Intel® Celeron® Processor 500 Series

Specification Update

For Platforms Based on Mobile Intel® 965 Express Chipset

March 2008

Revision 006

Document Number: 317667-006

Contents

Revision History

Revision	Description	Date
-001	Initial release	June 2007
-002	Added Note in Component Marking section to distinguish between code named Napa and Santa Rosa platforms.	September 2007
-003	• Clarification of TRANSLATION LOOKASIDE BUFFERS (TLBS) Invalidation • Added AR99 – AR102	October 2007
-004	• Add AR103	November 2007
-005	• Updated Summary Table of Changes • Updated AR8 • Added AR104	December 2007
-006	• Added AR105 • Updated Summary Table of Changes	January 2008

§

Preface

This document is an update to the specifications contained in the documents listed in the following Affected Documents table. It is a compilation of device and document errata and specification clarifications and changes, and is intended for hardware system manufacturers and for software developers of applications, operating system, and tools.

Information types defined in the Nomenclature section of this document are consolidated into this update document and are no longer published in other documents. This document may also contain information that has not been previously published.

Affected Documents

Document Title	Document Number/Location
Intel® Celeron® Processor 500 Series for Platforms Based on Mobile Intel® 965 Express Chipset Family Datasheet	316205

Related Documents

Document Title	Document Number/Location
Intel® 64 and IA-32 Architecture Software Developer's Manual Documentation Changes	252046
Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 1: Basic Architecture	253665
Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M	253666
Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z	253667
Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 3A: System Programming Guide	253668
Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 3B: System Programming Guide	253669
IA-32 Intel® Architecture Optimization Reference Manual	248966
Intel Processor Identification and the CPUID Instruction Application Note (AP-485)	241618
Intel® 64 and IA-32 Architectures Application Note TLBs, Paging-Structure Caches, and Their Invalidation	317080

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics (for example, core speed, L2 cache size, package type, etc.) as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number.

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

Specification Changes are modifications to the current published specifications. 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 specifications.

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

Note: Errata remain in the specification update throughout the product's lifecycle, or until a particular stepping is no longer commercially available. Under these circumstances, errata removed from the specification update are archived and available upon request. Specification changes, specification clarifications and documentation changes are removed from the specification update when the appropriate changes are made to the appropriate product specification or user documentation (datasheets, manuals, etc.).

§

Identification Information

Component Identification via Programming Interface

The Intel® Celeron® processor 500 series can be identified by the following register contents:

Family[1]	Model[2]	Model for A-1 step
0110	1111	10000

NOTES:

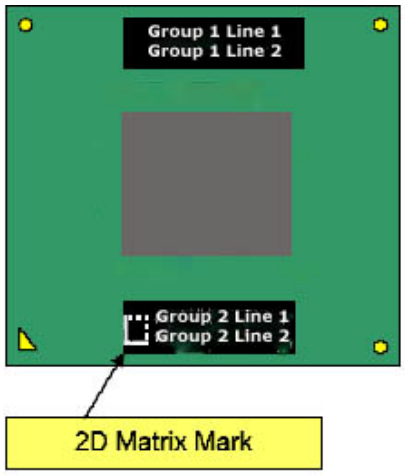
1. The family corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID registers accessible through boundary scan.
2. The model corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the device ID registers accessible through boundary scan.

The stepping of the Intel Celeron processor 500 series can be identified by software with its CPU signature:

Stepping	CPU Signature
A-1	10661h
E-1	06FAh

Component Marking Information

Figure 1. Intel® Celeron® Processor 500 Series (Micro-FCPGA) Markings



SAMPLE MARK EXAMPLE:
Group 1 Line 1: Unit Identifier ---
Group 1 Line 2: FPO S-Spec#

Group 2 Line 1: INTEL (m) © '05
Group 2 Line 2: ATPO Serial Number

For Pb-Free:
Group 2 Line 1: INTEL (m) © '05 (e1)

Table 1. Intel® Celeron® Processor 500 Series Component Markings

S-Spec	Package	Processor Number	Processor Stepping	FSB Frequency (MHz)	Speed (GHz)	Voltage (V)	Notes
SLA2F	Micro-FCPGA	540	A-1	533	1.86	1.30–0.95	1
SLA2G	Micro-FCPGA	530	A-1	533	1.73	1.30-0.95	1,5
SL9VA	Micro-FCPGA	530	A-1	533	1.73	1.30-0.95	1,4
SLA48	Micro-FCPGA	530	E-1	533	1.73	1.30-0.95	2,5
SLA2F	Micro-FCPGA	540	A-1	533	1.86	1.30–0.95	1,5
SLA47	Micro-FCPGA	540	E-1	533	1.86	1.30–0.95	2,5
SLA2E	Micro-FCPGA	550	A-1	533	2.00	1.30–0.95	1,5

NOTES:
1. Intel Celeron processor 500 series Standard Voltage based on a single core, 1-M L2 Cache

Summary Tables of Changes

The following table indicates the Specification Changes, Errata, Specification Clarifications or Documentation Changes, which apply to the listed CPU steppings. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or Specification Changes as noted. This table uses the following notations:

Codes Used in Summary Table

Stepping

X:	Erratum, Specification Change or Clarification that applies to this stepping.
(No mark) or (Blank Box):	This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

Status

Doc:	Document change or update that will be implemented.
Plan Fix:	This erratum may be fixed in a future stepping of the product.
Fixed:	This erratum has been previously fixed.
No Fix:	There are no plans to fix this erratum.

Row

Shaded:	This item is either new or modified from the previous version of the document.

Note: Each Specification Update item is prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Dual-Core Intel® Xeon® processor 7000$^{\Delta}$ sequence

C = Intel® Celeron® processor

D = Dual-Core Intel® Xeon® processor 2.80 GHz

E = Intel® Pentium® III processor

F = Intel® Pentium® processor Extreme Edition and Intel® Pentium® D processor

I = Dual-Core Intel® Xeon® processor 5000$^{\Delta}$ series

J = 64-bit Intel® Xeon® processor MP with 1-MB L2 cache

K = Mobile Intel® Pentium® III processor

L = Intel® Celeron® D processor

M = Mobile Intel® Celeron® processor

N = Intel ® Pentium® 4 processor

O = Intel ® Xeon® processor MP

P = Intel ® Xeon® processor

Q = Mobile Intel® Pentium® 4 processor supporting Hyper-Threading Technology on 90-nm process technology

R = Intel® Pentium® 4 processor on 90 nm process

S = 64-bit Intel® Xeon® processor with 800-MHz system bus (1-MB and 2-MB L2 cache versions)

T = Mobile Intel® Pentium® 4 processor–M

U = 64-bit Intel® Xeon® processor MP with up to 8-MB L3 cache

V = Mobile Intel® Celeron® processor on .13 micron process in micro-FCPGA package

W= Intel® Celeron®-M processor

X =Intel® Pentium® M processor on 90-nm process with 2-MB L2 cache and Intel® processors A100 and A110 with 512-KB L2 cache

Y = Intel® Pentium® M processor

Z = Mobile Intel® Pentium® 4 processor with 533-MHz system bus

AA= Intel® Pentium® D Processor 900$^{\Delta}$ Sequence and Intel® Pentium® processor Extreme Edition 955, 965

AB= Intel® Pentium® 4 processor 6x1 sequence

AC= Intel® Celeron® processor in 478-pin package

AD = Intel® Celeron® D processor on 65-nm process

AE = Intel® Core™ Duo processor and Intel® Core™ Solo processor on 65-nm process

AF = Dual-Core™ Intel® Xeon® processor LV

AG = Dual-Core Intel® Xeon® processor 5100$^{\Delta}$ series

AH= Intel® Core™2 Duo/Solo processor for Intel® Centrino® Duo processor technology

AI = Intel® Core™2 Extreme processor X6800$^{\Delta}$ and Intel® Core™2 Duo Desktop processor E6000$^{\Delta}$ and E4000$^{\Delta}$ sequence

AJ = Quad-Core Intel® Xeon® processor 5300$^{\Delta}$ series

AK = Intel® Core™2 Extreme quad-core processor QX6000 sequence and Intel® Core™2 Quad processor Q6000 sequence

AL = Dual-Core Intel® Xeon® processor 7100 $^{\Delta}$ series

AM = Intel® Celeron® processor 400 sequence

AN = Intel® Pentium® Dual-Core processor

AO = Quad-Core Intel® Xeon® processor 3200$^{\Delta}$ series

AP = Dual-Core Intel® Xeon® processor 3000$^{\Delta}$ series

AQ = Intel® Pentium® Dual-Core Desktop processor E2000$^{\Delta}$ sequence

AR = Intel® Celeron® Processor 500$^{\Delta}$ series

AS = Intel® Xeon® processor 7200, 7300$^{\Delta}$ series

AT = Intel® Celeron® processor 200 series

AV = Intel® Core™2 Extreme Processor QX9000 Sequence and Intel® Core™2 Quad Processor Q9000 Sequence processor

AX = Quad-Core Intel® Xeon® Processor 5400 Series

Note: Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

Number	Stepping	Stepping	Plans	ERRATA
	A1	E1		
AR1	X	X	No Fix	Writing the Local Vector Table (LVT) When an Interrupt Is Pending May Cause an Unexpected Interrupt
AR2	X	X	No Fix	LOCK# Asserted during a Special Cycle Shutdown Transaction May Unexpectedly Deassert
AR3	X	X	No Fix	Address Reported by Machine-Check Architecture (MCA) on Single-bit L2 ECC Errors May Be Incorrect
AR4	X	X	No Fix	VERW/VERR/LSL/LAR Instructions May Unexpectedly Update the Last Exception Record (LER) MSR
AR5	X	X	No Fix	DR3 Address Match on MOVD/MOVQ/MOVNTQ Memory Store Instruction May Incorrectly Increment Performance Monitoring Count for Saturating SIMD Instructions Retired (Event CFH)
AR6	X		Fixed	SYSRET May Incorrectly Clear RF (Resume Flag) in the RFLAGS Register
AR7	X	X	No Fix	General Protection Fault (#GP) for Instructions Greater than 15 Bytes May Be Preempted
AR8	X	X	No Fix	Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced before Higher Priority Interrupts
AR9	X	X	No Fix	The Processor May Report a #TS Instead of a #GP Fault
AR10	X	X	No Fix	A Write to an APIC Register Sometimes May Appear to Have Not Occurred
AR11	X	X	No Fix	Programming the Digital Thermal Sensor (DTS) Threshold May Cause Unexpected Thermal Interrupts
AR12	X	X	No Fix	Count Value for Performance-Monitoring Counter PMH_PAGE_WALK May Be Incorrect
AR13	X	X	No Fix	LER MSRs May Be Incorrectly Updated
AR14	X	X	No Fix	Performance Monitoring Events for Retired Instructions (C0H) May Not Be Accurate
AR15	X	X	No Fix	Performance Monitoring Event For Number Of Reference Cycles When The Processor Is Not Halted (3CH) Does Not Count According To The Specification
AR16		X	Fixed	Using 2M/4M Pages When A20M# Is Asserted May Result in Incorrect Address Translations
AR17	X	X	No Fix	Writing Shared Unaligned Data that Crosses a Cache Line without Proper Semaphores or Barriers May Expose a Memory Ordering Issue
AR18	X	X	No Fix	Code Segment Limit Violation May Occur On 4 Gigabyte Limit Check
AR19	X		Fixed	FP Inexact-Result Exception Flag May Not Be Set
AR20	X		Fixed	Global Pages in the Data Translation Look-Aside Buffer (DTLB) May Not Be Flushed by RSM instruction before Restoring the Architectural State from SMRAM

