IA-32 Intel® Architecture Software Developer's Manual

Documentation Changes

September 2003

Notice: The IA-32 Intel® Architecture 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 this specification update.

Document Number: 252046-005

Contents

Revision History

Version	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

Preface

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

Affected Documents/Related Documents

Document Title	Document Number
IA-32 Intel® Architecture Software Developer's Manual: Volume 1, Basic Architecture	245470-011
IA-32 Intel® Architecture Software Developer's Manual: Volume 2, Instruction Set Reference	245471-011
IA-32 Intel® Architecture Software Developer's Manual: Volume 3, System Programming Guide	245472-011

Nomenclature

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

Summary Table of Changes

The following table indicates documentation changes which apply to the IA-32 Intel Architecture. This table uses the following notations:

Codes Used in Summary Table

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

Summary Table of Documentation Changes

Number	DOCUMENTATION CHANGES
1.	IA32_THERM_CONTROL Has Been Changed to IA32_CLOCK_MODULATION
2.	“INTER-PRIVILEGE” Was Not Spelled Correctly in Pseudocode Entry
3.	Confusing Text Artifact Removed
4.	IA32_MISC_CTL has been Removed from the List of Architectural MSRs
5.	Typo Corrected in Figure 8-24
6.	Typo Corrected in Figure 8-23
7.	Corrupted Text Corrected
8.	Corrected an Error in PACKSSDW Illustration
9.	SSM Corrected to SMM
10.	Exiting from SMM Text Updated
11.	L1 Data Cache Context Mode Description Has Been Updated
12.	#DE Should Be #DB in Description of EFLAGS.RF
13.	There Have Been Revisions to the Table That States Priority among Simultaneous Exceptions and Interrupts
14.	Corrections to Page-Directory-Pointer-Table Entry Description
15.	Behavior Notes on the Accessed (A) Flag and Dirty (D) Flag

Documentation Changes

1. IA32_THERM_CONTROL Has Been Changed to IA32_CLOCK_MODULATION

The name of the Pentium M Processor MSR IA32_THERM_CONTROL has been changed to IA32_CLOCK_MODULATION. This was done to avoid confusion about the MSR's function.

The following corrected table segment is from Appendix B, Table B-3, the *IA-32 Intel Architecture Software Developer's Manual, Volume 3*. It shows the change

Register Address		Register Name	Bit Description
Hex	Dec		
19AH	410	IA32_CLOCK_MODULATION	**Clock Modulation.** (R/W) Enables and disables on-demand clock modulation and allows the selection of the on-demand clock modulation duty cycle. (See Section 13.15.3., *Software Controlled Clock Modulation*. NOTE: IA32_CLOCK_MODULATION MSR was originally named IA32_THERM_CONTROL MSR.

2. "INTER-PRIVILEGE" was not Spelled Correctly in Pseudocode Entry

The term inter-privilege was incorrectly spelled in pseudocode provided as part of the "INT n/INTO/INT 3—Call to Interrupt Procedure" section, Chapter 3, *IA-32 Intel Architecture Software Developer's Manual, Volume 2*.

The corrected text segment is reproduced below:

...INTER-PRIVILEGE-LEVEL-INTERRUPT

```
    (* PE=1, interrupt or trap gate, non-conforming code segment, DPL<CPL *)
    (* Check segment selector and descriptor for stack of new privilege level in current TSS *)
    IF current TSS is 32-bit TSS
        THEN
            TSSstackAddress ← (new code segment DPL * 8) + 4.....
```

3. Confusing Text Artifact Removed

There were some materials in the OPCODE table that should have been deleted. This error has been corrected. The corrected table segment (reproduced below) is in Appendix A, Table A-3, *IA-32 Intel Architecture Software Developer's Manual, Volume 2*. See address 0x0f0b

	8	9	A	B	C	D	E	F
0	INVD	WBINVD		**UD2**				

4. IA32_MISC_CTL Has Been Removed from the List of Architectural MSRs

We removed MSR IA32_MISC_CTL from the list of architectural MSRs. Note that this MSR is still listed in other locations.

The impacted segment (reproduced below) is from Appendix B, Table B-5, *IA-32 Intel Architecture Software Developer's Manual, Volume 3*. The change bars show where the table row was deleted.

