Intel® Itanium™ Processor

Specification Update

June 2001

Notice: The Intel® Itanium™ processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this specification update.
Contents

Revision History ..................................................................................................................5
Preface ..................................................................................................................................6
General Information......................................................................................................7
Summary Table of Changes ..............................................................................................8
Errata Summary Table .......................................................................................................9
Errata .................................................................................................................................11
Specification Changes ......................................................................................................17
Specification Clarifications ..............................................................................................18
Documentation Changes ..................................................................................................19
## Revision History

<table>
<thead>
<tr>
<th>Version Number</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>001</td>
<td>This document is the first specification update for the Itanium™ processor.</td>
<td>June 2001</td>
</tr>
</tbody>
</table>
Preface

This document is an update to the specifications contained in the Affected Documents/Related Documents in the table below. This document is a compilation of device and documentation errata, specification clarifications and changes. It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools.

Affected Documents/Related Documents

<table>
<thead>
<tr>
<th>Title</th>
<th>Document #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel® Itanium™ Processor at 800 MHz and 733 MHz Datasheet, Rev 1.0</td>
<td>249634</td>
</tr>
<tr>
<td>Intel® Itanium™ Processor Hardware Developer’s Manual, Rev 1.0</td>
<td>248701</td>
</tr>
<tr>
<td>Intel® IA-64 Software Developer’s Manual, Volume 1: IA-64 Application Architecture</td>
<td>245317</td>
</tr>
<tr>
<td>Intel® IA-64 Software Developer’s Manual, Volume 2: IA-64 System Architecture</td>
<td>245318</td>
</tr>
<tr>
<td>Intel® IA-64 Software Developer’s Manual, Volume 3: Instruction Set Reference</td>
<td>245319</td>
</tr>
<tr>
<td>Intel® IA-64 Software Developer’s Manual Specification Update</td>
<td>248699</td>
</tr>
<tr>
<td>Intel® Itanium™ Processor Family System Abstraction Layer Specification (available on the web)</td>
<td>245359</td>
</tr>
</tbody>
</table>

Nomenclature

S-Spec Number is used to identify products. Products are differentiated by their unique characteristics, e.g. core speed, L3 cache size, package types, etc. Care should be taken to read all notes associated with each S-Spec number.

Errata are design defects or errors. Errata may cause the Intel® Itanium™ processor’s behavior to deviate from published specifications. Hardware and software designed to be used with any given processor must assume that all errata documented for that stepping are present on all devices unless otherwise noted.

Specification Changes are modifications to the current published specifications for the Intel Itanium processor. These changes will be incorporated in the next release of the specifications.

Specification Clarifications describe a specification in greater detail or further highlight a specification’s impact to a complex design situation. These clarifications will be incorporated in the next release of the specification.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.
General Information

Intel® Itanium™ Processor Cartridge Marking

The cartridge mark for the product is a laser marked label attached to the end of the PAC418 cartridge opposite the power tab. The mark provides the following information:

- Product brand name
- Manufacture traceability (including manufacturing lot and unit identification)
- Manufacturing site
- Mask work and copyright protection for the processor and cache die
- Product identification (including processor core speed, cache size, bus speed)
- Intel company logo
- Two-dimensional product identification matrix for Intel internal use

![Cartridge Marking Diagram]

Intel® Itanium™ Processor Identification and Package Information

<table>
<thead>
<tr>
<th>S-Spec/QDF Number</th>
<th>Core Stepping</th>
<th>CPUID</th>
<th>Speed (MHz)</th>
<th>L3 Size (Mbytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL4LT</td>
<td>C0</td>
<td>0007000604h</td>
<td>733/133</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>SL4LS</td>
<td>C0</td>
<td>0007000604h</td>
<td>733/133</td>
<td>4</td>
<td>a</td>
</tr>
<tr>
<td>SL4LR</td>
<td>C0</td>
<td>0007000604h</td>
<td>800/133</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>SL4LQ</td>
<td>C0</td>
<td>0007000604h</td>
<td>800/133</td>
<td>4</td>
<td>a</td>
</tr>
</tbody>
</table>

a. The CPUID column in this table indicates the contents of bits 39:0 of CPUID Register 3. Bits 63:40 of this register are reserved.
Mixed Steppings in MP Systems

