TLBs, Paging-Structure Caches, and Their Invalidation

Application Note

April 2007

Document Number: 317080-001

Legal Statements

Contents

Revision History

Date	Revision	Description
April 2007	001	Initial release.

1 Introduction

The Intel® 64 and IA-32 architectures may accelerate the address-translation process by caching on the processor data from the structures in memory that control that process. Because the processor does not ensure that the data that it caches are always consistent with the structures in memory, it is important for software developers to understand how and when the processor may cache such data. They should also understand what actions software can take to remove cached data that may be inconsistent and when it should do so. The purpose of this application note is to provide software developers information about the relevant processor operation. This application note does not comprehend task switches and VMX transitions.

Section 2 gives an overview of the address-translation process, focusing on one mode of address translation called IA-32 mode. Section 3 and Section 4 describe how the processor may cache information in translation lookaside buffers (TLBs) and paging-structure caches. Section 5 explains how software can remove inconsistent cached information by invalidating parts of the TLBs and paging-structure caches. Section 6 and Section 7 describe extensions to the address-translation process that support large pages and global pages. Section 8 considers other modes of address translation. Section 9 describes special considerations for multiprocessor systems. Section 10 discusses an alternative behavior for INVLPG that may be implemented in the future.

2 Address Translation (Paging)

Processors supporting the Intel® 64 and IA-32 architectures translate **linear addresses**, generated by software, into **physical addresses**, which are used to access memory. The translation process is called **paging**. A logical processor uses paging if and only if PG (bit 31) is 1 in control register CR0.

Processors apply paging to all memory accesses that use linear addresses; this includes instruction fetches and ordinary data accesses. Paging also applies to prefetches and to memory accesses that are a result of speculative execution that would never actually occur in the executed code path; such accesses will not, however, cause page faults.

Details of the paging process depend on the **paging mode** of the processor as determined by the contents of various control registers. The different paging modes defined by the Intel® 64 and IA-32 architectures are identified in Section 2.1. Paging in one of these modes, IA-32e mode, is detailed in Section 2.2. Section 3 through Section 7 give details of the paging process by describing how those details apply to IA-32e mode. Section 8 describes the differences in behavior in other paging modes.

2.1 Paging Modes

The Intel® 64 and IA-32 architectures define three principal paging modes. The paging mode of a logical processor is determined by the setting of certain bits in certain control registers as follows:

- If PAE (bit 5) is 0 in control register CR4, paging translates from 32-bit linear addresses to 32-bit physical addresses. (This mode can translate to 36-bit physical addresses, but only for large pages.) This mode is described in more detail in Section 8.2.
- If CR4.PAE = 1 and LMA (bit 10) is 0 in the IA32_EFER MSR, paging translates from 32-bit linear addresses to 36-bit physical addresses. This mode is described in more detail in Section 8.1.
- If IA32_EFER.LMA = 1, paging translates from 48-bit linear addresses to 52-bit physical addresses.[1] This mode is called **IA-32e mode**. IA-32e mode is available only on processors that support the Intel® 64 architecture.

(The processor ensures that it is never the case that CR4.PAE = 0 and IA32_EFER.LMA = 1.)

The operation of these paging modes may be further refined, depending on whether large pages or global pages are enabled. These refinements are discussed in Section 6 and Section 7, respectively.

[1] Processor implementations may limit the size of the physical addresses to less than 52 bits. The physical-address width supported is enumerated using CPUID.

2.2 Paging in IA-32e Mode

In IA-32e mode, bits 47:12 of a 64-bit linear address is the 4-KByte **page number** of the linear address. (IA-32e mode requires that each of bits 63:48 of every linear address equal the value of bit 47.) (For simplicity, this section assumes that all pages are 4 KBytes in size. Section 6 discusses support for larger pages.) The processor uses the page number to determine the physical address of the 4-KByte **page frame** that the linear address accesses; bits 11:0 of the linear address determine the **offset** of the access within the page frame.

Paging is controlled by a hierarchical set of **paging structures** that reside in memory.[2] All paging structures are 4-KBytes in size (see Section 8.1 for one exception). Control register CR3 contains the 52-bit physical address of the root of the paging-structure hierarchy. The mechanism by which the paging structures translate a linear address to a physical address in IA-32e mode is illustrated in the following diagram and succeeding items.

