an eight-byte boundary to yield the best processor performance. The limit value for the GDT is expressed in bytes. As with segments, the limit value is added to the base address to get the address of the last valid byte. A limit value of 0 results in exactly one valid byte. Because segment descriptors are always 8 bytes long, the GDT limit should always be one less than an integral multiple of eight (that is, 8N – 1).

The first descriptor in the GDT is not used by the processor. A segment selector to this "null descriptor" does not generate an exception when loaded into a data-segment register (DS, ES, FS, or GS), but it always generates a general-protection exception (#GP) when an attempt is made to access memory using the descriptor. By initializing the segment registers with this segment selector, accidental reference to unused segment registers can be guaranteed to generate an exception.

The LDT is located in a system segment of the LDT type. The GDT must contain a segment descriptor for the LDT segment. If the system supports multiple LDTs, each must have a separate segment selector and segment descriptor in the GDT. The segment descriptor for an LDT can be located anywhere in the GDT. See Section 3.5., "System Descriptor Types", information on the LDT segment-descriptor type.

An LDT is accessed with its segment selector. To eliminate address translations when accessing the LDT, the segment selector, base linear address, limit, and access rights of the LDT are stored in the LDTR register (see Section 2.4., "Memory-Management Registers").

When the GDTR register is stored (using the SGDT instruction), a 48-bit "pseudo-descriptor" is stored in memory (see Figure 3-11). To avoid alignment check faults in user mode (privilege level 3), the pseudo-descriptor should be located at an odd word address (that is, address MOD 4 is equal to 2). This causes the processor to store an aligned word, followed by an aligned doubleword. User-mode programs normally do not store pseudo-descriptors, but the possibility of generating an alignment check fault can be avoided by aligning pseudo-descriptors in this way. The same alignment should be used when storing the IDTR register using the SIDT instruction. When storing the LDTR or task register (using the SLTR or STR instruction, respectively), the pseudo-descriptor should be located at a doubleword address (that is, address MOD 4 is equal to 0).

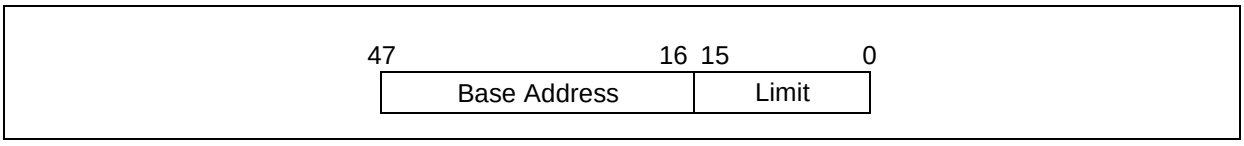

Figure 3-11. Pseudo-Descriptor Format

3.6. PAGING (VIRTUAL MEMORY) OVERVIEW

When operating in protected mode, the IA-32 architecture permits the linear address space to be mapped directly into a large physical memory (for example, 4 GBytes of RAM) or indirectly (using paging) into a smaller physical memory and disk storage. This latter method of mapping the linear address space is commonly referred to as virtual memory or demand-paged virtual memory.

When paging is used, the processor divides the linear address space into fixed-size pages (of 4 KBytes, 2 MBytes, or 4 MBytes in length) that can be mapped into physical memory and/or disk storage. When a program (or task) references a logical address in memory, the processor translates the address into a linear address and then uses its paging mechanism to translate the linear address into a corresponding physical address.

If the page containing the linear address is not currently in physical memory, the processor generates a page-fault exception (#PF). The exception handler for the page-fault exception typically directs the operating system or executive to load the page from disk storage into physical memory (perhaps writing a different page from physical memory out to disk in the process). When the page has been loaded in physical memory, a return from the exception handler causes the instruction that generated the exception to be restarted. The information that the processor uses to map linear addresses into the physical address space and to generate page-fault exceptions (when necessary) is contained in page directories and page tables stored in memory.