The Itanium processor supports multi-processor platforms using mixed processor steppings of N and N-1 on the same system bus. While Intel has done nothing to specifically prevent processors within a multi-processor environment from operating with mixed steppings beyond the N and N-1 configurations, there may be uncharacterized errata that exist in such configurations.
Summary Table of Changes

The following table indicates the errata, specification changes, specification clarifications, or documentation changes which apply to the Intel Itanium processor steppings. Intel may fix some of the errata in a future stepping of the component or in a future release of the Processor Abstraction Layer (PAL), and account for the other outstanding issues through documentation or specification changes as noted. This table uses the notations indicated below.

### Codes Used in Summary Table of Changes

#### Stepping

<table>
<thead>
<tr>
<th>X:</th>
<th>Errata exists in the stepping or PAL version indicated. Specification Change or Clarification that applies to this stepping.</th>
</tr>
</thead>
<tbody>
<tr>
<td>(No mark) or (Blank box):</td>
<td>This erratum is fixed in listed stepping or specification change does not apply to listed stepping or PAL version.</td>
</tr>
</tbody>
</table>

#### Page

| (Page): | Page location of item in this document. |

#### Status

| Doc: | Document change or update will be implemented. |
| Fix: | This erratum is intended to be fixed in a future step of the component, or in a future release of PAL. |
| Fixed: | This erratum has been previously fixed. |
| NoFix: | There are no plans to fix this erratum. |
| Eval: | Plans to fix this erratum are under evaluation. |

#### Row

Change bar to left of table row indicates this erratum is either new or modified from the previous version of this document.
## Errata Summary Table

### Errata

<table>
<thead>
<tr>
<th>No.</th>
<th>Processor Stepping</th>
<th>PAL Version</th>
<th>Pg</th>
<th>Status</th>
<th>Errata</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C0  6.6.21  6.6.23</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td>12</td>
<td>NoFix</td>
<td>IFA may contain incorrect address</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td></td>
<td>12</td>
<td>NoFix</td>
<td>AR.ITC returns an incorrect value</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td></td>
<td>12</td>
<td>NoFix</td>
<td>LTD line fill during DBR/IBR access may result in an incorrect line fill</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td></td>
<td>12</td>
<td>Fix</td>
<td>IA-32: Incorrect self-modifying code behavior</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td></td>
<td>13</td>
<td>NoFix</td>
<td>System bus pins driven low during JTAG continuity testing</td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td></td>
<td>13</td>
<td>Fix</td>
<td>INIT# signal not recognized properly</td>
</tr>
<tr>
<td>7</td>
<td>X</td>
<td></td>
<td>13</td>
<td>Fixed</td>
<td>BINIT condition prevents error logging</td>
</tr>
<tr>
<td>8</td>
<td>X</td>
<td></td>
<td>13</td>
<td>NoFix</td>
<td>Internal BINIT during low power light halt mode</td>
</tr>
<tr>
<td>9</td>
<td>X</td>
<td></td>
<td>13</td>
<td>NoFix</td>
<td>Incorrect error logging information for double-bit ECC error on PTC or interrupt transaction</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td></td>
<td>14</td>
<td>Fix</td>
<td>Processor hang during nested BINIT</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
<td></td>
<td>14</td>
<td>Fix</td>
<td>False BIL transactions during PAL_CACHE_FLUSH</td>
</tr>
<tr>
<td>12</td>
<td>X</td>
<td></td>
<td>14</td>
<td>Fix</td>
<td>ALL_STOPS_DISPERSED and EXPL_STOPS_DISPERSED not operating correctly</td>
</tr>
<tr>
<td>13</td>
<td>X</td>
<td></td>
<td>14</td>
<td>NoFix</td>
<td>Snoop hit to modified line in L2 cache with tag parity error</td>
</tr>
<tr>
<td>14</td>
<td>X</td>
<td></td>
<td>15</td>
<td>Fix</td>
<td>Infinite DBSY# hang due to livelock condition</td>
</tr>
<tr>
<td>15</td>
<td>X</td>
<td></td>
<td>15</td>
<td>NoFix</td>
<td>Unexpected writeback transaction in 2:11 mode with bus parking enabled</td>
</tr>
<tr>
<td>16</td>
<td>X</td>
<td></td>
<td>15</td>
<td>Fix</td>
<td>IA-32: Code with FP instruction followed by integer instruction with interrupt pending may not execute correctly</td>
</tr>
<tr>
<td>17</td>
<td>X</td>
<td></td>
<td>15</td>
<td>Fixed</td>
<td>Bus parking always enabled and not controllable by PAL</td>
</tr>
<tr>
<td>18</td>
<td>X</td>
<td></td>
<td>16</td>
<td>Fix</td>
<td>IA-32: FSINCOS may not generate FP precision exception</td>
</tr>
<tr>
<td>19</td>
<td>X</td>
<td></td>
<td>16</td>
<td>NoFix</td>
<td>chk.a.clr/ld.c.clr incorrectly clears its ALAT entry</td>
</tr>
<tr>
<td>20</td>
<td>X</td>
<td></td>
<td>16</td>
<td>Fix</td>
<td>IA-32: Back to back semaphores under certain circumstances may cause processor to hang</td>
</tr>
<tr>
<td>21</td>
<td>X</td>
<td></td>
<td>17</td>
<td>Fix</td>
<td>Modification of PFS under certain circumstances can lead to unexpected program behavior</td>
</tr>
</tbody>
</table>
## Specification Changes

