Errata for MIPS R4000/R4400 Microprocessor User's Manual

This errata supplements the "MIPS R4000/R4400 User's Manual" by Joe Heinrich and published by Prentice Hall, Englewood, NJ, 1993

November 8, 1994

MIPS Technologies Inc.
2011 N Shoreline Blvd
PO Box 7311
Mountain View, CA 94039-7311

Page 5

The MIPS language suite does not support Ada, PLL, COBOL

Page 40

The section "Cycle Timing for Multiply and Divide Instructions" and the contents of "Table 2-2: Multiply/Divide Instruction Cycle Timing", should read as follows:

Any multiply instruction in the integer pipe is transferred to the multiplier and the remaining instructions continue through the pipe. The product from the multiply instruction is saved into HI and LO registers.

If the multiply is followed by MFLO or MFHI any time before the product is available, the pipeline interlockes till the product is available.

Table 2-2 gives the execution time for integer multiply and divide operations. The "Total Cycles" column gives the total number of cycles required to execute the instruction. The "Overlap" column gives the number of cycles that overlap other CPU operations; that is, the number of cycles required between the present instruction and a subsequent MFHI or MFLO without incurring an interlock. If this value is zero, the operation is not performed in parallel with any other CPU operation.

Instruction	Total Cycles	Overlap
mult	12	10
multu	12	10
dmult	20	18
dmultu	20	18
div	75	0
divu	75	0
ddiv	139	0
ddivu	139	0

Page 41

The next to last paragraph: "If a conditional branch is not taken, ... is nullified." should be replaced by "If a conditional branch, in any branch likely instruction is not taken, the instruction in the branch likely delay slot is nullified."

Page 60

Add the following at the end of the paragraph:

Caution: R4000 always had a "strongly ordered" execution; however, with the addition of the "uncached store buffer" in the R4400, there is a potential of out of order execution to occur. (described in the section Strongly Ordered in Chapter 11, and Uncached Loads or Stores in Chapter 12).

Page 62

In subsection "Multiple Matches", 2nd. line; "...and the TLB can be disabled."
should be:
"... and the TLB may be disabled if more than several entries match to prevent permanent damage

to the part."

Page 67

Figure 4-4: Replace the footnote(*) with the following text:
The R4000 uses 64-bit address internally. In 32-bit mode, a valid address must be a 32-bit signed number (bits 63..32 == bit 31). In normal operation, it is not possible for a 32-bit user program to produce invalid addresses. If a kernel error puts values that are not 32-bit signed numbers into the 64-bit registers, then the user mode program can generate an invalid address. Use of invalid address produces undefined results.

Page 68

In the third para. change the two bullets to read as follows:
- when UX=0, 32-bit *useg* space is selected, TLB misses are handled by the 32-bit TLB refill exception handler.
- when UX=1, 64-bit *xuseg* space is selected, TLB misses are handled by the 64-bit XTLB refill exception handler.

Page 69

1) In "32-bit User Mode" remove the third paragraph which says, "In 32-bit..... misses".
2) In "64-bit User Mode" remove the third paragraph which says, "The extended..... misses".

Page 70

1) In the third para. change the two bullets to read as follows:
- when SX=0, 32-bit supervisor space is selected, TLB misses are handled by the 32-bit TLB refill exception handler.
- when SX=1, 64-bit supervisor space is selected, TLB misses are handled by the 64-bit XTLB refill exception handler.

2) Figure 4-5: Replace the footnote(*) with the following text:
The R4000 uses 64-bit address internally. In 32-bit mode, a valid address must be a 32-bit signed number (bits 63..32 == bit 31). Use of invalid address produces undefined results. In normal operation, a 32-bit supervisor program cannot create an invalid address in a register through arithmetic operations. 32-bit mode supervisor programs must refrain from creating addresses using base register + offset calculations that produce 32-bit two's complement overflow. There are two prohibited cases:
- offset with bit 15==0 and base register with bit 31==0; but, (base register + offset)'s bit 31==1
- offset with bit 15==1 and base register with bit 31==1; but, (base register + offset)'s bit 31==0.

Using this invalid address procuces an undefined result.

Page 73

In the second para. change the two bullets to read as follows:
- when KX=0, 32-bit Kernel space is selected, TLB misses are handled by the 32-bit TLB refill exception handler.

• when KX=1, 64-bit Kernel space is selected, TLB misses are handled by the 64-bit XTLB refill exception handler.

Page 74

Figure 4-6: Replace the footnote(*) with the following text:

The R4000 uses 64-bit address internally. In 32-bit mode, a valid address must be a 32-bit signed number (bits 63..32 == bit 31). Use of invalid address produces undefined results. Kernel mode is permitted to use 64 bit instructions in 32-bit mode, and it must refrain from using invalid addresses. 32-bit kernel programs must refrain from creating addresses using base register + offset calculations that produce 32-bit two's complement overflow. There are two prohibited cases:

• offset with bit 15==0 and base register with bit 31==0; but, (base register + offset)'s bit 31==1
• offset with bit 15==1 and base register with bit 31==1; but, (base register + offset)'s bit 31==0.

Page 75

In the section titled "32-bit Kernel Mode, User Space (kuseg), add the following para:

"When ERL=1 in the Status register, the user address region becomes a 2^{31}-byte unmapped (i.e., mapped direct to physical address), uncached address space. See the description of the Cache Error Exception in Chapter 5 for more detail."

Page 76

In the first paragraph, third line, replace the following line "...it is the current 2^{29} -byte (512-Mbyte) kernel physical space." with "it is the 2^{29} -byte (512-Mbyte) kernel physical space."

Page 77

1) Table 4-4: For Address Bit value of "A(63:62) = 11_2", Segment Size should be 2^{40}-2^{31} (instead of 2^{44}).

2) Replace the last para. with the following:

"When ERL=1 in the Status register, the user address region becomes a 2^{31}-byte unmapped (i.e., mapped direct to physical address), uncached address space. See the description of the Cache Error Exception in Chapter 5 for more detail."

Page 79

In the section, **64-bit Kernel Mode, Kernel Space (*xkseg*),** make following change:

- kernel virtual space, *xkseg*, the current kernel supervisor space;...

to:

- kernel virtual space, xkseg, the current kernel virtual space;....

Page 81

Figure 4-8: For 64bit mode, the bit numbers in the top frame should be from 192 to 255 (instead of from 96 to 255)

Page 82

The definition of "FILL" should read as follows:
Fill......... Reserved. Returns zero when read; ignored when written.

Page 91

1) In Table 4-12: For the description of "CM" delete the phrase. "This bit is set... 0 on a Soft Reset."
2) In Table 4-12: For the description of "EW", replace "1->Reserved" to "1, 2 & 3 -> Reserved"
3) In Table 4-12: For the description of "EP", replace "7->DDxxxxx" to "7->DDxxxxxx"' and "2 Doublewords every 7 cycles" to "2 Doublewords every 8 cycles".

Page 94

The field for TagHi register in figure 4-18 and 4-19, should be "undefined*" instead of "0".
* Note: The TagHi register in R4000/4400 should not be used. The consequence of the reading and writing of this register is undefined.

