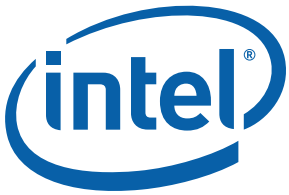

Intel® 64 and IA-32 Architectures Software Developer's Manual

Documentation Changes

August 2008

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: 252046-022

Contents

Revision History

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

Preface

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

Affected Documents

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

Nomenclature

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

Summary Tables of Changes

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

Codes Used in Summary Tables

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

Documentation Changes

No.	DOCUMENTATION CHANGES
1	Extended Control Registers (Chapter 2, Vol 3A)
2	Various Updates Throughout Chapter to Reference MAXPHYADDR as an Address Width (Chapter 3, Vol 3A)
3	Page-Level Protection and Execute-Disable Bit (Chapter 4, Vol 3A)
4	Machine-Check Exceptions (Chapter 5, Vol 3A)
5	Various Updates Throughout Chapter (Chapter 7, Vol 3A)
6	Preventing Caching (Chapter 10, Vol 3A)

Documentation Changes

1. Extended Control Registers (Chapter 2, Vol 3A)

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

2.6 Extended Control Registers (including the XFEATURE_ENABLED_MASK Register)

If CPUID.01H:ECX.XSAVE[bit 26] is 1, the processor supports one or more **extended control registers** (XCRs). Currently, the only such register defined is XCR0, the **XFEATURE_ENABLED_MASK register**. This register specifies the set of processor states that the operating system enables on that processor, e.g. x87 FPU States, SSE states, and other processor extended states that Intel 64 architecture may introduce in the future. The OS programs XCR0 to reflect the features it supports.

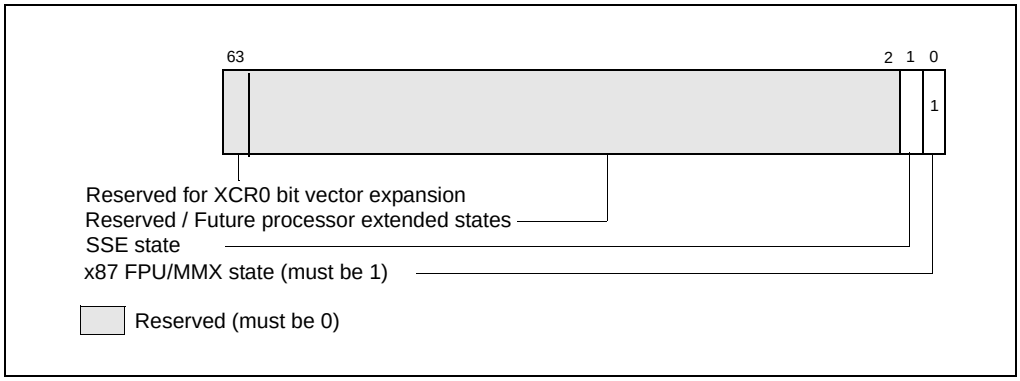

Figure 2-7. XFEATURE_ENABLED_MASK Register (XCR0)

Software can access XCR0 only if CR4.OSXSAVE[bit 18] = 1. (This bit is also readable as CPUID.01H:ECX.OSXSAVE[bit 27].) The layout of XCR0 is architected to allow software to use CPUID leaf function 0DH to enumerate the set of bits that the processor supports in XCR0 (see CPUID instruction in *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2A*). Each processor state (X87 FPU state, SSE state, or a future processor extended state) is represented by a bit in XCR0. The OS can enable future processor extended states in a forward manner by specifying the appropriate bit mask value using the XSETBV instruction according to the results of the CPUID leaf 0DH.

With the exception of bit 63, each bit in the XFEATURE_ENABLED_MASK register (XCR0) corresponds to a subset of the processor states. XCR0 thus provides space for up to 63

sets of processor state extensions. Bit 63 of XCR0 is reserved for future expansion and will not represent a processor extended state.

Currently, the XFEATURE_ENABLED_MASK register (XCR0) has two processor states defined, with up to 61 bits reserved for future processor extended states:

- XCR0.X87 (bit 0): If 1, indicates x87 FPU state (including MMX register states) is supported in the processor. Bit 0 must be 1. An attempt to write 0 causes a #GP exception.
- XCR0.SSE (bit 1): If 1, indicates MXCSR and XMM registers (XMM0-XMM15 in 64-bit mode, otherwise XMM0-XMM7) are supported by XSAVE/XRESTOR in the processor.

Any attempt to set a reserved bit (as determined by the contents of EAX and EDX after executing CPUID with EAX=0DH, ECX= 0H) in the XFEATURE_ENABLED_MASK register for a given processor will result in a #GP exception. An attempt to write 0 to XFEATURE_ENABLED_MASK.x87 (bit 0) will result in a #GP exception.

If a bit in the XFEATURE_ENABLED_MASK register is 1, XSAVE instruction can selectively (in conjunction with a save mask) save a partial or full set of processor states to memory (See XSAVE instruction in *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 2B*).

After reset all bits (except bit 0) in the XFEATURE_ENABLED_MASK register (XCR0) are cleared to zero. XCR0[0] is set to 1.

2. Various Updates Throughout Chapter to Reference MAXPHYADDR as an Address Width (Chapter 3, Vol 3A)

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

3.6.1 Paging Options

Paging is controlled by three flags in the processor's control registers:

- **PG (paging) flag.** Bit 31 of CR0 (available in all IA-32 processors beginning with the Intel386 processor).
- **PSE (page size extensions) flag.** Bit 4 of CR4 (introduced in the Pentium processor).
- **PAE (physical address extension) flag.** Bit 5 of CR4 (introduced in the Pentium Pro processors).

The PG flag enables the page-translation mechanism. The operating system or executive usually sets this flag during processor initialization. The PG flag must be set if the processor's page-translation mechanism is to be used to implement a demand-paged virtual memory system or if the operating system is designed to run more than one program (or task) in virtual-8086 mode.

The PSE flag enables large page sizes: 4-MByte pages or 2-MByte pages (when the PAE flag is set). When the PSE flag is clear, the more common page length of 4 KBytes is used. See Section 3.7.2, "Linear Address Translation (4-MByte Pages)", Section 3.8.3, "Linear Address Translation With PAE Enabled (2-MByte Pages)", and Section 3.9, "36-Bit Physical Addressing Using the PSE-36 Paging Mechanism" for more information about the use of the PSE flag.

The PAE flag provides a method of extending physical addresses to an extended physical address width, MAXPHYADDR.

NOTE

The MAXPHYADDR is 36 bits for processors that do not support CPUID leaf 80000008H, or indicated by CPUID.80000008H:EAX[bits 7:0] for processors that support CPUID leaf 80000008H.

This physical address extension can only be used when paging is enabled. It relies on an additional page directory pointer table that is used along with page directories and page tables to reference physical addresses above FFFFFFFFH. See Section 3.8, “36-Bit Physical Addressing Using the PAE Paging Mechanism”, for more information about extending physical addresses using the PAE flag.

When PAE flag is enabled for Intel 64 processors, the PAE mechanism is enhanced to support more than 36 bits of physical addressing (if the processor’s implementation supports more than 36 bits of physical addressing). This applies to IA-32e mode address translation (see Section 3.10, “PAE-Enabled Paging in IA-32e Mode”) and enhanced legacy PAE-enabled address translation (see Section 3.8.1, “Enhanced Legacy PAE Paging”).

The 36-bit page size extension (PSE-36) feature provides an alternate method of extending physical addressing to 36 bits. This paging mechanism uses the page size extension mode (enabled with the PSE flag) and modified page directory entries to reference physical addresses above FFFFFFFFH. The PSE-36 feature flag (bit 17 in the EDX register when the CPUID instruction is executed with a source operand of 1) indicates the availability of this addressing mechanism. See Section 3.9, “36-Bit Physical Addressing Using the PSE-36 Paging Mechanism”, for more information about the PSE-36 physical address extension and page size extension mechanism.