<table>
<thead>
<tr>
<th>No.</th>
<th>Processor Stepping</th>
<th>PAL Versions</th>
<th>Pg</th>
<th>Status</th>
<th>SPECIFICATION CHANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>6.6.21</td>
<td>6.6.23</td>
<td></td>
<td></td>
<td>None for this revision of the Specification Update</td>
</tr>
</tbody>
</table>

## Specification Clarifications

<table>
<thead>
<tr>
<th>No.</th>
<th>Processor Stepping</th>
<th>PAL Versions</th>
<th>Pg</th>
<th>Status</th>
<th>SPECIFICATION CLARIFICATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>6.6.21</td>
<td>6.6.23</td>
<td></td>
<td></td>
<td>None for this revision of the Specification Update</td>
</tr>
</tbody>
</table>

## Documentation Changes

<table>
<thead>
<tr>
<th>No.</th>
<th>Processor Stepping</th>
<th>PAL Versions</th>
<th>Pg</th>
<th>Status</th>
<th>DOCUMENTATION CHANGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>6.6.21</td>
<td>6.6.23</td>
<td></td>
<td></td>
<td>None for this revision of the Specification Update</td>
</tr>
</tbody>
</table>
Errata

1. IFA may contain incorrect address

**Problem:** The Interruption Fault Address (IFA) is not correctly reported for Instruction Access faults and Instruction Debug faults when the processor is executing Itanium instructions. The IFA is reported correctly when the processor is executing IA-32 instructions.

**Implication:** The IFA may report a wrong IP address on an Instruction Access-bit access or an Instruction Debug fault. Operating systems should not rely on the value in IFA when these faults occur.

**Workaround:** The IIP (Interruption Instruction Bundle Pointer) provides the same information as the IFA for Instruction Access faults and Instruction Debug faults. These fault handlers should use the IFA or the IIP based on IPSR.is:

```c
if (IPSR.is)
    use IFA
else
    use IIP
```

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.

2. AR.ITC returns an incorrect value

**Implication:** The 64-bit Interval Timer Counter register (AR.ITC) may return an incorrect value when the lower 32-bits are all F’s. In this case, the value returned in the upper 32-bits is incremented by 1. For example, when the value returned is 0x1FFFFFFFF, the actual value should be 0x0FFFFFFFF.

**Implication:** Software that utilizes the AR.ITC register will receive an incorrect value in this case.

**Workaround:** The workaround is for software to re-read the AR.ITC register when it detects all F’s in the lower 32-bits

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.

