Intel® 64 and IA-32 Architectures Software Developer's Manual

Documentation Changes

December 2011

Document Number: 252046-034

Contents

Revision History

Revision	Description	Date
-001	• Initial release	November 2002
-002	• Added 1-10 Documentation Changes. • Removed old Documentation Changes items that already have been incorporated in the published Software Developer's manual	December 2002
-003	• Added 9 -17 Documentation Changes. • Removed Documentation Change #6 - References to bits Gen and Len Deleted. • Removed Documentation Change #4 - VIF Information Added to CLI Discussion	February 2003
-004	• Removed Documentation changes 1-17. • Added Documentation changes 1-24.	June 2003
-005	• Removed Documentation Changes 1-24. • Added Documentation Changes 1-15.	September 2003
-006	• Added Documentation Changes 16- 34.	November 2003
-007	• Updated Documentation changes 14, 16, 17, and 28. • Added Documentation Changes 35-45.	January 2004
-008	• Removed Documentation Changes 1-45. • Added Documentation Changes 1-5.	March 2004
-009	• Added Documentation Changes 7-27.	May 2004
-010	• Removed Documentation Changes 1-27. • Added Documentation Changes 1.	August 2004
-011	• Added Documentation Changes 2-28.	November 2004
-012	• Removed Documentation Changes 1-28. • Added Documentation Changes 1-16.	March 2005
-013	• Updated title. • There are no Documentation Changes for this revision of the document.	July 2005
-014	• Added Documentation Changes 1-21.	September 2005
-015	• Removed Documentation Changes 1-21. • Added Documentation Changes 1-20.	March 9, 2006
-016	• Added Documentation changes 21-23.	March 27, 2006
-017	• Removed Documentation Changes 1-23. • Added Documentation Changes 1-36.	September 2006
-018	• Added Documentation Changes 37-42.	October 2006
-019	• Removed Documentation Changes 1-42. • Added Documentation Changes 1-19.	March 2007
-020	• Added Documentation Changes 20-27.	May 2007
-021	• Removed Documentation Changes 1-27. • Added Documentation Changes 1-6	November 2007
-022	• Removed Documentation Changes 1-6 • Added Documentation Changes 1-6	August 2008
-023	• Removed Documentation Changes 1-6 • Added Documentation Changes 1-21	March 2009

Revision	Description	Date
-024	• Removed Documentation Changes 1-21 • Added Documentation Changes 1-16	June 2009
-025	• Removed Documentation Changes 1-16 • Added Documentation Changes 1-18	September 2009
-026	• Removed Documentation Changes 1-18 • Added Documentation Changes 1-15	December 2009
-027	• Removed Documentation Changes 1-15 • Added Documentation Changes 1-24	March 2010
-028	• Removed Documentation Changes 1-24 • Added Documentation Changes 1-29	June 2010
-029	• Removed Documentation Changes 1-29 • Added Documentation Changes 1-29	September 2010
-030	• Removed Documentation Changes 1-29 • Added Documentation Changes 1-29	January 2011
-031	• Removed Documentation Changes 1-29 • Added Documentation Changes 1-29	April 2011
-032	• Removed Documentation Changes 1-29 • Added Documentation Changes 1-14	May 2011
-033	• Removed Documentation Changes 1-14 • Added Documentation Changes 1-38	October 2011
-034	• Removed Documentation Changes 1-38 • Added Documentation Changes 1-16	December 2011

§

Preface

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

Affected Documents

Document Title	Document Number/Location
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture	253665
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M	253666
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z	253667
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide, Part 1	253668
Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide, Part 2	253669

Nomenclature

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

Summary Tables of Changes

The following table indicates documentation changes which apply to the Intel® 64 and IA-32 architectures. This table uses the following notations:

Codes Used in Summary Tables

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

Documentation Changes

No.	DOCUMENTATION CHANGES
1	Updates to Chapter 7, Volume 1
2	Updates to Chapter 12, Volume 1
3	Updates to Chapter 13, Volume 1
4	Updates to Chapter 1, Volume 2A
5	Updates to Chapter 2, Volume 2A
6	Updates to Chapter 3, Volume 2A
7	Updates to Chapter 4, Volume 2B
8	Updates to Chapter 6, Volume 3A
9	Updates to Chapter 8, Volume 3A
10	Updates to Chapter 9, Volume 3A
11	Updates to Chapter 17, Volume 3B
12	Updates to Chapter 18, Volume 3B
13	Updates to Chapter 19, Volume 3B
14	Updates to Chapter 25, Volume 3C
15	Updates to Chapter 33, Volume 3C
16	Updates to Chapter 34, Volume 3C

Documentation Changes

1. Updates to Chapter 7, Volume 1

Change bars show changes to Chapter 7 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1:* Basic Architecture.

...

7.3 SUMMARY OF GP INSTRUCTIONS

General purpose instructions are divided into the following subgroups:

- Data transfer
- Binary arithmetic
- Decimal arithmetic
- Logical
- Shift and rotate
- Bit and byte
- Control transfer
- String
- I/O
- Enter and Leave
- Flag control
- Segment register
- Miscellaneous

Each sub-group of general-purpose instructions is discussed in the context of non-64-bit mode operation first. Changes in 64-bit mode beyond those affected by the use of the REX prefixes are discussed in separate sub-sections within each subgroup. For a simple list of general-purpose instructions by subgroup, see Chapter 5.

...

7.3.9 String Operations

The GP instructions includes a set of **string instructions** that are designed to access large data structures; these are introduced in Section 7.3.9.1. Section 7.3.9.2 describes how REP prefixes can be used with these instructions to perform more complex **repeated string operations**. Certain processors optimize repeated string operations with **fast-string operation**, as described in Section 7.3.9.3. Section 7.3.9.4 explains how string operations can be used in 64-bit mode.

7.3.9.1 String Instructions

The MOVS (Move String), CMPS (Compare string), SCAS (Scan string), LODS (Load string), and STOS (Store string) instructions permit large data structures, such as alpha-numeric character strings, to be moved and examined in memory. These instructions operate on individual elements in a string, which can be a byte, word, or doubleword. The string elements to be operated on are identified with the ESI (source string element) and EDI (destination string element) registers. Both of these registers contain absolute addresses (offsets into a segment) that point to a string element.

...

7.3.9.2 Repeated String Operations

Each of the string instructions described in Section 7.3.9.1 each perform one iteration of a string operation. To operate strings longer than a doubleword, the string instructions can be combined with a repeat prefix (REP) to create a repeating instruction or be placed in a loop.

When used in string instructions, the ESI and EDI registers are automatically incremented or decremented after each iteration of an instruction to point to the next element (byte, word, or doubleword) in the string. String operations can thus begin at higher addresses and work toward lower ones, or they can begin at lower addresses and work toward higher ones. The DF flag in the EFLAGS register controls whether the registers are incremented (DF = 0) or decremented (DF = 1). The STD and CLD instructions set and clear this flag, respectively.

The following repeat prefixes can be used in conjunction with a count in the ECX register to cause a string instruction to repeat:

- **REP** — Repeat while the ECX register not zero.
- **REPE/REPZ** — Repeat while the ECX register not zero and the ZF flag is set.
- **REPNE/REPNZ** — Repeat while the ECX register not zero and the ZF flag is clear.

When a string instruction has a repeat prefix, the operation executes until one of the termination conditions specified by the prefix is satisfied. The REPE/REPZ and REPNE/REPNZ prefixes are used only with the CMPS and SCAS instructions. Also, note that a REP STOS instruction is the fastest way to initialize a large block of memory.

7.3.9.3 Fast-String Operation

To improve performance, more recent processors support modifications to the processor's operation during the string store operations initiated with the MOVS, MOVSB, STOS, and STOSB instructions. This optimized operation, called **fast-string operation**, is used when the execution of one of those instructions meets certain initial conditions (see below). Instructions using fast-string operation effectively operate on the string in groups that may include multiple elements of the native data size (byte, word, doubleword, or quadword). With fast-string operation, the processor recognizes interrupts and data breakpoints only on boundaries between these groups. Fast-string operation is used only if the source and destination addresses both use either the WB or WC memory types.

The initial conditions for fast-string operation are implementation-specific and may vary with the native string size. Examples of parameters that may impact the use of fast-string operation include the following:

- the alignment indicated in the EDI and ESI alignment registers;

- the address order of the string operation;
- the value of the initial operation counter (ECX); and
- the difference between the source and destination addresses.

NOTE

Initial conditions for fast-string operation in future Intel 64 or IA-32 processor families may differ from above. The *Intel® 64 and IA-32 Architectures Optimization Reference Manual* may contain model-specific information.

Software can disable fast-string operation by clearing the fast-string-enable bit (bit 0) of IA32_MISC_ENABLE MSR. However, Intel recommends that system software always enable fast-string operation.

When fast-string operation is enabled (because IA32_MISC_ENABLE[0] = 1), some processors may further enhance the operation of the REP MOVSB and REP STOSB instructions. A processors supports these enhancements if CPUID.(EAX=07H, ECX=0H):EBX[bit 9] is 1. The *Intel® 64 and IA-32 Architectures Optimization Reference Manual* may include model-specific recommendations for use of these enhancements.

The stores produced by fast-string operation may appear to execute out of order. Software dependent upon sequential store ordering should not use string operations for the entire data structure to be stored. Data and semaphores should be separated. Order-dependent code should write to a discrete semaphore variable after any string operations to allow correctly ordered data to be seen by all processors. Atomicity of load and store operations is guaranteed only for native data elements of the string with native data size, and only if they are included in a single cache line. See Section 8.2.4, "Fast-String Operation and Out-of-Order Stores" of *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A.*

7.3.9.4 String Operations in 64-Bit Mode

The behavior of MOVS (Move String), CMPS (Compare string), SCAS (Scan string), LODS (Load string), and STOS (Store string) instructions in 64-bit mode is similar to their behavior in non-64-bit modes, with the following differences:

- The source operand is specified by RSI or DS:ESI, depending on the address size attribute of the operation.
- The destination operand is specified by RDI or DS:EDI, depending on the address size attribute of the operation.
- Operation on 64-bit data is supported by using the REX.W prefix.

When using REP prefixes for string operations in 64-bit mode, the repeat count is specified by RCX or ECX (depending on the address size attribute of the operation). The default address size is 64 bits.

...

2. Updates to Chapter 12, Volume 1

Change bars show changes to Chapter 12 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1:* Basic Architecture.

...

12.10.4 Packed Blending Instructions

SSE4.1 adds 6 instructions used for blending (BLENDPS, BLENDPD, BLENDVPS, BLENDVPD, PBLENDVB, PBLENDW).

Blending conditionally copies a data element in a source operand to the same element in the destination. SSE4.1 instructions improve blending operations for most field sizes. A single new SSE4.1 instruction can generally replace a sequence of 2 to 4 operations using previous architectures.

The variable blend instructions (BLENDVPS, PBLENDVPD, PBLENDW) introduce the use of control bits stored in an implicit XMM register (XMM0). The most significant bit in each field (the sign bit, for 2's complement integer or floating-point) is used as a selector. See Table 12-3.

...

3. Updates to Chapter 13, Volume 1

Change bars show changes to Chapter 13 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1:* Basic Architecture.

...

13.2.3 Arithmetic Primitives for 128-bit Vector and Scalar processing

Intel AVX provides a full complement of 128-bit numeric processing instructions that employ VEX-prefix encoding. These VEX-encoded instructions generally provide the same functionality over instructions operating on XMM register that are encoded using SIMD prefixes. The 128-bit numeric processing instructions in AVX cover floating-point and integer data processing; across 128-bit vector and scalar processing. Table 13-5 lists the state of promotion of legacy SIMD arithmetic ISA to VEX-128 encoding. Legacy SIMD floating-point arithmetic ISA promoted to VEX-256 encoding also support VEX-128 encoding (see Table 13-2).

...

13.2.4 Non-Arithmetic Primitives for 128-bit Vector and Scalar Processing

Intel AVX provides a full complement of data processing instructions that employ VEX-prefix encoding. These VEX-encoded instructions generally provide the same functionality over instructions operating on XMM register that are encoded using SIMD prefixes.

...

4. Updates to Chapter 1, Volume 2A

Change bars show changes to Chapter 1 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A:* Instruction Set Reference, A-L.

...

1.2 OVERVIEW OF VOLUME 2A, 2B AND 2C: INSTRUCTION SET REFERENCE

A description of *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volumes 2A, 2B & 2C*, content follows:

Chapter 1 — About This Manual. Gives an overview of all seven volumes of the *Intel® 64 and IA-32 Architectures Software Developer's Manual*. It also describes the notational conventions in these manuals and lists related Intel® manuals and documentation of interest to programmers and hardware designers.

Chapter 2 — Instruction Format. Describes the machine-level instruction format used for all IA-32 instructions and gives the allowable encodings of prefixes, the operand-identifier byte (ModR/M byte), the addressing-mode specifier byte (SIB byte), and the displacement and immediate bytes.

Chapter 3 — Instruction Set Reference, A-L. Describes Intel 64 and IA-32 instructions in detail, including an algorithmic description of operations, the effect on flags, the effect of operand- and address-size attributes, and the exceptions that may be generated. The instructions are arranged in alphabetical order. General-purpose, x87 FPU, Intel MMX™ technology, SSE/SSE2/SSE3/SSSE3/SSE4 extensions, and system instructions are included.

Chapter 4 — Instruction Set Reference, M-Z. Continues the description of Intel 64 and IA-32 instructions started in Chapter 3. It provides the balance of the alphabetized list of instructions and starts *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B*.

Chapter 5— Safer Mode Extensions Reference. Describes the safer mode extensions (SMX). SMX is intended for a system executive to support launching a measured environment in a platform where the identity of the software controlling the platform hardware can be measured for the purpose of making trust decisions. This chapter starts *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2C*.

Appendix A — Opcode Map. Gives an opcode map for the IA-32 instruction set.

Appendix B — Instruction Formats and Encodings. Gives the binary encoding of each form of each IA-32 instruction.

Appendix C — Intel® C/C++ Compiler Intrinsics and Functional Equivalents. Lists the Intel® C/C++ compiler intrinsics and their assembly code equivalents for each of the IA-32 MMX and SSE/SSE2/SSE3 instructions.

...

5. Updates to Chapter 2, Volume 2A

Change bars show changes to Chapter 2 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A:* Instruction Set Reference, A-L.

...

2.5 CONTROL REGISTERS

...

OSXMMEXCPT

Operating System Support for Unmasked SIMD Floating-Point Exceptions (bit 10 of CR4) — When set, indicates that the operating system supports the handling of unmasked SIMD floating-point exceptions through an exception handler that is invoked when a SIMD floating-point exception (#XF) is generated. SIMD floating-point exceptions are only generated by SSE/SSE2/SSE3/SSE4.1 SIMD floating-point instructions.

The operating system or executive must explicitly set this flag. If this flag is not set, the processor will generate an invalid opcode exception (#UD) whenever it detects an unmasked SIMD floating-point exception.

VMXE

VMX-Enable Bit (bit 13 of CR4) — Enables VMX operation when set. See Chapter 23, "Introduction to Virtual-Machine Extensions."

SMXE

SMX-Enable Bit (bit 14 of CR4) — Enables SMX operation when set. See Chapter 33, "VMX Instruction Reference" of *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3C*.

FSGSBASE

FSGSBASE-Enable Bit (bit 16 of CR4) — Enables the instructions RDFSBASE, RDGSBASE, WRFSBASE, and WRGSBASE.

PCIDE

PCID-Enable Bit (bit 17 of CR4) — Enables process-context identifiers (PCIDs) when set. See Section 4.10.1, "Process-Context Identifiers (PCIDs)". Can be set only in IA-32e mode (if IA32_EFER.LMA = 1).

...

6. Updates to Chapter 3, Volume 2A

Change bars show changes to Chapter 3 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A:* Instruction Set Reference, A-L.

...

CLFLUSH—Flush Cache Line

...

Real-Address Mode Exceptions

#GP	If any part of the operand lies outside the effective address space from 0 to FFFFH.
#UD	If CPUID.01H:EDX.CLFSH[bit 19] = 0.
	If the LOCK prefix is used.

...

CPUID—CPU Identification

Opcode	**Instruction**	**Op/ En**	**64-Bit Mode**	**Compat/ Leg Mode**	**Description**
0F A2	CPUID	NP	Valid	Valid	Returns processor identification and feature information to the EAX, EBX, ECX, and EDX registers, as determined by input entered in EAX (in some cases, ECX as well).

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
NP	NA	NA	NA	NA

Description

The ID flag (bit 21) in the EFLAGS register indicates support for the CPUID instruction. If a software procedure can set and clear this flag, the processor executing the procedure supports the CPUID instruction. This instruction operates the same in non-64-bit modes and 64-bit mode.

CPUID returns processor identification and feature information in the EAX, EBX, ECX, and EDX registers.[1] The instruction's output is dependent on the contents of the EAX register upon execution (in some cases, ECX as well). For example, the following pseudocode loads EAX with 00H and causes CPUID to return a Maximum Return Value and the Vendor Identification String in the appropriate registers:

1. On Intel 64 processors, CPUID clears the high 32 bits of the RAX/RBX/RCX/RDX registers in all modes.

```
MOV EAX, 00H
CPUID
```

Table 3-17 shows information returned, depending on the initial value loaded into the EAX register. Table 3-18 shows the maximum CPUID input value recognized for each family of IA-32 processors on which CPUID is implemented.

Two types of information are returned: basic and extended function information. If a value entered for CPUID.EAX is higher than the maximum input value for basic or extended function for that processor then the data for the highest basic information leaf is returned. For example, using the Intel Core i7 processor, the following is true:

```
CPUID.EAX = 05H (* Returns MONITOR/MWAIT leaf. *)
CPUID.EAX = 0AH (* Returns Architectural Performance Monitoring leaf. *)
CPUID.EAX = 0BH (* Returns Extended Topology Enumeration leaf. *)
CPUID.EAX = 0CH (* INVALID: Returns the same information as CPUID.EAX = 0BH. *)
CPUID.EAX = 80000008H (* Returns linear/physical address size data. *)
CPUID.EAX = 8000000AH (* INVALID: Returns same information as CPUID.EAX = 0BH. *)
```

If a value entered for CPUID.EAX is less than or equal to the maximum input value and the leaf is not supported on that processor then 0 is returned in all the registers. For example, using the Intel Core i7 processor, the following is true:

```
CPUID.EAX = 07H (*Returns EAX=EBX=ECX=EDX=0. *)
```

When CPUID returns the highest basic leaf information as a result of an invalid input EAX value, any dependence on input ECX value in the basic leaf is honored.

CPUID can be executed at any privilege level to serialize instruction execution. Serializing instruction execution guarantees that any modifications to flags, registers, and memory for previous instructions are completed before the next instruction is fetched and executed.

See also:

"Serializing Instructions" in Chapter 8, "Multiple-Processor Management," in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*

"Caching Translation Information" in Chapter 4, "Paging," in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A*.

Table 3-17 Information Returned by CPUID Instruction

Initial EAX Value	Information Provided about the Processor	
	Basic CPUID Information	
0H	EAX	Maximum Input Value for Basic CPUID Information (see Table 3-18)
	EBX	"Genu"
	ECX	"ntel"
	EDX	"inel"

Table 3-17 Information Returned by CPUID Instruction (Contd.)

Initial EAX Value	Information Provided about the Processor	
01H	EAX	Version Information: Type, Family, Model, and Stepping ID (see Figure 3-5)
	EBX	Bits 07-00: Brand Index Bits 15-08: CLFLUSH line size (Value ∗ 8 = cache line size in bytes) Bits 23-16: Maximum number of addressable IDs for logical processors in this physical package*. Bits 31-24: Initial APIC ID
	ECX EDX	Feature Information (see Figure 3-6 and Table 3-20) Feature Information (see Figure 3-7 and Table 3-21)
		NOTES: * The nearest power-of-2 integer that is not smaller than EBX[23:16] is the number of unique initial APIC IDs reserved for addressing different logical processors in a physical package. This field is only valid if CPUID.1.EDX.HTT[bit 28]= 1.
02H	EAX EBX ECX EDX	Cache and TLB Information (see Table 3-22) Cache and TLB Information Cache and TLB Information Cache and TLB Information
03H	EAX EBX	Reserved. Reserved.
	ECX	Bits 00-31 of 96 bit processor serial number. (Available in Pentium III processor only; otherwise, the value in this register is reserved.)
	EDX	Bits 32-63 of 96 bit processor serial number. (Available in Pentium III processor only; otherwise, the value in this register is reserved.)
		NOTES: Processor serial number (PSN) is not supported in the Pentium 4 processor or later. On all models, use the PSN flag (returned using CPUID) to check for PSN support before accessing the feature. See AP-485, *Intel Processor Identification and the CPUID Instruction* (Order Number 241618) for more information on PSN.
	CPUID leaves > 3 < 80000000 are visible only when IA32_MISC_ENABLE.BOOT_NT4[bit 22] = 0 (default).	
	Deterministic Cache Parameters Leaf	
04H		**NOTES:** Leaf 04H output depends on the initial value in ECX. See also: "INPUT EAX = 4: Returns Deterministic Cache Parameters for each level on page 3-224.

