	6-bit function field used to specify functions within the primary *opcode* SPECIAL

Figure 4-1 Immediate (I-Type) CPU Instruction Format

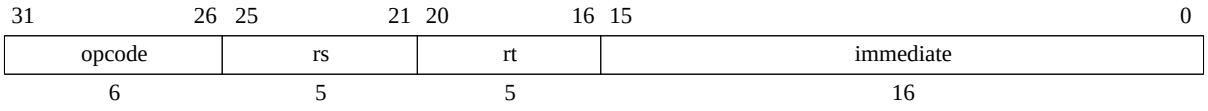


Figure 4-2 Jump (J-Type) CPU Instruction Format

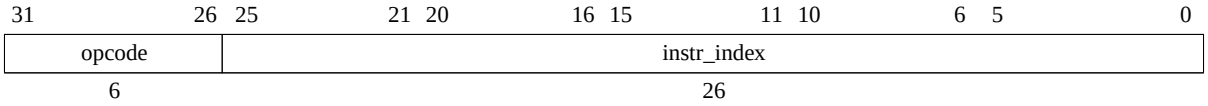
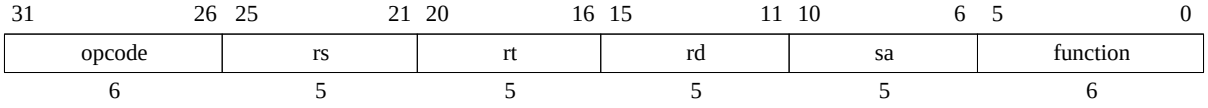


Figure 4-3 Register (R-Type) CPU Instruction Format



Instruction Bit Encodings

A.1 Instruction Encodings and Instruction Classes

Instruction encodings are presented in this section; field names are printed here and throughout the book in *italics*.

When encoding an instruction, the primary *opcode* field is encoded first. Most *opcode* values completely specify an instruction that has an *immediate* value or offset.

Opcode values that do not specify an instruction instead specify an instruction class. Instructions within a class are further specified by values in other fields. For instance, *opcode* REGIMM specifies the *immediate* instruction class, which includes conditional branch and trap *immediate* instructions.

A.2 Instruction Bit Encoding Tables

This section provides various bit encoding tables for the instructions of the MIPS32® ISA.

Figure A-1 shows a sample encoding table and the instruction *opcode* field this table encodes. Bits 31..29 of the *opcode* field are listed in the leftmost columns of the table. Bits 28..26 of the *opcode* field are listed along the topmost rows of the table. Both decimal and binary values are given, with the first three bits designating the row, and the last three bits designating the column.

An instruction's encoding is found at the intersection of a row (bits 31..29) and column (bits 28..26) value. For instance, the *opcode* value for the instruction labelled EX1 is 33 (decimal, row and column), or 011011 (binary). Similarly, the *opcode* value for EX2 is 64 (decimal), or 110100 (binary).

