



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

Documentation Changes

February 2014

Notice: The Intel[®] 64 and IA-32 architectures may contain design defects or errors known as errata that may cause the product to deviate from published specifications. Current characterized errata are documented in the specification updates.

Document Number: 252046-042



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

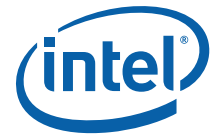
Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

Intel, the Intel logo, Pentium, Xeon, Intel NetBurst, Intel Core, Intel Core Solo, Intel Core Duo, Intel Core 2 Duo, Intel Core 2 Extreme, Intel Pentium D, Itanium, Intel SpeedStep, MMX, Intel Atom, and VTune are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copyright © 1997-2014 Intel Corporation. All rights reserved.



Contents

Revision History	4
Preface	7
Summary Tables of Changes	8
Documentation Changes	9



Revision History

Revision	Description	Date
-001	<ul style="list-style-type: none">Initial release	November 2002
-002	<ul style="list-style-type: none">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	<ul style="list-style-type: none">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	<ul style="list-style-type: none">Removed Documentation changes 1-17.Added Documentation changes 1-24.	June 2003
-005	<ul style="list-style-type: none">Removed Documentation Changes 1-24.Added Documentation Changes 1-15.	September 2003
-006	<ul style="list-style-type: none">Added Documentation Changes 16- 34.	November 2003
-007	<ul style="list-style-type: none">Updated Documentation changes 14, 16, 17, and 28.Added Documentation Changes 35-45.	January 2004
-008	<ul style="list-style-type: none">Removed Documentation Changes 1-45.Added Documentation Changes 1-5.	March 2004
-009	<ul style="list-style-type: none">Added Documentation Changes 7-27.	May 2004
-010	<ul style="list-style-type: none">Removed Documentation Changes 1-27.Added Documentation Changes 1.	August 2004
-011	<ul style="list-style-type: none">Added Documentation Changes 2-28.	November 2004
-012	<ul style="list-style-type: none">Removed Documentation Changes 1-28.Added Documentation Changes 1-16.	March 2005
-013	<ul style="list-style-type: none">Updated title.There are no Documentation Changes for this revision of the document.	July 2005
-014	<ul style="list-style-type: none">Added Documentation Changes 1-21.	September 2005
-015	<ul style="list-style-type: none">Removed Documentation Changes 1-21.Added Documentation Changes 1-20.	March 9, 2006
-016	<ul style="list-style-type: none">Added Documentation changes 21-23.	March 27, 2006
-017	<ul style="list-style-type: none">Removed Documentation Changes 1-23.Added Documentation Changes 1-36.	September 2006
-018	<ul style="list-style-type: none">Added Documentation Changes 37-42.	October 2006
-019	<ul style="list-style-type: none">Removed Documentation Changes 1-42.Added Documentation Changes 1-19.	March 2007
-020	<ul style="list-style-type: none">Added Documentation Changes 20-27.	May 2007
-021	<ul style="list-style-type: none">Removed Documentation Changes 1-27.Added Documentation Changes 1-6	November 2007
-022	<ul style="list-style-type: none">Removed Documentation Changes 1-6Added Documentation Changes 1-6	August 2008
-023	<ul style="list-style-type: none">Removed Documentation Changes 1-6Added Documentation Changes 1-21	March 2009



Revision	Description	Date
-024	<ul style="list-style-type: none"> Removed Documentation Changes 1-21 Added Documentation Changes 1-16 	June 2009
-025	<ul style="list-style-type: none"> Removed Documentation Changes 1-16 Added Documentation Changes 1-18 	September 2009
-026	<ul style="list-style-type: none"> Removed Documentation Changes 1-18 Added Documentation Changes 1-15 	December 2009
-027	<ul style="list-style-type: none"> Removed Documentation Changes 1-15 Added Documentation Changes 1-24 	March 2010
-028	<ul style="list-style-type: none"> Removed Documentation Changes 1-24 Added Documentation Changes 1-29 	June 2010
-029	<ul style="list-style-type: none"> Removed Documentation Changes 1-29 Added Documentation Changes 1-29 	September 2010
-030	<ul style="list-style-type: none"> Removed Documentation Changes 1-29 Added Documentation Changes 1-29 	January 2011
-031	<ul style="list-style-type: none"> Removed Documentation Changes 1-29 Added Documentation Changes 1-29 	April 2011
-032	<ul style="list-style-type: none"> Removed Documentation Changes 1-29 Added Documentation Changes 1-14 	May 2011
-033	<ul style="list-style-type: none"> Removed Documentation Changes 1-14 Added Documentation Changes 1-38 	October 2011
-034	<ul style="list-style-type: none"> Removed Documentation Changes 1-38 Added Documentation Changes 1-16 	December 2011
-035	<ul style="list-style-type: none"> Removed Documentation Changes 1-16 Added Documentation Changes 1-18 	March 2012
-036	<ul style="list-style-type: none"> Removed Documentation Changes 1-18 Added Documentation Changes 1-17 	May 2012
-037	<ul style="list-style-type: none"> Removed Documentation Changes 1-17 Added Documentation Changes 1-28 	August 2012
-038	<ul style="list-style-type: none"> Removed Documentation Changes 1-28 Add Documentation Changes 1-22 	January 2013
-039	<ul style="list-style-type: none"> Removed Documentation Changes 1-22 Add Documentation Changes 1-17 	June 2013
-040	<ul style="list-style-type: none"> Removed Documentation Changes 1-17 Add Documentation Changes 1-24 	September 2013
-041	<ul style="list-style-type: none"> Removed Documentation Changes 1-24 Add Documentation Changes 1-20 	February 2014
-042	<ul style="list-style-type: none"> Removed Documentation Changes 1-20 Add Documentation Changes 1-8 	February 2014

§



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
<i>Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1: Basic Architecture</i>	253665
<i>Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M</i>	253666
<i>Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z</i>	253667
<i>Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2C: Instruction Set Reference</i>	326018
<i>Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide, Part 1</i>	253668
<i>Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide, Part 2</i>	253669
<i>Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3C: System Programming Guide, Part 3</i>	326019

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 1, Volume 1
2	Updates to Chapter 2, Volume 1
3	Updates to Chapter 1, Volume 2A
4	Updates to Chapter 3, Volume 2A
5	Updates to Chapter 4, Volume 2B
6	Updates to Chapter 1, Volume 3A
7	Updates to Chapter 15, Volume 3B
8	Updates to Chapter 35, Volume 3C

Documentation Changes

1. Updates to Chapter 1, Volume 1

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

...

1.1 INTEL® 64 AND IA-32 PROCESSORS COVERED IN THIS MANUAL

This manual set includes information pertaining primarily to the most recent Intel 64 and IA-32 processors, which include:

- Pentium® processors
- P6 family processors
- Pentium® 4 processors
- Pentium® M processors
- Intel® Xeon® processors
- Pentium® D processors
- Pentium® processor Extreme Editions
- 64-bit Intel® Xeon® processors
- Intel® Core™ Duo processor
- Intel® Core™ Solo processor
- Dual-Core Intel® Xeon® processor LV
- Intel® Core™2 Duo processor
- Intel® Core™2 Quad processor Q6000 series
- Intel® Xeon® processor 3000, 3200 series
- Intel® Xeon® processor 5000 series
- Intel® Xeon® processor 5100, 5300 series
- Intel® Core™2 Extreme processor X7000 and X6800 series
- Intel® Core™2 Extreme processor QX6000 series
- Intel® Xeon® processor 7100 series
- Intel® Pentium® Dual-Core processor
- Intel® Xeon® processor 7200, 7300 series
- Intel® Xeon® processor 5200, 5400, 7400 series
- Intel® Core™2 Extreme processor QX9000 and X9000 series
- Intel® Core™2 Quad processor Q9000 series
- Intel® Core™2 Duo processor E8000, T9000 series
- Intel® Atom™ processor family
- Intel® Core™ i7 processor

- Intel® Core™ i5 processor
- Intel® Xeon® processor E7-8800/4800/2800 product families
- Intel® Core™ i7-3930K processor
- 2nd generation Intel® Core™ i7-2xxx, Intel® Core™ i5-2xxx, Intel® Core™ i3-2xxx processor series
- Intel® Xeon® processor E3-1200 product family
- Intel® Xeon® processor E5-2400/1400 product family
- Intel® Xeon® processor E5-4600/2600/1600 product family
- 3rd generation Intel® Core™ processors
- Intel® Xeon® processor E3-1200 v2 product family
- Intel® Xeon® processor E5-2400/1400 v2 product families
- Intel® Xeon® processor E5-4600/2600/1600 v2 product families
- Intel® Xeon® processor E7-8800/4800/2800 v2 product families
- 4th generation Intel® Core™ processors
- Intel® Xeon® processor E3-1200 v3 product family

P6 family processors are IA-32 processors based on the P6 family microarchitecture. This includes the Pentium® Pro, Pentium® II, Pentium® III, and Pentium® III Xeon® processors.

The Pentium® 4, Pentium® D, and Pentium® processor Extreme Editions are based on the Intel NetBurst® microarchitecture. Most early Intel® Xeon® processors are based on the Intel NetBurst® microarchitecture. Intel Xeon processor 5000, 7100 series are based on the Intel NetBurst® microarchitecture.

The Intel® Core™ Duo, Intel® Core™ Solo and dual-core Intel® Xeon® processor LV are based on an improved Pentium® M processor microarchitecture.

The Intel® Xeon® processor 3000, 3200, 5100, 5300, 7200 and 7300 series, Intel® Pentium® dual-core, Intel® Core™2 Duo, Intel® Core™2 Quad, and Intel® Core™2 Extreme processors are based on Intel® Core™ microarchitecture.

The Intel® Xeon® processor 5200, 5400, 7400 series, Intel® Core™2 Quad processor Q9000 series, and Intel® Core™2 Extreme processor QX9000, X9000 series, Intel® Core™2 processor E8000 series are based on Enhanced Intel® Core™ microarchitecture.

The Intel® Atom™ processor family is based on the Intel® Atom™ microarchitecture and supports Intel 64 architecture.

The Intel® Core™ i7 processor and Intel® Xeon® processor 3400, 5500, 7500 series are based on 45 nm Intel® microarchitecture code name Nehalem. Intel® microarchitecture code name Westmere is a 32nm version of Intel® microarchitecture code name Nehalem. Intel® Xeon® processor 5600 series, Intel Xeon processor E7 and various Intel Core i7, i5, i3 processors are based on Intel® microarchitecture code name Westmere. These processors support Intel 64 architecture.

The Intel® Xeon® processor E5 family, Intel® Xeon® processor E3-1200 family, Intel® Xeon® processor E7-8800/4800/2800 product families, Intel® Core™ i7-3930K processor, and 2nd generation Intel® Core™ i7-2xxx, Intel® Core™ i5-2xxx, Intel® Core™ i3-2xxx processor series are based on the Intel® microarchitecture code name Sandy Bridge and support Intel 64 architecture.

The Intel® Xeon® processor E7-8800/4800/2800 v2 product families, Intel® Xeon® processor E3-1200 v2 product family and the 3rd generation Intel® Core™ processors are based on the Intel® microarchitecture code name Ivy Bridge and support Intel 64 architecture.

The Intel® Xeon® processor E5-4600/2600/1600 v2 product families and Intel® Xeon® processor E5-2400/1400 v2 product families are based on the Intel® microarchitecture code name Ivy Bridge-EP and support Intel 64 architecture.

The Intel® Xeon® processor E3-1200 v3 product family and 4th Generation Intel® Core™ processors are based on the Intel® microarchitecture code name Haswell and support Intel 64 architecture.

P6 family, Pentium® M, Intel® Core™ Solo, Intel® Core™ Duo processors, dual-core Intel® Xeon® processor LV, and early generations of Pentium 4 and Intel Xeon processors support IA-32 architecture. The Intel® Atom™ processor Z5xx series support IA-32 architecture.

The Intel® Xeon® processor 3000, 3200, 5000, 5100, 5200, 5300, 5400, 7100, 7200, 7300, 7400 series, Intel® Core™2 Duo, Intel® Core™2 Extreme processors, Intel Core 2 Quad processors, Pentium® D processors, Pentium® Dual-Core processor, newer generations of Pentium 4 and Intel Xeon processor family support Intel® 64 architecture.

IA-32 architecture is the instruction set architecture and programming environment for Intel's 32-bit microprocessors. Intel® 64 architecture is the instruction set architecture and programming environment which is the superset of Intel's 32-bit and 64-bit architectures. It is compatible with the IA-32 architecture.

1.2 OVERVIEW OF VOLUME 1: BASIC ARCHITECTURE

A description of this manual's content follows:

Chapter 1 — About This Manual. Gives an overview of all five 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 — Intel® 64 and IA-32 Architectures. Introduces the Intel 64 and IA-32 architectures along with the families of Intel processors that are based on these architectures. It also gives an overview of the common features found in these processors and brief history of the Intel 64 and IA-32 architectures.

Chapter 3 — Basic Execution Environment. Introduces the models of memory organization and describes the register set used by applications.

Chapter 4 — Data Types. Describes the data types and addressing modes recognized by the processor; provides an overview of real numbers and floating-point formats and of floating-point exceptions.

Chapter 5 — Instruction Set Summary. Lists all Intel 64 and IA-32 instructions, divided into technology groups.

Chapter 6 — Procedure Calls, Interrupts, and Exceptions. Describes the procedure stack and mechanisms provided for making procedure calls and for servicing interrupts and exceptions.

Chapter 7 — Programming with General-Purpose Instructions. Describes basic load and store, program control, arithmetic, and string instructions that operate on basic data types, general-purpose and segment registers; also describes system instructions that are executed in protected mode.

Chapter 8 — Programming with the x87 FPU. Describes the x87 floating-point unit (FPU), including floating-point registers and data types; gives an overview of the floating-point instruction set and describes the processor's floating-point exception conditions.

Chapter 9 — Programming with Intel® MMX™ Technology. Describes Intel MMX technology, including MMX registers and data types; also provides an overview of the MMX instruction set.

Chapter 10 — Programming with Streaming SIMD Extensions (SSE). Describes SSE extensions, including XMM registers, the MXCSR register, and packed single-precision floating-point data types; provides an overview of the SSE instruction set and gives guidelines for writing code that accesses the SSE extensions.

Chapter 11 — Programming with Streaming SIMD Extensions 2 (SSE2). Describes SSE2 extensions, including XMM registers and packed double-precision floating-point data types; provides an overview of the SSE2 instruction set and gives guidelines for writing code that accesses SSE2 extensions. This chapter also describes SIMD floating-point exceptions that can be generated with SSE and SSE2 instructions. It also provides general guidelines for incorporating support for SSE and SSE2 extensions into operating system and applications code.

Chapter 12 — Programming with SSE3, SSSE3 and SSE4. Provides an overview of the SSE3 instruction set, Supplemental SSE3, SSE4, and guidelines for writing code that accesses these extensions.

Chapter 13 — Managing State Using the XSAVE Feature Set. Describes the XSAVE feature set instructions and explains how software can enable the XSAVE feature set and XSAVE-enabled features.

Chapter 14 — Programming with AVX, FMA and AVX2. Provides an overview of the Intel® AVX instruction set, FMA and Intel AVX2 extensions and gives guidelines for writing code that accesses these extensions.

Chapter 15 — Programming with Intel Transactional Synchronization Extensions. Describes the instruction extensions that support lock elision techniques to improve the performance of multi-threaded software with contended locks.

Chapter 16 — Input/Output. Describes the processor's I/O mechanism, including I/O port addressing, I/O instructions, and I/O protection mechanisms.

Chapter 17 — Processor Identification and Feature Determination. Describes how to determine the CPU type and features available in the processor.

Appendix A — EFLAGS Cross-Reference. Summarizes how the IA-32 instructions affect the flags in the EFLAGS register.

Appendix B — EFLAGS Condition Codes. Summarizes how conditional jump, move, and 'byte set on condition code' instructions use condition code flags (OF, CF, ZF, SF, and PF) in the EFLAGS register.

Appendix C — Floating-Point Exceptions Summary. Summarizes exceptions raised by the x87 FPU floating-point and SSE/SSE2/SSE3 floating-point instructions.

Appendix D — Guidelines for Writing x87 FPU Exception Handlers. Describes how to design and write MS-DOS* compatible exception handling facilities for FPU exceptions (includes software and hardware requirements and assembly-language code examples). This appendix also describes general techniques for writing robust FPU exception handlers.

Appendix E — Guidelines for Writing SIMD Floating-Point Exception Handlers. Gives guidelines for writing exception handlers for exceptions generated by SSE/SSE2/SSE3 floating-point instructions.

...

2. Updates to Chapter 2, Volume 1

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

...

2.1.20 The Second Generation Intel® Core™ Processor Family (2011)

The Second Generation Intel Core processor family spans Intel Core i7, i5 and i3 processors based on the Sandy Bridge microarchitecture. They are built from 32 nm process technology and have innovative features including:

- Intel Turbo Boost Technology for Intel Core i5 and i7 processors
- Intel Hyper-Threading Technology.
- Enhanced Intel Smart Cache and integrated memory controller.
- Processor graphics and built-in visual features like Intel® Quick Sync Video, Intel® Insider™ etc.
- Range of instruction set support up to AVX, AESNI, PCLMULQDQ, SSE4.2 and SSE4.1.

Intel Xeon processor E3-1200 product family is also based on the Sandy Bridge microarchitecture.

Intel Xeon processor E5-2400/1400 product families are based on the Sandy Bridge-EP microarchitecture.

Intel Xeon processor E5-4600/2600/1600 product families are based on the Sandy Bridge-EP microarchitecture and provide support for multiple sockets.

2.1.21 The Third Generation Intel® Core™ Processor Family (2012)

The Third Generation Intel Core processor family spans Intel Core i7, i5 and i3 processors based on the Ivy Bridge microarchitecture. The Intel Xeon processor E7-8800/4800/2800 v2 product families and Intel Xeon processor E3-1200 v2 product family are also based on the Ivy Bridge microarchitecture.

The Intel Xeon processor E5-2400/1400 v2 product families are based on the Ivy Bridge-EP microarchitecture.

The Intel Xeon processor E5-4600/2600/1600 v2 product families are based on the Ivy Bridge-EP microarchitecture and provide support for multiple sockets.

2.1.22 The Fourth Generation Intel® Core™ Processor Family (2013)

The Fourth Generation Intel Core processor family spans Intel Core i7, i5 and i3 processors based on the Haswell microarchitecture. Intel Xeon processor E3-1200 v3 product family is also based on the Haswell microarchitecture.

...

3. 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-M*.

...

1.1 INTEL® 64 AND IA-32 PROCESSORS COVERED IN THIS MANUAL

This manual set includes information pertaining primarily to the most recent Intel 64 and IA-32 processors, which include:

- Pentium® processors
- P6 family processors
- Pentium® 4 processors
- Pentium® M processors
- Intel® Xeon® processors
- Pentium® D processors
- Pentium® processor Extreme Editions
- 64-bit Intel® Xeon® processors
- Intel® Core™ Duo processor
- Intel® Core™ Solo processor
- Dual-Core Intel® Xeon® processor LV
- Intel® Core™2 Duo processor

- Intel® Core™2 Quad processor Q6000 series
- Intel® Xeon® processor 3000, 3200 series
- Intel® Xeon® processor 5000 series
- Intel® Xeon® processor 5100, 5300 series
- Intel® Core™2 Extreme processor X7000 and X6800 series
- Intel® Core™2 Extreme processor QX6000 series
- Intel® Xeon® processor 7100 series
- Intel® Pentium® Dual-Core processor
- Intel® Xeon® processor 7200, 7300 series
- Intel® Xeon® processor 5200, 5400, 7400 series
- Intel® Core™2 Extreme processor QX9000 and X9000 series
- Intel® Core™2 Quad processor Q9000 series
- Intel® Core™2 Duo processor E8000, T9000 series
- Intel® Atom™ processor family
- Intel® Core™ i7 processor
- Intel® Core™ i5 processor
- Intel® Xeon® processor E7-8800/4800/2800 product families
- Intel® Core™ i7-3930K processor
- 2nd generation Intel® Core™ i7-2xxx, Intel® Core™ i5-2xxx, Intel® Core™ i3-2xxx processor series
- Intel® Xeon® processor E3-1200 product family
- Intel® Xeon® processor E5-2400/1400 product family
- Intel® Xeon® processor E5-4600/2600/1600 product family
- 3rd generation Intel® Core™ processors
- Intel® Xeon® processor E3-1200 v2 product family
- Intel® Xeon® processor E5-2400/1400 v2 product families
- Intel® Xeon® processor E5-4600/2600/1600 v2 product families
- Intel® Xeon® processor E7-8800/4800/2800 v2 product families
- 4th generation Intel® Core™ processors
- Intel® Xeon® processor E3-1200 v3 product family

P6 family processors are IA-32 processors based on the P6 family microarchitecture. This includes the Pentium® Pro, Pentium® II, Pentium® III, and Pentium® III Xeon® processors.

The Pentium® 4, Pentium® D, and Pentium® processor Extreme Editions are based on the Intel NetBurst® microarchitecture. Most early Intel® Xeon® processors are based on the Intel NetBurst® microarchitecture. Intel Xeon processor 5000, 7100 series are based on the Intel NetBurst® microarchitecture.

The Intel® Core™ Duo, Intel® Core™ Solo and dual-core Intel® Xeon® processor LV are based on an improved Pentium® M processor microarchitecture.

The Intel® Xeon® processor 3000, 3200, 5100, 5300, 7200 and 7300 series, Intel® Pentium® dual-core, Intel® Core™2 Duo, Intel® Core™2 Quad, and Intel® Core™2 Extreme processors are based on Intel® Core™ microarchitecture.

The Intel® Xeon® processor 5200, 5400, 7400 series, Intel® Core™2 Quad processor Q9000 series, and Intel® Core™2 Extreme processor QX9000, X9000 series, Intel® Core™2 processor E8000 series are based on Enhanced Intel® Core™ microarchitecture.

The Intel® Atom™ processor family is based on the Intel® Atom™ microarchitecture and supports Intel 64 architecture.

The Intel® Core™ i7 processor and Intel® Xeon® processor 3400, 5500, 7500 series are based on 45 nm Intel® microarchitecture code name Nehalem. Intel® microarchitecture code name Westmere is a 32nm version of Intel® microarchitecture code name Nehalem. Intel® Xeon® processor 5600 series, Intel Xeon processor E7 and various Intel Core i7, i5, i3 processors are based on Intel® microarchitecture code name Westmere. These processors support Intel 64 architecture.

The Intel® Xeon® processor E5 family, Intel® Xeon® processor E3-1200 family, Intel® Xeon® processor E7-8800/4800/2800 product families, Intel® Core™ i7-3930K processor, and 2nd generation Intel® Core™ i7-2xxx, Intel® Core™ i5-2xxx, Intel® Core™ i3-2xxx processor series are based on the Intel® microarchitecture code name Sandy Bridge and support Intel 64 architecture.

