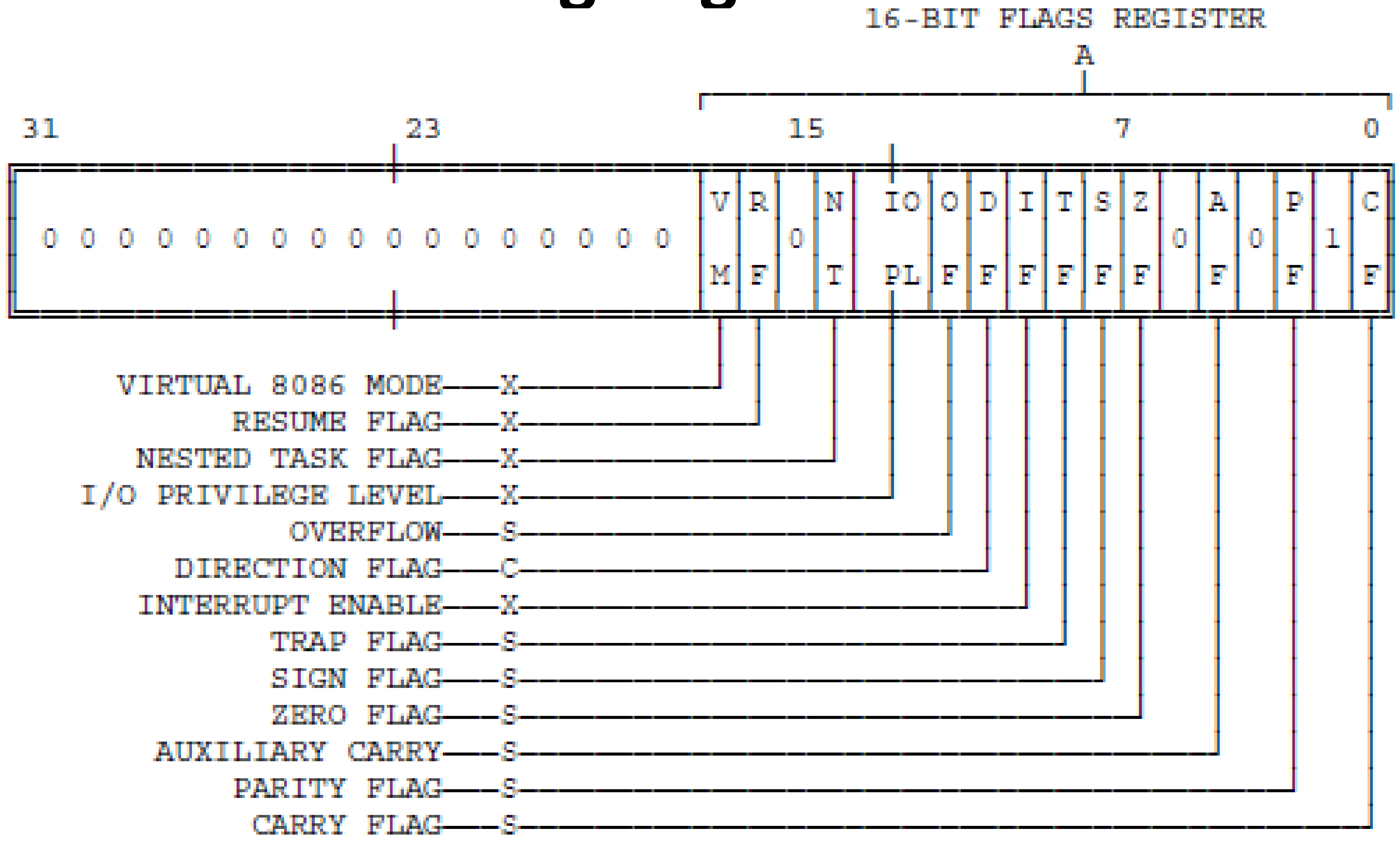
UNIT 2
Systems Architecture and Memory Management

System Registers

- EFLAGS
- System Registers
- Control Registers
- Debug Registers
- Test Registers

Flag Registers



S = STATUS FLAG, C = CONTROL FLAG, X = SYSTEM FLAG

- **VM** (Virtual 8086 Mode): If set while the Intel386 DX is in Protected Mode, the Intel386 DX will switch to Virtual 8086 operation.
- The VM bit can be set only in Protected Mode, by the IRET instruction (if current privilege level e 0)

- **RF** (Resume Flag): The RF flag is used in conjunction with the debug register breakpoints.
- When RF is set, it causes any debug fault to be ignored on the next instruction.

- **NT** (Nested Task): This flag applies to Protected Mode.
- NT is set to indicate that the execution of this task is nested within another task
- The value of NT in EFLAGS is tested by the IRET instruction to determine whether to do an inter-task return or an intra-task return.

IOPL (Input / Output Privilege Level)

- This two-bit field applies to Protected Mode.

  IOPL indicates the numerically maximum CPL(current privilege level) value permitted to execute I/O instructions without generating an Exception
- It also indicates the maximum CPL value allowing alteration of the IF (INTR Enable Flag) bit when new values are popped into the EFLAG register

- **IF** (INTR Enable Flag): The IF flag, when set, allows recognition of external interrupts signaled on the INTR pin.

- **TF** (Trap Enable Flag): When TF is set, the Intel386 DX generates an exception 1 trap after the next instruction is executed.
- When TF is reset, exception 1 traps occur only as a function of the breakpoint addresses loaded into debug registers DR0-DR3.