...

3.7 Page Translation using 32-Bit Physical Addressing

The following sections describe the IA-32 architecture’s page translation mechanism when using 32-bit physical addresses and a maximum physical address space of 4 GBytes. The 32-bit physical addressing described applies to IA-32 processors or when the following situations are all true:

- The processor supports Intel 64 architecture but IA-32e mode is not active.
- PAE or PSE mechanism is not active.

Section 3.8, “36-Bit Physical Addressing Using the PAE Paging Mechanism” and Section 3.9, “36-Bit Physical Addressing Using the PSE-36 Paging Mechanism” describe extensions to this page translation mechanism to support 36-bit physical addresses and a maximum physical address space of 64 GBytes.

Table 3-3. Page Sizes and Physical Address Sizes

PG Flag, CR0	PAE Flag, CR4	PSE Flag, CR4	PS Flag, PDE	PSE-36 CPUID Feature Flag	Page Size	Physical Address Size
0	X	X	X	X	—	Paging Disabled
1	0	0	X	X	4 KBytes	32 Bits
1	0	1	0	X	4 KBytes	32 Bits
1	0	1	1	0	4 MBytes	32 Bits
1	0	1	1	1	4 MBytes	36 Bits
1	1	X	0	X	4 KBytes	MAXPHYADDR[a]
1	1	X	1	X	2 MBytes	MAXPHYADDR[1]

NOTES:

a. MAXPHYADDR is 36-bits on processors that do not support CPUID.80000008H leaf. On processors that do support CPUID.80000008H, MAXPHYADDR is implementation-specific and indicated by CPUID.80000008H:EAX[bits 7:0].

...

3.7.6 Page-Directory and Page-Table Entries

Figure 3-14 shows the format for the page-directory and page-table entries when 4-KByte pages and 32-bit physical addresses are being used. Figure 3-15 shows the format for the page-directory entries when 4-MByte pages and 32-bit physical addresses are being used. The functions of the flags and fields in the entries in Figures 3-14 and 3-15 are as follows:

Page base address, bits 12 through 32

(Page-table entries for 4-KByte pages) — Specifies the physical address of the first byte of a 4-KByte page. The bits in this field are interpreted as the 20 most-significant bits of the physical address, which forces pages to be aligned on 4-KByte boundaries.

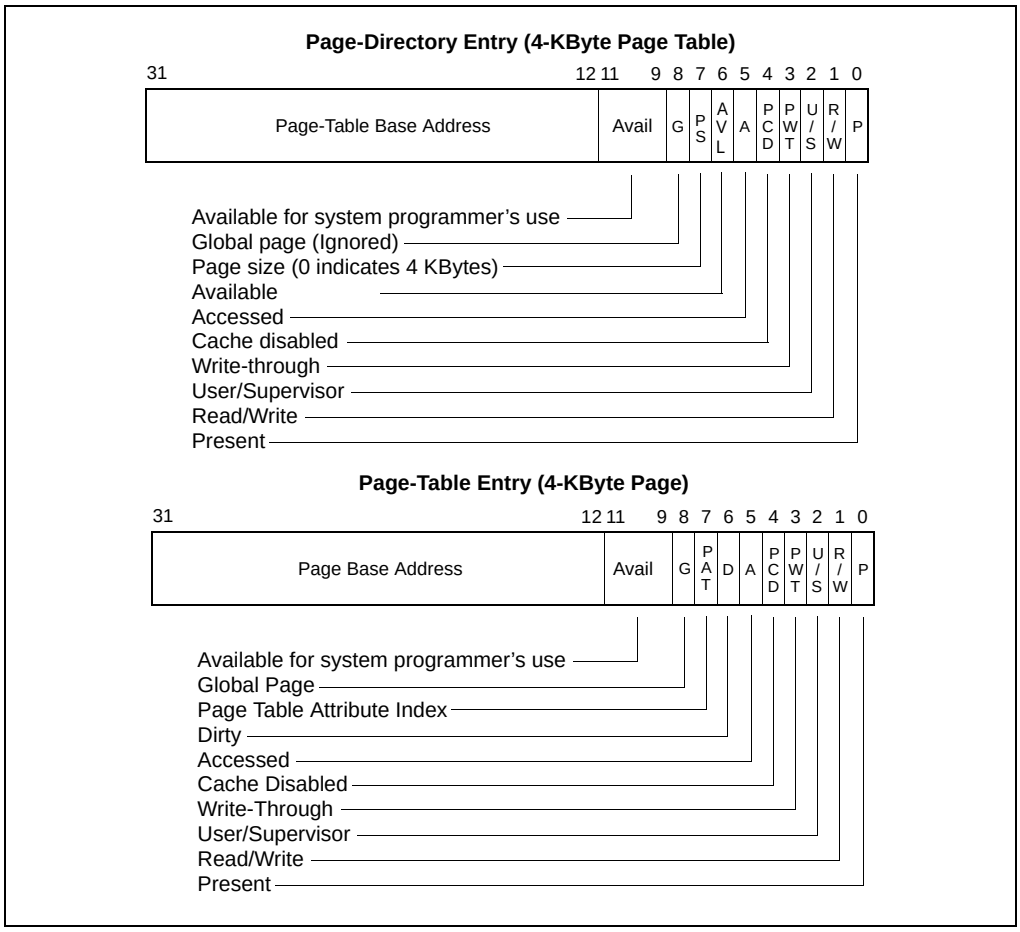

Figure 3-14. Format of Page-Directory and Page-Table Entries for 4-KByte Pages and 32-Bit Physical Addresses

(Page-directory entries for 4-KByte page tables) — Specifies the physical address of the first byte of a page table. The bits in this field are interpreted as the 20 most-significant bits of the physical address, which forces page tables to be aligned on 4-KByte boundaries.

(Page-directory entries for 4-MByte pages) — Specifies the physical address of the first byte of a 4-MByte page. Only bits 22 through 31 of this field are used (and bits 12 through 21 are reserved and must be set to 0, for IA-32 processors through the Pentium II processor). The base address bits are interpreted as the 10 most-significant bits of the physical address, which forces 4-MByte pages to be aligned on 4-MByte boundaries.

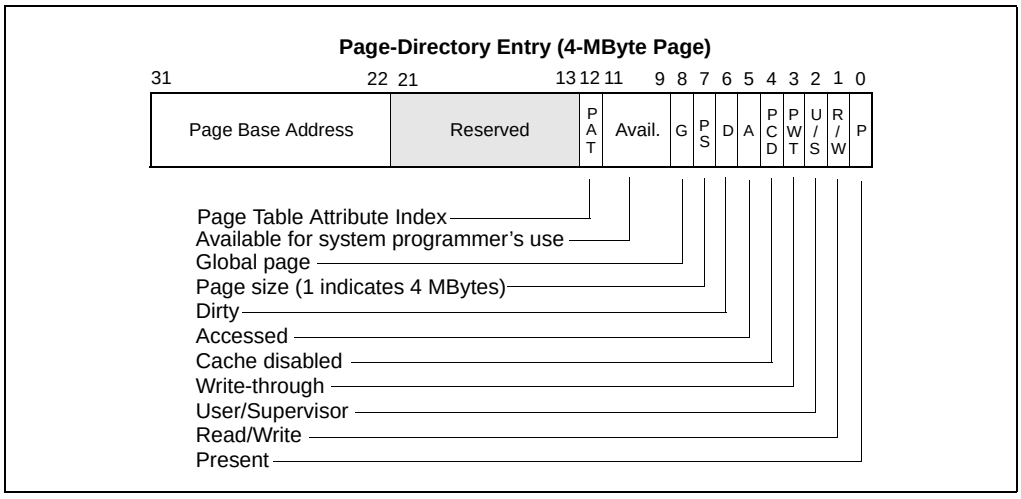

Figure 3-15. Format of Page-Directory Entries for 4-MByte Pages and 32-Bit Addresses

Present (P) flag, bit 0

Indicates whether the page or page table being pointed to by the entry is currently loaded in physical memory. When the flag is set, the page is in physical memory and address translation is carried out. When the flag is clear, the page is not in memory and, if the processor attempts to access the page, it generates a page-fault exception (#PF).