Number	Stepping	Stepping	Plans	ERRATA
	A1	E1		
AR21	X		Fixed	Sequential Code Fetch to Non-canonical Address May have Nondeterministic Results
AR22	X	X	No Fix	REP MOVS/STOS Executing with Fast Strings Enabled and Crossing Page Boundaries with Inconsistent Memory Types May Use an Incorrect Data Size or Lead to Memory-Ordering Violations.
AR23	X	X	No Fix	Some Bus Performance Monitoring Events May Not Count Local Events under Certain Conditions
AR24	X	X	No Fix	Premature Execution of a Load Operation Prior to Exception Handler Invocation
AR25	X	X	No Fix	General Protection (#GP) Fault May Not Be Signaled on Data Segment Limit Violation above 4-G Limit
AR26	X	X	No Fix	EIP May Be Incorrect after Shutdown in IA-32e Mode
AR27	X	X	No Fix	#GP Fault Is Not Generated on Writing IA32_MISC_ENABLE [34] When Execute Disable Is Not supported
AR28	X		Fixed	(E)CX May Get Incorrectly Updated Fast String REP MOVS or Fast String REP STOS with Large Data Structures
AR29	X		Fixed	Performance Monitoring Events for Retired Loads (CBH) and Instructions Retired (C0H) May Not Be Accurate
AR30	X	X	No Fix	Upper 32 bits of 'From' Address Reported through BTMs or BTSs May Be Incorrect
AR31	X		Fixed	Unsynchronized Cross-Modifying Code Operations Can Cause unexpected Instruction Execution Results
AR32	X	X	No Fix	MSRs Actual Frequency Clock Count (IA32_APERF) or Maximum Frequency Clock Count (IA32_MPERF) May Contain Incorrect Data after a Machine Check Exception (MCE)
AR33	X	X	No Fix	Incorrect Address Computed for Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update
AR34	X	X	No Fix	Split Locked Stores May Not Trigger the Monitoring Hardware
AR35	X		Fixed	REP CMPS/SCAS Operations May Terminate Early in 64-bit Mode When RCX >= 0X100000000
AR36	X		Fixed	FXSAVE/FXRSTOR Instructions which Store to the End of the Segment and Cause a Wrap to a Misaligned Base Address (Alignment <= 0x10h) May Cause FPU Instruction or Operand Pointer Corruption
AR37	X		Fixed	Cache Data Access Request from One Core Hitting a Modified Line in the L1 Data Cache of the Other Core May Cause Unpredictable System Behavior
AR38	X		Fixed	PREFETCHh Instruction Execution under Some Conditions May Lead to Processor Livelock
AR39	X		Fixed	PREFETCHh Instructions May Not Be Executed when Alignment Check (AC) Is Enabled
AR40	X		Fixed	Upper 32 Bits of the FPU Data (Operand) Pointer in the FXSAVE Memory Image May Be Unexpectedly All 1's after FXSAVE

Number	Stepping	Stepping	Plans	ERRATA
	A1	E1		
AR41	X		Fixed	Concurrent Multi-processor Writes to Non-dirty Page May Result in Unpredictable Behavior
AR42	X		Fixed	Performance Monitor IDLE_DURING_DIV (18h) Count May Not Be Accurate
AR43	X	X	No Fix	Values for LBR/BTS/BTM Will Be Incorrect after an Exit from SMM
AR44	X		Fixed	SYSCALL Immediately after Changing EFLAGS.TF May Not Behave According to the New EFLAGS.TF
AR45	X	X	No Fix	VM Bit Is Cleared on Second Fault Handled by Task Switch from Virtual-8086 (VM86)
AR46	X		Fixed	IA32_FMASK Is Reset during an INIT
AR47	X	X	No Fix	Code Breakpoint May Be Taken after POP SS Instruction If It Is followed by an Instruction That Faults
AR48	X	X	No Fix	Last Branch Records (LBR) Updates May Be Incorrect after a Task Switch
AR49	X	X	No Fix	IO_SMI Indication in SMRAM State Save Area May Be Set Incorrectly
AR50	X	X	No Fix	INIT Does Not Clear Global Entries in the TLB
AR51	X		Fixed	Using Memory Type Aliasing with Memory Types WB/WT May Lead to Unpredictable Behavior
AR52	X		Fixed	Update of Read/Write (R/W) or User/Supervisor (U/S) or Present (P) Bits without TLB Shootdown May Cause Unexpected Processor Behavior
AR53	X		Fixed	BTS Message May Be Lost When the STPCLK# Signal Is Active
AR54	X	X	No Fix	MOV To/From Debug Registers Causes Debug Exception
AR55	X	X	No Fix	EFLAGS Discrepancy on a Page Fault after a Multiprocessor TLB Shootdown
AR56	X	X	No Fix	LBR, BTS, BTM May Report a Wrong Address when an Exception/Interrupt Occurs in 64-bit Mode
AR57	X	X	No Fix	A Thermal Interrupt Is Not Generated when the Current Temperature Is Invalid
AR58	X	X	No Fix	CMPSB, LODSB, or SCASB in 64-bit Mode with Count Greater or Equal to 2^{48} May Terminate Early
AR59	X	X	No Fix	Returning to Real Mode from SMM with EFLAGS.VM Set May Result in Unpredictable System Behavior
AR60	X	X	No Fix	IRET under Certain Conditions May Cause an Unexpected Alignment Check Exception
AR61	X	X	No Fix	Performance Monitoring Event FP_ASSIST May Not Be Accurate
AR62	X		Fixed	CPL-Qualified BTS May Report Incorrect Branch-From Instruction Address
AR63	X		Fixed	PEBS Does Not Always Differentiate Between CPL-Qualified Events

Number	Stepping	Stepping	Plans	ERRATA
	A1	E1		
AR64	X	X	No Fix	PMI May Be Delayed to Next PEBS Event
AR65	X		Fixed	PEBS Buffer Overflow Status Will Not Be Indicated Unless IA32_DEBUGCTL[12] Is Set
AR66	X	X	No Fix	The BS Flag in DR6 May Be Set for Non-Single-Step #DB Exception
AR67	X	X	No Fix	An Asynchronous MCE during a Far Transfer May Corrupt ESP
AR68	X	X	No Fix	B0-B3 Bits in DR6 May Not Be Properly Cleared after Code Breakpoint
AR69	X	X	No Fix	BTM/BTS Branch-From Instruction Address May Be Incorrect for Software Interrupts
AR70	X		Fixed	REP Store Instructions in a Specific Situation May Cause the Processor to Hang
AR71	X	X	No Fix	Performance Monitor SSE Retired Instructions May Return Incorrect Values
AR72	X	X	No Fix	Performance Monitoring Events for L1 and L2 Miss May Not Be Accurate
AR73	X	X	No Fix	Store to WT Memory Data May Be Seen in Wrong Order by Two Subsequent Loads
AR74		X	No Fix	A MOV Instruction from CR8 Register with 16 Bit Operand Size Will Leave Bits 63:16 of the Destination Register Unmodified
AR75	X		Fixed	Debug Register May Contain Incorrect Information on a MOVSS or POPSS Instruction followed by SYSRET
AR76	X	X	No Fix	Single Step Interrupts with Floating Point Exception Pending May Be Mishandled
AR77	X	X	No Fix	Fault on ENTER Instruction May Result in Unexpected Values on Stack Frame
AR78	X	X	No Fix	Unaligned Accesses to Paging Structures May Cause the Processor to Hang
AR79	X	X	No Fix	INVLPG Operation for Large (2M/4M) Pages May Be Incomplete under Certain Conditions
AR80	X	X	No Fix	Page Access Bit May Be Set Prior to Signaling a Code Segment Limit Fault
AR81	X	X	Plan Fix	Update of Attribute Bits on Page Directories without Immediate TLB Shootdown May Cause Unexpected Processor Behavior
AR82	X		Fixed	Invalid Instructions May Lead to Unexpected Behavior
AR83	X	X	No Fix	EFLAGS, CR0, CR4 and the EXF4 Signal May Be Incorrect after Shutdown
AR84	X		Fixed	Performance Monitoring Counter MACRO_INSTS.DECODED May Not Count Some Decoded Instructions
AR85	X	X	Plan Fix	The Stack May Be Incorrect as a Result of VIP/VIF Check on SYSEXIT and SYSRET

Number	Stepping	Stepping	Plans	ERRATA
	A1	E1		
AR86	X	X	No Fix	Performance Monitoring Event SIMD_UOP_TYPE_EXEC.MUL is Counted Incorrectly for PMULUDQ Instruction
AR87	X	X	No Fix	Storage of PEBS Record Delayed Following Execution of MOV SS or STI
AR88	X	X	No Fix	Updating Code Page Directory Attributes without TLB Invalidation May Result in Improper Handling of Code #PF
AR89	X	X	Plan Fix	Performance Monitoring Event CPU_CLK_UNHALTED.REF May Not Count Clock Cycles According to the Processors Operating Frequency
AR90	X	X	Plan Fix	Store Ordering May Be Incorrect between WC and WP Memory Types
AR91	X		Fixed	Performance Monitoring Event BR_INST_RETIRED May Count CPUID Instructions as Branches
AR92	X	X	No Fix	Performance Monitoring Event MISALIGN_MEM_REF May over Count
AR93	X	X	No Fix	A REP STOS/MOVS to a MONITOR/MWAIT Address Range May Prevent Triggering of the Monitoring Hardware
AR94	X		Fixed	False Level One Data Cache Parity Machine-Check Exceptions May Be Signaled
AR95	X	X	No Fix	A Memory Access May Get a Wrong Memory Type Following a #GP due to WRMSR to an MTRR Mask
AR96	X	X	No Fix	PMI While LBR Freeze Enabled May Result in Old/Out-of-date LBR Information
AR97		x	No Fix	BIST Failure after Reset
AR99	X	X	No Fix	Instruction Fetch May Cause a Livelock during Snoops of the L1 Data Cache.
AR100	X	X	No Fix	Use of Memory Aliasing with Inconsistent Memory Type may Cause a System Hang or a Machine Check Exception
AR101	X	X	No Fix	A WB Store Following a REP STOS/MOVS or FXSAVE May Lead to Memory-Ordering Violations
AR102	X	X	No Fix	Using Memory Type Aliasing with Cacheable and WC Memory Types May Lead to Memory Ordering Violations
AR103	X	X	No Fix	RSM Instruction Execution under Certain Conditions May Cause Processor Hang or Unexpected Instruction Execution Results
AR104	X	X	Plan Fix	NMIs May Not Be Blocked by a VM-Entry Failure
AR105	X	X	No Fix	Benign Exception after a Double Fault May Not Cause a Triple Fault Shutdown

Number	SPECIFICATION CHANGES
	There are no Specification Changes in this Specification Update revision.

Number	SPECIFICATION CLARIFICATIONS
AR1	Clarification of Translation Lookaside Buffers (TLBS) Invalidation

Number	DOCUMENTATION CHANGES
	There are no Documentation Changes in this Specification Update revision.

§

Errata

AR1. Writing the Local Vector Table (LVT) When an Interrupt Is Pending May Cause an Unexpected Interrupt

Problem: If a local interrupt is pending when the LVT entry is written, an interrupt may be taken on the new interrupt vector even if the mask bit is set.

