Restrictions

Publication date: April 2002

Sony Computer Entertainment Inc.
1-1, Akasaka 7-chome, Minato-ku
Tokyo 107-0052 Japan

Sony Computer Entertainment America
919 East Hillsdale Blvd.
Foster City, CA 94404, U.S.A.

Sony Computer Entertainment Europe
30 Golden Square
London W1F 9LD, U.K.

About This Document

This document is a supplement to the system manuals. It describes certain types of problems, the origins of which are often not self-evident. These problems may arise under particular conditions or during use of specific feature combinations, and are related to the characteristics of the EE and GS implementations.

Changes Since Release of 5th Edition

Since release of the 5th Edition of the Restrictions for System Manuals, the following changes have been made. Note that each of these changes is indicated by a revision bar in the margin of the affected page.

Restrictions on EE

- Section (1) Loop, on page 6, was revised.
- Section (3) Microprogram and VCALLMS/VCALLMSR Instruction, on page 6, was revised.
- Section (10) Note on Divide Instructions (2) was added on page 7.
- Section (17) Undefined Instruction (2) was added on page 10.

Restrictions on GS

- Section (1) Depth Test, on page 11, was revised.

Contents

Notes on Hardware

EE and GS chips have the following restrictions. Take note for the use of the following features.

Restrictions on EE

(1) Loop

Do not create a short loop of 6 steps or less that includes a branch instruction.
In such a loop, a judgment whether to branch or not might be incorrect under special conditions. With the following loop, for example, if no hazards ("wait" cycles due to cache misses) occur to l0-a, l1-a, l0-b, and l1-b , it may end after executing once or twice.

```
label:      l0-a                  // Instruction in l0 pipe
            l1-a                  // Instruction in l1 pipe
            l0-b                  // Instruction in l0 pipe
            l1-b                  // Instruction in l1 pipe
            branch target=label   // Branching to label address
            l1-c                  // Instruction in l1 pipe
```

Concrete Example

```
addiu   $a1,$a1,1
addiu   $v1,$v1,4
addu    $s0,$s0,$a1
slti    $v0,$a1,80
bnez    $v0,20 <c+0x20>
sw      $s0,0($v1)
```

This restriction does not apply to a loop that includes another branch instruction.

(2) Arrangement of Program Code and Data

When arranging program code and data in adjoining addresses, put 5 or more NOP instructions, or a combination of SYNC.P and NOP instructions, on the boundary between them. When the data arranged next to the program code has a specific bit pattern, it is regarded as a CACHE instruction, and may fetch a wrong instruction, destroy the data cache, or affect the floating point divide of COP1.

(3) Microprogram and VCALLMS/VCALLMSR Instruction

It is necessary to prohibit VU microsubroutines from generating an interlock due to a resource hazard, data hazard, or SYNC instruction, at the last microinstruction (the instruction following Ebit=1). If there is a possibility of causing an interlock, avoid placing VCALLMS / VCALLMSR successively in the EE Core program. Do so by inserting either another macro instruction, or a move instruction with interlock (CFC2 / CTC2 / LQC2 / SQC2), in between. If this restriction is not followed, a microinstruction may not be executed at the occurrence of an exception.

(4) DMAtag Mismatch Detection Feature

Set VIFn_ERR.ME0 to 1 to disable DMAtag mismatch error detection. If this restriction is not followed, a DMAtag mismatch error interrupt may occur to a normal DMA packet.

(5) DMAtag when Using MFIFO

When using MFIFO, avoid placing ref/refs/refe tag at the end of the DMA packet that is sent from the source channel (ch-8), by taking the following steps.

[End of DMA Packet]
- ref/refs tag: Add cnt tag (QWC=0)
- refe tag: Substitute ref tag + end tag

If this restriction is not followed, the drain channel may stall due to a misunderstanding that MFIFO has become empty during data transfer.

(6) Interrupt Handler

Return from the interrupt handler after executing the EI instruction. If this restriction is not followed, an inconvenience may happen when an interrupt occurs immediately after executing the DI instruction.

