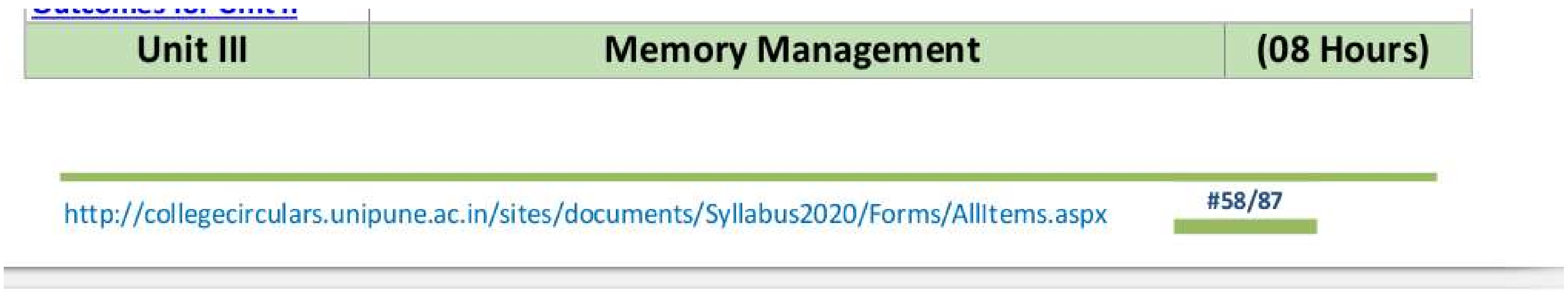
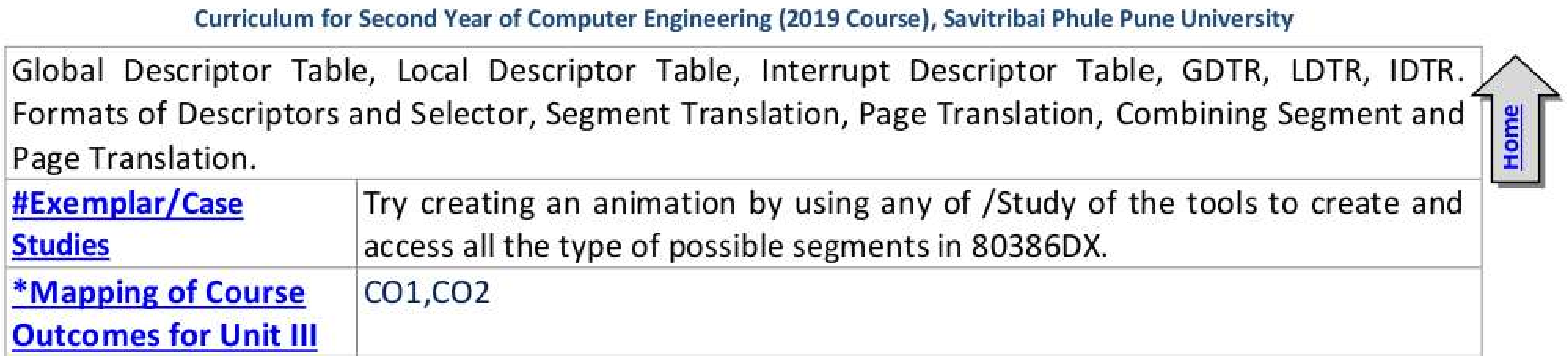
Unit III Memory Management



Unit III	Memory Management	(08 Hours)

http://collegecirculars.unipune.ac.in/sites/documents/Syllabus2020/Forms/AllItems.aspx #58/87



Curriculum for Second Year of Computer Engineering (2019 Course), Savitribai Phule Pune University

Global Descriptor Table, Local Descriptor Table, Interrupt Descriptor Table, GDTR, LDTR, IDTR. Formats of Descriptors and Selector, Segment Translation, Page Translation, Combining Segment and Page Translation.	
#Exemplar/Case Studies	Try creating an animation by using any of /Study of the tools to create and access all the type of possible segments in 80386DX.
*Mapping of Course Outcomes for Unit III	CO1,CO2

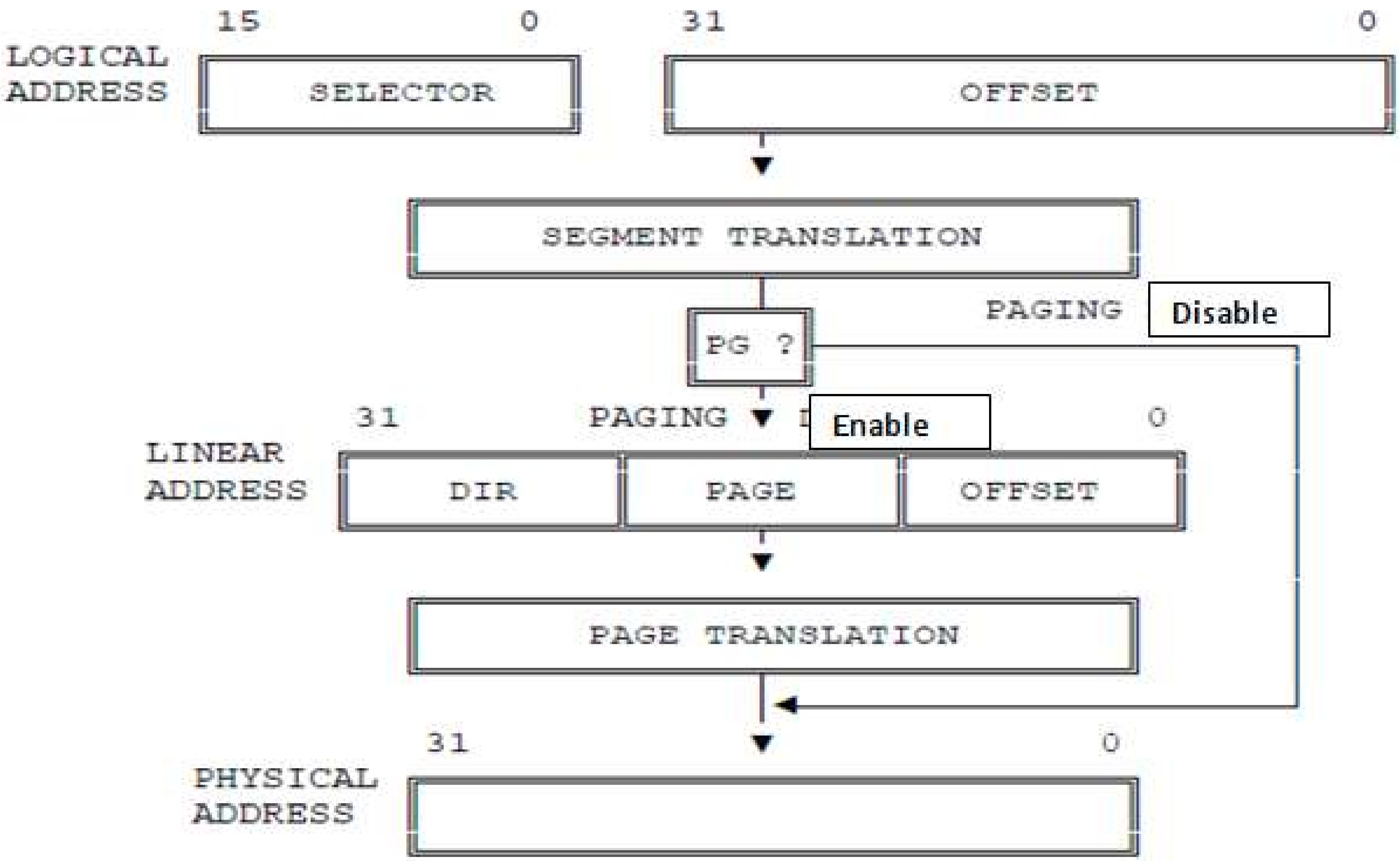
Home

Memory Management

- The 80386 transforms logical addresses (i.e., addresses as viewed by programmers) into physical address (i.e., actual addresses in physical memory) in two steps:

1. Segment translation, in which a logical address (consisting of a segment selector and segment offset) are converted to a linear address.
2. Page translation, in which a linear address is converted to a physical address. This step is optional, at the discretion of systems-software designers.

- These translations are performed in a way that is not visible to applications programmers.