79H	121	IA32_BIOS_UPDT_TRIG	BIOS_UPDT_TRIG	P6 Family Processors
8BH	139	IA32_BIOS_SIGN_ID	BIOS_SIGN/BBL_CR_D3	P6 Family Processors
FEH	254	IA32_MTRRCAP	MTRRcap	P6 Family Processors
174H	372	IA32_SYSENTER_CS	SYSENTER_CS_MSR	P6 Family Processors
175H	373	IA32_SYSENTER_ESP	SYSENTER_ESP_MSR	P6 Family Processors

5. Typo Corrected in Figure 8-24

ExINT should be ExtINT in Figure 8-24, located in the "Message Data Register Format" section, Chapter 8, *IA-32 Intel Architecture Software Developer's Manual, Volume 3*. The corrected figure is reproduced below.

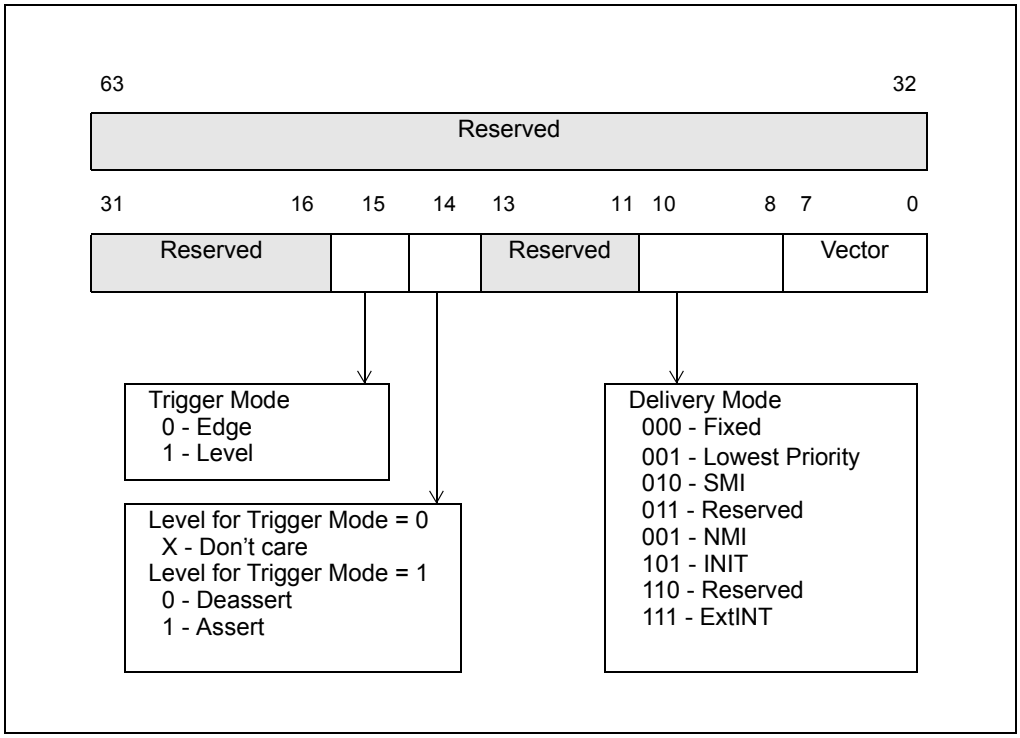
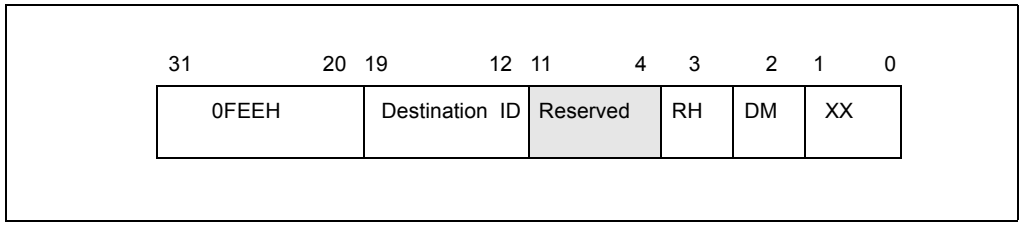


Figure 8-24. Layout of the MSI Message Data Register

6. Typo Corrected in Figure 8-23

0FEEH was incorrectly represented as 0FEEEH in Figure 8-23, located the "Message Address Register Format" section, Chapter 8, *IA-32 Intel Architecture Software Developer's Manual, Volume 3*.

The corrected figure is reproduced below.



7. Corrupted Text Corrected

There was some corrupted text in the "State of the Logical Processors" section, Chapter 7, *IA-32 Intel Architecture Software Developer's Manual, Volume 3*.