The Intel® Xeon® processor E7-8800/4800/2800 v2 product families, Intel® Xeon® processor E3-1200 v2 product family and the 3rd generation Intel® Core™ processors are based on the Intel® microarchitecture code name Ivy Bridge and support Intel 64 architecture.

The Intel® Xeon® processor E5-4600/2600/1600 v2 product families and Intel® Xeon® processor E5-2400/1400 v2 product families are based on the Intel® microarchitecture code name Ivy Bridge-EP and support Intel 64 architecture.

The Intel® Xeon® processor E3-1200 v3 product family and 4th Generation Intel® Core™ processors are based on the Intel® microarchitecture code name Haswell and support Intel 64 architecture.

P6 family, Pentium® M, Intel® Core™ Solo, Intel® Core™ Duo processors, dual-core Intel® Xeon® processor LV, and early generations of Pentium 4 and Intel Xeon processors support IA-32 architecture. The Intel® Atom™ processor Z5xx series support IA-32 architecture.

The Intel® Xeon® processor 3000, 3200, 5000, 5100, 5200, 5300, 5400, 7100, 7200, 7300, 7400 series, Intel® Core™2 Duo, Intel® Core™2 Extreme processors, Intel Core 2 Quad processors, Pentium® D processors, Pentium® Dual-Core processor, newer generations of Pentium 4 and Intel Xeon processor family support Intel® 64 architecture.

IA-32 architecture is the instruction set architecture and programming environment for Intel's 32-bit microprocessors. Intel® 64 architecture is the instruction set architecture and programming environment which is the superset of Intel's 32-bit and 64-bit architectures. It is compatible with the IA-32 architecture.

...

4. 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-M*.

...

CPUID—CPU Identification

...

NOTE

Software must confirm that a processor feature is present using feature flags returned by CPUID prior to using the feature. Software should not depend on future offerings retaining all features.

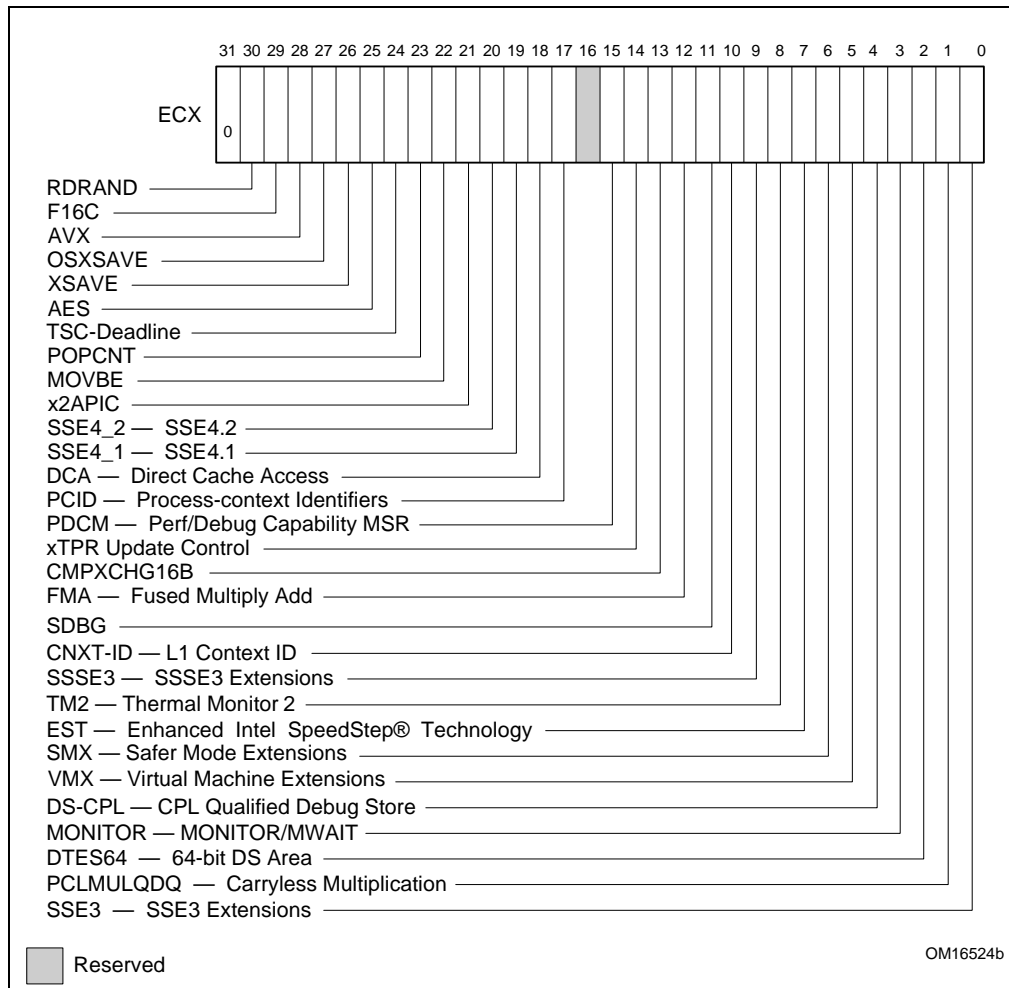


Figure 3-6 Feature Information Returned in the ECX Register

5. 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, N-Z*.

PUSH—Push Word, Doubleword or Quadword Onto the Stack

Opcode*	Instruction	Op/En	64-Bit Mode	Compat/Leg Mode	Description
FF /6	PUSH <i>r/m16</i>	M	Valid	Valid	Push <i>r/m16</i> .
FF /6	PUSH <i>r/m32</i>	M	N.E.	Valid	Push <i>r/m32</i> .
FF /6	PUSH <i>r/m64</i>	M	Valid	N.E.	Push <i>r/m64</i> .
50+ <i>rw</i>	PUSH <i>r16</i>	O	Valid	Valid	Push <i>r16</i> .
50+ <i>rd</i>	PUSH <i>r32</i>	O	N.E.	Valid	Push <i>r32</i> .
50+ <i>rd</i>	PUSH <i>r64</i>	O	Valid	N.E.	Push <i>r64</i> .
6A <i>ib</i>	PUSH <i>imm8</i>	I	Valid	Valid	Push <i>imm8</i> .
68 <i>iw</i>	PUSH <i>imm16</i>	I	Valid	Valid	Push <i>imm16</i> .
68 <i>id</i>	PUSH <i>imm32</i>	I	Valid	Valid	Push <i>imm32</i> .
0E	PUSH CS	NP	Invalid	Valid	Push CS.
16	PUSH SS	NP	Invalid	Valid	Push SS.
1E	PUSH DS	NP	Invalid	Valid	Push DS.
06	PUSH ES	NP	Invalid	Valid	Push ES.
0F A0	PUSH FS	NP	Valid	Valid	Push FS.
0F A8	PUSH GS	NP	Valid	Valid	Push GS.

NOTES:

* See IA-32 Architecture Compatibility section below.

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
M	ModRM:r/m (<i>r</i>)	NA	NA	NA
O	opcode + <i>rd</i> (<i>r</i>)	NA	NA	NA
I	<i>imm8/16/32</i>	NA	NA	NA
NP	NA	NA	NA	NA

...

XSAVES—Save Processor Extended States Supervisor

Opcode	Instruction	Op/En	64-Bit Mode	Compat/Leg Mode	Description
0F C7 /5	XSAVES <i>mem</i>	M	Valid	Valid	Save state components specified by EDX:EAX to <i>mem</i> with compaction, optimizing if possible.
REX.W+ 0F C7 /5	XSAVES64 <i>mem</i>	M	Valid	N.E.	Save state components specified by EDX:EAX to <i>mem</i> with compaction, optimizing if possible.

Instruction Operand Encoding

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

Description

Performs a full or partial save of processor state components to the XSAVE area located at the memory address specified by the destination operand. The implicit EDX:EAX register pair specifies a 64-bit instruction mask. The specific state components saved correspond to the bits set in the requested-feature bitmap (RFBM), the logical-AND of EDX:EAX and the logical-OR of XCRO with the IA32_XSS MSR. XSAVES may be executed only if CPL = 0.

The format of the XSAVE area is detailed in Section 13.4, “XSAVE Area,” of *Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 1*.

Section 13.10, “Operation of XSAVES,” of *Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 1* provides a detailed description of the operation of the XSAVES instruction. The following items provide a high-level outline:

- Execution of XSAVES is similar to that of XSAVEC. XSAVES differs from XSAVEC in that it can save state components corresponding to bits set in the IA32_XSS MSR and that it may use the modified optimization.
- XSAVES saves state component *i* only if RFBM[*i*] = 1 and XINUSE[*i*] = 1.¹ (XINUSE is a bitmap by which the processor tracks the status of various state components. See Section 13.5.4, “Processor Tracking of XSAVE-Managed State.”) Even if both bits are 1, XSAVES may optimize and not save state component *i* if (1) state component *i* has not been modified since the last execution of XRTOR or XRSTORS; and (2) this execution of XSAVES correspond to that last execution of XRTOR or XRSTORS as determined by XRSTOR_INFO (see the Operation section below).
- XSAVES does not modify bytes 511:464 of the legacy region of the XSAVE area (see Section 13.4.1, “Legacy Region of an XSAVE Area”).
- XSAVES writes the logical AND of RFBM and XINUSE to the XSTATE_BV field of the XSAVE header.² (See Section 13.4.2, “XSAVE Header.”) XSAVES sets bit 63 of the XCOMP_BV field and sets bits 62:0 of that field to RFBM[62:0]. XSAVES does not write to any parts of the XSAVE header other than the XSTATE_BV and XCOMP_BV fields.
- XSAVES always uses the compacted format of the extended region of the XSAVE area (see Section 13.4.3, “Extended Region of an XSAVE Area”).

Use of a destination operand not aligned to 64-byte boundary (in either 64-bit or 32-bit modes) results in a general-protection (#GP) exception. In 64-bit mode, the upper 32 bits of RDX and RAX are ignored.

Operation

RFBM ← XCRO AND EDX:EAX; /* bitwise logical AND */

IF in VMX non-root operation

THEN VMXNR ← 1;

ELSE VMXNR ← 0;

FI;

LAXA ← linear address of XSAVE area;

COMPMASK ← RFBM OR 80000000_00000000H;

-
1. There is an exception for state component 1 (SSE). MXCSR is part of SSE state, but XINUSE[1] may be 0 even if MXCSR does not have its initial value of 1F80H. In this case, the init optimization does not apply and XSAVEC will save SSE state as long as RFBM[1] = 1 and the modified optimization is not being applied.
 2. There is an exception for state component 1 (SSE). MXCSR is part of SSE state, but XINUSE[1] may be 0 even if MXCSR does not have its initial value of 1F80H. In this case, XSAVES sets XSTATE_BV[1] to 1 as long as RFBM[1] = 1.

```

IF XRSTOR_INFO = <CPL,VMXNR,LAXA,COMPMASK>
    THEN MODOPT ← 1;
    ELSE MODOPT ← 0;
FI;

IF RFBM[0] = 1 and XINUSE[0] = 1
    THEN store x87 state into legacy region of XSAVE area;
    /* might avoid saving if x87 state is not modified and MODOPT = 1 */
FI;
IF RFBM[1] = 1 and (XINUSE[1] = 1 or MXCSR ≠ 1F80H)
    THEN store SSE state into legacy region of XSAVE area;
    /* might avoid saving if SSE state is not modified and MODOPT = 1 */
FI;
IF RFBM[2] = 1 AND XINUSE[2] = 1
    THEN store AVX state into extended region of XSAVE area;
    /* might avoid saving if AVX state is not modified and MODOPT = 1 */
FI;

XSTATE_BV field in XSAVE header ← XINUSE AND RFBM;
XCOMP_BV field in XSAVE header ← COMPMASK;

```

Flags Affected

None.

Intel C/C++ Compiler Intrinsic Equivalent

```

XSAVES:    void _xsaves( void *, unsigned __int64);
XSAVES64:  void _xsaves64( void *, unsigned __int64);

```

Protected Mode Exceptions

#GP(0)	If a memory operand effective address is outside the CS, DS, ES, FS, or GS segment limit. If a memory operand is not aligned on a 64-byte boundary, regardless of segment.
#SS(0)	If a memory operand effective address is outside the SS segment limit.
#PF(fault-code)	If a page fault occurs.
#NM	If CR0.TS[bit 3] = 1.
#UD	If CPUID.01H: ECX.XSAVE[bit 26] = 0 or CPUID.(EAX=0DH,ECX=1):EAX.XSS[bit 3] = 0. If CR4.OSXSAVE[bit 18] = 0. If any of the LOCK, 66H, F3H or F2H prefixes is used.
#AC	If this exception is disabled a general protection exception (#GP) is signaled if the memory operand is not aligned on a 16-byte boundary, as described above. If the alignment check exception (#AC) is enabled (and the CPL is 3), signaling of #AC is not guaranteed and may vary with implementation, as follows. In all implementations where #AC is not signaled, a general protection exception is signaled in its place. In addition, the width of the alignment check may also vary with implementation. For instance, for a given implementation, an alignment check exception might be signaled for a 2-byte misalignment, whereas a general protection exception might be signaled for all other misalignments (4-, 8-, or 16-byte misalignments).

Real-Address Mode Exceptions

#GP	If a memory operand is not aligned on a 64-byte boundary, regardless of segment. If any part of the operand lies outside the effective address space from 0 to FFFFH.
#NM	If CR0.TS[bit 3] = 1.
#UD	If CPUID.01H: ECX.XSAVE[bit 26] = 0 or CPUID.(EAX=0DH,ECX=1):EAX.XSS[bit 3] = 0. If CR4.OSXSAVE[bit 18] = 0. If any of the LOCK, 66H, F3H or F2H prefixes is used.

Virtual-8086 Mode Exceptions

Same exceptions as in protected mode.

Compatibility Mode Exceptions

Same exceptions as in protected mode.

64-Bit Mode Exceptions

#GP(0)	If the memory address is in a non-canonical form. If a memory operand is not aligned on a 64-byte boundary, regardless of segment.
#SS(0)	If a memory address referencing the SS segment is in a non-canonical form.
#PF(fault-code)	If a page fault occurs.
#NM	If CR0.TS[bit 3] = 1.
#UD	If CPUID.01H: ECX.XSAVE[bit 26] = 0 or CPUID.(EAX=0DH,ECX=1):EAX.XSS[bit 3] = 0. If CR4.OSXSAVE[bit 18] = 0. If any of the LOCK, 66H, F3H or F2H prefixes is used.
#AC	If this exception is disabled a general protection exception (#GP) is signaled if the memory operand is not aligned on a 16-byte boundary, as described above. If the alignment check exception (#AC) is enabled (and the CPL is 3), signaling of #AC is not guaranteed and may vary with implementation, as follows. In all implementations where #AC is not signaled, a general protection exception is signaled in its place. In addition, the width of the alignment check may also vary with implementation. For instance, for a given implementation, an alignment check exception might be signaled for a 2-byte misalignment, whereas a general protection exception might be signaled for all other misalignments (4-, 8-, or 16-byte misalignments).

...

6. Updates to Chapter 1, Volume 3A

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

...

1.1 INTEL® 64 AND IA-32 PROCESSORS COVERED IN THIS MANUAL

This manual set includes information pertaining primarily to the most recent Intel 64 and IA-32 processors, which include:

- Pentium® processors
- P6 family processors
- Pentium® 4 processors
- Pentium® M processors
- Intel® Xeon® processors
- Pentium® D processors
- Pentium® processor Extreme Editions
- 64-bit Intel® Xeon® processors
- Intel® Core™ Duo processor
- Intel® Core™ Solo processor
- Dual-Core Intel® Xeon® processor LV
- Intel® Core™2 Duo processor
- Intel® Core™2 Quad processor Q6000 series
- Intel® Xeon® processor 3000, 3200 series
- Intel® Xeon® processor 5000 series
- Intel® Xeon® processor 5100, 5300 series
- Intel® Core™2 Extreme processor X7000 and X6800 series
- Intel® Core™2 Extreme processor QX6000 series
- Intel® Xeon® processor 7100 series
- Intel® Pentium® Dual-Core processor
- Intel® Xeon® processor 7200, 7300 series
- Intel® Xeon® processor 5200, 5400, 7400 series
- Intel® Core™2 Extreme processor QX9000 and X9000 series
- Intel® Core™2 Quad processor Q9000 series
- Intel® Core™2 Duo processor E8000, T9000 series
- Intel® Atom™ processor family
- Intel® Core™ i7 processor
- Intel® Core™ i5 processor
- Intel® Xeon® processor E7-8800/4800/2800 product families
- Intel® Core™ i7-3930K processor
- 2nd generation Intel® Core™ i7-2xxx, Intel® Core™ i5-2xxx, Intel® Core™ i3-2xxx processor series
- Intel® Xeon® processor E3-1200 product family
- Intel® Xeon® processor E5-2400/1400 product family
- Intel® Xeon® processor E5-4600/2600/1600 product family
- 3rd generation Intel® Core™ processors
- Intel® Xeon® processor E3-1200 v2 product family
- Intel® Xeon® processor E5-2400/1400 v2 product families
- Intel® Xeon® processor E5-4600/2600/1600 v2 product families
- Intel® Xeon® processor E7-8800/4800/2800 v2 product families
- 4th generation Intel® Core™ processors

- Intel® Xeon® processor E3-1200 v3 product family

P6 family processors are IA-32 processors based on the P6 family microarchitecture. This includes the Pentium® Pro, Pentium® II, Pentium® III, and Pentium® III Xeon® processors.

The Pentium® 4, Pentium® D, and Pentium® processor Extreme Editions are based on the Intel NetBurst® microarchitecture. Most early Intel® Xeon® processors are based on the Intel NetBurst® microarchitecture. Intel Xeon processor 5000, 7100 series are based on the Intel NetBurst® microarchitecture.

The Intel® Core™ Duo, Intel® Core™ Solo and dual-core Intel® Xeon® processor LV are based on an improved Pentium® M processor microarchitecture.

The Intel® Xeon® processor 3000, 3200, 5100, 5300, 7200 and 7300 series, Intel® Pentium® dual-core, Intel® Core™2 Duo, Intel® Core™2 Quad, and Intel® Core™2 Extreme processors are based on Intel® Core™ microarchitecture.

The Intel® Xeon® processor 5200, 5400, 7400 series, Intel® Core™2 Quad processor Q9000 series, and Intel® Core™2 Extreme processor QX9000, X9000 series, Intel® Core™2 processor E8000 series are based on Enhanced Intel® Core™ microarchitecture.

The Intel® Atom™ processor family is based on the Intel® Atom™ microarchitecture and supports Intel 64 architecture.

The Intel® Core™ i7 processor and Intel® Xeon® processor 3400, 5500, 7500 series are based on 45 nm Intel® microarchitecture code name Nehalem. Intel® microarchitecture code name Westmere is a 32nm version of Intel® microarchitecture code name Nehalem. Intel® Xeon® processor 5600 series, Intel Xeon processor E7 and various Intel Core i7, i5, i3 processors are based on Intel® microarchitecture code name Westmere. These processors support Intel 64 architecture.

The Intel® Xeon® processor E5 family, Intel® Xeon® processor E3-1200 family, Intel® Xeon® processor E7-8800/4800/2800 product families, Intel® Core™ i7-3930K processor, and 2nd generation Intel® Core™ i7-2xxx, Intel® Core™ i5-2xxx, Intel® Core™ i3-2xxx processor series are based on the Intel® microarchitecture code name Sandy Bridge and support Intel 64 architecture.

The Intel® Xeon® processor E7-8800/4800/2800 v2 product families, Intel® Xeon® processor E3-1200 v2 product family and the 3rd generation Intel® Core™ processors are based on the Intel® microarchitecture code name Ivy Bridge and support Intel 64 architecture.

The Intel® Xeon® processor E5-4600/2600/1600 v2 product families and Intel® Xeon® processor E5-2400/1400 v2 product families are based on the Intel® microarchitecture code name Ivy Bridge-EP and support Intel 64 architecture.

The Intel® Xeon® processor E3-1200 v3 product family and 4th Generation Intel® Core™ processors are based on the Intel® microarchitecture code name Haswell and support Intel 64 architecture.

P6 family, Pentium® M, Intel® Core™ Solo, Intel® Core™ Duo processors, dual-core Intel® Xeon® processor LV, and early generations of Pentium 4 and Intel Xeon processors support IA-32 architecture. The Intel® Atom™ processor Z5xx series support IA-32 architecture.

The Intel® Xeon® processor 3000, 3200, 5000, 5100, 5200, 5300, 5400, 7100, 7200, 7300, 7400 series, Intel® Core™2 Duo, Intel® Core™2 Extreme processors, Intel Core 2 Quad processors, Pentium® D processors, Pentium® Dual-Core processor, newer generations of Pentium 4 and Intel Xeon processor family support Intel® 64 architecture.

IA-32 architecture is the instruction set architecture and programming environment for Intel's 32-bit microprocessors. Intel® 64 architecture is the instruction set architecture and programming environment which is the superset of Intel's 32-bit and 64-bit architectures. It is compatible with the IA-32 architecture.

...

7. Updates to Chapter 15, Volume 3B

Change bars show changes to Chapter 15 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B: System Programming Guide, Part 2*.

...

15.3.1 Machine-Check Global Control MSRs

The machine-check global control MSRs include the IA32_MCG_CAP, IA32_MCG_STATUS, and optionally IA32_MCG_CTL and IA32_MCG_EXT_CTL. See Chapter 35, "Model-Specific Registers (MSRs)," for the addresses of these registers.

15.3.1.1 IA32_MCG_CAP MSR

The IA32_MCG_CAP MSR is a read-only register that provides information about the machine-check architecture of the processor. Figure 15-2 shows the layout of the register.