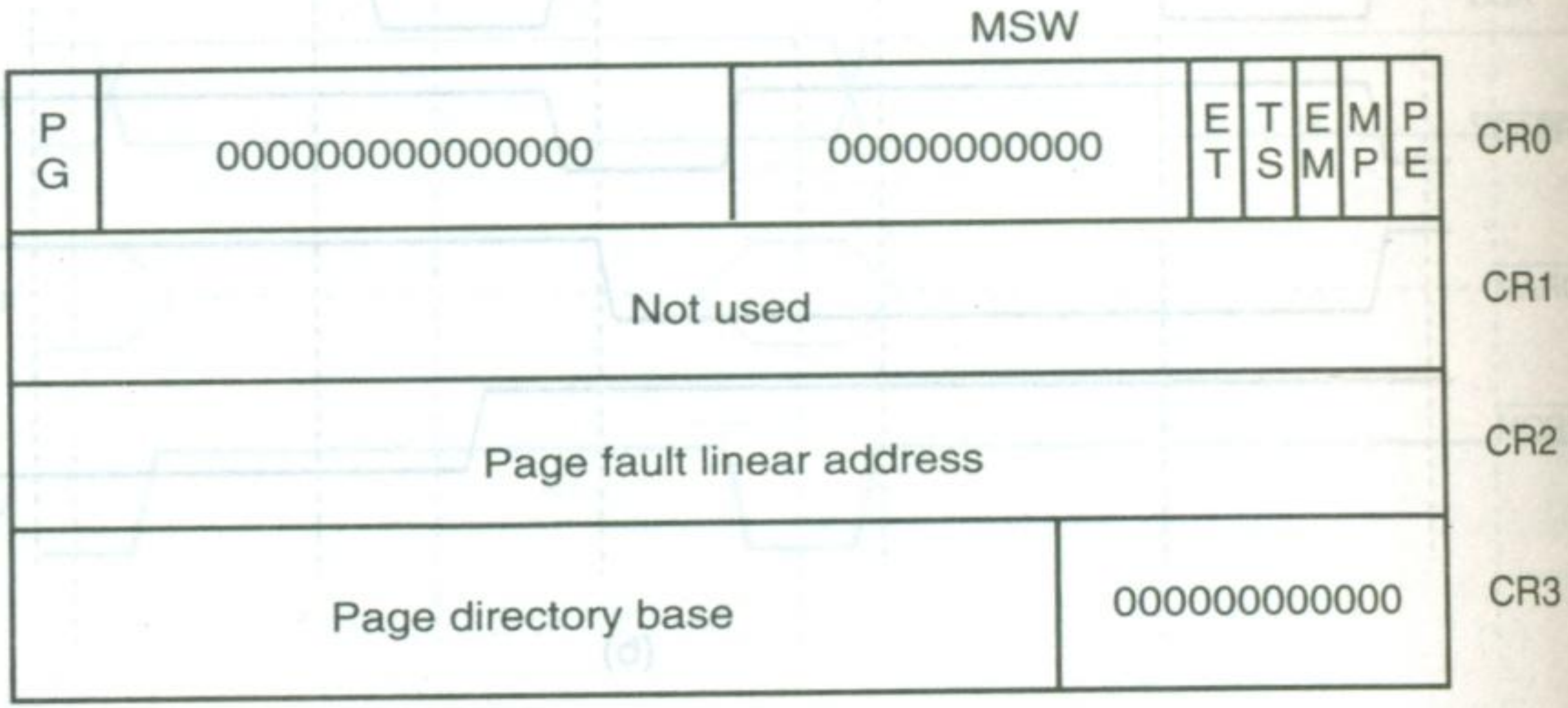
- **OF** (Overflow Flag) : It is set if the operation resulted in a signed overflow. Signed overflow occurs when the operation resulted in carry/borrow into the sign bit (high-order bit) of the result.

- **DF** (Direction Flag) : DF defines whether ESI and/or EDI registers post-decrement or post-increment during the string instructions.
- Post-decrement occurs if DF is set

Flags

- The arithmetic instructions use CF, SF, ZF, AF, PF, CF
- The control flag DF controls “STRING” instruction
- Clearing DF flag causes string instructions to auto increment or to process string from low to high address

Control Register



CR

- The CR0 is identical to the MSW (**Machine Status Word**) found in 80286 except that this is 32 bit wide.
- CR1 is not used in 80386.
- CR2 hold the **Linear Page Address of the last page accessed** before a page fault interrupt.
- CR3 holds the **Page Directory Base Address**.

PG (Paging Enable):

Selects **Page Table Translation of linear addresses into physical addresses when PG=1**. Page table translation allows any linear address to be assigned any physical memory location.

ET (Processor Extension Type):

Selects the 80287 coprocessor when ET=0 or the **80387 coprocessor when ET=1**.

This bit was installed because there was no 80387 available when the 80386 first appeared.

TS (Task Switch):

Indicate that the 80386 has switched tasks(In protected mode ,changing the content of TR places a 1 in TS).

EM (Emulate Processor Extension):

The emulate bit set **to cause a type 7 interrupt for each ESC instruction.** (ESC instructions are used to encode instruction for the 80387 coprocessor. So when it is set co-processor generates coprocessor not available fault).

[Type 7 Interrupt: Co-Processor Not Available.]

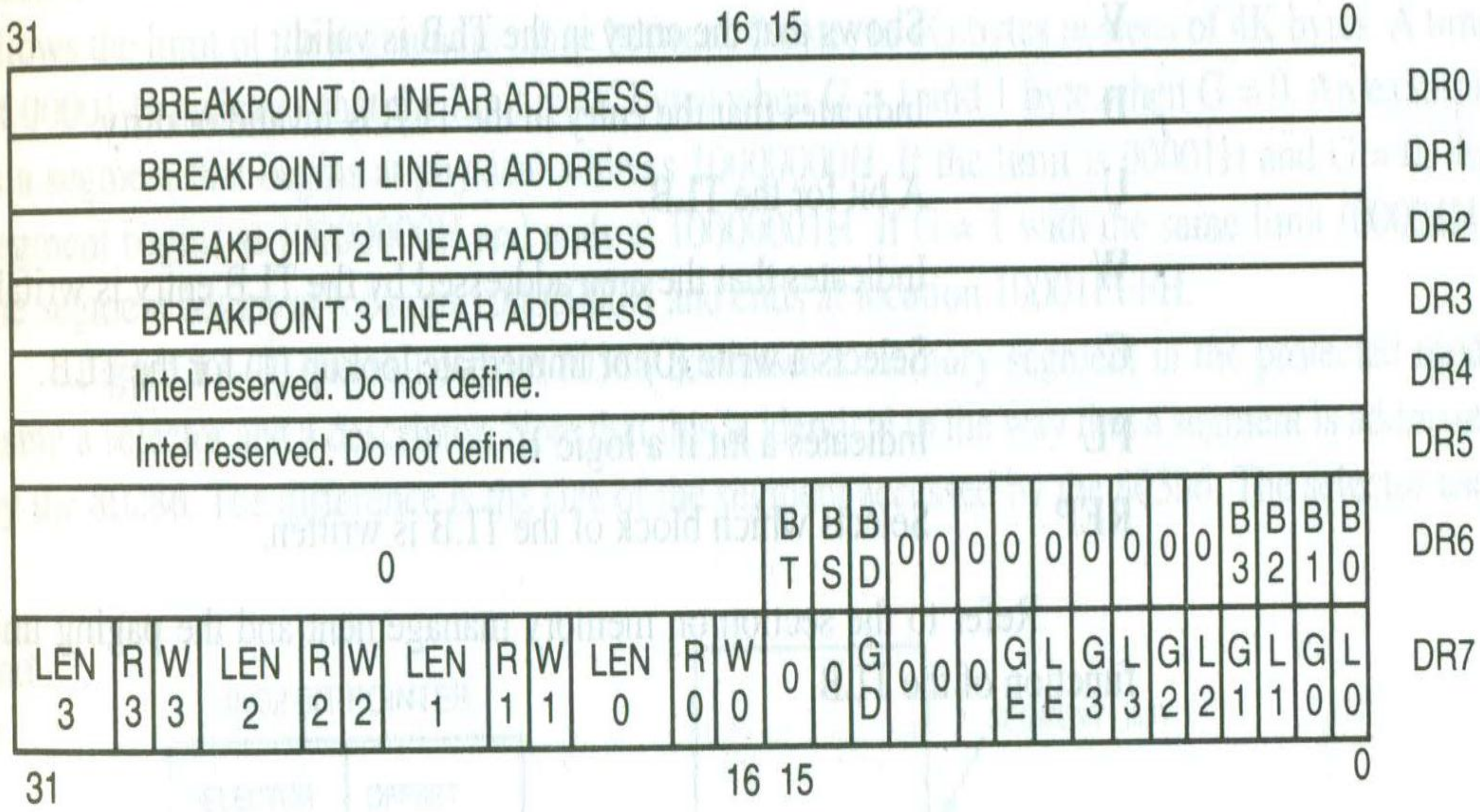
MP (Monitor Processor Extension):