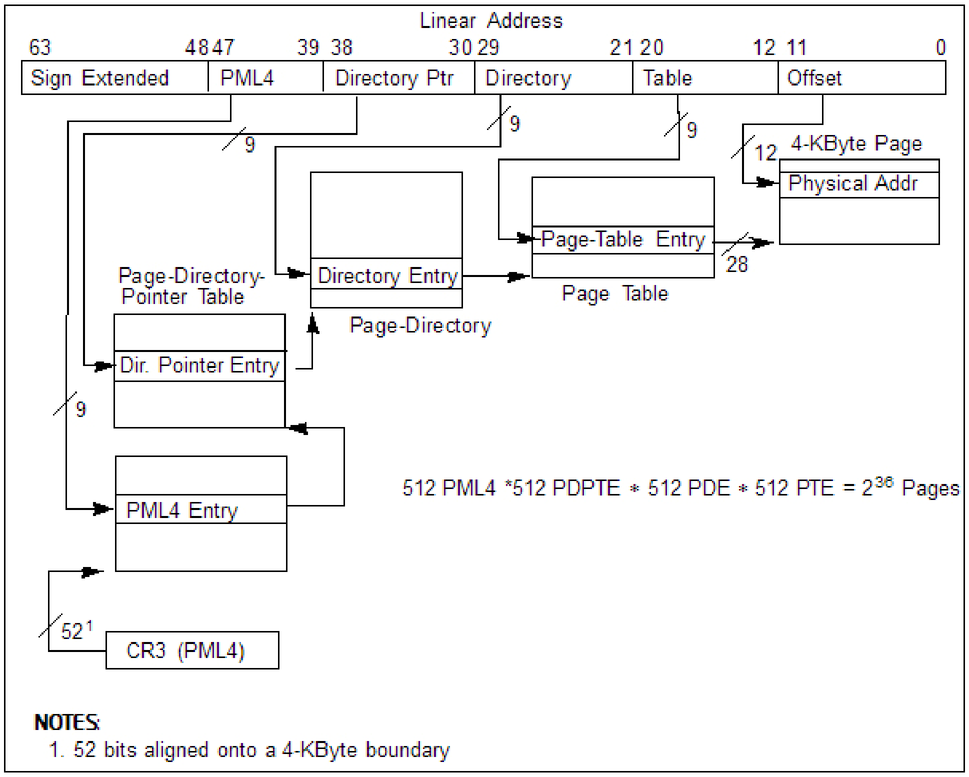

Figure 1. Address Translation for Paging Structures

[2] These structures may exist in memory or in the data-cache hierarchy. For simplicity, the remainder of this document refers only to memory even though the structures may be in a data cache.

- All paging structures comprise 512 8-byte (64-bit) entries.
- CR3 contains the physical address of the **PML4 table**. An entry in the PML4 table is selected by bits 47:39 of the linear address. This PML4 entry is used for all linear addresses in the 512-GByte region of linear addresses with the same value in bits 47:39.
- The selected PML4 entry contains the physical address of a page-directory-pointer table. A page-directory-pointer table entry (PDP) is selected by bits 38:30 of the linear address. This PDP is used for all linear addresses in the 1-GByte region of linear addresses with the same value in bits 47:30.
- The selected PDP contains the physical address of a page directory. A page-directory entry (PDE) is selected by bits 29:21 of the linear address. This PDE is used for all linear addresses in the 2-MByte region of linear addresses with the same value in bits 47:21.
- The selected PDE contains the physical address of a page table. A page-table entry (PTE) is selected by bits 20:12 of the linear address. This PTE is used for all linear addresses in the 4-KByte region of linear addresses with the same value in bits 47:12.
- The selected PTE contains the physical address of the page frame referenced by the linear address. The offset of the access within the page frame is the value of bits 11:0 of the linear address.

In addition to a physical address, each entry in a paging structure contains bits that control the translation process and how linear addresses may be used. These bits include the following:

- **Present** (bit 0). If this bit is 0 in any paging-structure entry used to translate a linear address, address translation stops and a page fault is generated. The physical address in the paging-structure entry is not used to access another paging structure.
- **Read/write** (bit 1). A linear address may be used by user software (CPL = 3) to write to memory only if this bit is 1 in all paging-structure entries (PML4, PDP, etc.) used to translate the linear address to a physical address. If the bit is 0 in any of those entries, an attempt by user software to write to memory using the linear address causes a page fault.[3]
- **User/supervisor** (bit 2). A linear address may be used by user software to access memory only if this bit is 1 in all paging-structure entries used to translate the linear address to a physical address. If the bit is 0 in any of those entries, an attempt by user software to access memory using the linear address causes a page fault.
- **Accessed** (bit 5). When a linear address is used to access memory, the processor sets this bit to 1 in all paging-structure entries used to translate the linear address to a physical address.
- **Dirty** (bit 6). When a linear address is used to write to memory, the processor sets this bit to 1 in the PTE used to translate the linear address to a physical address. This bit is not used in the entries of paging structures other than page tables; the processor ignores it. (See Section 6 for the case of large pages.)
- **Execute-disable** (bit 63). A linear address may be used to fetch instructions from memory only if this bit is 0 in all paging-structure entries used to translate the linear address to a physical address. If the bit is 1 in any of those entries, an attempt to fetch instructions from memory using the linear address causes a page fault.[4]

(Prefetches and memory accesses that are a result of speculative execution do not cause

[3] If WP (bit 16) is 1 in CR0, such writes cause page faults regardless of CPL.

[4] If NXE (bit 11) in the IA32_EFER MSR is 0, the execute-disable bit is considered reserved and must be 0 in all paging-structure entries.

page faults. If such an access encounters one of the faulting conditions identified above, it is aborted.)

See Chapter “Protected-Mode Memory Management” of the *Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 3A: System Programming Guide*, for more details of the paging process.

3 Translation Lookaside Buffers (TLBs)

Section 2.2 described how paging translates linear page numbers to physical page frames. The processor may accelerate the paging process by caching individual translations from page numbers to page frames in **translation lookaside buffers** (**TLBs**).[5]

Each entry in a TLB is an individual **translation**. Each translation is referenced by the page number of a linear address. It contains the following information from the paging-structure entries used to translate linear addresses with the page number:

- Page frame. This physical address in the PTE used to translate the page number.
- Read/write. The logical-AND of the read/write bits in all paging-structure entries used to translate the page number.
- User/supervisor. The logical-AND of the user/supervisor bits in all paging-structure entries used to translate the page number.
- Dirty. An indication of whether the dirty bit is 1 in the PTE used to translate the page number.
- Execute-disable. The logical-OR of the execute-disable bits in all paging-structure entries used to translate the page number.