Paging is different from segmentation through its use of fixed-size pages. Unlike segments, which usually are the same size as the code or data structures they hold, pages have a fixed size. If segmentation is the only form of address translation used, a data structure present in physical memory will have all of its parts in memory. If paging is used, a data structure can be partly in memory and partly in disk storage.

To minimize the number of bus cycles required for address translation, the most recently accessed page-directory and page-table entries are cached in the processor in devices called translation lookaside buffers (TLBs). The TLBs satisfy most requests for reading the current page directory and page tables without requiring a bus cycle. Extra bus cycles occur only when the TLBs do not contain a page-table entry, which typically happens when a page has not been accessed for a long time. See Section 3.11., “Translation Lookaside Buffers (TLBs)”, for more information on the TLBs.

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 3.8.2., “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 36 bits. 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.

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.

3.6.2. Page Tables and Directories

The information that the processor uses to translate linear addresses into physical addresses (when paging is enabled) is contained in four data structures:

- Page directory—An array of 32-bit page-directory entries (PDEs) contained in a 4-KByte page. Up to 1024 page-directory entries can be held in a page directory.
- Page table—An array of 32-bit page-table entries (PTEs) contained in a 4-KByte page. Up to 1024 page-table entries can be held in a page table. (Page tables are not used for 2-MByte or 4-MByte pages. These page sizes are mapped directly from one or more page-directory entries.)
- Page—A 4-KByte, 2-MByte, or 4-MByte flat address space.
- Page-Directory-Pointer Table—An array of four 64-bit entries, each of which points to a page directory. This data structure is only used when the physical address extension is enabled (see Section 3.8., “36-Bit Physical Addressing Using the PAE Paging Mechanism”).

These tables provide access to either 4-KByte or 4-MByte pages when normal 32-bit physical addressing is being used and to either 4-KByte or 2-MByte pages or 4-MByte pages only when extended (36-bit) physical addressing is being used. Table 3-3 shows the page size and physical address size obtained from various settings of the paging control flags and the PSE-36 CPUID feature flag. Each page-directory entry contains a PS (page size) flag that specifies whether the entry points to a page table whose entries in turn point to 4-KByte pages (PS set to 0) or whether the page-directory entry points directly to a 4-MByte (PSE and PS set to 1) or 2-MByte page (PAE and PS set to 1).

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. 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	36 Bits
1	1	X	1	X	2 MBytes	36 Bits

3.7.1. Linear Address Translation (4-KByte Pages)

Figure 3-12 shows the page directory and page-table hierarchy when mapping linear addresses to 4-KByte pages. The entries in the page directory point to page tables, and the entries in a page table point to pages in physical memory. This paging method can be used to address up to 2^{20} pages, which spans a linear address space of 2^{32} bytes (4 GBytes).