The correction is shown in the segment below. See the change bar.

7.6.1.1 State of the Logical Processors

The following features are considered part of the architectural state of a logical processor with HT 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:

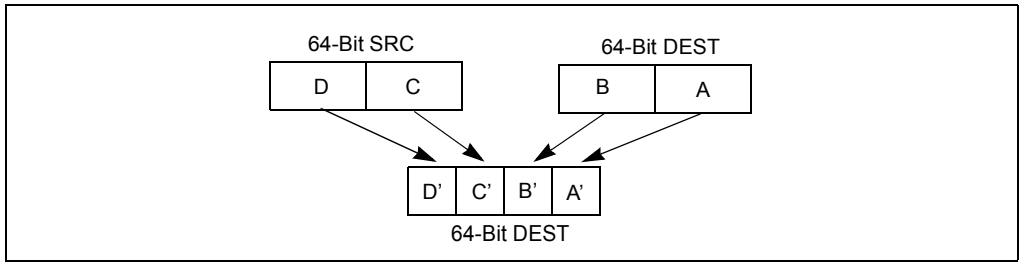
- General purpose registers (EAX, EBX, ECX, EDX, ESI, EDI, ESP, and EBP)

.....

8. Corrected an Error in PACKSSDW Illustration

Operation of the PACKSSDW instruction was incorrectly illustrated in Figure 3-6, the "PACKSSWB/PACKSSDW—Pack with Signed Saturation" section, Chapter 3, *IA-32 Intel Architecture Software Developer's Manual, Volume 2*.

The corrected figure is reproduced below.



9. SSM Corrected to SMM

In several places, SSM was still being used as an acronym for ‘system management mode.’ The correct usage is SMM. Corrections were made in the “Modes of Operation” section, Chapter 3, *IA-32 Intel Architecture Software Developer’s Manual, Volume 1*. The updated paragraph is reproduced below.

.....

System management mode (SMM). This mode provides a transparent mechanism for implementing platform-specific functions such as power management and system security. The processor enters SMM when the external SMM interrupt pin (SMI#) is activated or an SMI is received from the advanced programmable interrupt controller (APIC). In SMM, the processor switches to a separate address space while saving the basic context of the currently running program or task. SMM-specific code may then be executed transparently. Upon returning from SMM, the processor is placed back into its state prior to the system management interrupt. SMM was introduced with the Intel386™ SL and Intel486™ SL processors and became a standard IA-32 feature with the Pentium processor family.

.....

This change was also made in the “RSM—Resume from System Management Mode” section, Chapter 3, *IA-32 Intel Architecture Software Developer’s Manual, Volume 2*. The corrected segments are reproduced below.

.....

Returns program control from system management mode (SMM) to the application program or operating-system procedure that was interrupted when the processor received an SMM interrupt. The processor’s state is restored from the dump created upon entering SMM. If the processor detects invalid state information during state restoration, it enters the shutdown state....

...

```
ReturnFromSMM;
ProcessorState ← Restore(SMMDump);
```

...

10. Exiting from SMM Text Updated

A paragraph in the “Exiting from SMM” section, Chapter 13, *IA-32 Intel Architecture Software Developer’s Manual, Volume 3* has been updated. The information previously provided was not complete. The corrected text segment is reproduced below. See the change bar for location.

- (For the Pentium and Intel486 processors only.) If the address stored in the SMBASE register when an RSM instruction is executed is not aligned on a 32-Kbyte boundary. This restriction does not apply to the P6 family processors.

In the shutdown state, Intel processors stop executing instructions until a RESET#, INIT# or NMI# is asserted. Processors do recognize the FLUSH# signal in the shutdown state. While Pentium family processors recognize the SMI# signal in shutdown state, P6 family and Intel486 processors do not. Intel does not support using SMI# to recover from shutdown states for any processor family; the response of processors in this circumstance is not well defined. On Pentium 4 and later processors, shutdown will inhibit INTR and A20M but will not change any of the other inhibits. On these processors, NMIs will be inhibited if no action is taken in the SMM handler to uninhibit them (see Section 13.7.).

If the processor is in the HALT state when the SMI is received, the processor handles the return from SMM slightly differently (see Section 13.10., “Auto HALT Restart”). Also, the SMBASE address can be changed on a return from SMM (see Section 13.11., “SMBASE Relocation”).

