Revision 3.2

September, 1995

Charles Price

MIPS IV Instruction Set

MIPS IV Instruction Set

CPU Instruction Set

LQst of Figures

List of Tables

CPU Instruction Encoding tables

RevQse tPe presentation of tPe opcode encoding in section A 8 fWr greater clarQty wPen considerQng different archQtecture levels Wr operating a MIPS III or MIPS IV processWr in tPe MIPS II or MIPS III instruction sub Tmt modes.

TPere Qs a separate encoding tabl(fWr each archQtecture level. TPere Qs a table of tPe) TJ0 -1.3 TD-0.01 T

processWr in tPe MIPS II Wr MIPS III Qnstruction sub et modes.

TPere Qs a separate encoding table fWr each archQtecture level. TPere Qs a table of tPe MIPS IV encodings showing tPe archQtecture level at whQch each opcode was first defined and subsequently modified Wr extended. TPere Qs a separate table fWr each archQtecture revQsion In→II, II→III, and III→IV showing tPe changes made Qn tPat revQsion.

Tables A-1 and A-2 tabulate the supported lWad and store operations and indQcate the MIPS archQtecture level at whQch each operation was first supported. The instructions themselves are listed in the fWllWwing sections.

Table A-1 LWad/Store Operations Using Register + Offset AddressiVg Mode.

	CPU			coprocessor (except 0)	
Data Size	**LWad Signed**	**LWad Unsigned**	**Store**	**LWad**	**Store**
halfword	I	I	I		
word	I	III	I	I	I
doubleword	III		III	II	II
unaligned word	I		I		
unaligned doubleword	III				
liVked word (atWmQc Uodify)	II		II		
liVked doubleword (atomQc Uodify)	III		III		

Table A-2 LWad/Store Operations Using Register + Register AddressiVg Mode.

	flWatiVg-point coprocessor only	
Data Size	**LWad**	**Store**
word	IV	IV
doubleword	IV	IV

A 2.1.1 Delayed LWads

The MIPS I archQtecture defines delayed lWads; an instruction scheduling restrQction requires that an instruction immediately fWllWwing a lWad into register *Rn* cannWt use *Rn* as a source register. The time between the lWad instruction and the time the data is available is the “lWad delay slWt”. If nW useful instruction can be put into the lWad delay slWt, then a null operation (assembler mneUonQc NOP) must be inserted.

In MIPS II, this instruction scheduliVg restrQction is reUoved. Programs will execute correctly when the lWaded data is used by the instruction fWllowiVg the lWad, but this may require extra real cycles. Most processors cannWt actually lWad data quicSly enough fWr immediate use and the processor will be fWrced to waQt until the data is available. ScheduliVg lWad delay slWts is desirable fWr perfWrmaVce reasons even when it is nWt necessary fWr correctness.

A 2.1.3 Atomic Update LWads and Stores

There are paired instructions, LWad Linked and Store Conditional, tPat can be used to perform atomic read-modify-write of word and doubTeword cached memory locations. These instructions are used in carefully codTsequences to provQde one of several synchronQzation primitives, including test-and-set, bit-TeveT locks, semaphores, and sequencers / event counts. The indivQdual instruction descriptions describe how to use theU.

TabTe A-5 Atomic UUpe CPU LWad/Store Instructions

A 2.1.4 CWprocessor LWads and Stores

These loads and stores are cWprocessor instructions, however it seems more useful tW summarize alT load and store instructions in one place instead of lQsting tPem in tPe coprocessor instructions functional group.

Table A-9 3-Operand ALU Instructions

MnemoVic	Description	Defined in
ADD	Add Word	MIPS I
ADDU	Add Unsigned Word	I

A 2.2.6 Shifts

There are shift instructions tPat take tPe shift amount frWm a 5-bQt field in the instruction wWrd and shift instructions tPat take a shift amount frWm tPe lWw-Wrder bQts of a general register. The instrstruì wQth a fixed shift amount are lQmQted to a 5-bQt shift count, so thereTce separate instructions fWr doublewWrd shifts of 0-31 bQts and 32-63 bQts.

Table A-10 Shift Instructions

Table A-12 Jump Instructions JumpQng WithQn a 256 Megabyte Region

Mnemonic	DescrQption	Defined Qn
J	Jump	MIPS I
JAL	Jump and LQnk	I

Table A-13 Jump Instructions to Absolute Address

Mnemonic	Description	Defined Qn
JR	Jump Register	MIPS I
JALR	Jump and LQnk Register	I

Table A-111 PC-Relative Conditional Branch Instructions ComparQng 2 Registers

Mnemonic	DescrQption	Defined Qn
BEQ	Branch on Equal	MIPS I
BNE	Branch on Not Equal	I
BLEZ	Branch on Less Than or Equal to ZerW	I
BGTZ	Branch on Greater Than ZerW	I
BEQL	Branch on Equal LQkely	II
BNEL	Branch on Not Equal LQkely	II
BLEZL	Branch on Less Than or Equal to Zero LQkely	II
BGTZL	Branch on Greater Than ZerW Likely	II

Table A-15 PC-Relative Conditional Branch Instructions ComparQng AgaQnst ZerW

Mnemonic	Description	Defined Qn
BLTZ	Branch on Less Than ZerW	MIPS I
BGEZ	Branch on Greater Than or Equal to ZerW	I
BLTZAL	Branch on Less Than ZerW and LQnk	I

A 2.4 Miscellaneous Instructions

A 2.4.1 Exception Instructions

Exception Qnstructions have as their sole purpose causQng an exception that will transfer control to a software exception handler Qn the kernel. System call and breakpoQnt Qnstructions cause exceptions unconditionally. The trap Qnstructions cause exceptions conditionally based upon the result of a comparQson.

Table A-16 System Call and BreakpoQnt Instructions

prefetched into the cache. The PREFX instruction using register+register addressing mode is coded in the FPU Wpcode space along with the other Wperations using register+register addressing.