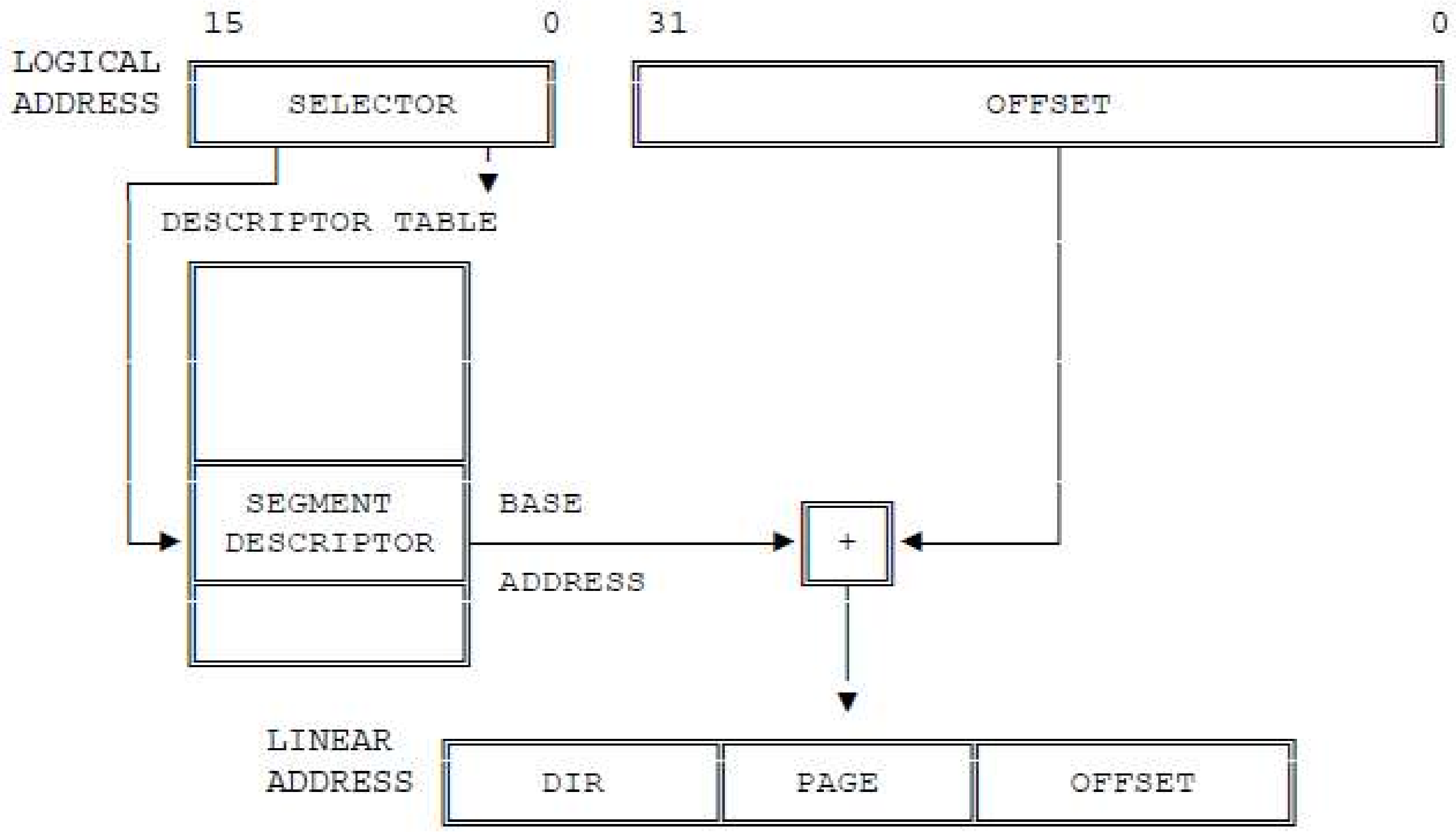


Address Translation Overview

Segment Translation

- Processor converts a logical address into a linear address
- To perform segment translation, the processor uses the following data structures:

1. Descriptors
2. Descriptor tables
3. Selectors
4. Segment Registers

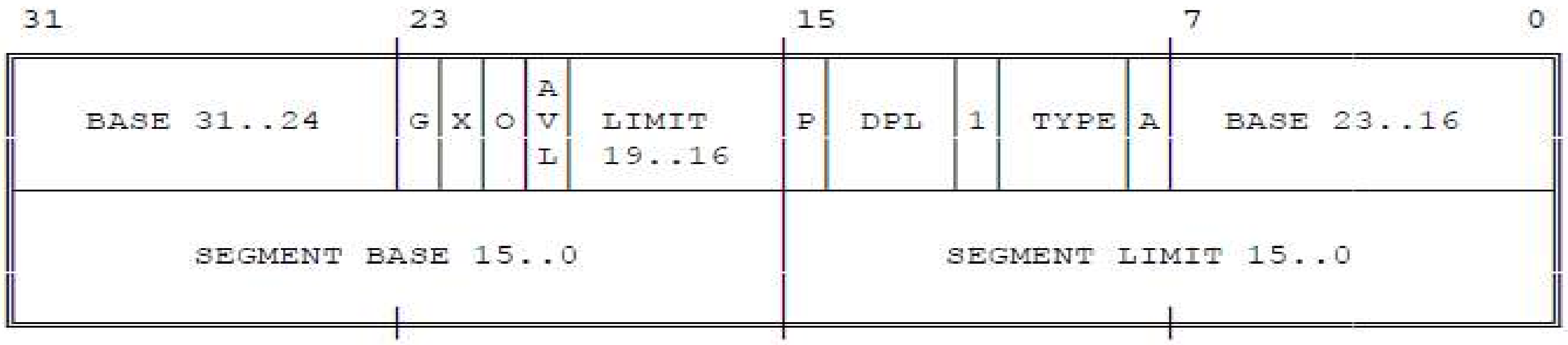


Segment Translation

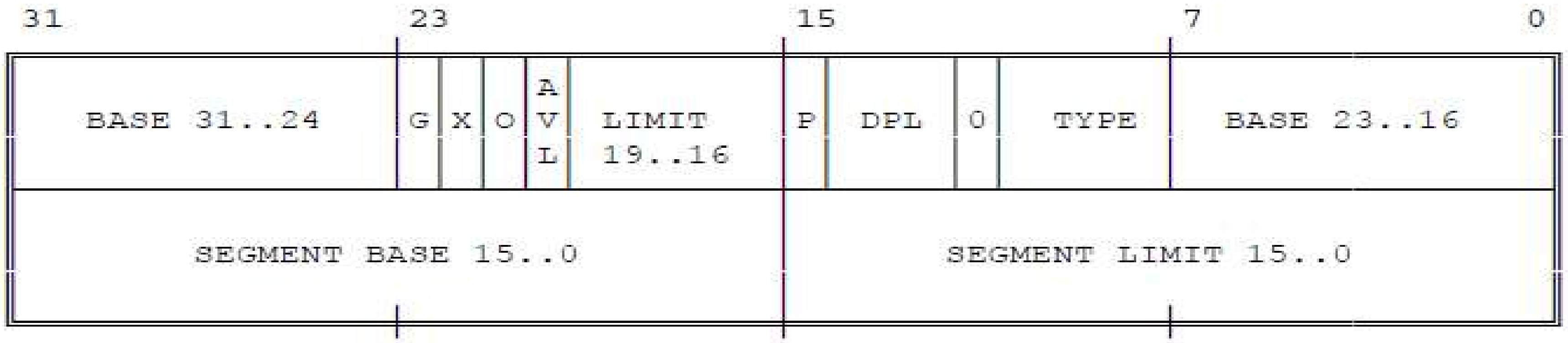
1.1 Descriptors

- The segment descriptor provides the processor with the data it needs to map a logical address into a linear address.
- Descriptors are created by compilers, linkers, loaders, or the operating system, not by applications programmers.
- 2 general descriptor formats (Next Slide)
- All types of segment descriptors take one of these formats.

DESCRIPTORS USED FOR APPLICATIONS CODE AND DATA SEGMENTS



DESCRIPTORS USED FOR SPECIAL SYSTEM SEGMENTS



A	- ACCESSED
AVL	- AVAILABLE FOR USE BY SYSTEMS PROGRAMMERS
DPL	- DESCRIPTOR PRIVILEGE LEVEL
G	- GRANULARITY
P	- SEGMENT PRESENT

General Segment-Descriptor Format

Segment-Descriptor Fields

- BASE : Defines the location of the segment within the 4 gigabyte linear address space. The processor concatenates the three fragments of the base address to form a single 32-bit value.