The processor does not set or clear this flag; it is up to the operating system or executive to maintain the state of the flag.

If the processor generates a page-fault exception, the operating system generally needs to carry out the following operations:

1. Copy the page from disk storage into physical memory.
2. Load the page address into the page-table or page-directory entry and set its present flag. Other flags, such as the dirty and accessed flags, may also be set at this time.
3. Invalidate the current page-table entry in the TLB (see Section 3.12, "Translation Lookaside Buffers (TLBs)", for a discussion of TLBs and how to invalidate them).
4. Return from the page-fault handler to restart the interrupted program (or task).

Read/write (R/W) flag, bit 1

Specifies the read-write privileges for a page or group of pages (in the case of a page-directory entry that points to a page table). When this flag is clear, the page is read only; when the flag is set, the page can be read and written into. This flag interacts with the U/S flag and the WP flag in register CR0. See Section 4.11, "Page-Level Protection", and Table 4-3 for a detailed discussion of the use of these flags.

User/supervisor (U/S) flag, bit 2

Specifies the user-supervisor privileges for a page or group of pages (in the case of a page-directory entry that points to a page table). When this flag is clear, the page is assigned the supervisor privilege level; when the flag is set, the page is assigned the user privilege level. This flag interacts

with the R/W flag and the WP flag in register CR0. See Section 4.11, “Page-Level Protection”, and Table 4-3 for a detail discussion of the use of these flags.

Page-level write-through (PWT) flag, bit 3

Controls the write-through or write-back caching policy of individual pages or page tables. When the PWT flag is set, write-through caching is enabled for the associated page or page table; when the flag is clear, write-back caching is enabled for the associated page or page table. The processor ignores this flag if the CD (cache disable) flag in CR0 is set. See Section 10.5, “Cache Control”, for more information about the use of this flag. See Section 2.5, “Control Registers”, for a description of a companion PWT flag in control register CR3.

Page-level cache disable (PCD) flag, bit 4

Controls the caching of individual pages or page tables. When the PCD flag is set, caching of the associated page or page table is prevented; when the flag is clear, the page or page table can be cached. This flag permits caching to be disabled for pages that contain memory-mapped I/O ports or that do not provide a performance benefit when cached. The processor ignores this flag (assumes it is set) if the CD (cache disable) flag in CR0 is set. See Chapter 10, “Memory Cache Control”, for more information about the use of this flag. See Section 2.5, “Control Registers”, for a description of a companion PCD flag in control register CR3.

Accessed (A) flag, bit 5

Indicates whether a page or page table has been accessed (read from or written to) when set. Memory management software typically clears this flag when a page or page table is initially loaded into physical memory. The processor then sets this flag the first time a page or page table is accessed.

This flag is a “sticky” flag, meaning that once set, the processor does not implicitly clear it. Only software can clear this flag. The accessed and dirty flags are provided for use by memory management software to manage the transfer of pages and page tables into and out of physical memory.

NOTE: The accesses used by the processor to set this bit may or may not be exposed to the processor’s Self-Modifying Code detection logic. If the processor is executing code from the same memory area that is being used for page table structures, the setting of the bit may or may not result in an immediate change to the executing code stream.

Dirty (D) flag, bit 6

Indicates whether a page has been written to when set. (This flag is not used in page-directory entries that point to page tables.) Memory management software typically clears this flag when a page is initially loaded into physical memory. The processor then sets this flag the first time a page is accessed for a write operation.

This flag is “sticky,” meaning that once set, the processor does not implicitly clear it. Only software can clear this flag. The dirty and accessed flags are provided for use by memory management software to manage the transfer of pages and page tables into and out of physical memory.

NOTE: The accesses used by the processor to set this bit may or may not be exposed to the processor’s Self-Modifying Code detection logic. If the processor is executing code from the same memory area that is being

used for page table structures, the setting of the bit may or may not result in an immediate change to the executing code stream.

Page size (PS) flag, bit 7 page-directory entries for 4-KByte pages

Determines the page size. When this flag is clear, the page size is 4 KBytes and the page-directory entry points to a page table. When the flag is set, the page size is 4 MBytes for normal 32-bit addressing (and 2 MBytes if extended physical addressing is enabled) and the page-directory entry points to a page. If the page-directory entry points to a page table, all the pages associated with that page table will be 4-KByte pages.

Page attribute table index (PAT) flag, bit 7 in page-table entries for 4-KByte pages and bit 12 in page-directory entries for 4-MByte pages

(Introduced in the Pentium III processor) — Selects PAT entry. For processors that support the page attribute table (PAT), this flag is used along with the PCD and PWT flags to select an entry in the PAT, which in turn selects the memory type for the page (see Section 10.12, “Page Attribute Table (PAT)”). For processors that do not support the PAT, this bit is reserved and should be set to 0.

Global (G) flag, bit 8

(Introduced in the Pentium Pro processor) — Indicates a global page when set. When a page is marked global and the page global enable (PGE) flag in register CR4 is set, the page-table or page-directory entry for the page is not invalidated in the TLB when register CR3 is loaded or a task switch occurs. This flag is provided to prevent frequently used pages (such as pages that contain kernel or other operating system or executive code) from being flushed from the TLB. Only software can set or clear this flag. For page-directory entries that point to page tables, this flag is ignored and the global characteristics of a page are set in the page-table entries. See Section 3.12, “Translation Lookaside Buffers (TLBs)”, for more information about the use of this flag. (This bit is reserved in Pentium and earlier IA-32 processors.)

Reserved and available-to-software bits

For all IA-32 processors. Bits 9, 10, and 11 are available for use by software. (When the present bit is clear, bits 1 through 31 are available to software, see Figure 3-16.) In a page-directory entry that points to a page table, bit 6 is reserved and should be set to 0. When the PSE and PAE flags in control register CR4 are set, the processor generates a page fault if reserved bits are not set to 0.

For Pentium II and earlier processors. Bit 7 in a page-table entry is reserved and should be set to 0. For a page-directory entry for a 4-MByte page, bits 12 through 21 are reserved and must be set to 0.

For Pentium III and later processors. For a page-directory entry for a 4-MByte page, bits 13 through 21 are reserved and must be set to 0.

...

3.8 36-BIT PHYSICAL ADDRESSING USING THE PAE PAGING MECHANISM

The PAE paging mechanism and support for 36-bit physical addressing were introduced into the IA-32 architecture in the Pentium Pro processors. Implementation of this feature in an IA-32 processor is indicated with CPUID feature flag PAE (bit 6 in the EDX register when the source operand for the CPUID instruction is 2). The physical address extension (PAE) flag in register CR4 enables the PAE mechanism and extends physical addresses from 32 bits to 36 bits (or to MAXPHYADDR bits). Here, the processor provides additional address line pins (4 for 36-bit physical addressing) to accommodate the additional address bits. To use this option, the following flags must be set:

- PG flag (bit 31) in control register CR0—Enables paging
- PAE flag (bit 5) in control register CR4 are set—Enables the PAE paging mechanism.