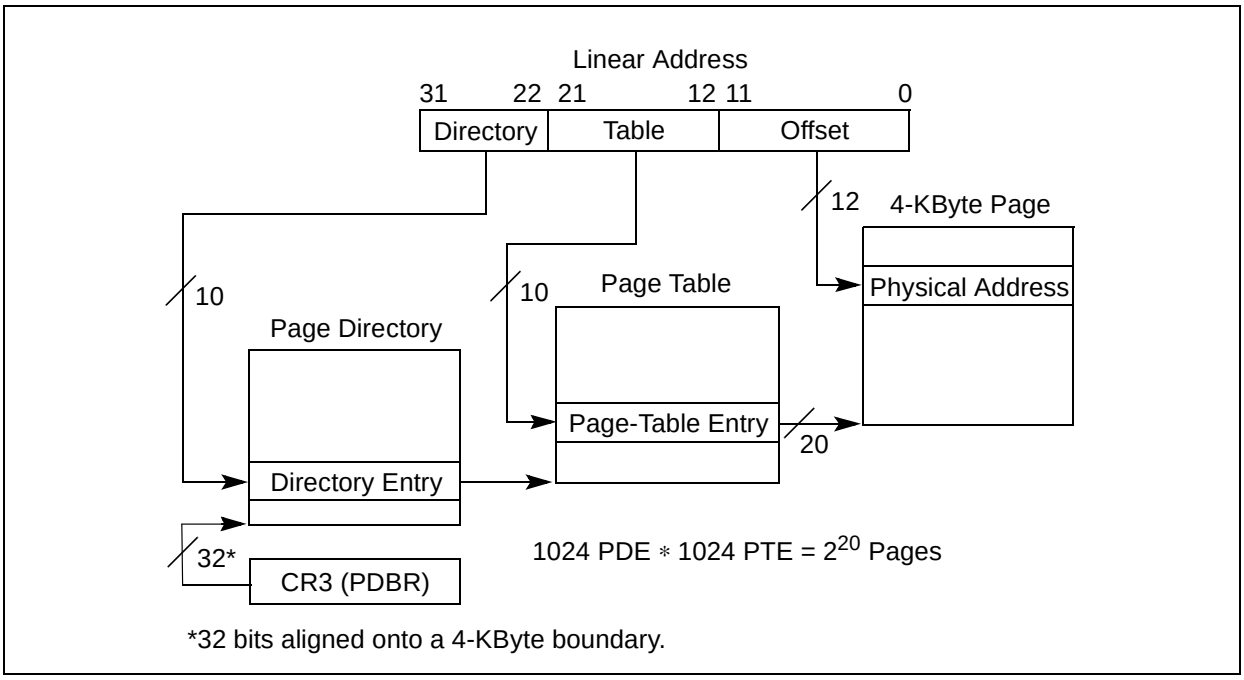

Figure 3-12. Linear Address Translation (4-KByte Pages)

To select the various table entries, the linear address is divided into three sections:

- Page-directory entry—Bits 22 through 31 provide an offset to an entry in the page directory. The selected entry provides the base physical address of a page table.
- Page-table entry—Bits 12 through 21 of the linear address provide an offset to an entry in the selected page table. This entry provides the base physical address of a page in physical memory.
- Page offset—Bits 0 through 11 provides an offset to a physical address in the page.

Memory management software has the option of using one page directory for all programs and tasks, one page directory for each task, or some combination of the two.

3.7.2. Linear Address Translation (4-MByte Pages)

Figure 3-12 shows how a page directory can be used to map linear addresses to 4-MByte pages. The entries in the page directory point to 4-MByte pages in physical memory. This paging method can be used to map up to 1024 pages into a 4-GByte linear address space.