Implication: An interrupt may immediately be generated with the new vector when a LVT entry is written, even if the new LVT entry has the mask bit set. If there is no Interrupt Service Routine (ISR) set up for that vector the system will GP fault. If the ISR does not do an End of Interrupt (EOI) the bit for the vector will be left set in the in-service register and mask all interrupts at the same or lower priority.

Workaround: Any vector programmed into an LVT entry must have an ISR associated with it, even if that vector was programmed as masked. This ISR routine must do an EOI to clear any unexpected interrupts that may occur. The ISR associated with the spurious vector does not generate an EOI, therefore the spurious vector should not be used when writing the LVT.

Status: For the steppings affected, see the Summary Tables of Changes.

AR2. LOCK# Asserted During a Special Cycle Shutdown Transaction May Unexpectedly Deassert

Problem: During a processor shutdown transaction, when LOCK# is asserted and if a DEFER# is received during a snoop phase and the Locked transaction is pipelined on the front side bus (FSB), LOCK# may unexpectedly deassert.

Implication: When this erratum occurs, the system may hang during shutdown. Intel has not observed this erratum with any commercially available systems or software.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR3. Address Reported by Machine-Check Architecture (MCA) on Single-bit L2 ECC Errors May Be Incorrect

Problem: When correctable Single-bit ECC errors occur in the L2 cache, the address is logged in the MCA address register (MCi_ADDR). Under some scenarios, the address reported may be incorrect.

Implication: Software should not rely on the value reported in MCi_ADDR, for Single-bit L2 ECC errors.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR4. VERW/VERR/LSL/LAR Instructions May Unexpectedly Update the Last Exception Record (LER) MSR

Problem: The LER MSR may be unexpectedly updated, if the resultant value of the Zero Flag (ZF) is zero after executing the following instructions.

1. VERR (ZF=0 indicates unsuccessful segment read verification)
2. VERW (ZF=0 indicates unsuccessful segment write verification)
3. LAR (ZF=0 indicates unsuccessful access rights load)
4. LSL (ZF=0 indicates unsuccessful segment limit load)

Implication: The value of the LER MSR may be inaccurate if VERW/VERR/LSL/LAR instructions are executed after the occurrence of an exception.

Workaround: Software exception handlers that rely on the LER MSR value should read the LER MSR before executing VERW/VERR/LSL/LAR instructions.

Status: For the steppings affected, see the Summary Tables of Changes.

AR5. DR3 Address Match on MOVD/MOVQ/MOVNTQ Memory Store Instruction May Incorrectly Increment Performance Monitoring Count for Saturating SIMD Instructions Retired (Event CFH)

Problem: Performance monitoring for Event CFH normally increments on saturating SIMD instruction retired. Regardless of DR7 programming, if the linear address of a retiring memory store MOVD/MOVQ/MOVNTQ instruction executed matches the address in DR3, the CFH counter may be incorrectly incremented.

Implication: The value observed for performance monitoring count for saturating SIMD instructions retired may be too high. The size of the error is dependent on the number of occurrences of the conditions described above, while the counter is active.

None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR6. SYSRET May Incorrectly Clear RF (Resume Flag) in the RFLAGS Register

Problem: In normal operation, SYSRET will restore the value of RFLAGS from R11 (the value previously saved upon execution of the SYSCALL instruction). Due to this erratum, the RFLAGS.RF bit will be unconditionally cleared after execution of the SYSRET instruction.

Implication: The SYSRET instruction can not be used if the RF flag needs to be set after returning from a system call. Intel has not observed this erratum with any commercially available software.

Workaround: Use the IRET instruction to return from a system call, if RF flag has to be set after the return.

Status: For the steppings affected, see the Summary Tables of Changes.

AR7. General Protection Fault (#GP) for Instructions Greater Than 15 Bytes May Be Preempted

Problem: When the processor encounters an instruction that is greater than 15 bytes in length, a #GP is signaled when the instruction is decoded. Under some circumstances, the #GP fault may be preempted by another lower priority fault (for example, Page Fault (#PF)). However, if the preempting lower priority faults are resolved by the operating system and the instruction retried, a #GP fault will occur.

Implication: Software may observe a lower-priority fault occurring before or in lieu of a #GP fault. Instructions of greater than 15 bytes in length can only occur if redundant prefixes are placed before the instruction.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR8. Pending x87 FPU Exceptions (#MF) Following STI May Be Serviced Before Higher Priority Interrupts

Problem: Interrupts that are pending prior to the execution of the STI (Set Interrupt Flag) instruction are serviced immediately after the STI instruction is executed. Because of this erratum, if following STI, an instruction that triggers a #MF is executed while STPCLK#, Enhanced Intel SpeedStep® Technology transitions or Thermal Monitor 1 events occur, the pending #MF may be serviced before higher priority interrupts.

Implication: Software may observe #MF being serviced before higher priority interrupts.

Workaround: None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR9. The Processor May Report a #TS Instead of a #GP Fault

Problem: A jump to a busy TSS (Task-State Segment) may cause a #TS (invalid TSS exception) instead of a #GP fault (general protection exception).

Implication: Operation systems that access a busy TSS may get invalid TSS fault instead of a #GP fault. Intel has not observed this erratum with any commercially available software.

Workaround: None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR10. A Write to an APIC Register Sometimes May Appear to Have Not Occurred

Problem: With respect to the retirement of instructions, stores to the uncacheable memory-based APIC register space are handled in a non-synchronized way. For example if an instruction that masks the interrupt flag, for example. CLI, is executed soon after an uncacheable write to the Task Priority Register (TPR) that lowers the APIC priority, the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR, but higher than the final TPR, to not be serviced until the interrupt enabled flag is finally set, that is. by STI instruction. Interrupts will remain pending and are not lost.

Implication: In this example the processor may allow interrupts to be accepted but may delay their service.

Workaround: This non-synchronization can be avoided by issuing an APIC register read after the APIC register write. This will force the store to the APIC register before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AR11. Programming the Digital Thermal Sensor (DTS) Threshold May Cause Unexpected Thermal Interrupts

Problem: Software can enable DTS thermal interrupts by programming the thermal threshold and setting the respective thermal interrupt enable bit. When programming DTS value, the previous DTS threshold may be crossed. This generates an unexpected thermal interrupt.

Implication: Software may observe an unexpected thermal interrupt occur after reprogramming the thermal threshold.

Workaround: In the ACPI/OS implement a workaround by temporarily disabling the DTS threshold interrupt before updating the DTS threshold value.

Status: For the steppings affected, see the Summary Tables of Changes.

AR12. Count Value for Performance-Monitoring Counter PMH_PAGE_WALK May Be Incorrect

Problem: Performance-Monitoring Counter PMH_PAGE_WALK is used to count the number of page walks resulting from Data Translation Look-Aside Buffer (DTLB) and Instruction Translation Look-Aside (ITLB) misses. Under certain conditions, this counter may be incorrect.

Implication: There may be small errors in the accuracy of the counter.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR13. LER MSRs May Be Incorrectly Updated

Problem: The LER (Last Exception Record) MSRs, MSR_LER_FROM_LIP (1DDH) and MSR_LER_TO_LIP (1DEH) may contain incorrect values after any of the following:

- Either STPCLK#, NMI (Non-Maskable Interrupt) or external interrupts
- CMP or TEST instructions with an uncacheable memory operand followed by a conditional jump
- STI/POP SS/MOV SS instructions followed by CMP or TEST instructions and then by a conditional jump.

Implication: When the conditions for this erratum occur, the value of the LER MSRs may be incorrectly updated.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR14. Performance Monitoring Events for Retired Instructions (C0H) May Not Be Accurate

Problem: The INST_RETIRED performance monitor may miscount retired instructions as follows:

- Repeat string and repeat I/O operations are not counted when a hardware interrupt is received during or after the last iteration of the repeat flow.
- VMLAUNCH and VMRESUME instructions are not counted.
- HLT and MWAIT instructions are not counted.

The following instructions, if executed during HLT or MWAIT events, are also not counted:

1. RSM from a C-state SMI during an MWAIT instruction.
2. RSM from an SMI during a HLT instruction.

Implication: There may be a smaller than expected value in the INST_RETIRED performance monitoring counter. The extent to which this value is smaller than expected is determined by the frequency of the above cases.

Workaround: None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR15. Performance Monitoring Event for Number of Reference Cycles When the Processor Is Not Halted (3CH) Does Not Count According to the Specification

Problem: The CPU_CLK_UNHALTED performance monitor with mask 1 counts bus clock cycles instead of counting the core clock cycles at the maximum possible ratio. The maximum possible ratio is computed by dividing the maximum possible core frequency by the bus frequency.

Implication: The CPU_CLK_UNHALTED performance monitor with mask 1 counts a value lower than expected. The value is lower by exactly one multiple of the maximum possible ratio.

Workaround: Multiply the performance monitor value by the maximum possible ratio.

Status: For the steppings affected, see the Summary Tables of Changes.

AR16. Using 2M/4M Pages When A20M# Is Asserted May Result in Incorrect Address Translations

Problem: An external A20M# pin if enabled forces address bit 20 to be masked (forced to zero) to emulates real-address mode address wraparound at 1 megabyte. However, if all of the following conditions are met, address bit 20 may not be masked:

- Paging is enabled
- A linear address has bit 20 set
- The address references a large page
- A20M# is enabled

Implication: When A20M# is enabled and an address references a large page the resulting translated physical address may be incorrect. This erratum has not been observed with any commercially available operating system.

Workaround: Operating systems should not allow A20M# to be enabled if the masking of address bit 20 could be applied to an address that references a large page. A20M# is normally only used with the first megabyte of memory.

Status: For the steppings affected, see the Summary Tables of Changes.

AR17. Writing Shared Unaligned Data that Crosses a Cache Line without Proper Semaphores or Barriers May Expose a Memory Ordering Issue

Problem: Software which is written so that multiple agents can modify the same shared unaligned memory location at the same time may experience a memory ordering issue if multiple loads access this shared data shortly thereafter. Exposure to this problem requires the use of a data write which spans a cache line boundary.

Implication: This erratum may cause loads to be observed out of order. Intel has not observed this erratum with any commercially available software or system.

Workaround: Software should ensure at least one of the following is true when modifying shared data by multiple agents:

- The shared data is aligned.
- Proper semaphores or barriers are used in order to prevent concurrent data accesses.

Status: For the steppings affected, see the Summary Tables of Changes.

AR18. Code Segment Limit Violation May Occur On 4 Gigabyte Limit Check

Problem: Code Segment limit violation may occur on 4 Gigabyte limit check when the code stream wraps around in a way that one instruction ends at the last byte of the segment and the next instruction begins at 0x0.

Implication: This is a rare condition that may result in a system hang. Intel has not observed this erratum with any commercially available software, or system.

Workaround: Avoid code that wraps around segment limit.

Status: For the steppings affected, see the Summary Tables of Changes.

AR19. FP Inexact-Result Exception Flag May Not Be Set

Problem: When the result of a floating-point operation is not exactly representable in the destination format (1/3 in binary form, for example), an inexact-result (precision) exception occurs. When this occurs, the PE bit (bit 5 of the FPU status word) is normally set by the processor. Under certain rare conditions, this bit may not be set when this rounding occurs. However, other actions taken by the processor (invoking the software exception handler if the exception is unmasked) are not affected. This erratum can only occur if one of the following FST instructions is one or two instructions after the floating-point operation which causes the precision exception:

- FST m32real
- FST m64real
- FSTP m32real
- FSTP m64real
- FSTP m80real
- FIST m16int
- FIST m32int
- FISTP m16int
- FISTP m32int
- FISTP m64int
- FISTTP m16int
- FISTTP m32int
- FISTTP m64int

Note: Even if this combination of instructions is encountered, there is also a dependency on the internal pipelining and execution state of both instructions in the processor.