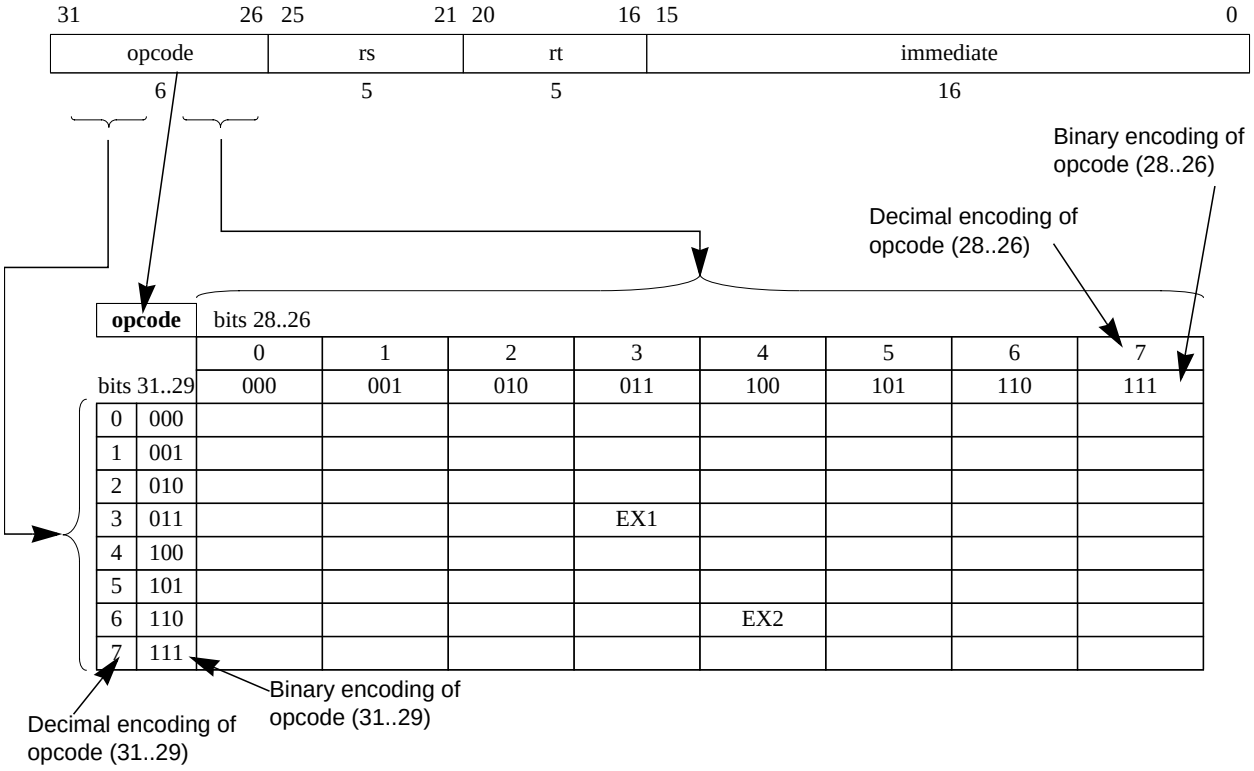


Figure A-1 Sample Bit Encoding Table

Tables A-2 through A-20 describe the encoding used for the MIPS32 ISA. Table A-1 describes the meaning of the symbols used in the tables.

Table A-1 Symbols Used in the Instruction Encoding Tables

Symbol	Meaning
*	Operation or field codes marked with this symbol are reserved for future use. Executing such an instruction must cause a Reserved Instruction Exception.
δ	(Also *italic* field name.) Operation or field codes marked with this symbol denotes a field class. The instruction word must be further decoded by examining additional tables that show values for another instruction field.
β	Operation or field codes marked with this symbol represent a valid encoding for a higher-order MIPS ISA level or a new revision of the Architecture. Executing such an instruction must cause a Reserved Instruction Exception.
∇	Operation or field codes marked with this symbol represent instructions which were only legal if 64-bit operations were enabled on implementations of Release 1 of the Architecture. In Release 2 of the architecture, operation or field codes marked with this symbol represent instructions which are legal if 64-bit floating point operations are enabled. In other cases, executing such an instruction must cause a Reserved Instruction Exception (non-coprocessor encodings or coprocessor instruction encodings for a coprocessor to which access is allowed) or a Coprocessor Unusable Exception (coprocessor instruction encodings for a coprocessor to which access is not allowed).

Table A-1 Symbols Used in the Instruction Encoding Tables

Symbol	Meaning
θ	Operation or field codes marked with this symbol are available to licensed MIPS partners. To avoid multiple conflicting instruction definitions, MIPS Technologies will assist the partner in selecting appropriate encodings if requested by the partner. The partner is not required to consult with MIPS Technologies when one of these encodings is used. If no instruction is encoded with this value, executing such an instruction must cause a Reserved Instruction Exception (*SPECIAL2* encodings or coprocessor instruction encodings for a coprocessor to which access is allowed) or a Coprocessor Unusable Exception (coprocessor instruction encodings for a coprocessor to which access is not allowed).
σ	Field codes marked with this symbol represent an EJTAG support instruction and implementation of this encoding is optional for each implementation. If the encoding is not implemented, executing such an instruction must cause a Reserved Instruction Exception. If the encoding is implemented, it must match the instruction encoding as shown in the table.
ε	Operation or field codes marked with this symbol are reserved for MIPS Application Specific Extensions. If the ASE is not implemented, executing such an instruction must cause a Reserved Instruction Exception.
ϕ	Operation or field codes marked with this symbol are obsolete and will be removed from a future revision of the MIPS32 ISA. Software should avoid using these operation or field codes.
⊕	Operation or field codes marked with this symbol are valid for Release 2 implementations of the architecture. Executing such an instruction in a Release 1 implementation must cause a Reserved Instruction Exception.