Is reset to indicate that the **arithmetic coprocessor is present in the system.**

PE (Protection Enable):

Is set **to select the Protected Mode of operation for the 80386.**It may also cleared to reenter the real mode.

Debug Register


31
16 15
0
BREAKPOINT 0 LINEAR ADDRESS
DR0
BREAKPOINT 1 LINEAR ADDRESS
DR1
BREAKPOINT 2 LINEAR ADDRESS
DR2
BREAKPOINT 3 LINEAR ADDRESS
DR3
Intel reserved. Do not define.
DR4
Intel reserved. Do not define.
DR5
0
BT
BS
BD
0 0 0 0 0 0 0 0 0
B3
B2
B1
B0
DR6
LEN 3
R 3
W 3
LEN 2
R 2
W 2
LEN 1
R 1
W 1
LEN 0
R 0
W 0
0
0
GD
0 0 0
GE
LE
G3
L3
G2
L2
G1
L1
G0
L0
DR7
31
16 15
0

- These are **used to control debug functions.**
- The first four debug register contain 32 bit linear break point addresses.
- The breakpoint addresses ,which may locate an instruction, **are constantly compared with the addresses generated by the program.**
- If a match occurs , **the 80386 will cause a type 1 interrupt(trap or debug) to occur**, if directed by debug registers DR6 or DR7.

BT:

If set the **debug interrupt was caused by a task switch.**

BS:

If set the **debug interrupt was caused by the TF bit** in the flag register.

BD:

If set the **debug interrupt was caused by an attempt to read the debug register with the GD bit set.**

The GD bit protects access to the debug registers.

B3-B0:

Indicate which of the 4 debug breakpoints addresses caused the debug interrupt.

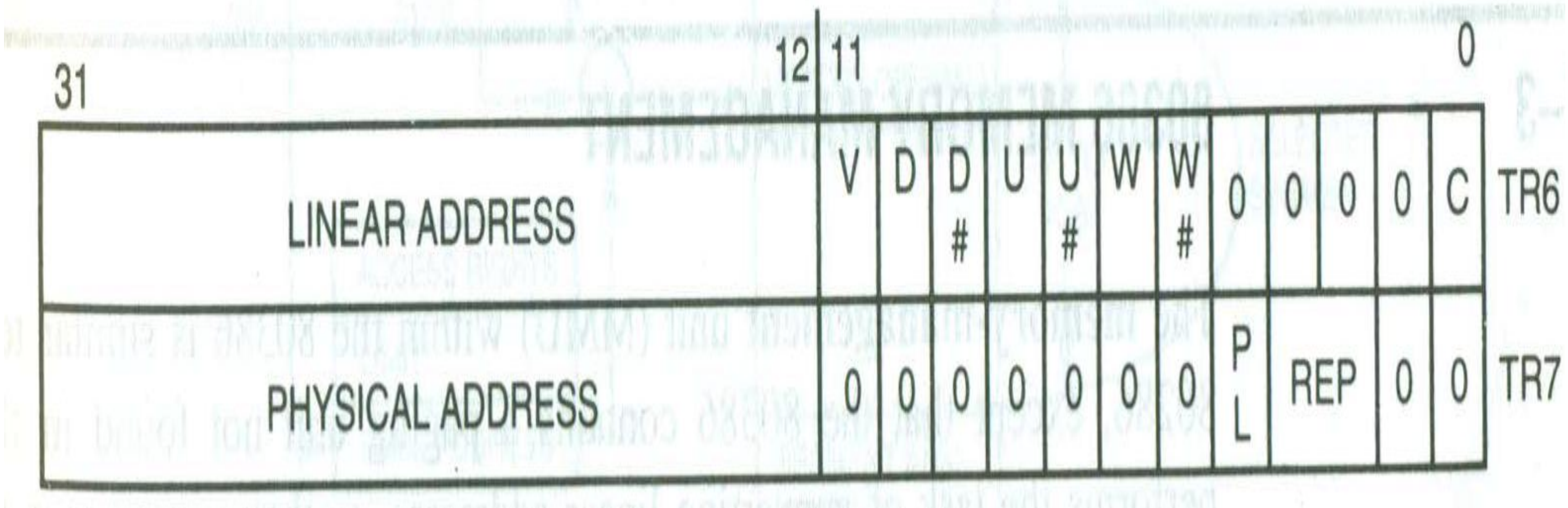
LEN:

Defines the **size of access at the breakpoint** address as 00(byte), 01(word), 10(Currently Not Used) or 11 (double word).