Implication: Inexact-result exceptions are commonly masked or ignored by applications, as it happens frequently, and produces a rounded result acceptable to most applications. The PE bit of the FPU status word may not always be set upon receiving an inexact-result exception. Thus, if these exceptions are unmasked, a floating-point error exception handler may not recognize that a precision exception occurred. Note that this is a "sticky" bit, that is, once set by an inexact-result condition; it remains set until cleared by software.

Workaround: This condition can be avoided by inserting either three NOPs or three non-floating-point non-Jcc instructions between the two floating-point instructions.

Status: For the steppings affected, see the Summary Tables of Changes.

AR20 Global Pages in the Data Translation Look-Aside Buffer (DTLB) May Not Be Flushed by RSM instruction before Restoring the Architectural State from SMRAM

Problem: The Resume from System Management Mode (RSM) instruction does not flush global pages from the Data Translation Look-Aside Buffer (DTLB) prior to reloading the saved architectural state.

Implication: If SMM turns on paging with global paging enabled and then maps any of linear addresses of SMRAM using global pages, RSM load may load data from the wrong location.

Workaround: Do not use global pages in system management mode.

Status: For the steppings affected, see the Summary Tables of Changes.

AR21 Sequential Code Fetch to Non-canonical Address May Have Nondeterministic Results

Problem: If code sequentially executes off the end of the positive canonical address space (falling through from address 00007fffffffffff to non- canonical address 0000800000000000), under some circumstances the code fetch will be converted to a canonical fetch at address ffff800000000000.

Implication: Due to this erratum, the processor may transfer control to an unintended address. The result of fetching code at that address is unpredictable and may include an unexpected trap or fault, or execution of the instructions found there.

Workaround: If the last page of the positive canonical address space is not allocated for code (4K page at 00007ffffffff000 or 2M page at 00007fffffe00000) then the problem cannot occur.

Status: For the steppings affected, see the Summary Tables of Changes.

AR22. REP MOVS/STOS Executing with Fast Strings Enabled and Crossing Page Boundaries with Inconsistent Memory Types May Use an Incorrect Data Size or Lead to Memory-Ordering Violations.

Problem: Under certain conditions as described in the Software Developers Manual section "Out-of-Order Stores For string operations in Pentium 4, Intel Xeon, and P6 Family Processors" the processor performs REP MOVS or REP STOS as fast strings. Due to this erratum fast string REP MOVS/REP STOS instructions that cross page boundaries from WB/WC memory types to UC/WP/WT memory types, may start using an incorrect data size or may observe memory ordering violations.

Implication: Upon crossing the page boundary the following may occur, dependent on the new page memory type:

1. UC the data size of each write will now always be 8 bytes, as opposed to the original data size.
2. WP the data size of each write will now always be 8 bytes, as opposed to the original data size and there may be a memory ordering violation.
3. WT there may be a memory ordering violation.

Workaround: Software should avoid crossing page boundaries from WB or WC memory type to UC, WP or WT memory type within a single REP MOVS or REP STOS instruction that will execute with fast strings enabled.

Status: For the steppings affected, see the Summary Tables of Changes.

AR23. Some Bus Performance Monitoring Events May Not Count Local Events under Certain Conditions

Problem: Many Performance Monitoring Events require core-specificity, which specifies which core's events are to be counted (local core, other core, or both cores). Due to this erratum, some Bus Performance Monitoring events may not count when the core-specificity is set to the local core.

The following Bus Transaction Performance Monitor events are supposed to count all local transactions:

- BUS_TRANS_ IO (Event: 6CH) – Will not count I/O level reads resulting from package resolved C-state
- BUS_TRANS_ANY (Event: 70H) – Will not count Stop-Grants

Implication: The count values for the affected events may be lower than expected. The degree of under count depends on the occurrence of erratum conditions while the affected events are active.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR24. Premature Execution of a Load Operation Prior to Exception Handler Invocation

Problem: If any of the below circumstances occur it is possible that the load portion of the instruction is executed before the exception handler is entered.

1. If an instruction that performs a memory load causes a code segment limit violation.
2. If a waiting X87 floating-point (FP) instruction or MMX™ technology instruction that performs a memory load has a floating-point exception pending.
3. If an MMX or SSE/SSE2/SSE3/SSSE3 extensions (SSE) instruction that performs a memory load and has either CR0.EM=1 (Emulation bit set), or a floating-point Top-of-Stack (FP TOS) not equal to 0, or a DNA exception pending.

Implication: In normal code execution where the target of the load operation is to write back memory there is no impact from the load being prematurely executed, or from the restart and subsequent re-execution of that instruction by the exception handler. If the target of the load is to uncached memory that has a system side-effect. Particularly, while CR0.TS [bit 3] is set, a MOVD/MOVQ with MMX/XMM register operands may issue a memory load before getting the DNA exception.

Workaround: Code which performs loads from memory that has side-effects can effectively workaround this behavior by using simple integer-based load instructions when accessing side-effect memory and by ensuring that all code is written such that a code segment limit violation cannot occur as a part of reading from side-effect memory.

Status: For the steppings affected, see the Summary Tables of Changes.

AR25. General Protection (#GP) Fault May Not Be Signaled on Data Segment Limit Violation above 4-G Limit

Problem: In 32-bit mode, memory accesses to flat data segments (base = 00000000h) that occur above the 4G limit (0ffffffffh) may not signal a #GP fault.

Implication: When such memory accesses occur in 32-bit mode, the system may not issue a #GP fault.

Workaround: Software should ensure that memory accesses in 32-bit mode do not occur above the 4G limit (0ffffffffh).

Status: For the steppings affected, see the Summary Tables of Changes.

AR26. EIP May Be Incorrect after Shutdown in IA-32e Mode

Problem: When the processor is going into shutdown state the upper 32 bits of the instruction pointer may be incorrect. This may be observed if the processor is taken out of shutdown state by NMI#.

Implication: A processor that has been taken out of the shutdown state may have an incorrect EIP. The only software which would be affected is diagnostic software that relies on a valid EIP.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR27. #GP Fault is Not Generated on Writing IA32_MISC_ENABLE [34] When Execute Disable Is Not Supported

Problem: A #GP fault is not generated on writing to IA32_MISC_ENABLE [34] bit in a processor which does not support Execute Disable functionality.

Implication: Writing to IA32_MISC_ENABLE [34] bit is silently ignored without generating a fault.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR28. (E)CX May Get Incorrectly Updated When Performing Fast String REP MOVS or Fast String REP STOS with Large Data Structures

Problem: When performing Fast String REP MOVS or REP STOS commands with data structures [(E)CX*Data Size] larger than the supported address size structure (64K for 16-bit address size and 4G for 32-bit address size) some addresses may be processed more than once. After an amount of data greater than or equal to the address size structure has been processed, external events (such as interrupts) will cause the (E)CX registers to be incremented by a value that corresponds to 64K bytes for 16 bit address size and 4G bytes for 32 bit address size.

Implication: (E)CX may contain an incorrect count which may cause some of the MOVS or STOS operations to re-execute. Intel has not observed this erratum with any commercially available software.

Workaround: Do not use values in (E)CX that when multiplied by the data size give values larger than the address space size (64K for 16-bit address size and 4G for 32-bit address size).

Status: For the steppings affected, see the Summary Tables of Changes.

AR29. Performance Monitoring Events for Retired Loads (CBH) and Instructions Retired (C0H) May Not Be Accurate

Problem: The following events may be counted as instructions that contain a load by the MEM_LOAD_RETIRED performance monitor events and may be counted as loads by the INST_RETIRED (mask 01H) performance monitor event:

- Prefetch instructions.
- x87 exceptions on FST* and FBSTP instructions.
- Breakpoint matches on loads, stores, and I/O instructions.
- Stores which update the A and D bits.
- Stores that split across a cache line.
- VMX transitions.
- Any instructions fetch that misses in the ITLB.

Implication: The MEM_LOAD_RETIRED and INST_RETIRED (mask 01H) performance monitor events may count a value higher than expected. The extent to which the values are higher than expected is determined by the frequency of the above events.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR30. Upper 32 bits of 'From' Address Reported through BTMs or BTSs May Be Incorrect

Problem: When a far transfer switches the processor from 32-bit mode to IA-32e mode, the upper 32 bits of the 'From' (source) addresses reported through the BTMs (Branch Trace Messages) or BTSs (Branch Trace Stores) may be incorrect.

Implication: The upper 32 bits of the 'From' address debug information reported through BTMs or BTSs may be incorrect during this transition.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR31. Unsynchronized Cross-Modifying Code Operations Can Cause Unexpected Instruction Execution Results

Problem: The act of one processor, or system bus master, writing data into a currently executing code segment of a second processor with the intent of having the second processor execute that data as code is called cross-modifying code (XMC). XMC that does not force the second processor to execute a synchronizing instruction, prior to execution of the new code, is called unsynchronized XMC. Software using unsynchronized XMC to modify the instruction byte stream of a processor can see unexpected or unpredictable execution behavior from the processor that is executing the modified code.

Implication: In this case, the phrase "unexpected or unpredictable execution behavior" encompasses the generation of most of the exceptions listed in the *Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 3A: System Programming Guide*, including a General Protection Fault (GPF) or other unexpected behaviors. In the event that unpredictable execution causes a GPF the application executing the unsynchronized XMC operation would be terminated by the operating system.

Workaround: In order to avoid this erratum, programmers should use the XMC synchronization algorithm as detailed in the Intel Architecture Software Developer's Manual Volume 3: System Programming Guide, Section: Handling Self- and Cross-Modifying Code.

Status: For the steppings affected, see the Summary Tables of Changes.

AR32. MSRs Actual Frequency Clock Count (IA32_APERF) or Maximum Frequency Clock Count (IA32_MPERF) May Contain Incorrect Data after a Machine Check Exception (MCE)

Problem: When an MCE occurs during execution of a RDMSR instruction for MSRs Actual Frequency Clock Count (IA32_APERF) or Maximum Frequency Clock Count (IA32_MPERF), the current and subsequent RDMSR instructions for these MSRs may contain incorrect data.

Implication: After an MCE event, accesses to the IA32_APERF and IA32_MPERF MSRs may return incorrect data. A subsequent reset will clear this condition.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR33. Incorrect Address Computed for Last Byte of FXSAVE/FXRSTOR Image Leads to Partial Memory Update

Problem: A partial memory state save of the 512-byte FXSAVE image or a partial memory state restore of the FXRSTOR image may occur if a memory address exceeds the 64KB limit while the processor is operating in 16-bit mode or if a memory address exceeds the 4GB limit while the processor is operating in 32-bit mode.

Implication: FXSAVE/FXRSTOR will incur a #GP fault due to the memory limit violation as expected but the memory state may be only partially saved or restored.

Workaround: Software should avoid memory accesses that wrap around the respective 16-bit and 32-bit mode memory limits.

Status: For the steppings affected, see the Summary Tables of Changes.