(TLB entries may contain other information as well. A processor may implement multiple TLBs, and some of these may be for special purposes, e.g., only for instruction fetches. Such special-purpose TLBs may not contain all this information if it is not necessary. For example, a TLB used only for instruction fetches need not contain information about the read/write and dirty bits.)

If the page number of a linear address corresponds to a TLB entry, the processor may use that TLB entry to determine the corresponding page frame and the bits used to control accesses with that linear address. In this case, the processor may not actually consult the paging structures in memory.

As suggested in Section 2.2, the processor does not cache a translation for a page number unless the present bits are 1 in all paging-structure entries used to translate that page number. In addition, the processor does not cache a translation for a page number until it has set the accessed bits in all the paging-structure entries used during translation.

The processor may cache a TLB entry at the time it translates a page number to a page frame, and the information cached in the TLB entry is determined at that time. If software modifies the relevant paging-structure entries after this translation, the TLB entry may not reflect the contents of the paging-structure entries.

Processors need not implement any TLBs. Processors that do implement TLBs may invalidate any TLB entry at any time. Software should not rely on the existence of TLBs or on the retention of TLB entries.

[5] This document uses the term **TLB** to refer to arrays that may cache multiple translations. An individual translation in an array is called a **TLB entry**.

4 Paging-Structure Caches

In addition to the TLBs, a processor may cache other information about the paging structures in memory.

4.1 Caches for Paging Structures

A processor may support any or of all the following paging-structure caches:

- **PML4 cache**. Each PML4-cache entry is referenced by a 9-bit value and is used for linear addresses for which bits 47:39 have that value. The entry contains information from the PML4 entry used to translate such linear addresses:
  - Page-directory-pointer table address. The physical address from the PML4 entry.
  - Read/write. The value of the read/write bit of the PML4 entry.
  - User/supervisor. The value of the user/supervisor bit of the PML4 entry.
  - Execute-disable. The value of the execute-disable bit of the PML4 entry.

  The following items detail how a processor may use the PML4 cache:

  - If a PML4-cache entry is used for a linear address, the processor may use that entry to locate the translation's page-directory-pointer table and the bits used to control accesses with that linear address. The processor might not use the PML4 entry in memory.
  - The processor does not create a PML4-cache entry unless the present bit is 1 in the PML4 entry in memory.
  - The processor does not create a PML4-cache entry until it has set the accessed bit in the PML4 entry in memory.
  - The processor may create a PML4-cache entry even if there are no translations for any linear address that might use that entry (e.g., because all entries in the referenced page-directory-pointer table are marked "not present").
  - The processor may create a PML4-cache entry at the time of translation, and the information in the entry is determined at that time. The entry may not reflect any subsequent modifications to the PML4 entry in memory.
- **PDP cache**. Each PDP-cache entry is referenced by an 18-bit value and is used for linear addresses for which bits 47:30 have that value. The entry contains information from the PML4 and PDP entries used to translate such linear addresses:
  - Page-directory address. The physical address from the PDP.
  - Read/write. The logical-AND of the read/write bits in the PML4 and PDP entries.
  - User/supervisor. The logical-AND of the user/supervisor bits in the PML4 and PDP entries.
  - Execute-disable. The logical-OR of the execute-disable bits in the PML4 and PDP entries.

  The following items detail how a processor may use the PDP cache:

  - If a PDP-cache entry is used for a linear address, the processor may use that entry to locate the translation's page directory and the bits used to control accesses with that linear address. The processor might not use the PML4 or PDP entries in memory.
  - The processor does not create a PDP-cache entry unless the present bit is 1 in the PDP in memory.

    - The processor does not create a PDP-cache entry until it has set the accessed bit in the PDP in memory.
    - The processor may create a PDP-cache entry even if there are no translations for any linear address that might use that entry (e.g., because all entries in the referenced page directory are marked "not present").
    - The processor may create a PDP-cache entry at the time of translation, and the information in the entry is determined at that time. The entry may not reflect any subsequent modifications to the PML4 and PDP entries in memory.
- **PDE cache**. Each PDE-cache entry is referenced by a 27-bit value and is used for linear addresses for which bits 47:21 have that value. The entry contains information from the PML4 entry, PDP, and PDE used to translate such linear addresses:
    - Page-table address. The physical address from the PDE.
    - Read/write. The logical-AND of the read/write bits in the PML4 entry, PDP, and PDE.
    - User/supervisor. The logical-AND of the user/supervisor bits in the PML4 entry, PDP, and PDE.
    - Execute-disable. The logical-OR of the execute-disable bits in the PML4 entry, PDP, and PDE.

  The following items detail how a processor may use the PDE cache:

    - If a PDE-cache entry is used for a linear address, the processor may use that entry to locate the translation's page table and the bits used to control accesses with that linear address. The processor might not use the PML4 entry, PDP, and PDE in memory.
    - The processor does not create a PDE-cache entry unless the present bit is 1 in the PDE in memory.
    - The processor does not create a PDE-cache entry until it has set the accessed bit in the PDE in memory.
    - The processor may create a PDE-cache entry even if there are no translations for any linear address that might use that entry (e.g., because all entries in the referenced page table are marked "not present").
    - The processor may create a PDE-cache entry at the time of translation, and the information in the entry is determined at that time. The entry may not reflect any subsequent modifications to the PML4 entry, PDP, and PDE in memory.