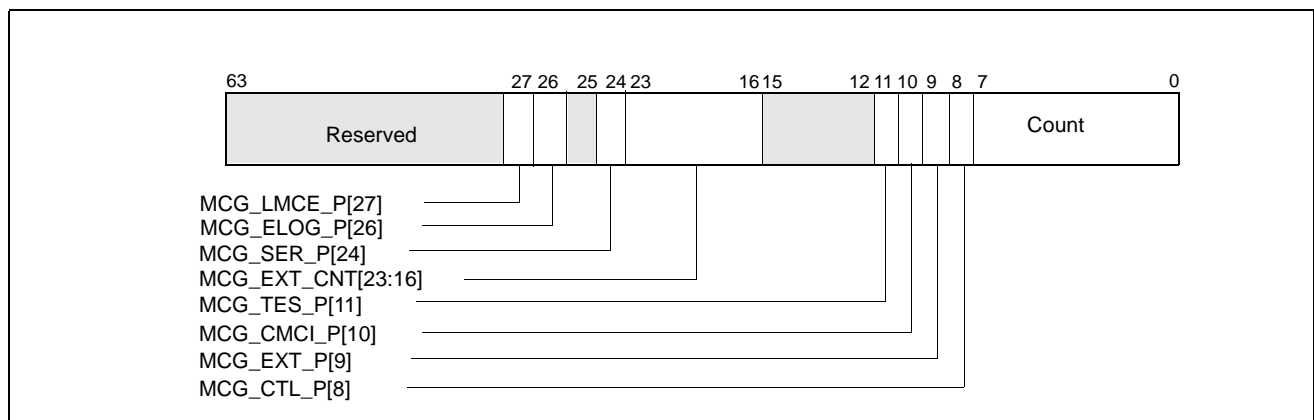


Figure 15-2 IA32_MCG_CAP Register

Where:

- **Count field, bits 7:0** — Indicates the number of hardware unit error-reporting banks available in a particular processor implementation.
- **MCG_CTL_P (control MSR present) flag, bit 8** — Indicates that the processor implements the IA32_MCG_CTL MSR when set; this register is absent when clear.
- **MCG_EXT_P (extended MSRs present) flag, bit 9** — Indicates that the processor implements the extended machine-check state registers found starting at MSR address 180H; these registers are absent when clear.
- **MCG_CMCI_P (Corrected MC error counting/signaling extension present) flag, bit 10** — Indicates (when set) that extended state and associated MSRs necessary to support the reporting of an interrupt on a corrected MC error event and/or count threshold of corrected MC errors, is present. When this bit is set, it does not imply this feature is supported across all banks. Software should check the availability of the necessary logic on a bank by bank basis when using this signaling capability (i.e. bit 30 settable in individual IA32_MCI_CTL2 register).

- **MCG_TES_P (threshold-based error status present) flag, bit 11** — Indicates (when set) that bits 56:53 of the IA32_MCi_STATUS MSR are part of the architectural space. Bits 56:55 are reserved, and bits 54:53 are used to report threshold-based error status. Note that when MCG_TES_P is not set, bits 56:53 of the IA32_MCi_STATUS MSR are model-specific.
- **MCG_EXT_CNT, bits 23:16** — Indicates the number of extended machine-check state registers present. This field is meaningful only when the MCG_EXT_P flag is set.
- **MCG_SER_P (software error recovery support present) flag, bit 24**— Indicates (when set) that the processor supports software error recovery (see Section 15.6), and IA32_MCi_STATUS MSR bits 56:55 are used to report the signaling of uncorrected recoverable errors and whether software must take recovery actions for uncorrected errors. Note that when MCG_TES_P is not set, bits 56:53 of the IA32_MCi_STATUS MSR are model-specific. If MCG_TES_P is set but MCG_SER_P is not set, bits 56:55 are reserved.
- **MCG_ELOG_P (extended error logging) flag, bit 26** — Indicates (when set) that the processor allows platform firmware to be invoked when an error is detected so that it may provide additional platform specific information in an ACPI format “Generic Error Data Entry” that augments the data included in machine check bank registers.

For additional information about extended error logging interface, see <http://www.intel.com/content/www/us/en/architecture-and-technology/enhanced-mca-logging-xeon-paper.html>

- **MCG_LMCE_P (local machine check exception) flag, bit 27** — Indicates (when set) that the following interfaces are present:
 - an extended state LMCE_S (located in bit 3 of IA32_MCG_STATUS), and
 - the IA32_MCG_EXT_CTL MSR, necessary to support Local Machine Check Exception (LMCE).

A non-zero MCG_LMCE_P indicates that, when LMCE is enabled as described in Section 15.3.1.5, some machine check errors may be delivered to only a single logical processor.

The effect of writing to the IA32_MCG_CAP MSR is undefined.

15.3.1.2 IA32_MCG_STATUS MSR

The IA32_MCG_STATUS MSR describes the current state of the processor after a machine-check exception has occurred (see Figure Figure 15-3).

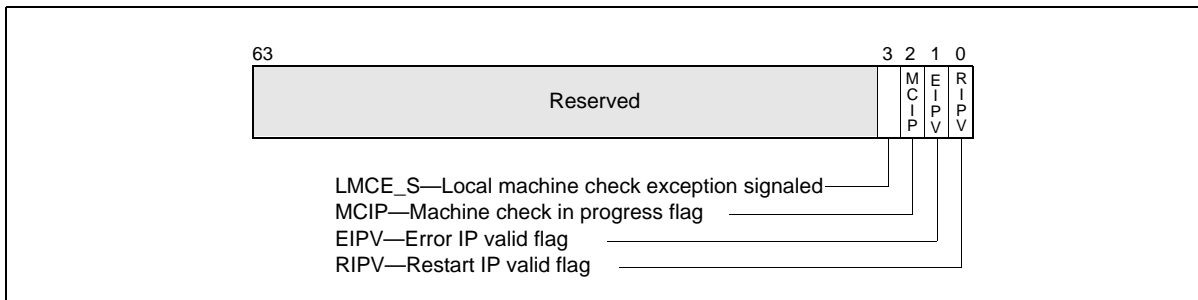


Figure 15-3 IA32_MCG_STATUS Register

Where:

- **RIPV (restart IP valid) flag, bit 0** — Indicates (when set) that program execution can be restarted reliably at the instruction pointed to by the instruction pointer pushed on the stack when the machine-check exception is generated. When clear, the program cannot be reliably restarted at the pushed instruction pointer.

- **EIPV (error IP valid) flag, bit 1** — Indicates (when set) that the instruction pointed to by the instruction pointer pushed onto the stack when the machine-check exception is generated is directly associated with the error. When this flag is cleared, the instruction pointed to may not be associated with the error.
- **MCIP (machine check in progress) flag, bit 2** — Indicates (when set) that a machine-check exception was generated. Software can set or clear this flag. The occurrence of a second Machine-Check Event while MCIP is set will cause the processor to enter a shutdown state. For information on processor behavior in the shutdown state, please refer to the description in Chapter 6, “Interrupt and Exception Handling”: “Interrupt 8—Double Fault Exception (#DF)”.
- **LMCE_S (local machine check exception signaled), bit 3** — Indicates (when set) that a local machine-check exception was generated. This indicates that the current machine-check event was delivered to only this logical processor.

Bits 63:04 in IA32_MCG_STATUS are reserved. An attempt to write to IA32_MCG_STATUS with any value other than 0 would result in #GP.

...

15.3.1.4 IA32_MCG_EXT_CTL MSR

The IA32_MCG_EXT_CTL MSR is present if the capability flag MCG_LMCE_P is set in the IA32_MCG_CAP MSR.

IA32_MCG_EXT_CTL.LMCE_EN (bit 0) allows the processor to signal some MCEs to only a single logical processor in the system.

If MCG_LMCE_P is not set in IA32_MCG_CAP, or platform software has not enabled LMCE by setting IA32_FEATURE_CONTROL.LMCE_ON (bit 20), any attempt to write or read IA32_MCG_EXT_CTL will result in #GP.

The IA32_MCG_EXT_CTL MSR is cleared on RESET.

Figure Figure 15-4 shows the layout of the IA32_MCG_EXT_CTL register

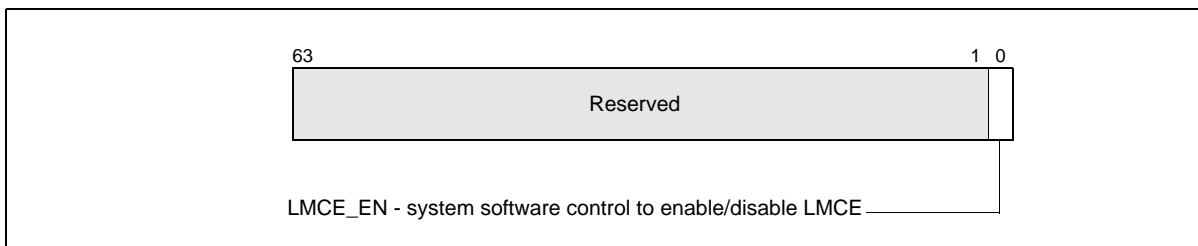


Figure 15-4 IA32_MCG_EXT_CTL Register

where

- **LMCE_EN (local machine check exception enable) flag, bit 0** - System software sets this to allow hardware to signal some MCEs to only a single logical processor. System software can set LMCE_EN only if the platform software has configured IA32_FEATURE_CONTROL as described in Section 15.3.1.5.

15.3.1.5 Enabling Local Machine Check

The intended usage of LMCE requires proper configuration by both platform software and system software. Platform software can turn LMCE on by setting bit 20 (LMCE_ON) in IA32_FEATURE_CONTROL MSR (MSR address 3AH).

System software must ensure that both IA32_FEATURE_CONTROL.Lock (bit 0) and IA32_FEATURE_CONTROL.LMCE_ON (bit 20) are set before attempting to set IA32_MCG_EXT_CTL.LMCE_EN (bit

0). When system software has enabled LMCE, then hardware will determine if a particular error can be delivered only to a single logical processor. Software should make no assumptions about the type of error that hardware can choose to deliver as LMCE. The severity and override rules stay the same as described in Table 15-7 to determine the recovery actions.

...

15.3.2.2 IA32_MCi_STATUS MSRS

Each IA32_MCi_STATUS MSR contains information related to a machine-check error if its VAL (valid) flag is set (see Figure Figure 15-6). Software is responsible for clearing IA32_MCi_STATUS MSRs by explicitly writing 0s to them; writing 1s to them causes a general-protection exception.

NOTE

Figure Figure 15-6 depicts the IA32_MCi_STATUS MSR when IA32_MCG_CAP[24] = 1, IA32_MCG_CAP[11] = 1 and IA32_MCG_CAP[10] = 1. When IA32_MCG_CAP[24] = 0 and IA32_MCG_CAP[11] = 1, bits 56:55 is reserved and bits 54:53 for threshold-based error reporting. When IA32_MCG_CAP[11] = 0, bits 56:53 are part of the “Other Information” field. The use of bits 54:53 for threshold-based error reporting began with Intel Core Duo processors, and is currently used for cache memory. See Section 15.4, “Enhanced Cache Error reporting,” for more information. When IA32_MCG_CAP[10] = 0, bits 52:38 are part of the “Other Information” field. The use of bits 52:38 for corrected MC error count is introduced with Intel 64 processor on which CPUID reports DisplayFamily_DisplayModel as 06H_1AH.

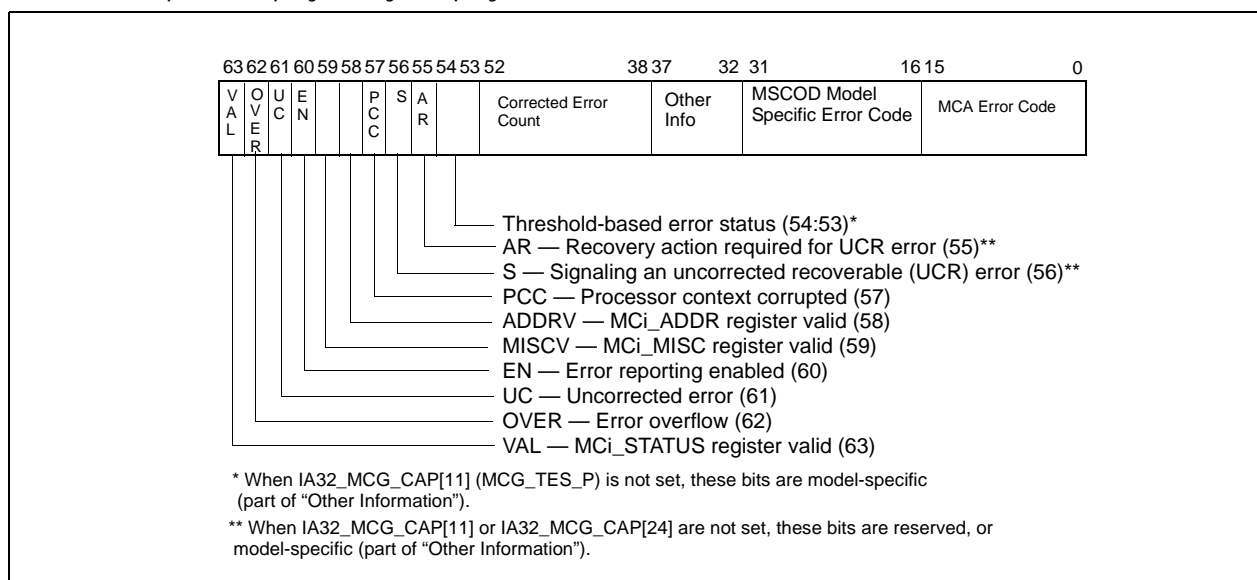


Figure 15-6 IA32_MCi_STATUS Register

Where:

- MCA (machine-check architecture) error code field, bits 15:0** — Specifies the machine-check architecture-defined error code for the machine-check error condition detected. The machine-check architecture-defined error codes are guaranteed to be the same for all IA-32 processors that implement the machine-check architecture. See Section 15.9, “Interpreting the MCA Error Codes,” and Chapter 16, “Interpreting Machine-Check Error Codes”, for information on machine-check error codes.

- **Model-specific error code field, bits 31:16** — Specifies the model-specific error code that uniquely identifies the machine-check error condition detected. The model-specific error codes may differ among IA-32 processors for the same machine-check error condition. See Chapter 16, “Interpreting Machine-Check Error Codes” for information on model-specific error codes.
- **Reserved, Error Status, and Other Information fields, bits 56:32** —
 - Bits **37:32** always contain “Other Information” that is implementation-specific and is not part of the machine-check architecture. Software that is intended to be portable among IA-32 processors should not rely on these values.
 - If IA32_MCG_CAP[10] is 0, bits **52:38** also contain “Other Information” (in the same sense as bits 37:32).
 - If IA32_MCG_CAP[10] is 1, bits **52:38** are architectural (not model-specific). In this case, bits 52:38 reports the value of a 15 bit counter that increments each time a corrected error is observed by the MCA recording bank. This count value will continue to increment until cleared by software. The most significant bit, 52, is a sticky count overflow bit.
 - If IA32_MCG_CAP[11] is 0, bits **56:53** also contain “Other Information” (in the same sense).
 - If IA32_MCG_CAP[11] is 1, bits **56:53** are architectural (not model-specific). In this case, bits 56:53 have the following functionality:
 - If IA32_MCG_CAP[24] is 0, bits 56:55 are reserved.
 - If IA32_MCG_CAP[24] is 1, bits 56:55 are defined as follows:
 - S (Signaling) flag, bit 56 - Signals the reporting of UCR errors in this MC bank. See Section 15.6.2 for additional detail.
 - AR (Action Required) flag, bit 55 - Indicates (when set) that MCA error code specific recovery action must be performed by system software at the time this error was signaled. See Section 15.6.2 for additional detail.
 - If the UC bit (Figure Figure 15-6) is 1, bits 54:53 are undefined.
 - If the UC bit (Figure Figure 15-6) is 0, bits 54:53 indicate the status of the hardware structure that reported the threshold-based error. See Table Table 15-1.

Table 15-1 Bits 54:53 in IA32_MCi_STATUS MSRs when IA32_MCG_CAP[11] = 1 and UC = 0

Bits 54:53	Meaning
00	No tracking - No hardware status tracking is provided for the structure reporting this event.
01	Green - Status tracking is provided for the structure posting the event; the current status is green (below threshold). For more information, see Section 15.4, “Enhanced Cache Error reporting”.
10	Yellow - Status tracking is provided for the structure posting the event; the current status is yellow (above threshold). For more information, see Section 15.4, “Enhanced Cache Error reporting”.
11	Reserved

- **PCC (processor context corrupt) flag, bit 57** — Indicates (when set) that the state of the processor might have been corrupted by the error condition detected and that reliable restarting of the processor may not be possible. When clear, this flag indicates that the error did not affect the processor’s state. Software restarting might be possible.
- **ADDRV (IA32_MCi_ADDR register valid) flag, bit 58** — Indicates (when set) that the IA32_MCi_ADDR register contains the address where the error occurred (see Section 15.3.2.3, “IA32_MCi_ADDR MSRs”). When clear, this flag indicates that the IA32_MCi_ADDR register is either not implemented or does not contain the address where the error occurred. Do not read these registers if they are not implemented in the processor.

- **MISCV (IA32_MCi_MISC register valid) flag, bit 59** — Indicates (when set) that the IA32_MCi_MISC register contains additional information regarding the error. When clear, this flag indicates that the IA32_MCi_MISC register is either not implemented or does not contain additional information regarding the error. Do not read these registers if they are not implemented in the processor.
- **EN (error enabled) flag, bit 60** — Indicates (when set) that the error was enabled by the associated EEj bit of the IA32_MCi_CTL register.
- **UC (error uncorrected) flag, bit 61** — Indicates (when set) that the processor did not or was not able to correct the error condition. When clear, this flag indicates that the processor was able to correct the error condition.
- **OVER (machine check overflow) flag, bit 62** — Indicates (when set) that a machine-check error occurred while the results of a previous error were still in the error-reporting register bank (that is, the VAL bit was already set in the IA32_MCi_STATUS register). The processor sets the OVER flag and software is responsible for clearing it. In general, enabled errors are written over disabled errors, and uncorrected errors are written over corrected errors. Uncorrected errors are not written over previous valid uncorrected errors. For more information, see Section 15.3.2.2.1, “Overwrite Rules for Machine Check Overflow”.
- **VAL (IA32_MCi_STATUS register valid) flag, bit 63** — Indicates (when set) that the information within the IA32_MCi_STATUS register is valid. When this flag is set, the processor follows the rules given for the OVER flag in the IA32_MCi_STATUS register when overwriting previously valid entries. The processor sets the VAL flag and software is responsible for clearing it.

...

15.3.2.5 IA32_MCi_CTL2 MSRs

The IA32_MCi_CTL2 MSR provides the programming interface to use corrected MC error signaling capability that is indicated by IA32_MCG_CAP[10] = 1. Software must check for the presence of IA32_MCi_CTL2 on a per-bank basis.

When IA32_MCG_CAP[10] = 1, the IA32_MCi_CTL2 MSR for each bank exists, i.e. reads and writes to these MSR are supported. However, signaling interface for corrected MC errors may not be supported in all banks.

The layout of IA32_MCi_CTL2 is shown in Figure Figure 15-9:

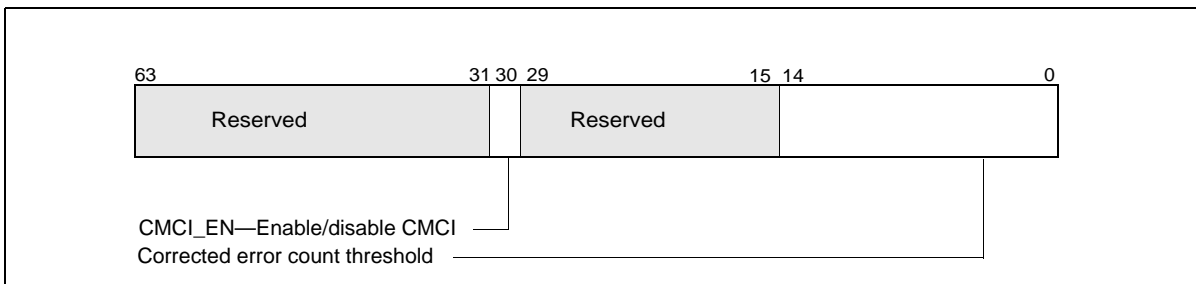


Figure 15-9 IA32_MCi_CTL2 Register

- **Corrected error count threshold, bits 14:0** — Software must initialize this field. The value is compared with the corrected error count field in IA32_MCi_STATUS, bits 38 through 52. An overflow event is signaled to the CMCI LVT entry (see Table 10-1) in the APIC when the count value equals the threshold value. The new LVT entry in the APIC is at 02F0H offset from the APIC_BASE. If CMCI interface is not supported for a particular bank (but IA32_MCG_CAP[10] = 1), this field will always read 0.
- **CMCI_EN (Corrected error interrupt enable/disable/indicator), bits 30** — Software sets this bit to enable the generation of corrected machine-check error interrupt (CMCI). If CMCI interface is not supported

for a particular bank (but IA32_MCG_CAP[10] = 1), this bit is writeable but will always return 0 for that bank. This bit also indicates CMCI is supported or not supported in the corresponding bank. See Section 15.5 for details of software detection of CMCI facility.

Some microarchitectural sub-systems that are the source of corrected MC errors may be shared by more than one logical processors. Consequently, the facilities for reporting MC errors and controlling mechanisms may be shared by more than one logical processors. For example, the IA32_MCI_CTL2 MSR is shared between logical processors sharing a processor core. Software is responsible to program IA32_MCI_CTL2 MSR in a consistent manner with CMCI delivery and usage.

After processor reset, IA32_MCI_CTL2 MSRs are zero'ed.

...

15.8 MACHINE-CHECK INITIALIZATION

To use the processors machine-check architecture, software must initialize the processor to activate the machine-check exception and the error-reporting mechanism.

Example Example 15-1 gives pseudocode for performing this initialization. This pseudocode checks for the existence of the machine-check architecture and exception; it then enables machine-check exception and the error-reporting register banks. The pseudocode shown is compatible with the Pentium 4, Intel Xeon, Intel Atom, P6 family, and Pentium processors.

Following power up or power cycling, IA32_MCI_STATUS registers are not guaranteed to have valid data until after they are initially cleared to zero by software (as shown in the initialization pseudocode in Example Example 15-1). In addition, when using P6 family processors, software must set MCI_STATUS registers to zero when doing a soft-reset.