AR34. Split Locked Stores May Not Trigger the Monitoring Hardware

Problem: Logical processors normally resume program execution following the MWAIT, when another logical processor performs a write access to a WB cacheable address within the address range used to perform the MONITOR operation. Due to this erratum, a logical processor may not resume execution until the next targeted interrupt event or O/S timer tick following a locked store that spans across cache lines within the monitored address range.

Implication: The logical processor that executed the MWAIT instruction may not resume execution until the next targeted interrupt event or O/S timer tick in the case where the monitored address is written by a locked store which is split across cache lines.

Workaround: Do not use locked stores that span cache lines in the monitored address range.

Status: For the steppings affected, see the Summary Tables of Changes.

AR35. REP CMPS/SCAS Operations May Terminate Early in 64-bit Mode When RCX >= 0X100000000

Problem: REP CMPS (Compare String) and SCAS (Scan String) instructions in 64-bit mode may terminate before the count in RCX reaches zero if the initial value of RCX is greater than or equal to 0X100000000.

Implication: Early termination of REP CMPS/SCAS operation may be observed and RFLAGS may be incorrectly updated.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AR36. FXSAVE/FXRSTOR Instructions Which Store to the End of the Segment and Cause a Wrap to a Misaligned Base Address (Alignment <= 0x10h) May Cause FPU Instruction or Operand Pointer Corruption

Problem: If a FXSAVE/FXRSTOR instruction stores to the end of the segment causing a wrap to a misaligned base address (alignment <= 0x10h), and one of the following conditions is satisfied:

1. 32-bit addressing, obtained by using address-size override, when in 64-bit mode.
2. 16-bit addressing in legacy or compatibility mode.

Then, depending on the wrap-around point, one of the below saved values may be corrupted:

- FPU Instruction Pointer Offset
- FPU Instruction Pointer Selector
- FPU Operand Pointer Selector
- FPU Operand Pointer Offset

Implication: This erratum could cause FPU instruction or operand pointer corruption and may lead to unexpected operations in the floating point exception handler.

Workaround: Avoid segment base mis-alignment and address wrap-around at the segment boundary.

Status: For the steppings affected, see the Summary Tables of Changes.

AR37. Cache Data Access Request from One Core Hitting a Modified Line in the L1 Data Cache of the Other Core May Cause Unpredictable System Behavior

Problem: When request for data from Core 1 results in a L1 cache miss, the request is sent to the L2 cache. If this request hits a modified line in the L1 data cache of Core 2, certain internal conditions may cause incorrect data to be returned to the Core 1.

Implication: This erratum may cause unpredictable system behavior.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AR38. PREFETCHh Instruction Execution under Some Conditions May Lead to Processor Livelock

Problem: PREFETCHh instruction execution after a split load and dependent upon ongoing store operations may lead to processor livelock.

Implication: Due to this erratum, the processor may livelock.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AR39. PREFETCHh Instructions May Not Be Executed When Alignment Check (AC) Is Enabled

Problem: PREFETCHT0, PREFETCHT1, PREFETCHT2 and PREFETCHNTA instructions may not be executed when alignment check is enabled.

Implication: PREFETCHh instructions may not perform the data prefetch if Alignment Check is enabled.

Workaround: Clear the AC flag (bit 18) in the EFLAGS register and/or the AM bit (bit 18) of Control Register CR0 to disable alignment checking.

Status: For the steppings affected, see the Summary Tables of Changes.

AR40. Upper 32 Bits of the FPU Data (Operand) Pointer in the FXSAVE Memory Image May Be Unexpectedly All 1's after FXSAVE

Problem: The upper 32 bits of the FPU Data (Operand) Pointer may incorrectly be set to all 1's instead of the expected value of all 0's in the FXSAVE memory image if all of the following conditions are true:

- The processor is in 64-bit mode.
- The last floating point operation was in compatibility mode
- Bit 31 of the FPU Data (Operand) Pointer is set.
- An FXSAVE instruction is executed

Implication: Software depending on the full FPU Data (Operand) Pointer may behave unpredictably.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR41. Concurrent Multi-processor Writes to Non-dirty Page May Result in Unpredictable Behavior

Problem: When a logical processor writes to a non-dirty page, and another logical-processor either writes to the same non-dirty page or explicitly sets the dirty bit in the corresponding page table entry, complex interaction with internal processor activity may cause unpredictable system behavior.

Implication: This erratum may result in unpredictable system behavior and hang.

Workaround: It is possible for BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AR42. Performance Monitor IDLE_DURING_DIV (18h) Count May Not Be Accurate

Problem: Performance monitoring events that count the number of cycles the divider is busy and no other execution unit operation or load operation is in progress may not be accurate.

Implication: The counter may reflect a value higher or lower than the actual number of events.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR43. Values for LBR/BTS/BTM Will Be Incorrect after an Exit from SMM

Problem: After a return from SMM (System Management Mode), the CPU will incorrectly update the LBR (Last Branch Record) and the BTS (Branch Trace Store), hence rendering their data invalid. The corresponding data if sent out as a BTM on the system bus will also be incorrect.

Note: This issue would only occur when one of the 3 above mentioned debug support facilities are used.

Implication: The value of the LBR, BTS, and BTM immediately after an RSM operation should not be used.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR44. SYSCALL Immediately after Changing EFLAGS.TF May Not Behave According to the New EFLAGS.TF

Problem: If a SYSCALL instruction follows immediately after EFLAGS.TF was updated and IA32_FMASK.TF (bit 8) is cleared, then under certain circumstances SYSCALL may behave according to the previous EFLAGS.TF.

Implication: When the problem occurs, SYSCALL may generate an unexpected debug exception, or may skip an expected debug exception.

Workaround: Mask EFLAGS.TF by setting IA32_FMASK.TF (bit 8).

Status: For the steppings affected, see the Summary Tables of Changes.

AR45. VM Bit is Cleared on Second Fault Handled by Task Switch from Virtual- 8086 (VM86)

Problem: Following a task switch to any fault handler that was initiated while the processor was in VM86 mode, if there is an additional fault while servicing the original task switch then the VM bit will be incorrectly cleared in EFLAGS, data segments will not be pushed and the processor will not return to the correct mode upon completion of the second fault handler via IRET.

Implication: When the OS recovers from the second fault handler, the processor will no longer be in VM86 mode. Normally, operating systems should prevent interrupt task switches from faulting, thus the scenario should not occur under normal circumstances.

Workaround: None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR46. IA32_FMASK Is Reset during an INIT

Problem: IA32_FMASK MSR (0xC0000084) is reset during INIT.

Implication: If an INIT takes place after IA32_FMASK is programmed, the processor will overwrite the value back to the default value.

Workaround: Operating system software should initialize IA32_FMASK after INIT.

Status: For the steppings affected, see the Summary Tables of Changes.

AR47. Code Breakpoint May Be Taken after POP SS Instruction If It Is followed by an Instruction That Faults

Problem: A POP SS instruction should inhibit all interrupts including Code Breakpoints until after execution of the following instruction. This allows sequential execution of POP SS and MOV eSP, eBP instructions without having an invalid stack during interrupt handling. However, a code breakpoint may be taken after POP SS if it is followed by an instruction that faults, this result in a code breakpoint being reported on an unexpected instruction boundary since both instructions should be atomic.

Implication: This can result in a mismatched Stack Segment and SP. Intel has not observed this erratum with any commercially available software, or system.

Workaround: As recommended in the *Intel® 64 and IA-32 Architecture Software Developer's Manual*, the use "POP SS" in conjunction with "MOV eSP, eBP" will avoid the failure since the "MOV" will not fault.

Status: For the steppings affected, see the Summary Tables of Changes.

AR48. Last Branch Records (LBR) Updates May Be Incorrect after a Task Switch

Problem: A Task-State Segment (TSS) task switch may incorrectly set the LBR_FROM value to the LBR_TO value.

Implication: The LBR_FROM will have the incorrect address of the Branch Instruction.

Workaround: None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR49. IO_SMI Indication in SMRAM State Save Area May Be Set Incorrectly

Problem: The IO_SMI bit in SMRAM's location 7FA4H is set to "1" by the CPU to indicate a System Management Interrupt (SMI) occurred as the result of executing an instruction that reads from an I/O port. Due to this erratum, the IO_SMI bit may be incorrectly set by:

- A non-I/O instruction.
- SMI is pending while a lower priority event interrupts.
- A REP I/O read.
- An I/O read that redirects to MWAIT.
- In systems supporting Intel® Virtualization Technology a fault in the middle of an IO operation that causes a VM Exit

Implication: SMM handlers may get false IO_SMI indication.

Workaround: The SMM handler has to evaluate the saved context to determine if the SMI was triggered by an instruction that read from an I/O port. The SMM handler must not restart an I/O instruction if the platform has not been configured to generate a synchronous SMI for the recorded I/O port address.

Status: For the steppings affected, see the Summary Tables of Changes.

AR50. INIT Does Not Clear Global Entries in the TLB

Problem: INIT may not flush a TLB entry when:

- The processor is in protected mode with paging enabled and the page global enable flag is set (PGE bit of CR4 register).
- G bit for the page table entry is set.
- TLB entry is present in TLB when INIT occurs.

Implication: Software may encounter unexpected page fault or incorrect address translation due to a TLB entry erroneously left in TLB after INIT.

Workaround: Write to CR3, CR4 (setting bits PSE, PGE or PAE) or CR0 (setting bits PG or PE) registers before writing to memory early in BIOS code to clear all the global entries from TLB.

Status: For the steppings affected, see the Summary Tables of Changes.

AR51. Using Memory Type Aliasing with Memory Types WB/WT May Lead to Unpredictable Behavior

Problem: Memory type aliasing occurs when a single physical page is mapped to two or more different linear addresses, each with different memory type. Memory type aliasing with the memory types WB and WT may cause the processor to perform incorrect operations leading to unpredictable behavior.

Implication: Software that uses aliasing of WB and WT memory types may observe unpredictable behavior.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AR52. Update of Read/Write (R/W) or User/Supervisor (U/S) or Present (P) Bits without TLB Shootdown May Cause Unexpected Processor Behavior

Problem: Updating a page table entry by changing R/W, U/S or P bits without TLB shootdown (as defined by the 4 step procedure in *Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 3A: System Programming Guide*), in conjunction with a complex sequence of internal processor micro-architectural events, may lead to unexpected processor behavior.

Implication: This erratum may lead to livelock, shutdown or other unexpected processor behavior. Intel has not observed this erratum with any commercially available system.

Workaround: None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR53. BTS Message May Be Lost When the STPCLK# Signal Is Active.

Problem: STPCLK# is asserted to enable the processor to enter a low-power state. Under some circumstances, when STPCLK# becomes active, the BTS (Branch Trace Store) message may be either lost and not written or written with corrupted branch address to the Debug Store area.

Implication: BTS messages may be lost or be corrupted in the presence of STPCLK# assertions.

Workaround: None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR54. MOV To/From Debug Registers Causes Debug Exception

Problem: When in V86 mode, if a MOV instruction is executed to/from a debug register, a general-protection exception (#GP) should be generated. However, in the case when the general detect enable flag (GD) bit is set, the observed behavior is that a debug exception (#DB) is generated instead.

Implication: With debug-register protection enabled (that is, the GD bit set), when attempting to execute a MOV on debug registers in V86 mode, a debug exception will be generated instead of the expected general-protection fault.

Workaround: In general, operating systems do not set the GD bit when they are in V86 mode. The GD bit is generally set and used by debuggers. The debug exception handler should check that the exception did not occur in V86 mode before continuing. If the exception did occur in V86 mode, the exception may be directed to the general-protection exception handler.