When the PAE paging mechanism is enabled, the processor supports two sizes of pages: 4-KByte and 2-MByte. As with 32-bit addressing, both page sizes can be addressed within the same set of paging tables (that is, a page-directory entry can point to either a 2-MByte page or a page table that in turn points to 4-KByte pages). To support extended physical addresses, the following changes are made to the paging data structures:

- The paging table entries are increased to 64 bits to accommodate 36-bit base physical addresses. Each 4-KByte page directory and page table can thus have up to 512 entries.
- A new table, called the page-directory-pointer table, is added to the linear-address translation hierarchy. This table has 4 entries of 64-bits each, and it lies above the page directory in the hierarchy. With the physical address extension mechanism enabled, the processor supports up to 4 page directories.
- The 20-bit page-directory base address field in register CR3 (PDBR) is replaced with a 27-bit page-directory-pointer-table base address field. The updated field provides the 27 most-significant bits of the physical address of the first byte of the page-directory pointer table (forcing the table to be located on a 32-byte boundary).

  Since CR3 now contains the page-directory-pointer-table base address, it can be referred to as the page-directory-pointer-table register (PDPTR). See Figure 3-17.
- Linear address translation is changed to allow mapping 32-bit linear addresses into the larger physical address space.

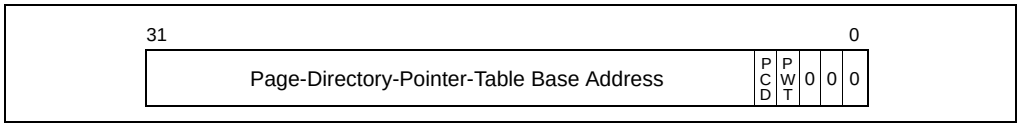

Figure 3-17. Register CR3 Format When the Physical Address Extension is Enabled

...

3.8.4 Accessing the Full Extended Physical Address Space With the Extended Page-Table Structure

The page-table structure described in the previous two sections allows up to 4 GBytes of the extended physical address space to be addressed at one time. Additional 4-GByte sections of physical memory can be addressed in either of two way:

- Change the pointer in register CR3 to point to another page-directory-pointer table, which in turn points to another set of page directories and page tables.
- Change entries in the page-directory-pointer table to point to other page directories, which in turn point to other sets of page tables.

3.8.5 Page-Directory and Page-Table Entries With Extended Addressing Enabled

Figure 3-20 shows the format for the page-directory-pointer-table, page-directory, and page-table entries when 4-KByte pages and extended physical addresses are being used. Figure shows the format for the page-directory-pointer-table and page-directory entries when 2-MByte pages and extended physical addresses are being used. The functions of the flags in these entries are the same as described in Section 3.7.6, “Page-Directory and Page-Table Entries”. The major differences in these entries are as follows:

- A page-directory-pointer-table entry is added.
- The size of the entries are increased from 32 bits to 64 bits.
- The maximum number of entries in a page directory or page table is 512.
- The base physical address field in each entry is extended to 24 bits for 36-bit physical addressing (or extended to MAXPHYADDR-12 bits if MAXPHYADDR is different than 36).

NOTE

Older IA-32 processors that implement the PAE mechanism use uncached accesses when loading page-directory-pointer table entries. This behavior is model specific and not architectural. More recent Intel 64 and IA-32 processors may cache page-directory-pointer table entries.

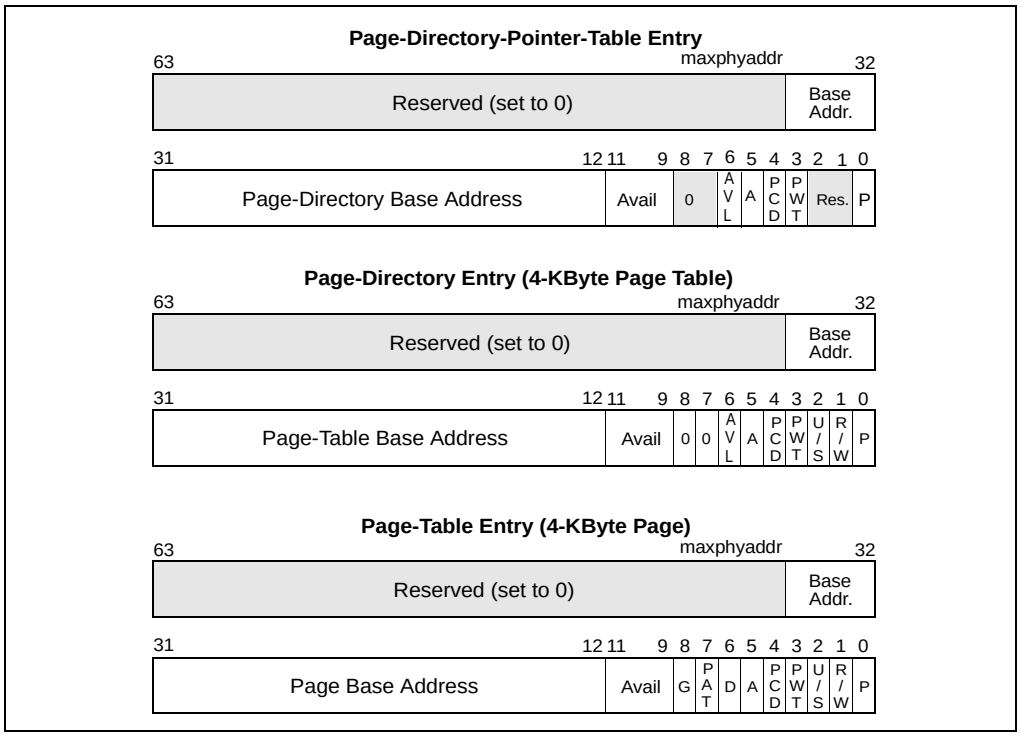

Figure 3-20. Format of Page-Directory-Pointer-Table, Page-Directory, and Page-Table Entries for 4-KByte Pages with PAE Enabled

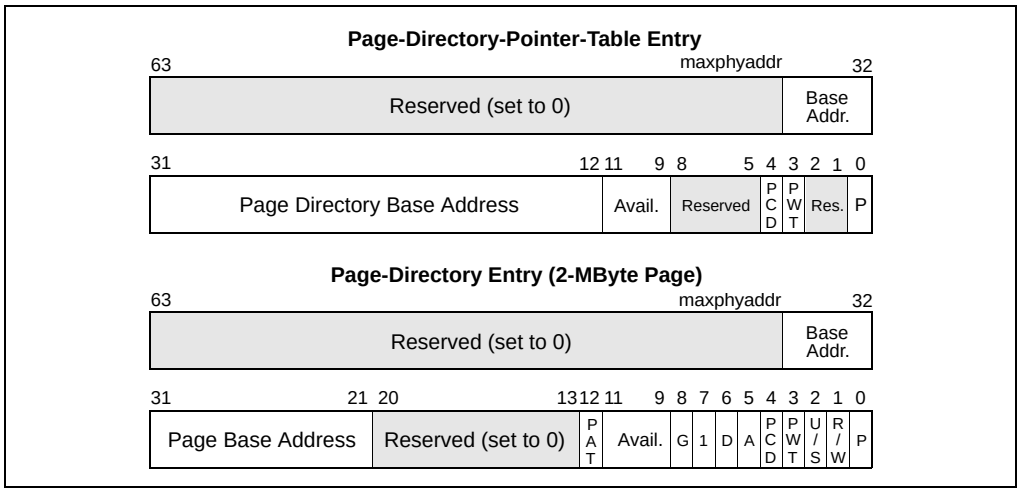

Figure 3-21. Format of Page-Directory-Pointer-Table and Page-Directory Entries for 2-MByte Pages with PAE Enabled

The base physical address in an entry specifies the following, depending on the type of entry:

- **Page-directory-pointer-table entry** — the physical address of the first byte of a 4-KByte page directory.
- **Page-directory entry** — the physical address of the first byte of a 4-KByte page table or a 2-MByte page.
- **Page-table entry** — the physical address of the first byte of a 4-KByte page.