Table 3-17 Information Returned by CPUID Instruction (Contd.)

Initial EAX Value	Information Provided about the Processor	
	EAX	Bits 04-00: Cache Type Field 0 = Null - No more caches 1 = Data Cache 2 = Instruction Cache 3 = Unified Cache 4-31 = Reserved
		Bits 07-05: Cache Level (starts at 1) Bit 08: Self Initializing cache level (does not need SW initialization) Bit 09: Fully Associative cache Bits 13-10: Reserved Bits 25-14: Maximum number of addressable IDs for logical processors sharing this cache*, ** Bits 31-26: Maximum number of addressable IDs for processor cores in the physical package*, ***, ****
	EBX	Bits 11-00: L = System Coherency Line Size* Bits 21-12: P = Physical Line partitions* Bits 31-22: W = Ways of associativity*
	ECX	Bits 31-00: S = Number of Sets*
	EDX	Bit 0: Write-Back Invalidate/Invalidate 0 = WBINVD/INVD from threads sharing this cache acts upon lower level caches for threads sharing this cache. 1 = WBINVD/INVD is not guaranteed to act upon lower level caches of non-originating threads sharing this cache. Bit 1: Cache Inclusiveness 0 = Cache is not inclusive of lower cache levels. 1 = Cache is inclusive of lower cache levels. Bit 2: Complex Cache Indexing 0 = Direct mapped cache. 1 = A complex function is used to index the cache, potentially using all address bits. Bits 31-03: Reserved = 0 **NOTES:** * Add one to the return value to get the result. ** The nearest power-of-2 integer that is not smaller than (1 + EAX[25:14]) is the number of unique initial APIC IDs reserved for addressing different logical processors sharing this cache *** The nearest power-of-2 integer that is not smaller than (1 + EAX[31:26]) is the number of unique Core_IDs reserved for addressing different processor cores in a physical package. Core ID is a subset of bits of the initial APIC ID. ****The returned value is constant for valid initial values in ECX. Valid ECX values start from 0.
	MONITOR/MWAIT Leaf	
05H	EAX	Bits 15-00: Smallest monitor-line size in bytes (default is processor's monitor granularity) Bits 31-16: Reserved = 0

Table 3-17 Information Returned by CPUID Instruction (Contd.)

Initial EAX Value	Information Provided about the Processor	
	EBX	Bits 15-00: Largest monitor-line size in bytes (default is processor's monitor granularity) Bits 31-16: Reserved = 0
	ECX	Bit 00: Enumeration of Monitor-Mwait extensions (beyond EAX and EBX registers) supported Bit 01: Supports treating interrupts as break-event for MWAIT, even when interrupts disabled Bits 31 - 02: Reserved
	EDX	Bits 03 - 00: Number of C0* sub C-states supported using MWAIT Bits 07 - 04: Number of C1* sub C-states supported using MWAIT Bits 11 - 08: Number of C2* sub C-states supported using MWAIT Bits 15 - 12: Number of C3* sub C-states supported using MWAIT Bits 19 - 16: Number of C4* sub C-states supported using MWAIT Bits 31 - 20: Reserved = 0 **NOTE:** * The definition of C0 through C4 states for MWAIT extension are processor-specific C-states, not ACPI C-states.
	Thermal and Power Management Leaf	
06H	EAX	Bit 00: Digital temperature sensor is supported if set Bit 01: Intel Turbo Boost Technology Available (see description of IA32_MISC_ENABLE[38]). Bit 02: ARAT. APIC-Timer-always-running feature is supported if set. Bit 03: Reserved Bit 04: PLN. Power limit notification controls are supported if set. Bit 05: ECMD. Clock modulation duty cycle extension is supported if set. Bit 06: PTM. Package thermal management is supported if set. Bits 31 - 07: Reserved
	EBX	Bits 03 - 00: Number of Interrupt Thresholds in Digital Thermal Sensor Bits 31 - 04: Reserved
	ECX	Bit 00: Hardware Coordination Feedback Capability (Presence of IA32_MPERF and IA32_APERF). The capability to provide a measure of delivered processor performance (since last reset of the counters), as a percentage of expected processor performance at frequency specified in CPUID Brand String Bits 02 - 01: Reserved = 0 Bit 03: The processor supports performance-energy bias preference if CPUID.06H:ECX.SETBH[bit 3] is set and it also implies the presence of a new architectural MSR called IA32_ENERGY_PERF_BIAS (1B0H) Bits 31 - 04: Reserved = 0
	EDX	Reserved = 0
	Structured Extended Feature Flags Enumeration Leaf (Output depends on ECX input value)	
07H		Sub leaf 0 (Input ECX = 0).
	EAX	Bits 31-00: Reports the maximum number of supported leaf 7 sub-leaves.

Table 3-17 Information Returned by CPUID Instruction (Contd.)

Initial EAX Value	Information Provided about the Processor	
	EBX	Bit 00: FSGSBASE. Supports RDFSBASE/RDGSBASE/WRFSBASE/WRGS-BASE if 1. Bit 06: Reserved Bit 07: SMEP. Supports Supervisor Mode Execution Protection if 1. Bit 08: Reserved Bit 09: Supports Enhanced REP MOVSB/STOSB if 1. Bit 10: INVPCID. If 1, supports INVPCID instruction for system software that manages process-context identifiers. Bit 31:11: Reserved
	ECX	Reserved
	EDX	Reserved.
	Direct Cache Access Information Leaf	
09H	EAX	Value of bits [31:0] of IA32_PLATFORM_DCA_CAP MSR (address 1F8H)
	EBX	Reserved
	ECX	Reserved
	EDX	Reserved
	Architectural Performance Monitoring Leaf	
0AH	EAX	Bits 07 - 00: Version ID of architectural performance monitoring Bits 15- 08: Number of general-purpose performance monitoring counter per logical processor Bits 23 - 16: Bit width of general-purpose, performance monitoring counter Bits 31 - 24: Length of EBX bit vector to enumerate architectural performance monitoring events
	EBX	Bit 00: Core cycle event not available if 1 Bit 01: Instruction retired event not available if 1 Bit 02: Reference cycles event not available if 1 Bit 03: Last-level cache reference event not available if 1 Bit 04: Last-level cache misses event not available if 1 Bit 05: Branch instruction retired event not available if 1 Bit 06: Branch mispredict retired event not available if 1 Bits 31- 07: Reserved = 0
	ECX	Reserved = 0
	EDX	Bits 04 - 00: Number of fixed-function performance counters (if Version ID > 1) Bits 12- 05: Bit width of fixed-function performance counters (if Version ID > 1) Reserved = 0
	Extended Topology Enumeration Leaf	

Table 3-17 Information Returned by CPUID Instruction (Contd.)

Initial EAX Value	Information Provided about the Processor	
0BH		**NOTES:** Most of Leaf 0BH output depends on the initial value in ECX. EDX output do not vary with initial value in ECX. ECX[7:0] output always reflect initial value in ECX. All other output value for an invalid initial value in ECX are 0. Leaf 0BH exists if EBX[15:0] is not zero.
	EAX	Bits 04-00: Number of bits to shift right on x2APIC ID to get a unique topology ID of the next level type*. All logical processors with the same next level ID share current level. Bits 31-05: Reserved.
	EBX	Bits 15 - 00: Number of logical processors at this level type. The number reflects configuration as shipped by Intel**. Bits 31- 16: Reserved.
	ECX	Bits 07 - 00: Level number. Same value in ECX input Bits 15 - 08: Level type***. Bits 31 - 16:: Reserved.
	EDX	Bits 31- 00: x2APIC ID the current logical processor. **NOTES:** * Software should use this field (EAX[4:0]) to enumerate processor topology of the system. ** Software must not use EBX[15:0] to enumerate processor topology of the system. This value in this field (EBX[15:0]) is only intended for display/diagnostic purposes. The actual number of logical processors available to BIOS/OS/Applications may be different from the value of EBX[15:0], depending on software and platform hardware configurations. *** The value of the "level type" field is not related to level numbers in any way, higher "level type" values do not mean higher levels. Level type field has the following encoding: 0 : invalid 1 : SMT 2 : Core 3-255 : Reserved
	Processor Extended State Enumeration Main Leaf (EAX = 0DH, ECX = 0)	
0DH		**NOTES:** Leaf 0DH main leaf (ECX = 0).
	EAX	Bits 31-00: Reports the valid bit fields of the lower 32 bits of XCR0. If a bit is 0, the corresponding bit field in XCR0 is reserved. Bit 00: legacy x87 Bit 01: 128-bit SSE Bit 02: 256-bit AVX Bits 31- 03: Reserved

Table 3-17 Information Returned by CPUID Instruction (Contd.)

Initial EAX Value		Information Provided about the Processor
	EBX	Bits 31-00: Maximum size (bytes, from the beginning of the XSAVE/XRSTOR save area) required by enabled features in XCR0. May be different than ECX if some features at the end of the XSAVE save area are not enabled.
	ECX	Bit 31-00: Maximum size (bytes, from the beginning of the XSAVE/XRSTOR save area) of the XSAVE/XRSTOR save area required by all supported features in the processor, i.e all the valid bit fields in XCR0.
	EDX	Bit 31-00: Reports the valid bit fields of the upper 32 bits of XCR0. If a bit is 0, the corresponding bit field in XCR0 is reserved.
	Processor Extended State Enumeration Sub-leaf (EAX = 0DH, ECX = 1)	
	EAX	Bits 31-01: Reserved Bit 00: XSAVEOPT is available;
	EBX	Reserved
	ECX	Reserved
	EDX	Reserved
	Processor Extended State Enumeration Sub-leaves (EAX = 0DH, ECX = n, n > 1)	
0DH		**NOTES:** Leaf 0DH output depends on the initial value in ECX. If ECX contains an invalid sub leaf index, EAX/EBX/ECX/EDX return 0. Each valid sub-leaf index maps to a valid bit in the XCR0 register starting at bit position 2
	EAX	Bits 31-0: The size in bytes (from the offset specified in EBX) of the save area for an extended state feature associated with a valid sub-leaf index, *n*. This field reports 0 if the sub-leaf index, *n*, is invalid*.
	EBX	Bits 31-0: The offset in bytes of this extended state component's save area from the beginning of the XSAVE/XRSTOR area. This field reports 0 if the sub-leaf index, *n*, is invalid*.
	ECX	This field reports 0 if the sub-leaf index, *n*, is invalid*; otherwise it is reserved.
	EDX	This field reports 0 if the sub-leaf index, *n*, is invalid*; otherwise it is reserved.
	Unimplemented CPUID Leaf Functions	
40000000H - 4FFFFFFFH		Invalid. No existing or future CPU will return processor identification or feature information if the initial EAX value is in the range 40000000H to 4FFFFFFFH.
	Extended Function CPUID Information	
80000000H	EAX	Maximum Input Value for Extended Function CPUID Information (see Table 3-18).
	EBX	Reserved
	ECX	Reserved
	EDX	Reserved

Table 3-17 Information Returned by CPUID Instruction (Contd.)

Initial EAX Value	Information Provided about the Processor	
80000001H	EAX	Extended Processor Signature and Feature Bits.
	EBX	Reserved
	ECX	Bit 00: LAHF/SAHF available in 64-bit mode Bits 31-01 Reserved
	EDX	Bits 10-00: Reserved Bit 11: SYSCALL/SYSRET available (when in 64-bit mode) Bits 19-12: Reserved = 0 Bit 20: Execute Disable Bit available Bits 25-21: Reserved = 0 Bit 26: 1-GByte pages are available if 1 Bit 27: RDTSCP and IA32_TSC_AUX are available if 1 Bits 28: Reserved = 0 Bit 29: Intel® 64 Architecture available if 1 Bits 31-30: Reserved = 0
80000002H	EAX	Processor Brand String
	EBX	Processor Brand String Continued
	ECX	Processor Brand String Continued
	EDX	Processor Brand String Continued
80000003H	EAX	Processor Brand String Continued
	EBX	Processor Brand String Continued
	ECX	Processor Brand String Continued
	EDX	Processor Brand String Continued
80000004H	EAX	Processor Brand String Continued
	EBX	Processor Brand String Continued
	ECX	Processor Brand String Continued
	EDX	Processor Brand String Continued
80000005H	EAX	Reserved = 0
	EBX	Reserved = 0
	ECX	Reserved = 0
	EDX	Reserved = 0
80000006H	EAX	Reserved = 0
	EBX	Reserved = 0
	ECX	Bits 07-00: Cache Line size in bytes Bits 11-08: Reserved Bits 15-12: L2 Associativity field * Bits 31-16: Cache size in 1K units
	EDX	Reserved = 0

Table 3-17 Information Returned by CPUID Instruction (Contd.)

Initial EAX Value	Information Provided about the Processor	
		NOTES: * L2 associativity field encodings: 00H - Disabled 01H - Direct mapped 02H - 2-way 04H - 4-way 06H - 8-way 08H - 16-way 0FH - Fully associative
80000007H	EAX	Reserved = 0
	EBX	Reserved = 0
	ECX	Reserved = 0
	EDX	Bits 07-00: Reserved = 0 Bit 08: Invariant TSC available if 1 Bits 31-09: Reserved = 0
80000008H	EAX	Linear/Physical Address size Bits 07-00: #Physical Address Bits* Bits 15-8: #Linear Address Bits Bits 31-16: Reserved = 0
	EBX	Reserved = 0
	ECX	Reserved = 0
	EDX	Reserved = 0 NOTES: * If CPUID.80000008H:EAX[7:0] is supported, the maximum physical address number supported should come from this field.

...

7. Updates to Chapter 4, Volume 2B

Change bars show changes to Chapter 4 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B:* Instruction Set Reference, M-Z.

...

4.1.4 Polarity

IntRes1 may then be further modified by performing a 1's complement, according to the value of the Imm8 Control Byte bit[4]. Optionally, a mask may be used such that only those IntRes1 bits which correspond to "valid" reg/mem input elements are complemented (note that the definition of a valid input element is dependant on the specific opcode and is defined in each opcode's description). The result of the possible negation is referred to as IntRes2.

...

MOVD/MOVQ—Move Doubleword/Move Quadword

Opcode/ Instruction	Op/ En	64/32-bit Mode	CPUID Feature Flag	Description
0F 6E /*r* MOVD *mm, r/m32*	RM	V/V	MMX	Move doubleword from *r/m32* to *mm.*
REX.W + 0F 6E /*r* MOVQ *mm, r/m64*	RM	V/N.E.	MMX	Move quadword from *r/m64* to *mm.*
0F 7E /*r* MOVD *r/m32, mm*	MR	V/V	MMX	Move doubleword from *mm* to *r/m32.*
REX.W + 0F 7E /*r* MOVQ *r/m64, mm*	MR	V/N.E.	MMX	Move quadword from *mm* to *r/m64.*
VEX.128.66.0F.W0 6E / VMOVD xmm1, r32/m32	RM	V/V	AVX	Move doubleword from r/m32 to xmm1.
VEX.128.66.0F.W1 6E /r VMOVQ xmm1, r64/m64	RM	V/N.E.	AVX	Move quadword from r/m64 to xmm1.
66 0F 6E /*r* MOVD *xmm, r/m32*	RM	V/V	SSE2	Move doubleword from *r/m32* to *xmm.*
66 REX.W 0F 6E /*r* MOVQ *xmm, r/m64*	RM	V/N.E.	SSE2	Move quadword from *r/m64* to *xmm.*
66 0F 7E /*r* MOVD *r/m32, xmm*	MR	V/V	SSE2	Move doubleword from *xmm* register to *r/m32.*
66 REX.W 0F 7E /*r* MOVQ *r/m64, xmm*	MR	V/N.E.	SSE2	Move quadword from *xmm* register to *r/m64.*
VEX.128.66.0F.W0 7E /r VMOVD r32/m32, xmm1	MR	V/V	AVX	Move doubleword from xmm1 register to r/m32.
VEX.128.66.0F.W1 7E /r VMOVQ r64/m64, xmm1	MR	V/N.E.	AVX	Move quadword from xmm1 register to r/m64.

...

MOVNTI—Store Doubleword Using Non-Temporal Hint

Opcode	Instruction	Op/ En	64-Bit Mode	Compat/ Leg Mode	Description
0F C3 /*r*	MOVNTI *m32, r32*	MR	Valid	Valid	Move doubleword from *r32* to *m32* using non-temporal hint.
REX.W + 0F C3 /*r*	MOVNTI *m64, r64*	MR	Valid	N.E.	Move quadword from *r64* to *m64* using non-temporal hint.

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
MR	ModRM:r/m (w)	ModRM:reg (r)	NA	NA

Description

Moves the doubleword integer in the source operand (second operand) to the destination operand (first operand) using a non-temporal hint to minimize cache pollution during the write to memory. The source operand is a general-purpose register. The destination operand is a 32-bit memory location.

The non-temporal hint is implemented by using a write combining (WC) memory type protocol when writing the data to memory. Using this protocol, the processor does not write the data into the cache hierarchy, nor does it fetch the corresponding cache line from memory into the cache hierarchy. The memory type of the region being written to can override the non-temporal hint, if the memory address specified for the non-temporal store is in an uncacheable (UC) or write protected (WP) memory region. For more information on non-temporal stores, see "Caching of Temporal vs. Non-Temporal Data" in Chapter 10 in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1*.

Because the WC protocol uses a weakly-ordered memory consistency model, a fencing operation implemented with the SFENCE or MFENCE instruction should be used in conjunction with MOVNTI instructions if multiple processors might use different memory types to read/write the destination memory locations.

In 64-bit mode, the instruction's default operation size is 32 bits. Use of the REX.R prefix permits access to additional registers (R8-R15). Use of the REX.W prefix promotes operation to 64 bits. See the summary chart at the beginning of this section for encoding data and limits.

Operation

```
DEST ← SRC;
```

Intel C/C++ Compiler Intrinsic Equivalent

MOVNTI:	void _mm_stream_si32 (int *p, int a)
MOVNTI:	void _mm_stream_si64(__int64 *p, __int64 a)

SIMD Floating-Point Exceptions

None.

Protected Mode Exceptions

#GP(0)	For an illegal memory operand effective address in the CS, DS, ES, FS or GS segments.
#SS(0)	For an illegal address in the SS segment.
#PF(fault-code)	For a page fault.
#UD	If CPUID.01H:EDX.SSE2[bit 26] = 0.
	If the LOCK prefix is used.

Real-Address Mode Exceptions

#GP	If a memory operand is not aligned on a 16-byte boundary, regardless of segment.
	If any part of the operand lies outside the effective address space from 0 to FFFFH.
#UD	If CPUID.01H:EDX.SSE2[bit 26] = 0.
	If the LOCK prefix is used.

Virtual-8086 Mode Exceptions

Same exceptions as in real address mode.

#PF(fault-code)	For a page fault.

Compatibility Mode Exceptions

Same exceptions as in protected mode.

64-Bit Mode Exceptions

#SS(0)	If a memory address referencing the SS segment is in a non-canonical form.
#GP(0)	If the memory address is in a non-canonical form.
#PF(fault-code)	For a page fault.
#UD	If CPUID.01H:EDX.SSE2[bit 26] = 0.
	If the LOCK prefix is used.
#AC(0)	If alignment checking is enabled and an unaligned memory reference is made while the current privilege level is 3.

...

MOVQ—Move Quadword

Opcode	Instruction	Op/ En	64/32-bit Mode	CPUID Feature Flag	Description
0F 6F /*r*	MOVQ *mm, mm/m64*	RM	V/V	MMX	Move quadword from *mm/m64* to *mm*.
0F 7F /*r*	MOVQ *mm/m64, mm*	MR	V/V	MMX	Move quadword from *mm* to *mm/m64*.
F3 0F 7E	MOVQ *xmm1, xmm2/m64*	RM	V/V	SSE2	Move quadword from *xmm2/mem64* to *xmm1*.
66 0F D6	MOVQ *xmm2/m64, xmm1*	MR	V/V	SSE2	Move quadword from *xmm1* to *xmm2/mem64*.

...

XADD—Exchange and Add

Opcode	Instruction	Op/ En	64-Bit Mode	Compat/ Leg Mode	Description
0F C0 /*r*	XADD *r/m8, r8*	MR	Valid	Valid	Exchange *r8* and *r/m8*; load sum into *r/m8*.
REX + 0F C0 /*r*	XADD *r/m8*, r8**	MR	Valid	N.E.	Exchange *r8* and *r/m8*; load sum into *r/m8*.
0F C1 /*r*	XADD *r/m16, r16*	MR	Valid	Valid	Exchange *r16* and *r/m16*; load sum into *r/m16*.
0F C1 /*r*	XADD *r/m32, r32*	MR	Valid	Valid	Exchange *r32* and *r/m32*; load sum into *r/m32*.
REX.W + 0F C1 /*r*	XADD *r/m64, r64*	MR	Valid	N.E.	Exchange *r64* and *r/m64*; load sum into *r/m64*.