Example 15-1 Machine-Check Initialization Pseudocode

Check CPUID Feature Flags for MCE and MCA support

IF CPU supports MCE

THEN

IF CPU supports MCA

THEN

IF (IA32_MCG_CAP.MCG_CTL_P = 1)

(* IA32_MCG_CTL register is present *)

THEN

IA32_MCG_CTL ← FFFFFFFFFFFFFFFFH;

(* enables all MCA features *)

FI

IF (IA32_MCG_CAP.MCG_LMCE_P = 1 and IA32_FEATURE_CONTROL.LOCK = 1 and IA32_FEATURE_CONTROL.LMCE_ON = 1)

(* IA32_MCG_EXT_CTL register is present and platform has enabled LMCE to permit system software to use LMCE *)

THEN

IA32_MCG_EXT_CTL ← IA32_MCG_EXT_CTL | 01H;

(* System software enables LMCE capability for hardware to signal MCE to a single logical processor*)

FI

(* Determine number of error-reporting banks supported *)

COUNT ← IA32_MCG_CAP.Count;

MAX_BANK_NUMBER ← COUNT - 1;

IF (Processor Family is 6H and Processor EXTMODEL.MODEL is less than 1AH)

THEN

(* Enable logging of all errors except for MCO_CTL register *)

FOR error-reporting banks (1 through MAX_BANK_NUMBER)

DO

```

        IA32_MCI_CTL ← 0FFFFFFFFFFFFFFFH;
    OD

ELSE
    (* Enable logging of all errors including MCO_CTL register *)
    FOR error-reporting banks (0 through MAX_BANK_NUMBER)
    DO
        IA32_MCI_CTL ← 0FFFFFFFFFFFFFFFH;
    OD
FI

(* BIOS clears all errors only on power-on reset *)
IF (BIOS detects Power-on reset)
THEN
    FOR error-reporting banks (0 through MAX_BANK_NUMBER)
    DO
        IA32_MCI_STATUS ← 0;
    OD
ELSE
    FOR error-reporting banks (0 through MAX_BANK_NUMBER)
    DO
        (Optional for BIOS and OS) Log valid errors
        (OS only) IA32_MCI_STATUS ← 0;
    OD
FI
FI

Setup the Machine Check Exception (#MC) handler for vector 18 in IDT

Set the MCE bit (bit 6) in CR4 register to enable Machine-Check Exceptions
FI
...

```

15.9.1 Simple Error Codes

Table Table 15-8 shows the simple error codes. These unique codes indicate global error information.

Table 15-8 IA32_MCI_Status [15:0] Simple Error Code Encoding

Error Code	Binary Encoding	Meaning
No Error	0000 0000 0000 0000	No error has been reported to this bank of error-reporting registers.
Unclassified	0000 0000 0000 0001	This error has not been classified into the MCA error classes.
Microcode ROM Parity Error	0000 0000 0000 0010	Parity error in internal microcode ROM
External Error	0000 0000 0000 0011	The BINIT# from another processor caused this processor to enter machine check. ¹
FRC Error	0000 0000 0000 0100	FRC (functional redundancy check) master/slave error
Internal Parity Error	0000 0000 0000 0101	Internal parity error.
SMM Handler Code Access Violation	0000 0000 0000 0110	An attempt was made by the SMM Handler to execute outside the ranges specified by SMRR.
Internal Timer Error	0000 0100 0000 0000	Internal timer error.
I/O Error	0000 1110 0000 1011	generic I/O error.

Table 15-8 IA32_MCi_Status [15:0] Simple Error Code Encoding (Contd.)

Internal Unclassified	0000 01xx xxxx xxxx	Internal unclassified errors. ²
-----------------------	---------------------	--

NOTES:

1. BINIT# assertion will cause a machine check exception if the processor (or any processor on the same external bus) has BINIT# observation enabled during power-on configuration (hardware strapping) and if machine check exceptions are enabled (by setting CR4.MCE = 1).
2. At least one X must equal one. Internal unclassified errors have not been classified.

...

15.9.3.1 Architecturally Defined SRAO Errors

The following two SRAO errors are architecturally defined.

- UCR Errors detected by memory controller scrubbing; and
- UCR Errors detected during L3 cache (L3) explicit writebacks.

The MCA error code encodings for these two architecturally-defined UCR errors corresponds to sub-classes of compound MCA error codes (see Table 15-9). Their values and compound encoding format are given in Table Table 15-15.

Table 15-15 MCA Compound Error Code Encoding for SRAO Errors

Type	MCACOD Value	MCA Error Code Encoding ¹
Memory Scrubbing	0xC0 - 0xCF	0000_0000_1100_CCCC 000F 0000 1MMM CCCC (Memory Controller Error), where Memory subfield MMM = 100B (memory scrubbing) Channel subfield CCCC = channel # or generic
L3 Explicit Writeback	0x17A	0000_0001_0111_1010 000F 0001 RRRR TTLL (Cache Hierarchy Error) where Request subfields RRRR = 0111B (Eviction) Transaction Type subfields TT = 10B (Generic) Level subfields LL = 10B

NOTES:

1. Note that for both of these errors the correction report filtering (F) bit (bit 12) of the MCA error is 0, indicating “normal” filtering.

Table Table 15-16 lists values of relevant bit fields of IA32_MCi_STATUS for architecturally defined SRAO errors.

Table 15-16 IA32_MCi_STATUS Values for SRAO Errors

SRAO Error	Valid	OVER	UC	EN	MISCV	ADDRV	PCC	S	AR	MCACOD
Memory Scrubbing	1	0	1	1	1	1	0	1	0	0xC0-0xCF
L3 Explicit Writeback	1	0	1	1	1	1	0	1	0	0x17A

For both the memory scrubbing and L3 explicit writeback errors, the ADDRv and MISCV flags in the IA32_MCi_STATUS register are set to indicate that the offending physical address information is available from the IA32_MCi_MISC and the IA32_MCi_ADDR registers. For the memory scrubbing and L3 explicit writeback errors, the address mode in the IA32_MCi_MISC register should be set as physical address mode (010b) and the address LSB information in the IA32_MCi_MISC register should indicate the lowest valid address bit in the address information provided from the IA32_MCi_ADDR register.

MCE signal is broadcast to all logical processors as outlined in Section 15.10.4.1. If LMCE is supported and enabled, some errors (not limited to UCR errors) may be delivered to only a single logical processor. System software should consult IA32_MCG_STATUS.LMCE_S to determine if the MCE signaled is only to this logical processor.

IA32_MCi_STATUS banks can be shared by logical processors within a core or within the same package. So several logical processors may find an SRAO error in the shared IA32_MCi_STATUS bank but other processors do not find it in any of the IA32_MCi_STATUS banks. Table Table 15-17 shows the RIPV and EIPV flag indication in the IA32_MCG_STATUS register for the memory scrubbing and L3 explicit writeback errors on both the reporting and non-reporting logical processors.

Table 15-17 IA32_MCG_STATUS Flag Indication for SRAO Errors

SRAO Type	Reporting Logical Processors		Non-reporting Logical Processors	
	RIPV	EIPV	RIPV	EIPV
Memory Scrubbing	1	0	1	0
L3 Explicit Writeback	1	0	1	0

15.9.3.2 Architecturally Defined SRAR Errors

The following two SRAR errors are architecturally defined.

- UCR Errors detected on data load; and
- UCR Errors detected on instruction fetch.

The MCA error code encodings for these two architecturally-defined UCR errors corresponds to sub-classes of compound MCA error codes (see Table 15-9). Their values and compound encoding format are given in Table Table 15-18.

Table 15-18 MCA Compound Error Code Encoding for SRAR Errors

Type	MCACOD Value	MCA Error Code Encoding ¹
Data Load	0x134	0000_0001_0011_0100 000F 0001 RRRR TTLL (Cache Hierarchy Error), where Request subfield RRRR = 0011B (Data Load) Transaction Type subfield TT= 01B (Data) Level subfield LL = 00B (Level 0)
Instruction Fetch	0x150	0000_0001_0101_0000 000F 0001 RRRR TTLL (Cache Hierarchy Error), where Request subfield RRRR = 0101B (Instruction Fetch) Transaction Type subfield TT= 00B (Instruction) Level subfield LL = 00B (Level 0)

NOTES:

1. Note that for both of these errors the correction report filtering (F) bit (bit 12) of the MCA error is 0, indicating “normal” filtering.

Table Table 15-19 lists values of relevant bit fields of IA32_MCi_STATUS for architecturally defined SRAR errors.

Table 15-19 IA32_MCi_STATUS Values for SRAR Errors

SRAR Error	Valid	OVER	UC	EN	MISCV	ADDRV	PCC	S	AR	MCACOD
Data Load	1	0	1	1	1	1	0	1	1	0x134
Instruction Fetch	1	0	1	1	1	1	0	1	1	0x150

For both the data load and instruction fetch errors, the ADDR_V and MISC_V flags in the IA32_MCI_STATUS register are set to indicate that the offending physical address information is available from the IA32_MCI_MISC and the IA32_MCI_ADDR registers. For the memory scrubbing and L3 explicit writeback errors, the address mode in the IA32_MCI_MISC register should be set as physical address mode (010b) and the address LSB information in the IA32_MCI_MISC register should indicate the lowest valid address bit in the address information provided from the IA32_MCI_ADDR register.

MCE signal is broadcast to all logical processors on the system on which the UCR errors are supported, except when the processor supports LMCE and LMCE is enabled by system software (see Section 15.3.1.5). The IA32_MCG_STATUS MSR allows system software to distinguish the affected logical processor of an SRAR error amongst logical processors that observed SRAR via MCI_STATUS bank.

...

15.9.4 Multiple MCA Errors

When multiple MCA errors are detected within a certain detection window, the processor may aggregate the reporting of these errors together as a single event, i.e. a single machine exception condition. If this occurs, system software may find multiple MCA errors logged in different MC banks on one logical processor or find multiple MCA errors logged across different processors for a single machine check broadcast event. In order to handle multiple UCR errors reported from a single machine check event and possibly recover from multiple errors, system software may consider the following:

- Whether it can recover from multiple errors is determined by the most severe error reported on the system. If the most severe error is found to be an unrecoverable error (VAL=1, UC=1, PCC=1 and EN=1) after system software examines the MC banks of all processors to which the MCA signal is broadcast, recovery from the multiple errors is not possible and system software needs to reset the system.
- When multiple recoverable errors are reported and no other fatal condition (e.g., overflowed condition for SRAR error) is found for the reported recoverable errors, it is possible for system software to recover from the multiple recoverable errors by taking necessary recovery action for each individual recoverable error. However, system software can no longer expect one to one relationship with the error information recorded in the IA32_MCI_STATUS register and the states of the RIPV and EIPV flags in the IA32_MCG_STATUS register as the states of the RIPV and the EIPV flags in the IA32_MCG_STATUS register may indicate the information for the most severe error recorded on the processor. System software is required to use the RIPV flag indication in the IA32_MCG_STATUS register to make a final decision of recoverability of the errors and find the restart-ability requirement after examining each IA32_MCI_STATUS register error information in the MC banks.

In certain cases where system software observes more than one SRAR error logged for a single logical processor, it can no longer rely on affected threads as specified in Table 15-20 above. System software is recommended to reset the system if this condition is observed.

...

15.10.4.1 Machine-Check Exception Handler for Error Recovery

When writing a machine-check exception (MCE) handler to support software recovery from Uncorrected Recoverable (UCR) errors, consider the following:

- When IA32_MCG_CAP [24] is zero, there are no recoverable errors supported and all machine-check are fatal exceptions. The logging of status and error information is therefore a baseline implementation requirement.
- When IA32_MCG_CAP [24] is 1, certain uncorrected errors called uncorrected recoverable (UCR) errors may be software recoverable. The handler can analyze the reported error information, and in some cases attempt to recover from the uncorrected error and continue execution.

- For processors on which CPUID reports DisplayFamily_DisplayModel as 06H_0EH and onward, an MCA signal is broadcast to all logical processors in the system (see CPUID instruction in Chapter 3, “Instruction Set Reference, A-M” in the *Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 2A*). Due to the potentially shared machine check MSR resources among the logical processors on the same package/core, the MCE handler may be required to synchronize with the other processors that received a machine check error and serialize access to the machine check registers when analyzing, logging and clearing the information in the machine check registers.
 - On processors that indicate ability for local machine-check exception (MCG_LMCE_P), hardware can choose to report the error to only a single logical processor if system software has enabled LMCE by setting IA32_MCG_EXT_CTL[LMCE_EN] = 1 as outlined in Section 15.3.1.5.
- The VAL (valid) flag in each IA32_MCi_STATUS register indicates whether the error information in the register is valid. If this flag is clear, the registers in that bank do not contain valid error information and should not be checked.
- The MCE handler is primarily responsible for processing uncorrected errors. The UC flag in each IA32_MCi_Status register indicates whether the reported error was corrected (UC=0) or uncorrected (UC=1). The MCE handler can optionally log and clear the corrected errors in the MC banks if it can implement software algorithm to avoid the undesired race conditions with the CMCI or CMC polling handler.
- For uncorrectable errors, the EIPV flag in the IA32_MCG_STATUS register indicates (when set) that the instruction pointed to by the instruction pointer pushed onto the stack when the machine-check exception is generated is directly associated with the error. When this flag is cleared, the instruction pointed to may not be associated with the error.
- The MCIP flag in the IA32_MCG_STATUS register indicates whether a machine-check exception was generated. When a machine check exception is generated, it is expected that the MCIP flag in the IA32_MCG_STATUS register is set to 1. If it is not set, this machine check was generated by either an INT 18 instruction or some piece of hardware signaling an interrupt with vector 18.

When IA32_MCG_CAP [24] is 1, the following rules can apply when writing a machine check exception (MCE) handler to support software recovery:

- The PCC flag in each IA32_MCi_STATUS register indicates whether recovery from the error is possible for uncorrected errors (UC=1). If the PCC flag is set for uncorrected errors (UC=1), recovery is not possible. When recovery is not possible, the MCE handler typically records the error information and signals the operating system to reset the system.
- The RIPV flag in the IA32_MCG_STATUS register indicates whether restarting the program execution from the instruction pointer saved on the stack for the machine check exception is possible. When the RIPV is set, program execution can be restarted reliably when recovery is possible. If the RIPV flag is not set, program execution cannot be restarted reliably. In this case the recovery algorithm may involve terminating the current program execution and resuming an alternate thread of execution upon return from the machine check handler when recovery is possible. When recovery is not possible, the MCE handler signals the operating system to reset the system.
- When the EN flag is zero but the VAL and UC flags are one in the IA32_MCi_STATUS register, the reported uncorrected error in this bank is not enabled. As uncorrected errors with the EN flag = 0 are not the source of machine check exceptions, the MCE handler should log and clear non-enabled errors when the S bit is set and should continue searching for enabled errors from the other IA32_MCi_STATUS registers. Note that when IA32_MCG_CAP [24] is 0, any uncorrected error condition (VAL =1 and UC=1) including the one with the EN flag cleared are fatal and the handler must signal the operating system to reset the system. For the errors that do not generate machine check exceptions, the EN flag has no meaning. See Chapter 19: Table 19-15 to find the errors that do not generate machine check exceptions.
- When the VAL flag is one, the UC flag is one, the EN flag is one and the PCC flag is zero in the IA32_MCi_STATUS register, the error in this bank is an uncorrected recoverable (UCR) error. The MCE handler needs to examine the S flag and the AR flag to find the type of the UCR error for software recovery and determine if software error recovery is possible.

- When both the S and the AR flags are clear in the IA32_MCi_STATUS register for the UCR error (VAL=1, UC=1, EN=x and PCC=0), the error in this bank is an uncorrected no-action required error (UCNA). UCNA errors are uncorrected but do not require any OS recovery action to continue execution. These errors indicate that some data in the system is corrupt, but that data has not been consumed and may not be consumed. If that data is consumed a non-UNCA machine check exception will be generated. UCNA errors are signaled in the same way as corrected machine check errors and the CMCI and CMC polling handler is primarily responsible for handling UCNA errors. Like corrected errors, the MCA handler can optionally log and clear UCNA errors as long as it can avoid the undesired race condition with the CMCI or CMC polling handler. As UCNA errors are not the source of machine check exceptions, the MCA handler should continue searching for uncorrected or software recoverable errors in all other MC banks.
- When the S flag in the IA32_MCi_STATUS register is set for the UCR error ((VAL=1, UC=1, EN=1 and PCC=0), the error in this bank is software recoverable and it was signaled through a machine-check exception. The AR flag in the IA32_MCi_STATUS register further clarifies the type of the software recoverable errors.
- When the AR flag in the IA32_MCi_STATUS register is clear for the software recoverable error (VAL=1, UC=1, EN=1, PCC=0 and S=1), the error in this bank is a software recoverable action optional (SRAO) error. The MCE handler and the operating system can analyze the IA32_MCi_STATUS [15:0] to implement MCA error code specific optional recovery action, but this recovery action is optional. System software can resume the program execution from the instruction pointer saved on the stack for the machine check exception when the RIPV flag in the IA32_MCG_STATUS register is set.
- When the OVER flag in the IA32_MCi_STATUS register is set for the SRAO error (VAL=1, UC=1, EN=1, PCC=0, S=1 and AR=0), the MCE handler cannot take recovery action as the information of the SRAO error in the IA32_MCi_STATUS register was potentially lost due to the overflow condition. Since the recovery action for SRAO errors is optional, restarting the program execution from the instruction pointer saved on the stack for the machine check exception is still possible for the overflowed SRAO error if the RIPV flag in the IA32_MCG_STATUS is set.
- When the AR flag in the IA32_MCi_STATUS register is set for the software recoverable error (VAL=1, UC=1, EN=1, PCC=0 and S=1), the error in this bank is a software recoverable action required (SRAR) error. The MCE handler and the operating system must take recovery action in order to continue execution after the machine-check exception. The MCA handler and the operating system need to analyze the IA32_MCi_STATUS [15:0] to determine the MCA error code specific recovery action. If no recovery action can be performed, the operating system must reset the system.
- When the OVER flag in the IA32_MCi_STATUS register is set for the SRAR error (VAL=1, UC=1, EN=1, PCC=0, S=1 and AR=1), the MCE handler cannot take recovery action as the information of the SRAR error in the IA32_MCi_STATUS register was potentially lost due to the overflow condition. Since the recovery action for SRAR errors must be taken, the MCE handler must signal the operating system to reset the system.
- When the MCE handler cannot find any uncorrected (VAL=1, UC=1 and EN=1) or any software recoverable errors (VAL=1, UC=1, EN=1, PCC=0 and S=1) in any of the IA32_MCi banks of the processors, this is an unexpected condition for the MCE handler and the handler should signal the operating system to reset the system.
- Before returning from the machine-check exception handler, software must clear the MCIP flag in the IA32_MCG_STATUS register. The MCIP flag is used to detect recursion. The machine-check architecture does not support recursion. When the processor receives a machine check when MCIP is set, it automatically enters the shutdown state.

Example Example 15-4 gives pseudocode for an MC exception handler that supports recovery of UCR.

Example 15-4 Machine-Check Error Handler Pseudocode Supporting UCR

```

MACHINE CHECK HANDLER: (* Called from INT 18 handler *)
NOERROR = TRUE;
ProcessorCount = 0;
IF CPU supports MCA
    THEN

```

```

RESTARTABILITY = TRUE;
IF (Processor Family = 6 AND DisplayModel ≥ 0EH) OR (Processor Family > 6)
  THEN
    IF ( MCG_LMCE = 1)
      MCA_BROADCAST = FALSE;
    ELSE
      MCA_BROADCAST = TRUE;
    FI;
    Acquire SpinLock;
    ProcessorCount++; (* Allowing one logical processor at a time to examine machine check registers *)
    CALL MCA ERROR PROCESSING; (* returns RESTARTABILITY and NOERROR *)
  ELSE
    MCA_BROADCAST = FALSE;
    (* Implement a rendezvous mechanism with the other processors if necessary *)
    CALL MCA ERROR PROCESSING;
  FI;
ELSE (* Pentium(R) processor compatible *)
  READ P5_MC_ADDR
  READ P5_MC_TYPE;
  RESTARTABILITY = FALSE;
FI;

IF NOERROR = TRUE
  THEN
    IF NOT (MCG_RIPV = 1 AND MCG_EIPV = 0)
      THEN
        RESTARTABILITY = FALSE;
      FI
    FI;

IF RESTARTABILITY = FALSE
  THEN
    Report RESTARTABILITY to console;
    Reset system;
  FI;

IF MCA_BROADCAST = TRUE
  THEN
    IF ProcessorCount = MAX_PROCESSORS
      AND NOERROR = TRUE
      THEN
        Report RESTARTABILITY to console;
        Reset system;
      FI;
    Release SpinLock;
    Wait till ProcessorCount = MAX_PROCESSORS on system;
    (* implement a timeout and abort function if necessary *)
  FI;
CLEAR IA32_MCG_STATUS;
RESUME Execution;
(* End of MACHINE CHECK HANDLER*)

MCA ERROR PROCESSING: (* MCA Error Processing Routine called from MCA Handler *)
IF MCIP flag in IA32_MCG_STATUS = 0
  THEN (* MCIP=0 upon MCA is unexpected *)
    RESTARTABILITY = FALSE;
  FI;
FOR each bank of machine-check registers
  DO

```

```

CLEAR_MC_BANK = FALSE;
READ IA32_MCI_STATUS;
IF VAL Flag in IA32_MCI_STATUS = 1
  THEN
    IF UC Flag in IA32_MCI_STATUS = 1
      THEN
        IF Bit 24 in IA32_MCG_CAP = 0
          THEN (* the processor does not support software error recovery *)
            RESTARTABILITY = FALSE;
            NOERROR = FALSE;
            GOTO LOG MCA REGISTER;
        FI;
        (* the processor supports software error recovery *)
        IF EN Flag in IA32_MCI_STATUS = 0 AND OVER Flag in IA32_MCI_STATUS=0
          THEN (* It is a spurious MCA Log. Log and clear the register *)
            CLEAR_MC_BANK = TRUE;
            GOTO LOG MCA REGISTER;
        FI;
        IF PCC Flag in IA32_MCI_STATUS = 1
          THEN (* processor context might have been corrupted *)
            RESTARTABILITY = FALSE;
          ELSE (* It is a uncorrected recoverable (UCR) error *)
            IF S Flag in IA32_MCI_STATUS = 0
              THEN
                IF AR Flag in IA32_MCI_STATUS = 0
                  THEN (* It is a uncorrected no action required (UCNA) error *)
                    GOTO CONTINUE; (* let CMCI and CMC polling handler to process *)
                ELSE
                  RESTARTABILITY = FALSE; (* S=0, AR=1 is illegal *)
                FI
              FI;
            FI;
            IF RESTARTABILITY = FALSE
              THEN (* no need to take recovery action if RESTARTABILITY is already false *)
                NOERROR = FALSE;
                GOTO LOG MCA REGISTER;
            FI;
            (* S in IA32_MCI_STATUS = 1 *)
            IF AR Flag in IA32_MCI_STATUS = 1
              THEN (* It is a software recoverable and action required (SRAR) error *)
                IF OVER Flag in IA32_MCI_STATUS = 1
                  THEN
                    RESTARTABILITY = FALSE;
                    NOERROR = FALSE;
                    GOTO LOG MCA REGISTER;
                FI
                IF MCACOD Value in IA32_MCI_STATUS is recognized
                  AND Current Processor is an Affected Processor
                  THEN
                    Implement MCACOD specific recovery action;
                    CLEAR_MC_BANK = TRUE;
                ELSE
                    RESTARTABILITY = FALSE;
                FI;
            FI;
            ELSE (* It is a software recoverable and action optional (SRAO) error *)
              IF OVER Flag in IA32_MCI_STATUS = 0 AND
                MCACOD in IA32_MCI_STATUS is recognized
                THEN
                  Implement MCACOD specific recovery action;
                FI;
              CLEAR_MC_BANK = TRUE;
            FI;
          FI;
        FI;
      FI;
    FI;
  FI;

```

```

                FI; AR
                FI; PCC
                NOERROR = FALSE;
                GOTO LOG MCA REGISTER;
            ELSE (* It is a corrected error; continue to the next IA32_MCI_STATUS *)
                GOTO CONTINUE;
        FI; UC
    FI; VAL
LOG MCA REGISTER:
    SAVE IA32_MCI_STATUS;
    If MISCV in IA32_MCI_STATUS
        THEN
            SAVE IA32_MCI_MISC;
        FI;
    IF ADDR in IA32_MCI_STATUS
        THEN
            SAVE IA32_MCI_ADDR;
        FI;
    IF CLEAR_MC_BANK = TRUE
        THEN
            SET all 0 to IA32_MCI_STATUS;
            If MISCV in IA32_MCI_STATUS
                THEN
                    SET all 0 to IA32_MCI_MISC;
                FI;
            IF ADDR in IA32_MCI_STATUS
                THEN
                    SET all 0 to IA32_MCI_ADDR;
                FI;
        FI;
    CONTINUE:
    OD;
(*END FOR *)
RETURN;
(* End of MCA ERROR PROCESSING*)

```

...