For all table entries (except for page-directory entries that point to 2-MByte pages), the bits in the page base address are interpreted as the 24 most-significant bits of a 36-bit physical address, or the (MAXPHYADDR - 12) most significant bits of the extended physical address width. Thus 4-KByte page tables and pages are aligned on 4-KByte boundaries. When a page-directory entry points to a 2-MByte page, the base address is interpreted as the 15 most-significant bits of a 36-bit physical address, or the (MAXPHYADDR - 21) most significant bits of the extended physical address width. Thus 2-MByte pages are aligned on 2-MByte boundaries.

The present flag (bit 0) in the page-directory-pointer-table entries can be set to 0 or 1. If the present flag is clear, the remaining bits in the page-directory-pointer-table entry are available to the operating system. If the present flag is set, the fields of the page-directory-pointer-table entry are defined in Figures 3-20 for 4-KByte pages and Figures for 2-MByte pages.

The page size (PS) flag (bit 7) in a page-directory entry determines if the entry points to a page table or a 2-MByte page. When this flag is clear, the entry points to a page table; when the flag is set, the entry points to a 2-MByte page. This flag allows 4-KByte and 2-MByte pages to be mixed within one set of paging tables.

Access (A) and dirty (D) flags (bits 5 and 6) are provided for table entries that point to pages.

Bits 9, 10, and 11 in all the table entries for the physical address extension are available for use by software. (When the present flag is clear, bits 1 through 63 are available to software.) All bits in Figure 3-14 that are marked reserved or 0 should be set to 0 by software and not accessed by software. When the PSE and/or PAE flags in control register CR4 are set, the processor generates a page fault (#PF) if reserved bits in page-directory and page-table entries are not set to 0, and it generates a general-protection exception (#GP) if reserved bits in a page-directory-pointer-table entry are not set to 0.

3.9 36-Bit Physical Addressing Using the PSE-36 Paging Mechanism

The PSE-36 paging mechanism provides an alternate method (from the PAE mechanism) of extending physical memory addressing to 36 bits. This mechanism uses the page size extension (PSE) mode and a modified page-directory table to map 4-MByte pages into a 64-GByte physical address space. The processor provides 4 additional address line pins to accommodate the additional address bits.

The PSE-36 mechanism was introduced into the IA-32 architecture with the Pentium III processors. The availability of this feature is indicated with the PSE-36 feature bit (bit 17 of the EDX register when the CPUID instruction is executed with a source operand of 1).

As is shown in Table 3-3, the following flags must be set or cleared to enable the PSE-36 paging mechanism:

- **PSE-36 CPUID feature flag** — When set, it indicates the availability of the PSE-36 paging mechanism on the IA-32 processor on which the CPUID instruction is executed.
- **PG flag (bit 31) in register CR0** — Set to 1 to enable paging.
- **PAE flag (bit 5) in control register CR4** — Clear to 0 to disable the PAE paging mechanism.
- **PSE flag (bit 4) in control register CR4 and the PS flag in PDE** — Set to 1 to enable the page size extension for 4-MByte pages.
- **Or the PSE flag (bit 4) in control register CR4** — Set to 1 and the PS flag (bit 7) in PDE— Set to 0 to enable 4-KByte pages with 32-bit addressing (below 4 GBytes).

Figure 3-22 shows how the expanded page directory entry can be used to map a 32-bit linear address to a 36-bit physical address. Here, the linear address is divided into two sections:

- **Page directory entry** — Bits 22 through 35 provide an offset to an entry in the page directory. The selected entry provides the 14 most significant bits of a 36-bit address, which locates the base physical address of a 4-MByte page.
- **Page offset** — Bits 0 through 21 provides an offset to a physical address in the page.

This paging method can be used to map up to 1024 pages into a 64-GByte physical address space.

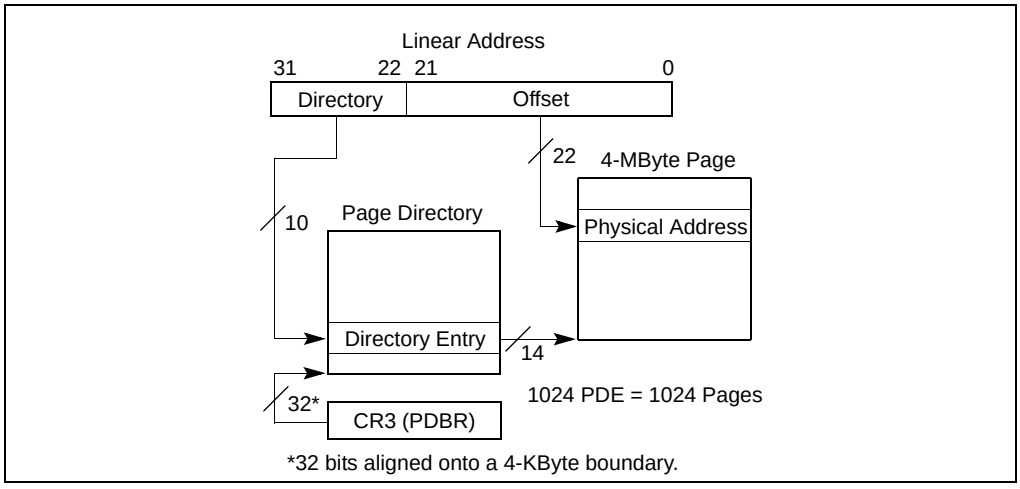

Figure 3-22. Linear Address Translation (4-MByte Pages)

Figure shows the format for the page-directory entries when 4-MByte pages and 36-bit physical addresses are being used. Section 3.7.6, “Page-Directory and Page-Table Entries” describes the functions of the flags and fields in bits 0 through 11.

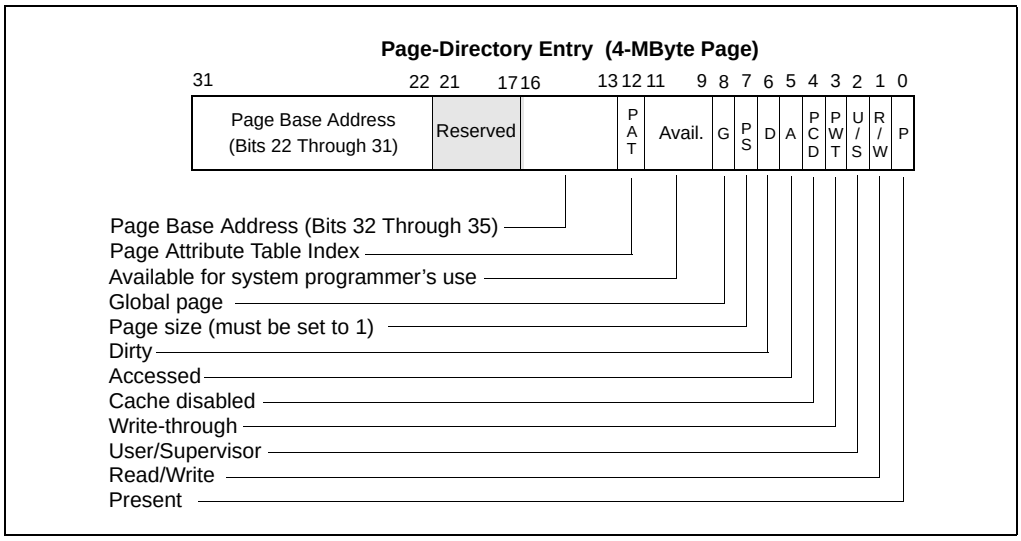

Figure 3-23. Format of Page-Directory Entries for 4-MByte Pages and 36-Bit Physical Addresses

3.10 PAE-ENABLED PAGING IN IA-32E MODE

Intel 64 architecture expands physical address extension (PAE) paging structures to potentially support mapping a 64-bit linear address to a 52-bit physical address. In the first implementation of Intel 64 architecture, PAE paging structures support translation of a 48-bit linear address into a 40-bit physical address. Other implementations may support different physical address sizes. The physical address width, MAXPHYADDR, is reported by CPUID.80000008H:EAX[bits 7:0].