NOTES:

* In 64-bit mode, r/m8 can not be encoded to access the following byte registers if a REX prefix is used: AH, BH, CH, DH.

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
MR	ModRM:r/m (r, w)	ModRM:reg (W)	NA	NA

...

8. Updates to Chapter 6, Volume 3A

Change bars show changes to Chapter 6 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A:* System Programming Guide, Part 1.

...

Interrupt 14—Page-Fault Exception (#PF)

Exception Class Fault.

Description

Indicates that, with paging enabled (the PG flag in the CR0 register is set), the processor detected one of the following conditions while using the page-translation mechanism to translate a linear address to a physical address:

- The P (present) flag in a page-directory or page-table entry needed for the address translation is clear, indicating that a page table or the page containing the operand is not present in physical memory.
- The procedure does not have sufficient privilege to access the indicated page (that is, a procedure running in user mode attempts to access a supervisor-mode page).

- Code running in user mode attempts to write to a read-only page. In the Intel486 and later processors, if the WP flag is set in CR0, the page fault will also be triggered by code running in supervisor mode that tries to write to a read-only page.
- An instruction fetch to a linear address that translates to a physical address in a memory page with the execute-disable bit set (for information about the execute-disable bit, see Chapter 4, "Paging").
- One or more reserved bits in page directory entry are set to 1. See description below of RSVD error code flag.

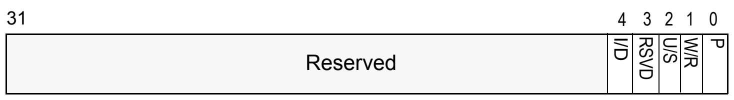
The exception handler can recover from page-not-present conditions and restart the program or task without any loss of program continuity. It can also restart the program or task after a privilege violation, but the problem that caused the privilege violation may be uncorrectable.

See also: Section 4.7, "Page-Fault Exceptions."

Exception Error Code

Yes (special format). The processor provides the page-fault handler with two items of information to aid in diagnosing the exception and recovering from it:

- An error code on the stack. The error code for a page fault has a format different from that for other exceptions (see Figure 6-9). The error code tells the exception handler four things:
  - — The P flag indicates whether the exception was due to a not-present page (0) or to either an access rights violation or the use of a reserved bit (1).
  - — The W/R flag indicates whether the memory access that caused the exception was a read (0) or write (1).
  - — The U/S flag indicates whether the processor was executing at user mode (1) or supervisor mode (0) at the time of the exception.
  - — The RSVD flag indicates that the processor detected 1s in reserved bits of the page directory, when the PSE or PAE flags in control register CR4 are set to 1. Note:
    - The PSE flag is only available in recent Intel 64 and IA-32 processors including the Pentium 4, Intel Xeon, P6 family, and Pentium processors.
    - The PAE flag is only available on recent Intel 64 and IA-32 processors including the Pentium 4, Intel Xeon, and P6 family processors.
    - In earlier IA-32 processors, the bit position of the RSVD flag is reserved and is cleared to 0.
  - — The I/D flag indicates whether the exception was caused by an instruction fetch. This flag is reserved and cleared to 0 if CR4.SMEP = 0 (supervisor-mode execution prevention is either unsupported or not enabled) and either CR4.PAE = 0 (32-bit paging is in use) or IA32_EFER.NXE= 0 (the execute-disable feature is either unsupported or not enabled). See Section 4.7, "Page-Fault Exceptions," for details.



P	0	The fault was caused by a non-present page.
	1	The fault was caused by a page-level protection violation.
W/R	0	The access causing the fault was a read.
	1	The access causing the fault was a write.
U/S	0	The access causing the fault originated when the processor was executing in supervisor mode.
	1	The access causing the fault originated when the processor was executing in user mode.
RSVD	0	The fault was not caused by reserved bit violation.
	1	The fault was caused by reserved bits set to 1 in a page directory.
I/D	0	The fault was not caused by an instruction fetch.
	1	The fault was caused by an instruction fetch.

Figure 6-9 Page-Fault Error Code

- The contents of the CR2 register. The processor loads the CR2 register with the 32-bit linear address that generated the exception. The page-fault handler can use this address to locate the corresponding page directory and page-table entries. Another page fault can potentially occur during execution of the page-fault handler; the handler should save the contents of the CR2 register before a second page fault can occur.[1] If a page fault is caused by a page-level protection violation, the access flag in the page-directory entry is set when the fault occurs. The behavior of IA-32 processors regarding the access flag in the corresponding page-table entry is model specific and not architecturally defined.

...

9. Updates to Chapter 8, Volume 3A

Change bars show changes to Chapter 8 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A:* System Programming Guide, Part 1.

8.2.4 Fast-String Operation and Out-of-Order Stores

Section 7.3.9.3 of *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1* described an optimization of repeated string operations called **fast-string operation**.

1. Processors update CR2 whenever a page fault is detected. If a second page fault occurs while an earlier page fault is being delivered, the faulting linear address of the second fault will overwrite the contents of CR2 (replacing the previous address). These updates to CR2 occur even if the page fault results in a double fault or occurs during the delivery of a double fault.

As explained in that section, the stores produced by fast-string operation may appear to execute out of order. Software dependent upon sequential store ordering should not use string operations for the entire data structure to be stored. Data and semaphores should be separated. Order-dependent code should write to a discrete semaphore variable after any string operations to allow correctly ordered data to be seen by all processors. Atomicity of load and store operations is guaranteed only for native data elements of the string with native data size, and only if they are included in a single cache line.

Section 8.2.4.1 and Section 8.2.4.2 provide further explain and examples.

8.2.4.1 Memory-Ordering Model for String Operations on Write-Back (WB) Memory

This section deals with the memory-ordering model for string operations on write-back (WB) memory for the Intel 64 architecture.

The memory-ordering model respects the follow principles:

1. Stores within a single string operation may be executed out of order.
2. Stores from separate string operations (for example, stores from consecutive string operations) do not execute out of order. All the stores from an earlier string operation will complete before any store from a later string operation.
3. String operations are not reordered with other store operations.

...

8.3 SERIALIZING INSTRUCTIONS

The Intel 64 and IA-32 architectures define several **serializing instructions**. These instructions force the processor to complete all modifications to flags, registers, and memory by previous instructions and to drain all buffered writes to memory before the next instruction is fetched and executed. For example, when a MOV to control register instruction is used to load a new value into control register CR0 to enable protected mode, the processor must perform a serializing operation before it enters protected mode. This serializing operation ensures that all operations that were started while the processor was in real-address mode are completed before the switch to protected mode is made.

The concept of serializing instructions was introduced into the IA-32 architecture with the Pentium processor to support parallel instruction execution. Serializing instructions have no meaning for the Intel486 and earlier processors that do not implement parallel instruction execution.

It is important to note that executing of serializing instructions on P6 and more recent processor families constrain speculative execution because the results of speculatively executed instructions are discarded. The following instructions are serializing instructions:

- **Privileged serializing instructions** — INVD, INVEPT, INVLPG, INVVPID, LGDT, LIDT, LLDT, LTR, MOV (to control register, with the exception of MOV CR8[1]), MOV (to debug register), WBINVD, and WRMSR[2].

1. MOV CR8 is not defined architecturally as a serializing instruction.
2. WRMSR to the IA32_TSC_DEADLINE MSR (MSR index 6E0H) and the X2APIC MSRs (MSR indices 802H to 83FH) are not serializing.

- **Non-privileged serializing instructions** — CPUID, IRET, and RSM.

When the processor serializes instruction execution, it ensures that all pending memory transactions are completed (including writes stored in its store buffer) before it executes the next _instruction. Nothing can pass a serializing instruction and a serializing instruction cannot pass any other instruction (read, write, instruction fetch, or I/O). For example, CPUID can be executed at any privilege level to serialize instruction execution with no effect on program flow, except that the EAX, EBX, ECX, and EDX registers are modified.

The following instructions are memory-ordering instructions, not serializing instructions. These drain the data memory subsystem. They do not serialize the instruction execution stream:[1]

- **Non-privileged memory-ordering instructions** — SFENCE, LFENCE, and MFENCE.

The SFENCE, LFENCE, and MFENCE instructions provide more granularity in controlling the serialization of memory loads and stores (see Section 8.2.5, "Strengthening or Weakening the Memory-Ordering Model").

The following additional information is worth noting regarding serializing instructions:

- The processor does not write back the contents of modified data in its data cache to external memory when it serializes instruction execution. Software can force modified data to be written back by executing the WBINVD instruction, which is a serializing instruction. The amount of time or cycles for WBINVD to complete will vary due to the size of different cache hierarchies and other factors. As a consequence, the use of the WBINVD instruction can have an impact on interrupt/event response time.

...

8.12 PROGRAMMING THE LINT0 AND LINT1 INPUTS

The following procedure describes how to program the LINT0 and LINT1 local APIC pins on a processor after multiple processors have been booted and initialized (as described in Section 8.11, "MP Initialization For P6 Family Processors"). In this example, LINT0 is programmed to be the ExtINT pin and LINT1 is programmed to be the NMI pin.

8.12.1 Constants

The following constants are defined:

```
LVT1EQU 0FEE00350H
LVT2EQU 0FEE00360H
LVT3 EQU 0FEE00370H
SVR   EQU 0FEE000F0H
```

1. LFENCE does provide some guarantees on instruction ordering. It does not execute until all prior instructions have completed locally, and no later instruction begins execution until LFENCE completes.

8.12.2 LINT[0:1] Pins Programming Procedure

Use the following to program the LINT[1:0] pins:

1. Mask 8259 interrupts.
2. Enable APIC via SVR (spurious vector register) if not already enabled.

```
MOV ESI, SVR                ; address of SVR
MOV EAX, [ESI]
OR  EAX, APIC_ENABLED       ; set bit 8 to enable (0 on reset)
MOV [ESI], EAX
```

3. Program LVT1 as an ExtINT which delivers the signal to the INTR signal of all processors cores listed in the destination as an interrupt that originated in an externally connected interrupt controller.

```
MOV ESI, LVT1
MOV EAX, [ESI]
AND EAX, 0FFFE58FFH; mask off bits 8-10, 12, 14 and 16
OR  EAX, 700H; Bit 16=0 for not masked, Bit 15=0 for edge
    ; triggered, Bit 13=0 for high active input
    ; polarity, Bits 8-10 are 111b for ExtINT
MOV [ESI], EAX; Write to LVT1
```

4. Program LVT2 as NMI, which delivers the signal on the NMI signal of all processor cores listed in the destination.

```
MOV ESI, LVT2
MOV EAX, [ESI]
AND EAX, 0FFFE58FFH; mask off bits 8-10 and 15
OR  EAX, 000000400H; Bit 16=0 for not masked, Bit 15=0 edge
    ; triggered, Bit 13=0 for high active input
    ; polarity, Bits 8-10 are 100b for NMI
MOV [ESI], EAX; Write to LVT2
    ;Unmask 8259 interrupts and allow NMI.
```

...

10. Updates to Chapter 9, Volume 3A

Change bars show changes to Chapter 9 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A:* System Programming Guide, Part 1.

...

9.1.2 Processor Built-In Self-Test (BIST)

Hardware may request that the BIST be performed at power-up. The EAX register is cleared (0H) if the processor passes the BIST. A nonzero value in the EAX register after the BIST indicates that a processor fault was detected. If the BIST is not requested, the contents of the EAX register after a hardware reset is 0H.

The overhead for performing a BIST varies between processor families. For example, the BIST takes approximately 30 million processor clock periods to execute on the Pentium 4 processor. This clock count is model-specific; Intel reserves the right to change the number of periods for any Intel 64 or IA-32 processor, without notification.

Table 9-1 IA-32 Processor States Following Power-up, Reset, or INIT

Register	Pentium 4 and Intel Xeon Processor	P6 Family Processor	Pentium Processor
EFLAGS[1]	00000002H	00000002H	00000002H
EIP	0000FFF0H	0000FFF0H	0000FFF0H
CR0	60000010H[2]	60000010H[2]	60000010H[2]
CR2, CR3, CR4	00000000H	00000000H	00000000H

CS	Selector = F000H
	Base = FFFF0000H
	Limit = FFFFH
	AR = Present, R/W, Accessed
		Selector = F000H
		Base = FFFF0000H
		Limit = FFFFH
		AR = Present, R/W, Accessed
			Selector = F000H
			Base = FFFF0000H
			Limit = FFFFH
			AR = Present, R/W, Accessed

SS, DS, ES, FS, GS	Selector = 0000H
	Base = 00000000H
	Limit = FFFFH
	AR = Present, R/W, Accessed
		Selector = 0000H
		Base = 00000000H
		Limit = FFFFH
		AR = Present, R/W, Accessed
			Selector = 0000H
			Base = 00000000H
			Limit = FFFFH
			AR = Present, R/W, Accessed

EDX	00000FxxH	000n06xxH[3]	000005xxH
EAX	0[4]	0[4]	0[4]
EBX, ECX, ESI, EDI, EBP, ESP	00000000H	00000000H	00000000H
ST0 through ST7[5]	Pwr up or Reset: +0.0 FINIT/FNINIT: Unchanged	Pwr up or Reset: +0.0 FINIT/FNINIT: Unchanged	Pwr up or Reset: +0.0 FINIT/FNINIT: Unchanged
x87 FPU Control Word[5]	Pwr up or Reset: 0040H FINIT/FNINIT: 037FH	Pwr up or Reset: 0040H FINIT/FNINIT: 037FH	Pwr up or Reset: 0040H FINIT/FNINIT: 037FH
x87 FPU Status Word[5]	Pwr up or Reset: 0000H FINIT/FNINIT: 0000H	Pwr up or Reset: 0000H FINIT/FNINIT: 0000H	Pwr up or Reset: 0000H FINIT/FNINIT: 0000H
x87 FPU Tag Word[5]	Pwr up or Reset: 5555H FINIT/FNINIT: FFFFH	Pwr up or Reset: 5555H FINIT/FNINIT: FFFFH	Pwr up or Reset: 5555H FINIT/FNINIT: FFFFH
x87 FPU Data Operand and CS Seg. Selectors[5]	Pwr up or Reset: 0000H FINIT/FNINIT: 0000H	Pwr up or Reset: 0000H FINIT/FNINIT: 0000H	Pwr up or Reset: 0000H FINIT/FNINIT: 0000H
x87 FPU Data Operand and Inst. Pointers[5]	Pwr up or Reset: 00000000H FINIT/FNINIT: 00000000H	Pwr up or Reset: 00000000H FINIT/FNINIT: 00000000H	Pwr up or Reset: 00000000H FINIT/FNINIT: 00000000H
MM0 through MM7[5]	Pwr up or Reset: 0000000000000000H INIT or FINIT/FNINIT: Unchanged	Pentium II and Pentium III Processors Only— Pwr up or Reset: 0000000000000000H INIT or FINIT/FNINIT: Unchanged	Pentium with MMX Technology Only— Pwr up or Reset: 0000000000000000H INIT or FINIT/FNINIT: Unchanged
XMM0 through XMM7	Pwr up or Reset: 0000000000000000H INIT: Unchanged	Pentium III processor Only— Pwr up or Reset: 0000000000000000H INIT: Unchanged	NA
MXCSR	Pwr up or Reset: 1F80H INIT: Unchanged	Pentium III processor only- Pwr up or Reset: 1F80H INIT: Unchanged	NA

Table 9-1 IA-32 Processor States Following Power-up, Reset, or INIT (Contd.)

Register	Pentium 4 and Intel Xeon Processor	P6 Family Processor	Pentium Processor

GDTR, IDTR	Base = 00000000H
	Limit = FFFFH
	AR = Present, R/W
		Base = 00000000H
		Limit = FFFFH
		AR = Present, R/W
			Base = 00000000H
			Limit = FFFFH
			AR = Present, R/W

LDTR, Task Register	Selector = 0000H
	Base = 00000000H
	Limit = FFFFH
	AR = Present, R/W
		Selector = 0000H
		Base = 00000000H
		Limit = FFFFH
		AR = Present, R/W
			Selector = 0000H
			Base = 00000000H
			Limit = FFFFH
			AR = Present, R/W

DR0, DR1, DR2, DR3	00000000H	00000000H	00000000H
DR6	FFFF0FF0H	FFFF0FF0H	FFFF0FF0H
DR7	00000400H	00000400H	00000400H
Time-Stamp Counter	Power up or Reset: 0H INIT: Unchanged	Power up or Reset: 0H INIT: Unchanged	Power up or Reset: 0H INIT: Unchanged
Perf. Counters and Event Select	Power up or Reset: 0H INIT: Unchanged	Power up or Reset: 0H INIT: Unchanged	Power up or Reset: 0H INIT: Unchanged
All Other MSRs	Pwr up or Reset: Undefined INIT: Unchanged	Pwr up or Reset: Undefined INIT: Unchanged	Pwr up or Reset: Undefined INIT: Unchanged
Data and Code Cache, TLBs	Invalid[6]	Invalid[6]	Invalid[6]
Fixed MTRRs	Pwr up or Reset: Disabled INIT: Unchanged	Pwr up or Reset: Disabled INIT: Unchanged	Not Implemented
Variable MTRRs	Pwr up or Reset: Disabled INIT: Unchanged	Pwr up or Reset: Disabled INIT: Unchanged	Not Implemented
Machine-Check Architecture	Pwr up or Reset: Undefined INIT: Unchanged	Pwr up or Reset: Undefined INIT: Unchanged	Not Implemented
APIC	Pwr up or Reset: Enabled INIT: Unchanged	Pwr up or Reset: Enabled INIT: Unchanged	Pwr up or Reset: Enabled INIT: Unchanged

NOTES:

1. The 10 most-significant bits of the EFLAGS register are undefined following a reset. Software should not depend on the states of any of these bits.
2. The CD and NW flags are unchanged, bit 4 is set to 1, all other bits are cleared.
3. Where "n" is the Extended Model Value for the respective processor.
4. If Built-In Self-Test (BIST) is invoked on power up or reset, EAX is 0 only if all tests passed. (BIST cannot be invoked during an INIT.)
5. The state of the x87 FPU and MMX registers is not changed by the execution of an INIT.
6. Internal caches are invalid after power-up and RESET, but left unchanged with an INIT.

...

11. Updates to Chapter 17, Volume 3B

Change bars show changes to Chapter 17 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B:* System Programming Guide, Part 2.

...

CHAPTER 17 DEBUGGING, BRANCH PROFILING, AND TIME-STAMP COUNTER

Intel 64 and IA-32 architectures provide debug facilities for use in debugging code and monitoring performance. These facilities are valuable for debugging application software, system software, and multitasking operating systems. Debug support is accessed using debug registers (DR0 through DR7) and model-specific registers (MSRs):

- Debug registers hold the addresses of memory and I/O locations called breakpoints. Breakpoints are user-selected locations in a program, a data-storage area in memory, or specific I/O ports. They are set where a programmer or system designer wishes to halt execution of a program and examine the state of the processor by invoking debugger software. A debug exception (#DB) is generated when a memory or I/O access is made to a breakpoint address.
- MSRs monitor branches, interrupts, and exceptions; they record addresses of the last branch, interrupt or exception taken and the last branch taken before an interrupt or exception.

...

17.3.1.2 Data Memory and I/O Breakpoint Exception Conditions

Data memory and I/O breakpoints are reported when the processor attempts to access a memory or I/O address specified in a breakpoint-address register (DR0 through DR3) that has been set up to detect data or I/O accesses (R/W flag is set to 1, 2, or 3). The processor generates the exception after it executes the instruction that made the access, so these breakpoint condition causes a trap-class exception to be generated.

Because data breakpoints are traps, an instruction that writes memory overwrites the original data before the debug exception generated by a data breakpoint is generated. If a debugger needs to save the contents of a write breakpoint location, it should save the original contents before setting the breakpoint. The handler can report the saved value after the breakpoint is triggered. The address in the debug registers can be used to locate the new value stored by the instruction that triggered the breakpoint.

If a data breakpoint is detected during an iteration of a string instruction executed with fast-string operation (see Section 7.3.9.3 of *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1*), delivery of the resulting debug exception may be delayed until completion of the corresponding group of iterations.

Intel486 and later processors ignore the GE and LE flags in DR7. In Intel386 processors, exact data breakpoint matching does not occur unless it is enabled by setting the LE and/or the GE flags.

For repeated INS and OUTS instructions that generate an I/O-breakpoint debug exception, the processor generates the exception after the completion of the first iteration.