3. L1D line fill during DBR/IBR access may result in an incorrect line fill

**Implication:** Under certain circumstances, if a line fill to the L1 data cache occurs simultaneously with Debug Breakpoint Register (DBR/IBR) accesses, the line fill to the L1 data cache may incorrectly occur to address zero. In general, this erratum can be encountered if the code sequence contains a ‘Move to Data Breakpoint Register’ instruction followed by a ‘Move to Instruction Breakpoint Register’ instruction.

**Implication:** Code that operates on the debug breakpoint registers in the above sequence may possibly overwrite address zero data. Subsequent use of this data may result in unpredictable behavior.

**Workaround:** Insert a serializing instruction ‘srlz.d’ in between the ‘mov dbr’ and the ‘mov ibr’ instruction.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.

4. IA-32: Incorrect self-modifying code behavior

**Problem:** Under certain circumstances, the processor may fail to correctly execute IA-32 self-modifying code (SMC). In the failing scenario, the processor does not wait for a previous store to complete prior to issuing and completing an instruction fetch to the same address.

**Implication:** IA-32 SMC may execute incorrectly resulting in unexpected program behavior.

**Workaround:** A workaround for this erratum is implemented in PAL 6.6.21 and later versions.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.
5. **System bus pins driven low during JTAG continuity testing**

**Problem:** During JTAG Boundary Scan continuity testing, the system bus pins may be driven low by the processor not allowing the pins to be forced to a high state.

**Implication:** The system bus RESET# pin cannot be tested for continuity as board-level component interconnect testing requires the system bus RESET# pin to be kept asserted.

**Workaround:** Assert the system bus RESET# pin during JTAG Boundary Scan continuity testing.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.

6. **INIT# signal not recognized properly**

**Problem:** The INIT# signal triggers an unmasked interrupt to the processor. When operating at the 2:11 bus-to-core frequency ratio, the assertion of the INIT# pin may not always be recognized by the processor, preventing the processor from taking the interrupt.

**Implication:** The processor may miss taking the INIT# interrupt when operating at the 2:11 bus-to-core frequency ratio. Note: This erratum does not impact the use of the INIT# pin for power-on configuration during reset.

**Workaround:** Either a system bus-based interrupt transaction or the Platform Management Interrupt (PMI)# input can be used to implement the same functionality.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.

7. **BINIT condition prevents error logging**

**Problem:** On a BINIT non-recoverable Machine Check Abort (MCA) condition, the processor may enter an infinite loop in the PAL MCA handler. This prevents hand-off to the SAL MCA handler from occurring.

**Implication:** On a BINIT MCA condition, the processor may encounter a hang. This will prevent the BINIT error from being logged. Note: Since BINIT indicates that a fatal condition has occurred which prevents reliable future operation, the system would normally be reset.

**Workaround:** None identified at this time.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.

8. **Internal BINIT during low power light halt mode**

**Problem:** The processor enters a light halt mode upon executing either the PAL_HAL T_LIGHT or the PAL_HAL T_LIGHT_SPECIAL calls. During light halt mode, if the processor encounters an internal BINIT (Bus Initialization) non-recoverable MCA condition, the processor may not flag the MCA as expected.

**Implication:** The processor may not flag the fatal error condition as expected during light halt mode.

**Workaround:** Flush and invalidate all cache lines by calling the PAL_CACHE_FLUSH call with inv = 1 before entering the light halt mode.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.

9. **Incorrect error logging information for double-bit ECC error on PTC or interrupt transaction**

**Problem:** The processor logs an incorrect request type and address information for a double-bit ECC error on an interrupt or an inbound ptc transaction.

**Implication:** For a double-bit ECC error on an inbound ptc transaction, the PAL machine check handler may incorrectly raise machine check abort. For a double-bit ECC error on an interrupt transaction, the
processor may incorrectly raise a Corrected Machine Check Interrupt (CMCI) instead of an MCA, leading to possible incorrect functionality.

**Workaround:**
The workaround implemented in PAL release 6.6.23 promotes all PAL continuable double-bit ECC system bus errors to OS-recoverable machine check aborts. The MCA logging for double-bit ECC system bus errors may be inaccurate, but errors will be contained.