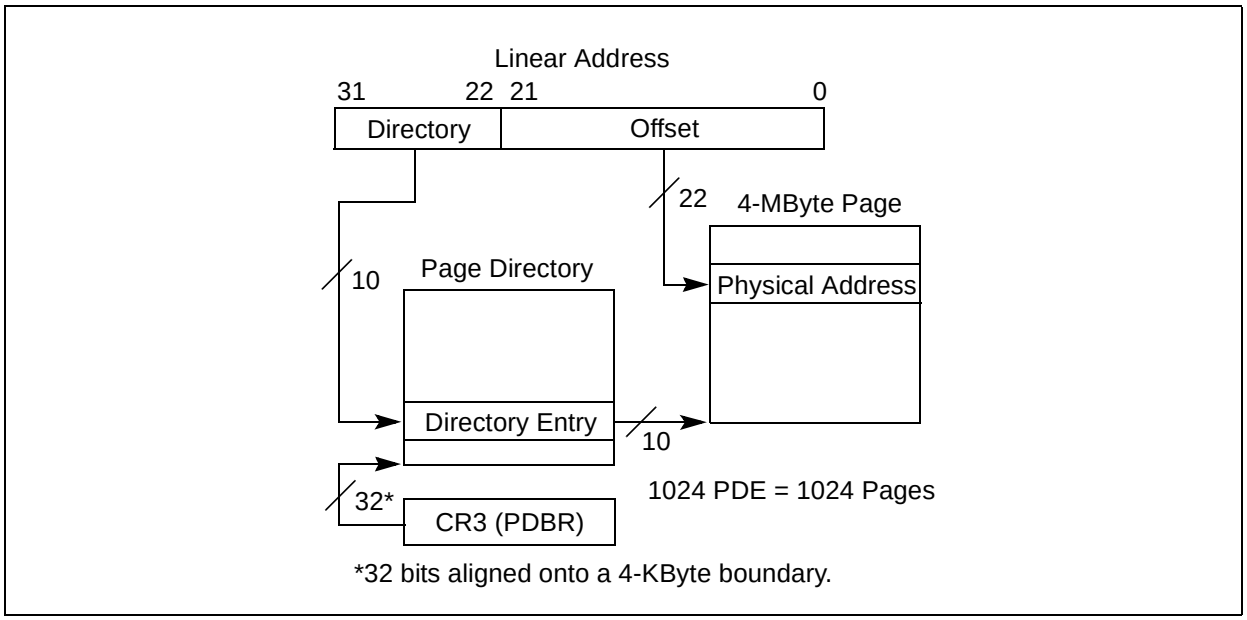

Figure 3-13. Linear Address Translation (4-MByte Pages)

The 4-MByte page size is selected by setting the PSE flag in control register CR4 and setting the page size (PS) flag in a page-directory entry (see Figure 3-14). With these flags set, the linear address is divided into two sections:

- Page directory entry—Bits 22 through 31 provide an offset to an entry in the page directory. The selected entry provides 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.

NOTE

(For the Pentium processor only.) When enabling or disabling large page sizes, the TLBs must be invalidated (flushed) after the PSE flag in control register CR4 has been set or cleared. Otherwise, incorrect page translation might occur due to the processor using outdated page translation information stored in the TLBs. See Section 10.9., "Invalidating the Translation Lookaside Buffers (TLBs)", for information on how to invalidate the TLBs.

3.7.3. Mixing 4-KByte and 4-MByte Pages

When the PSE flag in CR4 is set, both 4-MByte pages and page tables for 4-KByte pages can be accessed from the same page directory. If the PSE flag is clear, only page tables for 4-KByte pages can be accessed (regardless of the setting of the PS flag in a page-directory entry).

A typical example of mixing 4-KByte and 4-MByte pages is to place the operating system or executive's kernel in a large page to reduce TLB misses and thus improve overall system performance.

The processor maintains 4-MByte page entries and 4-KByte page entries in separate TLBs. So, placing often used code such as the kernel in a large page, frees up 4-KByte-page TLB entries for application programs and tasks.

3.7.4. Memory Aliasing

The IA-32 architecture permits memory aliasing by allowing two page-directory entries to point to a common page-table entry. Software that needs to implement memory aliasing in this manner must manage the consistency of the accessed and dirty bits in the page-directory and page-table entries. Allowing the accessed and dirty bits for the two page-directory entries to become inconsistent may lead to a processor deadlock.

3.7.5. Base Address of the Page Directory

The physical address of the current page directory is stored in the CR3 register (also called the page directory base register or PDBR). (See Figure 2-5 and Section 2.5., “Control Registers”, for more information on the PDBR.) If paging is to be used, the PDBR must be loaded as part of the processor initialization process (prior to enabling paging). The PDBR can then be changed either explicitly by loading a new value in CR3 with a MOV instruction or implicitly as part of a task switch. (See Section 6.2.1., “Task-State Segment (TSS)”, for a description of how the contents of the CR3 register is set for a task.)

There is no present flag in the PDBR for the page directory. The page directory may be not-present (paged out of physical memory) while its associated task is suspended, but the operating system must ensure that the page directory indicated by the PDBR image in a task's TSS is present in physical memory before the task is dispatched. The page directory must also remain in memory as long as the task is active.

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.