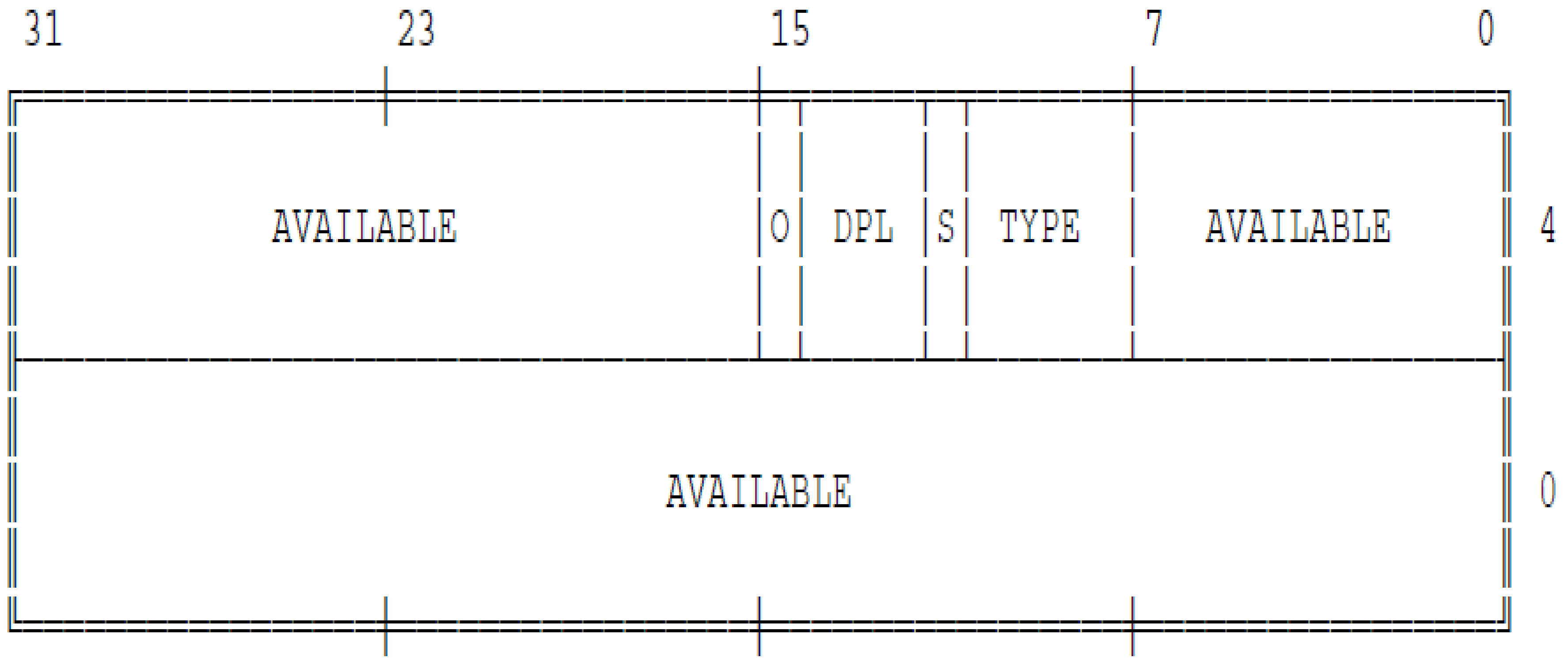
- LIMIT: Defines the size of the segment. When the processor concatenates the two parts of the limit field, a 20-bit value results. The processor interprets the limit field in one of two ways, depending on the setting of the granularity bit:
- In units of one byte, to define a limit of up to 1 megabyte
- In units of 4 Kilobytes, to define a limit of up to 4 gigabytes.

- Granularity bit: Specifies the units with which the LIMIT field is interpreted. When the bit is clear, the limit is interpreted in units of one byte; when set, the limit is interpreted in units of 4 Kilobytes.

- TYPE: Distinguishes between various kinds of descriptors.

- DPL (Descriptor Privilege Level): Used by the protection mechanism.

- Present bit:-If this bit is zero, the descriptor is not valid for use in address transformation; the processor will signal an exception when a selector for the descriptor is loaded into a segment register.



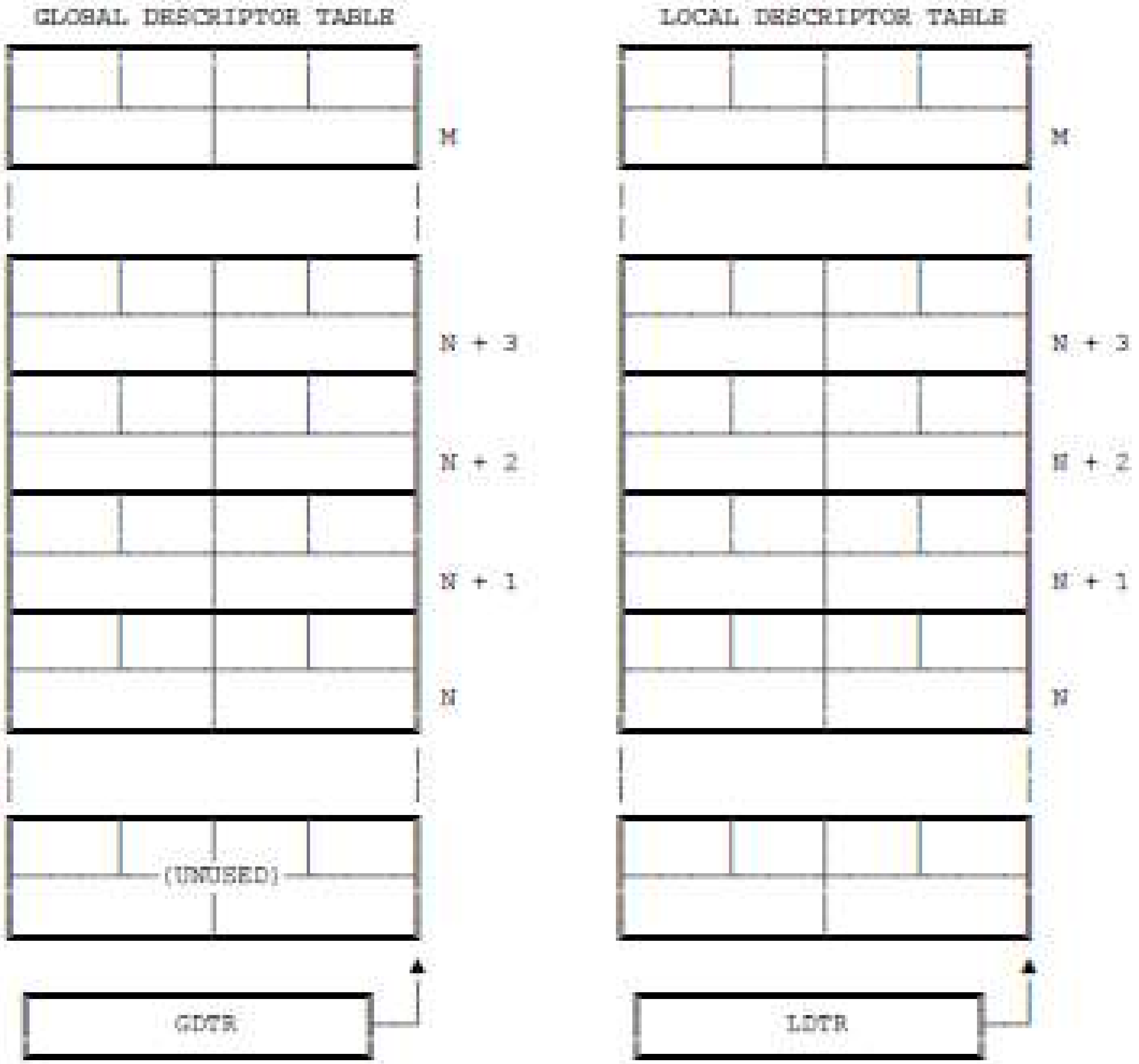
Format of Not-Present Descriptor

- Accessed Bit:-The processor sets this bit when the segment is accessed; i.e., a selector for the descriptor is loaded into a segment register or used by a selector test instruction.

Descriptor Table

- Segment descriptors are stored in either of two kinds of descriptor table:
  1. The global descriptor table (GDT)
  2. A local descriptor table (LDT)
- A descriptor table is simply a memory array of 8-byte entries that contain descriptors.
- A descriptor table is variable in length and may contain up to 8192 (2^{13}) descriptors.
- The first entry of the GDT (INDEX=0) is not used by the processor.
- The processor locates the GDT and the current LDT in memory by means of the GDTR and LDTR registers.
- These registers store the base addresses of the tables in the linear address space and store the segment limits.
- The instructions LGDT and SGDT give access to the GDTR .
- The instructions LLDT and SLDT give access to the LDTR.