**Status:**
For the steppings affected, see the Summary Table of Changes at the beginning of this section.

**10. Processor hang during nested BINIT**

**Problem:**
On an external assertion of a BINIT non-recoverable MCA within a certain time interval while the processor is servicing a previous external or internally generated BINIT non-recoverable MCA, the system bus may stall due to an infinite assertion of the Block Next Request (BNR)# signal.

**Implication:**
The processor may hang as a result of the nested BINIT condition.

**Workaround:**
None identified at this time.

**Status:**
For the steppings affected, see the Summary Table of Changes at the beginning of this section.

**11. False BIL transactions during PAL_CACHE_FLUSH**

**Problem:**
During PAL_FLUSH_CACHE, the processor may issue BIL transactions to incorrect addresses when run in cacheable mode with inv = 1. Although the false BIL transaction is issued, actual data is not modified and cache coherency is still maintained.

**Implication:**
False BIL transactions may be seen on the system bus during execution of the PAL_CACHE_FLUSH call.

**Workaround:**
Since the false BIL always targets a certain address range, depending on how PAL is mapped to writeback memory, the chipset can ignore any bus transactions issued by the processor to that address range.

**Status:**
For the steppings affected, see the Summary Table of Changes at the beginning of this section.

**12. ALL_STOPS_DISPERSED and EXPL_STOPS_DISPERSED not operating correctly**

**Problem:**
The performance monitoring event ALL_STOPS_DISPERSED counts the implicit and explicit stops dispersed, while the event EXPL_STOPS_DISPERSED counts the explicit stops dispersed. These counters incorrectly count their respective events.

**Implication:**
These performance monitoring events may report an incorrect count.

**Workaround:**
None identified at this time.

**Status:**
For the steppings affected, see the Summary Table of Changes at the beginning of this section.

**13. Snoop hit to modified line in L2 cache with tag parity error**

**Problem:**
On a snoop hit to a modified line in the L2 cache which encounters a tag parity error, the processor may incorrectly report the snoop response as a miss and hang. The processor does flag an MCA and raise Bus Error (BERR) as expected.

**Implication:**
The processor may hang as a result of encountering a snoop hit to a modified line in the L2 cache with a tag parity error. Also, since the L2 cache reports the snoop response as a miss, another processor can potentially modify the data for that cache line.

**Workaround:**
Enable BERR to BINIT promotion using the PAL_PROC_SET_FEATURES call.

**Status:**
For the steppings affected, see the Summary Table of Changes at the beginning of this section.
14. **Infinite DBSY# hang due to livelock condition**

**Problem:** In the event of several internal conditions and the specific alignment of these conditions, the processor may encounter a potential livelock. The conditions involve a stream of L3 cache traffic, victimization of modified lines from the L3 cache, a pending L2 cache fill, and a snoop request from an external bus agent that hits the same pending L2 cache line.

**Implication:** As a result of the livelock condition, the processor asserts the Data Bus Busy (DBSY)# signal without a corresponding data transfer, causing the system to hang. This erratum has only been observed running a synthetic cache stress test.

**Workaround:** None identified at this time.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.

15. **Unexpected writeback transaction in 2:11 mode with bus parking enabled**

**Problem:** Following are the necessary conditions for this erratum:

- The processor is configured in 2:11 bus-to-core ratio and bus parking is enabled.
- One processor Px owns two cache lines A and B in the M-state.
- A snoop request (due to BIL/BRIL transaction) for cache line A followed by a snoop request (due to BIL/BRIL transaction) for cache line B is made (by priority agent or processor Py).
- A specific timing relationship of IDS#, ADS#, and BPRI#/BRy# signals needs to be met.
- In parallel, Px tries to issue an explicit writeback for A followed by an explicit writeback for B.

As a result, the processor may incorrectly issue an explicit writeback transaction for B on the system bus, following the implicit writeback for A and B, which is a violation of the system bus protocol. In addition, the explicit writeback transaction for B may contain incorrect data.

If the processor has another read/write request that needs to be issued to the system bus immediately after the explicit writeback transaction for B, the request may be held pending indefinitely in the processor bus queue causing the processor to hang.