Note that information from a paging-structure entry may be included in entries in the paging-structure caches for other paging-structure entries referenced by the original entry. For example, if the read/write bit is 0 in a PML4 entry, then the read/write bit will be 0 in any PDP-cache entry for a PDP from the PDP table referenced by that PML4 entry. This is because the read/write bit of each such PDP-cache entry is the logical-AND of the read/write bits in the appropriate PML4 and PDP entries, and the former is 0 by assumption.

A processor may or may not implement any of the paging-structure caches. Software should rely on neither their presence nor their absence. The processor may invalidate entries in these caches at any time. Because the processor may create the cache entries at the time of translation and not update them following subsequent modifications to the paging structures in memory, software should take care to invalidate the cache entries appropriately when causing such modifications. The invalidation of TLBs and the paging-structure caches is described in Section 5.

4.2 Using the Paging-Structure Caches to Translate Linear Addresses

When a linear address is accessed, the processor may use the following procedure to determine the physical address to which it translates and whether the access should be allowed:

- If the processor finds a TLB entry for the page number (bits 47:12) of the linear address, it may use the page frame in that entry to determine the physical address and can use the bits (read/write, user/supervisor, and execute-disable) to determine whether the access should be allowed.[6]
- If the processor does not find a TLB entry, it may use bits 47:21 of the linear address to select an entry from the PDE cache. It can then use the physical address in that entry to complete the translation process (locating a PTE), identifying the page frame for the linear address and determining whether the access should be allowed based on the combination of the bits stored in the PDE-cache entry with those resulting from the translation process.
- If the processor does not find a TLB entry or a PDE-cache entry, it may use bits 47:30 of the linear address to select an entry from the PDP cache. It can then use the physical address in that entry to complete the translation process (locating a PDE and a PTE), identifying the page frame for the linear address and determining whether the access should be allowed based on the combination of the bits stored in the PDP-cache entry with those resulting from the translation process.
- If the processor does not find a TLB entry, a PDE-cache entry, or a PDP-cache entry, it may use bits 47:39 of the linear address to select an entry from the PML4 cache. It can then use the physical address in that entry to complete the translation process (locating a PDP, a PDE, and a PTE), identifying the page frame for the linear address and determining whether the access should be allowed based on the combination of the bits stored in the PML4-cache entry with those resulting from the translation process.

Any of the above steps would be skipped if the processor does not support the cache in question. If the processor does not find a TLB entry, a PDE-cache entry, a PDP-cache entry, or a PML4 entry for the linear address, it uses bits 47:12 of the linear address to traverse the entire paging-structure hierarchy, starting with the PML4 table referenced by the physical address in CR3, as described in Section 2.2.

[6] If the processor supports special-purpose TLBs, it may only use the TLBs relevant to the current memory access. For example, a processor would not use an instruction TLB for an ordinary data access.

The first row of Table 1 recapitulates which bits of a linear address are used to access entries in each of the paging-structure caches. The second and third rows apply to the other paging modes discussed in Section 8.

Table 1. Bits of Linear Address Used to Index the Paging Structure Caches

Paging Mode	PML4	PDP	PDE	PTE*
IA32_EFER.LMA = 1	47:39	47:30	47:21	47:12
IA32_EFER.LMA = 0 and CR0.PAE = 1 (see Section 8.1)	N/A	31:30†	31:21	31:12
IA32_EFER.LMA = 0 and CR0.PAE = 0 (see Section 8.2)	N/A	N/A	31:22	31:12

* No PTE is used for large-page translations (if PS is 1 in the PDE); see Section 6.

† For this paging mode, these entries are called PDPTRs and are all loaded together. They are invalidated only as described in Section 8.1.

4.3 Multiple Cached Entries for a Single Paging-Structure Entry

Note that multiple cached entries (in the paging-structure caches or TLBs) may contain information derived from a single paging-structure entry. The following items give some examples:

- Suppose that two PML4 entries contain the same physical address and thus reference the same page-directory-pointer table. Then any PDP in that table may result in two PDP-cache entries, each associated with a different set of linear addresses. Specifically, suppose that the n_1^{th} and n_2^{th} entries in the PML4 table contain the same physical address. This implies that the physical address in the m^{th} PDP in the page-directory-pointer table will appear in the PDP-cache entries associated with both p_1 and p_2, where $(p_1 \gg 9) = n_1$, $(p_2 \gg 9) = n_2$, and $(p_1 \,\&\, 1FFH) = (p_2 \,\&\, 1FFH) = m$. This is because both PDP-cache entries use the same PDP, one resulting from a reference from the n_1^{th} PML4 entry and one from the n_2^{th} PML4 entry.
- Suppose that the first PML4 entry (i.e., the one in position 0) contains the physical address X in CR3 (the physical address of the PML4 table). This implies the following:
  - The PML4-cache entry associated with linear addresses with 0 in bits 47:39 contains address X.
  - The PDP-cache entry associated with linear addresses with 0 in bits 47:30 contains address X. This is because the translation for a linear address for which the value of bits 47:30 is 0 uses the value of bits 47:39 (0) to locate a page-directory-pointer table at address X (the address of the PML4 table). It then uses the value of bits 38:30 (also 0) to find address X again and to store that address in the PDP-cache entry.
  - The PDE-cache entry associated with linear addresses with 0 in bits 47:21 contains address X for similar reasons.
  - The TLB entry for page number 0 (i.e., associated with linear addresses with 0 in bits 47:12) translates to page frame X » 12 for similar reasons.

  The same PML4 entry contributes its address X to all these cache entries because the self-referencing nature of the entry causes it to be used as a PML4 entry, a PDP, a PDE, and a PTE.