Page 103

Delete the 2nd para: "The processor does not write to the BadVAddr..... is set to a 1".
(The flowchart on pg. 145 & 147 needs to be updated accordingly)

Page 108

Change: Table 5-4: SR Bit: 1-> Indicates soft reset or NMI has occurred.
to: Table 5-4: SR Bit: 1-> Indicates warm reset or NMI has caused SoftReset Exception.

Page 111

Figure 5-7: The field Cause[15..8] should be labeled as IP7.... IP0, respectively.

Page 112

In the last line of the text: Change "Table 5-8" to "Figure 5-8"

Page 117

In Table 5-10, description of "EW" field, remove the following bullet:
- any external request, including intervention and snoop

Page 119

In the section titled "Exception Types",
1) The 2nd. para. should read as: "When the EXL bit in the Status register is 0, either User, Supervisor, or Kernel operating mode is specified......"
2) The last two sentences in the 3rd para should read as:
"After saving the appropriate state, the exception handler typically changes KSU to Kernel mode and resets the EXL bit back to 0. When restoring the state and restarting, the handler restores

the previous value of the KSU field and sets the EXL bit back to 1.”

Page 120

1)In figure 5-14: Reset Exception processing -

a) Change: Config <- CM||.........001|| 001|| undef^{6}; to: Config <- CM||.......... IC || DC || undef^{6}

b) Change “ErrorPC <- PC” to “ErrorPC <- RestartPC” and add this footnote “For the definition of RestartPC see section 5.1”

c) add the following lines after the line “ErrorPC <- RestartPC”:

```
if R4400
    CacheErr <- undef^8 || 0 || undef^23 /* Set EW bit to 0 */
endif
```

d) Change: PC <- 0xBFC0 0000; to: PC <- 0xFFFF FFFF BFC0 0000

2)In figure 5-15: Cache Error Exception Processing -

a) Change: CacheErr <- ....; to

```
if R4000
    CacheErr <- ER || EC || ED || ET || ES || EE || EB || EI || 0^2 || SIdx || PIdx
else
    CacheErr <- ER || EC || ED || ET || ES || EE || EB || EI || EW || 0 || SIdx || PIdx /* for R4400 */
endif
```

b) Change “SR22 =1” to “SR$_{22}$ = 1"

c) Change: PC <- 0xBFC0 0200 + 0x100; to: PC <- 0xFFFF FFFF BFC0 0200 + 0x100

d) Change: PC <- 0xA000 0000 + 0x100; to: PC <- 0xFFFF FFFF A000 0000 + 0x100

Page 121

1) In figure 5-16: Soft Reset and NMI Exception Process

a) Change “ErrorPC <- PC” to “ErrorPC <- RestartPC”

b) After the line “ErrorPC <- RestartPC” add the following:

```
if R4400
    CacheErr <- CacheErr_31:24 || 0 || CacheErr_22:0
endif
```

b) Change: PC <- 0xBFC0 0000; to: PC <- 0xFFFF FFFF BFC0 0000

2) In figure 5-17: General Exception Process, replace the entire box with the following:

```
T:  if SR1 = 0 then            */ if not EXL /*
            EPC ← RestartPC           */ If the instruction is in a branch delay slot, /*
                                      */ RestartPC holds the value of PC-4,  /*
                                      */ otherwise RestartPC = PC /*
            Cause ← BD || 0 || CE || 0^12 || Cause15:8 || 0 || ExcCode || 0^2
            if TLBrefill then         vector ← 0x000
            elseif XTLBrefill then    vector ← 0x080
            else   */ not a miss /*   vector ← 0x180
    else
            Cause ← Cause31 || 0 || CE || 0^12 || Cause15:8 || 0 || ExcCode || 0^2
            vector ← 0x180
    endif
    SR ← SR31:2 || 1 || SR0   */  EXL /*
    if SR22 = 1 then
       PC ← 0xFFFF FFFF BFC0 0200 + vector
    else
       PC ← 0xFFFF FFFF 8000 0000 + vector
    endif
```

Flowchart on page 145 and 147 need to be updated accordinly.

Page 122

Replace the next-to-last paragraph with:

When BEV=0, the vector base for the Cache Error exception is an uncached, unmapped address in kseg1 (0xA000 0000) in 32-bit mode, 0xFFFF FFFF A000 0000 in 64-bit mode). Handler for the cache error exception never executes from mapped or cached addresses even during normal system operation.

Page 124

Add the following bullets after the bullet "The Wired register is initialized to 0.":

- The EW bit in the CacheErr register is cleared (R4400 only)
- Config register is initialized with the boot mode bits read from the serial input.

Page125

1) Soft Reset Exception - Processing (3rd. line)
Change: When a Soft Reset occurs, the SR bit of the Status register is set....
to:
When SoftReset Exception occurs, the SR bit of the Status register is set to distinguish this exception from a Reset exception. However, there is no indication by the processor to differentiate between NMI and Soft Reset. If necessary, they need to be distinguish by the hardware outside the processor.
2) Add this bullet after the bullet "BEV bit of the Status register, which is set to 1.

- The EW bit in the CacheErr register is cleared (for R4400 only)
- *TS* bit of the *Status* register, which is set to 0
- PC is set to the reset vector 0xFFFF FFFF BFC0 0000

Page 126

Add this bullet after the bullet "BEV bit of the Status register, which is set to 1.

- The EW bit in the CacheErr register is cleared (for R4400 only).
- *TS* bit of the *Status* register, which is set to 0
- PC is set to the reset vector 0xFFFF FFFF BFC0 0000

Page 127

In the Processing section, 2nd para, 1st. sentence, change:
When this processing.....not properly alligned or referenced protected address space.
to:
When this processing.....not properly alligned or that referenced protected address space.

Page 132

1) Replace the first para. with:
The Cache Error exception occurs when either a secondary cache ECC error or primary cache parity error or SysAD bus parity/ECC error occurs and error detection is enabled. This exception is not maskable but the error detection can be disabled by setting either the DE bit or the ERL bit in the Status register.
2) Replace the first sentence of the Processing section with:
The processor sets the ERL bit in the *Status* register, saves the exception restart address in *ErrorEPC* register, records information about the error in the *CacheErr* register, and then transfers to a special vector that is always in uncached space.
3) In the beginning of the Servicing section add the following para:
Unlike other exception conditions, cache errors cannot be avoided while operating at exception level, so Cache Error exceptions must be handled from exception level. Any general register used by the handler must be saved before use and restored before return; this includes the registers available to regular exception handlers without save/restore. When *ERL*=1 in the *Status* register, the user address region becomes a 2^{31}-byte uncached space mapped directly to physical addresses, allowing the Cache Error handler to save registers to memory without using a register to construct the address. The handler can save and restore registers using operating system-reserved locations in low physical memory by using *R0* as the base register for load and store instructions.
4) In the end of the Servicing section add following para:
The exception handler cannot be interrupted by another Cache Error exception because error detection is disabled while *ERL* = 1, so the handler should avoid actions which might cause an unnoticed cache error. The R4400 (but not R4000) implements the *EW* bit in the *CacheErr* register to record a nonrecoverable error occurring while *ERL* = 1

Page 133

1) A Virtual Coherency exception occurs when one of the following conditions is true:
should be replaced by
A Virtual Coherency exception occurs when all of the following conditions are true:
2) In the "Servicing" section, replace the first paragraph with the following:
"Using the CACHE instructions, the primary cache lines at both the previous and the new virtual index should be invalidated[1] (and written back, if necessary), the PIdx field of the

secondary cache should be written with the new virtual index. Once this is completed, the program is continued.

