Restrictions

Table of Contents

1 CPU Errata for Final PSP™ Specifications

Using the Fill (D) or Fill with Lock (D) CPU CACHE instruction may cause the cache line to become dirty

Problem

If the Fill (D) or Fill with Lock (D) function of the CACHE instruction is used on an address that has already been cached in the D-cache, the line ends up getting marked as dirty regardless of whether a write operation has been performed to the respective line.

Solution

None.

Do not use the Fill (D) or Fill with Lock (D) function of the CACHE instruction on an area which is already known to be hit in the D-cache. If it is used, a writeback that is essentially unnecessary will end up getting issued, and CPU cycles and bus cycles will be wastefully spent. Memory is not destroyed since the data that is written back comes from the D-cache. The only effect is a performance penalty.

2 VFPU Errata for Final PSP™ Specifications

If a lvl.q or lvr.q instruction is used, the FPU will not work correctly

Problem

There is a hardware bug that causes an invalid instruction to be generated for the FPU if a VFPU lvl.q or lvr.q instruction is used.

Solution

Never use the lvl.q or lvr.q instructions. The lvl.q and lvr.q instructions have been removed from the VFPU instruction set.
At locations where the lvl.q or lvr.q instructions had been used in the past, replace those instructions with the lv.s and lv.q instructions.

Solution Implemented by psp-as

A change was made so that psp-as 1.7.1 or later cannot assemble the lvl.q or lvr.q instructions.
The assembler macro instruction ulv.q was changed so that the lvl.q and lvr.q instructions are no longer generated.

After the VFPU write buffer is used, the next VFPU instruction may produce invalid results

Problem

When an sv.q ,wb instruction is executed to an uncached area which performs a bus access via the VFPU write buffer, and the VFPU write buffer becomes full, then if the next VFPU instruction is an arithmetic instruction that writes to a matrix register file, an invalid value will end up getting written to the destination register.

Examples of Sequences for Which This Problem Occurs

The problem occurs when the following instructions are issued in the order shown below.

```
sv.q    c000, 0(t0), wb     // Write buffer write to an uncached area
vadd.s  s100, s101, s102    // Invalid result in s100
vadd.s  s200, s201, s202    // Correct result in s200
```

The problem also occurs if an ALLEGREX™ CPU or FPU instruction is placed in between

the sv.q,wb instruction and the next VFPU instruction.

```
sv.q    c000, 0(t0), wb     // Write buffer write to an uncached area
addu    t1, t2, t3          // Unrelated CPU or FPU instruction
vadd.s  s100, s101, s102    // Invalid result in s100
vadd.s  s200, s201, s202    // Correct result in s200
```

Explanation

The VFPU write buffer is a write-only bus master that is independent of the normal load store unit of the ALLEGREX™ CPU which is used by CPU and FPU instructions
When the VFPU sv.q,wb instruction is executed for an uncached area, since an access is performed directly to the bus without passing through the load store unit of the CPU, the instruction result can be written to memory without loading down the CPU.
The VFPU write buffer has an 8-stage FIFO structure. If there is no free space in the write buffer, there is a possibility that the sv.q,wb instruction will stall.

If a bus master other than the VFPU (CPU, GraphicsEngine, or DMAC) is occupying the bus when the VFPU write buffer attempts to perform a bus access, the access by the VFPU write buffer may be delayed. Also, although the VFPU write buffer runs at the bus clock (max. 166 MHz), since VFPU instructions are issued according to the normal CPU clock (max. 333 MHz), execution of a subsequent sv.q,wb instruction will immediately cause the VFPU write buffer to become full.

This errata item indicates that if the next VFPU instruction that is issued following an sv.q, wb instruction when the VFPU write buffer is full enters the VFPU pipeline before the write buffer stall is canceled, stall control will not be performed correctly and the operation result will be destroyed. The VFPU write buffer becomes full as a function of bus congestion conditions and the cache hit state, so it is difficult to statically analyze whether or not the problem will occur. Consequently, when the VFPU write buffer is to be used, be sure to take the following measures.