When IA-32e mode is enabled, linear address to physical address translation is different than in PAE-enabled protected mode. Address translation from a linear address to a physical address uses up to four levels of paging data structures. A new page mapping table, the page map level 4 table (PML4 table), is added on top of the page director pointer table.

Prior to activating IA-32e mode, PAE must be enabled by setting CR4.PAE = 1. PAE expands the size of page-directory entries (PDE) and page-table entries (PTE) from 32 bits to 64 bits. This change is made to support physical-address sizes of greater than 32 bits. An attempt to activate IA-32e mode prior to enabling PAE results in a general-protection exception, #GP.

PML4 tables are used in page translation only in IA-32e mode. They are not used when IA-32e mode is disabled, whether or not PAE is enabled. The existing page-directory pointer table is expanded to 512 eight-byte entries from four entries. As a result, nine bits of the linear address are used to index into a PDP table rather than two bits. The size of the page-directory entry (PDE) table and page-table entry (PTE) table remains 512 eight-byte entries, each indexed by nine linear-address bits. The total of linear-address index bits into the collection of paging data structures (PML4 + PDP + PDE + PTE + page offset) becomes 48. The method for translating the high-order 16 linear-address bits into a physical address is currently reserved.

The PS flag in the page directory entry (PDE.PS) selects between 4-KByte and 2-MByte page sizes. Because PDE.PS is used to control large page selection, the CR4.PSE bit is ignored.

...

3.10.3 Enhanced Paging Data Structures

Figure shows the format for the PML4 table, page-directory-pointer table, page-directory and page-table entries when 4-KByte pages are used in IA-32e mode. Figure 3-27 shows the format for the PML4 table, the page-directory-pointer table and page-directory entries when 2-MByte pages are used in IA-32e mode.

Except for the PML4 table; enhanced formats of page-directory-pointer table, page-directory, and page-table entries are also used in enhanced legacy PAE-enabled paging on processors that support Intel 64 architecture (see Section 3.8.1, “Enhanced Legacy PAE Paging”).

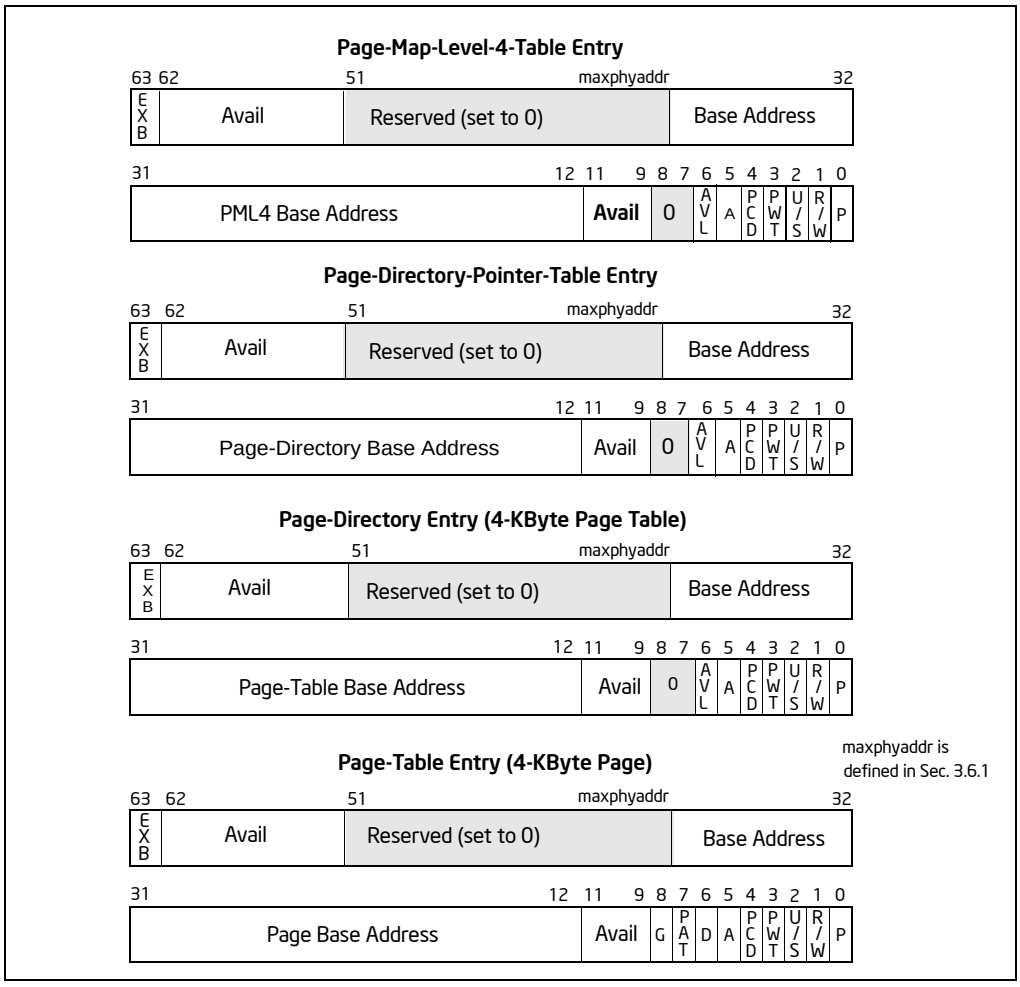

Figure 3-26. Format of Paging Structure Entries for 4-KByte Pages in IA-32e Mode

Except for bit 63, functions of the flags in these entries are as described in Section 3.7.6, “Page-Directory and Page-Table Entries”. The differences are:

- A PML4 table entry and a page-directory-pointer-table entry are added.
- Entries are increased from 32 bits to 64 bits.
- The maximum number of entries in a page directory, page table, or PML4 table is 512.
- The P, R/W, U/S, PWT, PCD, and A flags are implemented uniformly across all four levels.
- The base physical address field in each entry is extended to (MAXPHYADDR-12) bits (28 bits if the processor’s implementation supports a 40-bit physical address).
- Bits 62:52 are available for use by system programmers.
- Bit 63 is the execute-disable bit if the execute-disable bit feature is supported in the processor. If the feature is not supported, bit 63 is reserved. The functionality of the execute disable bit is described in Section 4.11, “Page-Level Protection”. It requires both PAE and enhanced paging data structures. Note that the execute disable bit can provide page protection in 32-bit PAE mode and IA-32e mode.