11. L1 Data Cache Context Mode Description Has Been Updated

In Appendix B, Table B-1, *IA-32 Intel Architecture Software Developer's Manual, Volume 3*; the "L1 Data Cache Context Mode (RW)" table cell has been updated. Information about adaptive mode was clarified.

The updated table segment is reproduced below.

Register Address		Register Name Fields and Flags	Shared/ Unique[1]	Bit Description
Hex	Dec			
		24		**L1 Data Cache Context Mode (R/W).** When set to 1, this bit places the L1 Data Cache into shared mode. When set to 0 (the default), this bit places the L1 Data Cache into adaptive mode. When the L1 Data Cache is running in adaptive mode and the CR3s are identical, data in L1 is shared across logical processors. Otherwise, data in L1 is not shared and cache use is competitive. NOTE: If the Context ID feature flag, ECX[10], is not set to 1 after executing CPUID with EAX = 1; the ability to switch modes is not supported and the BIOS must not alter the contents of IA32_MISC_ENABLE[24].

12. #DE Should Be #DB in Description of EFLAGS.RF

In the "System Flags and Fields in the EFLAGS Register" section, Chapter 2, *IA-32 Intel Architecture Software Developer's Manual, Volume 3*; there was a sentence that began "When set, this flag temporarily disables debug exceptions (#DE)". Debug exceptions are noted as #DB, not #DE. This error has been corrected.

The corrected entry is reproduced below.

RF **Resume (bit 16).** Controls the processor's response to instruction-breakpoint conditions. When set, this flag temporarily disables debug exceptions (#DB) from being generated for instruction breakpoints; although, other exception conditions can cause an exception to be generated. When clear, instruction breakpoints will generate debug exceptions.

The primary function of the RF flag is to allow the restarting of an instruction following a debug exception that was caused by an instruction breakpoint condition. Here, debugger software must set this flag in the EFLAGS image on the stack just prior to returning to the interrupted program with the IRETD instruction, to prevent the instruction breakpoint from causing another debug exception. The processor then automatically clears this flag after the instruction returned to has been successfully executed, enabling instruction breakpoint faults again.

See Section 15.3.1.1., *Instruction-Breakpoint Exception Condition*, for more information on the use of this flag.

13. There Have Been Revisions to the Table That States Priority among Simultaneous Exceptions and Interrupts

We have made a number of updates to Table 5-2, located in the "Priority Among Simultaneous Exceptions and Interrupts" section, Chapter 5, *IA-32 Intel Architecture Software Developer's Manual, Volume 3*.

The updated cells are reproduced below.

Priority	Descriptions (continued)...
5	External Interrupts - NMI Interrupts - Maskable Hardware Interrupts
6	Code Breakpoint Fault
7	Faults from Fetching Next Instruction - Code-Segment Limit Violation - Code Page Fault
8	Faults from Decoding the Next Instruction - Instruction length > 15 bytes - Invalid Opcode - Coprocessor Not Available
9 (Lowest)	Faults on Executing an Instruction - Overflow - Bound error - Invalid TSS - Segment Not Present - Stack fault - General Protection - Data Page Fault - Alignment Check - x87 FPU Floating-point exception - SIMD floating-point exception

14. Corrections to Page-Directory-Pointer-Table Entry Description

In Figure 3-20 and 3-21, located in the "Page-Directory and Page-Table Entries With Extended Addressing Enabled" section, Chapter 3, *IA-32 Intel Architecture Software Developer's Manual, Volume 3*; Bit 0 of both representations of the Page-Directory-Pointer-Table Entry now indicate P (showing the location of the 'present flag' bit).

The corrected tables are reproduced below.