Solution

When the VFPU instruction that is issued after an sv.q ,wb instruction is an instruction that is accompanied by a write to a matrix register file, insert a vnop instruction before it. Even if a CPU or FPU instruction is inserted after the sv.q,wb instruction, the vnop instruction is required since there is a possibility that the succeeding VFPU instruction will have this problem.

Among the VFPU instructions that can follow an sv.q,wb instruction, the following do not require an intervening vnop instruction.

```
sv.s / sv.q / sv.q,wb,
mfv,
mfvc
vnop / vsync / vflush
vpfxd / vpfxs / vpfxt
```

Note that when the VFPU instruction that is issued after an sv.q,wb instruction is an lv.s or lv.q instruction, due to the effect of another errata which is described later, you must also insert a vsync2 instruction before the lv.s or lv.q instruction. Even if you insert a vnop before an instruction that writes to the VFPU register file as explained above, you still need to insert a vsync2 before the lv.s or lv.q.

Solution Implemented by psp-as

With psp-as 1.10.4 or later, the vnop is automatically inserted and the workaround for the errata that will be described later is also done. Details are described later.

Restrictions on the Destination Register for the vhtfm2.p, vhtfm3.t, and vhtfm4.q Instructions

Problem

If a register number in the range \$64 to \$127 is specified for the destination register of a vhtfm2.p, vhtfm3.t, or vhtfm4.q, instruction, the operation result will be invalid.

Example of Code for Which This Problem Occurs

```
vhtfm2.p      c002, e100, c200
vhtfm2.p      c012, e100, c200
vhtfm2.p      c022, e100, c200
vhtfm2.p      c032, e100, c200
vhtfm2.p      r020, e100, c200
vhtfm2.p      r021, e100, c200
vhtfm2.p      r022, e100, c200
vhtfm2.p      r023, e100, c200
      :
vhtfm3.t      c001, e100, c200
vhtfm3.t      c011, e100, c200
vhtfm3.t      c021, e100, c200
vhtfm3.t      c031, e100, c200
vhtfm3.t      r010, e100, c200
vhtfm3.t      r011, e100, c200
vhtfm3.t      r012, e100, c200
vhtfm3.t      r013, e100, c200
      :
vhtfm3.q      c002, e100, c200
```

```
vhtfm4.q      c012, e100, c200
vhtfm4.q      c022, e100, c200
vhtfm4.q      c032, e100, c200
vhtfm4.q      r020, e100, c200
vhtfm4.q      r021, e100, c200
vhtfm4.q      r022, e100, c200
vhtfm4.q      r023, e100, c200
```

This problem occurs only when $64 to $127 is specified for a destination register. In other words, a vhtfm instruction which has opcode bit 6 set to 1 is invalid.

Explanation

If, for example, the scalar registers constituting a 4-dimensional vector c000 (register number $0) are the elements [s000, s001, s002, s003], there exist the following representations as vector register formats for referencing these four registers.

```
2-dimensional vector c000    $0      (s000, s001)
2-dimensional vector c002    $64     (s002, s003)
3-dimensional vector c000    $0      (s000, s001, s002)
3-dimensional vector c001    $64     (s001, s002, s003)
4-dimensional vector c000    $0      (s000, s001, s002, s003)
4-dimensional vector c002    $64     (s002, s003, s000, s001)
```

For each dimensionality, there is one format that represents register number $0 and another format that represents register number $64.
The formats that are affected by this errata item are the vector formats that represent register number $64. In other words, the formats for the 2-dimensional vector c002, 3-dimensional vector c001, and 4-dimensional vector c002 cannot be used as the destination of a vhtfm instruction.

A similar restriction exists for the r format.

```
2-dimensional vector r000    $32     (s000, s010)
2-dimensional vector r020    $96     (s020, s030)
3-dimensional vector r000    $32     (s000, s010, s020)
3-dimensional vector r010    $96     (s010, s020, s030)
4-dimensional vector r000    $32     (s000, s010, s020, s030)
4-dimensional vector r020    $96     (s020, s030, s000, s010)
```

Among these, register number $96 cannot be used as the destination of a vhtfm instruction.