[1]Foot Note: When the cache miss occurs, the processor refills the primary cache line at the present virtual index before taking an exception.

Page 140

In the Cause section, 2nd bullet should read:

- CP0 instuction, when.........process executes in User or Supervisor mode

Page 145

1) In figure 5-18, typo in the note for BADVA (middle of the page).
"Note: not set it if Bus Error Exception"
should read as
"Note: not set if it is a Bus Error Exception"
2) Change "Set BadVA" according to errata on pg. 103; and setting of "Cause 31 (BD)" according to the change on page 121.

Page 147

1) Change "Set BadVA" according to errata on pg. 103; and setting of "Cause 31 (BD)" according to the change on page 121.
2) In the flowchart, if BEV = 0, the "PC <- -x8000 0200 + Vec.Off" should be "PC <- -x8000 0000 + Vec.Off"

Page 149

In the flowchart, if BEV = 0, the "PC <- xA000 0100 + 100" should be "PC <- xA000 0000 + 100"

Page 162

1) Change in the first sentence: "When a floating-point exception ....only states affected are the Cause and flag bits."
to: "When a floating-point exception ....only state affected is the Cause bit."
2) Remove the last sentence of the 1st. para. - "Exceptions caused by an ...... the Cause field."
3) In the section "Enable Bits"- 1st sentence in the 3rd. paragraph should be as follows:
Before returning from a floating-point exception, software must first clear the Cause bits, using CTC1 instruction, to prevent a repeat of the interrupt.
4) In the section "Flag Bits", remove the 1st. para.- "The appropriate Flag ...... user handler."
5) At the end of the page add the following paragraph :
"When a FP exception is taken, the flag bits are not set by the hardware; FP exception software is responsible for setting these bits before invoking a user handler."

Page 168

In Table 6-10: Add the following instructions for the 64bit operations:
CVT.L.fmt, ROUND.L.fmt, TRUNC.L.fmt, CEIL.L.fmt, FLOOR.L.fmt. All the functions with

".W" are for 32-bit Fixed Point and that with ".L" are 64-bit Fixed Point

Page 169

In the section titled "Transfers between FPU and Memory" - 3rd. line should read as:
- Load Word To Coprocessor 1 (LWC1) or Store Word From... (instead of "...Store Word To..)

Page 170

In the section "Floating-Point Computaional Instructions", at the end of the first bullet remove "and square root operations." and added it at the end of the second bullet.

Page 174

Replace the Table 6-14 with the following:

Operation	Pipeline Cycles				Operation	Pipeline Cycles			
	S	D	W	L		S	D	W	L
ADD.fmt	4	4	(a)	(a)	**CVT.[W,L].fmt**	4	4	(a)	(a)
SUB.fmt	4	4	(a)	(a)	**C.fmt.cond**	3	3	(a)	(a)
MUL.fmt	7	8	(a)	(a)	**BC1T**	(b)	1	(b)	(b)
DIV.fmt	23	36	(a)	(a)	**BC1F**	(b)	1	(b)	(b)
SQRT.fmt	54	112	(a)	(a)	**BC1TL**	(b)	1	(b)	(b)
ABS.fmt	2	2	(a)	(a)	**BC1FL**	(b)	1	(b)	(b)
MOV.fmt	1	1	(a)	(a)	**LWC1**	(b)	3	(b)	(b)
NEG.fmt	2	2	(a)	(a)	**SWC1**	(b)	1	(b)	(b)
ROUND.[W,L].fmt	4	4	(a)	(a)	**LDC1**	(b)	3	(b)	(b)
TRUNC.[W,L].fmt	4	4	(a)	(a)	**SDC1**	(b)	1	(b)	(b)
CEIL.[W,L].fmt	4	4	(a)	(a)	**MTC1**	(b)	3	(b)	(b)
FLOOR.[W,L].fmt	4	4	(a)	(a)	**MFC1**	(b)	3	(b)	(b)
CVT.S.fmt	(a)	4	6	7	**CTC1**	(b)	3	(b)	(b)
CVT.D.fmt	2	(a)	5	4	**CFC1**	(b)	2	(b)	(b)

(a) .... These operations are illegal.
(b) .... These operations are undefined.

Page 179

In figure 6-16, for I5 and in figure 6-17, for I4 "the legal issue?" should be "No*", with the following footnote:
* While there is no resourse conflict in issuing this CMP.[S,D] instruction, the hardware does not allow it.

Page 181

In Table 6-16, change the Instruction type CVT.W.[S,D] or To CVT.[W,L].[S,D] or

ROUND.W.[S,D] or TRUNC.W.[S,D] or CEIL.W.[S,D] or FLOOR.W.[S,D]	ROUND.[W,L].[S,D] or TRUNC.[W,L].[S,D] or CEIL.[W,L].[S,D] or FLOOR.[W,L].[S,D]

Page 182 to 185

Replace all the contents of the "Resource Scheduling Rules" with the following:

Resource Scheduling Rules

The FPU Resource Scheduler issues instructions while adhering to the rules described below. These scheduling rules optimize op unit executions; if the rules are not followed, the hardware interlocks to guarantee correct operation.

DIV.[S,D] can start only when all of the following conditions are met in the RF stage:

- The *divider* is either idle, or in its second-to-last execution cycle.
- The *adder* is either idle, or in its second-to-last execution cycle.
- The *multiplier* is either idle, or in its second-to-last execution cycle.

Idle means an operation unit—adder, multiplier or divider—is either not processing any instruction, or is currently in its last execution cycle completing an instruction.

MUL.[S,D] can start only when all of the following conditions are met in the RF stage:

- The *multiplier* is one of the following:
  - idle, or in its second-to-last execution cycle.
  - not within the first two execution cycles (EX, EX+1) if the most recent instruction in the multiplier pipe is MUL.S
  - not within the first three execution cycles (EX..EX+2) if the most recent instruction in the multiplier pipe is MUL.D
- The *adder* is one of the following:
  - idle, or in its second-to-last execution cycle.
  - not processing the first execution cycle (EX) of CVT.S.L
- The adder is not processing a square root instruction
- The *divider* is one of the following:
  - idle, or in its second-to-last execution cycle.
  - in the first 8 execution cycles (EX..EX+7) of a DIV.S in the first 21 execution cycles, except for the second execution cycle, (cycles EX, EX+2..EX+20) of a DIV.D)

SQRT.[S,D] can start only when all of the following conditions are met in the RFstage:

- The *divider* is either idle, or in its second-to-last execution cycle.
- The *adder* is either idle, or in its second-to-last execution cycle.
- The *multiplier* is either idle, or in its second-to-last execution cycle.