**Implication:** The processor may issue an explicit writeback transaction with incorrect data when it is not expected to do so. In the case where the read/write request is held pending indefinitely inside the processor, the system may hang. This erratum has only been observed running a synthetic stress test.

**Workaround:** The workaround for this erratum is to disable bus parking.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.

16. **IA-32: Code with FP instruction followed by integer instruction with interrupt pending may not execute correctly**

**Problem:** In the event of certain internal conditions, IA-32 code that contains a floating-point instruction followed immediately by an integer instruction with an interrupt pending may not execute correctly.

**Implication:** IA-32 floating-point code in the above scenario may not execute correctly.

**Workaround:** A workaround for this erratum is implemented in PAL release 6.6.23.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.
17. **Bus parking always enabled and not controllable by PAL**

**Problem:** The processor can be configured to not park on the request bus when idle, if A15# is sampled deasserted at the asserted-to-deasserted transition of RESET#. Due to this erratum, PAL overrides the hardware setting and enables bus parking by default. Additionally, the PAL_BUS_GET_FEATURES call reports that bus parking (bit 29) is not controllable and does not allow the PAL_BUS_SET_FEATURES call to enable or disable bus parking.

**Implication:** PAL enables bus parking by default and overrides the hardware setting at reset. Additionally, bus parking cannot be controlled using the PAL_BUS_SET_FEATURES call.

**Workaround:** None identified at this time.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.

18. **IA-32: FSINCOS may not generate FP precision exception**

**Problem:** The IA-32 FP instruction, FSINCOS computes both the sine and cosine of a source operand, with the final floating-point operations in the computation expected to generate a precision exception. For certain source operand values, these final operations may be computed exactly, causing the expected floating-point precision exception not to be generated.

**Implication:** For certain source operands, the FSINCOS instruction fails to generate the expected floating-point precision exception.

**Workaround:** None identified at this time.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.

19. **chk.a.clr/ld.c.clr incorrectly clears its ALAT entry**

**Problem:** Under certain conditions following the execution of an advanced load, the speculation check instructions, chk.a.clr and ld.c.clr, may incorrectly report a miss in the Advanced Load Address Table (ALAT) indicating that the data speculation was unsuccessful.

**Implication:** As a result of the miss, the check load (ld.c) will reload the correct value from memory and the advanced load check (chk.a) will branch to compiler-generated recovery code.

**Workaround:** Replace chk.a.clr with chk.a.nc, and ld.c.clr with ld.c.nc.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.

20. **IA-32: Back to back semaphores under certain circumstances may cause processor to hang**

**Problem:** In the event of certain internal conditions involving the processor Virtual Hash Page Table (VHPT) walker, while the processor is executing a sequence of back-to-back IA-32 semaphore operations, the processor may hang.

**Implication:** IA-32 code operating on locked semaphores in the above scenario may cause the processor to hang. Note: Typical spin lock code containing semaphores is not affected by this erratum. This erratum has only been observed running a focused test in a system validation environment.

**Workaround:** Separate back-to-back IA-32 semaphores with a minimum of two NOPs.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.
21. **Modification of PFS under certain circumstances can lead to unexpected program behavior**

**Problem:** The Previous Function State register (PFS) is provided to accelerate procedure calling. Under certain circumstances involving a specific code sequence, and a set of internal architectural and timing conditions, modification of the PFS fields prior to br.ret may result in incorrect restoration of state on returning back to the caller procedure.

**Implication:** Execution of code that meets the above criteria may result in unexpected program behavior. The erratum is not exposed for typical code involving PFS restoration identical to the initial PFS value stored on a procedure call, or code that creates a new stack frame by loading a new CFM to PFS prior to br.ret. This erratum has only been observed running a focused test in a system validation environment.

**Workaround:** None identified at this time.

**Status:** For the steppings affected, see the Summary Table of Changes at the beginning of this section.
There are no Specification Changes for this revision of the *Intel® Itanium™ Processor Specification Update.*
There are no Specification Clarifications for this revision of the Intel® Itanium™ Processor Specification Update.
There are no Documentation Changes for this revision of the *Intel® Itanium™ Processor Specification Update.*