Table A-2 MIPS32 Encoding of the Opcode Field

opcode		*bits 28..26*							
		0	1	2	3	4	5	6	7
bits 31..29		000	001	010	011	100	101	110	111
0	000	*SPECIAL* δ	*REGIMM* δ	J	JAL	BEQ	BNE	BLEZ	BGTZ
1	001	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2	010	*COP0* δ	*COP1* δ	*COP2* θδ	*COP1X*[1] δ	BEQL ϕ	BNEL ϕ	BLEZL ϕ	BGTZL ϕ
3	011	β	β	β	β	*SPECIAL2* δ	JALX ε	ε	*SPECIAL3*[2] δ⊕
4	100	LB	LH	LWL	LW	LBU	LHU	LWR	β
5	101	SB	SH	SWL	SW	β	β	SWR	CACHE
6	110	LL	LWC1	LWC2 θ	PREF	β	LDC1	LDC2 θ	β
7	111	SC	SWC1	SWC2 θ	*	β	SDC1	SDC2 θ	β

1. In Release 1 of the Architecture, the COP1X opcode was called COP3, and was available as another user-available coprocessor. In Release 2 of the Architecture, a full 64-bit floating point unit is available with 32-bit CPUs, and the COP1X opcode is reserved for that purpose on all Release 2 CPUs. 32-bit implementations of Release 1 of the architecture are strongly discouraged from using this opcode for a user-available coprocessor as doing so will limit the potential for an upgrade path to a 64-bit floating point unit.
2. Release 2 of the Architecture added the SPECIAL3 opcode. Implementations of Release 1 of the Architecture signaled a Reserved Instruction Exception for this opcode.

Table A-3 MIPS32 *SPECIAL* Opcode Encoding of Function Field

function		*bits 2..0*							
		0	1	2	3	4	5	6	7
bits 5..3		000	001	010	011	100	101	110	111
0	000	SLL[1]	*MOVCI* δ	*SRL* δ	SRA	SLLV	*	*SRLV* δ	SRAV
1	001	JR[2]	JALR[2]	MOVZ	MOVN	SYSCALL	BREAK	*	SYNC
2	010	MFHI	MTHI	MFLO	MTLO	β	*	β	β
3	011	MULT	MULTU	DIV	DIVU	β	β	β	β
4	100	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
5	101	*	*	SLT	SLTU	β	β	β	β
6	110	TGE	TGEU	TLT	TLTU	TEQ	*	TNE	*
7	111	β	*	β	β	β	*	β	β

1. Specific encodings of the *rt*, *rd*, and *sa* fields are used to distinguish among the SLL, NOP, SSNOP and EHB functions.
2. Specific encodings of the hint field are used to distinguish JR from JR.HB and JALR from JALR.HB

Table A-4 MIPS32 *REGIMM* Encoding of rt Field

rt		*bits 18..16*							
		0	1	2	3	4	5	6	7
bits 20..19		000	001	010	011	100	101	110	111
0	00	BLTZ	BGEZ	BLTZL ϕ	BGEZL ϕ	*	*	*	*
1	01	TGEI	TGEIU	TLTI	TLTIU	TEQI	*	TNEI	*
2	10	BLTZAL	BGEZAL	BLTZALL ϕ	BGEZALL ϕ	*	*	*	*
3	11	*	*	*	*	*	*	*	SYNCI ⊕

Table A-5 MIPS32 *SPECIAL2* Encoding of Function Field

function		*bits 2..0*							
		0	1	2	3	4	5	6	7
bits 5..3		000	001	010	011	100	101	110	111
0	000	MADD	MADDU	MUL	θ	MSUB	MSUBU	θ	θ
1	001	θ	θ	θ	θ	θ	θ	θ	θ
2	010	θ	θ	θ	θ	θ	θ	θ	θ
3	011	θ	θ	θ	θ	θ	θ	θ	θ
4	100	CLZ	CLO	θ	θ	β	β	θ	θ
5	101	θ	θ	θ	θ	θ	θ	θ	θ
6	110	θ	θ	θ	θ	θ	θ	θ	θ
7	111	θ	θ	θ	θ	θ	θ	θ	SDBBP σ