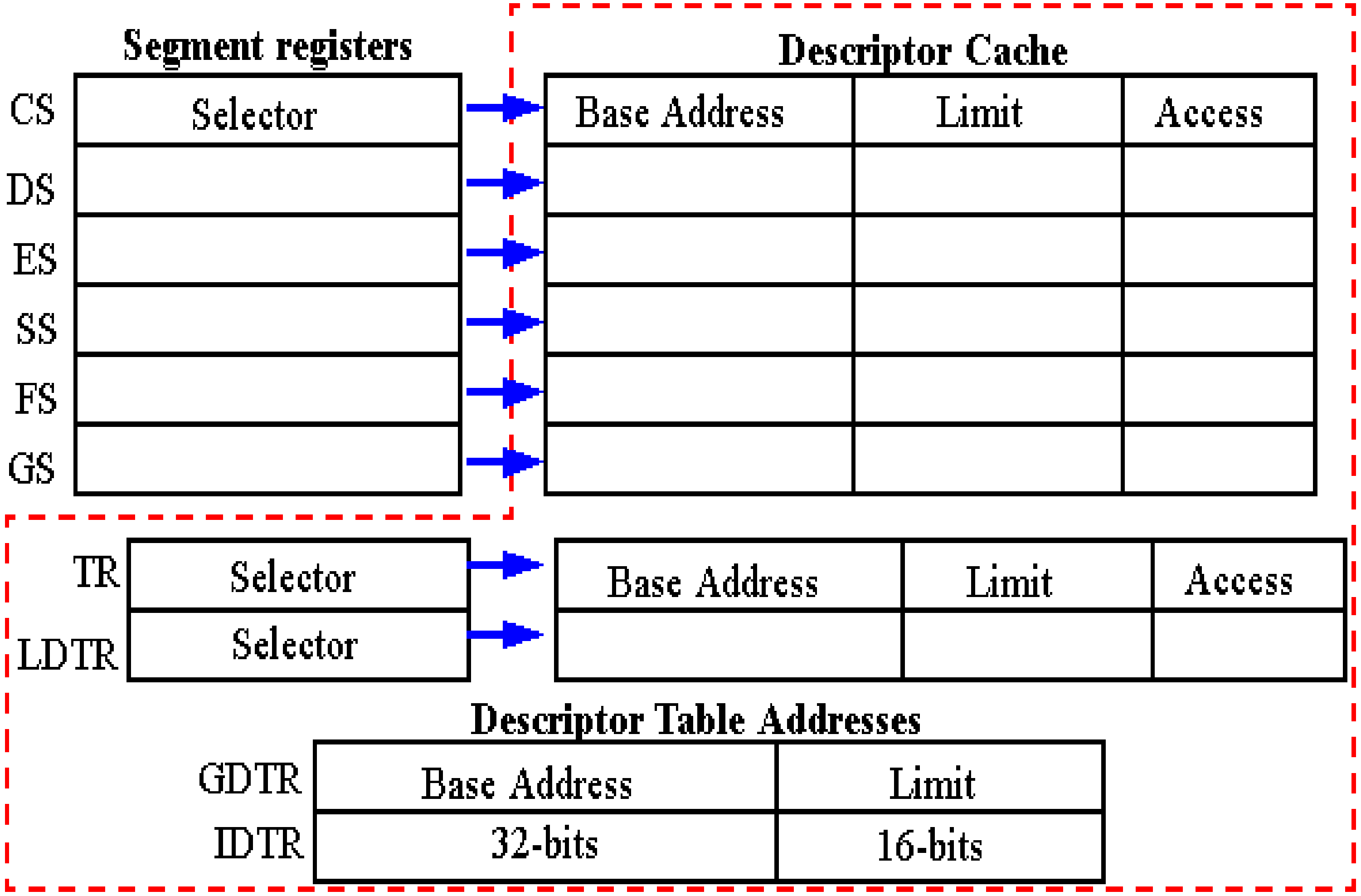
RW:

Selects the cause of action that that enabled breakpoint address as 00 (instruction access),01(data write), 10(Currently Not Used), 11(data read n write).

Test Register



Descriptor Cache



Segment Translation

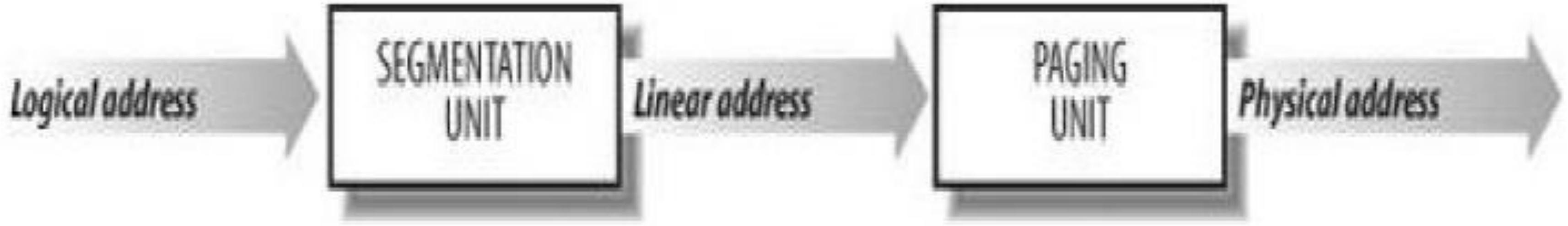
- Descriptors Tables
- Selectors
- Descriptor
- Segment Registers

Memory Addresses

- **Logical address**: Included in the machine language instructions to specify the address of an operand or of an instruction.
  - Embodies the well-known 80 x 86 segmented architecture.
  - Consists of a <u>segment and an offset.</u>

- **Linear address (virtual address):** A single 32-bit unsigned integer.
  - Can be used to address up to 4 GB
  - Usually represented in <u>hexadecimal notation</u>.

- **Physical address:** Used to address memory cells in memory chips.
  - Physical addresses are represented as 32-bit or 36-bit unsigned integers.


Logical address
SEGMENTATION UNIT
Linear address
PAGING UNIT
Physical address

Segmentation

- Starting with the 80286 model, Intel microprocessors perform address translation in two different ways called **real mode and protected mode.**

- Real mode exists mostly to maintain processor **compatibility with older models** and to **allow the operating system to bootstrap.**

Descriptor Tables

- Global Descriptor Table (GDT)
- Local Descriptor Table (LDT)
- Interrupt Descriptor Table (IDT)

Selectors and Descriptors

- **A logical address consists of two parts: a segment identifier and an offset that** specifies the relative address within the segment.