CVT.fmt, **NEG.[S,D]** or **ABS.[S,D]** instructions can only start when all of the following conditions are met in the RF stage:

- The *adder* is either idle, or in its second-to-last execution cycle.
- The *multiplier* is either idle, or in its second-to-last execution cycle.
- The *divider* is one of the following:

- idle, or in its second-to-last execution cycle.
- in the third through eighth execution cycle (EX+2..EX+7) of a DIV.S
- in the third through twenty-first execution cycle (EX+2..EX+20) of a DIV.D

ADD.[S,D], **SUB.[S,D]** or **C.COND.[S,D]** can only start when all of the following conditions are met in the RF stage:

- The *adder* is either idle, or in its second-to-last execution cycle.
- The *multiplier* is one of the following:
  - idle, or in its second-to-last execution cycle.
  - not in the third or fourth execution cycles (EX+2..EX+3) if the most recent instruction in the multiplier pipe is MUL.S
  - not in the fourth or fifth execution cycles (EX+3..EX+4) if the most recent instruction in the multiplier pipe is MUL.D
- The *divider* is one of the following:
  - idle, or in its second-to-last execution cycle.
  - in the third through eighth execution cycle (EX+2..EX+7) of a DIV.S
  - in the third through twenty-first execution cycle (EX+2..EX+20) of a DIV.D

<u>Page 189</u>

In the first paragraph, replace "When an exception occurs, both the corresponding Cause and Flag bits are set. If the corresponding Enable bit is set, the FPU generates....taken.

To: "When an exception occurs, the corresponding Cause bit is set. If the corresponding Enable bit is not set, the Flag bit is also set. If the corresponding Enable bit is set, the Flag bit is not set, the FPU generates ....taken.

<u>Page 190</u>

In Table 7-1, the entries for Rounding Mode and Default action in the case of the field U (Underflow exception) should be as follows:

Rounding Mode	Default action
RN	Modify underflow values to 0 with the sign of the intermediate result
RZ	Modify underflow values to 0 with the sign of the intermediate result
RP	Modify positive underflows to the format's smallest positive finite number; modify negative underflows to -0
RM	Modify negative underflows to the format's smallest negative finite number; modify positive underflows to 0

Page 191

Replace Table 7-2 *FPU Exception-Causing Conditions* with the one shown below:

FPA Internal Result	IEEE Standard 754	Trap Enable	Trap Disable	Notes
Inexact result	I	I	I	Loss of accuracy
Exponent overflow	O,I[a]	O,I	O,I	Normalized exponent > E_{max}
Division by zero	Z	Z	Z	Zero is (exponent = E_{min}-1, mantissa = 0)
Overflow on convert	V	E	E	Source out of integer range
Signaling NaN source	V	V	V	
Invalid operation	V	V	V	0/0, etc.
Exponent underflow	U	E	UI[b]	Normalized exponent < E_{min}
Denormalized or QNaN	None	E	E	Denormalized is (exponent = E_{min}-1 and mantissa <> 0)

a. The IEEE Standard 754 specifies an inexact exception on overflow only if the overflow trap is disabled.
b. Exponent underflow sets the U and I *Cause* bits if both the U and I *Enable* bits are not set and the *FS* bit is set; otherwise exponent underflow sets the E *Cause* bit.

Page 192

In the Section "Inexact Exception (I)", the first sentence: "The FPU generates the ..... overflows." should be replaced by :
" The FPU generates the Inexact exception if:

- the rounded result of an operation is not exact, OR
- the rounded result of an operation overflows, OR
- the rounded result of an operation underflows and both the Underflow and Inexact Enable bits are not set and the FS bit is set"

Page 193

1) Remove the fourth bullet: "Conversion of a floating-point ....in that format."
2) Add this bullet: " Comparison or a "Convert from floating-point" operation on a signaling NaN."
3) In the sub-section "Trap Disabled Results:", replace the description with the following sentence:
"A Quiet NaN is delivered to the destination register if no other software trap occurs."

Page 195

1) Replace the description of the section "Trap Enabled Results:" with the following sentence:
"If Underflow or Inexact traps are enabled, or if the FS bit is not set, then an unimplemented exception (E) is generated, and the result register is not modified."
2) Replace the description of the section "Trap Disabled Results:" with the following sentence:
"If Underflow and Inexact traps are not enabled and the FS bit is set, the result is determined by the rounding mode and the sign of the intermediate result (as listed in Table 7.1)."

Page 196

1) The first four bullets should read as follows:
- Denormalized operand, except for Compare instruction
- Quiet Not a Number operand, except for Compare instruction
- Denormalized result or Underflow, when either Underflow or Inexact Enable bits are set or the FS bit is not set.

Page 202

In the Description column of SysCmdP add the following:
When the System Interface is set to parity mode, the processor indicates a secondary cache ECC error by corrupting the state of the SysCmdP signal.

Page 216

1) Item 4 in the section "Power-on Reset" should read as follows:
The de-assertion of ColdReset* will synchronize the rising edge of the SClock and TClock with the rising edge of the next MasterClock (This would align the SClock, TClock and the RClock of all the processors in the multiprocessor system). However, these clocks are guaranteed to stabilize 64MasterClock cycles after the ColdReset* is de-asserted.
(TClock & RClock in Figure 9-1 & Figure 9-2 should also be changed to reflect the above)
2) At the end of item 2 in the section "Power-on Reset" please add:
Note that when the JTAG is not used, the processor only resets properly if the JTCK is tied low at rising VCCOk. If the JTAG is used, then clocking JTCK is OK even during power up.

Page 217

1) Change the following sentence:
"The master clock output, **MasterOut,** generates any reset-related signals for the processor that must be synchronous with **MasterClock"**
to: "The master clock output, **MasterOut**, can be used to generate any reset-related signals for the processor that must be synchronous with MasterClock". Note that the **MasterOut** is undefined during power up (see figure 9-1); so, the reset logic during power up should not depend on **MasterOut**.

Page 218

Item #1 has a typo- "...The MasterClock output ais.." should be "The MasterClock output is..."

Page 222

When serial Bit 8 (EISpltMd) is set to 1: Replace "Reserved" to "Secondary Cache Split".

Page 224

For Serial Bit 33-36, make the following substitution:

From	To
0-2	0-1

3-15	2-15
Number ...Min 3, Max 15	Number...Min 2, Max 15

Replace the following information for Serial Bit 41:

Serial Bit	Value	Mode Setting
41	0	NoMPmode OFF: After a SCache miss, the existing valid line in the SCache is Invalidated (after the writeback, if needed)
	1	NoMPmode ON: For the R4000/4400SC part, after a SCache miss, the existing valid line in the SCache, is not Invalidated to improve performance.