Table A-6 MIPS32 *SPECIAL3*[1] Encoding of Function Field for Release 2 of the Architecture

function		*bits 2..0*							
		0	1	2	3	4	5	6	7
bits 5..3		000	001	010	011	100	101	110	111
0	000	EXT ⊕	β	β	β	INS ⊕	β	β	β
1	001	*	*	*	*	*	*	*	*
2	010	*	*	*	*	*	*	*	*
3	011	*	*	*	*	*	*	*	*
4	100	*BSHFL* ⊕δ	*	*	*	β	*	*	*
5	101	*	*	*	*	*	*	*	*
6	110	*	*	*	*	*	*	*	*
7	111	*	*	*	RDHWR ⊕	*	*	*	*

1. Release 2 of the Architecture added the SPECIAL3 opcode. Implementations of Release 1 of the Architecture signaled a Reserved Instruction Exception for this opcode and all function field values shown above.

Table A-7 MIPS32 *MOVCI* Encoding of tf Bit

tf	*bit 16*	
	0	1
	MOVF	MOVT

Table A-8 MIPS32[1] *SRL* Encoding of Shift/Rotate

R	*bit 21*	
	0	1
	SRL	ROTR

1. Release 2 of the Architecture added the ROTR instruction. Implementations of Release 1 of the Architecture ignored bit 21 and treated the instruction as an SRL

Table A-9 MIPS32[1] *SRLV* Encoding of Shift/Rotate

R	*bit 6*	
	0	1
	SRLV	ROTRV

1. Release 2 of the Architecture added the ROTRV instruction. Implementations of Release 1 of the Architecture ignored bit 6 and treated the instruction as an SRLV

Table A-10 MIPS32 *BSHFL* Encoding of sa Field[1]

sa		*bits 8..6*							
		0	1	2	3	4	5	6	7
bits 10..9		000	001	010	011	100	101	110	111
0	00			WSBH					
1	01								
2	10	SEB							
3	11	SEH							

1. The sa field is sparsely decoded to identify the final instructions. Entries in this table with no mnemonic are reserved for future use by MIPS Technologies and may or may not cause a Reserved Instruction exception.

Table A-11 MIPS32 *COP0* Encoding of rs Field

rs		*bits 23..21*							
		0	1	2	3	4	5	6	7
bits 25..24		000	001	010	011	100	101	110	111
0	00	MFC0	β	*	*	MTC0	β	*	*
1	01	*	*	RDPGPR ⊕	*MFMC0*[1] δ⊕	*	*	WRPGPR ⊕	*
2	10	C0 δ							
3	11								

1. Release 2 of the Architecture added the MFMC0 function, which is further decoded as the DI and EI instructions.

Table A-12 MIPS32 *COP0* Encoding of Function Field When rs=*CO*

function		*bits 2..0*							
		0	1	2	3	4	5	6	7
bits 5..3		000	001	010	011	100	101	110	111
0	000	*	TLBR	TLBWI	*	*	*	TLBWR	*
1	001	TLBP	*	*	*	*	*	*	*
2	010	*	*	*	*	*	*	*	*
3	011	ERET	*	*	*	*	*	*	DERET σ
4	100	WAIT	*	*	*	*	*	*	*
5	101	*	*	*	*	*	*	*	*
6	110	*	*	*	*	*	*	*	*
7	111	*	*	*	*	*	*	*	*

Table A-13 MIPS32 *COP1* Encoding of rs Field

rs		*bits 23..21*							
		0	1	2	3	4	5	6	7
bits 25..24		000	001	010	011	100	101	110	111
0	00	MFC1	β	CFC1	MFHC1 ⊕	MTC1	β	CTC1	MTHC1 ⊕
1	01	*BC1* δ	*BC1ANY2* δεV	*BC1ANY4* δεV	*	*	*	*	*
2	10	*S* δ	*D* δ	*	*	*W* δ	*L* δ	*PS* δ	*
3	11	*	*	*	*	*	*	*	*