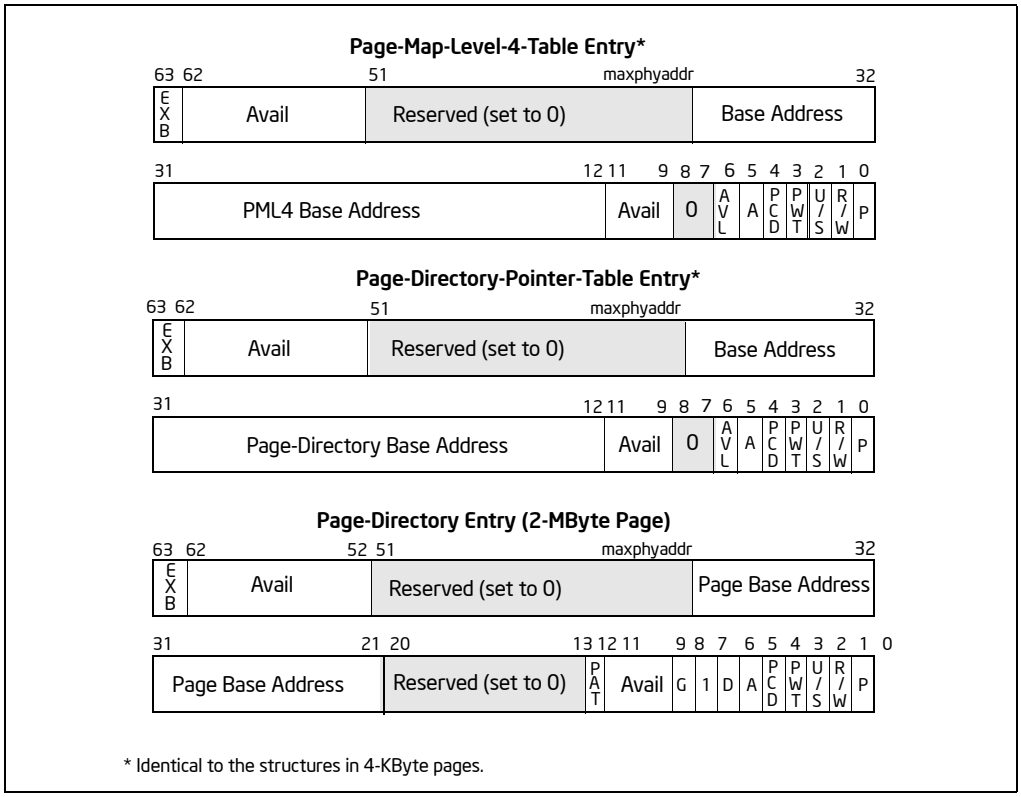

Figure 3-27. Format of Paging Structure Entries for 2-MByte Pages in IA-32e Mode

3.10.3.1 Intel® 64 Processors and Reserved Bit Checking

On processors supporting Intel 64 architecture and/or supporting the execute disable bit, the processor enforces reserved bit checking on paging mode specific bits.

Table 3-4 shows the reserved bits that are checked on Intel 64 processors when execute disable bit checking is either disabled or not supported. The 32-bit mode behavior in Table 3-4 also applies to IA-32 processors that support the execute-disable bit but not Intel 64 architecture.

If the execute disable bit is enabled in an IA-32 or Intel 64 processor, reserved bits in paging data structures for legacy 32-bit mode and 64-bit mode are shown in Table 3-5.

Mode	Paging Mode	Paging Structure	Check Bits
32-bit	4-KByte pages (PAE = 0, PSE = 0)	PDE and PT	No reserved bits checked
	4-MByte page (PAE = 0, PSE = 1)	PDE	Bit [21]
	4-KByte page (PAE = 0, PSE = 1)	PDE	No reserved bits checked
	4-KByte and 4-MByte page (PAE = 0, PSE = 1)	PTE	No reserved bits checked
	4-KByte and 2-MByte pages (PAE = 1, PSE = x)	PDP table entry	Bits [63:MAXPHYADDR] & [8:5] & [2:1]
	2-MByte page (PAE = 1, PSE = x)	PDE	Bits [63:MAXPHYADDR] & [20:13]
	4-KByte pages (PAE =1, PSE = x)	PDE	Bits [63:MAXPHYADDR]
	4-KByte and 2-MByte pages (PAE = 1, PSE = x)	PTE	Bits [63:MAXPHYADDR]
64-bit	4-KByte and 2-MByte pages (PAE = 1, PSE = x)	PML4E	Bit [63], bits [51:MAXPHYADDR], bits [8:7]
	4-KByte and 2-MByte pages (PAE = 1, PSE = x)	PDPTE	Bit [63], bits [51:MAXPHYADDR], bits [8:7]
	2-MByte page (PAE =1, PSE = x)	PDE, 2-MByte page	Bit [63], bits [51:MAXPHYADDR] & [20:13]
	4-KByte pages (PAE = 1, PSE = x)	PDE, 4-KByte page	Bit [63], bits [51:MAXPHYADDR]
	4-KByte and 2-MByte pages (PAE = 1, PSE = x)	PTE	Bit [63], bits [51:MAXPHYADDR]

Table 3-5. Reserved Bit Checking When Execute Disable Bit is Enabled

Mode	Paging Mode	Paging Structure	Check Bits
32-bit	4-KByte pages (PAE = 0, PSE = 0)	PDE and PT	No reserved bits checked
	4-MByte page (PAE = 0, PSE = 1)	PDE	Bit [21]
	4-KByte page (PAE = 0, PSE = 1)	PDE	No reserved bits checked
	4-KByte and 4-MByte page (PAE = 0, PSE = 1)	PTE	No reserved bits checked
	4-KByte and 2-MByte pages (PAE = 1, PSE = x)	PDP table entry	Bits [63:MAXPHYADDR] & [8:5] & [2:1]
	2-MByte page (PAE = 1, PSE = x)	PDE	Bits [62:MAXPHYADDR] & [20:13]
	4-KByte pages (PAE = 1, PSE = x)	PDE	Bits [62:MAXPHYADDR]
	4-KByte pages (PAE = 1, PSE = x)	PTE	Bits [62:MAXPHYADDR]
64-bit	4-KByte and 2-MByte pages (PAE = 1, PSE = x)	PML4E	Bits [51:MAXPHYADDR], bits [8:7]
	4-KByte and 2-MByte pages (PAE = 1, PSE = x)	PDPTE	Bits [51:MAXPHYADDR], bits [8:7]
	2-MByte page (PAE = 1, PSE = x)	PDE, 2-MByte page	Bits [51:MAXPHYADDR] & [20:13]
	4-KByte pages (PAE = 1, PSE = x)	PDE, 4-KByte page	Bits [51:MAXPHYADDR]
	4-KByte pages (PAE = 1, PSE = x)	PTE	Bits [51:MAXPHYADDR]

NOTE:
x = Bit does not impact behavior.

3. Page-Level Protection and Execute-Disable Bit (Chapter 4, Vol 3A)

Change bars show changes to Chapter 4 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide*.

4.13 PAGE-LEVEL PROTECTION AND EXECUTE-DISABLE BIT

In addition to page-level protection offered by the U/S and R/W flags, enhanced PAE-enabled paging structures (see Section 3.10.3, "Enhanced Paging Data Structures") provide the execute-disable bit. This bit offers additional protection for data pages.

An Intel 64 or IA-32 processor with the execute disable bit capability can prevent data pages from being used by malicious software to execute code. This capability is provided in:

- 32-bit protected mode with PAE enabled.
- IA-32e mode.

While the execute disable bit capability does not introduce new instructions, it does require operating systems to use a PAE-enabled environment and establish a page-granular protection policy for memory pages.

If the execute disable bit of a memory page is set, that page can be used only as data. An attempt to execute code from a memory page with the execute-disable bit set causes a page-fault exception.

Only 4-KBytes and 2-MBytes page sizes are supported by the execute-disable bit capability (seeTable 4-4). The execute-disable bit capability is not supported for page size enabled by CR4.PSE. Existing page-level protection mechanisms (see Section 4.11, "Page-Level Protection") continue to apply to memory pages independent of the execute-disable bit setting. The physical address range that can benefit from the execute-disable bit capability start from 0 to 2^MAXPHYADDR (or 2^32 if CPUID.80000008H is not supported). MAXPHYADDR is reported in CPUID.80000008H:EAX[bit 7:0].

Table 4-4. Page Sizes a Supported by Execute-Disable Bit Capability

PG Flag, CR0	PAE Flag, CR4	PS Flag, PDE	Page Size Supported
1	1	0	4 KBytes
1	1	1	2 MBytes

4. Machine-Check Exceptions (Chapter 5, Vol 3A)

Change bars show changes to Chapter 13 of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide*.

5.4.3 Machine-Check Exceptions

The P6 family and Pentium processors provide both internal and external machine-check mechanisms for checking the operation of the internal chip hardware and bus transactions. These mechanisms are implementation dependent. When a machine-check error is detected, the processor signals a machine-check exception (vector 18) and returns an error code.

See Chapter 5, "Interrupt 18—Machine-Check Exception (#MC)" and Chapter 14, "Machine-Check Architecture," for more information about the machine-check mechanism.