Table A-21 PrefetcP Using Register + Offset Address Mode

Table A-22 Prefetch Using Register + Register Address Mode

A 2.5 CWprocessor Instructions

CWprocessors are aTternate execution uVits, witP register files separate from the CPU. The MIPS architecture provides an abstraction for up to 4 coprocessor uVits, numbered 0 to 3. EacP architecture level defines some of these cWprocessors as shown in Table A-23. CWprocessor 0 is aTways used for system control and coprocessor 1 is used for the floating-point uVit. Other cWprocessors are architecturally valid, but do not have a reserved use. Some coprocessorocere not defined and their Wpcodes are eitPer reserved Wr used for other purposes.

Table A-23 CWprWcessor DefiVition and Use in the MIPS Architecture

coprocessor cWntrol registers, eacP set containing up to thirtytwo registers. CWprocessor cWmputational instructions may alter registers in eitPer set.

System control for all MIPS processors is implemented as cWprocessor 0 (CP0), the System CWntrol CWprocessor. It provides the processor control, memory management, and exception handling fuVctions. The CP0 instructionocere specific to each CPU and are documented with the CPU-specific information.

If a system includes a floating-point uVit, it is implemented as cWprocessor 1 (CP1). In MIPSIV, the FPU also uses the computation Wpcode space for coprocessor uVit 3, renamed COP1X. The FPU instructions are documented in Appendix B.

Cached Coherent

Cached

For early 32-bit prWcessors witPout MP support, cached is equivalent to cached noncoherent. If an instruction description mentions the cached noncoherent access type, the comment appTies equally to the cached access type in a prWcessor that Pas the cached access type.

For prWcessors witP MP support, cached is a collective term, e.g. “cached memory” or “cached access”, that includes both cached noncoherent and cached coherent. Such a coTlective use does not imply that cached is an access type, it means that the statement applies equally to cached noncoherent and cached coherent access types.

A 3.1Mixing References witP Different Access Types

A 4 Description of an Instruction

The CPU instructions are described in alpPabetic order. Each description contains several sections tPat contain specific information about tPe instruction. The content of tPe section is described in detail below. An example description is shown in Figure A-1.*Figure A-1 Example Instruction Description*

Instruction mnemonic and descriptive name

Instruction encoding constant and variable field names aVd values

Architecture level at

Short description

Symbolic description

A 4.1 Instruction mnemonic aVd name

The instruction mnemonic and name are printed as page headings for each page in 8.instruction description.

This section uses acronyms for register descriptions. “GPR *rt*” is CPU General Purpose Register specQfied by the instruction field *rt*. “FPR *fs*” is the Floating Point Operand Register specQfied by the instruction field *fs*. “CP1 register

Symbol	Meaning

Table A-26 Coprocessor General Register Access Functions

A 5.3.2 Load and Store Memorysteunctions

RegardleisWf byte ordering (big- or lQttle-endian), the addreisWf a halfword, word, or doubleword is the smalleit byte address among the bytes forming the obRect. F oÅg-endian ordering this is the most-signifQcant byte; for a lQttle-endian ordering this is the lea66signifQcaVt byte.

In the operation description pseudocode for Toad and store operations, the functions shWwn beTow are used to summarize the handlQng Wf vQrtual addreises and accessing physical memory. The size of the data Qtem to be Toaded o 0 red is pa6sed in the *AccessLength* fieTd. The valid constant names and values are shWwn in Table A-27. The bytes wQthin the addreised unQt of memorys(word for 32-bit processors or doubleword for 64-bQt processors) which are used caV be determined directlysfrom the AccessLength and the two or three TWw-order bits of the address.

StoreMemory (CCA, AccessLength, MemElem, pAddr, vAddr)

CCA:	Cache Coherence Algorithm: the method used to access caches and memory and resolve the reference.
AccessLength:	Length, in bytA,, of access.
MemElem:	Data in the width and alignment of a memory element.

FR bit is valid for all existing
MIPS 64-bit prWcessors at the tQUe of this writQng, however this is a privileged

NullifyCurrentInstruction()

Nullify the current instruction.
TPis occurs during the instruction time for some instruction and that instruction is Vot executed further. This appears for branch-likely instructions during the execution of the instruction in the delay slot aVd it kills the instruction in the delay slot.

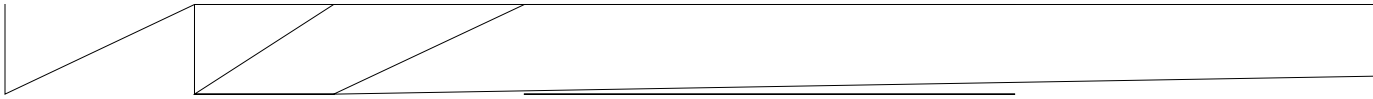
CoprocessorOperation (z, cop_fun)

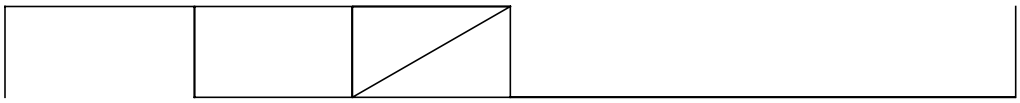
z	Coprocessor unit Vumber
cop_fun	Coprocessor function from function field of instruction

Perform the specified Coprocessor operation.

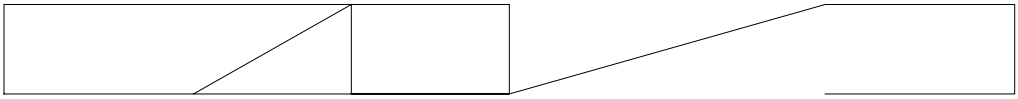
A 6 Individual CPU Instruction Descriptions

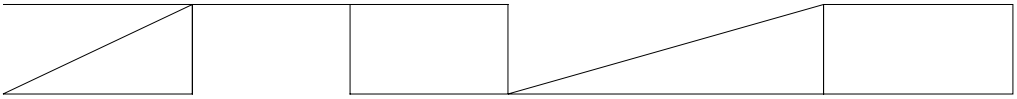
The user-mode CPU instructions are described in alphabetic order. See Description of an Instruction on page A-15 for a description of the inforUation in each instruction description.