Status: For the steppings affected, see the Summary Tables of Changes.

AR55. EFLAGS Discrepancy on a Page Fault after a Multiprocessor TLB Shootdown

Problem: This erratum may occur when the processor executes one of the following read-modify-write arithmetic instructions and a page fault occurs during the store of the memory operand: ADD, AND, BTC, BTR, BTS, CMPXCHG, DEC, INC, NEG, NOT, OR, ROL/ROR, SAL/SAR/SHL/SHR, SHLD, SHRD, SUB, XOR, and XADD. In this case, the EFLAGS value pushed onto the stack of the page fault handler may reflect the status of the register after the instruction would have completed execution rather than before it. The following conditions are required for the store to generate a page fault and call the operating system page fault handler:

1. The store address entry must be evicted from the DTLB by speculative loads from other instructions that hit the same way of the DTLB before the store has completed. DTLB eviction requires at least three-load operations that have linear address bits 15:12 equal to each other and address bits 31:16 different from each other in close physical proximity to the arithmetic operation.
2. The page table entry for the store address must have its permissions tightened during the very small window of time between the DTLB eviction and execution of the store. Examples of page permission tightening include from Present to Not Present or from Read/Write to Read Only, etc. 3. Another processor, without corresponding synchronization and TLB flush, must cause the permission change.

Implication: This scenario may only occur on a multiprocessor platform running an operating system that performs "lazy" TLB shootdowns. The memory image of the EFLAGS register on the page fault handler's stack prematurely contains the final arithmetic flag values although the instruction has not yet completed. Intel has not identified any operating systems that inspect the arithmetic portion of the EFLAGS register during a page fault nor observed this erratum in laboratory testing of software applications.

Workaround: No workaround is needed upon normal restart of the instruction, since this erratum is transparent to the faulting code and results in correct instruction behavior. Operating systems may ensure that no processor is currently accessing a page that is scheduled to have its page permissions tightened or have a page fault handler that ignores any incorrect state.

Status: For the steppings affected, see the Summary Tables of Changes.

AR56. LBR, BTS, BTM May Report a Wrong Address When an Exception / Interrupt Occurs in 64-bit Mode

Problem: An exception/interrupt event should be transparent to the LBR (Last Branch Record), BTS (Branch Trace Store) and BTM (Branch Trace Message) mechanisms. However, during a specific boundary condition where the exception/interrupt occurs right after the execution of an instruction at the lower canonical boundary (0x00007FFFFFFFFFFF) in 64-bit mode, the LBR return registers will save a wrong return address with bits 63 to 48 incorrectly sign extended to all 1's. Subsequent BTS and BTM operations which report the LBR will also be incorrect.

Implication: LBR, BTS and BTM may report incorrect information in the event of an exception/interrupt.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR57. A Thermal Interrupt Is Not Generated When the Current Temperature Is Invalid

Problem: When the DTS (Digital Thermal Sensor) crosses one of its programmed thresholds it generates an interrupt and logs the event (IA32_THERM_STATUS MSR (019Ch) bits [9, 7]). Due to this erratum, if the DTS reaches an invalid temperature (as indicated IA32_THERM_STATUS MSR bit[31]) it does not generate an interrupt even if one of the programmed thresholds is crossed and the corresponding log bits become set.

Implication: When the temperature reaches an invalid temperature the CPU does not generate a Thermal interrupt even if a programmed threshold is crossed.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR58. CMPSB, LODSB, or SCASB in 64-bit Mode with Count Greater or Equal to 2^{48} May Terminate Early

Problem: In 64-bit Mode CMPSB, LODSB, or SCASB executed with a repeat prefix and count greater than or equal to 2^{48} may terminate early. Early termination may result in one of the following.

- The last iteration not being executed
- Signaling of a canonical limit fault (#GP) on the last iteration

Implication: While in 64-bit mode, with count greater or equal to 2^{48}, repeat string operations CMPSB, LODSB or SCASB may terminate without completing the last iteration. Intel has not observed this erratum with any commercially available software.

Workaround: Do not use repeated string operations with RCX greater than or equal to 2^{48}.

Status: For the steppings affected, see the Summary Tables of Changes.

AR59. Returning to Real Mode from SMM with EFLAGS.VM Set May Result in Unpredictable System Behavior

Problem: Returning back from SMM mode into real mode while EFLAGS.VM is set in SMRAM may result in unpredictable system behavior.

Implication: If SMM software changes the values of the EFLAGS.VM in SMRAM, it may result in unpredictable system behavior. Intel has not observed this behavior in commercially available software.

Workaround: SMM software should not change the value of EFLAGS.VM in SMRAM.

Status: For the steppings affected, see the Summary Tables of Changes.

AR60. IRET under Certain Conditions May Cause an Unexpected Alignment Check Exception

Problem: In IA-32e mode, it is possible to get an Alignment Check Exception (#AC) on the IRET instruction even though alignment checks were disabled at the start of the IRET. This can only occur if the IRET instruction is returning from CPL3 code to CPL3 code. IRETs from CPL0/1/2 are not affected. This erratum can occur if the EFLAGS value on the stack has the AC flag set, and the interrupt handler's stack is misaligned. In IA-32e mode, RSP is aligned to a 16-byte boundary before pushing the stack frame.

Implication: In IA-32e mode, under the conditions given above, an IRET can get a #AC even if alignment checks are disabled at the start of the IRET. This erratum can only be observed with a software generated stack frame.

Workaround: Software should not generate misaligned stack frames for use with IRET.

Status: For the steppings affected, see the Summary Tables of Changes.

AR61. Performance Monitoring Event FP_ASSIST May Not Be Accurate

Problem: Performance monitoring event FP_ASSIST (11H) may be inaccurate as assist events will be counted twice per actual assist in the following specific cases:

- FADD and FMUL instructions with a NaN(Not a Number) operand and a memory operand
- FDIV instruction with zero operand value in memory

In addition, an assist event may be counted when DAZ (Denormals-Are-Zeros) and FTZ (Flush-To-Zero) flags are turned on even though no actual assist occurs.

Implication: The counter value for the performance monitoring event FP_ASSIST (11H) may be larger than expected. The size of the error is dependent on the number of occurrences of the above conditions while the event is active.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR62. CPL-Qualified BTS May Report Incorrect Branch-From Instruction Address

Problem: CPL (Current Privilege Level)-qualified BTS (Branch Trace Store) may report incorrect branch-from instruction address under the following conditions:

- Either BTS_OFF_OS [9] or BTS_OFF_USR [10] is selected in IA32_DEBUGCTLC MSR (1D9H).
- Privilege-level transitions occur between CPL > 0 and CPL 0 or vice versa.

Implication: Due to this erratum, the From address reported by BTS may be incorrect for the described conditions.

Workaround: None identified

Status: For the steppings affected, see the Summary Tables of Changes.

AR63. PEBS Does Not Always Differentiate between CPL-Qualified Events

Problem: Performance monitoring counter configured to sample PEBS (Precise Event Based Sampling) events at a certain privilege level may count samples at the wrong privilege level.

Implication: Performance monitoring counter may be higher than expected for CPL-qualified events. Do not use performance monitoring counters for precise event sampling when the precise event is dependent on the CPL value.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR64. PMI May Be Delayed to Next PEBS Event

Problem: After a PEBS (Precise Event-Based Sampling) event, the PEBS index is compared with the PEBS threshold, and the index is incremented with every event. If PEBS index is equal to the PEBS threshold, a PMI (Performance Monitoring Interrupt) should be issued. Due to this erratum, the PMI may be delayed by one PEBS event.

Implication: Debug Store Interrupt Service Routines may observe delay of PMI occurrence by one PEBS event.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR65. PEBS Buffer Overflow Status Will Not Be Indicated Unless IA32_DEBUGCTL[12] is Set

Problem: IA32_PERF_GLOBAL_STATUS MSR (38EH) bit [62] when set, indicates that a PEBS (Precise Event-Based Sampling) overflow has occurred and a PMI (Performance Monitor Interrupt) has been sent. Due to this erratum, this bit is not set unless IA32_DEBUGCTL MSR (1D9H) bit [12] (which stops all performance monitor counters upon a PMI) is also set.

Implication: Due to this erratum, IA32_PERF_GLOBAL_STATUS [62] will not signal that a PMI was generated due to a PEBS Overflow unless IA32_DEBUGCTL [12] is set.

Workaround: It is possible for the software to set IA32_DEBUGCTL [12] to avoid this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AR66. The BS Flag in DR6 May Be Set for Non-Single-Step #DB Exception

Problem: DR6 BS (Single Step, bit 14) flag may be incorrectly set when the TF (Trap Flag, bit 8) of the EFLAGS Register is set, and a #DB (Debug Exception) occurs due to one of the following:

- DR7 GD (General Detect, bit 13) being bit set
- INT1 instruction;
- Code breakpoint

Implication: The BS flag may be incorrectly set for non-single-step #DB exception.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR67. An Asynchronous MCE during a Far Transfer May Corrupt ESP

Problem: If an asynchronous machine check occurs during an interrupt, call through gate, FAR RET or IRET and in the presence of certain internal conditions, ESP may be corrupted.

Implication: If the MCE (Machine Check Exception) handler is called without a stack switch, then a triple fault will occur due to the corrupted stack pointer, resulting in a processor shutdown. If the MCE is called with a stack switch, for example when the CPL (Current Privilege Level) was changed or when going through an interrupt task gate, then the corrupted ESP will be saved on the stack or in the TSS (Task State Segment), and will not be used.

Workaround: Use an interrupt task gate for the machine check handler.

Status: For the steppings affected, see the Summary Tables of Changes.

AR68. B0-B3 Bits in DR6 May Not Be Properly Cleared after Code Breakpoint

Problem: B0-B3 bits (breakpoint conditions detect flags, bits [3:0]) in DR6 may not be properly cleared when the following sequence happens:

1. POP instruction to SS (Stack Segment) selector.
2. Next instruction is FP (Floating Point) that gets FP assist followed by code breakpoint.

Implication: B0-B3 bits in DR6 may not be properly cleared.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR69. BTM/BTS Branch-From Instruction Address May Be Incorrect for Software Interrupts

Problem: When BTM (Branch Trace Message) or BTS (Branch Trace Store) is enabled, a software interrupt may result in the overwriting of BTM/BTS branch-from instruction address by the LBR (Last Branch Record) branch-from instruction address.

Implication: A BTM/BTS branch-from instruction address may get corrupted for software interrupts.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR70. REP Store Instructions in a Specific Situation may cause the Processor to Hang

Problem: During a series of REP (repeat) store instructions a store may try to dispatch to memory prior to the actual completion of the instruction. This behavior depends on the execution order of the instructions, the timing of a speculative jump and the timing of an uncacheable memory store. All types of REP store instructions are affected by this erratum.

Implication: When this erratum occurs, the processor may live lock and/or result in a system hang.

Workaround: It is possible for BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AR71. Performance Monitor SSE Retired Instructions May Return Incorrect Values

Problem: The SIMD_INST_RETIRED (Event: C7H) is used to track retired SSE instructions. Due to this erratum, the processor may also count other types of instructions resulting in values higher than the number of actual retired SSE instructions.