Page 229

At the end of the section titled SyncIn/SyncOut, add the following:
The delay allowed between the SyncOut and SyncIn must be no more than [1/2*MasterClock cycle - 2nS]

Page 230

1) TClock section - 2nd. paragraph should read as follows:
TClock has the same frequency as the SClock and the edges of TClock aligns precisely with the edges of SClock when SyncIn is shorted to SyncOut. When any delay is added between SyncIn and SyncOut, the TClock at the pins will lead SClock and thus MasterClock by the same delay. If the delay between SyncIn and SyncOut is matched to any external delay between the TClock at the processor and the TClock at some external logic, then the TClock at the external logic end will now align to the SClock and the MasterClock.
2) RClock section - 1st. paragraph should read as follows:
The external agent uses RClock (receive clock) to clock its input registers. The processor generates RClock at the same frequency as TClock; but, RClock always leads TClock by 25 percent of the TClock cycle time. The relationship between RClock and TClock is independent of the delay between SyncIn and SyncOut.

Page 254

Add the following at the end of the first para:
Note: Regardless of the secondary cache size the processor always uses bits 35..17 of the physical address from the secondary cache; thus, it is important to initialize all bits of the STag with valid physical address regardless of the size of the secondary cache.

Page 259

Change the following sentence in the last paragraph, 3rd line
"However, there are exceptions. For example, the processor retains...."
to: "However, there is one exception. The processor retains..."

Page 270

For the section "Update" add the following at the end of the paragraph:

Note: If there is an update to a line which is in the Primary Instruction Cache(PICache) then the line in the SCache will be updated; but the line in the PICache will be invalidated.

Page 289

In section LL and SC:; 2nd. para. The first bullet for "The link is broken in the following circumstances:" should be as follows:

- if any external request (Invalidate, Snoop or Intervention) changes the state of the line containing the lock-variable to invalid.

Add the following bullet to the two existing ones:

- An external update to the cache line containing the lock-variable.

Page 319

In Table 12-3, for Page Attribute "Exclusive" the Processor Configuration should be only R4000MC (remove "R4000SC)

Page327-331

There is a discontinuity in numbering the figures; figure 12-16 does not exist.

Page 332

Last line should read: "..... performs another uncompelled change to slave state, asserting **Release***, after processing the external request.

Page 335

In the section "Processor Invalidate and Update Request Protocol", the first paragraph should be as follows:

1. Processor invalidate request or update request protocols are the same as a coherent word write request, except for the following differences:
   - Write request can be controlled by **WrRdy*** while the invalidate or update requests can be controlled by **RdRdy*.**
   - Unlike the write or update, the single data cycle is not used by the invalidate request protocol.
2. At the end of the 2nd. paragraph add the following:

The processor's pipeline stalls until:

a) either IvdAck* or IvdErr* is asserted - In the former case, the invalidate is considered successful and the processor continues. In the latter case, a bus error exception is taken.

OR

b) external agent sends an intervention or a snoop or an invalidate request with the cancellation bit (SysCmd[4]) bit =0 - In this case, the processor will complete the external request and then re-execute the instruction that caused the processor to send the invalidate or update request that was cancelled.

If the external request is sent with SysCmd(4)=1, indicating no cancellation, the processor, after responding to the external request, stalls again until one of the two conditions described above terminate the processor's invalidate or update request.

Page 338

At the end of the sentence in the ninth line: "Figures 12-24 through 12-27 illustrate .... below.
Add the following sentence: "All the waveforms are shown with zero delay from the SClock's rising edge; i.e. outputs are driven by the processor at the corresponding rising edge and inputs are driven by the external logic at the corresponding rising edge of the SClock.
Also at the end of the paragraph describing Figure 12-24: "Figure 12-24 illustrates .... WrRdy*.
Add the following sentence: "In this example, the WrRdy* must be deasserted in cycle 5 (sampled by the processor at the rising edge of cycle 6) to delay the issue cycle that would have occurred in cycle 7. Assertion of WrRdy* in cycle 8 (sampled by the processor at the rising edge of cycle 9) will cause the Write Request to be issued in cycle 10.

Page 342

In the section: External Arbitration Protocol change the following sentence:
"4. The external agent must begin driving the SysAD bus and the SysCmd bus two cycles after the......"
to: "4. The external agent must begin driving the SysAD bus and the SysCmd bus at least two cycles after the......"

Page 368

Table 12-16: Add the following footnote on the row SysCmd(2)/1:
"Cache line retained*"
Footnote: *The only case the processor will set this bit is if "Hit Writeback" causes the processor to do the Write Request (see page 259 for more description).

Page 370 , 371 & 372

In Table 12-20, 12-21 and 12-23, SysCmd(4) bit is used for Processor Unacknowledged Invalidate or Update Cancellation. So SysCmd(4) = 0 describes Invalidate or Update Cancelled.

Page 374

Figure 12-50: Bit 4 is only reserved for the processor data identifier; for the external data identifier, it is used to indicate whether the processor should check for error or not.

Page 375

In the paragraph that defines the SysCmd(5) bit, add the following after the first line:
However, in the case of a block response, the entire line must be delivered to the processor even if one of the double word has an error.

Page 382

Table 13-1: Change the number of cycles for t_{Rd2Cyc} from "3-15" to "2-15"

Page 384

For figures 13-2, 13-3, 13-4, 13-5:
SCAPar needs to be included. This signal will change one PCycle after the cycle in which the SCAddr(17..1) or SCAddr(0) changes.

Page 390

Figure 14-1: There should not be a serial connection through the "IC Package Pins (no connections between outermost squares) only through the "Boundary-scan cells"

Page 391 & 395

In figure 14-2 & 14-6, Boundary Scan Register should be from 1 to 319 (instead of 0 to 318)

Page 394

Figure 14-5: The numbering
"318 317 316 315...0"
should be replaced with numbers
"319 318 317 316...1"
Similarly, in the paragraphs that follows: change to "OE3 (bit 319)", "OE2 (bit 318)...." and "OE1 (bit 317)...."

Page 405

The first sentence of the second paragraph should read as follows:
The NMI* pin is latched by the rising edge of the SClock; however, as stated on pg. 126, NMI exception occurs in response to the falling edge of NMI* pin and it is not level sensitive.

Page 410

Third bullet needs a footnote as shown below: "It detects 3- or 4-bit errors within a nibble*.
* Nibble is defined by the group of four bits shown in figure 16-1 separated by the vertical lines.

Page 412

1) In the section "System Interface" add the following at the end of the first para:
...directly without changing the bits, to the System interface if the system interface is set to the "ECC mode". However, if the system interface is set to the "parity mode", the processor indicates a SCache ECC error to the system by corrupting the SysCmdP bit.
(This point should be reflected in section "System Interface Command Bus" on pg 413, in the last row/column of Table 16-2, pg. 427 and in SysCmdP definition on pg 202)
2) In the section "System Interface" 2nd. para, 2nd line - change the "NChck bit" to "SysCmd<4> bit"

Page 417

Item 2.: Change the last row in the "Column 6 ECC" and "Parity (even)" columns from 0's to 1's.