Solution

There are no measures to be taken.

Do not specify the registers described above for the destination register of a vhtfm instruction.

Solution Implemented by psp-as

Since the current psp-as assembler does not permit a register number in the range $64 to $127 to be specified in a vector format, no specific measures need be taken. If a change is made in the future so that register numbers in this range can be used, we expect that the register number of the destination register of a vhtfm instruction will be checked and an error will be reported for formats corresponding to $64 to $127.

After a vdiv.p, vdiv.t, vdiv.q, sv.q,wb, or vflush instruction, corrupted data may end up getting loaded by a following lv.s or lv.q instruction

Problem

If an lv.s or lv.q instruction is used as the VFPU instruction that follows either a vdiv.p, vdiv.t, vdiv.q, sv.q, wb, or vflush, if that lv.s or lv.q causes a D-cache miss, an invalid value may end up getting loaded in the lv.s or lv.q destination.

Example of Code for Which This Problem Occurs

```
vdiv.t      c000, c010, c020
lv.q        c030, 0(a0)

vdiv.q      c000, c010, c020
move        t0, t1
lv.q        c030, 0(a0)

vdiv.p      c000, c010, c020
vnop
lv.q        c030, 0(a0)

sv.q        c010, 0(t0), wb
lv.q        c030, 0(a0)

vflush
lv.q        c030, 0(a0)
```

This problem can occur even when a CPU instruction, FPU instruction, or VFPU instruction other than lv.s or lv.q is inserted before the lv.s or lv.q instruction that follows the vdiv.p, vdiv.t, vdiv.q, sv.q,wb, or vflush instruction.

Explanation

When a D-cache miss occurs, a load instruction such as lv.s or lv.q may issue a busy signal (DC_STALL) that causes a load instruction to stall until the D-cache refill has completed. Also, a VFPU instruction having a long pitch count such as vdiv.pt, vdiv.t, or vdiv.q or an sv.q,wb instruction or a vflush instruction, which depends on the bus state, may issue a coprocessor busy signal (COP_STALL) having a relatively long period, in order to suppress the issuance of the following VFPU instruction to the CPU.
There is a problem in the way these two busy signals are handled. When the completion of the refill due to the D-cache miss of the following load instruction (when DC_STALL disappears) ends up being earlier than the cancellation of the busy signal of the preceding VFPU instruction (when the COP_STALL disappears), the data sent to the VFPU by the CPU will be invalid.
Since the number of cycles until the refill is completed when a D-cache miss occurs varies according to the bus usage conditions, this timing cannot be predicted accurately. Also, since the pitch cycle count is particularly long for vdiv.p, vdiv.t, or vdiv.q, when the D-cache refill is theoretically completed at the fastest timing, the condition for generating this problem can end up getting satisfied.
Since the execution cycle count for the sv.q,wb instruction and vflush instruction may change due to an external cause such as the bus state, the timing condition for which this problem occurs may theoretically end up being satisfied.

Solution

(1) Do not use the vdiv.p, vdiv.t, or vdiv.q instructions. Replace them with multiple occurrences of the vdiv.s instruction.

```
vdiv.q      c000, c010, c020
lv.q        c100, 0(a0)
              ↓
vdiv.s      s000, s010, s020
vdiv.s      s001, s011, s021
vdiv.s      s002, s012, s022
vdiv.s      s003, s013, s023
lv.q        c100, 0(a0)
```

The vdiv.p, vdiv.t, and vdiv.q instructions were removed from the VFPU instruction set. Substitute the vdiv.s instruction at the locations where vdiv.p, vdiv.t, or vdiv.q instruction had previously been used. The vrcp + vmul instructions can also be used instead of the vdiv

instruction.

(2) Insert a vsync2 before an lv.s or lv.q instruction that follows a sv.q,wb instruction.