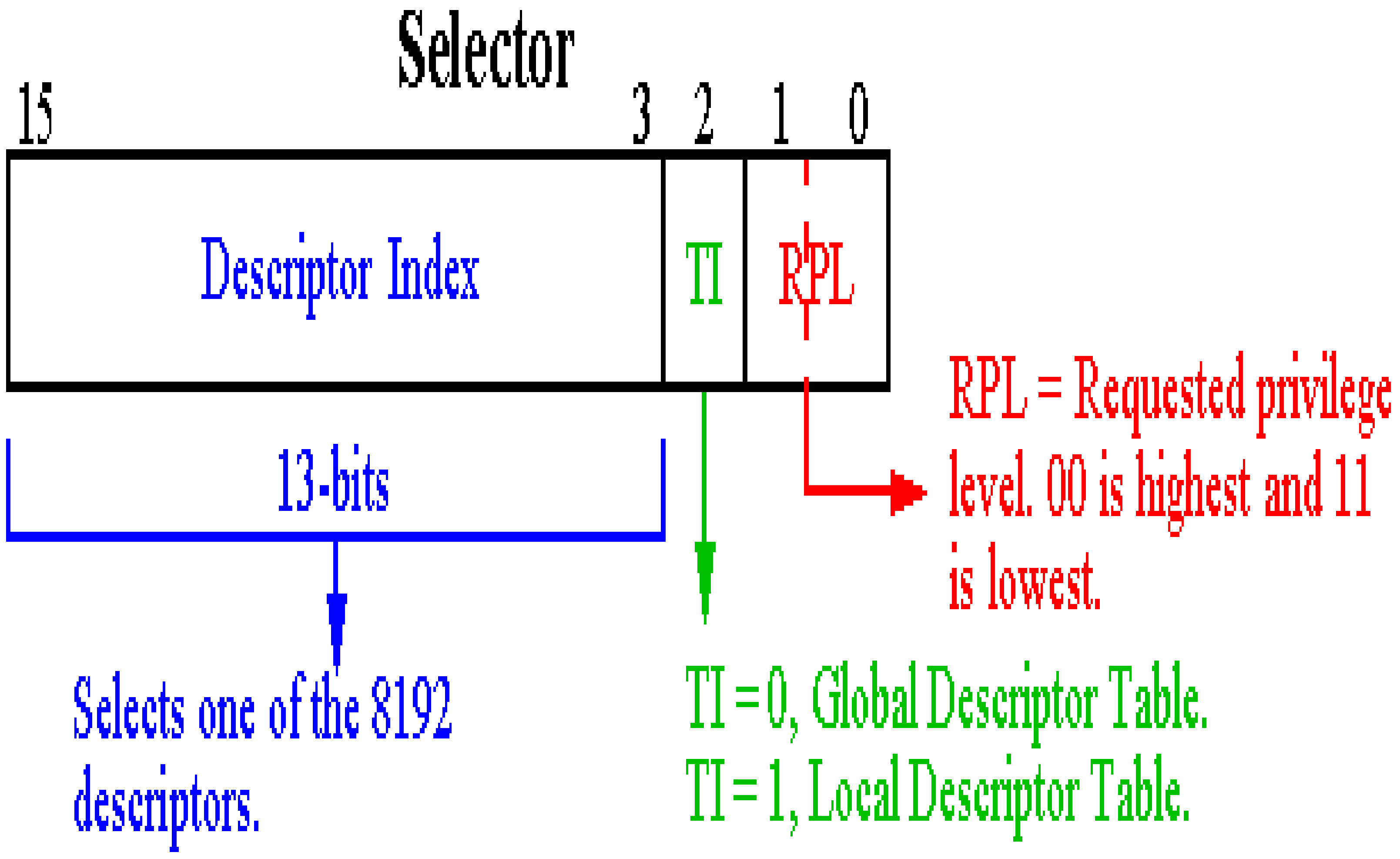
- The segment identifier is a 16-bit field called the **Segment Selector, while the offset** is a 32-bit field.

- To make it easy to retrieve segment selectors quickly, the processor provides **segmentation registers whose only purpose is to hold Segment Selectors;** these registers are called CS, SS, DS, ES, FS, and GS.

Selectors

Segment Registers are now called **Segment Selectors** and point to structure called a **Segment Descriptor.**

Segment selector contain a **13 bit index field** that is used to select one of **8192 segment** descriptor that resides either in Global Descriptor Table (**GDT**) or Local Descriptor Table (**LDT**).


Selector
15
3
2
1
0
Descriptor Index
TI
RPL
13-bits
Selects one of the 8192 descriptors.
TI = 0, Global Descriptor Table.
TI = 1, Local Descriptor Table.
RPL = Requested privilege level. 00 is highest and 11 is lowest.

- There is **only one GDT in protected mode.**
- **Protected mode tasks**, however ,may **each have their own LDT.**
- The **TI bit in the segment selector picks the appropriate descriptor table** during translation.
- Two **Requestor Privilege Level (RPL) bits are used in protection check** to determine if access to segment is allowed.

- Selector may be loaded into any of the six segment registers (CS,DS,SS,ES,FS,GS).

- A selector that has an index value of zero and points to GDT is called a **Null Selector.**

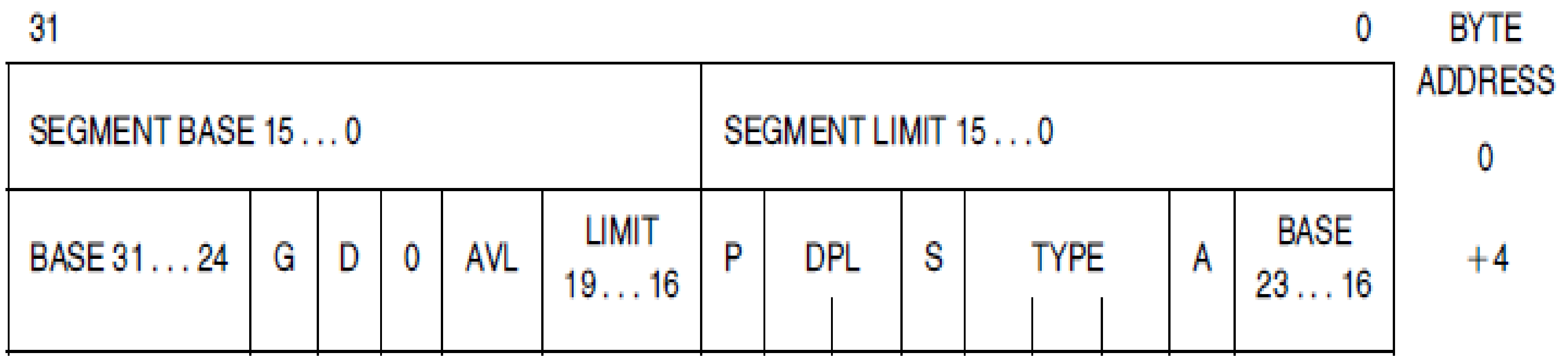
- This selector value is reserved to provide a method if initializing segment registers, since any **access using a null selector generate an exception** (General-protection Exception- INT 13).