Page A-33

In the "Operation:" box, for 64bit info:
change: "condition <-- ($GPR[rs]_{63}$ = 1) and (GPR[rs] = 0^{64})"
to: "condition <-- ($GPR[rs]_{63}$ = 1) or (GPR[rs] = 0^{64})"

Page A-34

1) In the "Operation:" box, for 32bit info:
change: "condition <-- ($GPR[rs]_{31}$ = 1) and (GPR[rs] = 0^{32})"
to: "condition <-- ($GPR[rs]_{31}$ = 1) or (GPR[rs] = 0^{32})"
2) In the "Operation:" box, for 64-bit info:
change: "condition <-- ($GPR[rs]_{63}$ = 1) and (GPR[rs] = 0^{64})"
to : " "condition <-- ($GPR[rs]_{63}$ = 1) or (GPR[rs] = 0^{64})"

Page A-44

1) Cache Ops - *Index WriteBack invalidate SD*
Change the entire description to:
Examine the cache state of the secondary data cache block at the index specified by the physical address. If the block is dirty (state is Dirty Exclusive or Shared), then writeback the data to memory. Like all secondary writebacks, the operation will write any modified data for the addresses from the primary data cache. The address to write is taken form the secondary cache tag. The PIdx field of the secondary tag is used to determine the locations in the primaries to check for matching primary blocks. In all cases, set the state of the secondary cache block and all matching primary sub-blocks to Invalid. No Invalidate is sent on the R4000's system interface.
2) Cache Ops - *Index Store Tag* - add at the end: "The processor uses computed parity in case of primary caches and uses *TagLo* in case of secondary cache."

Page A-46

1) Cache Ops - *Hit WriteBack D*,
Change: (2nd. line) ...write back the data to memory or the secondary cache, and clear the W bit.

to: ...write back the data. The Writeback bit is not cleared; a subsequent miss to the block will write it back again. This second writeback is redundant, but not incorrect.

2) Cache Ops - *Hit WriteBack D*, add to the end:
Implementation note: The Writeback bit is not cleared during this operation due to an artifact of the implementation. The Writeback bit is implemented as part of the data side of the cache array so that it can be written during a data write.

3) Cache Ops - *Hit WriteBack SD*, change the entire description to:
If the cache block contains the specified address, and the cache is Dirty Exclusive or Dirty Shared, write back the data to memory. The cache state is unchanged; a subsequent miss to the block will cause it to be written back again. This second writeback is redundant, but not incorrect. The CH bit in the Status register is set or cleared to indicate a hit or miss. The writeback looks in the primary data cache for modified data, but does not invalidate or clear the

Writeback bit in the primary data cache. *Implementation note*: The state of the secondary block is not changed to clean during this operation because the Writeback bit of matching sub-blocks cannot be cleared to put the primary block in a clean state.

4) Cache Ops - *Fill I*: At the end of the paragraph under "Operation" add the following: For the R4X00PC, the cache is filled from memory. For the R4X00SC and R4X00MC it is filled from the secondary cache whether or not the secondary cache block is valid or contains the specified address.

<u>Page A-81, A-82, A-92, A-93, A-125, A-139</u>

In the box titled "Operation", subsection "32 T:" replace lines 3, 4 & 5 with a copy of lines 3,4 & 5 from the subsection "64T:"

<u>Page A-84</u>

Remove the last paragraph: "Execution of the instruction....unusable exception"

<u>Page A18, A20, A-22, A-24, A-48, A-49. A-85, A-101, A-110, A-111, A-115, A-132, A-154</u>

Remove any direct or related references to CP3 from all of the above pages.

<u>Page A-94, A-96, A-126, A-128</u>

Remove the sentence in the 2nd para which says: "This instruction implicitly performs.....memory after the [LL, LLD, SC, SCD]."

<u>Page A-95</u>

In the box titled "Operation" subsection "32 T":
1) Replace the line "mem<- LoadMemory(...DATA) with a copy of lines 3, 4 & 5 from the subsection "64T:"
2) Replace in line 4 "mem" with "$\text{mem}_{31+8*\text{byte}...8*\text{byte}}$"
3) Remove last line "SyncOperation()" from both "32T:" and "64T:"

<u>Page A-97</u>

In the box titled "Operation:", remove the last line "SyncOperation()".

<u>Page A-99</u>

In the box titled "Operation" subsection "32 T":
1) Replace the line "mem<- LoadMemory(...DATA)" with a copy of lines 3, 4 & 5 from the subsection "64T:"
2) Replace in line 4 "mem" with "$\text{mem}_{31+8*\text{byte}...8*\text{byte}}$"

<u>Page A-101</u>

1) In the box titled "Operation", subsection "64 T:", line 4, replace "DOUBLEWORD" by

"WORD"

2) In the box titled "Operation", subsection "32 T:" replace lines 3, 4 & 5 by a copy of lines 3,4, 5 & 6 from the subsection for "64T:" (including the change shown in item 1).

Page A-103

In the box titled "Operation"

1) In subsection "64T:", line 10, replace "$mem_{31+32*word-8*byte...32*word} \|$...." with "$mem_{32*word+8*byte+7...32*word} \|$...."

2) In subsection "64T:" replace line 5 with "pAddr<-$pAddr_{PSIZE-1...2} \| 0^2$"

3) In subsection "32 T:" replace lines 3 to 8 by a copy of lines 3 to 10 (include changes shown in item 1 & 2)

4) In subsection "32 T:" replace the last line with "GPR[rt] <- temp"

Page A-105

In the 2nd para, last line, change: "In 64 bit mode, the loaded word is sign extended." to "In 64 bit mode, if bit 31 of the destination register is loaded, then the loaded word is sign extended."

Page A-106

In the box titled "Operation"

1) In subsection "64T:", replace line 10 with

temp <- "$GPR[rt]_{31...32-8*byte} \| mem_{31+32*word...32*word+8*byte}$"

2) In subsection "32 T:" replace lines 3 to 8 by a copy of lines 3 to 10 (include changes shown in item 1)

3) In subsection "32 T:" replace the last line with "GPR[rt] <- temp"

Page 107

In the table showing the results, some of the items are not guaranteed to be sign-extended, and some sign-extended indications S in the table must be changed to an X to reflect this.

Change:

VAddr	BigEndian CPU=0					BigEndianCPU=1				
0	S	S	S	S	...	S	S	S	S	...
1	S	S	S	S	...	S	S	S	S	...
2	S	S	S	S	...	S	S	S	S	...
3	S	S	S	S	...	S	S	S	S	...
4	S	S	S	S	...	S	S	S	S	...
5	S	S	S	S	...	S	S	S	S	...
6	S	S	S	S	...	S	S	S	S	...
7	S	S	S	S	...	S	S	S	S	...

To:

0	S	S	S	S	...	X	X	X	X	...
1	X	X	X	X	...	X	X	X	X	...
2	X	X	X	X	...	X	X	X	X	...
3	X	X	X	X	...	S	S	S	S	...
4	S	S	S	S	...	X	X	X	X	...
5	X	X	X	X	...	X	X	X	X	...