8. Updates to Chapter 35, Volume 3C

Change bars show changes to Chapter 35 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3C: System Programming Guide, Part 3*.

...

This chapter lists MSR addresses across Intel processor families. All MSR addresses 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 their 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-M" in the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A*). Table 35-1 lists the signature values of DisplayFamily and DisplayModel for various processor families or processor number series.

Table 35-1 CPUID Signature Values of DisplayFamily_DisplayModel

DisplayFamily_DisplayModel	Processor Families/Processor Number Series
06_4EH	Future Generation Intel Core Processor
06_3DH	Next Generation Intel Core Processor
06_3FH	Next Generation Intel Xeon Processor
06_3CH, 06_45H, 06_46H	4th Generation Intel Core Processor and Intel Xeon Processor E3-1200 v3 Product Family based on Haswell microarchitecture.
06_3EH	Intel Xeon Processor E7-8800 v2/E7-4800 v2/E7-2800 v2 Family based on Ivy Bridge-EP microarchitecture
06_3EH	Intel Xeon Processor E5-1600 v2/E5-2400 v2/E5-2600 v2 Product Families based on Ivy Bridge-EP microarchitecture, Intel Core i7-49xx Processor Extreme Edition
06_3AH	3rd Generation Intel Core Processor and Intel Xeon Processor E3-1200 v2 Product Family based on Ivy Bridge microarchitecture.
06_2DH	Intel Xeon Processor E5 Family based on Intel microarchitecture code name Sandy Bridge, Intel Core i7-39xx Processor Extreme Edition
06_2FH	Intel Xeon Processor E7 Family
06_2AH	Intel Xeon Processor E3-1200 Product Family; 2nd 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
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_4AH, 06_5AH, 06_5DH	Future Intel Atom Processor Based on Silvermont Microarchitecture
06_37H	Intel Atom Processor E3000 series
06_4DH	Intel Atom Processor C2000 series
06_36H	Intel Atom Processor S1000 Series
06_1CH, 06_26H, 06_27H, 06_35H, 06_36H	Intel Atom Processor family, Intel Atom processor D2000, N2000, E2000, Z2000, C1000 series
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

Table 35-1 CPUID Signature (Contd.)Values of DisplayFamily_DisplayModel (Contd.)

DisplayFamily_DisplayModel	Processor Families/Processor Number Series
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

35.1 ARCHITECTURAL MSRS

Many MSRs have carried over from one generation of IA-32 processors to the next and to Intel 64 processors. A subset of MSRs and associated bit fields, which do not change on future processor generations, are now considered architectural MSRs. For historical reasons (beginning with the Pentium 4 processor), these “architectural MSRs” were given the prefix “IA32_”. Table Table 35-2 lists the architectural MSRs, their addresses, their current names, their names in previous IA-32 processors, and bit fields that are considered architectural. MSR addresses outside Table Table 35-2 and certain bitfields in an MSR address that may overlap with architectural MSR addresses are model-specific. Code that accesses a machine specified MSR and that is executed on a processor that does not support that MSR will generate an exception.

Architectural MSR or individual bit fields in an architectural MSR may be introduced or transitioned at the granularity of certain processor family/model or the presence of certain CPUID feature flags. The right-most column of Table Table 35-2 provides information on the introduction of each architectural MSR or its individual fields. This information is expressed either as signature values of “DF_DM” (see Table Table 35-1) or via CPUID flags.

Certain bit field position may be related to the maximum physical address width, the value of which is expressed as “MAXPHYWID” in Table Table 35-2. “MAXPHYWID” is reported by CPUID.8000_0008H leaf.

MSR address range between 40000000H - 400000FFH is marked as a specially reserved range. All existing and future processors will not implement any features using any MSR in this range.

Table 35-2 IA-32 Architectural MSRs

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
0H	0	IA32_P5_MC_ADDR (P5_MC_ADDR)	See Section 35.18, “MSRs in Pentium Processors.”	Pentium Processor (05_01H)
1H	1	IA32_P5_MC_TYPE (P5_MC_TYPE)	See Section 35.18, “MSRs in Pentium Processors.”	DF_DM = 05_01H
6H	6	IA32_MONITOR_FILTER_SIZE	See Section 8.10.5, “Monitor/Mwait Address Range Determination.”	0F_03H
10H	16	IA32_TIME_STAMP_COUNTER (TSC)	See Section 17.13, “Time-Stamp Counter.”	05_01H
17H	23	IA32_PLATFORM_ID (MSR_PLATFORM_ID)	Platform ID (R0) The operating system can use this MSR to determine “slot” information for the processor and the proper microcode update to load.	06_01H

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		49:0	Reserved.	
		52:50	Platform Id (RO) Contains information concerning the intended platform for the processor. 52 51 50 0 0 0 Processor Flag 0 0 0 1 Processor Flag 1 0 1 0 Processor Flag 2 0 1 1 Processor Flag 3 1 0 0 Processor Flag 4 1 0 1 Processor Flag 5 1 1 0 Processor Flag 6 1 1 1 Processor Flag 7	
		63:53	Reserved.	
1BH	27	IA32_APIC_BASE (APIC_BASE)		06_01H
		7:0	Reserved	
		8	BSP flag (R/W)	
		9	Reserved	
		10	Enable x2APIC mode	06_1AH
		11	APIC Global Enable (R/W)	
		(MAXPHYWID - 1):12	APIC Base (R/W)	
		63: MAXPHYWID	Reserved	
3AH	58	IA32_FEATURE_CONTROL	Control Features in Intel 64 Processor (R/W)	If CPUID.01H: ECX[bit 5 or bit 6] = 1
		0	Lock bit (R/WO): (1 = locked). When set, locks this MSR from being written, writes to this bit will result in GP(0). Note: Once the Lock bit is set, the contents of this register cannot be modified. Therefore the lock bit must be set after configuring support	If CPUID.01H:ECX[bit 5 or bit 6] = 1
			for Intel Virtualization Technology and prior to transferring control to an option ROM or the OS. Hence, once the Lock bit is set, the entire IA32_FEATURE_CONTROL_MSR contents are preserved across RESET when PWRGOOD is not deasserted.	

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		1	Enable VMX inside SMX operation (R/WL): This bit enables a system executive to use VMX in conjunction with SMX to support Intel® Trusted Execution Technology. BIOS must set this bit only when the CPUID function 1 returns VMX feature flag and SMX feature flag set (ECX bits 5 and 6 respectively).	If CPUID.01H:ECX[bit 5 and bit 6] are set to 1
		2	Enable VMX outside SMX operation (R/WL): This bit enables VMX for system executive that do not require SMX. BIOS must set this bit only when the CPUID function 1 returns VMX feature flag set (ECX bit 5).	If CPUID.01H:ECX[bit 5 or bit 6] = 1
		7:3	Reserved	
		14:8	SENTER Local Function Enables (R/WL): When set, each bit in the field represents an enable control for a corresponding SENTER function. This bit is supported only if CPUID.1:ECX.[bit 6] is set	If CPUID.01H:ECX[bit 6] = 1
		15	SENTER Global Enable (R/WL): This bit must be set to enable SENTER leaf functions. This bit is supported only if CPUID.1:ECX.[bit 6] is set	If CPUID.01H:ECX[bit 6] = 1
		19:16	Reserved	
		20	LMCE On (R/WL): When set, system software can program the MSRs associated with LMCE to configure delivery of some machine check exceptions to a single logical processor.	
		63:21	Reserved	
3BH	59	IA32_TSC_ADJUST	Per Logical Processor TSC Adjust (R/Write to clear)	If CPUID.(EAX=07H, ECX=0H): EBX[1] = 1
		63:0	THREAD_ADJUST: Local offset value of the IA32_TSC for a logical processor. Reset value is Zero. A write to IA32_TSC will modify the local offset in IA32_TSC_ADJUST and the content of IA32_TSC, but does not affect the internal invariant TSC hardware.	

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
79H	121	IA32_BIOS_UPDT_TRIG (BIOS_UPDT_TRIG)	<p>BIOS Update Trigger (W)</p> <p>Executing a WRMSR instruction to this MSR causes a microcode update to be loaded into the processor. See Section 9.11.6, "Microcode Update Loader."</p> <p>A processor may prevent writing to this MSR when loading guest states on VM entries or saving guest states on VM exits.</p>	06_01H
8BH	139	IA32_BIOS_SIGN_ID (BIOS_SIGN/ BBL_CR_D3)	<p>BIOS Update Signature (RO)</p> <p>Returns the microcode update signature following the execution of CPUID.01H.</p> <p>A processor may prevent writing to this MSR when loading guest states on VM entries or saving guest states on VM exits.</p>	06_01H
		31:0	Reserved	
		63:32	<p>It is recommended that this field be pre-loaded with 0 prior to executing CPUID.</p> <p>If the field remains 0 following the execution of CPUID; this indicates that no microcode update is loaded. Any non-zero value is the microcode update signature.</p>	
9BH	155	IA32_SMM_MONITOR_CTL	SMM Monitor Configuration (R/W)	If CPUID.01H: ECX[bit 5 or bit 6] = 1
		0	Valid (R/W)	
		1	Reserved	
		2	Controls SMI unblocking by VMXOFF (see Section 34.14.4)	If IA32_VMX_MISC[bit 28]
		11:3	Reserved	
		31:12	MSEG Base (R/W)	
		63:32	Reserved	
9EH	158	IA32_SMBASE	Base address of the logical processor's SMRAM image (RO, SMM only)	If IA32_VMX_MISC[bit 15]
C1H	193	IA32_PMC0 (PERFCTR0)	General Performance Counter 0 (R/W)	If CPUID.0AH: EAX[15:8] > 0
C2H	194	IA32_PMC1 (PERFCTR1)	General Performance Counter 1 (R/W)	If CPUID.0AH: EAX[15:8] > 1
C3H	195	IA32_PMC2	General Performance Counter 2 (R/W)	If CPUID.0AH: EAX[15:8] > 2
C4H	196	IA32_PMC3	General Performance Counter 3 (R/W)	If CPUID.0AH: EAX[15:8] > 3

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
C5H	197	IA32_PMC4	General Performance Counter 4 (R/W)	If CPUID.0AH: EAX[15:8] > 4
C6H	198	IA32_PMC5	General Performance Counter 5 (R/W)	If CPUID.0AH: EAX[15:8] > 5
C7H	199	IA32_PMC6	General Performance Counter 6 (R/W)	If CPUID.0AH: EAX[15:8] > 6
C8H	200	IA32_PMC7	General Performance Counter 7 (R/W)	If CPUID.0AH: EAX[15:8] > 7
E7H	231	IA32_MPERF	Maximum Qualified Performance Clock Counter (R/Write to clear)	If CPUID.06H: ECX[0] = 1
		63:0	CO_MCNT: C0 Maximum Frequency Clock Count Increments at fixed interval (relative to TSC freq.) when the logical processor is in C0. Cleared upon overflow / wrap-around of IA32_APERF.	
E8H	232	IA32_APERF	Actual Performance Clock Counter (R/Write to clear)	If CPUID.06H: ECX[0] = 1
		63:0	CO_ACNT: C0 Actual Frequency Clock Count Accumulates core clock counts at the coordinated clock frequency, when the logical processor is in C0. Cleared upon overflow / wrap-around of IA32_MPERF.	
FEH	254	IA32_MTRRCAP (MTRRcap)	MTRR Capability (RO) Section 11.11.2.1, "IA32_MTRR_DEF_TYPE MSR."	06_01H
		7:0	VCNT: The number of variable memory type ranges in the processor.	
		8	Fixed range MTRRs are supported when set.	
		9	Reserved.	
		10	WC Supported when set.	
		11	SMRR Supported when set.	
		63:12	Reserved.	
174H	372	IA32_SYSENTER_CS	SYSENTER_CS_MSR (R/W)	06_01H
		15:0	CS Selector	
		63:16	Reserved.	
175H	373	IA32_SYSENTER_ESP	SYSENTER_ESP_MSR (R/W)	06_01H

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
176H	374	IA32_SYSENTER_EIP	SYSENTER_EIP_MSR (R/W)	06_01H
179H	377	IA32_MCG_CAP (MCG_CAP)	Global Machine Check Capability (RO)	06_01H
		7:0	Count: Number of reporting banks.	
		8	MCG_CTL_P: IA32_MCG_CTL is present if this bit is set	
		9	MCG_EXT_P: Extended machine check state registers are present if this bit is set	
		10	MCP_CMCI_P: Support for corrected MC error event is present.	06_1AH
		11	MCG_TES_P: Threshold-based error status register are present if this bit is set.	
		15:12	Reserved	
		23:16	MCG_EXT_CNT: Number of extended machine check state registers present.	
		24	MCG_SER_P: The processor supports software error recovery if this bit is set.	
		25	Reserved.	
		26	MCG_ELOG_P: Indicates that the processor allows platform firmware to be invoked when an error is detected so that it may provide additional platform specific information in an ACPI format "Generic Error Data Entry" that augments the data included in machine check bank registers.	06_3EH
		27	MCG_LMCE_P: Indicates that the processor support extended state in IA32_MCG_STATUS and associated MSR necessary to configure Local Machine Check Exception (LMCE).	06_3EH
		63:28	Reserved.	
17AH	378	IA32_MCG_STATUS (MCG_STATUS)	Global Machine Check Status (R/W0)	06_01H
		0	RIPV. Restart IP valid	06_01H
		1	EIPV. Error IP valid	06_01H
		2	MCIP. Machine check in progress	06_01H
		3	LMCE_S.	If IA32_MCG_CAP.LMCE_P =1
		63:4	Reserved.	
17BH	379	IA32_MCG_CTL (MCG_CTL)	Global Machine Check Control (R/W)	06_01H
180H-185H	384-389	Reserved		06_0EH ¹

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
186H	390	IA32_PERFEVTSELO (PERFEVTSELO)	Performance Event Select Register 0 (R/W)	If CPUID.0AH: EAX[15:8] > 0
		7:0	Event Select: Selects a performance event logic unit.	
		15:8	UMask: Qualifies the microarchitectural condition to detect on the selected event logic.	
		16	USR: Counts while in privilege level is not ring 0.	
		17	OS: Counts while in privilege level is ring 0.	
		18	Edge: Enables edge detection if set.	
		19	PC: enables pin control.	
		20	INT: enables interrupt on counter overflow.	
		21	AnyThread: When set to 1, it enables counting the associated event conditions occurring across all logical processors sharing a processor core. When set to 0, the counter only increments the associated event conditions occurring in the logical processor which programmed the MSR.	
		22	EN: enables the corresponding performance counter to commence counting when this bit is set.	
		23	INV: invert the CMASK.	
		31:24	CMASK: When CMASK is not zero, the corresponding performance counter increments each cycle if the event count is greater than or equal to the CMASK.	
63:32	Reserved.			
187H	391	IA32_PERFEVTSEL1 (PERFEVTSEL1)	Performance Event Select Register 1 (R/W)	If CPUID.0AH: EAX[15:8] > 1
188H	392	IA32_PERFEVTSEL2	Performance Event Select Register 2 (R/W)	If CPUID.0AH: EAX[15:8] > 2
189H	393	IA32_PERFEVTSEL3	Performance Event Select Register 3 (R/W)	If CPUID.0AH: EAX[15:8] > 3
18AH-197H	394-407	Reserved		06_0EH ²
198H	408	IA32_PERF_STATUS	(RO)	0F_03H
		15:0	Current performance State Value	
		63:16	Reserved.	

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
199H	409	IA32_PERF_CTL	(R/W)	0F_03H
		15:0	Target performance State Value	
		31:16	Reserved.	
		32	IDA Engage. (R/W) When set to 1: disengages IDA	06_0FH (Mobile)
		63:33	Reserved.	
19AH	410	IA32_CLOCK_MODULATION	Clock Modulation Control (R/W) See Section 14.7.3, "Software Controlled Clock Modulation."	0F_0H
		0	Extended On-Demand Clock Modulation Duty Cycle:	If CPUID.06H:EAX[5] = 1
		3:1	On-Demand Clock Modulation Duty Cycle: Specific encoded values for target duty cycle modulation.	
		4	On-Demand Clock Modulation Enable: Set 1 to enable modulation.	
		63:5	Reserved.	
19BH	411	IA32_THERM_INTERRUPT	Thermal Interrupt Control (R/W) Enables and disables the generation of an interrupt on temperature transitions detected with the processor's thermal sensors and thermal monitor. See Section 14.7.2, "Thermal Monitor."	0F_0H
		0	High-Temperature Interrupt Enable	
		1	Low-Temperature Interrupt Enable	
		2	PROCHOT# Interrupt Enable	
		3	FORCEPR# Interrupt Enable	
		4	Critical Temperature Interrupt Enable	
		7:5	Reserved.	
		14:8	Threshold #1 Value	
		15	Threshold #1 Interrupt Enable	
		22:16	Threshold #2 Value	
		23	Threshold #2 Interrupt Enable	
		24	Power Limit Notification Enable	If CPUID.06H:EAX[4] = 1
		63:25	Reserved.	

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
19CH	412	IA32_THERM_STATUS	Thermal Status Information (RO) Contains status information about the processor's thermal sensor and automatic thermal monitoring facilities. See Section 14.7.2, "Thermal Monitor"	OF_OH
		0	Thermal Status (RO):	
		1	Thermal Status Log (R/W):	
		2	PROCHOT # or FORCEPR# event (RO)	
		3	PROCHOT # or FORCEPR# log (R/WCO)	
		4	Critical Temperature Status (RO)	
		5	Critical Temperature Status log (R/WCO)	
		6	Thermal Threshold #1 Status (RO)	If CPUID.01H:ECX[8] = 1
		7	Thermal Threshold #1 log (R/WCO)	If CPUID.01H:ECX[8] = 1
		8	Thermal Threshold #2 Status (RO)	If CPUID.01H:ECX[8] = 1
		9	Thermal Threshold #1 log (R/WCO)	If CPUID.01H:ECX[8] = 1
		10	Power Limitation Status (RO)	If CPUID.06H:EAX[4] = 1
		11	Power Limitation log (R/WCO)	If CPUID.06H:EAX[4] = 1
		15:12	Reserved.	
		22:16	Digital Readout (RO)	If CPUID.06H:EAX[0] = 1
		26:23	Reserved.	
		30:27	Resolution in Degrees Celsius (RO)	If CPUID.06H:EAX[0] = 1
		31	Reading Valid (RO)	If CPUID.06H:EAX[0] = 1
63:32	Reserved.			
1A0H	416	IA32_MISC_ENABLE	Enable Misc. Processor Features (R/W) Allows a variety of processor functions to be enabled and disabled.	
		0	Fast-Strings Enable When set, the fast-strings feature (for REP MOVS and REP STORS) is enabled (default); when clear, fast-strings are disabled.	OF_OH
		2:1	Reserved.	

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		3	Automatic Thermal Control Circuit Enable (R/W) 1 = Setting this bit enables the thermal control circuit (TCC) portion of the Intel Thermal Monitor feature. This allows the processor to automatically reduce power consumption in response to TCC activation. 0 = Disabled (default). Note: In some products clearing this bit might be ignored in critical thermal conditions, and TM1, TM2 and adaptive thermal throttling will still be activated.	0F_0H
		6:4	Reserved	
		7	Performance Monitoring Available (R) 1 = Performance monitoring enabled 0 = Performance monitoring disabled	0F_0H
		10:8	Reserved.	
		11	Branch Trace Storage Unavailable (RO) 1 = Processor doesn't support branch trace storage (BTS) 0 = BTS is supported	0F_0H
		12	Precise Event Based Sampling (PEBS) Unavailable (RO) 1 = PEBS is not supported; 0 = PEBS is supported.	06_0FH
		15:13	Reserved.	
		16	Enhanced Intel SpeedStep Technology Enable (R/W) 0 = Enhanced Intel SpeedStep Technology disabled 1 = Enhanced Intel SpeedStep Technology enabled	06_0DH
		17	Reserved.	

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		18	<p>ENABLE MONITOR FSM (R/w)</p> <p>When this bit is set to 0, the MONITOR feature flag is not set (CPUID.01H:ECX[bit 3] = 0). This indicates that MONITOR/MWAIT are not supported.</p> <p>Software attempts to execute MONITOR/MWAIT will cause #UD when this bit is 0.</p> <p>When this bit is set to 1 (default), MONITOR/MWAIT are supported (CPUID.01H:ECX[bit 3] = 1).</p> <p>If the SSE3 feature flag ECX[0] is not set (CPUID.01H:ECX[bit 0] = 0), the OS must not attempt to alter this bit. BIOS must leave it in the default state. Writing this bit when the SSE3 feature flag is set to 0 may generate a #GP exception.</p>	0F_03H
		21:19	Reserved.	
		22	<p>Limit CPUID Maxval (R/w)</p> <p>When this bit is set to 1, CPUID.00H returns a maximum value in EAX[7:0] of 3.</p> <p>BIOS should contain a setup question that allows users to specify when the installed OS does not support CPUID functions greater than 3.</p> <p>Before setting this bit, BIOS must execute the CPUID.0H and examine the maximum value returned in EAX[7:0]. If the maximum value is greater than 3, the bit is supported.</p> <p>Otherwise, the bit is not supported. Writing to this bit when the maximum value is greater than 3 may generate a #GP exception.</p> <p>Setting this bit may cause unexpected behavior in software that depends on the availability of CPUID leaves greater than 3.</p>	0F_03H
		23	<p>xTPR Message Disable (R/w)</p> <p>When set to 1, xTPR messages are disabled. xTPR messages are optional messages that allow the processor to inform the chipset of its priority.</p>	if CPUID.01H:ECX[14] = 1
		33:24	Reserved.	