5 Invalidation of TLBs and Paging-Structure Caches

As noted in Section 3 and Section 4, the processor may create entries in the TLBs and the paging-structure caches when linear addresses are being translated and may retain these entries even after the paging structures used to create them have been modified. To ensure that address translation uses the modified paging structures, software should take action to invalidate any cached entries that may contain information that has since been modified.

5.1 Invalidation Instructions

It is recommended that software use the following instructions to invalidate entries in the TLBs and the paging-structure caches:

- INVLPG. This instruction takes a single operand, which is a linear address. The instruction invalidates any TLB entries for the page number of that linear address including those for global pages (see Section 7). INVLPG also invalidates all entries in all paging-structure caches regardless of the linear addresses to which they correspond.
- MOV to CR3. This instruction invalidates all TLB entries except those for global pages. It also invalidates all entries in all paging-structure caches.
- MOV to CR4. If this instruction modifies PGE (bit 7) of CR4, it invalidates all TLB entries (including those for global pages) and all entries in all paging-structure caches.

The processor is always free to invalidate additional entries in the TLBs and paging-structure caches. The following are some examples:

- INVLPG may invalidate TLB entries for pages other than the one corresponding to its linear-address operand.
- MOV to CR3 may invalidate TLB entries for global pages.
- On a processor supporting Hyper-Threading Technology, invalidations performed on one logical processor may invalidate entries in the TLBs and paging-structure caches used by other logical processors.

(Other instructions and operations may invalidate entries in the TLBs and the paging-structure caches, but the instructions identified above are recommended.)

In addition to the instructions identified above, page faults invalidate entries in the TLBs and paging-structure caches. In particular, a page fault resulting from an attempt to use a linear address will invalidate any PML4-cache, PDP-cache, and PDE-cache entries that would be used for that linear address as well as any TLB entry for that address's page number.[7] These invalidations ensure that the page fault will not recur (if the faulting instruction is re-executed) if it would not be caused by the contents of the paging structures in memory (and if, therefore, it resulted from cached entries that were not invalidated after the paging structures were modified in memory).

[7] Unlike INVLPG, page faults do not invalidate all entries in the paging-structure caches, only those that would be used to translate the faulting linear address.

5.2 Recommended Invalidation

The following items provide some recommendations regarding when software should perform invalidations:

- If a PTE is modified, software should execute INVLPG for a linear address with a page number whose translation uses that PTE. (If the PTE may be used in the translation of different page numbers, software should execute INVLPG for linear addresses with each of those page numbers; alternatively, it could use MOV to CR3 or MOV to CR4.)
- If a PML4 entry, a PDP, or a PDE is modified, software may use one of the following approaches depending upon the types and number of translations controlled by the modified entry:
  - Execute INVLPG for linear addresses with each of the page numbers with valid translations that would use the entry. However, if all page numbers that would use the entry result in invalid translations (e.g., because all entries in the paging structure referenced by the modified entry are marked "not present"), it remains necessary to execute INVLPG at least once.
  - Execute MOV to CR3 if the modified entry controls no global pages.
  - Execute MOV to CR4 to modify CR4.PGE.
- If the nature of the paging structures is such that a single entry may be used for multiple purposes (see Section 4.3), software should perform invalidations for all of these purposes. For example, if a single entry might serve as both a PDE and PTE, it may be necessary to execute INVLPG with two (or more) linear addresses, one that uses the entry as a PDE and one that uses it as a PTE. (Alternatively, software could use MOV to CR3 or MOV to CR4.)

5.3 Optional Invalidation

The following items describe cases in which software may choose not to invalidate and the potential consequences of that choice:

- If a paging-structure entry is modified to transition the present bit from 0 to 1, no invalidation is necessary (assuming that an invalidation was performed the last time the present bit was transitioned from 1 to 0). This is because no TLB entry or paging-structure cache entry will be created with information from a paging-structure entry that is marked "not present."
- If a paging-structure entry is modified to transition the accessed bit from 0 to 1, no invalidation is necessary (assuming that an invalidation was performed the last time the accessed bit was transitioned from 1 to 0). This is because no TLB entry or paging-structure cache entry will be created with information from a paging-structure entry that is marked "not accessed."
- If a paging-structure entry is modified to transition the read/write bit from 0 to 1, failure to perform an invalidation may result in a "spurious" page fault (e.g., in response to an attempt to write to memory) but no other adverse behavior. Such a page fault will occur at most once for each affected linear address (see Section 5.1).
- If a paging-structure entry is modified to transition the user/supervisor bit from 0 to 1, failure to perform an invalidation may result in a "spurious" page fault (e.g., in response to an attempt to access to memory while CPL = 3) but no other adverse behavior. Such a page fault will occur at most once for each affected linear address (see Section 5.1).

- If a paging-structure entry is modified to transition the execute-disable bit from 1 to 0, failure to perform an invalidation may result in a “spurious” page fault (e.g., in response to an attempt to fetch instructions from memory) but no other adverse behavior. Such a page fault will occur at most once for each affected linear address (see Section 5.1).
- If a paging-structure entry is modified to transition the accessed bit from 1 to 0, failure to perform an invalidation may result in the processor not setting that bit in response to a subsequent access to a linear address whose translation may use the entry. Software cannot interpret the bit being clear as an indication that such an access has not occurred.
- If a PTE is modified to transition the dirty bit from 1 to 0, failure to perform an invalidation may result in the processor not setting that bit in response to a subsequent write to a linear address whose translation may use the entry. Software cannot interpret the bit being clear as an indication that such a write has not occurred.