Repeated INS and OUTS instructions generate a data-breakpoint debug exception after the iteration in which the memory address breakpoint location is accessed.

...

17.4.9 BTS and DS Save Area

The **Debug store (DS)** feature flag (bit 21), returned by CPUID.1:EDX[21] Indicates that the processor provides the debug store (DS) mechanism. This mechanism allows BTMs to be stored in a memory-resident BTS buffer. See Section 17.4.5, "Branch Trace Store (BTS)." Precise event-based sampling (PEBS, see Section 18.4.4, "Precise Event Based Sampling (PEBS),") also uses the DS save area provided by debug store mechanism. When CPUID.1:EDX[21] is set, the following BTS facilities are available:

- The BTS_UNAVAILABLE flag in the IA32_MISC_ENABLE MSR indicates (when clear) the availability of the BTS facilities, including the ability to set the BTS and BTINT bits in the MSR_DEBUGCTLA MSR.
- The IA32_DS_AREA MSR can be programmed to point to the DS save area.

The debug store (DS) save area is a software-designated area of memory that is used to collect the following two types of information:

- **Branch records —** When the BTS flag in the IA32_DEBUGCTL MSR is set, a branch record is stored in the BTS buffer in the DS save area whenever a taken branch, interrupt, or exception is detected.
- **PEBS records —** When a performance counter is configured for PEBS, a PEBS record is stored in the PEBS buffer in the DS save area after the counter overflow occurs. This record contains the architectural state of the processor (state of the 8 general purpose registers, EIP register, and EFLAGS register) at the next occurrence of the PEBS event that caused the counter to overflow. When the state information has been logged, the counter is automatically reset to a preselected value, and event counting begins again.

NOTE

On processors based on Intel Core microarchitecture and for Intel Atom processor family, PEBS is supported only for a subset of the performance events.

NOTES

DS save area and recording mechanism is not available in the SMM. The feature is disabled on transition to the SMM mode. Similarly DS recording is disabled on the generation of a machine check exception and is cleared on processor RESET and INIT. DS recording is available in real address mode.

The BTS and PEBS facilities may not be available on all processors. The availability of these facilities is indicated by the BTS_UNAVAILABLE and PEBS_UNAVAILABLE flags, respectively, in the IA32_MISC_ENABLE MSR (see Chapter 34).

...

12. Updates to Chapter 18, Volume 3B

Change bars show changes to Chapter 18 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B:* System Programming Guide, Part 2.

...

18.1 PERFORMANCE MONITORING OVERVIEW

Performance monitoring was introduced in the Pentium processor with a set of model-specific performance-monitoring counter MSRs. These counters permit selection of processor performance parameters to be monitored and measured. The information obtained from these counters can be used for tuning system and compiler performance.

In Intel P6 family of processors, the performance monitoring mechanism was enhanced to permit a wider selection of events to be monitored and to allow greater control events to be monitored. Next, Pentium 4 and Intel Xeon processors introduced a new performance monitoring mechanism and new set of performance events.

The performance monitoring mechanisms and performance events defined for the Pentium, P6 family, Pentium 4, and Intel Xeon processors are not architectural. They are all model specific (not compatible among processor families). Intel Core Solo and Intel Core Duo processors support a set of architectural performance events and a set of non-architectural performance events. Processors based on Intel Core microarchitecture and Intel® Atom™ microarchitecture support enhanced architectural performance events and non-architectural performance events.

Starting with Intel Core Solo and Intel Core Duo processors, there are two classes of performance monitoring capabilities. The first class supports events for monitoring performance using counting or sampling usage. These events are non-architectural and vary from one processor model to another. They are similar to those available in Pentium M processors. These non-architectural performance monitoring events are specific to the microarchitecture and may change with enhancements. They are discussed in Section 18.3, "Performance Monitoring (Intel® Core™ Solo and Intel® Core™ Duo Processors)." Non-architectural events for a given microarchitecture can not be enumerated using CPUID; and they are listed in Chapter 19, "Performance-Monitoring Events."

The second class of performance monitoring capabilities is referred to as architectural performance monitoring. This class supports the same counting and sampling usages, with a smaller set of available events. The visible behavior of architectural performance events is consistent across processor implementations. Availability of architectural performance monitoring capabilities is enumerated using the CPUID.0AH. These events are discussed in Section 18.2.

See also:

— Section 18.2, "Architectural Performance Monitoring"

— Section 18.3, "Performance Monitoring (Intel® Core™ Solo and Intel® Core™ Duo Processors)"

— Section 18.4, "Performance Monitoring (Processors Based on Intel® Core™ Microarchitecture)"

— Section 18.5, "Performance Monitoring (Processors Based on Intel® Atom™ Microarchitecture)"

- Section 18.6, “Performance Monitoring for Processors Based on Intel® Microarchitecture Code Name Nehalem”
- Section 18.7, “Performance Monitoring for Processors Based on Intel® Microarchitecture Code Name Westmere”
- Section 18.8, “Performance Monitoring for Processors Based on Intel® Microarchitecture Code Name Sandy Bridge”
- Section 18.9, “Next Generation Intel Core Processor Performance Monitoring Facility”
- Section 18.10, “Performance Monitoring (Processors Based on Intel NetBurst® Microarchitecture)”
- Section 18.11, “Performance Monitoring and Intel Hyper-Threading Technology in Processors Based on Intel NetBurst® Microarchitecture”
- Section 18.14, “Performance Monitoring and Dual-Core Technology”
- Section 18.15, “Performance Monitoring on 64-bit Intel Xeon Processor MP with Up to 8-MByte L3 Cache”
- Section 18.17, “Performance Monitoring (P6 Family Processor)”
- Section 18.18, “Performance Monitoring (Pentium Processors)”

...

18.8 PERFORMANCE MONITORING FOR PROCESSORS BASED ON INTEL® MICROARCHITECTURE CODE NAME SANDY BRIDGE

Intel Core i7, i5, i3 processors 2xxx series are based on Intel microarchitecture code name Sandy Bridge, this section describes the performance monitoring facilities provided in the processor core. The core PMU supports architectural performance monitoring capability with version ID 3 (see Section 18.2.2.2) and a host of non-architectural monitoring capabilities.

Architectural performance monitoring events and non-architectural monitoring events are programmed using fixed counters and programmable counters/event select MSRS described in Section 18.2.2.2.

The core PMU’s capability is similar to those described in Section 18.6.1 and Section 18.7, with some differences and enhancements relative to Intel microarchitecture code name Westmere summarized in Table 18-19.

Table 18-19 Core PMU Comparison

Box	Sandy Bridge	Westmere	Comment
# of Fixed counters per thread	3	3	Use CPUID to enumerate # of counters.
# of general-purpose counters per core	8	8	
Counter width (R,W)	R:48 , W: 32/48	R:48, W:32	See Section 18.2.2.3.

Table 18-19 Core PMU Comparison

Box	Sandy Bridge	Westmere	Comment
# of programmable counters per thread	4 or (8 if a core not shared by two threads)	4	Use CPUID to enumerate # of counters.
Precise Event Based Sampling (PEBS) Events	See Table 18-21	See Table 18-10	IA32_PMC4-IA32_PMC7 do not support PEBS.
PEBS-Load Latency	See Section 18.8.4.2; Data source encoding, STLB miss encoding, Lock transaction encoding	Data source encoding	
PEBS-Precise Store	Section 18.8.4.3	No	
PEBS-PDIR	yes (using precise INST_RETIRED.ALL)	No	
Off-core Response Event	MSR 1A6H and 1A7H; Extended request and response types	MSR 1A6H and 1A7H, limited response types	Nehalem supports 1A6H only.

...

18.8.4 PEBS Support in Intel® microarchitecture code name Sandy Bridge

Processors based on Intel microarchitecture code name Sandy Bridge support PEBS, similar to those offered in prior generation, with several enhanced features. The key components and differences of PEBS facility relative to Intel microarchitecture code name Westmere is summarized in Table 18-20.

Table 18-20 PEBS Facility Comparison

Box	Sandy Bridge	Westmere	Comment
Valid IA32_PMCx	PMC0-PMC3	PMC0-PMC3	No PEBS on PMC4-PMC7
PEBS Buffer Programming	Section 18.6.1.1	Section 18.6.1.1	Unchanged
IA32_PEBS_ENABLE Layout	Figure 18-28	Figure 18-14	
PEBS record layout	Physical Layout same as Table 18-12	Table 18-12	Enhanced fields at offsets 98H, A0H, A8H
PEBS Events	See Table 18-21	See Table 18-10	IA32_PMC4-IA32_PMC7 do not support PEBS.
PEBS-Load Latency	See Table 18-22	Table 18-13	
PEBS-Precise Store	yes; see Section 18.8.4.3	No	IA32_PMC3 only
PEBS-PDIR	yes	No	IA32_PMC1 only

Table 18-20 PEBS Facility Comparison

Box	Sandy Bridge	Westmere	Comment
SAMPLING Restriction	Small SAV(CountDown) value incur higher overhead than prior generation.		

Only IA32_PMC0 through IA32_PMC3 support PEBS.

NOTE

PEBS events are only valid when the following fields of IA32_PERFEVTSELx are all zero: AnyThread, Edge, Invert, CMask.

In IA32_PEBS_ENABLE MSR, bit 63 is defined as PS_ENABLE: When set, this enables IA32_PMC3 to capture precise store information. Only IA32_PMC3 supports the precise store facility. In typical usage of PEBS, the bit fields in IA32_PEBS_ENABLE are written to when the agent software starts PEBS operation; the enabled bit fields should be modified only when re-programming another PEBS event or cleared when the agent uses the performance counters for non-PEBS operations.

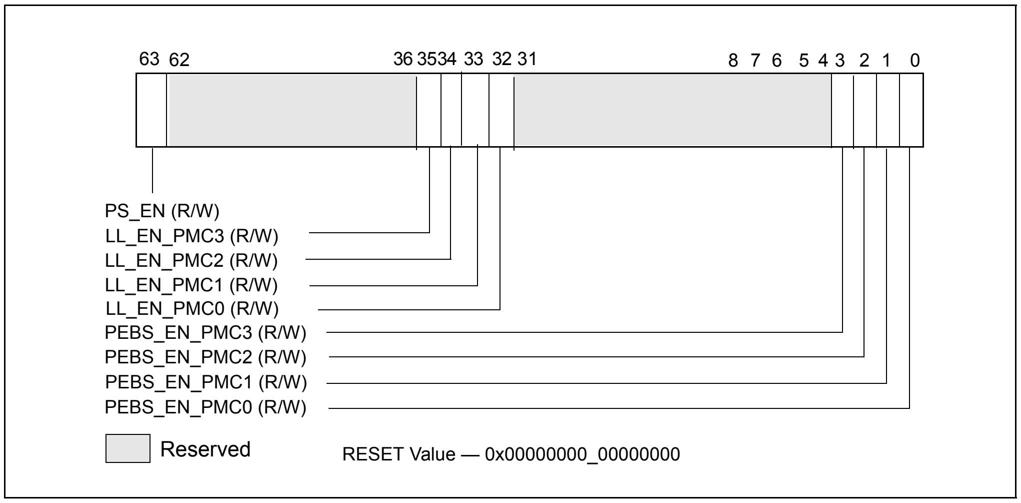
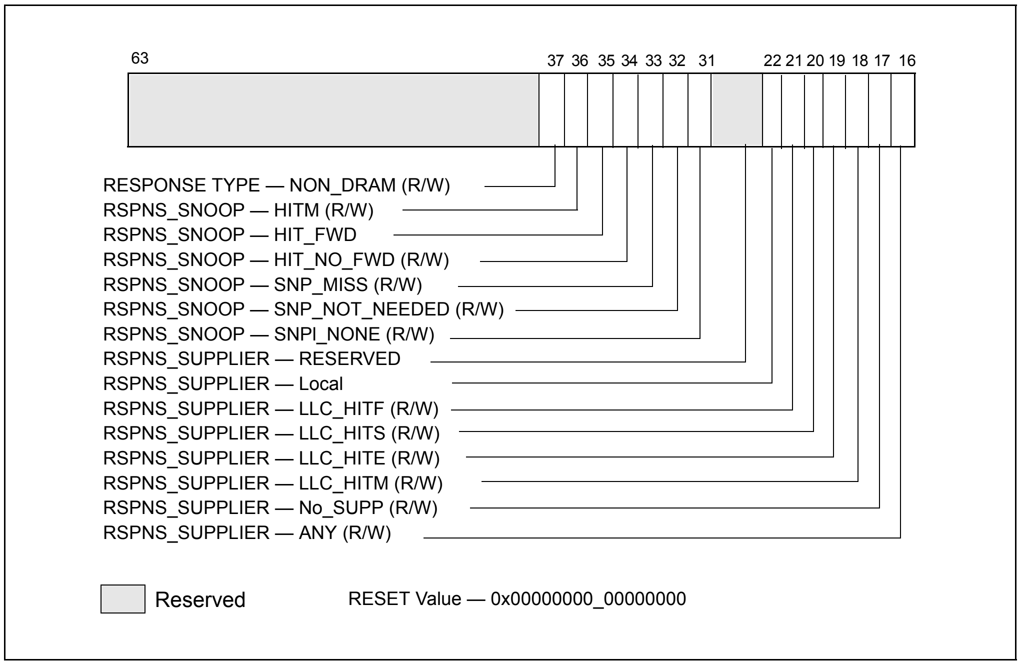


Figure 18-28 Layout of IA32_PEBS_ENABLE MSR

...



63
37 36 35 34 33 32 31
22 21 20 19 18 17 16
RESPONSE TYPE — NON_DRAM (R/W)
RSPNS_SNOOP — HITM (R/W)
RSPNS_SNOOP — HIT_FWD
RSPNS_SNOOP — HIT_NO_FWD (R/W)
RSPNS_SNOOP — SNP_MISS (R/W)
RSPNS_SNOOP — SNP_NOT_NEEDED (R/W)
RSPNS_SNOOP — SNPI_NONE (R/W)
RSPNS_SUPPLIER — RESERVED
RSPNS_SUPPLIER — Local
RSPNS_SUPPLIER — LLC_HITF (R/W)
RSPNS_SUPPLIER — LLC_HITS (R/W)
RSPNS_SUPPLIER — LLC_HITE (R/W)
RSPNS_SUPPLIER — LLC_HITM (R/W)
RSPNS_SUPPLIER — No_SUPP (R/W)
RSPNS_SUPPLIER — ANY (R/W)
Reserved
RESET Value — 0x00000000_00000000

Figure 18-30 Response_Supplier and Snoop Info Fields for MSR_OFFCORE_RSP_x

To properly program this extra register, software must set at least one request type bit and a valid response type pattern. Otherwise, the event count reported will be zero. It is permissible and useful to set multiple request and response type bits in order to obtain various classes of off-core response events. Although MSR_OFFCORE_RSP_x allow an agent software to program numerous combinations that meet the above guideline, not all combinations produce meaningful data.

Table 18-26 MSR_OFFCORE_RSP_x Response Supplier Info Field Definition

Subtype	Bit Name	Offset	Description
Common	Any	16	(R/W). Catch all value for any response types.
Supplier Info	NO_SUPP	17	(R/W). No Supplier Information available
	LLC_HITM	18	(R/W). M-state initial lookup stat in L3.
	LLC_HITE	19	(R/W). E-state
	LLC_HITS	20	(R/W). S-state
	LLC_HITF	21	(R/W). F-state
	LOCAL	22	(R/W). Local DRAM Controller
	Reserved	30:23	Reserved

To specify a complete offcore response filter, software must properly program bits in the request and response type fields. A valid request type must have at least one bit set in the non-reserved bits of 15:0. A valid response type must be a non-zero value of the following expression:

ANY | [('OR' of Supplier Info Bits) & ('OR' of Snoop Info Bits)]

If "ANY" bit is set, the supplier and snoop info bits are ignored.

Table 18-27 MSR_OFFCORE_RSP_x Snoop Info Field Definition

Subtype	Bit Name	Offset	Description
Snoop Info	SNP_NONE	31	(R/W). No details on snoop-related information
	SNP_NOT_NEEDED	32	(R/W). No snoop was needed to satisfy the request.
	SNP_MISS	33	(R/W). A snoop was needed and it missed all snooped caches: -For LLC Hit, ReslHitl was returned by all cores -For LLC Miss, Rspl was returned by all sockets and data was returned from DRAM.
	SNP_NO_FWD	34	(R/W). A snoop was needed and it hits in at least one snooped cache. Hit denotes a cache-line was valid before snoop effect. This includes: -Snoop Hit w/ Invalidation (LLC Hit, RFO) -Snoop Hit, Left Shared (LLC Hit/Miss, IFetch/Data_RD) -Snoop Hit w/ Invalidation and No Forward (LLC Miss, RFO Hit S) In the LLC Miss case, data is returned from DRAM.
	SNP_FWD	35	(R/W). A snoop was needed and data was forwarded from a remote socket. This includes: -Snoop Forward Clean, Left Shared (LLC Hit/Miss, IFetch/Data_RD/RFT).
	HITM	36	(R/W). A snoop was needed and it HitM-ed in local or remote cache. HitM denotes a cache-line was in modified state before effect as a results of snoop. This includes: -Snoop HitM w/ WB (LLC miss, IFetch/Data_RD) -Snoop Forward Modified w/ Invalidation (LLC Hit/Miss, RFO) -Snoop MtoS (LLC Hit, IFetch/Data_RD).
	NON_DRAM	37	(R/W). Target was non-DRAM system address. This includes MMIO transactions.

...

18.8.7 Next Generation Intel Xeon Processor Performance Monitoring Facility

The Next Generation Intel Xeon processor is based on Intel microarchitecture code name Sandy Bridge. The performance monitoring facilities in the processor core generally are the same as those described in Section 18.8 through Section 18.8.5. However, the MSR_OFFCORE_RSP_0/MSR_OFFCORE_RSP_1 Response Supplier Info field shown in Table 18-26 applies to Intel Core Processors with CPUID signature of DisplayFamily_DisplayModel encoding of 06_2AH; next generation Intel Xeon processor with CPUID signature of DisplayFamily_DisplayModel encoding of 06_2DH supports an additional field for remote DRAM controller shown in Table 18-29. Additionally, the are some small differences in the non-architectural performance monitoring events. See (Table 19-4).

Table 18-29 MSR_OFFCORE_RSP_x Supplier Info Field Definition for Next Generation Intel Xeon Processor

Subtype	Bit Name	Offset	Description
Common	Any	16	(R/W). Catch all value for any response types.
Supplier Info	NO_SUPP	17	(R/W). No Supplier Information available
	LLC_HITM	18	(R/W). M-state initial lookup stat in L3.
	LLC_HITE	19	(R/W). E-state
	LLC_HITS	20	(R/W). S-state
	LLC_HITF	21	(R/W). F-state
	LOCAL	22	(R/W). Local DRAM Controller
	Remote	30:23	(R/W): Remote DRAM Controller (either all 0s or all 1s)

18.9 NEXT GENERATION INTEL CORE PROCESSOR PERFORMANCE MONITORING FACILITY

The Next Generation Intel Core processor is based on Intel microarchitecture code name Ivy Bridge. The performance monitoring facilities in the processor core generally are the same as those described in Section 18.8 through Section 18.8.5. The non-architectural performance monitoring events supported by the processor core are listed in Table 19-4.

...

13. Updates to Chapter 19, Volume 3B

Change bars show changes to Chapter 19 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B:* System Programming Guide, Part 2.

...

This chapter lists the performance-monitoring events that can be monitored with the Intel 64 or IA-32 processors. The ability to monitor performance events and the events that can be monitored in these processors are mostly model-specific, except for architectural performance events, described in Section 19.1.

Non-architectural performance events (i.e. model-specific events) are listed for each generation of microarchitecture:

- Section 19.2 - Processors based on Intel® microarchitecture code name Ivy Bridge
- Section 19.3 - Processors based on Intel® microarchitecture code name Sandy Bridge
- Section 19.4 - Processors based on Intel® microarchitecture code name Nehalem
- Section 19.5 - Processors based on Intel® microarchitecture code name Westmere

- Section 19.6 - Processors based on Enhanced Intel® Core™ microarchitecture
- Section 19.7 - Processors based on Intel® Core™ microarchitecture
- Section 19.8 - Processors based on Intel® Atom™ microarchitecture
- Section 19.9 - Intel® Core™ Solo and Intel® Core™ Duo processors
- Section 19.10 - Processors based on Intel NetBurst® microarchitecture
- Section 19.11 - Pentium® M family processors
- Section 19.12 - P6 family processors
- Section 19.13 - Pentium® processors

NOTE

These performance-monitoring events are intended to be used as guides for performance tuning. The counter values reported by the performance-monitoring events are approximate and believed to be useful as relative guides for tuning software. Known discrepancies are documented where applicable.