Descriptors

- A descriptor is a series of **8 bytes that describe and locate a memory segment.**
- It contain **32 bit base address that specifies the beginning of the segment** of memory controlled by the descriptor.
- The **size of segment is indicated by a 20 bit limit field** and the state of the Granularity Bit (G bit).

- **A segment descriptor provides** the 80386 with the **data it needs to map a logical address into a linear address.**

- These descriptors are <u>not</u> created by programs, but **created by Compilers, Linkers, Loaders, or the Operating System.**

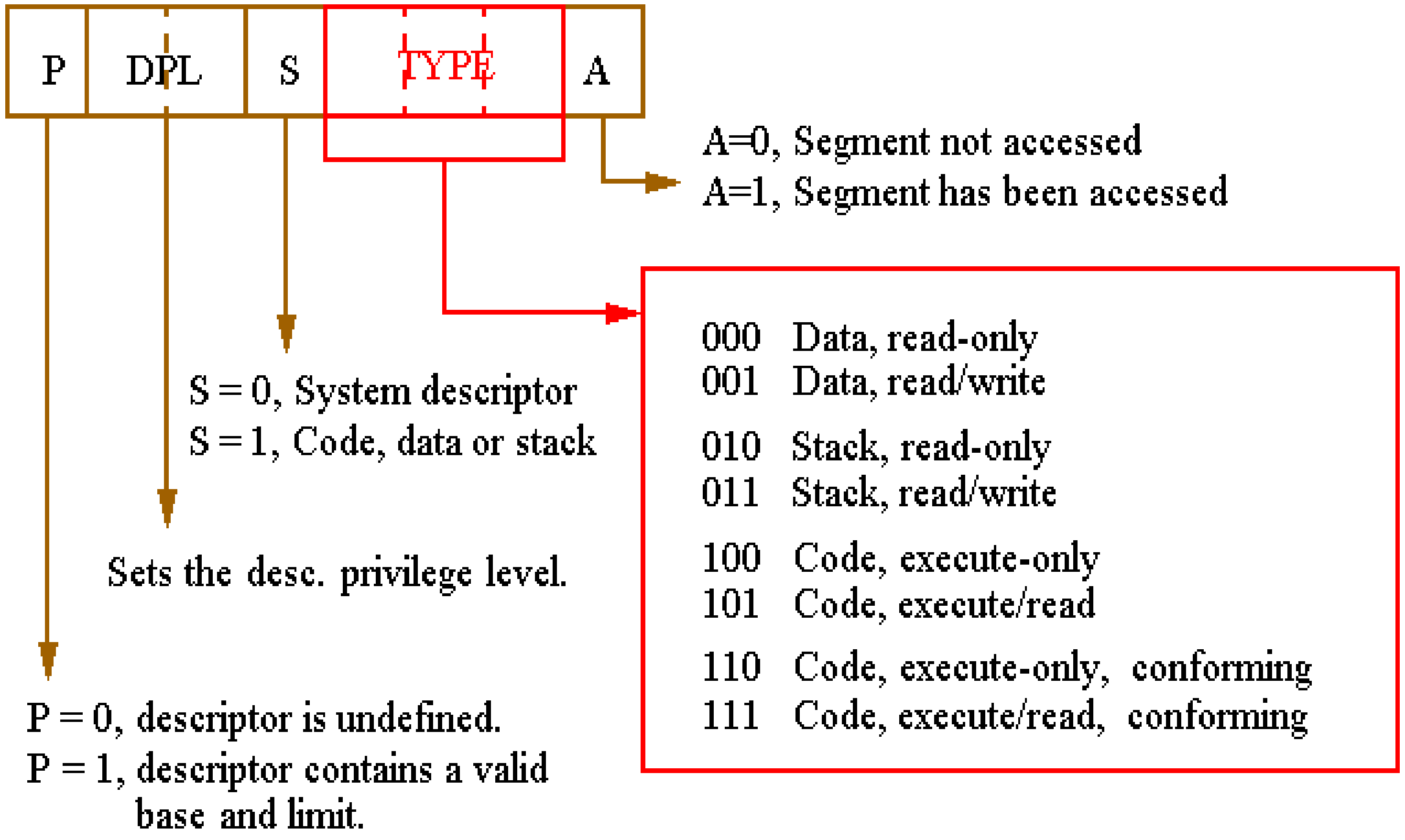


31											0	BYTE ADDRESS
SEGMENT BASE 15 . . . 0						SEGMENT LIMIT 15 . . . 0						0
BASE 31 . . . 24	G	D	0	AVL	LIMIT 19 . . . 16	P	DPL	S	TYPE	A	BASE 23 . . . 16	+4

BASE	Base Address of the segment
LIMIT	The length of the segment
P	Present Bit 1=Present 0=Not Present
DPL	Descriptor Privilege Level 0–3
S	Segment Descriptor 0=System Descriptor 1=Code or Data Segment Descriptor
TYPE	Type of Segment
A	Accessed Bit
G	Granularity Bit 1=Segment length is page granular 0=Segment length is byte granular
D	Default Operation Size (recognized in code segment descriptors only) 1=32-bit segment 0=16-bit segment
0	Bit must be zero (0) for compatibility with future processors
AVL	Available field for user or OS

NOTE:
In a maximum-size segment (ie. a segment with G=1 and segment limit 19...0=FFFFFH), the lowest 12 bits of the segment base should be zero (ie. segment base 11...000=000H).

Access Right Byte


P
DPL
S
TYPE
A
A=0, Segment not accessed
A=1, Segment has been accessed
S = 0, System descriptor
S = 1, Code, data or stack
Sets the desc. privilege level.
P = 0, descriptor is undefined.
P = 1, descriptor contains a valid base and limit.

000	Data, read-only
001	Data, read/write
010	Stack, read-only
011	Stack, read/write
100	Code, execute-only
101	Code, execute/read
110	Code, execute-only, conforming
111	Code, execute/read, conforming

- When **G (Granularity bit)** is set, the limit bit represent the number of **4kb pages** contains in the segment.

- This allows the **size of segment to be of any length from 4KB to 4GB.**

- When this bit is cleared (G = 0) the 20-bit limit field is assumed to be measured in units of 1 byte. If it is set (G = 1), the limit field is in units of 4 KB.