Section 4.3 describes situations in which a single paging-structure entry may contain information cached in multiple entries in the paging-structure caches. Because all entries in these caches are invalidated by any execution of INVLPG, it is not necessary to follow the modification of such a paging-structure entry by executing INVLPG multiple times solely for the purpose of invalidating these multiple cached entries. (It may be necessary to do so to invalidate multiple TLB entries.)

5.4 Delayed Invalidation

Required invalidations may be delayed under some circumstances. For example, when freeing a portion of the linear-address space (by marking paging-structure entries “not present), invalidation using INVLPG may be delayed if software does not re-allocate that portion of the linear-address space or the memory that had been associated with it. However, because of speculative execution (or errant software), there may be accesses to the freed portion of the linear-address space before the invalidations occur. In this case, the following can happen:

- Reads can occur to the freed portion of the linear-address space. Therefore, invalidation should not be delayed for an address range that has read side effects.
- The processor may retain TLB entries and paging-structure entries for an extended period of time. Software should not assume that the processor will not use entries in the TLBs and paging-structure caches associated with a linear address simply because time has passed.
- As noted in Section 4.1, the processor may create an entry in a paging-structure cache even if there are no translations for any linear address that might use that entry. Thus, if software has marked “not present” all entries in page table, the processor may subsequently create a PDE-cache entry for the PDE that references that page table (assuming the PDE itself is marked “present”).
- If software attempts to write to the freed portion of the linear-address space, the processor might not generate a page fault. (Such an attempt would likely be the result of a software error.) For that reason, the page frames previously associated with the freed portion of the linear-address space should not be reallocated for another purpose until the appropriate invalidations have been performed.

6 Large Pages

The paging mechanism described in Section 2.2 is for 4-KByte pages: the page number is in bits 47:12 of a linear address, and bits 11:0 contain the offset within the page. The Intel® 64 and IA-32 architectures also allow for **large pages** when either PSE (bit 4) or PAE (bit 5) of CR4 is 1.

When PS (bit 7) is 1 in a PDE, the entry is not used to reference a page table (with the physical address in bits 51:12 of the entry) but instead to identify a 2-MByte **large-page frame** (with physical address in bits 51:21 of the entry); bits 20:0 of the linear address determine the offset within the large-page frame.[8] In addition, bit 6 of the PDE is used as a dirty bit, just as it is for PTEs.

When a PDE is used to identify a large page, its treatment is modified with respect to the TLBs and the paging-structure caches:

- No PDE-cache entry is created from the PDE.
- The processor may create TLB entries from the PDE.
    - Each large-page TLB entry contains a translation from a **large-page number** (bits 47:21) of a linear address using the entry to the large-page frame identified by the PDE. Such translations are called **large-page translations** to distinguish them from the **ordinary translations** used for 4-KByte pages.
    - Each large-page TLB entry includes read/write, user/supervisor, and execute-disable information as do TLB entries for ordinary translations. Large-page TLB entries do not include information from a PTE, since no PTE is used to create large-page entries.
    - Each large-page TLB entry includes the value of the dirty bit from the PDE.
    - The processor may cache large-page TLB entries in a structure or structures separate from those used for ordinary translations. Alternatively, it may use the same physical structure both for PDE-cache entries and for TLB entries for large pages.
    - INVLPG invalidates any large-page TLB entries associated with the large-page number from the linear address in the instruction's operand.
    - MOV to CR3 and MOV to CR4 invalidate large-page TLB entries as they do other TLB entries.

The following items provide some recommendations regarding invalidation for large pages:

- If a PDE for a large page is modified, software should execute INVLPG for a linear address with large-page number whose translation uses that PDE. (If the PDE may be used in the translation of different large page numbers, software should execute INVLPG for linear addresses with each of those large page numbers; alternatively, it could use MOV to CR3 or MOV to CR4.)
- If a PDE for a large page is modified to transition the dirty bit from 1 to 0, failure to perform an invalidation may result in the processor not setting that bit in response to a subsequent write to a linear address whose translation may use the entry. Software cannot interpret the bit being clear as an indication that such a write has not occurred.

Some processors may not always use large-page translations for large pages. A processor may instead create one or more ordinary translations for a large page. Like other ordinary translations, such a translation would be associated with bits 47:12 of the linear address

[8] At this time, PS is defined only for PDEs; it is ignored in PML4 entries and PDPs.

(even though only bits 47:21 are the large-page number; bits 20:12 are part of the offset). In such a translation, the high-order 27 bits of the page frame are derived from the physical address in the PDE used to create the translation, while the low-order 9 bits are bits 20:12 of the linear address of the access for which the translation is created. The processor distinguishes ordinary translations derived from large pages from those derived from 4-KByte pages.

There is no way for software to be aware that ordinary translations have been used for a large page. In addition, there is no need for software to perform different actions in this case. In particular, the INVLPG instruction provides the same assurances that it provides when the linear address in its operand is associated with a large-page translation. It will invalidate all ordinary translations derived from the large-page number in bits 47:21 of the linear address in the instruction’s operand.

If future processors support even larger pages by allowing PS to be set in PDPs or in PML4 entries, the treatment of the affected paging-structure entries will be modified in a manner similar to that described above for PDEs.

7 Global Pages

The Intel® 64 and IA-32 architectures also allow for **global pages** when PGE (bit 7) is 1 in CR4. When G (bit 6) is 1 in a PTE, any TLB entry cached for a linear address using that PTE is considered **global**. The same is true for G in a PDE that identifies a large page frame (i.e., in which PS is 1).[9] (See Section "Page-Directory and Page-Table Entries" of the *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3A: System Programming Guide*, for more information about global pages.)