Purpose:





A 8 CPU Instruction Encoding

TPis section descrQbes the enccog of user-level, i.e. non-prQvileged, CPU instructions for the four levels of the MIPS arcPitecture, MIPS I through MIPS IV. EacP arcPitecture level includes the instructions in the previous level;† MIPS IV includes all instructions in MIPS I, MIPS II, and MIPS III. TPis section presents eight different views of the instruction encoding.

- Separag enccding tables for each arcPitecture level.
- A MIPS IV enccong table showg the arcPitecture level at wPicP eacP opcode was orQgina66y defined and subsequently Ucofied (if Ucdified).
- Separage enccding tables for each arcPitecture revision showg the cPanges made durg thag revision.

A 8.1 Instruction Decode

Instruction field Vames are printed inboTd in this section.

TPe prQmary opcode fieTd is decoded first. Most opcode va6ues completely specify an instruction thag has an immediage value or offset.Opcode values thag do not specify an instruction specify an instruction class. Instructions wthin a class are further specified by va6ues in Wther fields. Theopcode values *SPECIAL and REGIMM specify instruction cTasses. The COP0, COP1*, COP2, COP3, and *COP1X instruction classes are not CPU instructions; they are discussed in section A 8.3.*

A 8.1.1 ***SPECIAL*** Instruction CTass

TPe opcode =SPECIAL instruction class encodes 3-register cWmputationa6 instructions, Rump register, and sWme specia6 purpose instructions. The class is further decoded by examin g the format fieTd. The formag values fu66y specify the CPU instructions; the *MOVCI*

Instructions encoded by

		000	001	010	011	100	101	110	111
0	000			J	JAL	BEQ	BNE	BLEZ	BGTZ
1	001	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2	010								
3	011	*	*						
4	100	LB	LH	LWL	LW	LBU	LHU	LWR	*
5	101	SB	SH	SWL	SW	*	*	SWR	*
6	110	*	LWC1	LWC2	LWC3				
7	111	*	SWC1						

Instructions encoded by field when opcode field = SPECIAL.

		000	001	010	011	100	101	110	111
0	000	SLL	*	SRL	SRA	SLLV	*	SRLV	SRAV
1	001	JR	JALR	*	*	SYSCALL	BREAK	*	*
2	010	MFHI	MTHI	MFLO	MTLO				
3	011	MULT	MULTU	DIV	DIVU				
4	100	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR

Instructions encoded by

	000	001	010	011	100	101	110	111
0 000			J	JAL	BEQ	BNE	BLEZ	BGTZ
1 001	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
2 010					BEQL	BNEL	BLEZL	BGTZL
3 011	DADDI	DADDIU	LDL	LDR				
4 100	LB	LH	LWL	LW	LBU	LHU	LWR	LWU
101	SB	SH	SWL	SW	SDL	SDR	SWR	
6 110	LL	LWC1		LLD	LDC1			
7 111	SC	SWC1		ΣΧΔ	ΣΔΧ1			

Instructions encoded by field when opcode field = SPECIAL.

	000	001	010	011	100	101	-4610	111
0 000	SLL	*	SRL	SRA	SLLV	*	SRLV	SRAV
1 001	JR	JALR	*	*	SYSCALL	BREAK	*	SYNC
2 010	MFHI	MTHI	MFLO	MTLO	DSLLV	*	DSRLV	DSRAV
3 0-461	MULT	MULTU	DIV	DIVU	DMULT	DMULTU	DDIV	DDIVU
4 100	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
101	*	*	SLT	SLTU	DADD	DADDU	DSUB	DSUBU
6 -4610	TGE	TGEU	TLT	TLTU	TEQ	*	TNE	*
7 111	DSLL	*	DSRL	DSRA	DSLL32	*	D 25L32	DSRA32

bQts 18..16 Instructions encoded by the field when opcode field = REGIMM.

	000	001	010	011	100	101	110	111
0 00	BLTZ	BGEZ	BLTZL	BGEZL				

Table A-41 Architecture Level in Which CPU Instructions are Defined or Extended.

SDC2 SDC3

functi on	bQts 2..0							
			Instructions encoded by function			Þeld when Wpcode Þeld = SPECIAL.		
bQts 5..3	0	1	2	3	4	5	6	7

31 26	
Wpcode	

An instruction encoding is shown if the instruction is added or modified in this revision.

Instructions encoded by Wpcode field.

Wpcode bits 28..26 / bits 31..29	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 000								
1 001								
2 010				* (was *COP3*)				
3 011	DADDI	DADDIU	LDL	LDR				
4 100								LWU
5 101					SDL	SDR		
6 110				* (was LWC3)	LLD			LD (was LDC3)
7 111				* (was SWC3)	SCD			SD (was SDC3)

31 26	5 0
Wpcode = *SPECIAL*	**function**

Instructions encoded by function field when Wpcode field = SPECIAL.

function bits 2..0 / bits 5..3	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 000								
1 001								
2 010					DSLLV		DSRLV	DSRAV
3 011					DMULT	DMULTU	DDIV	DDIVU
4 100								
5 101					DADD	DADDU	DSUB	DSUBU
6 110								
7 111	DSLL		DSRL	DSRA	DSLL32		DSRL32	DSRA32

Table A-44 CPU Instruction EncodQng Changes - MIPS IV RevQsion

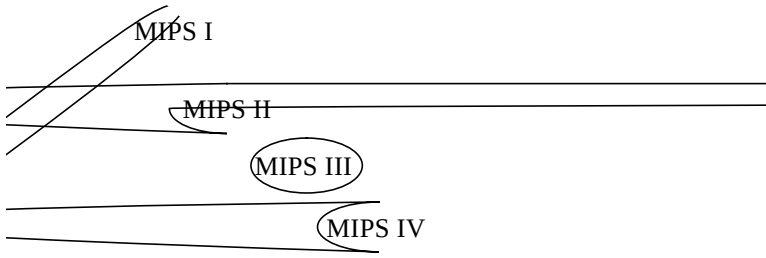
An Qnstruction encodQng Qs shWwn Qf the Qnstruction Qs added or modQfied Qn thQs revQsion.

opcode bits 31..29	bits 28..26	Instructions encoded by opcode field.						
	0	1	2	3	4	5	6	7
	000	001	010	011	100	101	110	111
0 000								
1 001								
2 010				*COP1X* δ,π				
3 011								
4 100								
5 101								
6 110				PREF				

function bits 5..3	bits 2..0	function field when opcode field = SPECIAL.						
	0	1	2	3	4	5	6	7
	000	001	010	011	100	101	110	111
0 000								
3 011								
4 100								
6 110								
7 111								

rt bits 20..19	bits 18..16	Instructions encoded by the rt field when opcode field = REGIMM.						
	0	1	2	3	4	5	6	7
	000	001	010	011	100	101	110	111
0 00								

B 1 Introduction



B 2.1 Floating-point fWrmats

- 32-bit Single precision floating-point (type S)
- 64-bit Double precisitioÑ37oating-point (type D)

1. Numbers of the fWrm: (-1)

1,

is also important for NaNs.