6	X X X X ...	X X X X ...
7	X X X X ...	S S S S ...

and in the key below add (after the key for S)

X either unchanged or sign-extend of destination$_{31}$

Page A-108

The opcode for LWU should be 0x 100111 (and not 101111)

Page A-127

In the box titled "Operation:"
1) In the subsection "32 T:", replace line 3 with a copy of line 3 & 4 from the subsection "64T:"
2) In both subsection "32 T:" and "64 T:", remove the last line which says - "SyncOperation()"

Page A-129

In the box titled "Operation:", remove the last line which says - "SyncOperation()"

Page A-140

1) The field [25...21] for the SLL opcode is "00000" and not "rs"
2) In the section titled "Description:" last para should read as follows:
In 64 bit mode, the 32 bit result is sign extended when placed in the destination register. It is sign extended for all shift amounts, including zero; SLL with a zero shift amount truncates a 64-bit value to 32-bits and sign extends this 32-bit value. SLL, unlike nearly all other word operations, does not require an operand to be a properly sign-extended word value to produce a valid sign-extended word result.
Note: SLL with a shift amount of zero may be treated as a NOP by some assemblers at some optimization levels. If using SLL with zero shift to truncate 64-bit values, check the assembler being used.

Page A-141

1) The field [25...21] for the SLLV opcode is "rs" not "00000"
2) In the section titled "Description:" last para should read as follows:
In 64 bit mode, the 32 bit result is sign extended when placed in the destination register. It is sign extended for all shift amounts, including zero; SLLV with a zero shift amount truncates a 64-bit value to 32 bits and sign extends this 32-bit value. SLLV, unlike nearly all other word operations, does not require the operand to be a properly sign-extended word value to produce a valid sign-extended word result.
Note: SLLV with a shift amount of zero may be treated as a NOP by some assemblers at some optimization levels. If using SLLV with zero shift to truncate 64-bit values, check the assembler being used.

Page A-152,

In the box titled "Operation:"
1) In the subsection "32 T:", replace line 3 with a copy of line 3, 4 & 5 from the subsection "64T:"

Page A-153

In the box titled "Operation:"
1) In the subsection "32 T:", replace line 3 with a copy of line 3 & 4 from the subsection "64T:"

Page 156, A-159

In the box titled "Operation:"
1) In the subsection "32 T:", replace line 3 to 8 with a copy of line 3 to 12 from the subsection "64T:"

Page A-169

In the section "Operation:"

1) change:	32	T:	Index<-- 1 $\|\|$ 0^{25} $\|\|$ 1^{6}
to:	32	T:	Index<-- 1 $\|\|$ 0^{25} $\|\|$ Undefined^{6}
2) change:	64	T:	Index<-- 1 $\|\|$ 0^{31}
to:	32	T:	Index<-- 1 $\|\|$ 0^{25} $\|\|$ Undefined^{6}

Page A-181

1) Replace the Opcode 67 LDC3 to LDε & 77 SDC3 to SDε

Page A-182

For the description of the key e, replace the first paragraph with the following:
e Operation codes marked with epsilon are valid when the processor is operating either in the Kernal mode or in the 64-bit non-Kernal (user or supervisor) mode.

Page B-2

In the last para, 2nd to last line, change "...unimplemented instruction trap." to "...unimplemented operation trap."

Page B-6

In the 3rd. paragraph, 4th line - Change FADD to ADD

Page B-8

In paragraph 2 - Change "...aligned-word data items." to "...aligned data items."

Page B-10

In Table B-5: The Operation column for Code 8, 9, 10, 11 & 37 should be "Convert to 64-bit (long) fixed-point..." (instead of "Convert to single fixed-point...")

Page B-11

Replace the box with the following one:

value $\longleftarrow$ ValueFPR(fpr,fmt)

```
if SR_26 = 1 then  /* 64-bit wide FGRs */
      case fmt of
            S, W:
                  value ← FGR[fpr]_31...0
                  return
            D, L:
                  value ← FGR[fpr]
                  return
      endcase
elseif fpr_0 = 0 then   /* valid specifier, 32-bit wide FGRs */
      case fmt of
            S, W:
                  value ← FGR[fpr]
                  return
            D, L:
                  value ← FGR[fpr+1] || FGR[fpr]
                  return
      endcase
else   /* undefined result for odd 32-bit reg #s */
      value ← undefined
endif
```

Page B-12

Replace the box with the following one:

StoreFPR(fpr, fmt, value)

```
if SR₂₆ = 1 then  /* 64-bit wide FGRs */
    case fmt of
        S, W:
            FGR[fpr] ← undefined³² || value
            return
        D, L:
            FGR[fpr] ← value
            return
    endcase
elseif fpr₀ = 0 then   /* valid specifier, 32-bit wide FGRs */
    case fmt of
        S, W:
            FGR[fpr+1] ← undefined
            FGR[fpr] ← value
            return
        D, L:
            FGR[fpr+1] ← value₆₃...₃₂
            FGR[fpr] ← value₃₁...₀
            return
    endcase
else   /* undefined result for odd 32-bit reg #s */
    undefined_result
endif
```

Page B-15

In the first paragraph of "Description",

1) change the phrase (3rd. line) "If the result of the last floating-point compare is false," to "If the result of the last floating-point compare is false(zero),"
2) At the end of the same para. add: "There must be at least one instruction between C.cond.fmt and BC1F."

Page B-16

In the first paragraph of "Description",

1) change: "...floating-point compare is false," to "...floating-point compare is false (zero),"
2) At the end of the same para. add: "There must be at least one instruction between C.cond.fmt and BC1FL."

Page B-17

In the first paragraph of "Description",

1) change: "...floating-point compare is true," to "...floating-point compare is true (one),"
2) At the end of the same para. add: "There must be at least one instruction between C.cond.fmt and BC1T."

Page B-18

In the first paragraph of "Description",
1) change: "...floating-point compare is true," to "...floating-point compare is true (one),"
2) At the end of the same para. add: "There must be at least one instruction between C.cond.fmt and BC1TL."

Page B-19

At the end of 2nd. para. add: "There must be at least one instruction between the compare and the branch."

Page B-21

CEIL.L.fmt - Description-1st. paragraph, 2nd line should read: ".....arithmetically converted to the long........"

Page B-25

In the section titled "Description:", last para. should read as follows:
"The contents of general register *rt* are undefined for the instruction immediately following CFC1."

Page B-26

In the section titled "Description:", last para, last sentence should read as follows:
"The contents of general register *fs* are undefined for the instruction immediately following CTC1."

Page B-28

In the section titled "Description" Add to the last paragraph:
"The operation is not defined if bit 0 of any register specification is set and the FR bit in the Status register equals zero."

Page B-31

In the 1st. para, 2nd line - change "...arithmetically divided." to "...the value in *fs* is divided by the value in *ft.*"

Page B-32

1) In the section titled "Description", 2nd. para should read as follows:
"The contents of general register *rt* are undefined for the instruction immediately following DMFC1."
2) Replace the Operation box with:

```
64    T:     if SR_26 = 1 then   /* 64-bit wide FGRs */
                  data ← FGR[fs]
             elseif fs_0 = 0 then   /* valid specifier, 32-bit wide FGRs */
                  data ← FGR[fs+1] || FGR[fs]
             else   /* undefined for odd 32-bit reg #s */
                  data ⟵ undefined^64
             endif
      T+1:   GPR[rt] ← data
```

3) Add in a new section "Coprocessor Exceptions:"
- Unimplemented operation exception

Page B-33

1) In the section titled "Description", 2nd. para should read as follows:
"The contents of general register *fs* are undefined for the instruction immediately following DMTC1."
2) Replace the Operation box with:

```
64    T:     data ← GPR[rt]
      T+1:   if SR_26 = 1 then   /* 64-bit wide FGRs */
                  FGR[fs] ← data
             elseif fs_0 = 0 then   /*valid specifier, 32-bit wide valid FGRs */
                  FGR[fs+1] ← data_63...32
                  FGR[fs] ← data_31...0
             else   /* undefined result for odd 32-bit reg #s */
                  undefined_result
             endif
```

3) Add in a new section "Coprocessor Exceptions:"
- Unimplemented operation exception

Page B-34

In the section titled "Description" -1st. paragraph, 2nd line should read: ".....arithmetically converted to the long fixed-point format........"