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		34	<p>XD Bit Disable (R/W)</p> <p>When set to 1, the Execute Disable Bit feature (XD Bit) is disabled and the XD Bit extended feature flag will be clear (CPUID.80000001H: EDX[20]=0).</p> <p>When set to a 0 (default), the Execute Disable Bit feature (if available) allows the OS to enable PAE paging and take advantage of data only pages.</p> <p>BIOS must not alter the contents of this bit location, if XD bit is not supported.. Writing this bit to 1 when the XD Bit extended feature flag is set to 0 may generate a #GP exception.</p>	if CPUID.80000001H:EDX[20] = 1
		63:35	Reserved.	
1B0H	432	IA32_ENERGY_PERF_BIAS	Performance Energy Bias Hint (R/W)	if CPUID.6H:ECX[3] = 1
		3:0	<p>Power Policy Preference:</p> <p>0 indicates preference to highest performance.</p> <p>15 indicates preference to maximize energy saving.</p>	
		63:4	Reserved.	
1B1H	433	IA32_PACKAGE_THERM_STATUS	<p>Package Thermal Status Information (RO)</p> <p>Contains status information about the package's thermal sensor.</p> <p>See Section 14.8, "Package Level Thermal Management."</p>	if CPUID.06H: EAX[6] = 1
		0	Pkg Thermal Status (RO):	
		1	Pkg Thermal Status Log (R/W):	
		2	Pkg PROCHOT # event (RO)	
		3	Pkg PROCHOT # log (R/WCO)	
		4	Pkg Critical Temperature Status (RO)	
		5	Pkg Critical Temperature Status log (R/WCO)	
		6	Pkg Thermal Threshold #1 Status (RO)	
		7	Pkg Thermal Threshold #1 log (R/WCO)	
		8	Pkg Thermal Threshold #2 Status (RO)	
		9	Pkg Thermal Threshold #1 log (R/WCO)	
10	Pkg Power Limitation Status (RO)			

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		11	Pkg Power Limitation log (R/WCO)	
		15:12	Reserved.	
		22:16	Pkg Digital Readout (RO)	
		63:23	Reserved.	
1B2H	434	IA32_PACKAGE_THERM_INTERRUPT	Pkg Thermal Interrupt Control (R/W) Enables and disables the generation of an interrupt on temperature transitions detected with the package's thermal sensor. See Section 14.8, "Package Level Thermal Management."	If CPUID.06H: EAX[6] = 1
		0	Pkg High-Temperature Interrupt Enable	
		1	Pkg Low-Temperature Interrupt Enable	
		2	Pkg PROCHOT# Interrupt Enable	
		3	Reserved.	
		4	Pkr Overheat Interrupt Enable	
		7:5	Reserved.	
		14:8	Pkg Threshold #1 Value	
		15	Pkg Threshold #1 Interrupt Enable	
		22:16	Pkg Threshold #2 Value	
		23	Pkg Threshold #2 Interrupt Enable	
		24	Pkg Power Limit Notification Enable	
		63:25	Reserved.	
		1D9H	473	IA32_DEBUGCTL (MSR_DEBUGCTLA, MSR_DEBUGCTLB)
0	LBR: Setting this bit to 1 enables the processor to record a running trace of the most recent branches taken by the processor in the LBR stack.			06_01H
1	BTF: Setting this bit to 1 enables the processor to treat EFLAGS.TF as single-step on branches instead of single-step on instructions.			06_01H
5:2	Reserved.			
6	TR: Setting this bit to 1 enables branch trace messages to be sent.			06_0EH
7	BTS: Setting this bit enables branch trace messages (BTMs) to be logged in a BTS buffer.			06_0EH

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		8	BTINT: When clear, BTMs are logged in a BTS buffer in circular fashion. When this bit is set, an interrupt is generated by the BTS facility when the BTS buffer is full.	06_0EH
		9	1: BTS_OFF_OS: When set, BTS or BTM is skipped if CPL = 0.	06_0FH
		10	BTS_OFF_USR: When set, BTS or BTM is skipped if CPL > 0.	06_0FH
		11	FREEZE_LBRS_ON_PMI: When set, the LBR stack is frozen on a PMI request.	If CPUID.01H: ECX[15] = 1 and CPUID.0AH: EAX[7:0] > 1
		12	FREEZE_PERFMON_ON_PMI: When set, each ENABLE bit of the global counter control MSR are frozen (address 3BFH) on a PMI request	If CPUID.01H: ECX[15] = 1 and CPUID.0AH: EAX[7:0] > 1
		13	ENABLE_UNCORE_PMI: When set, enables the logical processor to receive and generate PMI on behalf of the uncore.	06_1AH
		14	FREEZE_WHILE_SMM: When set, freezes perfmon and trace messages while in SMM.	if IA32_PERF_CAPABILITIES[12] = '1
		63:15	Reserved.	
1F2H	498	IA32_SMRR_PHYSBASE	SMRR Base Address (Writeable only in SMM) Base address of SMM memory range.	If IA32_MTRR_CAP[SMRR] = 1
		7:0	Type. Specifies memory type of the range.	
		11:8	Reserved.	
		31:12	PhysBase. SMRR physical Base Address.	
		63:32	Reserved.	
1F3H	499	IA32_SMRR_PHYSMASK	SMRR Range Mask. (Writeable only in SMM) Range Mask of SMM memory range.	If IA32_MTRR_CAP[SMRR] = 1
		10:0	Reserved.	
		11	Valid Enable range mask.	
		31:12	PhysMask SMRR address range mask.	
		63:32	Reserved.	
1F8H	504	IA32_PLATFORM_DCA_CAP	DCA Capability (R)	06_0FH

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
1F9H	505	IA32_CPU_DCA_CAP	If set, CPU supports Prefetch-Hint type.	
1FAH	506	IA32_DCA_0_CAP	DCA type 0 Status and Control register.	06_2EH
		0	DCA_ACTIVE: Set by HW when DCA is fuse-enabled and no defeatures are set.	06_2EH
		2:1	TRANSACTION	06_2EH
		6:3	DCA_TYPE	06_2EH
		10:7	DCA_QUEUE_SIZE	06_2EH
		12:11	Reserved.	06_2EH
		16:13	DCA_DELAY: Writes will update the register but have no HW side-effect.	06_2EH
		23:17	Reserved.	06_2EH
		24	SW_BLOCK: SW can request DCA block by setting this bit.	06_2EH
		25	Reserved.	06_2EH
		26	HW_BLOCK: Set when DCA is blocked by HW (e.g. CRO.CD = 1).	06_2EH
	31:27	Reserved.	06_2EH	
200H	512	IA32_MTRR_PHYSBASE0 (MTRRphysBase0)	See Section 11.11.2.3, "Variable Range MTRRs."	06_01H
201H	513	IA32_MTRR_PHYSMASK0	MTRRphysMask0	06_01H
202H	514	IA32_MTRR_PHYSBASE1	MTRRphysBase1	06_01H
203H	515	IA32_MTRR_PHYSMASK1	MTRRphysMask1	06_01H
204H	516	IA32_MTRR_PHYSBASE2	MTRRphysBase2	06_01H
205H	517	IA32_MTRR_PHYSMASK2	MTRRphysMask2	06_01H
206H	518	IA32_MTRR_PHYSBASE3	MTRRphysBase3	06_01H
207H	519	IA32_MTRR_PHYSMASK3	MTRRphysMask3	06_01H
208H	520	IA32_MTRR_PHYSBASE4	MTRRphysBase4	06_01H
209H	521	IA32_MTRR_PHYSMASK4	MTRRphysMask4	06_01H
20AH	522	IA32_MTRR_PHYSBASE5	MTRRphysBase5	06_01H
20BH	523	IA32_MTRR_PHYSMASK5	MTRRphysMask5	06_01H
20CH	524	IA32_MTRR_PHYSBASE6	MTRRphysBase6	06_01H
20DH	525	IA32_MTRR_PHYSMASK6	MTRRphysMask6	06_01H
20EH	526	IA32_MTRR_PHYSBASE7	MTRRphysBase7	06_01H
20FH	527	IA32_MTRR_PHYSMASK7	MTRRphysMask7	06_01H
210H	528	IA32_MTRR_PHYSBASE8	MTRRphysBase8	if IA32_MTRR_CAP[7:0] > 8

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
211H	529	IA32_MTRR_PHYSMASK8	MTRRphysMask8	if IA32_MTRR_CAP[7:0] > 8
212H	530	IA32_MTRR_PHYSBASE9	MTRRphysBase9	if IA32_MTRR_CAP[7:0] > 9
213H	531	IA32_MTRR_PHYSMASK9	MTRRphysMask9	if IA32_MTRR_CAP[7:0] > 9
250H	592	IA32_MTRR_FIX64K_00000	MTRRfix64K_00000	06_01H
258H	600	IA32_MTRR_FIX16K_80000	MTRRfix16K_80000	06_01H
259H	601	IA32_MTRR_FIX16K_A0000	MTRRfix16K_A0000	06_01H
268H	616	IA32_MTRR_FIX4K_C0000 (MTRRfix4K_C0000)	See Section 11.11.2.2, "Fixed Range MTRRs."	06_01H
269H	617	IA32_MTRR_FIX4K_C8000	MTRRfix4K_C8000	06_01H
26AH	618	IA32_MTRR_FIX4K_D0000	MTRRfix4K_D0000	06_01H
26BH	619	IA32_MTRR_FIX4K_D8000	MTRRfix4K_D8000	06_01H
26CH	620	IA32_MTRR_FIX4K_E0000	MTRRfix4K_E0000	06_01H
26DH	621	IA32_MTRR_FIX4K_E8000	MTRRfix4K_E8000	06_01H
26EH	622	IA32_MTRR_FIX4K_F0000	MTRRfix4K_F0000	06_01H
26FH	623	IA32_MTRR_FIX4K_F8000	MTRRfix4K_F8000	06_01H
277H	631	IA32_PAT	IA32_PAT (R/W)	06_05H
		2:0	PA0	
		7:3	Reserved.	
		10:8	PA1	
		15:11	Reserved.	
		18:16	PA2	
		23:19	Reserved.	
		26:24	PA3	
		31:27	Reserved.	
		34:32	PA4	
		39:35	Reserved.	
		42:40	PA5	
		47:43	Reserved.	
		50:48	PA6	
		55:51	Reserved.	
58:56	PA7			
63:59	Reserved.			

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
280H	640	IA32_MCO_CTL2	(R/W)	06_1AH
		14:0	Corrected error count threshold.	
		29:15	Reserved.	
		30	CMCI_EN	
		63:31	Reserved.	
281H	641	IA32_MC1_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_1AH
282H	642	IA32_MC2_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_1AH
283H	643	IA32_MC3_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_1AH
284H	644	IA32_MC4_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_1AH
285H	645	IA32_MC5_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_1AH
286H	646	IA32_MC6_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_1AH
287H	647	IA32_MC7_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_1AH
288H	648	IA32_MC8_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_1AH
289H	649	IA32_MC9_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
28AH	650	IA32_MC10_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
28BH	651	IA32_MC11_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
28CH	652	IA32_MC12_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
28DH	653	IA32_MC13_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
28EH	654	IA32_MC14_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
28FH	655	IA32_MC15_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
290H	656	IA32_MC16_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
291H	657	IA32_MC17_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
292H	658	IA32_MC18_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
293H	659	IA32_MC19_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
294H	660	IA32_MC20_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
295H	661	IA32_MC21_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_2EH
296H	662	IA32_MC22_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_3EH
297H	663	IA32_MC23_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_3EH
298H	664	IA32_MC24_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_3EH
299H	665	IA32_MC25_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_3EH
29AH	666	IA32_MC26_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_3EH
29BH	667	IA32_MC27_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_3EH
29CH	668	IA32_MC28_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_3EH
29DH	669	IA32_MC29_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_3EH

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
29EH	670	IA32_MC30_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_3EH
29FH	671	IA32_MC31_CTL2	(R/W) same fields as IA32_MCO_CTL2.	06_3EH
2FFH	767	IA32_MTRR_DEF_TYPE	MTRRdefType (R/W)	06_01H
		2:0	Default Memory Type	
		9:3	Reserved.	
		10	Fixed Range MTRR Enable	
		11	MTRR Enable	
		63:12	Reserved.	
309H	777	IA32_FIXED_CTR0 (MSR_PERF_FIXED_CTR0)	Fixed-Function Performance Counter 0 (R/W): Counts Instr_Retired.Any.	If CPUID.0AH: EDX[4:0] > 0
30AH	778	IA32_FIXED_CTR1 (MSR_PERF_FIXED_CTR1)	Fixed-Function Performance Counter 1 0 (R/W): Counts CPU_CLK_Unhalted.Core	If CPUID.0AH: EDX[4:0] > 1
30BH	779	IA32_FIXED_CTR2 (MSR_PERF_FIXED_CTR2)	Fixed-Function Performance Counter 0 0 (R/W): Counts CPU_CLK_Unhalted.Ref	If CPUID.0AH: EDX[4:0] > 2
345H	837	IA32_PERF_CAPABILITIES	RO	If CPUID.01H: ECX[15] = 1
		5:0	LBR format	
		6	PEBS Trap	
		7	PEBSSaveArchRegs	
		11:8	PEBS Record Format	
		12	1: Freeze while SMM is supported.	
		13	1: Full width of counter writable via IA32_A_PMCx.	
		63:14	Reserved.	
38DH	909	IA32_FIXED_CTR_CTRL (MSR_PERF_FIXED_CTR_CTRL)	Fixed-Function Performance Counter Control (R/W) Counter increments while the results of ANDing respective enable bit in IA32_PERF_GLOBAL_CTRL with the corresponding OS or USR bits in this MSR is true.	If CPUID.0AH: EAX[7:0] > 1
		0	ENO_OS: Enable Fixed Counter 0 to count while CPL = 0.	
		1	ENO_Usr: Enable Fixed Counter 0 to count while CPL > 0.	

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		2	AnyThread: When set to 1, it enables counting the associated event conditions occurring across all logical processors sharing a processor core. When set to 0, the counter only increments the associated event conditions occurring in the logical processor which programmed the MSR.	If CPUID.OAH: EAX[7:0] > 2
		3	EN0_PMI: Enable PMI when fixed counter 0 overflows.	
		4	EN1_OS: Enable Fixed Counter 1 to count while CPL = 0.	
		5	EN1_Usr: Enable Fixed Counter 1 to count while CPL > 0.	
		6	AnyThread: When set to 1, it enables counting the associated event conditions occurring across all logical processors sharing a processor core. When set to 0, the counter only increments the associated event conditions occurring in the logical processor which programmed the MSR.	If CPUID.OAH: EAX[7:0] > 2
		7	EN1_PMI: Enable PMI when fixed counter 1 overflows.	
		8	EN2_OS: Enable Fixed Counter 2 to count while CPL = 0.	
		9	EN2_Usr: Enable Fixed Counter 2 to count while CPL > 0.	
		10	AnyThread: When set to 1, it enables counting the associated event conditions occurring across all logical processors sharing a processor core. When set to 0, the counter only increments the associated event conditions occurring in the logical processor which programmed the MSR.	If CPUID.OAH: EAX[7:0] > 2
		11	EN2_PMI: Enable PMI when fixed counter 2 overflows.	
		63:12	Reserved.	
38EH	910	IA32_PERF_GLOBAL_STATUS (MSR_PERF_GLOBAL_STATUS)	Global Performance Counter Status (RO)	If CPUID.OAH: EAX[7:0] > 0
		0	Ovf_PMC0: Overflow status of IA32_PMC0.	If CPUID.OAH: EAX[7:0] > 0
		1	Ovf_PMC1: Overflow status of IA32_PMC1.	If CPUID.OAH: EAX[7:0] > 0
		2	Ovf_PMC2: Overflow status of IA32_PMC2.	06_2EH
		3	Ovf_PMC3: Overflow status of IA32_PMC3.	06_2EH

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		31:4	Reserved.	
		32	Ovf_FixedCtr0: Overflow status of IA32_FIXED_CTR0.	If CPUID.0AH: EAX[7:0] > 1
		33	Ovf_FixedCtr1: Overflow status of IA32_FIXED_CTR1.	If CPUID.0AH: EAX[7:0] > 1
		34	Ovf_FixedCtr2: Overflow status of IA32_FIXED_CTR2.	If CPUID.0AH: EAX[7:0] > 1
		60:35	Reserved.	
		61	Ovf_Uncore: Uncore counter overflow status.	If CPUID.0AH: EAX[7:0] > 2
		62	OvfBuf: DS SAVE area Buffer overflow status.	If CPUID.0AH: EAX[7:0] > 0
		63	CondChg: status bits of this register has changed.	If CPUID.0AH: EAX[7:0] > 0
38FH	911	IA32_PERF_GLOBAL_CTRL (MSR_PERF_GLOBAL_CTRL)	Global Performance Counter Control (R/W) Counter increments while the result of ANDing respective enable bit in this MSR with the corresponding OS or USR bits in the general-purpose or fixed counter control MSR is true.	If CPUID.0AH: EAX[7:0] > 0
		0	EN_PMC0	If CPUID.0AH: EAX[7:0] > 0
		1	EN_PMC1	If CPUID.0AH: EAX[7:0] > 0
		31:2	Reserved.	
		32	EN_FIXED_CTR0	If CPUID.0AH: EAX[7:0] > 1
		33	EN_FIXED_CTR1	If CPUID.0AH: EAX[7:0] > 1
		34	EN_FIXED_CTR2	If CPUID.0AH: EAX[7:0] > 1
		63:35	Reserved.	
390H	912	IA32_PERF_GLOBAL_OVF_CTRL (MSR_PERF_GLOBAL_OVF_CTRL)	Global Performance Counter Overflow Control (R/W)	If CPUID.0AH: EAX[7:0] > 0
		0	Set 1 to Clear Ovf_PMC0 bit.	If CPUID.0AH: EAX[7:0] > 0
		1	Set 1 to Clear Ovf_PMC1 bit.	If CPUID.0AH: EAX[7:0] > 0
		31:2	Reserved.	
		32	Set 1 to Clear Ovf_FIXED_CTR0 bit.	If CPUID.0AH: EAX[7:0] > 1
		33	Set 1 to Clear Ovf_FIXED_CTR1 bit.	If CPUID.0AH: EAX[7:0] > 1
		34	Set 1 to Clear Ovf_FIXED_CTR2 bit.	If CPUID.0AH: EAX[7:0] > 1
		60:35	Reserved.	
61	Set 1 to Clear Ovf_Uncore: bit.	06_2EH		

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		62	Set 1 to Clear OvfBuf: bit.	If CPUID.0AH: EAX[7:0] > 0
		63	Set to 1 to clear CondChg: bit.	If CPUID.0AH: EAX[7:0] > 0
3F1H	1009	IA32_PEBS_ENABLE	PEBS Control (R/W)	
		0	Enable PEBS on IA32_PMC0.	06_0FH
		1-3	Reserved or Model specific .	
		31:4	Reserved.	
		35-32	Reserved or Model specific .	
		63:36	Reserved.	
		400H	1024	IA32_MCO_CTL
401H	1025	IA32_MCO_STATUS	MCO_STATUS	P6 Family Processors
402H	1026	IA32_MCO_ADDR ¹	MCO_ADDR	P6 Family Processors
403H	1027	IA32_MCO_MISC	MCO_MISC	P6 Family Processors
404H	1028	IA32_MC1_CTL	MC1_CTL	P6 Family Processors
405H	1029	IA32_MC1_STATUS	MC1_STATUS	P6 Family Processors
406H	1030	IA32_MC1_ADDR ²	MC1_ADDR	P6 Family Processors
407H	1031	IA32_MC1_MISC	MC1_MISC	P6 Family Processors
408H	1032	IA32_MC2_CTL	MC2_CTL	P6 Family Processors
409H	1033	IA32_MC2_STATUS	MC2_STATUS	P6 Family Processors
40AH	1034	IA32_MC2_ADDR ¹	MC2_ADDR	P6 Family Processors
40BH	1035	IA32_MC2_MISC	MC2_MISC	P6 Family Processors
40CH	1036	IA32_MC3_CTL	MC3_CTL	P6 Family Processors
40DH	1037	IA32_MC3_STATUS	MC3_STATUS	P6 Family Processors
40EH	1038	IA32_MC3_ADDR ¹	MC3_ADDR	P6 Family Processors
40FH	1039	IA32_MC3_MISC	MC3_MISC	P6 Family Processors
410H	1040	IA32_MC4_CTL	MC4_CTL	P6 Family Processors
411H	1041	IA32_MC4_STATUS	MC4_STATUS	P6 Family Processors
412H	1042	IA32_MC4_ADDR ¹	MC4_ADDR	P6 Family Processors
413H	1043	IA32_MC4_MISC	MC4_MISC	P6 Family Processors
414H	1044	IA32_MC5_CTL	MC5_CTL	06_0FH
415H	1045	IA32_MC5_STATUS	MC5_STATUS	06_0FH
416H	1046	IA32_MC5_ADDR ¹	MC5_ADDR	06_0FH
417H	1047	IA32_MC5_MISC	MC5_MISC	06_0FH
418H	1048	IA32_MC6_CTL	MC6_CTL	06_1DH
419H	1049	IA32_MC6_STATUS	MC6_STATUS	06_1DH

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
41AH	1050	IA32_MC6_ADDR ⁷	MC6_ADDR	06_1DH
41BH	1051	IA32_MC6_MISC	MC6_MISC	06_1DH
41CH	1052	IA32_MC7_CTL	MC7_CTL	06_1AH
41DH	1053	IA32_MC7_STATUS	MC7_STATUS	06_1AH
41EH	1054	IA32_MC7_ADDR ⁷	MC7_ADDR	06_1AH
41FH	1055	IA32_MC7_MISC	MC7_MISC	06_1AH
420H	1056	IA32_MC8_CTL	MC8_CTL	06_1AH
421H	1057	IA32_MC8_STATUS	MC8_STATUS	06_1AH
422H	1058	IA32_MC8_ADDR ⁷	MC8_ADDR	06_1AH
423H	1059	IA32_MC8_MISC	MC8_MISC	06_1AH
424H	1060	IA32_MC9_CTL	MC9_CTL	06_2EH
425H	1061	IA32_MC9_STATUS	MC9_STATUS	06_2EH
426H	1062	IA32_MC9_ADDR ⁷	MC9_ADDR	06_2EH
427H	1063	IA32_MC9_MISC	MC9_MISC	06_2EH
428H	1064	IA32_MC10_CTL	MC10_CTL	06_2EH
429H	1065	IA32_MC10_STATUS	MC10_STATUS	06_2EH
42AH	1066	IA32_MC10_ADDR ⁷	MC10_ADDR	06_2EH
42BH	1067	IA32_MC10_MISC	MC10_MISC	06_2EH
42CH	1068	IA32_MC11_CTL	MC11_CTL	06_2EH
42DH	1069	IA32_MC11_STATUS	MC11_STATUS	06_2EH
42EH	1070	IA32_MC11_ADDR ⁷	MC11_ADDR	06_2EH
42FH	1071	IA32_MC11_MISC	MC11_MISC	06_2EH
430H	1072	IA32_MC12_CTL	MC12_CTL	06_2EH
431H	1073	IA32_MC12_STATUS	MC12_STATUS	06_2EH
432H	1074	IA32_MC12_ADDR ⁷	MC12_ADDR	06_2EH
433H	1075	IA32_MC12_MISC	MC12_MISC	06_2EH
434H	1076	IA32_MC13_CTL	MC13_CTL	06_2EH
435H	1077	IA32_MC13_STATUS	MC13_STATUS	06_2EH
436H	1078	IA32_MC13_ADDR ⁷	MC13_ADDR	06_2EH
437H	1079	IA32_MC13_MISC	MC13_MISC	06_2EH
438H	1080	IA32_MC14_CTL	MC14_CTL	06_2EH
439H	1081	IA32_MC14_STATUS	MC14_STATUS	06_2EH
43AH	1082	IA32_MC14_ADDR ⁷	MC14_ADDR	06_2EH
43BH	1083	IA32_MC14_MISC	MC14_MISC	06_2EH