Table B-2 Value Wf Single or Double Floating-Point ForUat Encoding

B 2.1.1 NorUalized and DenorUalized Numbers

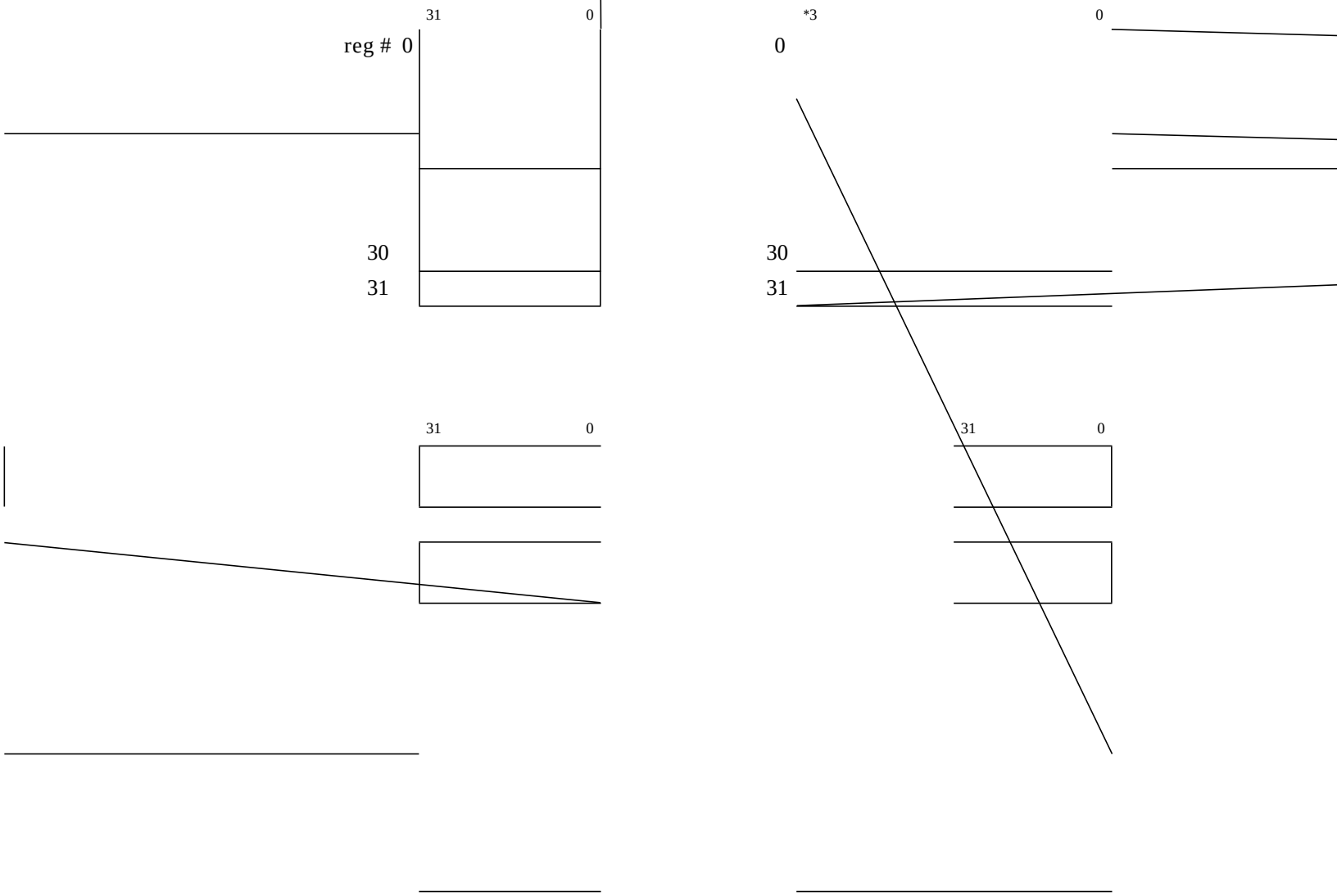
For single and double forUats, each representable Vonzero numerical value has just WVe encoding; numbers are kept in VorUalized forU. The higP-order bit Wf the *p*-bit Uantissa, which lies t3.the left Wf the binary point, is “hidden”, and Vot recorded in the fraction field. The encoding rules permit the value of this bit to be determined by looking at the value Wf the exponent. When the unbiased expoVent is in the range*E_min* t3 *E_Uax*

computation. A correctly sQgned is generated as the default retult in divQsion by

int

TPere are separate 32-bit aVd 64-bit wQde register modeTs. MIPS I defines tPe 32-bit wQde register model. MIPS III defines tPe 64-bit modeT. To support programs for earlier architecture definitions, processors provQding tPe 64-bit MIPS III register

B 3.1 Organization



#0

1

B 3.3 Formatted Operand Layout

The contents of this register are unpredictable and undefQned after a processor reset or a power-up event. Software should QnitialQze this register.