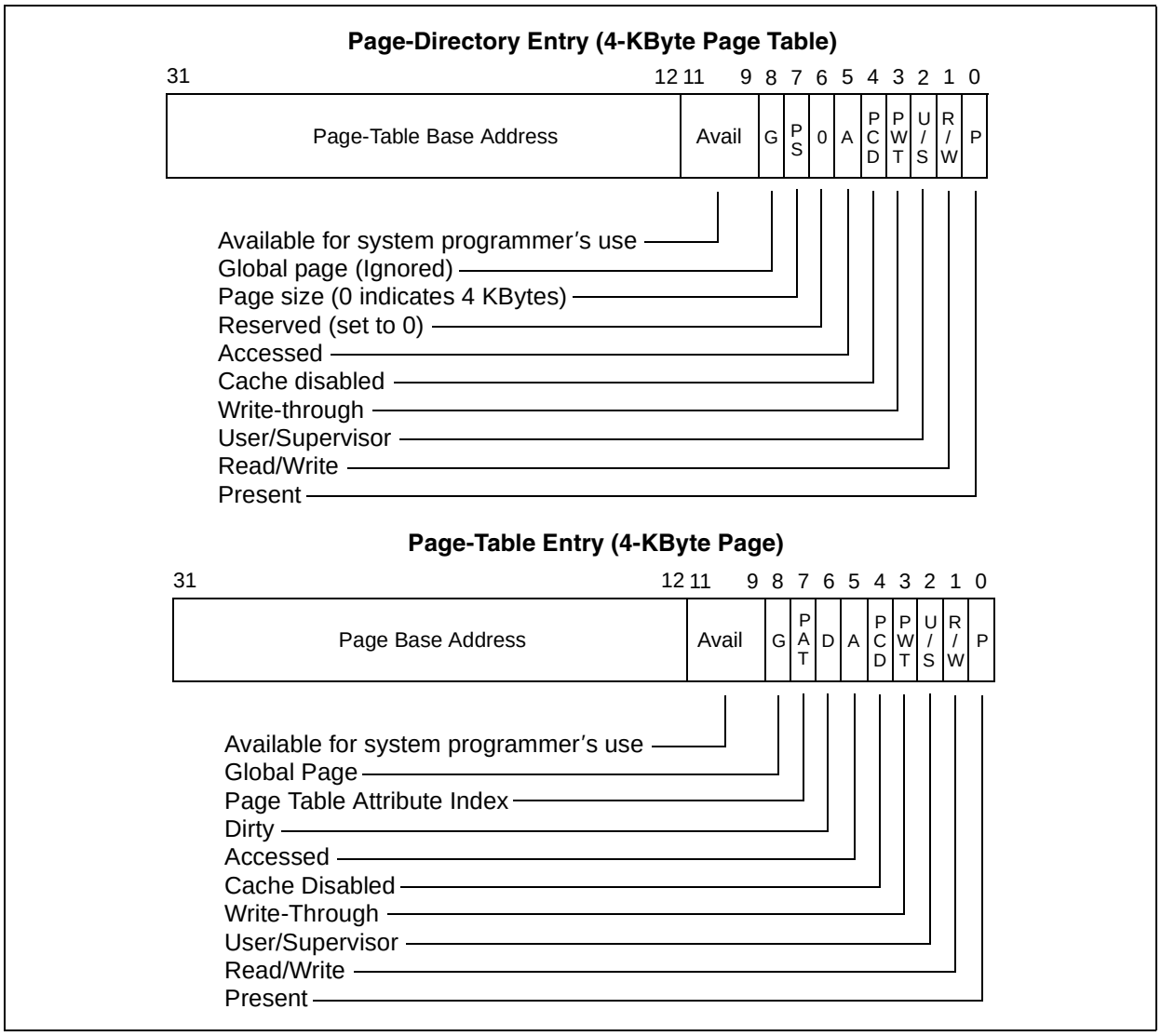

Figure 3-14. Format of Page-Directory and Page-Table Entries for 4-KByte Pages and 32-Bit Physical Addresses

(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.