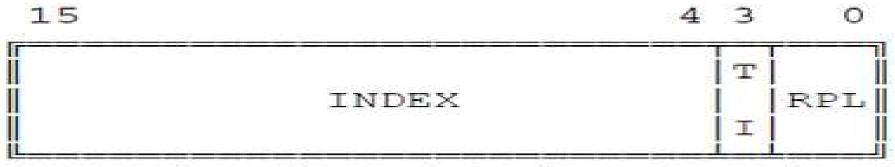
Figure 5-5. Descriptor Tables



Selectors

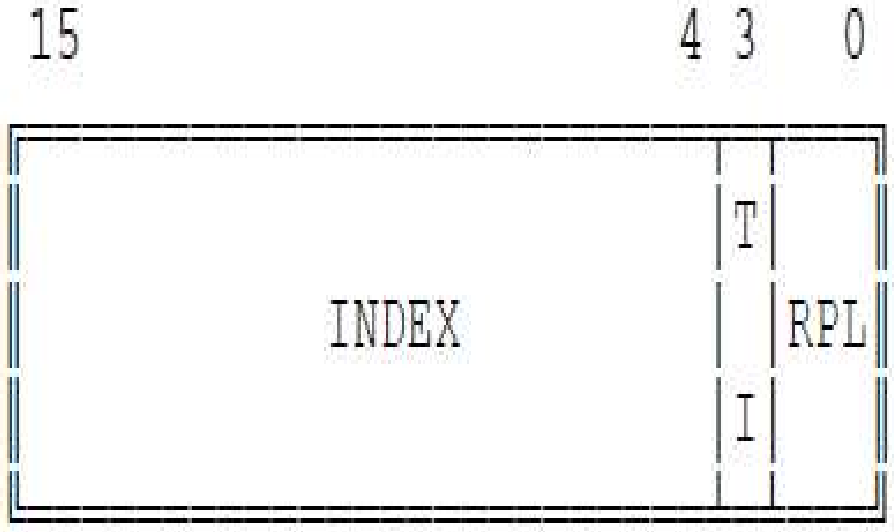
- The selector portion of a logical address identifies a descriptor by specifying a descriptor table and indexing a descriptor within that table.
- Selectors may be visible to applications programs as a field within a pointer variable, but the values of selectors are usually assigned (fixed up) by linkers or linking loaders. The format of a selector

Figure 5-6. Format of a Selector



TI - TABLE INDICATOR
RPL - REQUESTOR'S PRIVILEGE LEVEL

Figure 5-6. Format of a Selector



TI - TABLE INDICATOR
RPL - REQUESTOR'S PRIVILEGE LEVEL

Index: Selects one of 8192 descriptors in a descriptor table. The processor simply multiplies this index value by 8 (the length of a descriptor), and adds the result to the base address of the descriptor table in order to access the appropriate segment descriptor in the table.

•Table Indicator: Specifies to which descriptor table the selector refers. A zero indicates the GDT; a one indicates the current LDT.

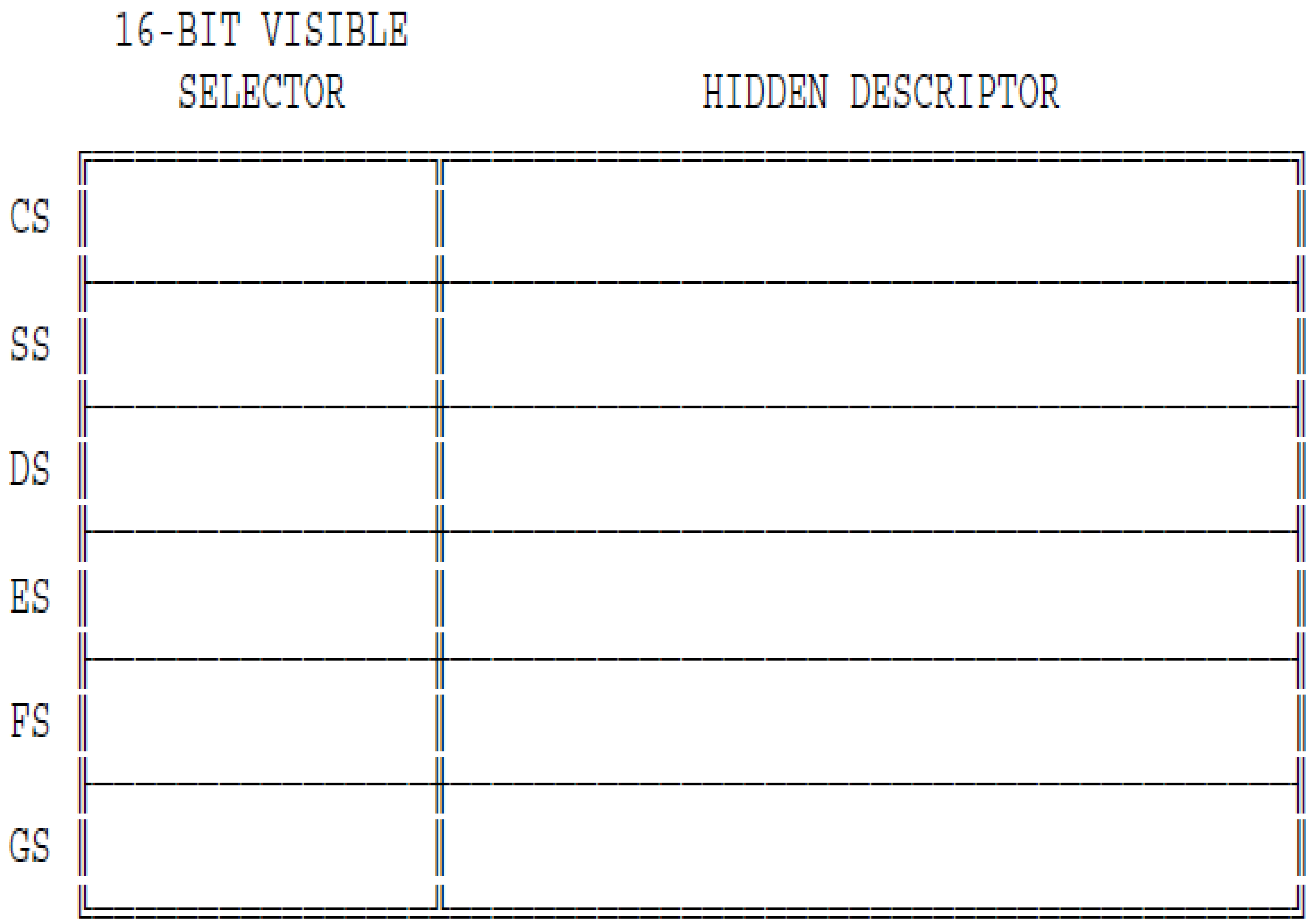
•Requested Privilege Level: Used by the protection mechanism.

- Because the first entry of the GDT is not used by the processor, a selector that has an index of zero and a table indicator of zero (i.e., a selector that points to the first entry of the GDT), can be used as a null selector.
- The processor does not cause an exception when a segment register (other than CS or SS) is loaded with a null selector.
- It will, however, cause an exception when the segment register is used to access memory.
- This feature is useful for initializing unused segment registers so as to trap accidental references.

Segment Registers

- The 80386 stores information from descriptors in segment registers, thereby avoiding the need to consult a descriptor table every time it accesses memory.
- Every segment register has a "visible" portion and an "invisible" portion
- The visible portions of these segment address registers are manipulated by programs as if they were simply 16-bit registers.
- The invisible portions are manipulated by the processor.