When the VFPU instruction that is issued after an sv.q,wb instruction is an lv.s or lv.q instruction, insert a vsync2 before the lv.s or lv.q.

```
sv.q          c000, 0(t0), wb
sv.q          c000, 16(t0), wb
lv.q          c100, 0(a0)

              ↓
sv.q          c000, 0(t0), wb
sv.q          c000, 16(t0), wb
vsync2
lv.q          c100, 0(a0)
```

The vsync2 instruction in this case can be replaced by five vnop instructions.

```
sv.q          c000, 0(t0), wb
sv.q          c000, 16(t0), wb
lv.q          c100, 0(a0)

              ↓
sv.q          c000, 0(t0), wb
sv.q          c000, 16(t0), wb
vnop
vnop
vnop
vnop
vnop
lv.q          c100, 0(a0)
```

Note that when an instruction for writing to the VFPU register file is issued after the sv.q,wb instruction but before the lv.s or lv.q instruction is executed, a vnop must also be inserted before that instruction due to effect of another errata, which was described earlier. Even when the workaround of inserting a vnop before an instruction for writing to the VFPU register file is done, the workaround of inserting a vsync2 before an lv.s or lv.q instruction, which is to be executed after it, cannot be omitted.

```
sv.q          c000, 0(t0), wb
sv.q          c000, 16(t0), wb
vnop                                  // Required for vadd.s
vadd.q        c010, c020, c030
vsync2                                // Required for lv.q
lv.q          c100, 0(a0)
```

On the other hand, when the workaround of inserting the vsync2 is executed first, the workaround of inserting the vnop can be omitted.

```
sv.q        c000, 0(t0), wb
sv.q        c000, 16(t0), wb
vsync2                              // Required for lv.q
lv.q        c100, 0(a0)
vadd.q      c010, c020, c030
```

(3) Insert a vsync2 before an lv.s or lv.q instruction that is used after a vflush.

When the VFPU instruction that is issued after the vflush instruction is an lv.s or lv.q, insert a vsync2 before the lv.s or lv.q.

```
vflush
lv.q        c100, 0(a0)

          ↓
vflush
vsync2
lv.q        c100, 0(a0)
```

The vsync2 in this case can be replaced by four vnop instructions.

```
vflush
lv.q        c100, 0(a0)

          ↓
vflush
vnop
vnop
vnop
vnop
lv.q        c100, 0(a0)
```

Solution Implemented by psp-as

- With psp-as 1.10.4 or later, vdiv.p, vdiv.t, and vdiv.q are macro instructions that are automatically divided up into vdiv.s instructions.
- When an lv.s or lv.q instruction follows an sv.q,wb instruction, a vsync2 instruction is automatically inserted.
- When an lv.s or lv.q instruction follows a vflush instruction, a vsync2 instruction is automatically inserted.
- When a jump/branch instruction is executed without executing the vsync2 after an svq,wb or vflush instruction is executed, a vsync2 instruction is automatically inserted.

Although the vdiv.p, vdiv.t, and vdiv.q instructions were removed as instructions from the VFPU specifications, source code compatibility is maintained except when these instructions are placed in the branch delay slot.
The vsync2 instruction can be assembled with psp-as 1.10.4 and later.

The psp-as assembler of psp-as 1.10.4 or later executes the insertion of vsync2 or vnop according to the following algorithm.