- Two **Descriptor Privilege Level (DPL)** bits specifies the **privilege level required to access the segment.**

- An attempt by less privilege task to use the segment result in exception.

P (Present Bit):

Indicate **whether the segment is present in memory**. A segment-not-present exception is generated if this bit is clear when the segment descriptor is accessed.

S (Segment Descriptor) :

When set, **indicate that the segment is a system segment**. When clear, the segment is a code or data segment.

D (Default Operation Size):

For <u>**code**</u> segment, **D controls the default operand and address size** (16 bit when D is clear versus 32 bit when set).

For data segment, D controls how stack is manipulated (via SP /ESP with 16/32 bit pushes/pops)

AVL : Available to programmer.

Example of D bit uses

- **for Code segment**
  - D = 0 means 16-bit 80286 code
  - D = 1 means 32-bit 80386+ code

- **for Stack Segment**
  - D = 0
    - ✓ Stack operations are 16-bit wide,
    - ✓ SP is used as a stack pointer,
    - ✓ Maximum stack size is FFFF (64 KB)
  - D = 1
    - ✓ Stack operations are 32-bit wide,
    - ✓ ESP is used as a stack pointer,
    - ✓ Maximum stack size is FFFFFFFF (4GB)

Available (AVL) bit

- The AVL (available) field specifies whether the descriptor is available for user or it is for use by operating system.

– **AVL=0**	not available for user, used by OS
– **AVL=1**	available for user

Type field

E (Executable):

Executable selects a stack segment (E=0) or a code segment (E=1) .E also defines the function of the next two bits.

X (Expansion):

If E=0,then X **indicates the direction of expansion for the data segment** . If X=0,the segment expand upward , as in a data segment.

RW (Read Write):

If E=0,then the read/write bit indicate that the data segment may be written or not.

If E=1,then RW indicate that the code segment may be read (RW=0) or not read (RW=1).

A (Accessed Bit):

Accessed is set each time that the microprocessor accesses the segment.

It is sometimes used by operating system to keep track of which segments have been accessed.