Implication: The event monitor instruction SIMD_INST_RETIRED may report count higher than expected.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR72. Performance Monitoring Events for L1 and L2 Miss May Not Be Accurate

Problem: Performance monitoring events 0CBh with an event mask value of 02h or 08h (MEM_LOAD_RETIRED.L1_LINE_MISS or MEM_LOAD_RETIRED.L2_LINE_MISS) may under count the cache miss events.

Implication: These performance monitoring events may show a count which is lower than expected; the amount by which the count is lower is dependent on other conditions occurring on the same load that missed the cache.

Workaround: None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR73. Store to WT Memory Data May Be Seen in Wrong Order by Two Subsequent Loads

Problem: When data of Store to WT memory is used by two subsequent loads of one thread and another thread performs cacheable write to the same address the first load may get the data from external memory or L2 written by another core, while the second load will get the data straight from the WT Store.

Implication: Software that uses WB to WT memory aliasing may violate proper store ordering.

Workaround: Do not use WB to WT aliasing.

Status: For the steppings affected, see the Summary Tables of Changes.

AR74. A MOV Instruction from CR8 Register with 16 Bit Operand Size Will Leave Bits 63:16 of the Destination Register Unmodified

Problem: Moves to/from control registers are supposed to ignore REW.W and the 66H (operand size) prefix. In systems supporting Intel Virtualization Technology, when the processor is operating in VMX non-root operation and "use TPR shadow" VM-execution control is set to 1, a MOV instruction from CR8 with a 16 bit operand size (REX.W =0 and 66H prefix) will only store 16 bits and leave bits 63:16 at the destination register unmodified, instead of storing zeros in them.

Implication: Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR75. Debug Register May Contain Incorrect Information on a MOVSS or POPSS Instruction Followed by SYSRET

Problem: In IA-32e mode, if a MOVSS or POPSS instruction with a debug breakpoint is followed by the SYSRET instruction; incorrect information may exist in the Debug Status Register (DR6).

Implication: When debugging or when developing debuggers, this behavior should be noted. This erratum does not occur under normal usage of the MOVSS or POPSS instructions (that is, following them with a MOV ESP instruction).

Workaround: Do not attempt to put a breakpoint on MOVSS and POPSS instructions that are followed by a SYSRET.

Status: For the steppings affected, see the Summary Tables of Changes.

AR76. Single Step Interrupts with Floating Point Exception Pending May Be Mishandled

Problem: In certain circumstances, when a floating point exception (#MF) is pending during single-step execution, processing of the single-step debug exception (#DB) may be mishandled.

Implication: When this erratum occurs, #DB will be incorrectly handled as follows:

- #DB is signaled before the pending higher priority #MF (Interrupt 16)
- #DB is generated twice on the same instruction

Workaround: None Identified

Status: For the steppings affected, see the Summary Tables of Changes.

AR77. Fault on ENTER Instruction May Result in Unexpected Values on Stack Frame

Problem: The ENTER instruction is used to create a procedure stack frame. Due to this erratum, if execution of the ENTER instruction results in a fault, the dynamic storage area of the resultant stack frame may contain unexpected values (i.e., residual stack data as a result of processing the fault).

Implication: Data in the created stack frame may be altered following a fault on the ENTER instruction. Please refer to "Procedure Calls For Block-Structured Languages" in the *Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 1: Basic Architecture*, for information on the usage of the ENTER instructions. This erratum is not expected to occur in ring 3. Faults are usually processed in ring 0 and stack switch occurs when transferring to ring 0. Intel has not observed this erratum on any commercially-available software.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR78. Unaligned Accesses to Paging Structures May Cause the Processor to Hang

Problem: When an unaligned access is performed on paging structure entries, accessing a portion of two different entries simultaneously, the processor may live lock.

Implication: When this erratum occurs, the processor may live lock causing a system hang.

Workaround: Do not perform unaligned accesses on paging structure entries.

Status: For the steppings affected, see the Summary Tables of Changes.

AR79. INVLPG Operation for Large (2M/4M) Pages May Be Incomplete under Certain Conditions

Problem: The INVLPG instruction may not completely invalidate Translation Look-aside Buffer (TLB) entries for large pages (2-M/4-M) when both of the following conditions exist:

- Address range of the page being invalidated spans several Memory Type Range Registers (MTRRs) with different memory types specified.
- INVLPG operation is preceded by a Page Assist Event (Page Fault (#PF) or an access that results in either A or D bits being set in a Page Table Entry (PTE)

Implication: Stale translations may remain valid in TLB after a PTE update resulting in unpredictable system behavior. Intel has not observed this erratum with any commercially available software

Workaround: Software should ensure that the memory type specified in the MTRRs is the same for the entire address range of the large page.

Status: For the steppings affected, see the Summary Tables of Changes.

AR80. Page Access Bit May Be Set Prior to Signaling a Code Segment Limit Fault

Problem: If code segment limit is set close to the end of a code page, then due to this erratum the memory page Access bit (A bit) may be set for the subsequent page prior to general protection fault on code segment limit.

Implication: When this erratum occurs, a non-accessed page present in memory following a page that contains the code segment limit may be tagged as accessed

Workaround: Non-present or non-executable page can be placed after the limit of the code segment to prevent this erratum.

Status: For the steppings affected, see the Summary Tables of Changes.

AR81. Update of Attribute Bits on Page Directories without Immediate TLB Shootdown May Cause Unexpected Processor Behavior

Problem: Updating a page directory entry (or page map level 4 table entry or page directory pointer table entry in IA-32e mode) by changing R/W, U/S or P bits without immediate TLB shootdown (as described by the 4 step procedure in "Propagation of Page Table and Page Directory Entry Changes to Multiple Processors" in *Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 3A: System Programming Guide*), in conjunction with a complex sequence of internal processor micro-architectural events, may lead to unexpected processor behavior.

Implication: This erratum may lead to livelock, shutdown or other unexpected processor behavior. Intel has not observed this erratum with any commercially available system.

Workaround: It is possible for BIOS to contain a workaround for this erratum.

Status: Plan Fix. For the steppings affected, see the Summary Tables of Changes.

AR82. Invalid Instructions May Lead to Unexpected Behavior

Problem: Invalid instructions due to undefined opcodes or instructions exceeding the maximum instruction length (due to redundant prefixes placed before the instruction) may lead, under complex circumstances, to unexpected behavior.

Implication: The processor may behave unexpectedly due to invalid instructions. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR83. EFLAGS, CR0, CR4 and the EXF4 Signal May Be Incorrect after Shutdown

Problem: When the processor is going into shutdown due to an RSM inconsistency failure, EFLAGS, CR0 and CR4 may be incorrect. In addition the EXF4 signal may still be asserted. This may be observed if the processor is taken out of shutdown by NMI#.

Implication: A processor that has been taken out of shutdown may have an incorrect EFLAGS, CR0 and CR4. In addition the EXF4 signal may still be asserted.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR84. Performance Monitoring Counter MACRO_INSTS.DECODED May Not Count Some Decoded Instructions

Problem: MACRO_INSTS.DECODED performance monitoring counter (Event 0AAH, Umask 01H) counts the number of macro instructions decoded, but not necessarily retired. The event is undercounted when the decoded instructions are a complete loop iteration that is decoded in one cycle and the loop is streamed by the LSD (Loop Stream Detector), as described in the *IA-32 Intel® Architecture Optimization Reference Manual*.

Implication: The count value returned by the performance monitoring counter MACRO_INST.DECODED may be lower than expected. The degree of undercounting is dependent on the occurrence of loop iterations that are decoded in one cycle and whether the loop is streamed by the LSD while the counter is active.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR85. The Stack May Be Incorrect as a Result of VIP/VIF Check on SYSEXIT and SYSRET

Problem: The stack size may be incorrect under the following scenario:

1. The stack size was changed due to a SYSEXIT or SYSRET

2. PVI (Protected Mode Virtual Interrupts) mode was enabled (CR4.PVI == 1)

3. Both the VIF (Virtual Interrupt Flag) and VIP (Virtual Interrupt Pending) flags of the EFLAGS register are set

Implication: If this erratum occurs the stack size may be incorrect, consequently this may result in unpredictable system behavior. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: Plan Fix. For the steppings affected, see the Summary Tables of Changes.

AR86. Performance Monitoring Event SIMD_UOP_TYPE_EXEC.MUL is Counted Incorrectly for PMULUDQ Instruction

Problem: Performance Monitoring Event SIMD_UOP_TYPE_EXEC.MUL (Event select 0B3H, Umask 01H) counts the number of SIMD packed multiply micro-ops executed. The count for PMULUDQ micro-ops might be lower than expected. No other instruction is affected.

Implication: The count value returned by the performance monitoring event SIMD_UOP_TYPE_EXEC.MUL may be lower than expected. The degree of undercount depends on actual occurrences of PMULUDQ instructions, while the counter is active.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR87. Storage of PEBS Record Delayed Following Execution of MOV SS or STI

Problem: When a performance monitoring counter is configured for PEBS (Precise Event Based Sampling), overflow of the counter results in storage of a PEBS record in the PEBS buffer. The information in the PEBS record represents the state of the next instruction to be executed following the counter overflow. Due to this erratum, if the counter overflow occurs after execution of either MOV SS or STI, storage of the PEBS record is delayed by one instruction.

Implication: When this erratum occurs, software may observe storage of the PEBS record being delayed by one instruction following execution of MOV SS or STI. The state information in the PEBS record will also reflect the one instruction delay.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR88. Updating Code Page Directory Attributes without TLB Invalidation May Result in Improper Handling of Code #PF

Problem: Code #PF (Page Fault exception) is normally handled in lower priority order relative to both code #DB (Debug Exception) and code Segment Limit Violation #GP (General Protection Fault). Due to this erratum, code #PF may be handled incorrectly, if all of the following conditions are met:

- A PDE (Page Directory Entry) is modified without invalidating the corresponding TLB (Translation Look-aside Buffer) entry
- Code execution transitions to a different code page such that both
  - The target linear address corresponds to the modified PDE
  - The PTE (Page Table Entry) for the target linear address has an A (Accessed) bit that is clear
- One of the following simultaneous exception conditions is present following the code transition
  - Code #DB and code #PF
  - Code Segment Limit Violation #GP and code #PF

Implication: Software may observe either incorrect processing of code #PF before code Segment Limit Violation #GP or processing of code #PF in lieu of code #DB.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR89. Performance Monitoring Event CPU_CLK_UNHALTED.REF May Not Count Clock Cycles According to the Processors Operating Frequency

Problem: Performance Counter MSR_PERF_FIXED_CTR2 (MSR 30BH) that counts CPU_CLK_UNHALTED.REF clocks should count these clock cycles at a constant rate that is determined by the maximum resolved boot frequency, as programmed by BIOS. Due to this erratum, the rate is instead set by the maximum core-clock to bus-clock ratio of the processor, as indicated by hardware.

Implication: No functional impact as a result of this erratum. If the maximum resolved boot frequency as programmed by BIOS is different from the frequency implied by the maximum core-clock to bus-clock ratio of the processor as indicated by hardware, then the following effects may be observed:

Workaround: Performance Monitoring Event CPU_CLK_UNHALTED.REF will count at a rate different than the TSC (Time Stamp Counter)

Workaround: When running a system with several processors that have different maximum core-clock to bus-clock ratios, CPU_CLK_UNHALTED.REF monitoring events at each processor will be counted at different rates and therefore will not be comparable.

Workaround: Calculate the ratio of the rates at which the TSC and the CPU_CLK_UNHALTED.REF performance monitoring event count (this can be done by measuring simultaneously their counted value while executing code) and adjust the CPU_CLK_UNHALTED.REF event count to the maximum resolved boot frequency using this ratio.