Figure B-12 MIPS I - FPU ContrWl and Status Register (FCSR)

31	24	23	22	18	17	12	11	7	6	2	1	0
0		*c*	*0*		*cafe*		*enables*		*flags*		*RM*	
8		*1*	*5*		*6*		*5*		*5*		*2*	

E	V	Z	O	U	I	V	Z	O	U	I	V	Z	O	U	I
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2

Figure B-13 MIPS III - FPU ContrWl and Status Register (FCSR)

31	25	24	23	22	18	17	12	11	7	6	2	1	0
0		*FS*	*c*	*0*		*cafse*		*enables*		*flags*		*RM*	
7		*1*	*1*	*5*		*6*		*5*		*5*		*2*	

E	V	Z	O	U	I	V	Z	O	U	I	V	Z	O	U	I
17	16	15	17	13	12	11	10	9	8	7	6	5	4	3	2

Figure B-14 MIPS IV - FPU ContrWl and Status Register (FCSR)

FCC	*FS*	*FCC*	*0*	*cafe*	*enables*	*flags*	*RM*
7	*1*	*1*	*5*	*6*	*5*	*5*	*2*

7	6	5	4	3	2	1	0
31	30	29	28	27	26	25	23

E	V	Z	O	U	I	V	Z	O	U	I	V	Z	O	U	I
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2

All fields Qn the FCSR are readable and writable.

FCC *FloatQng-PoQnt Condition Codes. These bits record the result of FP compares and are tested for FP conditional branches; the FCC bit tW use is specified in the compare or branch instruction. The 0 th FCC bit is the saUe as the c* bit Qn MIPS I.

FS Flush tW Zero. When FS is set, denormalQzed results are flushed tW zero

B 4 Values in FP Registers

UnTike the CPU, the FPU does Vot interpret the binary encoding of source operands or produce a binary encoding of results for every operation. The value held in a floating-point operand register (FPR) has a format, or type and it may only be used by instructions that operate on that format. The format of a value is either *uninterpreted*, *unkVowr*5 r one of the valid VumerQc formats: single and double floating-point and word and long fixed-point. The way that the formatted value in an FPR is set and changed is summarized in the state diagraU in Figure B-15 and is discussed below.

The value in an FPR is always set when a value is written to the register. When a data transfer instruction writes binary data into an FPR (a load), the FPR gets a binary value that is *uninterpreted*. A computational or FP register move instruction that produces a result of type *fmt* puts a value of type *fmt* into the result register.

uninterpreted

: ExaUpTe formats

LWad: DestiVation Wf LWC1, LDC1, MTC1, or DMTC1 instructions.

Store: SWurce Wperand WfSWC1, SDC1, MFC1, or DMFC1 instructions.

SWurce operand of coUprocessor instruction expecting format “

There are five types of exceptions that shalT be signaled when detected. The signaT entails setting a status flag, taking a trap, or possibly doing both. With each exception should be associated a trap under user contrWl,

ThQs function Qs implemented in the MIPS FPU architecture with the cause, enable, a-1 flag fQelds of the contrWl a-1 status regQster. The flag bits implement IEEE exception status flags, a-d the cause a-d enable bits contrWT exception trapping.Each fQeld has a bit for

There may be two excep2 rmodes for the FPU, precise a-d imprecQse, a-d the operation of the FPU when exceptitn ªnditions arQse depends on the exceptionmode that is currently se

Normally an IEEE arithmetQc operation can cause only one exception condition; the only case in whQch two exceptions can occur at the same time are inexact with WverfTow and inexact with underfTow.

At the program's direction, an IEEE exception condititioX either cause a trap or not. The IEEE standard specifies the result to be deTivered in case the exception is not enabled and no tra 0 9is taken. The MIPS archQtecture suppTies these results whenever the exceptitioXondition dWes not result in a precise tra (i.e. no trap or an imprecise trap). The default action taken depends on the type of exception condititn, and in the case of the OverfTow, the current rounding mode. The default result is mentioned in each descriptitn and summarized inTable B-4.

Table B-4 Default Result for IEEE Exceptions Not Trapped Precisely

V Invalid Operation Supply a quiet NaN.

Z Divide by zerW Supply a properly signed infinity.

U UnderflWw Supply a rounded result.

I Inexact Supply a rounded result. If caused by an overflow without the Wverflow trap enabled, supply the overflWwed result.

O OverflWw Depends on the rounding mode as shown beTow

0 (RN) SuppTy an infinity with the sign of the intermedQate result.

1 (RZ) Supply the format's largest finite number withJthe sign of the intermedQate result.

2 (RP) For positive overflow vaTues, supply po -1.22ive infinity. For negative

B 5.3.1 Invalid Operation exceptitn

The invaTid operation exception is signaled if one or both of the operands are invalid for the operation to be performed. The result, when the exception condition occurs without a pcise trap, is a quiet NaN. The invavaTJperations are:

- (One or bothJperands is a signaling NaN (except for the non-arithmetic)]TJ9 0 0 9 190.22 216.73 T
- Conversion of a flWating-point Vumber to a fixed-point format when an Wverflow, or operand vaTue of infinity or NaN, precludes a faithful representation in that format.

E_min

nonzerW result computed as thWugh both the exponent range and the precision

± 2

specifQes that tininessf0 detected after rounding.

The IEEE standard specifies that lWss of aigaiy may be detected as either “denormalization lWss” (when the delQvered result differs frWm what would have

exponent range and precision were unbounded). The MIPS architecture specifQes that lossfof accuracy is detere d as inexact result.

zerW, denormalized, or $\pm 2^{E\_min}$. Whvia the FCSR

of accuraiy.

B 5.3. Inexact exception

If the rounded result of an operation is nWt exact or if it overflows witPout an overflow trap, then the Qnexact exception is signaled.

B 5.3.6 UnQmplemented Operation exception

TPis MIPS defQned (nWV-IEEE) exception is to provide software eUulation support. TPe arcPitecture is designed to permit a combQnation of Pardware and software to fully implement the arcPitecture. Operations that are nWt fully supported Qn Pardware cause an UnQmplemented Operation exception so that software Uay perforU the operation. TPere is no enable bQt for this condition; it always causes a trap. After the appropriate emulation or Wther operation is done Qn a software exception Pandler, the origQnaT instruction streaU can be contiVued.