(7) Undefined Instruction Exception Handler

In the reserved instruction (RI) exception handler routine, do not execute a COP1 divide instruction (DIV.S, SQRT.S, or RSQRT.S) in the first 18 instructions. If this restriction is not followed, the calculation result by the floating-point divider (FDIV) may cling to a certain value.

(8) Restriction on Data Value Breakpoint

When using a data value breakpoint (BPC.DRE=1, BPC.DVE=1), specify the blocking mode (Status.NBL=0). The breakpoint conditions for the data value breakpoint are established when the specified data value is read from the specified address. In the non-blocking mode, however, the breakpoint conditions are considered to be established, even when concurrently executing an instruction to load a non-specified data value from the specified address and an instruction to load the specified data value from a non-specified address.

(9) Note on Divide Instructions (1)

When executing a COP1 divide instruction (DIV.S/SQRT.S/RSQRT.S) at the D stage, if the pipeline stalls for any reason, the stall continues until the divide instruction ends.

(10) Note on Divide Instructions (2)

COP1 (FPU) divide instructions (DIV.S / SQRT.S / RSQRT.S) and COP2 (VU0) divide instructions (DIV / SQRT / RSQRT / VDIV / VSQRT / VRSQRT) may return incorrect calculation results under certain conditions. Do not place these instructions in
- 1 instruction in the destination of the branch instruction and 1 instruction in the following address
- 3 instructions following the branch instruction (branch delay slot and the following 2 instructions)
- 1 instruction directly preceded by the SYNC.P instruction
- 2 instructions at the beginning of a VU0 microprogram.

(11) DI Instruction

Execute the following code or the like when disabling an interrupt by the DI instruction.

```
#define DI()
{
  u_int stat;
  do {
    asm volatile ("di");
    asm volatile ("sync.p");
    asm volatile ("mfc0     %0, $12" : "=r"(stat):);
  } while (stat & 0x00010000);
}
```

If this method is not applied, the EIE bit becomes 0 (interrupt disable) in the interrupt handler when an interrupt occurs immediately after the DI instruction. Depending on the OS implementation, thread switching, etc. generated in the above state may cause an inconvenience with the interrupt kept disabled.

(12) CACHE/TLB Instruction

Instructions that operate the cache or TLB must be directly preceded by and followed by a SYNC instruction. Detailed information is given to the respective instructions. If this restriction is not followed, an inconvenience may be caused when a COP0 Unusable exception occurs.

(13) CACHE Instructions

In the following instructions, which operate the instruction cache or BTAC, specify a virtual address with MSB (VA[31]) = 0 to the argument. Specifying a virtual address with MSB = 1 may result in a malfunction.

CACHE IXIN	I$ index invalidate
CACHE IXLTG	I$ index load tag
CACHE IXSTG	I$ index store tag
CACHE IXLDT	I$ index load data
CACHE IXSDT	I$ index store data
CACHE BXLBT	index load BTAC
CACHE BXSBT	index store BTAC
CACHE BFH	BTAC flush
CACHE BHINBT	hit invalidate BTAC

(14) CACHE IFL Instruction

If an address specified by the CACHE IFL instruction already exists in the instruction cache, a malfunction, such as an unpredictable TLB miss exception, may occur.
Take steps (e.g. execute the CACHE IHIN instruction in advance) to avoid the specified address from existing in the instruction cache.

(15) TLBR Instruction

The TLBR instruction must not be immediately followed by a jump/branch instruction. Four instructions or more are required between them, excluding the SYNC.P instruction next to the TLBR instruction. Also, the TLBR instruction must not be placed at the end of a page. Six instructions or more from the end of the page are required for the TLBR instruction.
If this restriction is not followed, an inconvenience may be caused when an ITLB miss occurs immediately after the TLBR instruction cancellation, due to the occurrence of an exception, etc.