5. Debugging and Performance Monitoring (Chapter 7, Vol 3A)

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

7.2.4 Out-of-Order Stores For String Operations

The Intel Core 2 Duo, Intel Core, Pentium 4, and P6 family processors modify the processors operation during the string store operations (initiated with the MOVS and STOS instructions) to maximize performance. Once the "fast string" operations initial conditions are met (as described below), the processor will essentially operate on, from an external perspective, the string in a cache line by cache line mode. This results in the processor looping on issuing a cache-line read for the source address and an invalidation on the external bus for the destination address, knowing that all bytes in the destination cache line will be modified, for the length of the string. In this mode interrupts will only be accepted by the processor on cache line boundaries. It is possible in this mode that the destination line invalidations, and therefore stores, will be issued on the external bus out of order.

Code dependent upon sequential store ordering should not use the string operations for the entire data structure to be stored. Data and semaphores should be separated. Order dependent code should use a discrete semaphore uniquely stored to after any string operations to allow correctly ordered data to be seen by all processors.

"Fast string" operation can be disabled by clearing the fast-string-enable bit (bit 0) of IA32_MISC_ENABLES MSR.

Initial conditions for "fast string" operations are implementation specific. Example conditions include:

- EDI and ESI must be 8-byte aligned for the Pentium III processor. EDI must be 8-byte aligned for the Pentium 4 processor.
- String operation must be performed in ascending address order.
- The initial operation counter (ECX) must be equal to or greater than 64.
- Source and destination must not overlap by less than a cache line (64 bytes, for Intel Core 2 Duo, Intel Core, Pentium M, and Pentium 4 processors; 32 bytes P6 family and Pentium processors).

- The memory type for both source and destination addresses must be either WB or WC.

NOTE

Initial conditions for “fast string“ operation in future Intel 64 or IA-32 processor families may differ from above.

...

7.8.1 State of the Logical Processors

The following features are part of the architectural state of logical processors within Intel 64 or IA-32 processors supporting Hyper-Threading Technology. The features can be subdivided into three groups:

- Duplicated for each logical processor
- Shared by logical processors in a physical processor
- Shared or duplicated, depending on the implementation

The following features are duplicated for each logical processor:

- General purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, ESP, and EBP)
- Segment registers (CS, DS, SS, ES, FS, and GS)
- EFLAGS and EIP registers. Note that the CS and EIP/RIP registers for each logical processor point to the instruction stream for the thread being executed by the logical processor.
- x87 FPU registers (ST0 through ST7, status word, control word, tag word, data operand pointer, and instruction pointer)
- MMX registers (MM0 through MM7)
- XMM registers (XMM0 through XMM7) and the MXCSR register
- Control registers and system table pointer registers (GDTR, LDTR, IDTR, task register)
- Debug registers (DR0, DR1, DR2, DR3, DR6, DR7) and the debug control MSRs
- Machine check global status (IA32_MCG_STATUS) and machine check capability (IA32_MCG_CAP) MSRs
- Thermal clock modulation and ACPI Power management control MSRs
- Time stamp counter MSRs
- Most of the other MSR registers, including the page attribute table (PAT). See the exceptions below.
- Local APIC registers.
- Additional general purpose registers (R8-R15), XMM registers (XMM8-XMM15), control register, IA32_EFER on Intel 64 processors.

The following features are shared by logical processors:

- Memory type range registers (MTRRs)

Whether the following features are shared or duplicated is implementation-specific:

- IA32_MISC_ENABLE MSR (MSR address 1A0H)
- Machine check architecture (MCA) MSRs (except for the IA32_MCG_STATUS and IA32_MCG_CAP MSRs)

Performance monitoring control and counter MSRs

...

7.8.8 IA32_MISC_ENABLE MSR

The IA32_MISC_ENABLE MSR (MSR address 1A0H) is generally shared between the logical processors in a processor core supporting Hyper-Threading Technology. However, some bit fields within IA32_MISC_ENABLES MSR may be duplicated per logical processor. The partition of shared or duplicated bit fields within IA32_MISC_ENABLES is implementation dependent. Software should program duplicated fields carefully on all logical processors in the system to ensure consistent behavior.

...

7.8.13.2 Processor Translation Lookaside Buffers (TLBs)

In processors supporting Hyper-Threading Technology, data cache TLBs are shared. The instruction cache TLB may be duplicated or shared in each logical processor, depending on implementation specifics of different processor families.

Entries in the TLBs are tagged with an ID that indicates the logical processor that initiated the translation. This tag applies even for translations that are marked global using the page global feature for memory paging.

When a logical processor performs a TLB invalidation operation, only the TLB entries that are tagged for that logical processor are guaranteed to be flushed. This protocol applies to all TLB invalidation operations, including writes to control registers CR3 and CR4 and uses of the INVLPG instruction.

...

7.9.3 Performance Monitoring Counters

Performance counters and their companion control MSRs are shared between two logical processors sharing a processor core if the processor core supports Hyper-Threading Technology and is based on Intel NetBurst microarchitecture. They are not shared between logical processors in different cores or different physical packages. As a result, software must manage the use of these resources, based on the topology of performance monitoring resources. Performance counter interrupts, events, and precise event

monitoring support can be set up and allocated on a per thread (per logical processor) basis.

See Section 18.17, “Performance Monitoring and Hyper-Threading Technology in Processors Based on Intel NetBurst Microarchitecture.”

7.9.4 IA32_MISC_ENABLE MSR

Some bit fields in IA32_MISC_ENABLE MSR (MSR address 1A0H) may be shared between two logical processors sharing a processor core, or may be shared between different cores in a physical processor. See Appendix B, “Model-Specific Registers (MSRs)”.

6. Preventing Caching (Chapter 10, Vol 3A)

Change bars show changes to Chapter 10 of the *Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 3A: System Programming Guide*.

7.8.8 Preventing Caching

To disable the L1, L2, and L3 caches after they have been enabled and have received cache fills, perform the following steps:

1. Enter the no-fill cache mode. (Set the CD flag in control register CR0 to 1 and the NW flag to 0.
2. Flush all caches using the WBINVD instruction.
3. Disable the MTRRs and set the default memory type to uncached or set all MTRRs for the uncached memory type (see the discussion of the discussion of the TYPE field and the E flag in Section 10.11.2.1, “IA32_MTRR_DEF_TYPE MSR”).

The caches must be flushed (step 2) after the CD flag is set to insure system memory coherency. If the caches are not flushed, cache hits on reads will still occur and data will be read from valid cache lines.

The intent of the three separate steps listed above address three distinct requirements: (i) discontinue new data replacing existing data in the cache (ii) ensure data already in the cache are evicted to memory, (iii) ensure subsequent memory references observe UC memory type semantics. Different processor implementation of caching control hardware may allow some variation of software implementation of these three requirements. See note below.

NOTES

Setting the CD flag in control register CR0 modifies the processor’s caching behaviour as indicated in Table 10-5, but setting the CD flag alone may not be sufficient across all processor families to force the effective memory type for all physical memory to be UC nor does it force strict memory ordering, due to hardware implementation variations across different processor families. To force the UC memory type and strict memory ordering on all of physical memory, it is sufficient to either program the MTRRs for all physical memory to be UC memory type or disable all MTRRs.

For the Pentium 4 and Intel Xeon processors, after the sequence of steps given above has been executed, the cache lines containing the code

between the end of the WBINVD instruction and before the MTRRS have actually been disabled may be retained in the cache hierarchy. Here, to remove code from the cache completely, a second WBINVD instruction must be executed after the MTRRs have been disabled.

For Intel Atom processors, setting the CD flag forces all physical memory to observe UC semantics (without requiring memory type of physical memory to be set explicitly). Consequently, software does not need to issue a second WBINVD as some other processor generations might require.

§