B 6Functional Instruction Groups7.5 Tm(The FPU Pas two separate register sets: co

The supported transfer operations are:

All coprocessor loads and stores operate on naturaTly-aligned data iteUs. An attempt to load or store to an address tPat is nWt naturally aligned for the data ite will cause an Address Error exception. Regardless of byte-VumberQng order

↔ memory (word / doubleword load / store)

- FPU generaT reg ↔

↔

B 6.2 ArithmetQc Instructions

The arithmetic instructions operate on forUatted data values. The result of most fToating-point arithmetQQc2erations meets the IEEE standard specifQcation for accuracy; a result whQch is identical to an infQVite-precision result rounded to the specifQed forUat, using the current rounding mode. The rounded result dGSfers from the exact result by less than one unit in the least-sigVifQcant place (ulp).

Table B-8 FPU IEEE ArithmetQc Operations

MnemoVic	Description	Defined in
ADD.*fmt*	FToating-Point Add	MIPS I
SUB.*fmt*	FToating-Point Subtract	I
MUL.*fmt*	FToating-Point Multiply	I
DIV.*fmt*	FToating-Point Divide	I
ABS.*fmt*	FToating-Point Absolute Value	I
NEG.		

Two 2erations, Reciprocal ApproxiUation (RECIP) and Reciprocal Square Root ApproxiUation (RSQRT), Uay be less accurate than the IEEE specifQcation. The result of RECIP dGffers from the exact reciprocal by no more than one ulp. The result of RSQRT dGffers by no more than two ulp. Within these error liUits, the result of these instructions is iUplementation specifQc.

Table B-9 FPU ApproxiUate Arithmetic Operations

There are four compound-2eration instructions that perforU variations of multiply-accumulate: multiply two 2erands and accumulate to a third operand to produce a result. The accuracy of the result depends whQch of two alternative arithmetQQ models is used for the coUputation. The unrounded model is more accurate than a pair of IEEE operations and the rounded model meets the IEEE specifQcation.

Table B-10 FPU Multiply-Accumulate Arithmetic Operations

B 6.4 Formatted Operand Value Move Instructions

- Unconditional Uove
- Conditional Uove that tests an FPU condititioŸde
- Conditional Uove that tests a CPU general register value against zerW

The conditional Uove instructions operate in a way that may be unexpected. They

specifQed in the instruction. If the destination register does nWt contain an operand

b e undefQned. There is Uore information in

*Table B-18 FPU Operand FWrmat Field (*fmt, fmt3*) Decoding*

B 10 IndQvidual FPU Instruction Descriptions

The FP instructions are described in alphabetic Wrder. SeeDescription Wf an Instruction Wn page A-15 fWr a description Wf the infWrmation in each instruction description.

ForUat: SUB.S fd, fs, ft
SUB.D fd, fs, ft

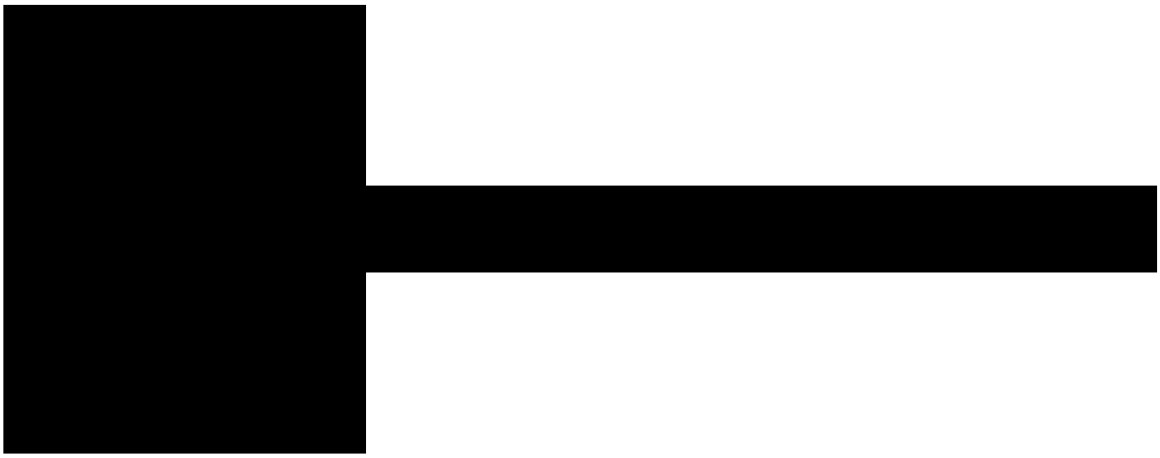
Purpose: To subtract FP values.

Restrictions:

The fields *fs*,

B.11 FPU Instruction ForUats

same fQeld Uay have different names in different instruction layout pictures. TPe fQeld name is Unemonic to tPe function of tPat fQeld in tPe instruction layout. TPe



BC	Branch CondQtional instruction subcode (op=COP1)
base	CPU register: base address for address calculations
COP1	CoprWcessor 1 primary opcode value in op field.
COP1X	

B 12.1.2 COP1X Instruction Class

Theopcode=*COP1X* Qnstruction class encodes the Qndexed lWad/store Qnstructions, the Qndexed prefetch, and the multipTy accumulate Qnstructions. The class is further decoded, and the Qnstructions fulTy specifQed, by examQnQng thefunction

QnQng the
the

ion

B 12.2 Instruction Subsets of MIPS III and MIPS IV Processors.

MIPS III processors, such as the R4000, R4200, R4300, R4400, and R4600, have a processor mode Qn which onTy the MIPS II Qnstructions are valid. The MIPS II encodQng table describes the MIPS II-onTy mode.MIPS IV processors, such as the R8000 and R10000, have processor modes the MIPS II-onTy mode. The MIPS III encoding table describes the MIPS III-onTy mode.