(16) Write Operations to EPC/ErrorEPC Register

When writing to the EPC/ErrorEPC register with the MTC0 instruction, put any of the SYNC / COP0 / Load / Store instructions that are executed only in the I1 Pipe immediately after the MTC0 instruction. An exception immediately after a write operation with the MTC0 instruction may result in a loss of data written with the MTC0 instruction. This is because the process to write the PC value to the EPC/ErrorEPC register is performed due to the occurrence of an exception.

(17) Undefined Instructions (1)

Undefined instructions with specific bit patterns may be misunderstood in the following cases and cause an inconvenience.

a) The next undefined instruction is in the delay slot of the Branch-Likely instruction.

Inst[31:26] = 010000 &&

Inst[25:21] = 1**** &&

Inst[5: 0] = 01****

If a branch does not occur, instructions in BDS+1 and BDS+2 may not execute.

b) The conditional branch instruction is directly preceded or followed by the next undefined instruction.

Inst[31:26] = 010011 &&

Inst[25:21] = 01000 &&

Inst[18:16] = 000

The branch target address may be mistaken with the above or succeeding branch instructions.

(18) Undefined Instruction (2)

Do not execute the following undefined instructions with specific bit pattern, since they interfere with the operation.

a) Undefined instructions that interfere with floating-point calculations

Inst[31:26]== 010001 &&

Inst[25:23]== 1*0 &&

(Inst[5: 0]== 010**1 || Inst[5:0]==*1*011)

Floating-point calculation results may cling to a certain value. This problem also occurs when this bit pattern exists in the data area next to the program code. Therefore, it is necessary to put 5 or more NOP instructions, or a combination of SYNC.P and NOP instructions on the boundary between the program code and data.

b) Undefined instructions that affect the data cache

Inst[31:26]==101111 &&

(Inst[20:16]== 10101 || 10111 || 11001 || 11011 ||

11101 || 11110 || 11111)

The data cache may be destroyed. An undefined instruction exception does not occcur.

c) Undefined instructions that affect TLB entries

Inst[31:26]==010000 &&

Inst[25:21]== 1**** &&

(Inst[5: 0]==000*** || 0****1 || *01*** || ****1*)

TLB entries may be destroyed.

(19) i Bit of VIFcode

If a VIFcode with the i bit set to 1 is executed in succession to a specific VIFcode (MSCAL, MSCNT, MSCALF, FLUSH, FLUSHE, or FLUSHA), two interrupts may occur in some conditions, even when they are in different packets.

The VIFcode NOP must directly follow the specific VIFcode above and directly precede the VIFcode with the i bit set to 1.

Restrictions on GS

(1) Depth Test

When the ZTE field of the TEST_n register that controls the depth test is switched ON/OFF, the test may not be performed properly. Fix the ZTE bit to 1 (ON). If you are not executing the depth test nor using the Z buffer, instead of setting as ZTE=0, set as ZTST=01(Always). At the same time, set the ZBUF register as ZMSK=1. The GS internal operation in this state is equivalent to the ZTE=0 state.

(2) GS Interrupt Handler

The GS IMR (Interrupt Mask Control) register must be in the masked state while an interrupt from the GS to the EE (SIGNAL/FINISH/HSync/VSync) is being handled. If the register is not masked, the EE may not detect any additional interrupts from the GS. This process is unnecessary for start and termination of V-Blank interrupts, since they differ from the GS interrupt.

(3) Texture Width

The width of textures transferred to GS local memory is restricted by the texel format in use.

Texel Format	Texture Width
PSMCT32	Multiple of 2 Pixels
PSMCT24	Multiple of 8 Pixels
PSMCT16, 16S	Multiple of 4 Pixels
PSMT8, 8H, PSMT4, 4HH, 4HL	Multiple of 8 Pixels

If this restriction is not followed, a part of the data (generally one or more pixels) may not be written correctly to local memory in some rare cases. (The value that had been previously written remains in the PIXEL where the data writing failed.) There will be no offset of the transfer position, due to the writing failure.

(This page is left blank intentionally)