- When an sv.q,wb instruction appears, set the vsync2 insertion flag and vnop insertion flag.
- When a vflush instruction appears, set the vsync2 insertion flag.
- When an lv.s or lv.q instruction appears while the vsync2 insertion flag is set, insert a vsync2 instruction and clear the vsync2 and vnop insertion flags.
- When an sv.?, mfv, mfvc, vpfx?, or vsync2 instruction appears while the vsync2 insertion flag is set, clear the vsync2 and vnop insertion flags.
- When a VFPU instruction that is not sv.?, mfv, mfvc, or vpfx? appears while the vnop insertion flag is set, insert a vnop and clear the vnop insertion flag.
- When a vnop or vsync instruction appears while the vnop insertion flag is set, insert a vnop and clear the vnop insertion flag.
- When a branch/jump instruction appears while the vsync2 insertion flag is set, insert a vsync2 command. The insertion position is the delay slot of the branch/jump instruction for .set reorder and before the branch/jump instruction for .set noreorder. However, if the vsync2 instruction had been placed manually in the delay slot when .set noreorder was set, the vsync2 insertion is canceled.
- If a sv.q,wb instruction appears in the delay slot when .set noreorder is set, insert a vsync2 instruction before the branch/jump instruction.
- If a div.p, div.t, div.q, vflush, or sv.q,wb instruction appears in the delay slot of a branch likely instruction when .set noreorder is set, return an assemble error.

3 GraphicsEngine Errata for Final PSP™ Specifications

Dummy command is required when using skinning or morphing

Problem

If the CMD_BONED or CMD_WEIGHT0-7 commands are used immediately after a drawing command is performed during skinning or morphing, the matrix or weight that is being used may be overwritten.

Sample Sequence for Which the Problem Occurs

If the CMD_BONED or CMD_WEIGHT0-7 commands are used without inserting a drawing command within four commands immediately after the following drawing commands that are performed during skinning or morphing, the matrix or weight that is being used may be overwritten.

CMD_PRIM / CMD_BEZIER / CMD_SPLINE

Solution

If the CMD_BONED or CMD_WEIGHT0-7 commands are used to update the matrix or weight immediately after a drawing command is performed during skinning or morphing, a 0xFF000000 dummy command should be inserted after the relevant drawing command.

```
CMD_BONEN
CMD_BONED
    :
CMD_BONED
CMD_PRIM
0xFF000000          // dummy command
CMD_BONEN
    :
CMD_BONED
```

CLUT loading fails when pixel format DXT1 is selected

Problem

If the color lookup table (CLUT) is loaded when the texture buffer pixel format is set to DXT1, CLUT loading may fail.

Explanation

When an attempt is made to load the CLUT after DXT1 data is expanded, the expected CLUT may not be able to be obtained because the S3TC decoding process doesn't terminate and continues to function even when the CLUT is loaded.
This problem can be avoided by always setting the texture buffer pixel format to something other than DXT1 before loading the CLUT.
This problem does not occur when the texture buffer pixel format is DXT3 or DXT5.

Solution

Avoid this problem by always setting the texture buffer pixel format to something other than DXT1 before loading the CLUT.

Garbage may appear when drawing an extremely large primitive

Problem

When the following primitives are used, if the size of the primitive in the screen coordinate system is extremely large, garbage may appear alongside the primitive.
PRIM_LINES
PRIM_LINE_STRIP
PRIM_TRIANGLES
PRIM_TRIANGLE_STRIP

Explanation

With a line primitive or triangle primitive, a problem may occur when the difference between the Y-coordinates in the screen coordinate system after a perspective transformation of two vertices is performed, is greater than or equal to 0xAAAA (2730.625) where the values represent 16-bit unsigned fixed point numbers.
Garbage will not appear with respect to the primitive's X or Z direction.

Solution

If one of the above primitives is extremely large in the Y direction, divide the primitive.

A Graphics Engine area transfer may not finish

Problem

When a GE area transfer command (CMD_XSTART) is executed, if parameters have been

set so that

XBW1+SX1 > 1024

or

XBW2+SX2 > 1024

the area will not be transferred, and the GE will hang.

XBW1 and XBW2, which are the horizontal size of the transfer buffer, are parameters specified by CMD_XBW1 and CMD_XBW2. SX1 and SX2, which are the upper left corner of the transfer area, are parameters specified by CMD_XPOS1 and CMD_XPOS2.

Solution

If the above conditions are satisfied but you need to perform a transfer, either divide up the area before performing the transfer or use some other method other than a GE area transfer to perform the transfer.

Solution Implemented by libgu

Beginning with devkit 2.8, an error is returned when a command for performing an area transfer as described above, is issued.