...

19.2 PERFORMANCE MONITORING EVENTS FOR NEXT GENERATION INTEL® CORE™ PROCESSORS

Next generation Intel® Core™ Processors are based on the Intel microarchitecture code name Ivy Bridge. They support architectural performance-monitoring events listed in Table 19-1. Non-architectural performance-monitoring events in the processor core are listed in Table 19-2. The events in Table 19-2 apply to processors with CPUID signature of DisplayFamily_DisplayModel encoding with the following values: 06_3AH.

Table 19-2 Non-Architectural Performance Events In the Processor Core of Next Generation Intel Core i7, i5, i3 Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
03H	02H	LD_BLOCKS.STORE_FORWARD	loads blocked by overlapping with store buffer that cannot be forwarded .	
05H	01H	MISALIGN_MEM_REF.LOADS	Speculative cache-line split load uops dispatched to L1D.	
05H	02H	MISALIGN_MEM_REF.STORES	Speculative cache-line split Store-address uops dispatched to L1D.	
07H	01H	LD_BLOCKS_PARTIAL.ADDRESS_ALIAS	False dependencies in MOB due to partial compare on address.	
08H	81H	DTLB_LOAD_MISSES.DEMAND_LD_MISS_CAUSES_A_WALK	Misses in all TLB levels that cause a page walk of any page size from demand loads.	
08H	82H	DTLB_LOAD_MISSES.DEMAND_LD_WALK_COMPLETED	Misses in all TLB levels that caused page walk completed of any size by demand loads.	

Table 19-2 Non-Architectural Performance Events In the Processor Core of Next Generation Intel Core i7, i5, i3 Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
08H	84H	DTLB_LOAD_MISSES. DEMAND_LD_WALK_ DURATION	Cycle PMH is busy with a walk due to demand loads.	
0EH	01H	UOPS_ISSUED.ANY	Increments each cycle the # of Uops issued by the RAT to RS. Set Cmask = 1, Inv = 1, Any= 1to count stalled cycles of this core.	Set Cmask = 1, Inv = 1to count stalled cycles
14H	01H	ARITH.FPU_DIV_ACT IVE	Cycles that the divider is active, includes INT and FP. Set 'edge =1, cmask=1' to count the number of divides.	
24H	01H	L2_RQSTS.DEMAND_ DATA_RD_HIT	Demand Data Read requests that hit L2 cache	
24H	03H	L2_RQSTS.ALL_DEM AND_DATA_RD	Counts any demand and L1 HW prefetch data load requests to L2.	
24H	04H	L2_RQSTS.RFO_HITS	Counts the number of store RFO requests that hit the L2 cache.	
24H	08H	L2_RQSTS.RFO_MISS	Counts the number of store RFO requests that miss the L2 cache.	
24H	0CH	L2_RQSTS.ALL_RFO	Counts all L2 store RFO requests.	
24H	10H	L2_RQSTS.CODE_RD _HIT	Number of instruction fetches that hit the L2 cache.	
24H	20H	L2_RQSTS.CODE_RD _MISS	Number of instruction fetches that missed the L2 cache.	
24H	30H	L2_RQSTS.ALL_COD E_RD	Counts all L2 code requests.	
27H	01H	L2_STORE_LOCK_RQ STS.MISS	RFOs that miss cache lines	
27H	08H	L2_STORE_LOCK_RQ STS.HIT_M	RFOs that hit cache lines in M state	
27H	0FH	L2_STORE_LOCK_RQ STS.ALL	RFOs that access cache lines in any state	
28H	01H	L2_L1D_WB_RQSTS. MISS	Not rejected writebacks that missed LLC.	
28H	04H	L2_L1D_WB_RQSTS. HIT_E	Not rejected writebacks from L1D to L2 cache lines in E state.	
28H	08H	L2_L1D_WB_RQSTS. HIT_M	Not rejected writebacks from L1D to L2 cache lines in M state.	
2EH	4FH	LONGEST_LAT_CACH E.REFERENCE	This event counts requests originating from the core that reference a cache line in the last level cache.	see Table 19-1

Table 19-2 Non-Architectural Performance Events In the Processor Core of Next Generation Intel Core i7, i5, i3 Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
2EH	41H	LONGEST_LAT_CACHE.MISS	This event counts each cache miss condition for references to the last level cache.	see Table 19-1
3CH	00H	CPU_CLK_UNHALTED.THREAD_P	Counts the number of thread cycles while the thread is not in a halt state. The thread enters the halt state when it is running the HLT instruction. The core frequency may change from time to time due to power or thermal throttling.	see Table 19-1
3CH	01H	CPU_CLK_THREAD_UNHALTED.REF_XCLK	Increments at the frequency of XCLK (100 MHz) when not halted.	see Table 19-1
48H	01H	L1D_PEND_MISS.PENDING	Increments the number of outstanding L1D misses every cycle. Set Cmaks = 1 and Edge =1 to count occurrences.	Counter 2 only; Set Cmask = 1 to count cycles.
49H	01H	DTLB_STORE_MISSES.MISS_CAUSES_A_WALK	Miss in all TLB levels causes an page walk of any page size (4K/2M/4M/1G).	
49H	02H	DTLB_STORE_MISSES.WALK_COMPLETED	Miss in all TLB levels causes a page walk that completes of any page size (4K/2M/4M/1G).	
49H	04H	DTLB_STORE_MISSES.WALK_DURATION	Cycles PMH is busy with this walk.	
49H	10H	DTLB_STORE_MISSES.STLB_HIT	Store operations that miss the first TLB level but hit the second and do not cause page walks	
4CH	01H	LOAD_HIT_PRE.SW_PF	Not SW-prefetch load dispatches that hit fill buffer allocated for S/W prefetch.	
51H	01H	L1D.REPLACEMENT	Counts the number of lines brought into the L1 data cache.	
58H	01H	MOVE_ELIMINATION.INT_NOT_ELIMINATED	Number of integer Move Elimination candidate uops that were not eliminated.	
58H	02H	MOVE_ELIMINATION.SIMD_NOT_ELIMINATED	Number of SIMD Move Elimination candidate uops that were not eliminated.	
5CH	01H	CPL_CYCLES.RING0	Unhalted core cycles when the thread is in ring 0	Use Edge to count transition
5CH	02H	CPL_CYCLES.RING123	Unhalted core cycles when the thread is not in ring 0	
5EH	01H	RS_EVENTS.EMPTY_CYCLES	Cycles the RS is empty for the thread.	

Table 19-2 Non-Architectural Performance Events In the Processor Core of Next Generation Intel Core i7, i5, i3 Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
5FH	01H	TLB_ACCESS.LOAD_STLB_HIT	Counts load operations that missed 1st level DTLB but hit the 2nd level.	
60H	01H	OFFCORE_REQUESTS_OUTSTANDING.DEMAND_DATA_RD	Offcore outstanding Demand Data Read transactions in SQ to uncore. Set Cmask=1 to count cycles.	
60H	04H	OFFCORE_REQUESTS_OUTSTANDING.DEMAND_RFO	Offcore outstanding RFO store transactions in SQ to uncore. Set Cmask=1 to count cycles.	
60H	08H	OFFCORE_REQUESTS_OUTSTANDING.ALL_DATA_RD	Offcore outstanding cacheable data read transactions in SQ to uncore. Set Cmask=1 to count cycles.	
63H	01H	LOCK_CYCLES.SPLIT_LOCK_UC_LOCK_DURATION	Cycles in which the L1D and L2 are locked, due to a UC lock or split lock.	
63H	02H	LOCK_CYCLES.CACHE_LOCK_DURATION	Cycles in which the L1D is locked.	
79H	02H	IDQ.EMPTY	Counts cycles the IDQ is empty.	
79H	04H	IDQ.MITE_UOPS	Increment each cycle # of uops delivered to IDQ from MITE path. Set Cmask = 1 to count cycles.	Can combine Umask 04H and 20H
79H	08H	IDQ.DSB_UOPS	Increment each cycle. # of uops delivered to IDQ from DSB path. Set Cmask = 1 to count cycles.	Can combine Umask 08H and 10H
79H	30H	IDQ.MS_UOPS	Increment each cycle # of uops delivered to IDQ from MS by either DSB or MITE. Set Cmask = 1 to count cycles.	Can combine Umask 04H, 08H and 30H
80H	02H	ICACHE.MISSES	Number of Instruction Cache, Streaming Buffer and Victim Cache Misses. Includes UC accesses.	
85H	01H	ITLB_MISSES.MISS_CAUSES_A_WALK	Misses in all ITLB levels that cause page walks	
85H	02H	ITLB_MISSES.WALK_COMPLETED	Misses in all ITLB levels that cause completed page walks	
85H	04H	ITLB_MISSES.WALK_DURATION	Cycle PMH is busy with a walk.	
85H	10H	ITLB_MISSES.STLB_HIT	Number of cache load STLB hits. No page walk.	
87H	01H	ILD_STALL.LCP	Stalls caused by changing prefix length of the instruction.	
87H	04H	ILD_STALL.IQ_FULL	Stall cycles due to IQ is full.	

Table 19-2 Non-Architectural Performance Events In the Processor Core of Next Generation Intel Core i7, i5, i3 Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
88H	01H	BR_INST_EXEC.COND	Qualify conditional near branch instructions executed, but not necessarily retired.	Must combine with umask 40H, 80H
88H	02H	BR_INST_EXEC.DIRECT_JMP	Qualify all unconditional near branch instructions excluding calls and indirect branches.	Must combine with umask 80H
88H	04H	BR_INST_EXEC.INDIRECT_JMP_NON_CALL_RET	Qualify executed indirect near branch instructions that are not calls nor returns.	Must combine with umask 80H
88H	08H	BR_INST_EXEC.RETURN_NEAR	Qualify indirect near branches that have a return mnemonic.	Must combine with umask 80H
88H	10H	BR_INST_EXEC.DIRECT_NEAR_CALL	Qualify unconditional near call branch instructions, excluding non call branch, executed.	Must combine with umask 80H
88H	20H	BR_INST_EXEC.INDIRECT_NEAR_CALL	Qualify indirect near calls, including both register and memory indirect, executed.	Must combine with umask 80H
88H	40H	BR_INST_EXEC.NONTAKEN	Qualify non-taken near branches executed.	Applicable to umask 01H only
88H	80H	BR_INST_EXEC.TAKEN	Qualify taken near branches executed. Must combine with 01H,02H, 04H, 08H, 10H, 20H	
88H	FFH	BR_INST_EXEC.ALL_BRANCHES	Counts all near executed branches (not necessarily retired).	
89H	01H	BR_MISP_EXEC.COND	Qualify conditional near branch instructions mispredicted.	Must combine with umask 40H, 80H
89H	04H	BR_MISP_EXEC.INDIRECT_JMP_NON_CALL_RET	Qualify mispredicted indirect near branch instructions that are not calls nor returns.	Must combine with umask 80H
89H	08H	BR_MISP_EXEC.RETURN_NEAR	Qualify mispredicted indirect near branches that have a return mnemonic.	Must combine with umask 80H
89H	10H	BR_MISP_EXEC.DIRECT_NEAR_CALL	Qualify mispredicted unconditional near call branch instructions, excluding non call branch, executed.	Must combine with umask 80H
89H	20H	BR_MISP_EXEC.INDIRECT_NEAR_CALL	Qualify mispredicted indirect near calls, including both register and memory indirect, executed.	Must combine with umask 80H
89H	40H	BR_MISP_EXEC.NONTAKEN	Qualify mispredicted non-taken near branches executed,.	Applicable to umask 01H only
89H	80H	BR_MISP_EXEC.TAKEN	Qualify mispredicted taken near branches executed. Must combine with 01H,02H, 04H, 08H, 10H, 20H	

Table 19-2 Non-Architectural Performance Events In the Processor Core of Next Generation Intel Core i7, i5, i3 Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
89H	FFH	BR_MISP_EXEC.ALL_BRANCHES	Counts all near executed branches (not necessarily retired).	
9CH	01H	IDQ_UOPS_NOT_DELIVERED.CORE	Count number of non-delivered uops to RAT per thread.	Use Cmask to qualify uop b/w
A1H	01H	UOPS_DISPATCHED_PORT.PORT_0	Cycles which a Uop is dispatched on port 0.	
A1H	02H	UOPS_DISPATCHED_PORT.PORT_1	Cycles which a Uop is dispatched on port 1.	
A1H	04H	UOPS_DISPATCHED_PORT.PORT_2_LD	Cycles which a load uop is dispatched on port 2.	
A1H	08H	UOPS_DISPATCHED_PORT.PORT_2_STA	Cycles which a store address uop is dispatched on port 2.	
A1H	0CH	UOPS_DISPATCHED_PORT.PORT_2	Cycles which a Uop is dispatched on port 2.	
A1H	10H	UOPS_DISPATCHED_PORT.PORT_3_LD	Cycles which a load uop is dispatched on port 3.	
A1H	20H	UOPS_DISPATCHED_PORT.PORT_3_STA	Cycles which a store address uop is dispatched on port 3.	
A1H	30H	UOPS_DISPATCHED_PORT.PORT_3	Cycles which a Uop is dispatched on port 3.	
A1H	40H	UOPS_DISPATCHED_PORT.PORT_4	Cycles which a Uop is dispatched on port 4.	
A1H	80H	UOPS_DISPATCHED_PORT.PORT_5	Cycles which a Uop is dispatched on port 5.	
A2H	01H	RESOURCE_STALLS.ANY	Cycles Allocation is stalled due to Resource Related reason.	
A2H	04H	RESOURCE_STALLS.RS	Cycles stalled due to no eligible RS entry available.	
A2H	08H	RESOURCE_STALLS.SB	Cycles stalled due to no store buffers available. (not including draining form sync).	
A2H	10H	RESOURCE_STALLS.ROB	Cycles stalled due to re-order buffer full.	
ABH	01H	DSB2MITE_SWITCHES.COUNT	Number of DSB to MITE switches.	
ABH	02H	DSB2MITE_SWITCHES.PENALTY_CYCLES	Cycles DSB to MITE switches caused delay.	
ACH	08H	DSB_FILL.EXCEED_DSB_LINES	DSB Fill encountered > 3 DSB lines.	
AEH	01H	ITLB.ITLB_FLUSH	Counts the number of ITLB flushes, includes 4k/2M/4M pages.	

Table 19-2 Non-Architectural Performance Events In the Processor Core of Next Generation Intel Core i7, i5, i3 Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
B0H	01H	OFFCORE_REQUEST S.DEMAND_DATA_RD	Demand data read requests sent to uncore.	
B0H	02H	OFFCORE_REQUEST S.DEMAND_CODE_RD	Demand code read requests sent to uncore.	
B0H	04H	OFFCORE_REQUEST S.DEMAND_RFO	Demand RFO read requests sent to uncore., including regular RFOs, locks, ItoM	
B0H	08H	OFFCORE_REQUEST S.ALL_DATA_RD	Data read requests sent to uncore (demand and prefetch).	
B1H	01H	UOPS_DISPATCHED.T HREAD	Counts total number of uops to be dispatched per-thread each cycle. Set Cmask = 1, INV =1 to count stall cycles.	
B1H	02H	UOPS_DISPATCHED.C ORE	Counts total number of uops to be dispatched per-core each cycle.	Do not need to set ANY
B7H	01H	OFF_CORE_RESPONS E_0	see Section 18.8.5, “Off-core Response Performance Monitoring”; PMC0 only.	Requires programming MSR 01A6H
BBH	01H	OFF_CORE_RESPONS E_1	See Section 18.8.5, “Off-core Response Performance Monitoring”. PMC3 only.	Requires programming MSR 01A7H
C0H	00H	INST_RETIRED.ANY_ P	Number of instructions at retirement	See Table 19-1
C0H	01H	INST_RETIRED.PREC _DIST	Precise instruction retired event with HW to reduce effect of PEBS shadow in IP distribution	PMC1 only; Must quiesce other PMCs.
C1H	08H	OTHER_ASSISTS.AVX _STORE	Number of assists associated with 256-bit AVX store operations.	
C1H	10H	OTHER_ASSISTS.AVX _TO_SSE	Number of transitions from AVX-256 to legacy SSE when penalty applicable.	
C1H	20H	OTHER_ASSISTS.SSE _TO_AVX	Number of transitions from SSE to AVX-256 when penalty applicable.	
C2H	01H	UOPS_RETIRED.ALL	Counts the number of micro-ops retired, Use cmask=1 and invert to count active cycles or stalled cycles.	Supports PEBS
C2H	02H	UOPS_RETIRED.RETI RE_SLOTS	Counts the number of retirement slots used each cycle.	
C3H	02H	MACHINE_CLEARS.M EMORY_ORDERING	Counts the number of machine clears due to memory order conflicts.	

Table 19-2 Non-Architectural Performance Events In the Processor Core of Next Generation Intel Core i7, i5, i3 Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
C3H	20H	MACHINE_CLEARS.MASKMOV	Counts the number of executed AVX masked load operations that refer to an illegal address range with the mask bits set to 0.	
C4H	00H	BR_INST_RETIRED.ALL_BRANCHES	Branch instructions at retirement	See Table 19-1
C4H	01H	BR_INST_RETIRED.CONDITIONAL	Counts the number of conditional branch instructions retired.	Supports PEBS
C4H	02H	BR_INST_RETIRED.NEAR_CALL	Direct and indirect near call instructions retired.	
C4H	04H	BR_INST_RETIRED.ALL_BRANCHES	Counts the number of branch instructions retired.	
C4H	08H	BR_INST_RETIRED.NEAR_RETURN	Counts the number of near return instructions retired.	
C4H	10H	BR_INST_RETIRED.NOT_TAKEN	Counts the number of not taken branch instructions retired.	
C4H	20H	BR_INST_RETIRED.NEAR_TAKEN	Number of near taken branches retired.	
C4H	40H	BR_INST_RETIRED.FAR_BRANCH	Number of far branches retired.	
C5H	00H	BR_MISP_RETIRED.ALL_BRANCHES	Mispredicted branch instructions at retirement	See Table 19-1
C5H	01H	BR_MISP_RETIRED.CONDITIONAL	Mispredicted conditional branch instructions retired.	Supports PEBS
C5H	02H	BR_MISP_RETIRED.NEAR_CALL	Direct and indirect mispredicted near call instructions retired.	
C5H	04H	BR_MISP_RETIRED.ALL_BRANCHES	Mispredicted macro branch instructions retired.	
C5H	10H	BR_MISP_RETIRED.NOT_TAKEN	Mispredicted not taken branch instructions retired.	
C5H	20H	BR_MISP_RETIRED.TAKEN	Mispredicted taken branch instructions retired.	
CAH	08H	FP_ASSIST.SIMD_OUTPUT	Number of SIMD FP assists due to Output values	
CAH	10H	FP_ASSIST.SIMD_INPUT	Number of SIMD FP assists due to input values	
CAH	1EH	FP_ASSIST.ANY	Cycles with any input/output SSE* or FP assists	
CCH	20H	ROB_MISC_EVENTS.LBR_INSERTS	Count cases of saving new LBR records by hardware.	
CDH	01H	MEM_TRANS_RETIRED.LOAD_LATENCY	Sample loads with specified latency threshold. PMC3 only.	Specify threshold in MSR 0x3F6

Table 19-2 Non-Architectural Performance Events In the Processor Core of Next Generation Intel Core i7, i5, i3 Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
CDH	02H	MEM_TRANS_RETIRED.PRECISE_STORE	Sample stores and collect precise store operation via PEBS record. PMC3 only.	See Section 18.8.4.3
D0H	01H	MEM_UOP_RETIRED.LOADS	Qualify retired memory uops that are loads. Combine with umask 10H, 20H, 40H, 80H.	Supports PEBS
D0H	02H	MEM_UOP_RETIRED.STORES	Qualify retired memory uops that are stores. Combine with umask 10H, 20H, 40H, 80H.	
D0H	10H	MEM_UOP_RETIRED.STLB_MISS	Qualify retired memory uops with STLB miss. Must combine with umask 01H, 02H, to produce counts.	
D0H	20H	MEM_UOP_RETIRED.LOCK	Qualify retired memory uops with lock. Must combine with umask 01H, 02H, to produce counts.	
D0H	40H	MEM_UOP_RETIRED.SPLIT	Qualify retired memory uops with line split. Must combine with umask 01H, 02H, to produce counts.	
D0H	80H	MEM_UOP_RETIRED.ALL	Qualify any retired memory uops. Must combine with umask 01H, 02H, to produce counts.	
D1H	01H	MEM_LOAD_UOPS_RETIRED.L1_HIT	Retired load uops with L1 cache hits as data sources.	Supports PEBS
D1H	02H	MEM_LOAD_UOPS_RETIRED.L2_HIT	Retired load uops with L2 cache hits as data sources.	
D1H	04H	MEM_LOAD_UOPS_RETIRED.LLC_HIT	Retired load uops with LLC cache hits as data sources.	
D1H	40H	MEM_LOAD_UOPS_RETIRED.HIT_LFB	Retired load uops which data sources were load uops missed L1 but hit FB due to preceding miss to the same cache line with data not ready.	
D2H	02H	MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_HIT	Retired load uops which data sources were LLC and cross-core snoop hits in on-pkg core cache.	Supports PEBS
D2H	04H	MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_HITM	Retired load uops which data sources were HitM responses from shared LLC.	
D2H	08H	MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_NONE	Retired load uops which data sources were hits in LLC without snoops required.	
D3H	01H	MEM_LOAD_UOPS_LLC_MISS_RETIRED.LOCAL_DRAM	Retired load uops which data sources missed LLC but serviced from local dram.	Supports PEBS.