Figure 5-7. Segment Registers

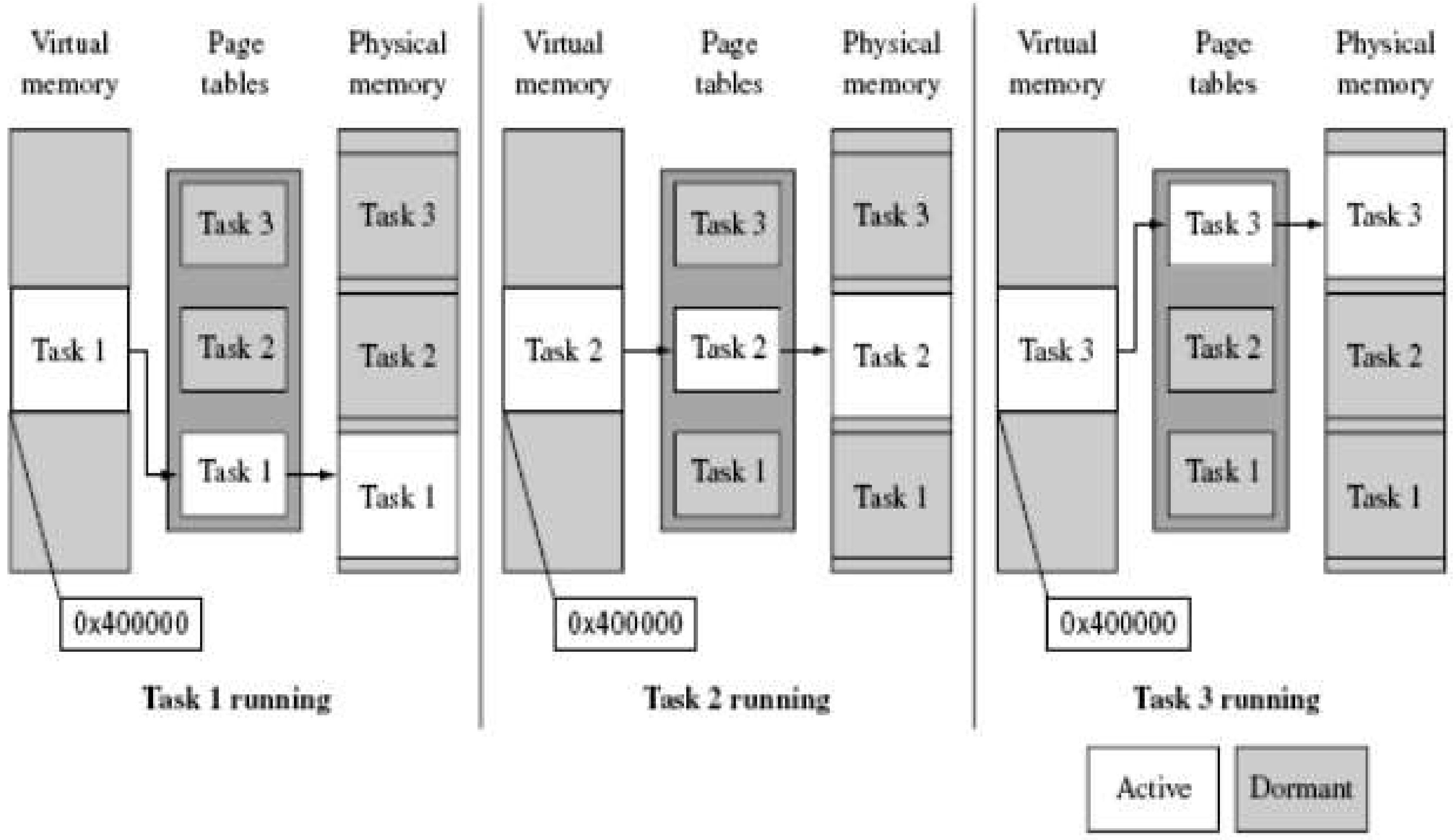


Instructions

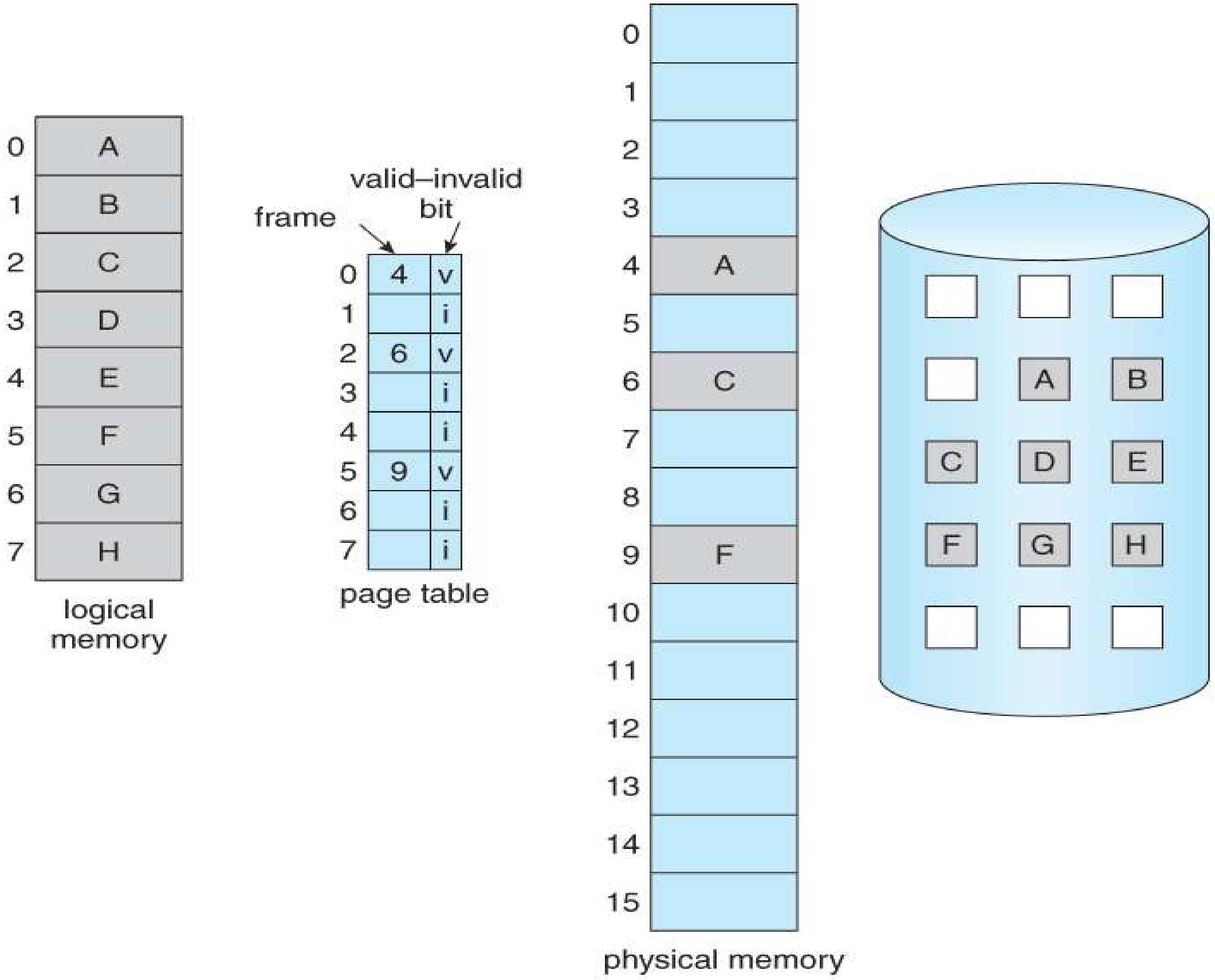
- The operations that load these registers are normal program instructions.
- These instructions are of two classes:

1. Direct load instructions; for example, MOV, POP, LDS, LSS, LGS, LFS : These instructions explicitly reference the segment registers.
2. Implied load instructions; for example, far CALL and JMP : These instructions implicitly reference the CS register, and load it with a new value.

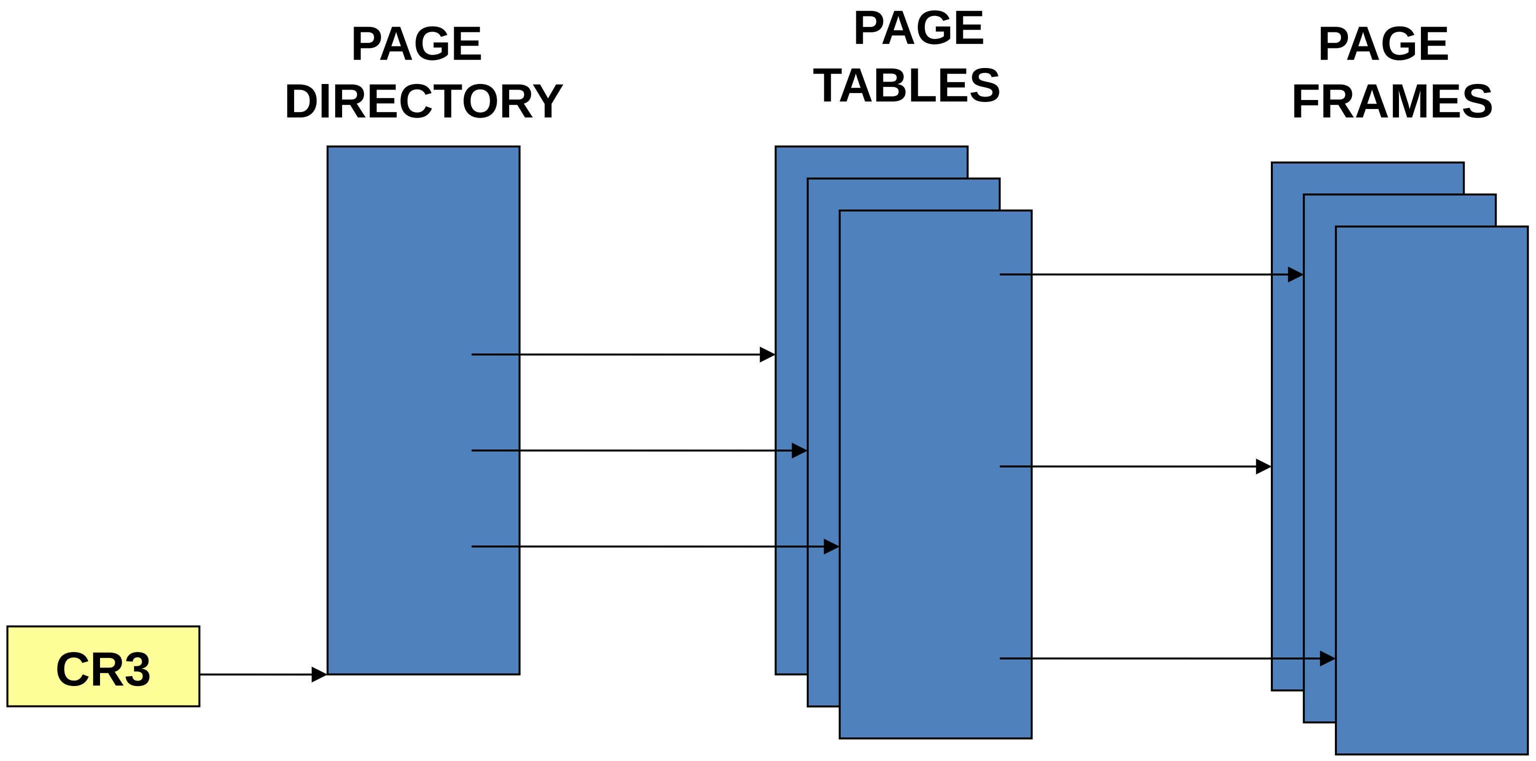
Paging


Virtual memory
Page tables
Physical memory
Task 1
Task 2
Task 3
0x400000
Task 1 running
Task 2 running
Task 3 running
Active
Dormant