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
43CH	1084	IA32_MC15_CTL	MC15_CTL	06_2EH
43DH	1085	IA32_MC15_STATUS	MC15_STATUS	06_2EH
43EH	1086	IA32_MC15_ADDR ⁷	MC15_ADDR	06_2EH
43FH	1087	IA32_MC15_MISC	MC15_MISC	06_2EH
440H	1088	IA32_MC16_CTL	MC16_CTL	06_2EH
441H	1089	IA32_MC16_STATUS	MC16_STATUS	06_2EH
442H	1090	IA32_MC16_ADDR ⁷	MC16_ADDR	06_2EH
443H	1091	IA32_MC16_MISC	MC16_MISC	06_2EH
444H	1092	IA32_MC17_CTL	MC17_CTL	06_2EH
445H	1093	IA32_MC17_STATUS	MC17_STATUS	06_2EH
446H	1094	IA32_MC17_ADDR ⁷	MC17_ADDR	06_2EH
447H	1095	IA32_MC17_MISC	MC17_MISC	06_2EH
448H	1096	IA32_MC18_CTL	MC18_CTL	06_2EH
449H	1097	IA32_MC18_STATUS	MC18_STATUS	06_2EH
44AH	1098	IA32_MC18_ADDR ⁷	MC18_ADDR	06_2EH
44BH	1099	IA32_MC18_MISC	MC18_MISC	06_2EH
44CH	1100	IA32_MC19_CTL	MC19_CTL	06_2EH
44DH	1101	IA32_MC19_STATUS	MC19_STATUS	06_2EH
44EH	1102	IA32_MC19_ADDR ⁷	MC19_ADDR	06_2EH
44FH	1103	IA32_MC19_MISC	MC19_MISC	06_2EH
450H	1104	IA32_MC20_CTL	MC20_CTL	06_2EH
451H	1105	IA32_MC20_STATUS	MC20_STATUS	06_2EH
452H	1106	IA32_MC20_ADDR ⁷	MC20_ADDR	06_2EH
453H	1107	IA32_MC20_MISC	MC20_MISC	06_2EH
454H	1108	IA32_MC21_CTL	MC21_CTL	06_2EH
455H	1109	IA32_MC21_STATUS	MC21_STATUS	06_2EH
456H	1110	IA32_MC21_ADDR ⁷	MC21_ADDR	06_2EH
457H	1111	IA32_MC21_MISC	MC21_MISC	06_2EH
480H	1152	IA32_VMX_BASIC	Reporting Register of Basic VMX Capabilities (R/O) See Appendix A.1, "Basic VMX Information."	If CPUID.01H:ECX.[bit 5] = 1
481H	1153	IA32_VMX_PINBASED_CTL	Capability Reporting Register of Pin-based VM-execution Controls (R/O) See Appendix A.3.1, "Pin-Based VM-Execution Controls."	If CPUID.01H:ECX.[bit 5] = 1

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
482H	1154	IA32_VMX_PROCBASED_CTL5	Capability Reporting Register of Primary Processor-based VM-execution Controls (R/O) See Appendix A.3.2, "Primary Processor-Based VM-Execution Controls."	If CPUID.01H:ECX.[bit 5] = 1
483H	1155	IA32_VMX_EXIT_CTL5	Capability Reporting Register of VM-exit Controls (R/O) See Appendix A.4, "VM-Exit Controls."	If CPUID.01H:ECX.[bit 5] = 1
484H	1156	IA32_VMX_ENTRY_CTL5	Capability Reporting Register of VM-entry Controls (R/O) See Appendix A.5, "VM-Entry Controls."	If CPUID.01H:ECX.[bit 5] = 1
485H	1157	IA32_VMX_MISC	Reporting Register of Miscellaneous VMX Capabilities (R/O) See Appendix A.6, "Miscellaneous Data."	If CPUID.01H:ECX.[bit 5] = 1
486H	1158	IA32_VMX_CRO_FIXED0	Capability Reporting Register of CRO Bits Fixed to 0 (R/O) See Appendix A.7, "VMX-Fixed Bits in CRO."	If CPUID.01H:ECX.[bit 5] = 1
487H	1159	IA32_VMX_CRO_FIXED1	Capability Reporting Register of CRO Bits Fixed to 1 (R/O) See Appendix A.7, "VMX-Fixed Bits in CRO."	If CPUID.01H:ECX.[bit 5] = 1
488H	1160	IA32_VMX_CR4_FIXED0	Capability Reporting Register of CR4 Bits Fixed to 0 (R/O) See Appendix A.8, "VMX-Fixed Bits in CR4."	If CPUID.01H:ECX.[bit 5] = 1
489H	1161	IA32_VMX_CR4_FIXED1	Capability Reporting Register of CR4 Bits Fixed to 1 (R/O) See Appendix A.8, "VMX-Fixed Bits in CR4."	If CPUID.01H:ECX.[bit 5] = 1
48AH	1162	IA32_VMX_VMCS_ENUM	Capability Reporting Register of VMCS Field Enumeration (R/O) See Appendix A.9, "VMCS Enumeration."	If CPUID.01H:ECX.[bit 5] = 1
48BH	1163	IA32_VMX_PROCBASED_CTL52	Capability Reporting Register of Secondary Processor-based VM-execution Controls (R/O) See Appendix A.3.3, "Secondary Processor-Based VM-Execution Controls."	If (CPUID.01H:ECX.[bit 5] and IA32_VMX_PROCBASED_CTL5[bit 63])
48CH	1164	IA32_VMX_EPT_VPID_CAP	Capability Reporting Register of EPT and VPID (R/O) See Appendix A.10, "VPID and EPT Capabilities."	If (CPUID.01H:ECX.[bit 5], IA32_VMX_PROCBASED_CTL5[bit 63], and either IA32_VMX_PROCBASED_CTL52[bit 33] or IA32_VMX_PROCBASED_CTL52[bit 37])

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
48DH	1165	IA32_VMX_TRUE_PINBASED_CTL5	Capability Reporting Register of Pin-based VM-execution Flex Controls (R/O) See Appendix A.3.1, "Pin-Based VM-Execution Controls."	If (CPUID.01H:ECX.[bit 5] = 1 and IA32_VMX_BASIC[bit 55])
48EH	1166	IA32_VMX_TRUE_PROCBASED_CTL5	Capability Reporting Register of Primary Processor-based VM-execution Flex Controls (R/O) See Appendix A.3.2, "Primary Processor-Based VM-Execution Controls."	If(CPUID.01H:ECX.[bit 5] = 1 and IA32_VMX_BASIC[bit 55])
48FH	1167	IA32_VMX_TRUE_EXIT_CTL5	Capability Reporting Register of VM-exit Flex Controls (R/O) See Appendix A.4, "VM-Exit Controls."	If(CPUID.01H:ECX.[bit 5] = 1 and IA32_VMX_BASIC[bit 55])
490H	1168	IA32_VMX_TRUE_ENTRY_CTL5	Capability Reporting Register of VM-entry Flex Controls (R/O) See Appendix A.5, "VM-Entry Controls."	If(CPUID.01H:ECX.[bit 5] = 1 and IA32_VMX_BASIC[bit 55])
491H	1169	IA32_VMX_VMFUNC	Capability Reporting Register of VM-function Controls (R/O)	If(CPUID.01H:ECX.[bit 5] = 1 and IA32_VMX_BASIC[bit 55])
4C1H	1217	IA32_A_PMC0	Full Width Writable IA32_PMC0 Alias (R/W)	(If CPUID.0AH: EAX[15:8] > 0) & IA32_PERF_CAPABILITIES[13] = 1
4C2H	1218	IA32_A_PMC1	Full Width Writable IA32_PMC1 Alias (R/W)	(If CPUID.0AH: EAX[15:8] > 1) & IA32_PERF_CAPABILITIES[13] = 1
4C3H	1219	IA32_A_PMC2	Full Width Writable IA32_PMC2 Alias (R/W)	(If CPUID.0AH: EAX[15:8] > 2) & IA32_PERF_CAPABILITIES[13] = 1
4C4H	1220	IA32_A_PMC3	Full Width Writable IA32_PMC3 Alias (R/W)	(If CPUID.0AH: EAX[15:8] > 3) & IA32_PERF_CAPABILITIES[13] = 1
4C5H	1221	IA32_A_PMC4	Full Width Writable IA32_PMC4 Alias (R/W)	(If CPUID.0AH: EAX[15:8] > 4) & IA32_PERF_CAPABILITIES[13] = 1
4C6H	1222	IA32_A_PMC5	Full Width Writable IA32_PMC5 Alias (R/W)	(If CPUID.0AH: EAX[15:8] > 5) & IA32_PERF_CAPABILITIES[13] = 1

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
4C7H	1223	IA32_A_PMC6	Full Width Writable IA32_PMC6 Alias (R/W)	If CPUID.0AH:EAX[15:8] > 6) & IA32_PERF_CAPABILITIES[13] = 1
4C8H	1224	IA32_A_PMC7	Full Width Writable IA32_PMC7 Alias (R/W)	If CPUID.0AH:EAX[15:8] > 7) & IA32_PERF_CAPABILITIES[13] = 1
4D0H	1232	IA32_MCG_EXT_CTL	(R/W)	If IA32_MCG_CAP.LMCE_P = 1
		0	LMCE_EN.	
		63:1	Reserved.	
600H	1536	IA32_DS_AREA	DS Save Area (R/W) Points to the linear address of the first byte of the DS buffer management area, which is used to manage the BTS and PEBS buffers. See Section 18.12.4, “Debug Store (DS) Mechanism.”	OF_OH
		63:0	The linear address of the first byte of the DS buffer management area, if IA-32e mode is active.	
		31:0	The linear address of the first byte of the DS buffer management area, if not in IA-32e mode.	
		63:32	Reserved if not in IA-32e mode.	
6E0H	1760	IA32_TSC_DEADLINE	TSC Target of Local APIC’s TSC Deadline Mode (R/W)	If(CPUID.01H:ECX.[bit 25] = 1
770H	1904	IA32_PM_ENABLE	Enable/disable HWP (R/W)	If(CPUID.06H:EAX.[bit 7] = 1
		0	HWP_ENABLE (R/W1-Once). See Section 14.4.2, “Enabling HWP”	If(CPUID.06H:EAX.[bit 7] = 1
		63:1	Reserved.	
771H	1905	IA32_HWP_CAPABILITIES	HWP Performance Range Enumeration (RO)	If(CPUID.06H:EAX.[bit 7] = 1
		7:0	Highest_Performance See Section 14.4.3, “HWP Performance Range and Dynamic Capabilities”	If(CPUID.06H:EAX.[bit 7] = 1

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		15:8	Guaranteed_Performance See Section 14.4.3, "HWP Performance Range and Dynamic Capabilities"	If(CPUID.06H:EAX.[bit 7] = 1
		23:16	Most_Efficient_Performance See Section 14.4.3, "HWP Performance Range and Dynamic Capabilities"	If(CPUID.06H:EAX.[bit 7] = 1
		31:24	Lowest_Performance See Section 14.4.3, "HWP Performance Range and Dynamic Capabilities"	If(CPUID.06H:EAX.[bit 7] = 1
		63:32	Reserved.	
772H	1906	IA32_HWP_REQUEST_PKG	Power Management Control Hints for All Logical Processors in a Package (R/W)	If(CPUID.06H:EAX.[bit 11] = 1
		7:0	Minimum_Performance See Section 14.4.4, "Managing HWP"	If(CPUID.06H:EAX.[bit 11] = 1
		15:8	Maximum_Performance See Section 14.4.4, "Managing HWP"	If(CPUID.06H:EAX.[bit 11] = 1
		23:16	Desired_Performance See Section 14.4.4, "Managing HWP"	If(CPUID.06H:EAX.[bit 11] = 1
		31:24	Energy_Performance_Preference See Section 14.4.4, "Managing HWP"	If(CPUID.06H:EAX.[bit 11] = 1 and CPUID.06HEAX.[bit 10] = 1
		41:32	Activity_Window See Section 14.4.4, "Managing HWP"	If(CPUID.06H:EAX.[bit 11] = 1 and CPUID.06HEAX.[bit 9] = 1
		63:42	Reserved.	
773H	1907	IA32_HWP_INTERRUPT	Control HWP Native Interrupts (R/W)	If(CPUID.06H:EAX.[bit 8] = 1
		0	EN_Guaranteed_Performance_Change. See Section 14.4.6, "HWP Notifications"	If(CPUID.06H:EAX.[bit 8] = 1
		1	EN_Excursion_Minimum. See Section 14.4.6, "HWP Notifications"	If(CPUID.06H:EAX.[bit 8] = 1
		63:2	Reserved.	
774H	1908	IA32_HWP_REQUEST	Power Management Control Hints to a Logical Processor (R/W)	If(CPUID.06H:EAX.[bit 7] = 1
		7:0	Minimum_Performance See Section 14.4.4, "Managing HWP"	If(CPUID.06H:EAX.[bit 7] = 1
		15:8	Maximum_Performance See Section 14.4.4, "Managing HWP"	If(CPUID.06H:EAX.[bit 7] = 1

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		23:16	Desired_Performance See Section 14.4.4, "Managing HWP"	If (CPUID.06H:EAX.[bit 7] = 1
		31:24	Energy_Performance_Preference See Section 14.4.4, "Managing HWP"	If CPUID.06HEAX.[bit 7] = 1 and (CPUID.06H:EAX.[bit 10] = 1
		41:32	Activity_Window See Section 14.4.4, "Managing HWP"	If CPUID.06HEAX.[bit 7] = 1 and (CPUID.06H:EAX.[bit 9] = 1
		42	Package_Control See Section 14.4.4, "Managing HWP"	If CPUID.06HEAX.[bit 7] = 1 and (CPUID.06H:EAX.[bit 11] = 1
		63:43	Reserved.	
777H	1911	IA32_HWP_STATUS	Log bits indicating changes to Guaranteed & excursions to Minimum (R/W)	If (CPUID.06H:EAX.[bit 7] = 1
		0	Guaranteed_Performance_Change (R/WCO). See Section 14.4.5, "HWP Feedback"	If (CPUID.06H:EAX.[bit 7] = 1
		1	Reserved.	
		2	Excursion_To_Minimum (R/WCO). See Section 14.4.5, "HWP Feedback"	If (CPUID.06H:EAX.[bit 7] = 1
		63:3	Reserved.	
802H	2050	IA32_X2APIC_APICID	x2APIC ID Register (R/O) See x2APIC Specification	If (CPUID.01H:ECX.[bit 21] = 1)
803H	2051	IA32_X2APIC_VERSION	x2APIC Version Register (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
808H	2056	IA32_X2APIC_TPR	x2APIC Task Priority Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
80AH	2058	IA32_X2APIC_PPR	x2APIC Processor Priority Register (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
80BH	2059	IA32_X2APIC_EOI	x2APIC EOI Register (W/O)	If (CPUID.01H:ECX.[bit 21] = 1)
80DH	2061	IA32_X2APIC_LDR	x2APIC Logical Destination Register (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
80FH	2063	IA32_X2APIC_SIVR	x2APIC Spurious Interrupt Vector Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
810H	2064	IA32_X2APIC_ISR0	x2APIC In-Service Register Bits 31:0 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
811H	2065	IA32_X2APIC_ISR1	x2APIC In-Service Register Bits 63:32 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
812H	2066	IA32_X2APIC_ISR2	x2APIC In-Service Register Bits 95:64 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
813H	2067	IA32_X2APIC_ISR3	x2APIC In-Service Register Bits 127:96 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
814H	2068	IA32_X2APIC_ISR4	x2APIC In-Service Register Bits 159:128 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
815H	2069	IA32_X2APIC_ISR5	x2APIC In-Service Register Bits 191:160 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
816H	2070	IA32_X2APIC_ISR6	x2APIC In-Service Register Bits 223:192 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
817H	2071	IA32_X2APIC_ISR7	x2APIC In-Service Register Bits 255:224 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
818H	2072	IA32_X2APIC_TMR0	x2APIC Trigger Mode Register Bits 31:0 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
819H	2073	IA32_X2APIC_TMR1	x2APIC Trigger Mode Register Bits 63:32 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
81AH	2074	IA32_X2APIC_TMR2	x2APIC Trigger Mode Register Bits 95:64 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
81BH	2075	IA32_X2APIC_TMR3	x2APIC Trigger Mode Register Bits 127:96 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
81CH	2076	IA32_X2APIC_TMR4	x2APIC Trigger Mode Register Bits 159:128 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
81DH	2077	IA32_X2APIC_TMR5	x2APIC Trigger Mode Register Bits 191:160 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
81EH	2078	IA32_X2APIC_TMR6	x2APIC Trigger Mode Register Bits 223:192 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
81FH	2079	IA32_X2APIC_TMR7	x2APIC Trigger Mode Register Bits 255:224 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
820H	2080	IA32_X2APIC_IRR0	x2APIC Interrupt Request Register Bits 31:0 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
821H	2081	IA32_X2APIC_IRR1	x2APIC Interrupt Request Register Bits 63:32 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
822H	2082	IA32_X2APIC_IRR2	x2APIC Interrupt Request Register Bits 95:64 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
823H	2083	IA32_X2APIC_IRR3	x2APIC Interrupt Request Register Bits 127:96 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
824H	2084	IA32_X2APIC_IRR4	x2APIC Interrupt Request Register Bits 159:128 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
825H	2085	IA32_X2APIC_IRR5	x2APIC Interrupt Request Register Bits 191:160 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
826H	2086	IA32_X2APIC_IRR6	x2APIC Interrupt Request Register Bits 223:192 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
827H	2087	IA32_X2APIC_IRR7	x2APIC Interrupt Request Register Bits 255:224 (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
828H	2088	IA32_X2APIC_ESR	x2APIC Error Status Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
82FH	2095	IA32_X2APIC_LVT_CMCI	x2APIC LVT Corrected Machine Check Interrupt Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
830H	2096	IA32_X2APIC_ICR	x2APIC Interrupt Command Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
832H	2098	IA32_X2APIC_LVT_TIMER	x2APIC LVT Timer Interrupt Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
833H	2099	IA32_X2APIC_LVT_THERMAL	x2APIC LVT Thermal Sensor Interrupt Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
834H	2100	IA32_X2APIC_LVT_PMI	x2APIC LVT Performance Monitor Interrupt Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
835H	2101	IA32_X2APIC_LVT_LINT0	x2APIC LVT LINT0 Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
836H	2102	IA32_X2APIC_LVT_LINT1	x2APIC LVT LINT1 Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
837H	2103	IA32_X2APIC_LVT_ERROR	x2APIC LVT Error Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
838H	2104	IA32_X2APIC_INIT_COUNT	x2APIC Initial Count Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
839H	2105	IA32_X2APIC_CUR_COUNT	x2APIC Current Count Register (R/O)	If (CPUID.01H:ECX.[bit 21] = 1)
83EH	2110	IA32_X2APIC_DIV_CONF	x2APIC Divide Configuration Register (R/W)	If (CPUID.01H:ECX.[bit 21] = 1)
83FH	2111	IA32_X2APIC_SELF_IPI	x2APIC Self IPI Register (W/O)	If (CPUID.01H:ECX.[bit 21] = 1)
C80H	3200	IA32_DEBUG_INTERFACE	Silicon Debug Feature Control (R/W)	If (CPUID.01H:ECX.[bit 11] = 1)
		0	Enable (R/W). BIOS set 1 to enable Silicon debug features. Default is 0	If (CPUID.01H:ECX.[bit 11] = 1)
		29:1	Reserved.	
		30	Lock (R/W): If 1, locks any further change to the MSR. The lock bit is set automatically on the first SMI assertion even if not explicitly set by BIOS. Default is 0.	If (CPUID.01H:ECX.[bit 11] = 1)

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		31	Debug Occurred (R/O): This sticky bit is set by hardware to indicate the status of bit 0. Default is 0.	If (CPUID.01H:ECX.[bit 11] = 1
		63:32	Reserved.	
C8DH	3213	IA32_QM_EVTSEL	QoS Monitoring Event Select Register (R/W)	If (CPUID.(EAX=07H, ECX=0);EBX.[bit 12] = 1)
		7:0	Event ID: ID of a supported QoS monitoring event to report via IA32_QM_CTR.	
		31: 8	Reserved.	
		N+31:32	Resource Monitoring ID: ID for QoS monitoring hardware to report monitored data via IA32_QM_CTR.	N = Ceil (Log ₂ (CPUID.(EAX= 0FH, ECX=0H).EBX[31:0] +1))
		63:N+32	Reserved.	
C8EH	3214	IA32_QM_CTR	QoS Monitoring Counter Register (R/O)	If (CPUID.(EAX=07H, ECX=0);EBX.[bit 12] = 1)
		61:0	Resource Monitored Data	
		62	Unavailable: If 1, indicates data for this RMID is not available or not monitored for this resource or RMID.	
		63	Error: If 1, indicates and unsupported RMID or event type was written to IA32_PQR_QM_EVTSEL.	
C8FH	3215	IA32_PQR_ASSOC	QoS Resource Association Register (R/W)	If (CPUID.(EAX=07H, ECX=0);EBX.[bit 12] = 1)
		N-1:0	Resource Monitoring ID (R/W): ID for QoS monitoring hardware to track internal operation, e.g. memory access.	N = Ceil (Log ₂ (CPUID.(EAX= 0FH, ECX=0H).EBX[31:0] +1))
		31:N	Reserved	
		63:32	COS (R/W). The class of service (COS) to enforce (on writes); returns the current COS when read.	If (CPUID.(EAX=07H, ECX=0);EBX.[bit 15] = 1)
0C90H - 0D8FH		Reserved MSR Address Space for Platform QoS Enforcement Mask Registers	See Section 17.15.2.1, “Enumeration and Detection Support of CQE”	
C90H	3216	IA32_L3_QOS_MASK_0	L3 CQE Mask for COS0 (R/W)	If (CPUID.(10H, 0);EBX[bit 1] != 0)
		31:0	Capacity Bit Mask (R/W).	