Page B-39

Replace the Operation box with the following:

32 T: vAddr ← $((offset_{15})^{16} \| offset_{15...0}) + GPR[base]$
64 T: vAddr ← $((offset_{15})^{48} \| offset_{15...0}) + GPR[base]$

32, 64 (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
data ← LoadMemory(uncached, DOUBLEWORD, pAddr, vAddr, DATA)
if SR_{26} = 1 then /* 64-bit wide FGRs */
FGR[ft] ← data
elseif ft_0 = 0 then /* valid specifier, 32-bit wide FGRs */
FGR[ft+1] ← $data_{63...32}$
FGR[ft] ← $data_{31...0}$
else /* undefined result if odd */
undefined_result
endif

<u>Page 41</u>

Replace the Operation box with the following:

32 T: vAddr ← $((offset_{15})^{16} \| offset_{15...0}) + GPR[base]$
64 T: vAddr ← $((offset_{15})^{48} \| offset_{15...0}) + GPR[base]$

32, 64 (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
pAddr ← $pAddr_{PSIZE-1...3} \| (pAddr_{2...0}$ xor (ReverseEndian $\| 0^2$))
mem ← LoadMemory(uncached, WORD, pAddr, vAddr, DATA)
byte ← $vAddr_{2...0}$ xor (BigEndianCPU $\| 0^2$)
/* "mem" is aligned 64-bits from memory. Pick out correct bytes. */
if SR_{26} = 1 then /* 64-bit wide FGRs */
FGR[ft] ← $undefined^{32} \| mem_{31+8*byte...8*byte}$
else /* 32-bit wide FGRs */
FGR[ft] ← $mem_{31+8*byte...8*byte}$
endif

<u>Page B-42</u>

1) In the section titled "Description", 2nd. para should read as follows:
"The contents of general register *rt* are undefined for the instruction immediately following MFC1."
2) Replace the Operation box with the following:

32 T: data ← $FGR[fs]_{31...0}$
T+1: GPR[rt] ← data

64 T: data ← $FGR[fs]_{31...0}$
T+1: GPR[rt] ← $(data_{31})^{32} \|$ data

Page B-44

1) In the section titled "Description", 2nd. para should read as follows:
"The contents of general register *fs* are undefined for the instruction immediately following MTC1."
2) Replace the Operation box with the following:

```
32,64   T:      data ← GPR[rt]31...0
        T+1:    if SR26 = 1 then   /* 64-bit wide FGRs */
                     FGR[fs] ← undefined^32 || data
                else   /* 32-bit wide FGRs */
                     FGR[fs] ← data
                endif
```

Page B-52

1) Replace the Operation box with the following:

```
32      T:      vAddr ← (offset15)^16 || offset15...0) + GPR[base]
64      T:      vAddr ← (offset15)^48 || offset15...0) + GPR[base]

32,64           (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
                if SR26 = 1   /* 64-bit wide FGRs */
                     data ← FGR[ft]
                elseif ft0 = 0 then   /* valid specifier, 32-bit wide FGRs */
                     data ← FGR[ft+1] || FGR[ft]
                else   /* undefined for odd 32-bit reg #s */
                     data ← undefined^64
                endif
                StoreMemory(uncached, DOUBLEWORD, data, pAddr, vAddr, DATA)
```

Page B-54

In the 1st. para, 2nd line, change "...arithmetically subtracted." to "...the value in *ft* is subtracted from the value in *fs.*"

Page 56

Replace the Operation box with the following:

32 T: vAddr ← $((offset_{15})^{16} \,||\, offset_{15...0}) + GPR[base]$
64 T: vAddr ← $((offset_{15})^{48} \,||\, offset_{15...0}) + GPR[base]$

32, 64 (pAddr, uncached) ← AddressTranslation (vAddr, DATA)
pAddr ← $pAddr_{PSIZE-1...3} \,||\, (pAddr_{2...0}$ xor (ReverseEndian || 0^2))
byte ← $vAddr_{2...0}$ xor (BigEndianCPU || 0^2)
/* the bytes of the word are put in the correct byte lanes in
 * "data" for a 64-bit path to memory */
if SR_{26} = 1 then /* 64-bit wide FGRs */
data ← $FGR[ft]_{63-8*byte...0} \,||\, 0^{8*byte}$
else /* 32-bit wide FGRs */
data ← $0^{32-8*byte} \,||\, FGR[ft] \,||\, 0^{8*byte}$
endif
StoreMemory (uncached, WORD, data, pAddr, vAddr, DATA)

Page B-57

In the section titled "Description:", 1st. paragraph, 2nd line should read: ".....arithmetically converted to the long fixed-point format........"

Page B-61

In figure B-3, table titled "sub": In the first two rows change all γ 's with δ's

Page B-62

For the key item η, change: "Valid for 64-bit mode only." to "Operation marked with a η are valid only when MIPS III instructions are enabled. An attempt to execute these without MIPS III instruction enabled will cause an unimplemented operation exception."

Page D-2

In section "D.1 Mode Bits" - last line, add the following:
If the di/dt control mechanism is enabled, it is recommended to load the mode bits InitP<3:0> and InitN<3:0> to the values which provide the slowest slew rate.

Page F-1

Equation: (Destination stage of A) - [(Source stage of B) - 1]
should read: (Destination stage of A) - [(Source stage of B) + 1]

Page F-2

1) For Operation *Instruction fetch* and *Load/Store* in the source column
... Config.KOC, Config.IB...
should be
... Config.KO, Config.IB...

2) For Operation *MTC0* and *MFC0* replace the contents with the following:

MTC0/DMTC0	gpr rt (interlocked)	CP0 reg.	7
MFC0/DMFC0	CP0 reg. 5-7	gpr rt (interlocked)	

3) For Operation *CACHE Index Store Tag* change the source stage # from 7 to 8

4) For Operation *Instruction fetch exception*
   a) change the stage #3 to 4
   b) Add XContext in the list of Destination

5) Add the following row:

CACHE ops	PCache line -ε	PCache line -ε

Footnote: ε There must be two non-load and non-CACHE instructions between a store and a CACHE instruction directed to the same primary cache line as the store.

Reader's Comments

Please FAX your comments about this document to:

Anjaneya Thakar: Fax: (415) 390-6170

Address: P.O. Box 7311, MS 952, Mountain View, CA 94039-7311

Areas of Improvement:

Errors:

Reader Information:

Name:

Company:

Address:

Phone: FAX:

Thank you for your feedback.