Page Translation



Two-Level Translation Scheme



Page Translation

- Page Frame
- Linear Address
- Page Tables
- Page-Table Entries
- Page-Translation Cache

Page Translation

- Optional step
- IInd phase of address translation
- 80386 transforms a linear address into a physical address
- Implements the basic features needed for page-oriented virtual-memory systems and page-level protection

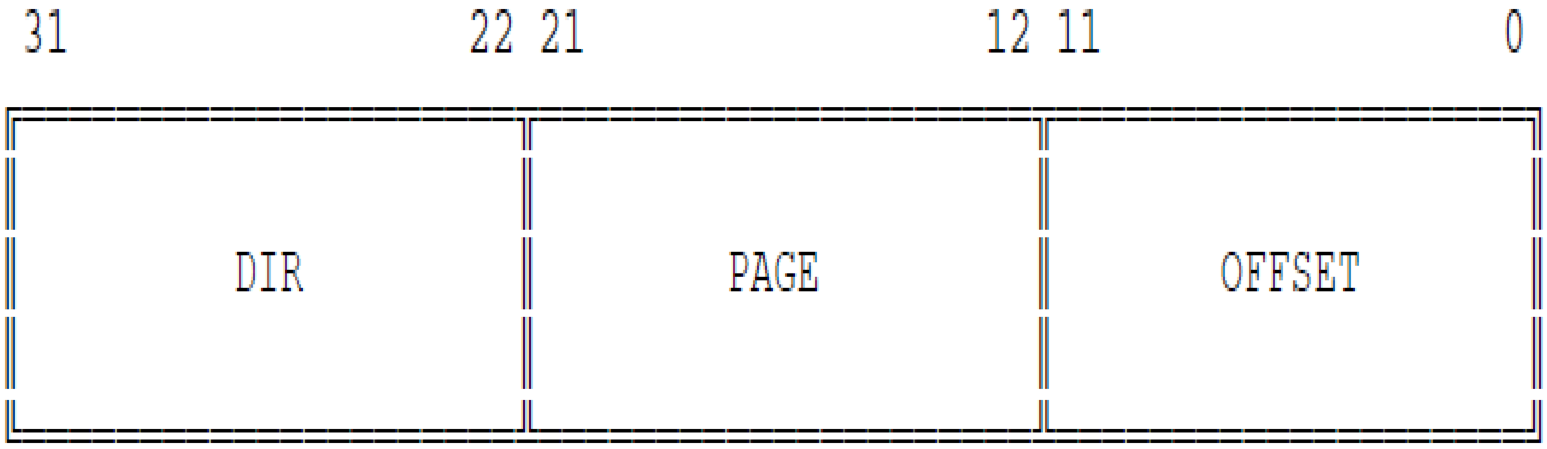
- Page translation is in effect only when the PG bit of CR0 is set.
- This bit is typically set by OS during software initialization.
- The PG bit must be set if OS is to implement multiple virtual 8086 tasks, page-oriented protection, or page-oriented virtual memory.

1.Page Frame

- A page frame is a 4K-byte unit of contiguous addresses of physical memory.
- Pages begin on byte boundaries and are fixed in size.

2. **Linear Address**

- A linear address refers indirectly to a physical address by specifying a page table, a page within that table, and an offset within that page.
- Format of a linear address

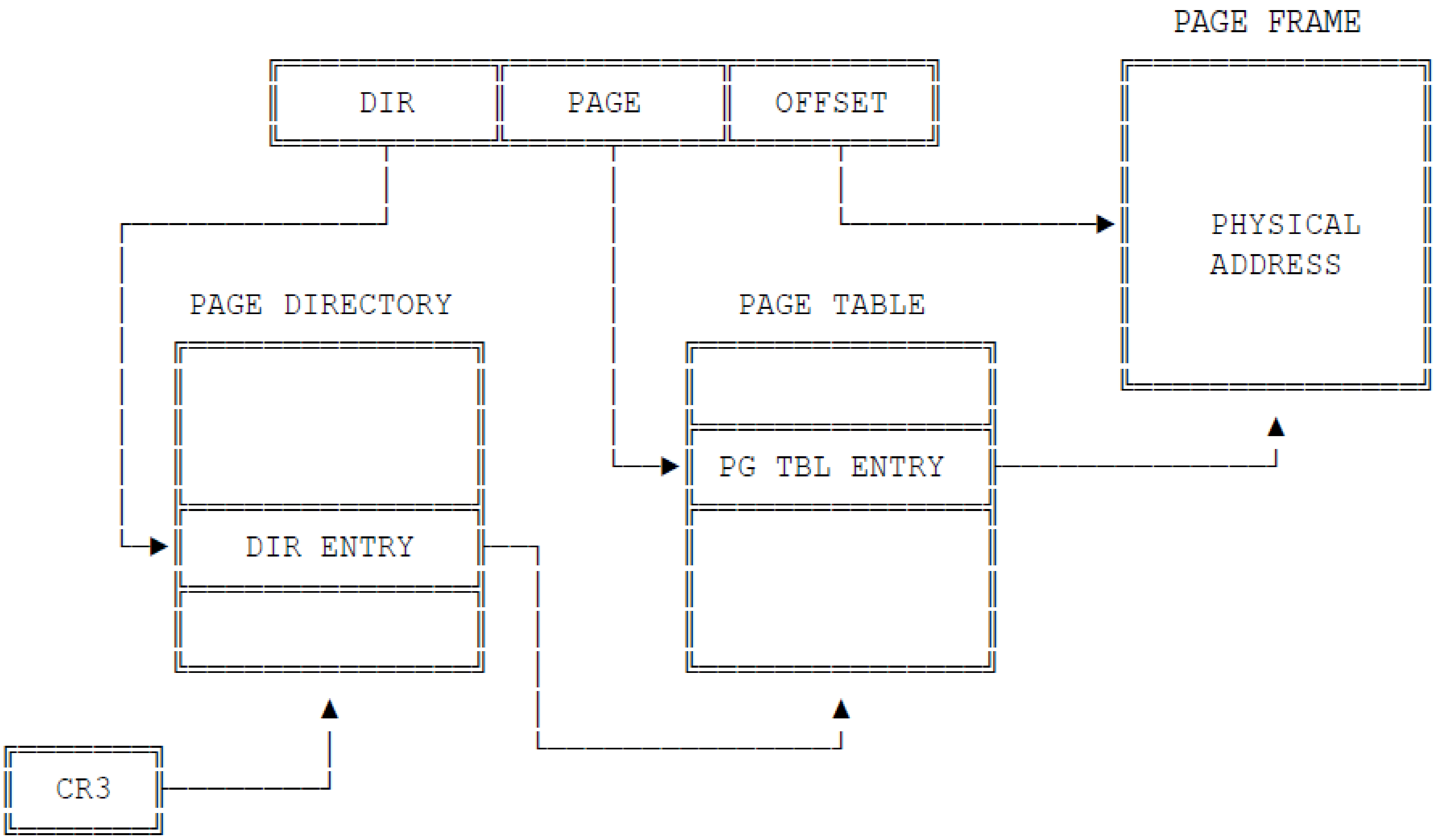


Format of a Linear Address

DIR, PAGE, OFFSET

- Processor converts the DIR, PAGE, and OFFSET fields of a linear address into the physical address by consulting two levels of page tables.
- The addressing mechanism uses the DIR field as an index into a page directory, uses the PAGE field as an index into the page table determined by the page directory, and uses the OFFSET field to address a byte within the page determined by the page table.

Figure 5-9. Page Translation



3. **Page Tables**

- A page table is simply an array of 32-bit page specifiers.
- A page table is itself a page, and therefore contains 4 Kilobytes of memory or at most 1K 32-bit entries.