Table A-14 MIPS32 *COP1* Encoding of Function Field When rs=*S*

function		*bits 2..0*							
		0	1	2	3	4	5	6	7
bits 5..3		000	001	010	011	100	101	110	111
0	000	ADD	SUB	MUL	DIV	SQRT	ABS	MOV	NEG
1	001	ROUND.L V	TRUNC.L V	CEIL.L V	FLOOR.L V	ROUND.W	TRUNC.W	CEIL.W	FLOOR.W
2	010	*	MOVCF δ	MOVZ	MOVN	*	RECIP V	RSQRT V	*
3	011	*	*	*	*	RECIP2 εV	RECIP1 εV	RSQRT1 εV	RSQRT2 εV
4	100	*	CVT.D	*	*	CVT.W	CVT.L V	CVT.PSV	*
5	101	*	*	*	*	*	*	*	*
6	110	C.F CABS.F εV	C.UN CABS.UN εV	C.EQ CABS.EQ εV	C.UEQ CABS.UEQ εV	C.OLT CABS.OLT εV	C.ULT CABS.ULT εV	C.OLE CABS.OLE εV	C.ULE CABS.ULE εV
7	111	C.SF CABS.SF εV	C.NGLE CABS.NGLE εV	C.SEQ CABS.SEQ εV	C.NGL CABS.NGL εV	C.LT CABS.LT εV	C.NGE CABS.NGE εV	C.LE CABS.LE εV	C.NGT CABS.NGT εV

Table A-15 MIPS32 *COP1* Encoding of Function Field When rs=*D*

function		bits 2..0							
		0	1	2	3	4	5	6	7
bits 5..3		000	001	010	011	100	101	110	111
0	000	ADD	SUB	MUL	DIV	SQRT	ABS	MOV	NEG
1	001	ROUND.L ∇	TRUNC.L ∇	CEIL.L ∇	FLOOR.L ∇	ROUND.W	TRUNC.W	CEIL.W	FLOOR.W
2	010	*	MOVCF δ	MOVZ	MOVN	*	RECIP ∇	RSQRT ∇	*
3	011	*	*	*	*	RECIP2 ε∇	RECIP1 ε∇	RSQRT1 ε∇	RSQRT2 ε∇
4	100	CVT.S	*	*	*	CVT.W	CVT.L ∇	*	*
5	101	*	*	*	*	*	*	*	*
6	110	C.F CABS.F ε∇	C.UN CABS.UN ε∇	C.EQ CABS.EQ ε∇	C.UEQ CABS.UEQ ε∇	C.OLT CABS.OLT ε∇	C.ULT CABS.ULT ε∇	C.OLE CABS.OLE ε∇	C.ULE CABS.ULE ε∇
7	111	C.SF CABS.SF ε∇	C.NGLE CABS.NGLE ε∇	C.SEQ CABS.SEQ ε∇	C.NGL CABS.NGL ε∇	C.LT CABS.LT ε∇	C.NGE CABS.NGE ε∇	C.LE CABS.LE ε∇	C.NGT CABS.NGT ε∇

Table A-16 MIPS32 *COP1* Encoding of Function Field When rs=*W or L*[1]

function		bits 2..0							
		0	1	2	3	4	5	6	7
bits 5..3		000	001	010	011	100	101	110	111
0	000	*	*	*	*	*	*	*	*
1	001	*	*	*	*	*	*	*	*
2	010	*	*	*	*	*	*	*	*
3	011	*	*	*	*	*	*	*	*
4	100	CVT.S	CVT.D	*	*	*	*	CVT.PS.PW ε∇	*
5	101	*	*	*	*	*	*	*	*
6	110	*	*	*	*	*	*	*	*
7	111	*	*	*	*	*	*	*	*