Status: For the steppings affected, see the Summary Tables of Changes.

AR90. Store Ordering May Be Incorrect between WC and WP Memory Types

Problem: According to *Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 3A: System Programming Guide*, WP (Write Protected) stores should drain the WC (Write Combining) buffers in the same way as UC (Uncacheable) memory type stores do. Due to this erratum, WP stores may not drain the WC buffers.

Implication: Memory ordering may be violated between WC and WP stores.

Workaround: None Identified

Status: For the steppings affected, see the Summary Tables of Changes.

AR91. Performance Monitoring Event BR_INST_RETIRED May Count CPUID Instructions as Branches

Problem: Performance monitoring event BR_INST_RETIRED (C4H) counts retired branch instructions. Due to this erratum, two of its sub-events mistakenly count for CPUID instructions as well. Those sub events are: BR_INST_RETIRED.PRED_NOT_TAKEN (Umask 01H) and BR_INST_RETIRED.ANY (Umask 00H).

Implication: The count value returned by the performance monitoring event BR_INST_RETIRED.PRED_NOT_TAKEN or BR_INST_RETIRED.ANY may be higher than expected. The extent of over counting depends on the occurrence of CPUID instructions, while the counter is active.

Workaround: None Identified.

Status: For the steppings affected, see the Summary Tables of Changes.

AR92. Performance Monitoring Event MISALIGN_MEM_REF May Over Count

Problem: Performance monitoring event MISALIGN_MEM_REF (05H) is used to count the number of memory accesses that cross an 8-byte boundary and are blocked until retirement. Due to this erratum, the performance monitoring event MISALIGN_MEM_REF also counts other memory accesses.

Implication: The performance monitoring event MISALIGN_MEM_REF may over count. The extent of the over counting depends on the number of memory accesses retiring while the counter is active.

Workaround: None Identified

Status: For the steppings affected, see the Summary Tables of Changes.

AR93. A REP STOS/MOVS to a MONITOR/MWAIT Address Range May Prevent Triggering of the Monitoring Hardware

Problem: The MONITOR instruction is used to arm the address monitoring hardware for the subsequent MWAIT instruction. The hardware is triggered on subsequent memory store operations to the monitored address range. Due to this erratum, REP STOS/MOVS fast string operations to the monitored address range may prevent the actual triggering store to be propagated to the monitoring hardware.

Implication: A logical processor executing an MWAIT instruction may not immediately continue program execution if a REP STOS/MOVS targets the monitored address range.

Workaround: Software can avoid this erratum by not using REP STOS/MOVS store operations within the monitored address range.

Status: For the steppings affected, see the Summary Tables of Changes.

AR94. False Level One Data Cache Parity Machine-Check Exceptions May Be Signaled

Problem: Executing an instruction stream containing invalid instructions/data may generate a false Level One Data Cache parity machine-check exception.

Implication: The false Level One Data Cache parity machine-check exception is reported as an uncorrected machine-check error. An uncorrected machine-check error is treated as a fatal exception by the operating system and may cause a shutdown and/or reboot.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the Summary Tables of Changes

AR95. A Memory Access May Get a Wrong Memory Type Following a #GP due to WRMSR to an MTRR Mask

Problem: The TLB (Translation Lookaside Buffer) may indicate a wrong memory type on a memory access to a large page (2M/4M Byte) following the recovery from a #GP (General Protection Fault) due to a WRMSR to one of the IA32_MTRR_PHYSMASKn MSRs with reserved bits set.

Implication: When this erratum occurs, a memory access may get an incorrect memory type leading to unexpected system operation. As an example, an access to a memory mapped I/O device may be incorrectly marked as cacheable, become cached, and never make it to the I/O device. Intel has not observed this erratum with any commercially available software.

Workaround: Software should not attempt to set reserved bits of IA32_MTRR_PHYSMASK*n* MSRs.

Status: For the steppings affected, see the Summary Tables of Changes.

AR96. PMI While LBR Freeze Enabled May Result in Old/Out-of-Date LBR Information

Problem: When Precise Event-Based Sampling (PEBS) is configured with Performance Monitoring Interrupt (PMI) on PEBS buffer overflow enabled and Last Branch Record (LBR) Freeze on PMI enabled by setting FREEZE_LBRS_ON_PMI flag (bit 11) to 1 in IA32_DEBUGCTL (MSR 1D9H), the LBR stack is frozen upon the occurrence of a hardware PMI request. Due to this erratum, the LBR freeze may occur too soon (i.e., before the hardware PMI request).

Implication: Following a PMI occurrence, the PMI handler may observe old/out-of-date LBR information that does not describe the last few branches before the PEBS sample that triggered the PMI.

Workaround: None identified

Status: For the steppings affected, see the Summary Tables of Changes.

AR97. BIST Failure after Reset

Problem: The processor may show an erroneous BIST (built-in self test) result in bit [17] of EAX register when coming out of reset.

Implication: When this erratum occurs, an erroneous BIST failure will be reported in EAX bit [17]. This failure can be ignored since it is not accurate.

Workaround: It is possible for BIOS to workaround this erratum by masking off bit [17] of the EAX register after coming out of reset.

Status: For the steppings affected, see the Summary Tables of Changes.

AR99. Instruction Fetch May Cause a Livelock during Snoops of the L1 Data Cache

Problem: A livelock may be observed in rare conditions when instruction fetch causes multiple level one data cache snoops.

Implication: Due to this erratum, a livelock may occur. Intel has not observed this erratum with any commercially available software.

Workaround: It is possible for BIOS to contain a workaround for this erratum]

Status: For the steppings affected, see the Summary Tables of Changes.

AR100. Use of Memory Aliasing with Inconsistent Memory Type may Cause a System Hang or a Machine Check Exception

Problem: Software that implements memory aliasing by having more than one linear addresses mapped to the same physical page with different cache types may cause the system to hang or to report a machine check exception (MCE). This would occur if one of the addresses is non-cacheable and used in a code segment and the other is a cacheable address. If the cacheable address finds its way into the instruction cache, and the non-cacheable address is fetched in the IFU, the processor may invalidate the non-cacheable address from the fetch unit. Any micro-architectural event that causes instruction restart will be expecting this instruction to still be in the fetch unit and lack of it will cause a system hang or an MCE.

Implication: This erratum has not been observed with commercially available software.

Workaround: Although it is possible to have a single physical page mapped by two different linear addresses with different memory types, Intel has strongly discouraged this practice as it may lead to undefined results. Software that needs to implement memory aliasing should manage the memory type consistency.

Status: For the steppings affected, see the Summary Tables of Changes.

AR101. A WB Store Following a REP STOS/MOVS or FXSAVE May Lead to Memory-Ordering Violations

Problem: Under certain conditions, as described in the *Intel® 64 and IA-32 Architecture Software Developer's Manual*, section "Out-of-Order Stores For String Operations in Pentium 4, Intel Xeon, and P6 Family Processors", the processor may perform REP MOVS or REP STOS as write combining stores (referred to as "fast strings") for optimal performance. FXSAVE may also be internally implemented using write combining stores. Due to this erratum, stores of a WB (write back) memory type to a cache line previously written by a preceding fast string/FXSAVE instruction may be observed before string/FXSAVE stores.

Implication: A write-back store may be observed before a previous string or FXSAVE related store. Intel has not observed this erratum with any commercially available software.

Workaround: Software desiring strict ordering of string/FXSAVE operations relative to subsequent write-back stores should add an MFENCE or SFENCE instruction between the string/FXSAVE operation and following store-order sensitive code such as that used for synchronization.

Status: For the steppings affected, see the Summary Tables of Changes.

AR102. Using Memory Type Aliasing with Cacheable and WC Memory Types May Lead to Memory Ordering Violations

Problem: Memory type aliasing occurs when a single physical page is mapped to two or more different linear addresses, each with different memory types. Memory type aliasing with a cacheable memory type and WC (write combining) may cause the processor to perform incorrect operations leading to memory ordering violations for WC operations.

Implication: Software that uses aliasing between cacheable and WC memory types may observe memory ordering errors within WC memory operations. Intel has not observed this erratum with any commercially available software.

Workaround: None identified. Intel does not support the use of cacheable and WC memory type aliasing, and WC operations are defined as weakly ordered.

Status: For the steppings affected, see the Summary Tables of Changes.

AR103 RSM Instruction Execution under Certain Conditions May Cause Processor Hang or Unexpected Instruction Execution Results

Problem: RSM instruction execution, under certain conditions triggered by a complex sequence of internal processor micro-architectural events, may lead to processor hang, or unexpected instruction execution results.

Implication: In the above sequence, the processor may live lock or hang, or RSM instruction may restart the interrupted processor context through a nondeterministic EIP offset in the code segment, resulting in unexpected instruction execution, unexpected exceptions or system hang. Intel has not observed this erratum with any commercially available software.

Workaround: It is possible for the BIOS to contain a workaround for this erratum. Please contact your Intel sales representative for availability.

Status: For the steppings affected, see the Summary Tables of Changes.

AR104 NMIs May Not Be Blocked by a VM-Entry Failure

Problem: The *Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 3B: System Programming Guide*, Part 2 specifies that, following a VM-entry failure during or after loading guest state, "the state of blocking by NMI is what it was before VM entry." If non-maskable interrupts (NMIs) are blocked and the "virtual NMIs" VM-execution control set to 1, this erratum may result in NMIs not being blocked after a VM-entry failure during or after loading guest state.

Implication: VM-entry failures that cause NMIs to become unblocked may cause the processor to deliver an NMI to software that is not prepared for it.

Workaround: VMM software should configure the virtual-machine control structure (VMCS) so that VM-entry failures do not occur.

Status: For the steppings affected, see the Summary Tables of Changes.

AR105 Benign Exception after a Double Fault May Not Cause a Triple Fault Shutdown

Problem: According to the *Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 3A: System Programming Guide*, if another exception occurs while attempting to call the double-fault handler, the processor enters shutdown mode. However due to this erratum, only Contributory Exceptions and Page Faults will cause a triple fault shutdown, whereas a benign exception may not.

Implication: If a benign exception occurs while attempting to call the double-fault handler, the processor may hang or may handle the benign exception. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the Summary Tables of Changes.

§

Specification Changes

There are no specification changes for this specification update revision.

§

Specification Clarifications

AR1 Clarification of Translation Lookaside Buffers (TLBS) Invalidation

Section 10.9 INVALIDATING THE TRANSLATION LOOKASIDE BUFFERS (TLBS) of the *Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 3A: System Programming Guide* will be modified to include the presence of page table structure caches, such as the page directory cache, which Intel processors implement. This information is needed to aid operating systems in managing page table structure invalidations properly.

Intel will update the *Intel® 64 and IA-32 Architecture Software Developer's Manual, Volume 3A: System Programming Guide* in the coming months. Until that time, an application note, *TLBs, Paging-Structure Caches, and Their Invalidation* (http://www.intel.com/products/processor/manuals/index.htm) is available which provides more information on the paging structure caches and TLB invalidation.

In rare instances, improper TLB invalidation may result in unpredictable system behavior, such as system hangs or incorrect data. Developers of operating systems should take this documentation into account when designing TLB invalidation algorithms. For the processors affected, Intel has provided a recommended update to system and BIOS vendors to incorporate into their BIOS to resolve this issue.

§

Documentation Changes

There are no documentation changes for this specification update revision.

§