The following items detail the invalidation of global TLB entries:

- An execution of INVLPG invalidates any TLB entries for the page number of the linear address that is its operand, even if the entries are global.
- An execution of the MOV to CR3 instruction may not invalidate global TLB entries.
- An execution of the MOV to CR4 instruction that transitions CR4.PGE from 1 to 0 invalidates all TLB entries, including global TLB entries. This is because, after the instruction completes, there are no global TLB entries (because CR4.PGE = 0).
- An execution of the MOV to CR4 instruction that transitions CR4.PGE from 0 to 1 invalidates all TLB entries. Because CR4.PGE was 0 before the instruction executed, there were no global TLB entries.

The G bit is used only in PTEs and in PDEs that identify a large page frame. As noted earlier, information from these entries is cached only in the TLBs and not in the paging-structure caches. Because of this, the global-page architecture does not affect the behavior of the paging-structure caches.

[9] G is not used for PML4 entries or PDPs. It is also not used for PDEs that do not identify a large page frame (i.e., in which PS is 0). The processor ignores the value of the bit in all of these entries.

8 Other Paging Modes

Section 2.2 provided the details of paging for a processor that is in IA-32e mode (for which IA32_EFER.LMA = 1). The treatment of other paging modes is similar. Some details are given in the following sections.

8.1 32-Bit Paging with PAE

If IA32_EFER.LMA = 0 and CR4.PAE = 1, paging translates 32-bit linear addresses to 36-bit physical addresses. The following items describe how paging in this mode operates differently from that described in Section 2.2 for IA-32e mode:

- CR3 contains the 32-bit physical address of the page-directory-pointer table, which contains 4 8-byte (64-bit) entries, called **PDPTRs**.
- each of the following bits must be 0 in each PDPTR:
    - Bit 63. In IA-32e mode, this is a PDP's execute-disable bit. If IA32_EFER.LMA = 0 and CR4.PAE = 1, the processor uses only the PDE and the PTE to determine whether software can fetch instructions from memory using a linear address.
    - Bits 8:6. In IA-32e mode, the processor ignores these bits in a PDP.
    - Bit 5. In IA-32e mode, this is a PDP's accessed bit. The processor does not set accessed bits in the PDPTRs.
    - Bit 2. In IA-32e mode, this is a PDP's user/supervisor bit. If IA32_EFER.LMA = 0 and CR4.PAE = 1, the processor uses only the PDE and the PTE to determine whether software running at CPL = 3 can access a memory using a linear address.
    - Bit 1. In IA-32e mode, this is a PDP's read/write bit. If IA32_EFER.LMA = 0 and CR4.PAE = 1, the processor uses only the PDE and the PTE to determine whether software running at CPL = 3 can write to memory using a linear address.
- Bits 31:30 of the linear address select a PDPTR.
- The PDPTR contains the 36-bit physical address of a page directory.
- Bits 29:21 of the linear address select a PDE.
- If PS is 1 in the PDE, the entry contains the 36-bit physical address of a 2-MByte page frame (bits 31:21 of the linear address are the large-page number).
- If PS is 0 in the PDE, the entry contains the 36-bit physical address of a page table.
- Bits 20:12 of the linear address select a PTE.
- The PTE contain the 36-bit physical address of the page frame.
- Bits 31:12 of the linear address are the page number.

The following items describe how operation of the paging-structure caches differs in this mode from that described in Section 4 and Section 5:

- The processor does not maintain a PDP cache as described in Section 4. The processor always caches information from the four page-directory-pointer-table entries. These entries are not cached at the time of address translation. Instead, they are always cached as part of the execution of the following instructions:
    - A MOV to CR3 that occurs with IA32_EFER.LMA = 0 and CR4.PAE = 1.
    - A MOV to CR4 that results in CR4.PAE = 1, that occurs with IA32_EFER.LMA = 0 and CR0.PG = 1, and that modifies at least one of CR4.PAE, CR4.PGE, and CR4.PSE.
    - A MOV to CR0 that modifies CR0.PG and that occurs with IA32_EFER.LMA = 0 and CR4.PAE = 1.

These instructions fault if they would load a PDPTR that sets any of the bits that must be 0 (see above). These cached entries are not modified by any other operations.[10] In particular, executions of INVLPG do not affect these cached entries.

- The processor may use a PDE cache. Each PDE-cache entry is referenced by an 11-bit value and is used for linear addresses for which bits 31:21 have that value. Entries in the PDE cache are invalidated as described in Section 5.1.

8.2 32-Bit Paging without PAE

If IA32_EFER.LMA = CR4.PAE = 0, paging translates 32-bit linear addresses. Ordinary translations are to 32-bit physical addresses. Large pages are 4 Mbytes in size and can be used only if CR4.PSE = 1. Large-page translations are to 36-bit physical addresses unless CPUID.01H.EDX.PSE-36[bit 17] = 0, in which case they are to 32-bit physical addresses.