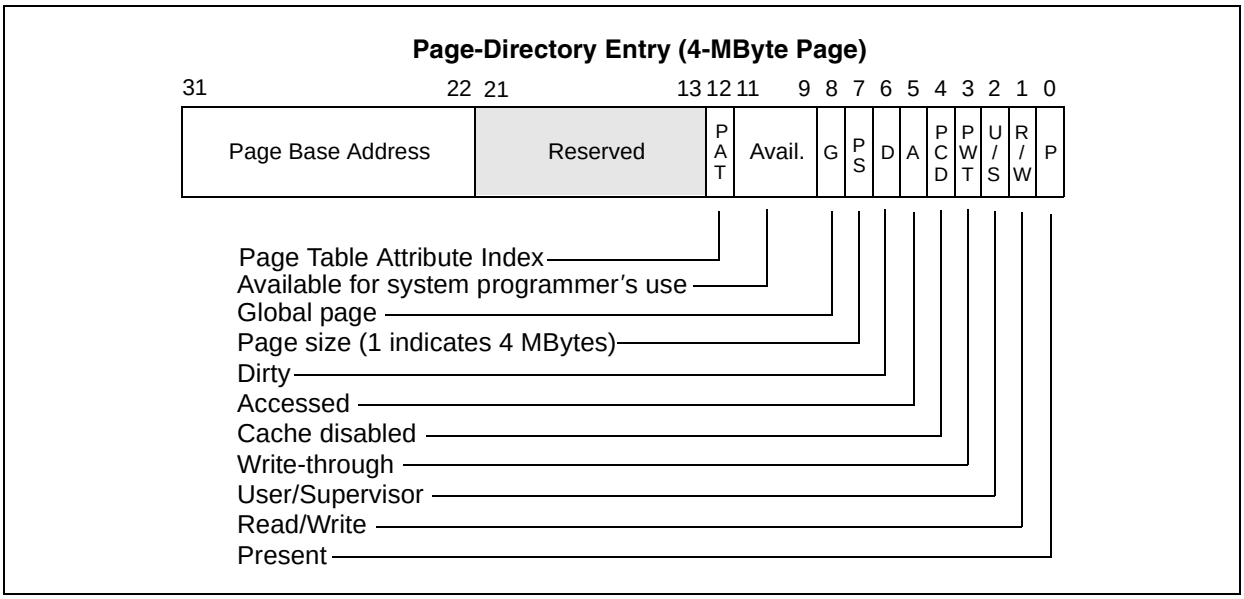

Figure 3-15. Format of Page-Directory Entries for 4-MByte Pages and 32-Bit Addresses

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 3.11., "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 4-2 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 4-2 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.

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.

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 3.11., "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.

3.7.7. Not Present Page-Directory and Page-Table Entries

When the present flag is clear for a page-table or page-directory entry, the operating system or executive may use the rest of the entry for storage of information such as the location of the page in the disk storage system (see Figure 3-16).

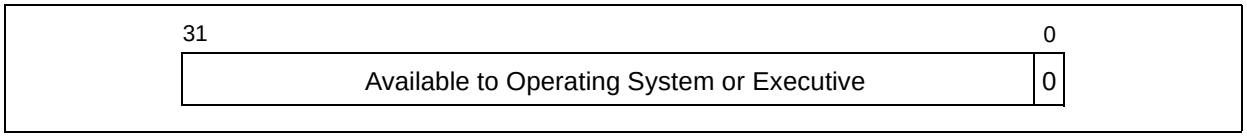

Figure 3-16. Format of a Page-Table or Page-Directory Entry for a Not-Present Page

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. Here, the processor provides 4 additional address line pins 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 the 36-bit 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 (PDPR) is replaced with a 27-bit page-directory-pointer-table base address field (see Figure 3-17). (In this case, register CR3 is called the PDPTR.) This field provides the 27 most-significant bits of the physical address of the first byte of the page-directory-pointer table, which forces the table to be located on a 32-byte boundary.
- Linear address translation is changed to allow mapping 32-bit linear addresses into the larger physical address space.