Table Levels

- Two levels of tables are used to address a page of memory.
- At the higher level is a page directory.
- The page directory addresses up to 1K page tables of the second level.
- A page table of the second level addresses up to 1K pages.
- All the tables addressed by one page directory, therefore, can address 1M pages (2^{20}).
- Because each page contains 4K bytes 2^{12} bytes), the tables of one page directory can span the entire physical address space of the 80386 (2^{20} times $2^{12} = 2^{32}$).

CR3 Usage

- The physical address of the current page directory is stored in the CPU register CR3, also called the page directory base register (PDBR).
- Memory management software has the option of using one page directory for all tasks, one page directory for each task, or some combination of the two.

4. Page-Table Entries

- Entries in either level of page tables have the same format.
- Format

1.Page Frame Address
2.Present Bit
3.Accessed and Dirty Bits
4.Read/Write and User/Supervisor Bits

Figure 5-10. Format of a Page Table Entry

```
 31                                  12 11                      0
╔═══════════════════════════════════════╤═══════╤═══╤═╤═╤═══╤═╤═╤═╗
║                                       │       │   │ │ │   │U│R│ ║
║      PAGE FRAME ADDRESS 31..12        │ AVAIL │0 0│D│A│0 0│/│/│P║
║                                       │       │   │ │ │   │S│W│ ║
╚═══════════════════════════════════════╧═══════╧═══╧═╧═╧═══╧═╧═╧═╝
```

P	-	PRESENT
R/W	-	READ/WRITE
U/S	-	USER/SUPERVISOR
D	-	DIRTY
AVAIL	-	AVAILABLE FOR SYSTEMS PROGRAMMER USE

NOTE: 0 INDICATES INTEL RESERVED. DO NOT DEFINE.

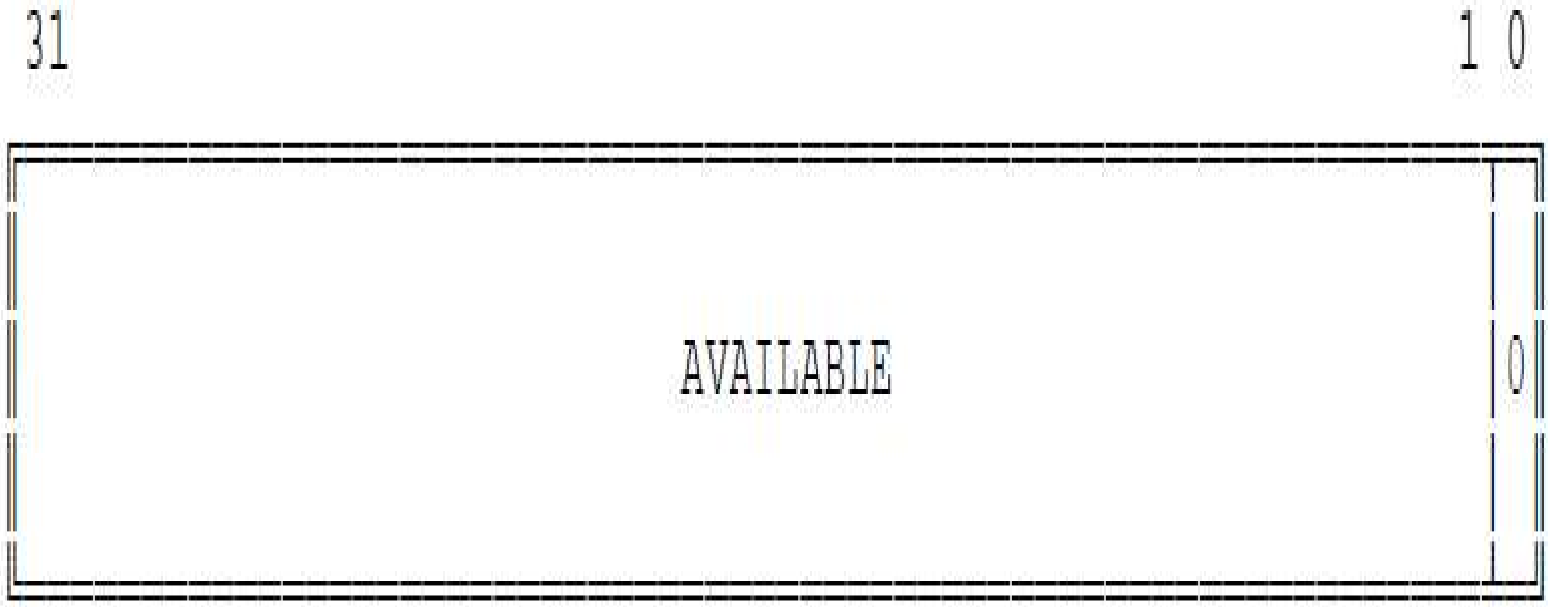
4.1 Page Frame Address

- The page frame address specifies the physical starting address of a page.
- Because pages are located on 4K boundaries, the low-order 12 bits are always zero.
- In a page directory, the page frame address is the address of a page table.
- In a second-level page table, the page frame address is the address of the page frame that contains the desired memory operand.

4.2 **Present Bit**

- The Present bit indicates whether a page table entry can be used in address translation.
- P=1 indicates that the entry can be used (Page in the memory).
- When P=0 in either level of page tables, the entry is not valid for address translation, and the rest of the entry is available for software use; none of the other bits in the entry is tested by the hardware (Page is not in the physical memory).
- Format of a page-table entry when P=0

Figure 5-11. Invalid Page Table Entry



- If P=0 in either level of page tables when an attempt is made to use a page-table entry for address translation, the processor signals a page exception.
- In software systems that support paged virtual memory, the page-not-present exception handler can bring the required page into physical memory.
- The instruction that caused the exception can then be reexecuted.

4.3 **Accessed and Dirty Bits**

- These bits provide data about page usage in both levels of the page tables.
- With the exception of the dirty bit in a page directory entry, these bits are set by the hardware
- The processor does not clear any of these bits.

- The processor sets the corresponding accessed bits in both levels of page tables to one before a read or write operation to a page.
- The processor sets the dirty bit in the second-level page table to one before a write to an address covered by that page table entry.
- The dirty bit in directory entries is undefined.

- An OS that supports paged virtual memory can use these bits to determine what pages to eliminate from physical memory when the demand for memory exceeds the physical memory available.
- The operating system is responsible for testing and clearing these bits.

4.4 **Read/Write and User/Supervisor Bits**

- These bits are not used for address translation
- Use : for page-level protection, which the processor performs at the same time as address translation.
- Only two types of pages are recognized by the protection mechanism:

1. Read-only access (R/W=0).
2. Read/write access (R/W=1).

Imp…..

- With pages, there are two levels of privilege:

1. Supervisor level (U/S=0)—for the OS, other system software (such as device drivers), and protected system data (such as page tables).
2. User level (U/S=1)—for application code and data.

Imp…..

- When the processor is running at supervisor level, all pages are accessible.
- When the processor is running at user level, only pages from the user level are accessible.

5.Page Translation Cache

- Processor stores the most recently used page-table data in an on-chip cache.
- Only if the necessary paging information is not in the cache must both levels of page tables be referenced.
- The existence of the page-translation cache is invisible to applications programmers but not to systems programmers; OS programmers must flush the cache whenever the page tables are changed.

- The page-translation cache can be flushed by either of two methods:

1. By reloading CR3 with a MOV instruction;

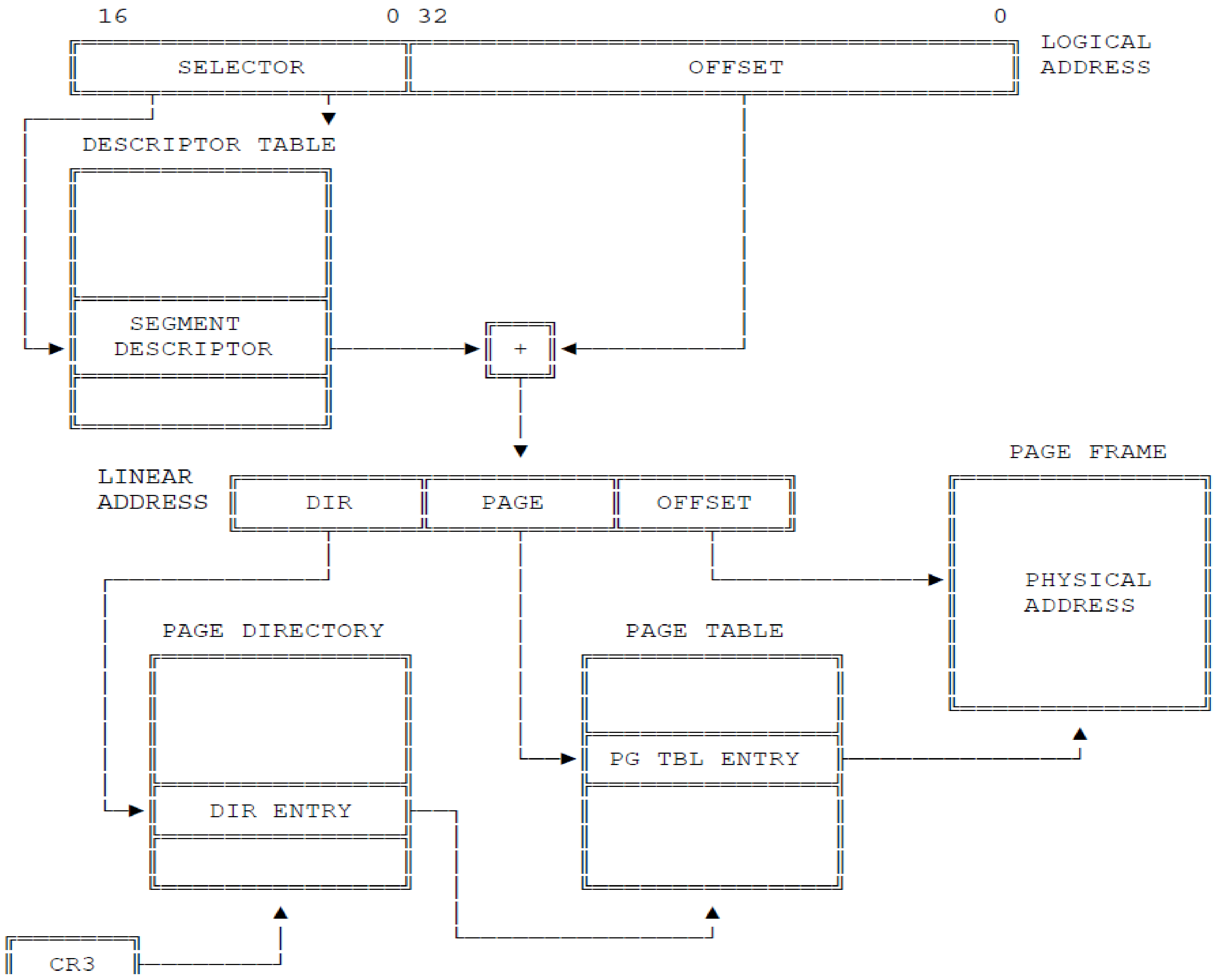
   for example: MOV CR3, EAX

2. By performing a task switch to a TSS (Task State Segment) that has a different CR3 image than the current TSS.

Combining Segment and Page Translation

- By appropriate choice of options and parameters to both phases, memory-management software can implement several different styles of memory management.

Figure 5-12. 80306 Addressing Machanism



5.1 **"Flat" Architecture**

- When the 80386 is used to execute software designed for architectures that don't have segments, it may be expedient to effectively "turn off" the segmentation features of the 80386.
- The 80386 does not have a mode that disables segmentation, but the same effect can be achieved by initially loading the segment registers with selectors for descriptors that encompass the entire 32-bit linear address space

5.2 **Segments Spanning Several Pages**

•The architecture of the 80386 permits segments to be larger or smaller than the size of a page (4 Kilobytes).

•For example, suppose a segment is used to address and protect a large data structure that spans 132 Kilobytes.

•In a software system that supports paged virtual memory, it is not necessary for the entire structure to be in physical memory at once.

• The structure is divided into 33 pages, any number of which may not be present

5.3 Pages Spanning Several Segments

•Segments may be smaller than the size of a page.
•For example, consider a small data structure such as a semaphore.
• Because of the protection and sharing provided by segments it may be useful to create a separate segment for each semaphore.
•But, because a system may need many semaphores, it is not efficient to allocate a page for each.
•Therefore, it may be useful to cluster many

5.4 Non-Aligned Page and Segment Boundaries

- The architecture of the 80386 does not enforce any correspondence between the boundaries of pages and segments.
- It is perfectly permissible for a page to contain the end of one segment and the beginning of another.
- Likewise, a segment may contain the end of one page and the beginning of another.

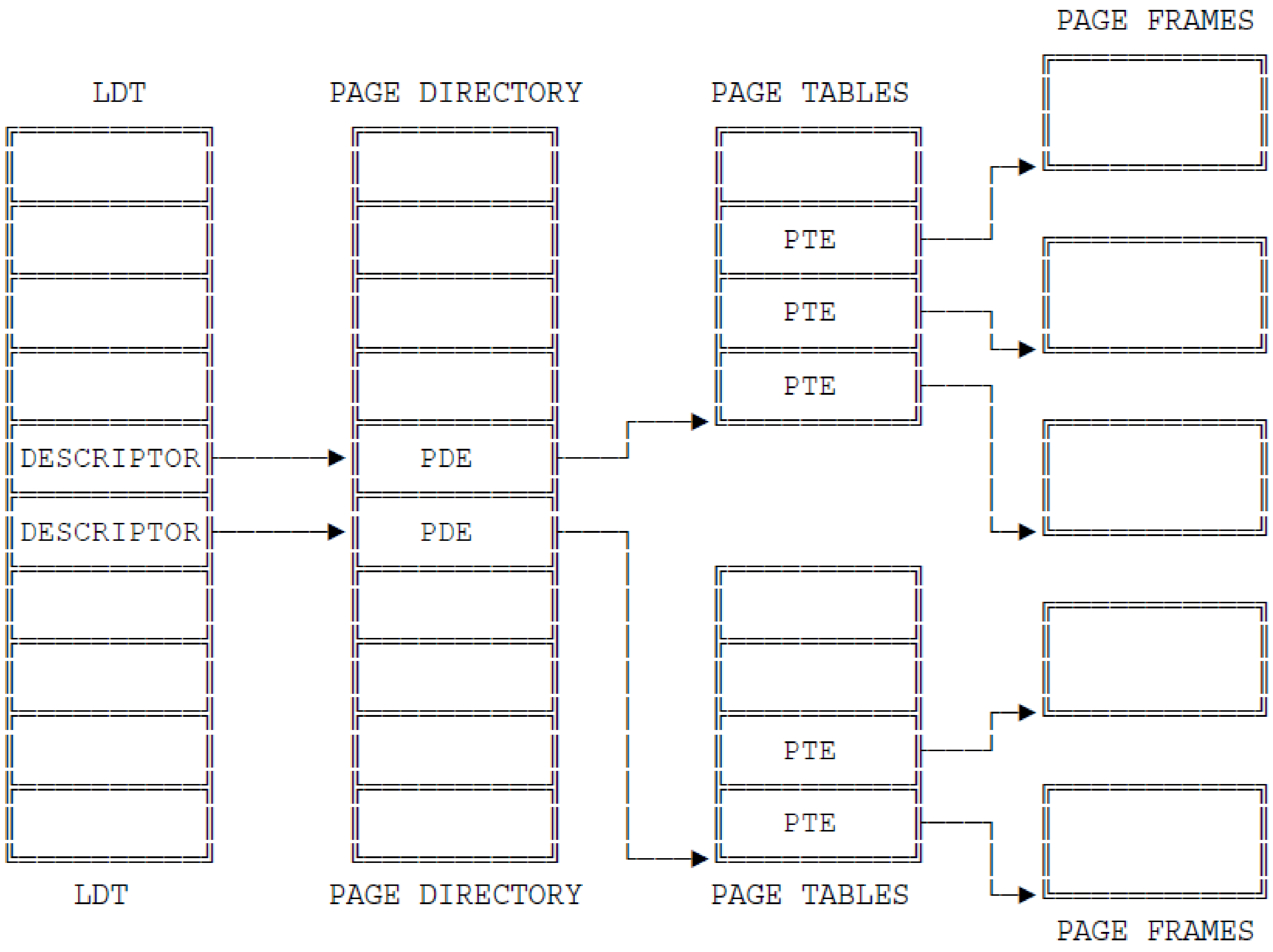
5.5 Aligned Page and Segment Boundaries

- Memory-management software may be simpler, however, if it enforces some correspondence between page and segment boundaries.
- For example, if segments are allocated only in units of one page, the logic for segment and page allocation can be combined.
- There is no need for logic to account for partially used pages.

5.6 Page-Table per Segment

- An approach to space management that provides even further simplification of space-management software is to maintain a one-to-one correspondence between segment descriptors and page-directory entries.
- Sample next
- Each descriptor has a base address in which the low-order 22 bits are zero; in other words, the base address is mapped by the first entry of a page table.
- A segment may have any limit from 1 to 4

Figure 5-13. Descriptor per Page Table


PAGE FRAMES
LDT
PAGE DIRECTORY
PAGE TABLES
PTE
PTE
PTE
DESCRIPTOR
PDE
DESCRIPTOR
PDE
PTE
PTE
LDT
PAGE DIRECTORY
PAGE TABLES
PAGE FRAMES