The following items detail the differences between operation in this mode and that in IA-32e mode:

- Each page directory and page table comprises 1024 32-bit entries.
- No entries have execute-disable bits; the processor behaves as if all execute-disable bits were 0.
- CR3 contains the 32-bit physical address of the page directory.
- Bits 31:22 of the linear address select a PDE.
- If PS is 0 in the PDE, the following apply:
  - The PDE contains the high 20 bits of the 32-bit physical address of a page table.
  - Bits 21:12 of the linear address select a PTE.
  - The 4-KByte page frame has a 32-bit address identified by bits 31:12 of the PTE.
  - Bits 11:0 of the linear address are the offset of the access within the page frame; thus, bits 31:12 of the linear address are the page number.
- If PS is 1 in the PDE, the following apply:
  - If CPUID.01H.EDX.PSE-36[bit 17] = 0, the 4-MByte page frame has the 32-bit address identified by bits 31:22 of the PDE.
  - If CPUID.01H.EDX.PSE-36[bit 17] = 1, the 4-MByte large-page frame has the 36-bit address identified by 14 bits in the PDE:
    - The high-order 4 bits are bits 16:13 of the PDE.
    - The low-order 10 bits are bits 31:22 of the PDE.
  - Bits 21:0 of the linear address are the offset of the access within the large-page frame; thus, bits 31:22 of the linear address are the large-page number.

In this paging mode, the processor may use a PDE cache. Each PDE-cache entry is referenced by a 10-bit value and is used for linear addresses for which bits 31:22 have that value. Entries in the PDE cache are invalidated as described in Section 5.

[10] As noted in Section 1, this application note does not comprehend task switches and VMX transitions.

9 Propagation of Paging-Structure Changes to Multiple Processors

As noted in Section 5, software that modifies a paging-structure entry may need to invalidate entries in the TLBs and paging-structure caches that were derived from the modified entry before it was modified. In a system containing more than one logical processor, software must account for the fact that there may be entries in the TLBs and paging-structure caches of logical processors other than the one used to modify the paging-structure entry. The process of propagating the changes to a paging-structure entry is commonly referred to as "TLB shootdown."

TLB shootdown can be done using memory-based semaphores and/or interprocessor interrupts (IPI). The following items describe a simple but inefficient example of a TLB shootdown algorithm for processors supporting the Intel® 64 and IA-32 architectures:

1. Begin barrier: Stop all but one logical processor; that is, cause all but one to execute the HLT instruction or to enter a spin loop.
2. Allow the active logical processor to change the necessary paging-structure entries.
3. Allow all logical processors to perform invalidations appropriate to the modifications to the paging-structure entries.
4. Allow all logical processors to resume normal operation.

Alternative, performance-optimized, TLB shootdown algorithms may be developed; however, software developers must take care to ensure that the following conditions are met:

- All logical processors that are using the paging structures that are being modified must participate and perform appropriate invalidations after the modifications are made.
- If the modifications to the paging-structure entries are made before the barrier or there is no barrier, the operating system must ensure one of the following: (1) that the affected linear-address range is not used between the time of modification and the time of invalidation; or (2) that it is prepared to deal with the consequences of the affected linear-address range being used during that period. For example, if the operating system does not allow pages being freed to be reallocated for another purpose until after the required invalidations, writes to those pages by errant software will not unexpectedly modify memory that is in use.
- Software must be prepared to deal with reads, code fetches, and prefetch requests to the affected linear-address range that are a result of speculative execution that would never actually occur in the executed code path.

When multiple logical processors are using the same linear-address space at the same time, they must coordinate before any request to modify the paging-structure entries that control that linear-address space. In these cases, the barrier in the TLB shootdown routine may not be required. For example, when freeing a range of linear addresses, some other mechanism can assure no logical processor is using that range before the request to free it is made. In this case, a logical processor freeing the range can mark as "not present" the PTEs associated with the range, free the physical page frames associated with the range, and then signal the other logical processors using that linear-address space to perform the necessary invalidations.   All the affected logical processors must complete their invalidations before the linear-address range and the physical page frames previously associated with that range can be reallocated.

10 Alternative INVLPG Behavior

This section describes an alternative behavior of the INVLPG instruction. Any processor supporting this behavior would report it through CPUID in a manner to be determined. The processor would not exhibit the new behavior (and would thus continue to exhibit the behavior described in Section 5.1) unless the alternative were explicitly enabled by software in a manner to be determined.

The behavior of INVLPG described in Section 5.1 treats the TLBs and the paging-structure caches differently:

- The instruction is required to invalidate only those TLB entries corresponding to the page number of the linear address in the instruction's operand.
- The instruction is required to invalidate all entries in all paging-structure caches regardless of whether they corresponding to the linear address in the instruction's operand.

Because performance may be enhanced when entries in the paging-structure caches are retained longer, it may be advantageous to invalidate only those entries corresponding to the linear address in the operand to INVLPG. Future processors may do so as follows (in place of the behavior described above):

- In IA-32e mode:
  - Invalidate any PML4-cache entry associated with the value of bits 47:39 in the linear address in the instruction's operand.
  - Invalidate any PDP-cache entry associated with the value of bits 47:30 in the linear address in the instruction's operand.
  - Invalidate any PDE-cache entry associated with the value of bits 47:21 in the linear address in the instruction's operand.
- Outside IA-32e mode with CR4.PAE = 1: Invalidate any PDE-cache entry associated with the value of bits 31:21 in the instruction's operand.
- Outside IA-32e mode with CR4.PAE = 0: Invalidate any PDE-cache entry associated with the value of bits 31:22 in the instruction's operand.

(In all cases, the processor may invalidate more entries than required.)

While this more selective invalidation may improve performance, software enabling it should ensure that it does not rely on the broader invalidation provided by existing processors. Specifically, it may be necessary to execute INVLPG multiple times following the modification of a single paging-structure entry if information from that entry may be cached in multiple entries in the paging-structure caches (see Section 4.3).