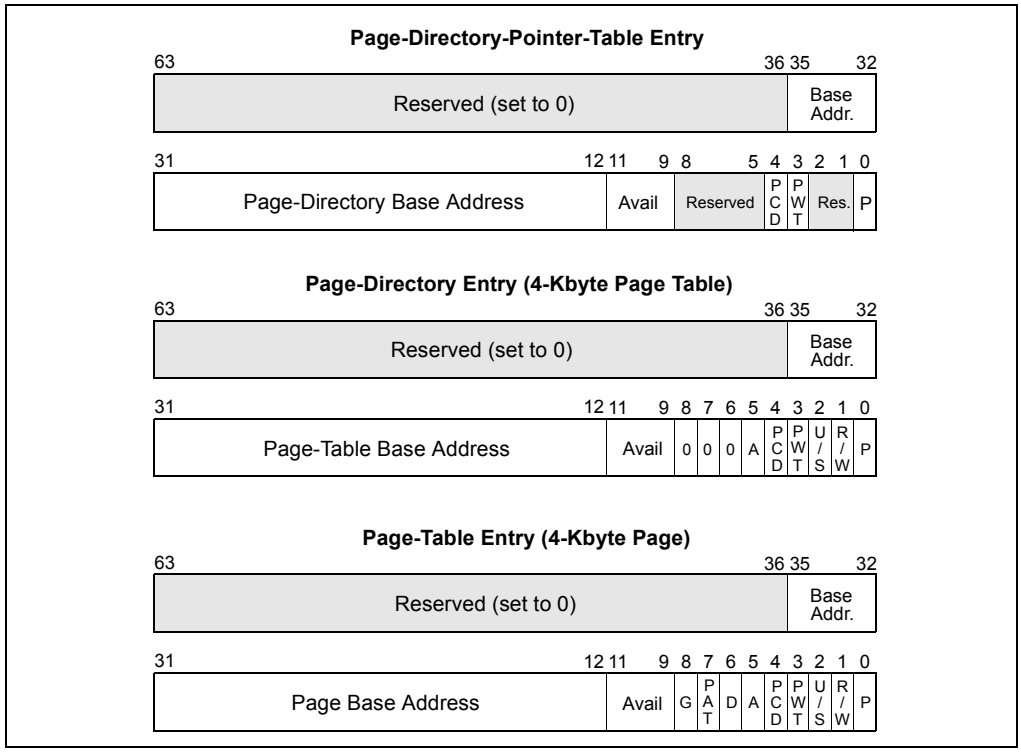


Figure 3-20. Format of Page-Directory-Pointer-Table, Page-Directory, and Page-Table Entries for 4-Kbyte Pages with PAE Enabled

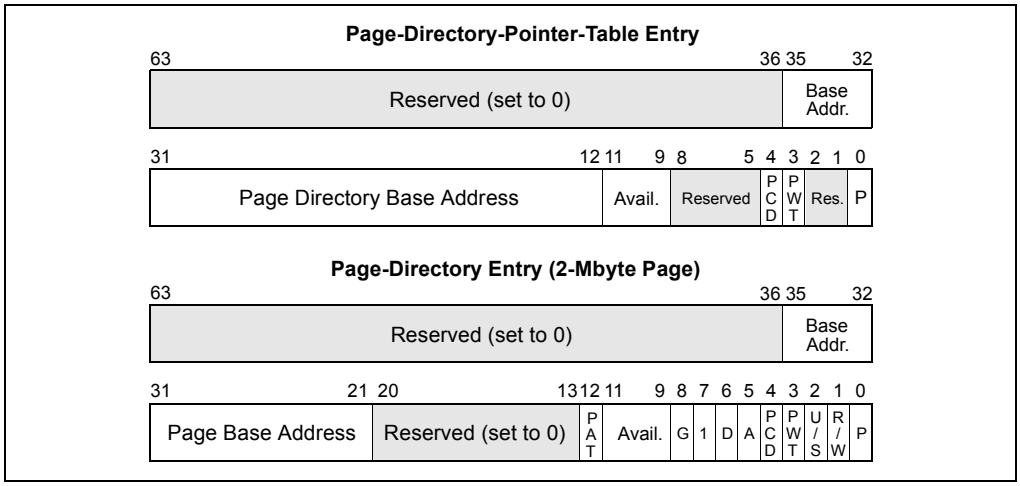


Figure 3-21. Format of Page-Directory-Pointer-Table and Page-Directory Entries for 2-Mbyte Pages with PAE Enabled

In addition, the paragraph discussing the present flag has been updated. this text is also located in the "Page-Directory and Page-Table Entries With Extended Addressing Enabled" section, Chapter 3, *IA-32 Intel Architecture Software Developer's Manual, Volume 3*.

The applicable text segment is reproduced below. Note the change bar.

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, which forces page tables and pages to be 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, which forces pages to be 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 Figure 3-20, for 4-Kb pages, and Figure 3-21 for 2-Mb 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.

15. Behavior Notes on the Accessed (A) Flag and Dirty (D) Flag

Notes have been added to two sub-paragraphs of the "Page-Directory and Page-Table Entries" section, Chapter 3, *IA-32 Intel Architecture Software Developer's Manual, Volume 3*. The notes clarify a limitation on the processor's Self-Modifying Code detection logic in the Accessed (A) flag and Dirty (D) flag context.

The applicable sections are reproduced below. See the change bars.

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.