Table 19-2 Non-Architectural Performance Events In the Processor Core of Next Generation Intel Core i7, i5, i3 Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
F0H	01H	L2_TRANS.DEMAND_DATA_RD	Demand Data Read requests that access L2 cache	
F0H	02H	L2_TRANS.RFO	RFO requests that access L2 cache	
F0H	04H	L2_TRANS.CODE_RD	L2 cache accesses when fetching instructions	
F0H	10H	L2_TRANS.L1D_WB	L1D writebacks that access L2 cache	
F0H	20H	L2_TRANS.L2_FILL	L2 fill requests that access L2 cache	
F0H	40H	L2_TRANS.L2_WB	L2 writebacks that access L2 cache	
F0H	80H	L2_TRANS.ALL_REQUESTS	Transactions accessing L2 pipe	
F1H	01H	L2_LINES_IN.I	L2 cache lines in I state filling L2	Counting does not cover rejects.
F1H	02H	L2_LINES_IN.S	L2 cache lines in S state filling L2	Counting does not cover rejects.
F1H	04H	L2_LINES_IN.E	L2 cache lines in E state filling L2	Counting does not cover rejects.
F1H	07H	L2_LINES_IN.ALL	L2 cache lines filling L2	Counting does not cover rejects.
F2H	01H	L2_LINES_OUT.DEMAND_CLEAN	Clean L2 cache lines evicted by demand	
F2H	02H	L2_LINES_OUT.DEMAND_DIRTY	Dirty L2 cache lines evicted by demand	
F2H	0AH	L2_LINES_OUT.DIRTY_ALL	Dirty L2 cache lines filling the L2	Counting does not cover rejects.

19.3 PERFORMANCE MONITORING EVENTS FOR INTEL® CORE™ PROCESSOR 2XXX SERIES

Second generation Intel® Core™ Processor 2xxx Series are based on the Intel microarchitecture code name Sandy Bridge. They support architectural performance-monitoring events listed in Table 19-1. Non-architectural performance-monitoring events in the processor core are listed in Table 19-3, Table 19-4, and Table 19-5. The events in Table 19-3 apply to processors with CPUID signature of DisplayFamily_DisplayModel encoding with the following values: 06_2AH and 06_2DH. The events in Table 19-4 apply to processors with CPUID signature 06_2AH. The events in Table 19-5 apply to processors with CPUID signature 06_2DH.

Table 19-3 Non-Architectural Performance Events In the Processor Core common to Intel Core i7, i5, i3 Processors 2xxx Series and Next Generation Intel Xeon Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
03H	01H	LD_BLOCKS.DATA_UNKNOWN	blocked loads due to store buffer blocks with unknown data.	
03H	02H	LD_BLOCKS.STORE_FORWARD	loads blocked by overlapping with store buffer that cannot be forwarded .	
03H	08H	LD_BLOCKS.NO_SR	# of Split loads blocked due to resource not available.	
03H	10H	LD_BLOCKS.ALL_BLOCK	Number of cases where any load is blocked but has no DCU miss.	
05H	01H	MISALIGN_MEM_REF.LOADS	Speculative cache-line split load uops dispatched to L1D.	
05H	02H	MISALIGN_MEM_REF.STORES	Speculative cache-line split Store-address uops dispatched to L1D.	
07H	01H	LD_BLOCKS_PARTIAL.ADDRESS_ALIAS	False dependencies in MOB due to partial compare on address.	
07H	08H	LD_BLOCKS_PARTIAL.ALL_STA_BLOCK	The number of times that load operations are temporarily blocked because of older stores, with addresses that are not yet known. A load operation may incur more than one block of this type.	
08H	01H	DTLB_LOAD_MISSES.MISS_CAUSES_A_WALK	Misses in all TLB levels that cause a page walk of any page size.	
08H	02H	DTLB_LOAD_MISSES.WALK_COMPLETED	Misses in all TLB levels that caused page walk completed of any size.	
08H	04H	DTLB_LOAD_MISSES.WALK_DURATION	Cycle PMH is busy with a walk.	
08H	10H	DTLB_LOAD_MISSES.STLB_HIT	Number of cache load STLB hits. No page walk.	
0DH	03H	INT_MISC.RECOVERY_CYCLES	Cycles waiting to recover after Machine Clears or JEClear. Set Cmask= 1.	Set Edge to count occurrences
0DH	40H	INT_MISC.RAT_STALL_CYCLES	Cycles RAT external stall is sent to IDQ for this thread.	
0EH	01H	UOPS_ISSUED.ANY	Increments each cycle the # of Uops issued by the RAT to RS. Set Cmask = 1, Inv = 1, Any= 1to count stalled cycles of this core.	Set Cmask = 1, Inv = 1to count stalled cycles
10H	01H	FP_COMP_OPS_EXE.X87	Counts number of X87 uops executed.	

Table 19-3 Non-Architectural Performance Events In the Processor Core common to Intel Core i7, i5, i3 Processors 2xxx Series and Next Generation Intel Xeon Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
10H	10H	FP_COMP_OPS_EXE.SSE_FP_PACKED_DOUBLE	Counts number of SSE* double precision FP packed uops executed.	
10H	20H	FP_COMP_OPS_EXE.SSE_FP_SCALAR_SINGLE	Counts number of SSE* single precision FP scalar uops executed.	
10H	40H	FP_COMP_OPS_EXE.SSE_PACKED SINGLE	Counts number of SSE* single precision FP packed uops executed.	
10H	80H	FP_COMP_OPS_EXE.SSE_SCALAR_DOUBLE	Counts number of SSE* double precision FP scalar uops executed.	
11H	01H	SIMD_FP_256.PACKED_SINGLE	Counts 256-bit packed single-precision floating-point instructions	
11H	02H	SIMD_FP_256.PACKED_DOUBLE	Counts 256-bit packed double-precision floating-point instructions	
14H	01H	ARITH.FPU_DIV_ACTIVE	Cycles that the divider is active, includes INT and FP. Set 'edge =1, cmask=1' to count the number of divides.	
17H	01H	INSTS_WRITTEN_TO_IQ.INSTS	Counts the number of instructions written into the IQ every cycle.	
24H	01H	L2_RQSTS.DEMAND_DATA_RD_HIT	Demand Data Read requests that hit L2 cache	
24H	03H	L2_RQSTS.ALL_DEMAND_DATA_RD	Counts any demand and L1 HW prefetch data load requests to L2.	
24H	04H	L2_RQSTS.RFO_HITS	Counts the number of store RFO requests that hit the L2 cache.	
24H	08H	L2_RQSTS.RFO_MISS	Counts the number of store RFO requests that miss the L2 cache.	
24H	0CH	L2_RQSTS.ALL_RFO	Counts all L2 store RFO requests.	
24H	10H	L2_RQSTS.CODE_RD_HIT	Number of instruction fetches that hit the L2 cache.	
24H	20H	L2_RQSTS.CODE_RD_MISS	Number of instruction fetches that missed the L2 cache.	
24H	30H	L2_RQSTS.ALL_CODE_RD	Counts all L2 code requests.	
24H	40H	L2_RQSTS.PF_HIT	Requests from L2 Hardware prefetcher that hit L2.	
24H	80H	L2_RQSTS.PF_MISS	Requests from L2 Hardware prefetcher that missed L2.	
24H	C0H	L2_RQSTS.ALL_PF	Any requests from L2 Hardware prefetchers	

Table 19-3 Non-Architectural Performance Events In the Processor Core common to Intel Core i7, i5, i3 Processors 2xxx Series and Next Generation Intel Xeon Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
27H	01H	L2_STORE_LOCK_RQSTS.MISS	RFOs that miss cache lines	
27H	04H	L2_STORE_LOCK_RQSTS.HIT_E	RFOs that hit cache lines in E state	
27H	08H	L2_STORE_LOCK_RQSTS.HIT_M	RFOs that hit cache lines in M state	
27H	0FH	L2_STORE_LOCK_RQSTS.ALL	RFOs that access cache lines in any state	
28H	04H	L2_L1D_WB_RQSTS.HIT_E	Not rejected writebacks from L1D to L2 cache lines in E state.	
28H	08H	L2_L1D_WB_RQSTS.HIT_M	Not rejected writebacks from L1D to L2 cache lines in M state.	
2EH	4FH	LONGEST_LAT_CACHE.REFERENCE	This event counts requests originating from the core that reference a cache line in the last level cache.	see Table 19-1
2EH	41H	LONGEST_LAT_CACHE.MISS	This event counts each cache miss condition for references to the last level cache.	see Table 19-1
3CH	00H	CPU_CLK_UNHALTED.THREAD_P	Counts the number of thread cycles while the thread is not in a halt state. The thread enters the halt state when it is running the HLT instruction. The core frequency may change from time to time due to power or thermal throttling.	see Table 19-1
3CH	01H	CPU_CLK_THREAD_UNHALTED.REF_XCLK	Increments at the frequency of XCLK (100 MHz) when not halted.	see Table 19-1
48H	01H	L1D_PEND_MISS.PENDING	Increments the number of outstanding L1D misses every cycle. Set Cmaks = 1 and Edge =1 to count occurrences.	Counter 2 only; Set Cmask = 1 to count cycles.
49H	01H	DTLB_STORE_MISSES.MISS_CAUSES_A_WALK	Miss in all TLB levels causes an page walk of any page size (4K/2M/4M/1G).	
49H	02H	DTLB_STORE_MISSES.WALK_COMPLETED	Miss in all TLB levels causes a page walk that completes of any page size (4K/2M/4M/1G).	
49H	04H	DTLB_STORE_MISSES.WALK_DURATION	Cycles PMH is busy with this walk.	
49H	10H	DTLB_STORE_MISSES.STLB_HIT	Store operations that miss the first TLB level but hit the second and do not cause page walks	

Table 19-3 Non-Architectural Performance Events In the Processor Core common to Intel Core i7, i5, i3 Processors 2xxx Series and Next Generation Intel Xeon Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
4CH	01H	LOAD_HIT_PRE.SW_PF	Not SW-prefetch load dispatches that hit fill buffer allocated for S/W prefetch.	
4CH	02H	LOAD_HIT_PRE.HW_PF	Not SW-prefetch load dispatches that hit fill buffer allocated for H/W prefetch.	
4EH	02H	HW_PRE_REQ.DL1_MISS	Hardware Prefetch requests that miss the L1D cache. A request is being counted each time it access the cache & miss it, including if a block is applicable or if hit the Fill Buffer for example.	This accounts for both L1 streamer and IP-based (IPP) HW prefetchers.
51H	01H	L1D.REPLACEMENT	Counts the number of lines brought into the L1 data cache.	
51H	02H	L1D.ALLOCATED_IN_M	Counts the number of allocations of modified L1D cache lines.	
51H	04H	L1D.EVICTION	Counts the number of modified lines evicted from the L1 data cache due to replacement.	
51H	08H	L1D.ALL_M_REPLACEMENT	Cache lines in M state evicted out of L1D due to Snoop HitM or dirty line replacement	
59H	20H	PARTIAL_RAT_STALLS.FLAGS_MERGE_UOP	Increments the number of flags-merge uops in flight each cycle. Set Cmask = 1 to count cycles.	
59H	40H	PARTIAL_RAT_STALLS.SLOW_LEA_WINDOW	Cycles with at least one slow LEA uop allocated.	
59H	80H	PARTIAL_RAT_STALLS.MUL_SINGLE_UOP	Number of Multiply packed/scalar single precision uops allocated.	
5BH	0CH	RESOURCE_STALLS2.ALL_FL_EMPTY	Cycles stalled due to free list empty	
5BH	0FH	RESOURCE_STALLS2.ALL_PRF_CONTROL	Cycles stalled due to control structures full for physical registers	
5BH	40H	RESOURCE_STALLS2.BOB_FULL	Cycles Allocator is stalled due Branch Order Buffer.	
5BH	4FH	RESOURCE_STALLS2.OOO_RSRC	Cycles stalled due to out of order resources full	
5CH	01H	CPL_CYCLES.RING0	Unhalted core cycles when the thread is in ring 0	Use Edge to count transition
5CH	02H	CPL_CYCLES.RING123	Unhalted core cycles when the thread is not in ring 0	
5EH	01H	RS_EVENTS.EMPTY_CYCLES	Cycles the RS is empty for the thread.	

Table 19-3 Non-Architectural Performance Events In the Processor Core common to Intel Core i7, i5, i3 Processors 2xxx Series and Next Generation Intel Xeon Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
60H	01H	OFFCORE_REQUEST S_OUTSTANDING.DE MAND_DATA_RD	Offcore outstanding Demand Data Read transactions in SQ to uncore. Set Cmask=1 to count cycles.	
60H	04H	OFFCORE_REQUEST S_OUTSTANDING.DE MAND_RFO	Offcore outstanding RFO store transactions in SQ to uncore. Set Cmask=1 to count cycles.	
60H	08H	OFFCORE_REQUEST S_OUTSTANDING.AL L_DATA_RD	Offcore outstanding cacheable data read transactions in SQ to uncore. Set Cmask=1 to count cycles.	
63H	01H	LOCK_CYCLES.SPLIT_ LOCK_UC_LOCK_DUR ATION	Cycles in which the L1D and L2 are locked, due to a UC lock or split lock.	
63H	02H	LOCK_CYCLES.CACHE _LOCK_DURATION	Cycles in which the L1D is locked.	
79H	02H	IDQ.EMPTY	Counts cycles the IDQ is empty.	
79H	04H	IDQ.MITE_UOPS	Increment each cycle # of uops delivered to IDQ from MITE path. Set Cmask = 1 to count cycles.	Can combine Umask 04H and 20H
79H	08H	IDQ.DSB_UOPS	Increment each cycle. # of uops delivered to IDQ from DSB path. Set Cmask = 1 to count cycles.	Can combine Umask 08H and 10H
79H	10H	IDQ.MS_DSB_UOPS	Increment each cycle # of uops delivered to IDQ when MS busy by DSB. Set Cmask = 1 to count cycles MS is busy. Set Cmask=1 and Edge =1 to count MS activations.	Can combine Umask 08H and 10H
79H	20H	IDQ.MS_MITE_UOPS	Increment each cycle # of uops delivered to IDQ when MS is busy by MITE. Set Cmask = 1 to count cycles.	Can combine Umask 04H and 20H
79H	30H	IDQ.MS_UOPS	Increment each cycle # of uops delivered to IDQ from MS by either DSB or MITE. Set Cmask = 1 to count cycles.	Can combine Umask 04H, 08H and 30H
80H	02H	ICACHE.MISSES	Number of Instruction Cache, Streaming Buffer and Victim Cache Misses. Includes UC accesses.	
85H	01H	ITLB_MISSES.MISS_C AUSES_A_WALK	Misses in all ITLB levels that cause page walks	
85H	02H	ITLB_MISSES.WALK_ COMPLETED	Misses in all ITLB levels that cause completed page walks	
85H	04H	ITLB_MISSES.WALK_ DURATION	Cycle PMH is busy with a walk.	
85H	10H	ITLB_MISSES.STLB_H IT	Number of cache load STLB hits. No page walk.	

Table 19-3 Non-Architectural Performance Events In the Processor Core common to Intel Core i7, i5, i3 Processors 2xxx Series and Next Generation Intel Xeon Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
87H	01H	ILD_STALL.LCP	Stalls caused by changing prefix length of the instruction.	
87H	04H	ILD_STALL.IQ_FULL	Stall cycles due to IQ is full.	
88H	01H	BR_INST_EXEC.COND	Qualify conditional near branch instructions executed, but not necessarily retired.	Must combine with umask 40H, 80H
88H	02H	BR_INST_EXEC.DIRECT_JMP	Qualify all unconditional near branch instructions excluding calls and indirect branches.	Must combine with umask 80H
88H	04H	BR_INST_EXEC.INDIRECT_JMP_NON_CALL_RET	Qualify executed indirect near branch instructions that are not calls nor returns.	Must combine with umask 80H
88H	08H	BR_INST_EXEC.RETURN_NEAR	Qualify indirect near branches that have a return mnemonic.	Must combine with umask 80H
88H	10H	BR_INST_EXEC.DIRECT_NEAR_CALL	Qualify unconditional near call branch instructions, excluding non call branch, executed.	Must combine with umask 80H
88H	20H	BR_INST_EXEC.INDIRECT_NEAR_CALL	Qualify indirect near calls, including both register and memory indirect, executed.	Must combine with umask 80H
88H	40H	BR_INST_EXEC.NONTAKEN	Qualify non-taken near branches executed.	Applicable to umask 01H only
88H	80H	BR_INST_EXEC.TAKEN	Qualify taken near branches executed. Must combine with 01H,02H, 04H, 08H, 10H, 20H	
88H	FFH	BR_INST_EXEC.ALL_BRANCHES	Counts all near executed branches (not necessarily retired).	
89H	01H	BR_MISP_EXEC.COND	Qualify conditional near branch instructions mispredicted.	Must combine with umask 40H, 80H
89H	04H	BR_MISP_EXEC.INDIRECT_JMP_NON_CALL_RET	Qualify mispredicted indirect near branch instructions that are not calls nor returns.	Must combine with umask 80H
89H	08H	BR_MISP_EXEC.RETURN_NEAR	Qualify mispredicted indirect near branches that have a return mnemonic.	Must combine with umask 80H
89H	10H	BR_MISP_EXEC.DIRECT_NEAR_CALL	Qualify mispredicted unconditional near call branch instructions, excluding non call branch, executed.	Must combine with umask 80H
89H	20H	BR_MISP_EXEC.INDIRECT_NEAR_CALL	Qualify mispredicted indirect near calls, including both register and memory indirect, executed.	Must combine with umask 80H
89H	40H	BR_MISP_EXEC.NONTAKEN	Qualify mispredicted non-taken near branches executed,.	Applicable to umask 01H only

Table 19-3 Non-Architectural Performance Events In the Processor Core common to Intel Core i7, i5, i3 Processors 2xxx Series and Next Generation Intel Xeon Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
89H	80H	BR_MISP_EXEC.TAKEN	Qualify mispredicted taken near branches executed. Must combine with 01H,02H, 04H, 08H, 10H, 20H	
89H	FFH	BR_MISP_EXEC.ALL_BRANCHES	Counts all near executed branches (not necessarily retired).	
9CH	01H	IDQ_UOPS_NOT_DELIVERED.CORE	Count number of non-delivered uops to RAT per thread.	Use Cmask to qualify uop b/w
A1H	01H	UOPS_DISPATCHED_PORT.PORT_0	Cycles which a Uop is dispatched on port 0.	
A1H	02H	UOPS_DISPATCHED_PORT.PORT_1	Cycles which a Uop is dispatched on port 1.	
A1H	04H	UOPS_DISPATCHED_PORT.PORT_2_LD	Cycles which a load uop is dispatched on port 2.	
A1H	08H	UOPS_DISPATCHED_PORT.PORT_2_STA	Cycles which a store address uop is dispatched on port 2.	
A1H	0CH	UOPS_DISPATCHED_PORT.PORT_2	Cycles which a Uop is dispatched on port 2.	
A1H	10H	UOPS_DISPATCHED_PORT.PORT_3_LD	Cycles which a load uop is dispatched on port 3.	
A1H	20H	UOPS_DISPATCHED_PORT.PORT_3_STA	Cycles which a store address uop is dispatched on port 3.	
A1H	30H	UOPS_DISPATCHED_PORT.PORT_3	Cycles which a Uop is dispatched on port 3.	
A1H	40H	UOPS_DISPATCHED_PORT.PORT_4	Cycles which a Uop is dispatched on port 4.	
A1H	80H	UOPS_DISPATCHED_PORT.PORT_5	Cycles which a Uop is dispatched on port 5.	
A2H	01H	RESOURCE_STALLS.ANY	Cycles Allocation is stalled due to Resource Related reason.	
A2H	02H	RESOURCE_STALLS.LB	Counts the cycles of stall due to lack of load buffers.	
A2H	04H	RESOURCE_STALLS.RS	Cycles stalled due to no eligible RS entry available.	
A2H	08H	RESOURCE_STALLS.SB	Cycles stalled due to no store buffers available. (not including draining form sync).	
A2H	10H	RESOURCE_STALLS.ROB	Cycles stalled due to re-order buffer full.	
A2H	20H	RESOURCE_STALLS.FCSW	Cycles stalled due to writing the FPU control word.	
A2H	40H	RESOURCE_STALLS.MXCSR	Cycles stalled due to the MXCSR register rename occurring to close to a previous MXCSR rename.	