1. Format type L is legal only if 64-bit floating point operations are enabled.

Table A-17 MIPS64 *COP1* Encoding of Function Field When rs=PS[1]

function		bits 2..0							
		0	1	2	3	4	5	6	7
bits 5..3		000	001	010	011	100	101	110	111
0	000	ADD ∇	SUB ∇	MUL ∇	*	*	ABS ∇	MOV ∇	NEG ∇
1	001	*	*	*	*	*	*	*	*
2	010	*	MOVCF δ∇	MOVZ ∇	MOVN ∇	*	*	*	*
3	011	ADDR ε∇	*	MULR ε∇	*	RECIP2 ε∇	RECIP1 ε∇	RSQRT1 ε∇	RSQRT2 ε∇
4	100	CVT.S.PU ∇	*	*	*	CVT.PW.PS ε∇	*	*	*
5	101	CVT.S.PL ∇	*	*	*	PLL.PS ∇	PLU.PS ∇	PUL.PS ∇	PUU.PS ∇
6	110	C.F ∇ CABS.F ε∇	C.UN ∇ CABS.UN ε∇	C.EQ ∇ CABS.EQ ε∇	C.UEQ ∇ CABS.UEQ ε∇	C.OLT ∇ CABS.OLT ε∇	C.ULT ∇ CABS.ULT ε∇	C.OLE ∇ CABS.OLE ε∇	C.ULE ∇ CABS.ULE ε∇
7	111	C.SF ∇ CABS.SF ε∇	C.NGLE ∇ CABS.NGLEε∇	C.SEQ ∇ CABS.SEQ ε∇	C.NGL ∇ CABS.NGL ε∇	C.LT ∇ CABS.LT ε∇	C.NGE ∇ CABS.NGE ε∇	C.LE ∇ CABS.LE ε∇	C.NGT ∇ CABS.NGT ε∇

1. Format type PS is legal only if 64-bit floating point operations are enabled.

Table A-18 MIPS32 *COP1* Encoding of tf Bit When rs=*S, D, or PS*, Function=*MOVCF*

tf	bit 16	
	0	1
	MOVF.fmt	MOVT.fmt

Table A-19 MIPS32 *COP2* Encoding of rs Field

rs		*bits 23..21*							
		0	1	2	3	4	5	6	7
bits 25..24		000	001	010	011	100	101	110	111
0	00	MFC2 θ	β	CFC2 θ	MFHC2 θ⊕	MTC2 θ	β	CTC2 θ	MTHC2 θ⊕
1	01	BC2 θ	*	*	*	*	*	*	*
2	10	*C2* θδ							
3	11								

Table A-20 MIPS64 *COP1X* Encoding of Function Field[1]

function		*bits 2..0*							
		0	1	2	3	4	5	6	7
bits 5..3		000	001	010	011	100	101	110	111
0	000	LWXC1 ∇	LDXC1 ∇	*	*	*	LUXC1 ∇	*	*
1	001	SWXC1 ∇	SDXC1 ∇	*	*	*	SUXC1 ∇	*	PREFX ∇
2	010	*	*	*	*	*	*	*	*
3	011	*	*	*	*	*	*	ALNV.PS ∇	*
4	100	MADD.S ∇	MADD.D ∇	*	*	*	*	MADD.PS ∇	*
5	101	MSUB.S ∇	MSUB.D ∇	*	*	*	*	MSUB.PS ∇	*
6	110	NMADD.S ∇	NMADD.D ∇	*	*	*	*	NMADD.PS ∇	*
7	111	NMSUB.S ∇	NMSUB.D ∇	*	*	*	*	NMSUB.PS ∇	*

1. COP1X instructions are legal only if 64-bit floating point operations are enabled.

A.3 Floating Point Unit Instruction Format Encodings

Instruction format encodings for the floating point unit are presented in this section. This information is a tabular presentation of the encodings described in tables Table A-13 and Table A-20 above.

Table A-21 Floating Point Unit Instruction Format Encodings

***fmt* field (bits 25..21 of COP1 opcode)**		***fmt3* field (bits 2..0 of COP1X opcode)**					
Decimal	**Hex**	**Decimal**	**Hex**	**Mnemonic**	**Name**	**Bit Width**	**Data Type**
0..15	00..0F	—	—	Used to encode Coprocessor 1 interface instructions (MFC1, CTC1, etc.). Not used for format encoding.			
16	10	0	0	S	Single	32	Floating Point
17	11	1	1	D	Double	64	Floating Point
18..19	12..13	2..3	2..3	Reserved for future use by the architecture.			
20	14	4	4	W	Word	32	Fixed Point
21	15	5	5	L	Long	64	Fixed Point
22	16	6	6	PS	Paired Single	2×32	Floating Point
23	17	7	7	Reserved for future use by the architecture.			