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
		63:32	Reserved.	
C90H+n	3216+n	IA32_L3_QOS_MASK_n	L3 CQE Mask for COSn (R/W)	n = CPUID.(10H, 1):EDX[15:0]
		31:0	Capacity Bit Mask (R/W).	
		63:32	Reserved.	
DA0H	3488	IA32_XSS	Extended Supervisor State Mask (R/W)	If(CPUID.(0DH, 1):EAX.[bit 3] = 1
		7:0	Reserved	
		8	Trace Packet Configuration State (R/W).	
		63:9	Reserved.	
DB0H	3504	IA32_PKG_HDC_CTL	Package Level Enable/disable HDC (R/W)	If(CPUID.06H:EAX.[bit 13] = 1
		0	HDC_Pkg_Enable (R/W). Force HDC idling or wake up HDC-idled logical processors in the package. See Section 14.5.2, "Package level Enabling HDC"	If(CPUID.06H:EAX.[bit 13] = 1
		63:1	Reserved.	
DB1H	3505	IA32_PM_CTL1	Enable/disable HWP (R/W)	If(CPUID.06H:EAX.[bit 13] = 1
		0	HDC_Allow_Block (R/W). Allow/Block this logical processor for package level HDC control. See Section 14.5.3	If(CPUID.06H:EAX.[bit 13] = 1
		63:1	Reserved.	
DB2H	3506	IA32_THREAD_STALL	Per-Logical_Processor HDC Idle Residency (R/O)	If(CPUID.06H:EAX.[bit 13] = 1
		63:0	Stall_Cycle_Cnt (R/W). Stalled cycles due to HDC forced idle on this logical processor. See Section 14.5.4.1	If(CPUID.06H:EAX.[bit 13] = 1
4000_0000H - 4000_00FFH		Reserved MSR Address Space	All existing and future processors will not implement MSR in this range.	

Table 35-2 IA-32 Architectural MSRs (Contd.)

Register Address		Architectural MSR Name and bit fields (Former MSR Name)	MSR/Bit Description	Introduced as Architectural MSR
Hex	Decimal			
C000_0080H		IA32_EFER	Extended Feature Enables	If (CPUID.80000001.EDX.[bit 20] or CPUID.80000001.EDX.[bit 29])
		0	SYSCALL Enable (R/W) Enables SYSCALL/SYSRET instructions in 64-bit mode.	
		7:1	Reserved.	
		8	IA-32e Mode Enable (R/W) Enables IA-32e mode operation.	
		9	Reserved.	
		10	IA-32e Mode Active (R) Indicates IA-32e mode is active when set.	
		11	Execute Disable Bit Enable (R/W)	
		63:12	Reserved.	
C000_0081H		IA32_STAR	System Call Target Address (R/W)	If CPUID.80000001.EDX.[bit 29] = 1
C000_0082H		IA32_LSTAR	IA-32e Mode System Call Target Address (R/W)	If CPUID.80000001.EDX.[bit 29] = 1
C000_0084H		IA32_FMASK	System Call Flag Mask (R/W)	If CPUID.80000001.EDX.[bit 29] = 1
C000_0100H		IA32_FS_BASE	Map of BASE Address of FS (R/W)	If CPUID.80000001.EDX.[bit 29] = 1
C000_0101H		IA32_GS_BASE	Map of BASE Address of GS (R/W)	If CPUID.80000001.EDX.[bit 29] = 1
C000_0102H		IA32_KERNEL_GS_BASE	Swap Target of BASE Address of GS (R/W)	If CPUID.80000001.EDX.[bit 29] = 1
C000_0103H		IA32_TSC_AUX	Auxiliary TSC (Rw)	If CPUID.80000001H: EDX[27] = 1
		31:0	AUX: Auxiliary signature of TSC	
		63:32	Reserved.	

NOTES:

1. In processors based on Intel NetBurst® microarchitecture, MSR addresses 180H-197H are supported, software must treat them as model-specific. Starting with Intel Core Duo processors, MSR addresses 180H-185H, 188H-197H are reserved.

2. The *_ADDR MSR may or may not be present; this depends on flag settings in IA32_MCI_STATUS. See Section 15.3.2.3 and Section 15.3.2.4 for more information.

...

35.9.2 Additional MSRs Supported by Intel® Xeon Processor E7 v2 Family

Intel® Xeon Processor E7 v2 Family (based on Ivy Bridge-EP microarchitecture) with CPUID DisplayFamily_DisplayModel signature 06_3EH supports the MSR interfaces listed in Table 35-14, Table 35-18, and Table Table 35-19.

Table 35-19 Additional MSRs Supported by Intel® Xeon Processors E7 v2 Family with DisplayFamily_DisplayModel Signature 06_3EH

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
3AH	58	IA32_FEATURE_CONTROL	Thread	Control Features in Intel 64 Processor (R/W) See Table Table 35-2.
		0		Lock (R/WL)
		1		Enable VMX inside SMX operation (R/WL)
		2		Enable VMX outside SMX operation (R/WL)
		14:8		SENTER local functions enables (R/WL)
		15		SENTER global functions enable (R/WL)
		20		LMCE_ON (R/WL)
		63:21		Reserved.
179H	377	IA32_MCG_CAP	Thread	Global Machine Check Capability (R/O)
		7:0		Count
		8		MCG_CTL_P
		9		MCG_EXT_P
		10		MCP_CMCI_P
		11		MCG_TES_P
		15:12		Reserved.
		23:16		MCG_EXT_CNT
		24		MCG_SER_P
		25		Reserved.
		26		MCG_ELOG_P
		27		MCG_LMCE_P
		63:28		Reserved.
17AH	378	IA32_MCG_STATUS	Thread	(R/W0)
		0		RIPV

Table 35-19 Additional MSRs Supported by Intel® Xeon Processors E7 v2 Family with DisplayFamily_DisplayModel Signature 06_3EH

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
		1		EIPV
		2		MCIP
		3		LMCE signaled
		63:4		Reserved.
1AEH	430	MSR_TURBO_RATIO_LIMIT1	Package	Maximum Ratio Limit of Turbo Mode RO if MSR_PLATFORM_INFO.[28] = 0, RW if MSR_PLATFORM_INFO.[28] = 1
		7:0	Package	Maximum Ratio Limit for 9C Maximum turbo ratio limit of 9 core active.
		15:8	Package	Maximum Ratio Limit for 10C Maximum turbo ratio limit of 10core active.
		23:16	Package	Maximum Ratio Limit for 11C Maximum turbo ratio limit of 11 core active.
		31:24	Package	Maximum Ratio Limit for 12C Maximum turbo ratio limit of 12 core active.
		39:32	Package	Maximum Ratio Limit for 13C Maximum turbo ratio limit of 13 core active.
		47:40	Package	Maximum Ratio Limit for 14C Maximum turbo ratio limit of 14 core active.
		55:48	Package	Maximum Ratio Limit for 15C Maximum turbo ratio limit of 15 core active.
		63:56		Reserved
29DH	669	IA32_MC29_CTL2	Package	See Table Table 35-2.
29EH	670	IA32_MC30_CTL2	Package	See Table Table 35-2.
29FH	671	IA32_MC31_CTL2	Package	See Table Table 35-2.
41BH	1051	IA32_MC6_MISC	Package	Misc MAC information of Integrated I/O. (R/O) see Section 15.3.2.4
		5:0		Recoverable Address LSB
		8:6		Address Mode
		15:9		Reserved
		31:16		PCI Express Requestor ID
		39:32		PCI Express Segment Number
		63:32		Reserved
474H	1140	MSR_MC29_CTL	Package	See Section 15.3.2.1, "IA32_MCi_CTL MSRs."
475H	1141	MSR_MC29_STATUS	Package	See Section 15.3.2.2, "IA32_MCi_STATUS MSRS," and Chapter 16.

Table 35-19 Additional MSRs Supported by Intel® Xeon Processors E7 v2 Family with DisplayFamily_DisplayModel Signature 06_3EH

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
476H	1142	MSR_MC29_ADDR	Package	See Section 15.3.2.3, "IA32_MCi_ADDR MSRs."
477H	1143	MSR_MC29_MISC	Package	See Section 15.3.2.4, "IA32_MCi_MISC MSRs."
478H	1144	MSR_MC30_CTL	Package	See Section 15.3.2.1, "IA32_MCi_CTL MSRs."
479H	1145	MSR_MC30_STATUS	Package	See Section 15.3.2.2, "IA32_MCi_STATUS MSRS," and Chapter 16.
47AH	1146	MSR_MC30_ADDR	Package	See Section 15.3.2.3, "IA32_MCi_ADDR MSRs."
47BH	1147	MSR_MC30_MISC	Package	See Section 15.3.2.4, "IA32_MCi_MISC MSRs."
47CH	1148	MSR_MC31_CTL	Package	See Section 15.3.2.1, "IA32_MCi_CTL MSRs."
47DH	1149	MSR_MC31_STATUS	Package	See Section 15.3.2.2, "IA32_MCi_STATUS MSRS," and Chapter 16.
47EH	1150	MSR_MC31_ADDR	Package	See Section 15.3.2.3, "IA32_MCi_ADDR MSRs."
47FH	1147	MSR_MC31_MISC	Package	See Section 15.3.2.4, "IA32_MCi_MISC MSRs."

...

35.10 MSRS IN THE 4TH GENERATION INTEL® CORE™ PROCESSORS (BASED ON HASWELL MICROARCHITECTURE)

The 4th generation Intel® Core™ processor family and Intel Xeon processor E3-1200v3 product family (based on Haswell microarchitecture), with CPUID DisplayFamily_DisplayModel signature 06_3CH/06_45H/06_46H, support the MSR interfaces listed in Table 35-14, Table 35-15, Table 35-17, and Table Table 35-20.

Table 35-20 Additional MSRs Supported by 4th Generation Intel® Core Processors (based on Haswell microarchitecture)

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
3BH	59	IA32_TSC_ADJUST	THREAD	Per-Logical-Processor TSC ADJUST (R/W) See Table Table 35-2.
CEH	206	MSR_PLATFORM_INFO	Package	See Table 35-17
186H	390	IA32_PERFEVTSELO	THREAD	Performance Event Select for Counter 0 (R/W) Supports all fields described inTable Table 35-2 and the fields below.
		32		IN_TX: see Section 18.11.5.1 When IN_TX (bit 32) is set, AnyThread (bit 21) should be cleared to prevent incorrect results
187H	391	IA32_PERFEVTSEL1	THREAD	Performance Event Select for Counter 1 (R/W) Supports all fields described inTable Table 35-2 and the fields below.

Table 35-20 Additional MSRs Supported by 4th Generation Intel® Core Processors (based on Haswell microarchitecture) (Contd.)

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
		32		IN_TX: see Section 18.11.5.1 When IN_TX (bit 32) is set, AnyThread (bit 21) should be cleared to prevent incorrect results
188H	392	IA32_PERFEVTSEL2	THREAD	Performance Event Select for Counter 2 (R/W) Supports all fields described inTable Table 35-2 and the fields below.
		32		IN_TX: see Section 18.11.5.1 When IN_TX (bit 32) is set, AnyThread (bit 21) should be cleared to prevent incorrect results
		33		IN_TXCP: see Section 18.11.5.1 When IN_TXCP=1 & IN_TX=1 and in sampling, spurious PMI may occur and transactions may continuously abort near overflow conditions. Software should favor using IN_TXCP for counting over sampling. If sampling, software should use large “sample-after” value after clearing the counter configured to use IN_TXCP and also always reset the counter even when no overflow condition was reported.
189H	393	IA32_PERFEVTSEL3	THREAD	Performance Event Select for Counter 3 (R/W) Supports all fields described inTable Table 35-2 and the fields below.
		32		IN_TX: see Section 18.11.5.1 When IN_TX (bit 32) is set, AnyThread (bit 21) should be cleared to prevent incorrect results
491H	1169	IA32_VMX_FMFUNC	THREAD	Capability Reporting Register of VM-function Controls (R/O) See Table Table 35-2
648H	1608	MSR_CONFIG_TDP_NOMINAL	Package	Nominal TDP Ratio (R/O) See Table 35-17
649H	1609	MSR_CONFIG_TDP_LEVEL1	Package	ConfigTDP Level 1 ratio and power level (R/O). See Table 35-17
64AH	1610	MSR_CONFIG_TDP_LEVEL2	Package	ConfigTDP Level 2 ratio and power level (R/O). See Table 35-17
64BH	1611	MSR_CONFIG_TDP_CONTROL	Package	ConfigTDP Control (R/w) See Table 35-17
64CH	1612	MSR_TURBO_ACTIVATION_RATIO	Package	ConfigTDP Control (R/w) See Table 35-17
690H	1680	MSR_CORE_PERF_LIMIT_REASONS	Package	Indicator of Frequency Clipping in Processor Cores (R/W) (frequency refers to processor core frequency)
		0		PROCHOT Status (RO) When set, processor core frequency is reduced below the operating system request due to assertion of external PROCHOT.

Table 35-20 Additional MSRs Supported by 4th Generation Intel® Core Processors (based on Haswell microarchitecture) (Contd.)

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
		1		Thermal Status (R0) When set, frequency is reduced below the operating system request due to a thermal event.
		3:2		Reserved.
		4		Graphics Driver Status (R0) When set, frequency is reduced below the operating system request due to Processor Graphics driver override.
		5		Autonomous Utilization-Based Frequency Control Status (R0) When set, frequency is reduced below the operating system request because the processor has detected that utilization is low.
		6		VR Therm Alert Status (R0) When set, frequency is reduced below the operating system request due to a thermal alert from the Voltage Regulator.
		7		Reserved.
		8		Electrical Design Point Status (R0) When set, frequency is reduced below the operating system request due to electrical design point constraints (e.g. maximum electrical current consumption).
		9		Core Power Limiting Status (R0) When set, frequency is reduced below the operating system request due to domain-level power limiting.
		10		Package-Level Power Limiting PL1 Status (R0) When set, frequency is reduced below the operating system request due to package-level power limiting PL1.
		11		Package-Level PL2 Power Limiting Status (R0) When set, frequency is reduced below the operating system request due to package-level power limiting PL2.
		12		Max Turbo Limit Status (R0) When set, frequency is reduced below the operating system request due to multi-core turbo limits.
		13		Turbo Transition Attenuation Status (R0) When set, frequency is reduced below the operating system request due to Turbo transition attenuation. This prevents performance degradation due to frequent operating ratio changes.
		15:14		Reserved
		16		PROCHOT Log When set, indicates that the corresponding PROCHOT Status bit is set. Software can write 0 to this bit to clear PROCHOT Status.

Table 35-20 Additional MSRs Supported by 4th Generation Intel® Core Processors (based on Haswell microarchitecture) (Contd.)

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
		17		Thermal Log When set, indicates that the corresponding Thermal status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Thermal Status.
		19:18		Reserved.
		20		Graphics Driver Log When set, indicates that the corresponding Graphics Driver status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Graphics Driver Status.
		21		Autonomous Utilization-Based Frequency Control Log When set, indicates that the corresponding Autonomous Utilization-Based Frequency Control status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Autonomous Utilization-Based Frequency Control Status.
		22		VR Therm Alert Log When set, indicates that the corresponding VR Therm Alert Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear VR Therm Alert Status.
		23		Reserved.
		24		Electrical Design Point Log When set, indicates that the corresponding EDP Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear EDP Status.
		25		Core Power Limiting Log When set, indicates that the corresponding Core Power Limiting Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Core Power Limiting Status.
		26		Package-Level PL1 Power Limiting Log When set, indicates that the corresponding Package-level Power Limiting PL1 Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Package-level Power Limiting PL1 Status.
		27		Package-Level PL2 Power Limiting Log When set, indicates that the corresponding Package-level Power Limiting PL2 Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Package-level Power Limiting PL2 Status.
		28		Max Turbo Limit Log When set, indicates that the corresponding Max Turbo Limit Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Max Turbo Limit Status.

Table 35-20 Additional MSRs Supported by 4th Generation Intel® Core Processors (based on Haswell microarchitecture) (Contd.)

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
		29		Turbo Transition Attenuation Log When set, indicates that the corresponding Turbo Transition Attenuation Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Turbo Transition Attenuation Status.
		63:30		Reserved.
6B0H	1712	MSR_GRAPHICS_PERF_LIMIT_REASONS	Package	Indicator of Frequency Clipping in the Processor Graphics (R/W) (frequency refers to processor graphics frequency)
		0		PROCHOT Status (R0) When set, frequency is reduced below the operating system request due to assertion of external PROCHOT.
		1		Thermal Status (R0) When set, frequency is reduced below the operating system request due to a thermal event.
		3:2		Reserved.
		4		Graphics Driver Status (R0) When set, frequency is reduced below the operating system request due to Processor Graphics driver override.
		5		Reserved.
		6		VR Therm Alert Status (R0) When set, frequency is reduced below the operating system request due to a thermal alert from the Voltage Regulator.
		7		Reserved.
		8		Electrical Design Point Status (R0) When set, frequency is reduced below the operating system request due to electrical design point constraints (e.g. maximum electrical current consumption).
		9		Graphics Power Limiting Status (R0) When set, frequency is reduced below the operating system request due to domain-level power limiting.
		10		Package-Level Power Limiting PL1 Status (R0) When set, frequency is reduced below the operating system request due to package-level power limiting PL1.
		11		Package-Level PL2 Power Limiting Status (R0) When set, frequency is reduced below the operating system request due to package-level power limiting PL2.
		15:12		Reserved

Table 35-20 Additional MSRs Supported by 4th Generation Intel® Core Processors (based on Haswell microarchitecture) (Contd.)

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
		16		PROCHOT Log When set, indicates that the corresponding PROCHOT Status bit is set. Software can write 0 to this bit to clear PROCHOT Status.
		17		Thermal Log When set, indicates that the corresponding Thermal status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Thermal Status.
		19:18		Reserved.
		20		Graphics Driver Log When set, indicates that the corresponding Graphics Driver status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Graphics Driver Status.
		21		Reserved.
		22		VR Therm Alert Log When set, indicates that the corresponding VR Therm Alert Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear VR Therm Alert Status.
		23		Reserved.
		24		Electrical Design Point Log When set, indicates that the corresponding EDP Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear EDP Status.
		25		Graphics Power Limiting Log When set, indicates that the corresponding Graphics Power Limiting Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Graphics Power Limiting Status.
		26		Package-Level PL1 Power Limiting Log When set, indicates that the corresponding Package-level Power Limiting PL1 Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Package-level Power Limiting PL1 Status.
		27		Package-Level PL2 Power Limiting Log When set, indicates that the corresponding Package-level Power Limiting PL2 Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Package-level Power Limiting PL2 Status.
		63:28		Reserved.
6B1H	1713	MSR_RING_PERF_LIMIT_REASONS	Package	Indicator of Frequency Clipping in the Ring Interconnect (R/W) (frequency refers to ring interconnect in the uncore)

Table 35-20 Additional MSRs Supported by 4th Generation Intel® Core Processors (based on Haswell microarchitecture) (Contd.)

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
		0		PROCHOT Status (R0) When set, frequency is reduced below the operating system request due to assertion of external PROCHOT.
		1		Thermal Status (R0) When set, frequency is reduced below the operating system request due to a thermal event.
		5:2		Reserved.
		6		VR Therm Alert Status (R0) When set, frequency is reduced below the operating system request due to a thermal alert from the Voltage Regulator.
		7		Reserved.
		8		Electrical Design Point Status (R0) When set, frequency is reduced below the operating system request due to electrical design point constraints (e.g. maximum electrical current consumption).
		9		Reserved.
		10		Package-Level Power Limiting PL1 Status (R0) When set, frequency is reduced below the operating system request due to package-level power limiting PL1.
		11		Package-Level PL2 Power Limiting Status (R0) When set, frequency is reduced below the operating system request due to package-level power limiting PL2.
		15:12		Reserved
		16		PROCHOT Log When set, indicates that the corresponding PROCHOT Status bit is set. Software can write 0 to this bit to clear PROCHOT Status.
		17		Thermal Log When set, indicates that the corresponding Thermal status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Thermal Status.
		21:18		Reserved.
		22		VR Therm Alert Log When set, indicates that the corresponding VR Therm Alert Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear VR Therm Alert Status.
		23		Reserved.

Table 35-20 Additional MSRs Supported by 4th Generation Intel® Core Processors (based on Haswell microarchitecture) (Contd.)

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
		24		Electrical Design Point Log When set, indicates that the corresponding EDP Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear EDP Status.
		25		Reserved.
		26		Package-Level PL1 Power Limiting Log When set, indicates that the corresponding Package-level Power Limiting PL1 Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Package-level Power Limiting PL1 Status.
		27		Package-Level PL2 Power Limiting Log When set, indicates that the corresponding Package-level Power Limiting PL2 Status bit was set since it was last cleared by software. Software can write 0 to this bit to clear Package-level Power Limiting PL2 Status.
		63:28		Reserved.
C80H	32	IA32_DEBUG_FEATURE	Package	Silicon Debug Feature Control (R/W) See Table Table 35-2.

...

35.11 MSRS IN NEXT GENERATION INTEL® XEON® PROCESSORS

The following MSRs are available in next generation of Intel® Xeon® Processor Family (CPUID DisplayFamily_DisplayModel = 06_3F) if CPUID.(EAX=07H, ECX=0):EBX.QoS[bit 12] = 1.

Table 35-23 Additional MSRs Supported by Next Generation Intel® Xeon® Processors

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
C8DH	3113	IA32_QM_EVTSEL	THREAD	QoS Monitoring Event Select Register (R/W).
		7:0		EventID (RW)
		31:8		Reserved.
		41:32		RMID (RW)
		63:42		Reserved.
C8EH	3114	IA32_QM_CTR	THREAD	QoS Monitoring Counter Register (R/O).
		61:0		Resource Monitored Data

Table 35-23 Additional MSRs Supported by Next Generation Intel® Xeon® Processors

Register Address		Register Name	Scope	Bit Description
Hex	Dec			
		62		Unavailable: If 1, indicates data for this RMID is not available or not monitored for this resource or RMID.
		63		Error: If 1, indicates and unsupported RMID or event type was written to IA32_PQR_QM_EVTSEL.
C8FH	3115	IA32_PQR_ASSOC	THREAD	QoS Resource Association Register (R/W).
		9:0		RMID
		63: 10		Reserved

...