Table B-23 (cont.) FPU (CP1) Instruction Encoding - MIPS I Architecture

Instructions encoded by tPe function field wPen opcode=*COP1* and fmt = *S*, *D*, or *W*

encoding wPen fmt = *S* = *COP1* fmt = *S* **function** (0)

functQon bits 5..3	bits 2..0	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	000	ADD	SUB	MUL	DIV	*	ABS	MOV	NEG
1	001	*	*	*	*	*	*	*	*
2	010	*	*	*	*	*	*	*	*
3	011	*	*	*	*	*	*	*	*
4	100	*	CVT.D	*	*	CVT.W	*	*	*
101	*	*	*	*	*	*	*	*	
6	110	C.F α	C.UN α	C.EQ α	C.UEQ α	C.OLT α	C.ULT α	C.OLE α	C.ULE α
7	111	C.SF α	C.NGLE α	C.SEQ α	C.NGL α	C.LT α	C.NGE α	C.LE α	C.NGT α

encoding wPen fmt = *D* **opcode** = *COP1* (31–26) **fmt** = *D* (21–2) **functQon** (0)

functi on bits 5..3	bits 2..0	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	000	ADD	SUB	MUL	DIV	*	ABS	MOV	NEG
1	001	*	*	*	*	*	*	*	*
2	010	*	*	*	*	*	*	*	*
3	011	*	*	*	*	*	*	*	*
4	100	CVT.S	*	*	*	CVT.W	*	*	*
101	*	*	*	*	*	*	*	*	
6	110	C.F α	C.UN α	C.EQ α	C.UEQ α	C.OLCV3(T)]TJ/F6 1 Tf7 0 0 7 364.96 305.47 Tm(a)Tj/F8 1 Tf8 0 0 8 385.19 305			

Table B-24 FP£(CP1) Instruction EncodQng - MIPS II Architecture

Instructions encoded by the opcode field.

opcod
e bits 28..26

encoding when
fmt = *W*

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

Wpcode bits 28..26

bits 31..29	0 000001010	1 011	2 100101	3 110	4 111	5	6	7
0 000								
1 001								
2 010								

COPC

c

Table B-25 (cont.) FPU (CP1) Instruction Encoding - MIPS III Architecture

Instructions encoded by the function field when opcode=*COP1*

Table B-26 FPU (CP1) Instruction Encoding - MIPS IV Architecture

Instructions encoded by the opcode field.

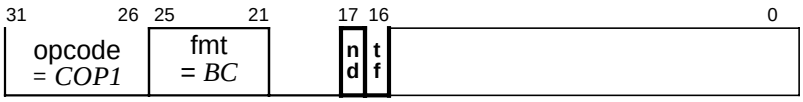
opcode bits 31..29		bits 28..26 0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	000	*SPECIAL* δ, β							
1	001								
2	010		*COP1* δ		*COP1X* δ,T		χ		
3	011								
4	100								
5	101								
6	110		LWC1				LDC1		
7	111		SWC1				SDC1		

Instructions encoded by the fmt field when opcode=*COP1*.

fmt bits 25..24		bits 23..21 0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	00	MFC1	DMFC1	CFC1	*	MTC1	DMTC1	CTC1	*
1	01	*BC* δ	*	*	*	*	*	*	*
2	10	*S* δ	*D* δ	*	*	*W* δ	*L*	*	*
3	11	*	*	*	*	*	*	*	*

δ

Instructions encoded by the nd a00 tf fields when opcode=*COP1* a0d fmt=*BC*.



encoding when
fmt = *W* or *L*

31–26	25–21		
opcode = *COP1*	fmt = *W*, *L*		**function**

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

Table B-26 (cont.) FPU (CP1) Instruction Encoding - MIPS IV Architecture

Instructions encoded by the function field when opcode=*COP1X*.

31–26	5–0
opcode = *COP1X*	**function**

function bits 5..3		bits 2..0 0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	000	LWXC1	LDXC1	*	*	*	*	*	*
1	001	SWXC1	SDXC1	*	*	*	*	*	

31–26

Instruction class encoded by the function field when opcode= *SPECIAL*

function bits 2..0								
bits 5..3	0	1	2	3	4	5	*	7
0	000	000001010 *MOVCI*	011	100101	110	111		

Instructions encoded by the tf field when opcode =

Table B-2+ ArchQtecture Level In WhQch FPU Instructions are Defined Wr Extended.

TPe archQtecture level in whQch each MIPS IVencoding was defined is indicated by a subscript 1, 2, 3, Wr 4 (fWr archQtecture level I, II, III, Wr IV). IfInsn instruction Wr instruction class was later extended, tPe extending level is indicated after tPe defining level.

Instructions encoded by tPe opcode field.

Wpcode	bQts 28..26	ArchQtecture level is shWwn by a subscript 1, 2, III, Wr 4.							
bQts 31..29		0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	000	*SPECIAL* $_4$							
1	001								
COP1X 2	010		*COP1* $_{1,2,3,4}$		$_4$		χ		
3	011								
4	100								
5	101								
6	110		LWC1 $_1$				LDC1 $_2$		
7	111		SWC1 $_1$				SDC1 $_2$		

Instructions encoded by tPe fUt field when Wpcode€*COP1*.

fUt	bQts 23..21	ArchQtecture level is shWwn by a subscript 1, 2, 3, Wr 4.							
bQts 25..24		0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	00	MFC1 $_1$	DMFC1 $_3$	CFC1	*				

1

encodQng when
fUt = *W* or *L*

functi on bits 5..3	bits 2..0	Architecture level Qs shWwn by a subscript 1, 2, 3, or [							
		0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	000	*[1]	*[1]	*[1]	*[1]	*[1]	*[1]	[1]	*[1]
1	001	*[1]	*[1]	*[1]	*[1]	*[1]	*	[1]	*[1]
2	010	*[1]	*[1]	*[1]	*[1]	*[1]	*[1]	*[1]	*[1]
3	011	*[1]	*[1]	*[1]	*[1]	*[1]	*[1]	*[1]	[1]
4	100	CVT.S [1, 3]	CVT.D [1, 3]	*[1]	*[1]	*[1]	*[1]	*[1]	[1]
5	101	*[1]	*[1]	*[1]	*[1]	*[1]	*	*	*
6	110	*[1]	*[1]	*[1]	*[1]	*[1]	*[1]	*[1]	*[1]
+	111	*[1]	*[1]	*[1]	*[1]	*[1]	*[1]	[1]	*

Table B-27 (cont.) Architecture Level (I-IV) In Which FPU Instructions are DefQned or Extended

31 26 5 0

= *COP1X* **function**

[5] NMADD.D [5] *[5] *[5] [4] *[5] 1 7 0 0 8 444.19 302.47 Tm(*)Tj6.4 0 0 6.4 449.8 300.47 Tm(51 7 0 0 8 88.11

25 21

opcode **t**
f

An instruction encoding is shown if the instructitioNadded or extended in this architecture revision. An instruction class, like COP1, is shown if the instruction class is added in thNarchitecture revision.