Table 19-3 Non-Architectural Performance Events In the Processor Core common to Intel Core i7, i5, i3 Processors 2xxx Series and Next Generation Intel Xeon Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
A2H	80H	RESOURCE_STALLS.OTHER	Cycles stalled while execution was stalled due to other resource issues.	
ABH	01H	DSB2MITE_SWITCHES.COUNT	Number of DSB to MITE switches.	
ABH	02H	DSB2MITE_SWITCHES.PENALTY_CYCLES	Cycles DSB to MITE switches caused delay.	
ACH	02H	DSB_FILL.OTHER_CANCEL	Cases of cancelling valid DSB fill not because of exceeding way limit	
ACH	08H	DSB_FILL.EXCEED_DSB_LINES	DSB Fill encountered > 3 DSB lines.	
ACH	0AH	DSB_FILL.ALL_CANCEL	Cases of cancelling valid Decode Stream Buffer (DSB) fill not because of exceeding way limit	
AEH	01H	ITLB.ITLB_FLUSH	Counts the number of ITLB flushes, includes 4k/2M/4M pages.	
B0H	01H	OFFCORE_REQUESTS.DEMAND_DATA_RD	Demand data read requests sent to uncore.	
B0H	04H	OFFCORE_REQUESTS.DEMAND_RFO	Demand RFO read requests sent to uncore., including regular RFOs, locks, ItoM	
B0H	08H	OFFCORE_REQUESTS.ALL_DATA_RD	Data read requests sent to uncore (demand and prefetch).	
B1H	01H	UOPS_DISPATCHED.THREAD	Counts total number of uops to be dispatched per-thread each cycle. Set Cmask = 1, INV =1 to count stall cycles.	
B1H	02H	UOPS_DISPATCHED.CORE	Counts total number of uops to be dispatched per-core each cycle.	Do not need to set ANY
B2H	01H	OFFCORE_REQUESTS_BUFFER.SQ_FULL	Offcore requests buffer cannot take more entries for this thread core.	
B6H	01H	AGU_BYPASS_CANCEL.COUNT	Counts executed load operations with all the following traits: 1. addressing of the format [base + offset], 2. the offset is between 1 and 2047, 3. the address specified in the base register is in one page and the address [base+offset] is in another page.	
B7H	01H	OFF_CORE_RESPONSE_0	see Section 18.8.5, "Off-core Response Performance Monitoring"; PMC0 only.	Requires programming MSR 01A6H
BBH	01H	OFF_CORE_RESPONSE_1	See Section 18.8.5, "Off-core Response Performance Monitoring". PMC3 only.	Requires programming MSR 01A7H

Table 19-3 Non-Architectural Performance Events In the Processor Core common to Intel Core i7, i5, i3 Processors 2xxx Series and Next Generation Intel Xeon Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
BDH	01H	TLB_FLUSH.DTLB_THREAD	DTLB flush attempts of the thread-specific entries	
BDH	20H	TLB_FLUSH.STLB_ANY	Count number of STLB flush attempts	
BFH	05H	L1D_BLOCKS.BANK_CONFLICT_CYCLES	Cycles when dispatched loads are cancelled due to L1D bank conflicts with other load ports	cmask=1
C0H	00H	INST_RETIRED.ANY_P	Number of instructions at retirement	See Table 19-1
C0H	01H	INST_RETIRED.PREC_DIST	Precise instruction retired event with HW to reduce effect of PEBS shadow in IP distribution	PMC1 only; Must quiesce other PMCs.
C0H	02H	INST_RETIRED.X87	X87 instruction retired event	
C1H	02H	OTHER_ASSISTS.ITLB_MISS_RETIRED	Instructions that experienced an ITLB miss.	
C1H	08H	OTHER_ASSISTS.AVX_STORE	Number of assists associated with 256-bit AVX store operations.	
C1H	10H	OTHER_ASSISTS.AVX_TO_SSE	Number of transitions from AVX-256 to legacy SSE when penalty applicable.	
C1H	20H	OTHER_ASSISTS.SSE_TO_AVX	Number of transitions from SSE to AVX-256 when penalty applicable.	
C2H	01H	UOPS_RETIRED.ALL	Counts the number of micro-ops retired, Use cmask=1 and invert to count active cycles or stalled cycles.	Supports PEBS
C2H	02H	UOPS_RETIRED.RETIRE_SLOTS	Counts the number of retirement slots used each cycle.	
C3H	02H	MACHINE_CLEARS.MEMORY_ORDERING	Counts the number of machine clears due to memory order conflicts.	
C3H	04H	MACHINE_CLEARS.SMC	Counts the number of times that a program writes to a code section.	
C3H	20H	MACHINE_CLEARS.MASKMOV	Counts the number of executed AVX masked load operations that refer to an illegal address range with the mask bits set to 0.	
C4H	00H	BR_INST_RETIRED.ALL_BRANCHES	Branch instructions at retirement	See Table 19-1
C4H	01H	BR_INST_RETIRED.CONDITIONAL	Counts the number of conditional branch instructions retired.	Supports PEBS
C4H	02H	BR_INST_RETIRED.NEAR_CALL	Direct and indirect near call instructions retired.	

Table 19-3 Non-Architectural Performance Events In the Processor Core common to Intel Core i7, i5, i3 Processors 2xxx Series and Next Generation Intel Xeon Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
C4H	04H	BR_INST_RETIRED.ALL_BRANCHES	Counts the number of branch instructions retired.	
C4H	08H	BR_INST_RETIRED.NEAR_RETURN	Counts the number of near return instructions retired.	
C4H	10H	BR_INST_RETIRED.NOT_TAKEN	Counts the number of not taken branch instructions retired.	
C4H	20H	BR_INST_RETIRED.NEAR_TAKEN	Number of near taken branches retired.	
C4H	40H	BR_INST_RETIRED.FAR_BRANCH	Number of far branches retired.	
C5H	00H	BR_MISP_RETIRED.ALL_BRANCHES	Mispredicted branch instructions at retirement	See Table 19-1
C5H	01H	BR_MISP_RETIRED.CONDITIONAL	Mispredicted conditional branch instructions retired.	Supports PEBS
C5H	02H	BR_MISP_RETIRED.NEAR_CALL	Direct and indirect mispredicted near call instructions retired.	
C5H	04H	BR_MISP_RETIRED.ALL_BRANCHES	Mispredicted macro branch instructions retired.	
C5H	10H	BR_MISP_RETIRED.NOT_TAKEN	Mispredicted not taken branch instructions retired.	
C5H	20H	BR_MISP_RETIRED.TAKEN	Mispredicted taken branch instructions retired.	
CAH	02H	FP_ASSIST.X87_OUTPUT	Number of X87 assists due to output value.	
CAH	04H	FP_ASSIST.X87_INPUT	Number of X87 assists due to input value.	
CAH	08H	FP_ASSIST.SIMD_OUTPUT	Number of SIMD FP assists due to Output values	
CAH	10H	FP_ASSIST.SIMD_INPUT	Number of SIMD FP assists due to input values	
CAH	1EH	FP_ASSIST.ANY	Cycles with any input/output SSE* or FP assists	
CCH	20H	ROB_MISC_EVENTS.LBR_INSERTS	Count cases of saving new LBR records by hardware.	
CDH	01H	MEM_TRANS_RETIRED.LOAD_LATENCY	Sample loads with specified latency threshold. PMC3 only.	Specify threshold in MSR 0x3F6
CDH	02H	MEM_TRANS_RETIRED.PRECISE_STORE	Sample stores and collect precise store operation via PEBS record. PMC3 only.	See Section 18.8.4.3
D0H	01H	MEM_UOP_RETIRED.LOADS	Qualify retired memory uops that are loads. Combine with umask 10H, 20H, 40H, 80H.	Supports PEBS

Table 19-3 Non-Architectural Performance Events In the Processor Core common to Intel Core i7, i5, i3 Processors 2xxx Series and Next Generation Intel Xeon Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
D0H	02H	MEM_UOP_RETIRED.STORES	Qualify retired memory uops that are stores. Combine with umask 10H, 20H, 40H, 80H.	
D0H	10H	MEM_UOP_RETIRED.STLB_MISS	Qualify retired memory uops with STLB miss. Must combine with umask 01H, 02H, to produce counts.	
D0H	20H	MEM_UOP_RETIRED.LOCK	Qualify retired memory uops with lock. Must combine with umask 01H, 02H, to produce counts.	
D0H	40H	MEM_UOP_RETIRED.SPLIT	Qualify retired memory uops with line split. Must combine with umask 01H, 02H, to produce counts.	
D0H	80H	MEM_UOP_RETIRED.ALL	Qualify any retired memory uops. Must combine with umask 01H, 02H, to produce counts.	
D1H	01H	MEM_LOAD_UOPS_RETIRED.L1_HIT	Retired load uops with L1 cache hits as data sources.	Supports PEBS
D1H	02H	MEM_LOAD_UOPS_RETIRED.L2_HIT	Retired load uops with L2 cache hits as data sources.	
D1H	40H	MEM_LOAD_UOPS_RETIRED.HIT_LFB	Retired load uops which data sources were load uops missed L1 but hit FB due to preceding miss to the same cache line with data not ready.	
D2H	01H	MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_MISS	Retired load uops which data sources were LLC hit and cross-core snoop missed in on-pkg core cache.	Supports PEBS
D2H	02H	MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_HIT	Retired load uops which data sources were LLC and cross-core snoop hits in on-pkg core cache.	
D2H	04H	MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_HITM	Retired load uops which data sources were HitM responses from shared LLC.	
D2H	08H	MEM_LOAD_UOPS_LLC_HIT_RETIRED.XSNP_NONE	Retired load uops which data sources were hits in LLC without snoops required.	
D4H	02H	MEM_LOAD_UOPS_MISC_RETIRED.LLC_MISS	Retired load uops with unknown information as data source in cache serviced the load.	Supports PEBS.
F0H	01H	L2_TRANS.DEMAND_DATA_RD	Demand Data Read requests that access L2 cache	
F0H	02H	L2_TRANS.RFO	RFO requests that access L2 cache	
F0H	04H	L2_TRANS.CODE_RD	L2 cache accesses when fetching instructions	

Table 19-3 Non-Architectural Performance Events In the Processor Core common to Intel Core i7, i5, i3 Processors 2xxx Series and Next Generation Intel Xeon Processors

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
F0H	08H	L2_TRANS.ALL_PF	L2 or LLC HW prefetches that access L2 cache	including rejects.
F0H	10H	L2_TRANS.L1D_WB	L1D writebacks that access L2 cache	
F0H	20H	L2_TRANS.L2_FILL	L2 fill requests that access L2 cache	
F0H	40H	L2_TRANS.L2_WB	L2 writebacks that access L2 cache	
F0H	80H	L2_TRANS.ALL_REQUESTS	Transactions accessing L2 pipe	
F1H	01H	L2_LINES_IN.I	L2 cache lines in I state filling L2	Counting does not cover rejects.
F1H	02H	L2_LINES_IN.S	L2 cache lines in S state filling L2	Counting does not cover rejects.
F1H	04H	L2_LINES_IN.E	L2 cache lines in E state filling L2	Counting does not cover rejects.
F1H	07H	L2_LINES_IN.ALL	L2 cache lines filling L2	Counting does not cover rejects.
F2H	01H	L2_LINES_OUT.DEMAND_CLEAN	Clean L2 cache lines evicted by demand	
F2H	02H	L2_LINES_OUT.DEMAND_DIRTY	Dirty L2 cache lines evicted by demand	
F2H	04H	L2_LINES_OUT.PF_CLEAN	Clean L2 cache lines evicted by L2 prefetch	
F2H	08H	L2_LINES_OUT.PF_DIRTY	Dirty L2 cache lines evicted by L2 prefetch	
F2H	0AH	L2_LINES_OUT.DIRTY_ALL	Dirty L2 cache lines filling the L2	Counting does not cover rejects.
F4H	10H	SQ_MISC.SPLIT_LOCK	Split locks in SQ	

Non-architecture performance monitoring events in the processor core that are applicable only to the Intel processor with CPUID signature of DisplayFamily_DisplayModel 06_2AH are listed in Table 19-4.

Table 19-4 Non-Architectural Performance Events applicable only to the Processor Core for Intel Core i7, i5, i3 Processor 2xxx Series

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
D1H	04H	MEM_LOAD_UOPS_RETIRED.LLC_HIT	Retired load uops which data sources were data hits in LLC without snoops required.	Supports PEBS
B7H/BBH	01H	OFF_CORE_RESPONSE_N	Sub-events of OFF_CORE_RESPONSE_N (suffix N = 0, 1) programmed using MSR 01A6H/01A7H with values shown in the comment column.	
		OFFCORE_RESPONSE.ALL_CODE_RD.LLC_HIT_N		0x10003C0244
		OFFCORE_RESPONSE.ALL_CODE_RD.LLC_HIT.NO_SNOOP_NEEDED_N		0x1003C0244
		OFFCORE_RESPONSE.ALL_CODE_RD.LLC_HIT.SNOOP_MISS_N		0x2003C0244
		OFFCORE_RESPONSE.ALL_CODE_RD.LLC_HIT.MISS_DRAM_N		0x300400244
		OFFCORE_RESPONSE.ALL_DATA_RD.LLC_HIT.ANY_RESPONSE_N		0x3F803C0091
		OFFCORE_RESPONSE.ALL_DATA_RD.LLC_MISS.DRAM_N		0x300400091
		OFFCORE_RESPONSE.ALL_PF_CODE_RD.LLC_HIT.ANY_RESPONSE_N		0x3F803C0240
		OFFCORE_RESPONSE.ALL_PF_CODE_RD.LLC_HIT.HIT_OTHER_CORE_NO_FWD_N		0x4003C0240
		OFFCORE_RESPONSE.ALL_PF_CODE_RD.LLC_HIT.HITM_OTHER_CORE_N		0x10003C0240
		OFFCORE_RESPONSE.ALL_PF_CODE_RD.LLC_HIT.NO_SNOOP_NEEDED_N		0x1003C0240
		OFFCORE_RESPONSE.ALL_PF_CODE_RD.LLC_HIT.SNOOP_MISS_N		0x2003C0240
		OFFCORE_RESPONSE.ALL_PF_CODE_RD.LLC_MISS.DRAM_N		0x300400240
		OFFCORE_RESPONSE.ALL_PF_DATA_RD.LLC_MISS.DRAM_N		0x300400090
		OFFCORE_RESPONSE.ALL_PF_RFO.LLC_HIT.ANY_RESPONSE_N		0x3F803C0120
		OFFCORE_RESPONSE.ALL_PF_RFO.LLC_HIT.HIT_OTHER_CORE_NO_FWD_N		0x4003C0120
		OFFCORE_RESPONSE.ALL_PF_RFO.LLC_HIT.HITM_OTHER_CORE_N		0x10003C0120
		OFFCORE_RESPONSE.ALL_PF_RfO.LLC_HIT.NO_SNOOP_NEEDED_N		0x1003C0120
		OFFCORE_RESPONSE.ALL_PF_RFO.LLC_HIT.SNOOP_MISS_N		0x2003C0120
		OFFCORE_RESPONSE.ALL_PF_RFO.LLC_MISS.DRAM_N		0x300400120
		OFFCORE_RESPONSE.ALL_READS.LLC_MISS.DRAM_N		0x3004003F7

Table 19-4 Non-Architectural Performance Events applicable only to the Processor Core for Intel Core i7, i5, i3 Processor 2xxx Series

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
		OFFCORE_RESPONSE.ALL_RFO.LLC_HIT.ANY_RESPONSE_N		0x3F803C0122
		OFFCORE_RESPONSE.ALL_RFO.LLC_HIT.HIT_OTHER_CORE_NO_FWD_N		0x4003C0122
		OFFCORE_RESPONSE.ALL_RFO.LLC_HIT.HITM_OTHER_CORE_N		0x10003C0122
		OFFCORE_RESPONSE.ALL_RFO.LLC_HIT.NO_SNOOP_NEEDED_N		0x1003C0122
		OFFCORE_RESPONSE.ALL_RFO.LLC_HIT.SNOOP_MISS_N		0x2003C0122
		OFFCORE_RESPONSE.ALL_RFO.LLC_MISS.DRAM_N		0x300400122
		OFFCORE_RESPONSE.DEMAND_CODE_RD.LLC_HIT.HIT_OTHER_CORE_NO_FWD_N		0x4003C0004
		OFFCORE_RESPONSE.DEMAND_CODE_RD.LLC_HIT.HITM_OTHER_CORE_N		0x10003C0004
		OFFCORE_RESPONSE.DEMAND_CODE_RD.LLC_HIT.NO_SNOOP_NEEDED_N		0x1003C0004
		OFFCORE_RESPONSE.DEMAND_CODE_RD.LLC_HIT.SNOOP_MISS_N		0x2003C0004
		OFFCORE_RESPONSE.DEMAND_CODE_RD.LLC_MISS.DRAM_N		0x300400004
		OFFCORE_RESPONSE.DEMAND_DATA_RD.LLC_MISS.DRAM_N		0x300400001
		OFFCORE_RESPONSE.DEMAND_RFO.LLC_HIT.ANY_RESPONSE_N		0x3F803C0002
		OFFCORE_RESPONSE.DEMAND_RFO.LLC_HIT.HIT_OTHER_CORE_NO_FWD_N		0x4003C0002
		OFFCORE_RESPONSE.DEMAND_RFO.LLC_HIT.HITM_OTHER_CORE_N		0x10003C0002
		OFFCORE_RESPONSE.DEMAND_RFO.LLC_HIT.NO_SNOOP_NEEDED_N		0x1003C0002
		OFFCORE_RESPONSE.DEMAND_RFO.LLC_HIT.SNOOP_MISS_N		0x2003C0002
		OFFCORE_RESPONSE.DEMAND_RFO.LLC_MISS.DRAM_N		0x300400002
		OFFCORE_RESPONSE.OTHER.ANY_RESPONSE_N		0x18000
		OFFCORE_RESPONSE.PF_L2_CODE_RD.LLC_HIT.HIT_OTHER_CORE_NO_FWD_N		0x4003C0040
		OFFCORE_RESPONSE.PF_L2_CODE_RD.LLC_HIT.HITM_OTHER_CORE_N		0x10003C0040
		OFFCORE_RESPONSE.PF_L2_CODE_RD.LLC_HIT.NO_SNOOP_NEEDED_N		0x1003C0040
		OFFCORE_RESPONSE.PF_L2_CODE_RD.LLC_HIT.SNOOP_MISS_N		0x2003C0040
		OFFCORE_RESPONSE.PF_L2_CODE_RD.LLC_MISS.DRAM_N		0x300400040
		OFFCORE_RESPONSE.PF_L2_DATA_RD.LLC_MISS.DRAM_N		0x300400010

Table 19-4 Non-Architectural Performance Events applicable only to the Processor Core for Intel Core i7, i5, i3 Processor 2xxx Series

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
		OFFCORE_RESPONSE.PF_L2_RFO.LLC_HIT.ANY_RESPONSE_N		0x3F803C0020
		OFFCORE_RESPONSE.PF_L2_RFO.LLC_HIT.HIT_OTHER_CORE_NO_FWD_N		0x4003C0020
		OFFCORE_RESPONSE.PF_L2_RFO.LLC_HIT.HITM_OTHER_CORE_N		0x10003C0020
		OFFCORE_RESPONSE.PF_L2_RFO.LLC_HIT.NO_SNOOP_NEEDED_N		0x1003C0020
		OFFCORE_RESPONSE.PF_L2_RFO.LLC_HIT.SNOOP_MISS_N		0x2003C0020
		OFFCORE_RESPONSE.PF_L2_RFO.LLC_MISS.DRAM_N		0x300400020
		OFFCORE_RESPONSE.PF_LLC_CODE_RD.LLC_HIT.HIT_OTHER_CORE_NO_FWD_N		0x4003C0200
		OFFCORE_RESPONSE.PF_LLC_CODE_RD.LLC_HIT.HITM_OTHER_CORE_N		0x10003C0200
		OFFCORE_RESPONSE.PF_LLC_CODE_RD.LLC_HIT.NO_SNOOP_NEEDED_N		0x1003C0200
		OFFCORE_RESPONSE.PF_LLC_CODE_RD.LLC_HIT.SNOOP_MISS_N		0x2003C0200
		OFFCORE_RESPONSE.PF_LLC_CODE_RD.LLC_MISS.DRAM_N		0x300400200
		OFFCORE_RESPONSE.PF_LLC_DATA_RD.LLC_MISS.DRAM_N		0x300400080
		OFFCORE_RESPONSE.PF_LLC_RFO.LLC_HIT.ANY_RESPONSE_N		0x3F803C0100
		OFFCORE_RESPONSE.PF_LLC_RFO.LLC_HIT.HIT_OTHER_CORE_NO_FWD_N		0x4003C0100
		OFFCORE_RESPONSE.PF_LLC_RFO.LLC_HIT.HITM_OTHER_CORE_N		0x10003C0100
		OFFCORE_RESPONSE.PF_LLC_RFO.LLC_HIT.NO_SNOOP_NEEDED_N		0x1003C0100
		OFFCORE_RESPONSE.PF_LLC_RFO.LLC_HIT.SNOOP_MISS_N		0x2003C0100
		OFFCORE_RESPONSE.PF_LLC_RFO.LLC_MISS.DRAM_N		0x300400100

Non-architecture performance monitoring events in the processor core that are applicable only to the next generation Intel Xeon processor with CPUID signature of DisplayFamily_DisplayModel 06_2DH are listed in Table 19-5.