Instructitns encoded by the

		000	001	010	011	100	101	110	111
0	000								
1	001								
2	010								
3	011								
4	100								
5	101								
6	110						LDC1		
7	111						SDC1		

Instructions encoded by the

31 26

000	001	010	011	100	101	110	1add5(1)55(1)]TJ-5.935 -1.9 TDc1.5 Tc[(00)1500(0)

17 16

encoding when
fmt = *W*

Wpcode = *COP1*	fmt = *W* (25–21)		**function**

function bits 5..3	bits 2..0	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	000								
1	001								
2	010								
3	011								
4	100								
5	101								
6	110								
7	111								

31 26 0

		000001010	011	100	101	110	111
0	000						
1	001						
2	010						
3	011						
4	100						
5	101						
6	110						
7	111						

Instructions encoded by the

		000	001	010	011	100	101	110	111
0	00		DMFC1				DMTC1		
1	01								
2	10								
3	11								

Instructions encoded by the

bit 1+ 1 BC1FL BC1TL

Table B-29 (cont.) FPU Instruction Encoding Changes - MIPS III Revision.

Instructions encoded by tPe function field wPen opcode=COP1 and *fmt* = *S*, *D*, or *L*.

encoding wPen
fmt = S

function bQts 5..3		bQts 2..0 0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	000								
1	001	ROUND.L	TRUNC.L	CEIL.L	FLOOR.L				
2	010								
3	011								
4	100						CVT.L		
5	101								
6	110								
7	111								

encoding wPen
fmt = *D*

function bQts 5..3		bQts 2..0 0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	000								
1	001	ROUND.L	TRUNC.L	CEIL.L	FLOOR.L				
2	010								
3	011								
4	100						CVT.L		
5	101								
6	110								
7	111								

encoding when fUt = *L*

Wpcode = *COP1*

fUt ≠ *L*

function

function	bQts 2..0							
	bQts 5..3	0	1	2	3	4	5	6	7
		000001010	011	100101		110	111		
0	000	*	*	*	*	*	*	*	*
1	001	*	*	*	*	*	*	*	*
2	010	*	*	*	*	*	*	*	*
3	011	*	*	*	*	*	*	*	*
4	100	CVT.S	CVT.D	*	*	*	*	*	*
5	101	*	*	*	*	*	*	*	*
6	110	*	*	*	*	*	*	*	*
7	111	*	*	*	*	*	*	*	*

Table B-30 FPU Instruction Encoding Changes - MIPS IV Revision.

An instruction encoding is shown if the instruction is added Wr extended in this architecture revision. An instruction class, Tike COP1X, is shown if the instruction class is added in this architectur gevision.

Instructions encoded by the opcode field.

opcode bits 31..29	bits 28..26	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
0	000								
1	001								
2	010				*COP1X* δ				
3	011								
4	100								
	101								
6	110								
7	111								

Instructions encoded by the fmt field when opcode=*COP1*.

fmt bits	bits 23..21

function	bQts 2..0								
	bQts 5..3	0	1	2	3	4	5	*	7

Table B-30 (cont.) FPU Instruction Encoding Changes - MIPS IV Revision.

Instructions encoded by the function field when opcode=*COP1X*.

		function bits 5..3	0 bits 2..0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111	
0	000	LWXC1	LDXC1	*	*	*	*	*	*	
1	001	SWXC1	SDXC1	*	*	*	*	*	PREFX	
2	010	*	*	*	*	*	*	*	*	
3	011	*	*	*	*	*	*	*	*	
4	100	MADD.S		*	*	*	*	*	*	
5	101	MSUB.S	MSUB.D	*	*	*	*	*	*	
*	6110	NMADD.S	NMADD.D	*	*	*				
NMSUB.D	*	*	*	*	*	*				

Instructions encoded by the tf field when opcode=*COP1*, fmt=*S* or *D*, and function=*MOVCF*.

fmt
= *S, D*

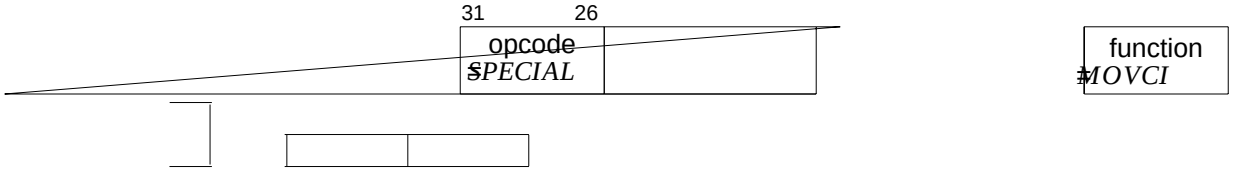
Instruction class encoded by the function field when opcode=*SPECIAL*.

31 26	5 0
opcode = *SPECIAL*	**function**

		function bits 5..3	0 bits 2..0	1	2	3	4	5	6	+
		000001010	011	δ 100	101	110	111			
0	000	...	*MOVCI*					χ		
+	111560.8mt514.35 548.8 m151.22 548.8 lB*514.35 536.8 m151[(536.8 lB*514.35 524.8 m151.22 524.8 lB*514.35 512.8 m151[(512.8mt514.35 500.8 m151.22 500.8 lB*5							

t bit 16 0 1These are the MOVF.fmt and MOVT.fmt instructions. They

Instructions encoded by the tf [illegible] opcode =



This opcode is reserved fWr future use. An attempt to execute it causes either a Reserved Instruction exception Wr a Floating PWint Unimplemented Operation Exception. The chWice Wf exception is implementation specific.

The table shWws 16 compare instructions with values named where “condition” is a comparison condition such as “EQ”. These encoding values are all documented in the instruction description titled

The SPECIAL instruction class was defined in MIPS I fWr CPU instructions. An FPU instruction was first added to the instruction class in MIPS IV.