Table 19-5 Non-Architectural Performance Events Applicable only to the processor core of Next Generation Intel Xeon processor

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
B7H/BBH	01H	OFF_CORE_RESPONSE_N	Sub-events of OFF_CORE_RESPONSE_N (suffix N = 0, 1) programmed using MSR 01A6H/01A7H with values shown in the comment column.	
		OFFCORE_RESPONSE.DEMAND_CODE_RD.LLC_MISS.ANY_RESPONSE_N		0x3FFFC00004
		OFFCORE_RESPONSE.DEMAND_CODE_RD.LLC_MISS.LOCAL_DRAM_N		0x600400004
		OFFCORE_RESPONSE.DEMAND_CODE_RD.LLC_MISS.REMOTE_DRAM_N		0x67F800004
		OFFCORE_RESPONSE.DEMAND_CODE_RD.LLC_MISS.REMOTE_HIT_FWD_N		0x87F800004
		OFFCORE_RESPONSE.DEMAND_CODE_RD.LLC_MISS.REMOTE_HITM_N		0x107FC00004
		OFFCORE_RESPONSE.DEMAND_DATA_RD.LLC_MISS.ANY_DRAM_N		0x67FC00001
		OFFCORE_RESPONSE.DEMAND_DATA_RD.LLC_MISS.ANY_RESPONSE_N		0x3F803C0001
		OFFCORE_RESPONSE.DEMAND_DATA_RD.LLC_MISS.LOCAL_DRAM_N		0x600400001
		OFFCORE_RESPONSE.DEMAND_DATA_RD.LLC_MISS.REMOTE_DRAM_N		0x67F800001
		OFFCORE_RESPONSE.DEMAND_DATA_RD.LLC_MISS.REMOTE_HIT_FWD_N		0x87F800001
		OFFCORE_RESPONSE.DEMAND_DATA_RD.LLC_MISS.REMOTE_HITM_N		0x107FC00001
		OFFCORE_RESPONSE.PF_L2_CODE_RD.LLC_MISS.ANY_RESPONSE_N		0x3F803C0040
		OFFCORE_RESPONSE.PF_L2_DATA_RD.LLC_MISS.ANY_DRAM_N		0x67FC00010
		OFFCORE_RESPONSE.PF_L2_DATA_RD.LLC_MISS.ANY_RESPONSE_N		0x3F803C0010
		OFFCORE_RESPONSE.PF_L2_DATA_RD.LLC_MISS.LOCAL_DRAM_N		0x600400010
		OFFCORE_RESPONSE.PF_L2_DATA_RD.LLC_MISS.REMOTE_DRAM_N		0x67F800010
		OFFCORE_RESPONSE.PF_L2_DATA_RD.LLC_MISS.REMOTE_HIT_FWD_N		0x87F800010
		OFFCORE_RESPONSE.PF_L2_DATA_RD.LLC_MISS.REMOTE_HITM_N		0x107FC00010

Table 19-5 Non-Architectural Performance Events Applicable only to the processor core of Next Generation Intel Xeon processor

Event Num.	Umask Value	Event Mask Mnemonic	Description	Comment
		OFFCORE_RESPONSE.PF_LLC_CODE_RD.LLC_MISS.ANY_RESPONSE_N		0x3FFFC00200
		OFFCORE_RESPONSE.PF_LLC_DATA_RD.LLC_MISS.ANY_RESPONSE_N		0x3FFFC00080

...

14. Updates to Chapter 25, Volume 3C

Change bars show changes to Chapter 25 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3C:* System Programming Guide, Part 3.

...

25.7.4.2 General Operation of the VMFUNC Instruction

The VMFUNC instruction causes an invalid-opcode exception (#UD) if the "enable VM functions" VM-execution controls is 0[1] or the value of EAX is greater than 63 (only VM functions 0–63 can be enable). Otherwise, the instruction causes a VM exit if the bit at position EAX is 0 in the VM-function controls (the selected VM function is not enabled). If such a VM exit occurs, the basic exit reason used is 59 (3BH), indicating "VMFUNC", and the length of the VMFUNC instruction is saved into the VM-exit instruction-length field. If the instruction causes neither an invalid-opcode exception nor a VM exit due to a disabled VM function, it performs the functionality of the VM function specified by the value in EAX.

Individual VM functions may perform additional fault checking (e.g., one might cause a general-protection exception if CPL > 0). In addition, specific VM functions may include checks that might result in a VM exit. If such a VM exit occurs, VM-exit information is saved as described in the previous paragraph. The specification of a VM function may indicate that additional VM-exit information is provided.

The specific behavior of the EPTP-switching VM function (including checks that result in VM exits) is given in Section 25.7.4.3.

25.7.4.3 EPTP Switching

EPTP switching is VM function 0. This VM function allows software in VMX non-root operation to load a new value for the EPT pointer (EPTP), thereby establishing a different EPT paging-structure hierarchy (see Section 28.2 for details of the operation of EPT). Software is limited to selecting from a list of potential EPTP values configured in advance by software in VMX root operation.

Specifically, the value of ECX is used to select an entry from the EPTP list, the 4-KByte structure referenced by the EPTP-list address (see Section 24.6.14; because this structure contains 512 8-Byte entries, VMFUNC causes a VM exit if ECX ≥ 512). If the selected entry is a valid EPTP value (it would not cause VM entry to fail; see Section 26.2.1.1), it is stored in the EPTP field of the current VMCS and is used for subsequent accesses using guest-physical addresses. The following pseudocode provides details:

```
IF ECX ≥ 512
   THEN VM exit;
   ELSE
       tent_EPTP ← 8 bytes from EPTP-list address + 8 * ECX;
       IF tent_EPTP is not a valid EPTP value (would cause VM entry to fail if in EPTP)
           THEN VMexit;
           ELSE
```

1. "Enable VM functions" is a secondary processor-based VM-execution control. If bit 31 of the primary processor-based VM-execution controls is 0, VMX non-root operation functions as if the "enable VM functions" VM-execution control were 0. See Section 24.6.2.

```
                write tent_EPTP to the EPTP field in the current VMCS;
                start using tent_EPTP as the new EPTP value for address translation;
        FI;
FI;
```

Execution of the EPTP-switching VM function does not modify the state of any registers; no flags are modified.

As noted in Section 25.7.4.2, an execution of the EPTP-switching VM function that causes a VM exit (as specified above), uses the basic exit reason 59, indicating "VMFUNC". The length of the VMFUNC instruction is saved into the VM-exit instruction-length field. No additional VM-exit information is provided.

An execution of VMFUNC loads EPTP from the EPTP list (and thus does not cause a fault or VM exit) is called an **EPTP-switching VMFUNC**. After an EPTP-switching VMFUNC, control passes to the next instruction. The logical processor starts creating and using guest-physical and combined mappings associated with the new value of bits 51:12 of EPTP; the combined mappings created and used are associated with the current VPID and PCID (these are not changed by VMFUNC).[1] If the "enable VPID" VM-execution control is 0, an EPTP-switching VMFUNC invalidates combined mappings associated with VPID 0000H (for all PCIDs and for all EP4TA values, where EP4TA is the value of bits 51:12 of EPTP).

Because an EPTP-switching VMFUNC may change the translation of guest-physical addresses, it may affect use of the guest-physical address in CR3. The EPTP-switching VMFUNC cannot itself cause a VM exit due to an EPT violation or an EPT misconfiguration due to the translation of that guest-physical address through the new EPT paging structures. The following items provide details that apply if CR0.PG = 1:

- If 32-bit paging or IA-32e paging is in use (either CR4.PAE = 0 or IA32_EFER.LMA = 1), the next memory access with a linear address uses the translation of the guest-physical address in CR3 through the new EPT paging structures. As a result, this access may cause a VM exit due to an EPT violation or an EPT misconfiguration encountered during that translation.
- If PAE paging is in use (CR4.PAE = 1 and IA32_EFER.LMA = 0), an EPTP-switching VMFUNC **does not** load the four page-directory-pointer-table entries (PDPTEs) from the guest-physical address in CR3. The logical processor continues to use the four guest-physical addresses already present in the PDPTEs. The guest-physical address in CR3 is not translated through the new EPT paging structures (until some operation that would load the PDPTEs).

  The EPTP-switching VMFUNC cannot itself cause a VM exit due to an EPT violation or an EPT misconfiguration encountered during the translation of a guest-physical address in any of the PDPTEs. A subsequent memory access with a linear address uses the translation of the guest-physical address in the appropriate PDPTE through the new EPT paging structures. As a result, such an access may cause a VM exit due to an EPT violation or an EPT misconfiguration encountered during that translation.

...

1. If the "enable VPID" VM-execution control is 0, the current VPID is 0000H; if CR4.PCIDE = 0, the current PCID is 000H.

15. Updates to Chapter 33, Volume 3C

Change bars show changes to Chapter 33 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3C:* System Programming Guide, Part 3.

...

33.1 OVERVIEW

This chapter describes the virtual-machine extensions (VMX) for the Intel 64 and IA-32 architectures. VMX is intended to support virtualization of processor hardware and a system software layer acting as a host to multiple guest software environments. The virtual-machine extensions (VMX) includes five instructions that manage the virtual-machine control structure (VMCS), four instructions that manage VMX operation, two TLB-management instructions, and two instructions for use by guest software. Additional details of VMX are described in *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3C*.

The behavior of the VMCS-maintenance instructions is summarized below:

- **VMPTRLD —** This instruction takes a single 64-bit source operand that is in memory. It makes the referenced VMCS active and current, loading the current-VMCS pointer with this operand and establishes the current VMCS based on the contents of VMCS-data area in the referenced VMCS region. Because this makes the referenced VMCS active, a logical processor may start maintaining on the processor some of the VMCS data for the VMCS.
- **VMPTRST —** This instruction takes a single 64-bit destination operand that is in memory. The current-VMCS pointer is stored into the destination operand.
- **VMCLEAR —** This instruction takes a single 64-bit operand that is in memory. The instruction sets the launch state of the VMCS referenced by the operand to "clear", renders that VMCS inactive, and ensures that data for the VMCS have been written to the VMCS-data area in the referenced VMCS region. If the operand is the same as the current-VMCS pointer, that pointer is made invalid.
- **VMREAD —** This instruction reads a component from the VMCS (the encoding of that field is given in a register operand) and stores it into a destination operand that may be a register or in memory.
- **VMWRITE —** This instruction writes a component to the VMCS (the encoding of that field is given in a register operand) from a source operand that may be a register or in memory.

The behavior of the VMX management instructions is summarized below:

- **VMLAUNCH —** This instruction launches a virtual machine managed by the VMCS. A VM entry occurs, transferring control to the VM.
- **VMRESUME —** This instruction resumes a virtual machine managed by the VMCS. A VM entry occurs, transferring control to the VM.
- **VMXOFF —** This instruction causes the processor to leave VMX operation.
- **VMXON —** This instruction takes a single 64-bit source operand that is in memory. It causes a logical processor to enter VMX root operation and to use the memory referenced by the operand to support VMX operation.

The behavior of the VMX-specific TLB-management instructions is summarized below:

- **INVEPT —** This instruction invalidates entries in the TLBs and paging-structure caches that were derived from extended page tables (EPT).
- **INVVPID —** This instruction invalidates entries in the TLBs and paging-structure caches based on a Virtual-Processor Identifier (VPID).

None of the instructions above can be executed in compatibility mode; they generate invalid-opcode exceptions if executed in compatibility mode.

The behavior of the guest-available instructions is summarized below:

- **VMCALL —** This instruction allows software in VMX non-root operation to call the VMM for service. A VM exit occurs, transferring control to the VMM.
- **VMFUNC —** This instruction allows software in VMX non-root operation to invoke a VM function (processor functionality enabled and configured by software in VMX root operation) without a VM exit.

...

VMFUNC—Invoke VM function

Opcode	Instruction	Description
0F 01 D4	VMFUNC	Invoke VM function specified in EAX.

Description

This instruction allows software in VMX non-root operation to invoke a VM function, which is processor functionality enabled and configured by software in VMX root operation. The value of EAX selects the specific VM function being invoked.

The behavior of each VM function (including any additional fault checking) is specified in Section 25.7.4, “VM Functions,” in *Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 3C*.

Operation

```
Perform functionality of the VM function specified in EAX;
```

Flags Affected

Depends on the VM function specified in EAX. See Section 25.7.4, “VM Functions,” in *Intel 64 and IA-32 Architecture Software Developer’s Manual, Volume 3C*.

Protected Mode Exceptions (not including those defined by specific VM functions)

#UD	If executed outside VMX non-root operation.
	If “enable VM functions” VM-execution control is 0.
	If EAX ≥ 64.

Real-Address Mode Exceptions

Same exceptions as in protected mode.

Virtual-8086 Exceptions

Same exceptions as in protected mode.

Compatibility Mode Exceptions

Same exceptions as in protected mode.

64-Bit Mode Exceptions

Same exceptions as in protected mode.

...

16. Updates to Chapter 34, Volume 3C

Change bars show changes to Chapter 34 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3C:* System Programming Guide, Part 3.

...

This chapter lists MSRs provided in Intel® Core™ 2 processor family, Intel® Atom™, Intel® Core™ Duo, Intel® Core™ Solo, Pentium® 4 and Intel® Xeon® processors, P6 family processors, and Pentium® processors in Tables 34-13, 34-18, and 34-19, respectively. All MSRs listed can be read with the RDMSR and written with the WRMSR instructions.

Register addresses are given in both hexadecimal and decimal. The register name is the mnemonic register name and the bit description describes individual bits in registers.

Model specific registers and its bit-fields may be supported for a finite range of processor families/models. To distinguish between different processor family and/or models, software must use CPUID.01H leaf function to query the combination of DisplayFamily and DisplayModel to determine model-specific availability of MSRs (see CPUID instruction in Chapter 3, "Instruction Set Reference, A-L" in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A*). Table 34-1 lists the signature values of DisplayFamily and DisplayModel for various processor families or processor number series.

Table 34-1 CPUID Signature Values of DisplayFamily_DisplayModel

DisplayFamily_DisplayModel	Processor Families/Processor Number Series
06_3AH	Next Generation Intel Core processor based on Intel microarchitecture Ivy Bridge
06_2DH	Next Generation Intel Xeon processor
06_2FH	Intel Xeon processor E7 family
06_2AH	Intel Xeon processor E3 family; Second Generation Intel Core i7, i5, i3 Processors 2xxx Series
06_2EH	Intel Xeon processor 7500, 6500 series
06_25H, 06_2CH	Intel Xeon processors 3600, 5600 series, Intel Core i7, i5 and i3 Processors
06_1EH, 06_1FH	Intel Core i7 and i5 Processors
06_1AH	Intel Core i7 Processor, Intel Xeon Processor 3400, 3500, 5500 series
06_1DH	Intel Xeon Processor MP 7400 series

Table 34-1 CPUID Signature (Contd.)Values of DisplayFamily_DisplayModel (Contd.)

DisplayFamily_DisplayModel	Processor Families/Processor Number Series
06_17H	Intel Xeon Processor 3100, 3300, 5200, 5400 series, Intel Core 2 Quad processors 8000, 9000 series
06_0FH	Intel Xeon Processor 3000, 3200, 5100, 5300, 7300 series, Intel Core 2 Quad processor 6000 series, Intel Core 2 Extreme 6000 series, Intel Core 2 Duo 4000, 5000, 6000, 7000 series processors, Intel Pentium dual-core processors
06_0EH	Intel Core Duo, Intel Core Solo processors
06_0DH	Intel Pentium M processor
06_1CH	Intel Atom processor
0F_06H	Intel Xeon processor 7100, 5000 Series, Intel Xeon Processor MP, Intel Pentium 4, Pentium D processors
0F_03H, 0F_04H	Intel Xeon Processor, Intel Xeon Processor MP, Intel Pentium 4, Pentium D processors
06_09H	Intel Pentium M processor
0F_02H	Intel Xeon Processor, Intel Xeon Processor MP, Intel Pentium 4 processors
0F_0H, 0F_01H	Intel Xeon Processor, Intel Xeon Processor MP, Intel Pentium 4 processors
06_7H, 06_08H, 06_0AH, 06_0BH	Intel Pentium III Xeon Processor, Intel Pentium III Processor
06_03H, 06_05H	Intel Pentium II Xeon Processor, Intel Pentium II Processor
06_01H	Intel Pentium Pro Processor
05_01H, 05_02H, 05_04H	Intel Pentium Processor, Intel Pentium Processor with MMX Technology

...

34.4 MSRS IN THE INTEL® MICROARCHITECTURE CODE NAME NEHALEM

Table 34-5 lists model-specific registers (MSRs) that are common for Intel® microarchitecture code name Nehalem. These include Intel Core i7 and i5 processor family. Architectural MSR addresses are also included in Table 34-5. These processors have a CPUID signature with DisplayFamily_DisplayModel of 06_1AH, 06_1EH, 06_1FH, 06_2EH, see Table 34-1. Additional MSRs specific to 06_1AH, 06_1EH, 06_1FH are listed in Table 34-6. Some MSRs listed in these tables are used by BIOS. More information about these MSR can be found at http://biosbits.org.

The column "Scope" represents the package/core/thread scope of individual bit field of an MSR. "Thread" means this bit field must be programmed on each logical processor independently. "Core" means the bit field must be programmed on each processor core independently, logical processors in the same core will be affected by change of this bit on the other logical processor in the same core. "Package" means the bit field must be programmed once for each physical package. Change of a bit filed with a package scope will affect all logical processors in that physical package.

Table 34-5 MSRs in Processors Based on Intel Microarchitecture Code Name Nehalem

Register Address		Register Name	Shared/ Unique	Bit Description
Hex	Dec			
…				
1ADH	428	MSR_TURBO_POWER_CURRENT_LIMIT		See http://biosbits.org.
		14:0	Package	**TDP Limit (R/W)** TDP limit in 1/8 Watt granularity.
		15	Package	**TDP Limit Override Enable (R/W)** A value = 0 indicates override is not active, and a value = 1 indicates active.
		30:16	Package	**TDC Limit (R/W)** TDC limit in 1/8 Amp granularity.
		31	Package	**TDC Limit Override Enable (R/W)** A value = 0 indicates override is not active, and a value = 1 indicates active.
		63:32		Reserved.
…				

...

34.7 MSRS IN THE INTEL® ATOM™ PROCESSOR FAMILY

Table 34-4 lists model-specific registers (MSRs) for Intel Atom processor family, architectural MSR addresses are also included in Table 34-4. These processors have a CPUID signature with DisplayFamily_DisplayModel of 06_1CH, see Table 34-1.

The column "Shared/Unique" applies to logical processors sharing the same core in processors based on the Intel Atom microarchitecture. "Unique" means each logical processor has a separate MSR, or a bit field in an MSR governs only a logical processor. "Shared" means the MSR or the bit field in an MSR address governs the operation of both logical processors in the same core.

Table 34-10 MSRs Supported by Intel Processors Based on Intel Microarchitecture Code Name Sandy Bridge

Register Address		Register Name	Shared/ Unique	Bit Description
Hex	Dec			
...				
1A7H	422	MSR_OFFCORE_RSP_1	Thread	Offcore Response Event Select Register (R/W)
1AAH	426	MSR_MISC_PWR_MGMT		See http://biosbits.org.
1ADH	428	MSR_TURBO_PWR_CURRENT_LIMIT		See http://biosbits.org.
...				

...

34.8 MSRS IN THE NEXT GENERATION INTEL CORE PROCESSOR (INTEL® MICROARCHITECTURE CODE NAME IVY BRIDGE)

Next Generation Intel Core processor (Intel® microarchitecture code name Ivy Bridge) supports the MSR interfaces listed in Table 34-10 and Table 34-11.

...