Types of Descriptors

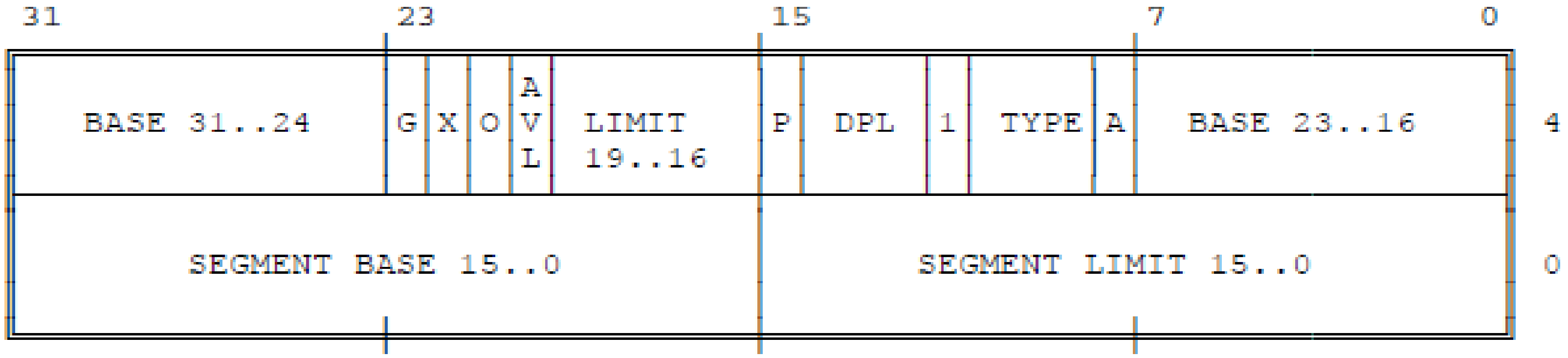
- **System:**

1. LDT
2. Task State Segment (TSS)
3. GATE
    1. Call
    2. Interrupt
    3. Task
    4. Trap

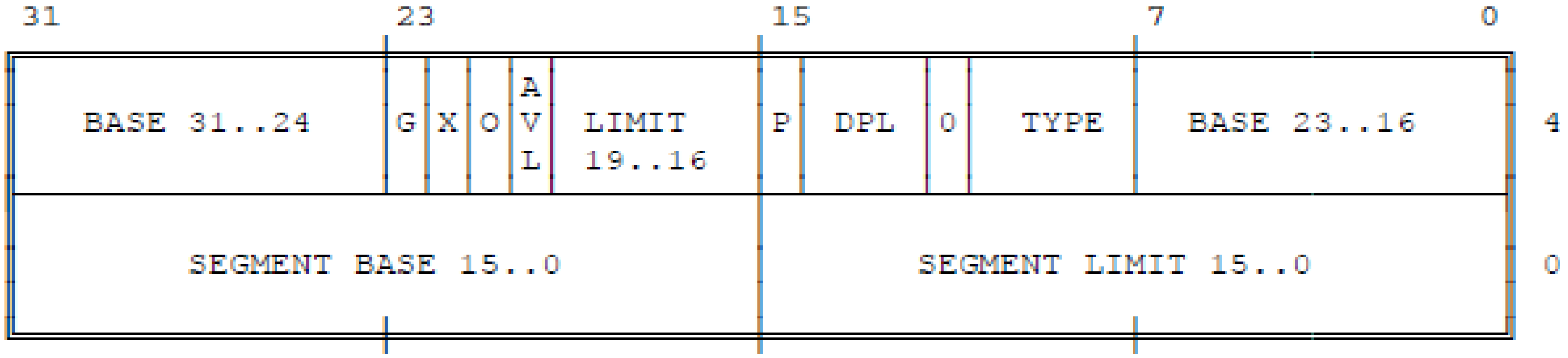
- **Non System:**

1. Data
2. Stack
3. Code

DESCRIPTORS USED FOR APPLICATIONS CODE AND DATA SEGMENTS

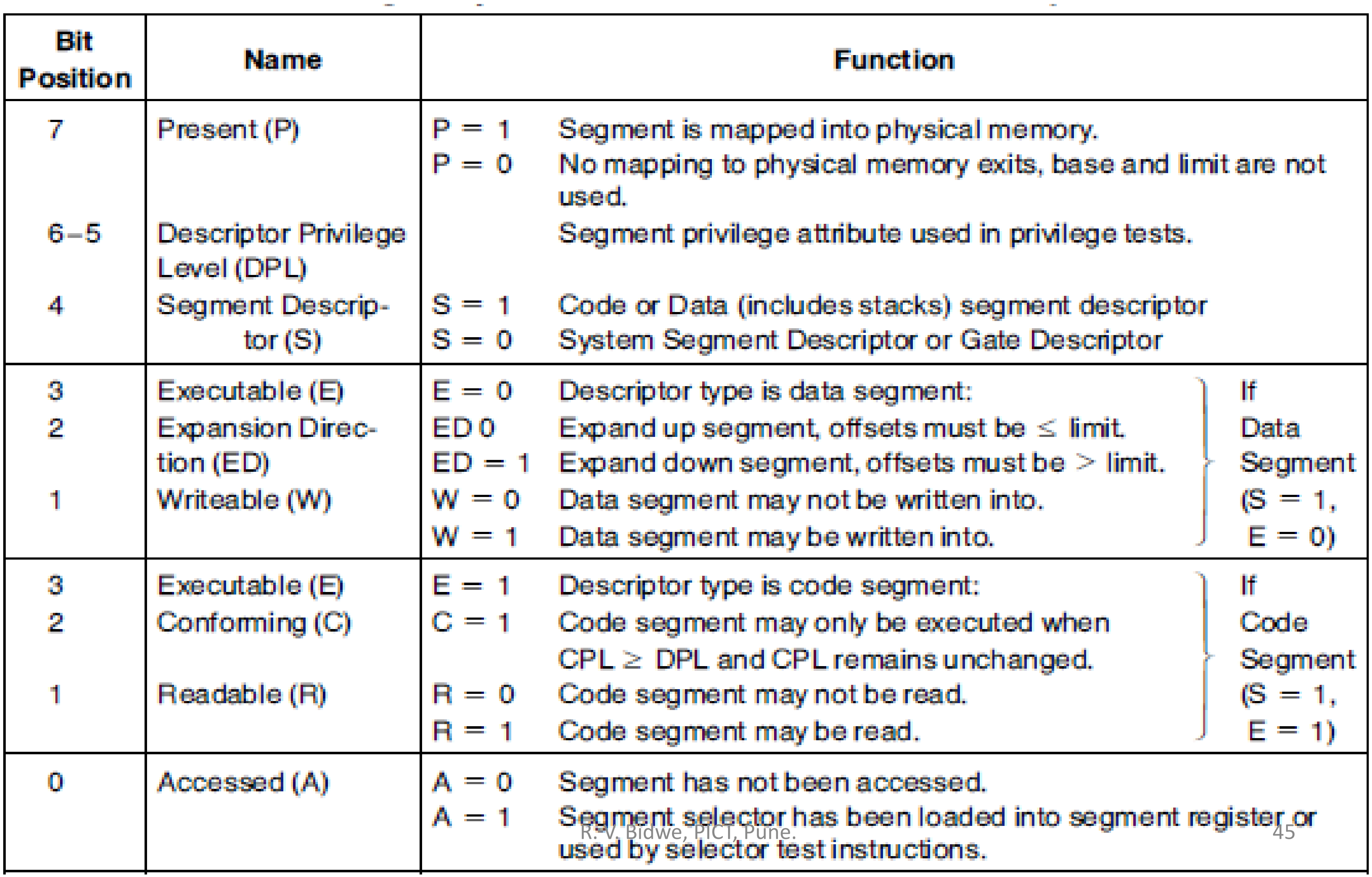


DESCRIPTORS USED FOR SPECIAL SYSTEM SEGMENTS



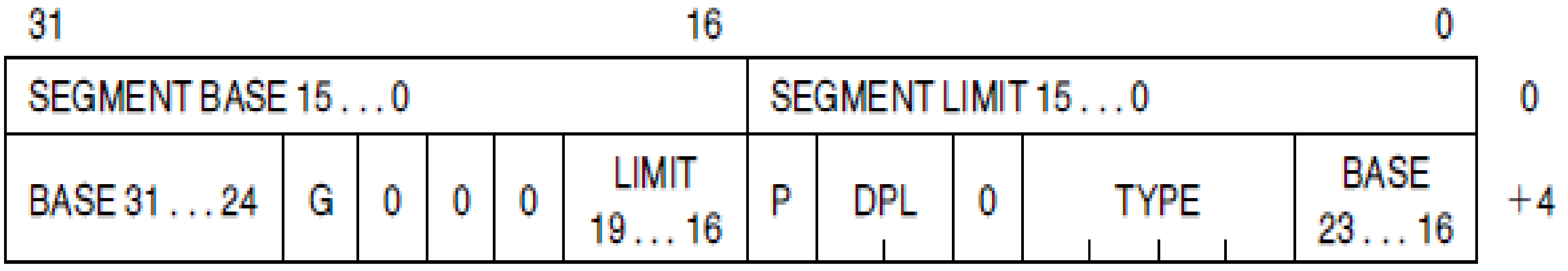
A	-	ACCESSED
AVL	-	AVAILABLE FOR USE BY SYSTEMS PROGRAMMERS
DPL	-	DESCRIPTOR PRIVILEGE LEVEL
G	-	GRANULARITY
P	-	SEGMENT PRESENT

Code and Data Descriptions



Bit Position	Name	Function		
7	Present (P)	P = 1	Segment is mapped into physical memory.	
		P = 0	No mapping to physical memory exits, base and limit are not used.	
6–5	Descriptor Privilege Level (DPL)		Segment privilege attribute used in privilege tests.	
4	Segment Descriptor (S)	S = 1	Code or Data (includes stacks) segment descriptor	
		S = 0	System Segment Descriptor or Gate Descriptor	
3	Executable (E)	E = 0	Descriptor type is data segment:	If Data Segment (S = 1, E = 0)
2	Expansion Direction (ED)	ED 0	Expand up segment, offsets must be ≤ limit.	
		ED = 1	Expand down segment, offsets must be > limit.	
1	Writeable (W)	W = 0	Data segment may not be written into.	
		W = 1	Data segment may be written into.	
3	Executable (E)	E = 1	Descriptor type is code segment:	If Code Segment (S = 1, E = 1)
2	Conforming (C)	C = 1	Code segment may only be executed when CPL ≥ DPL and CPL remains unchanged.	
1	Readable (R)	R = 0	Code segment may not be read.	
		R = 1	Code segment may be read.	
0	Accessed (A)	A = 0	Segment has not been accessed.	
		A = 1	Segment selector has been loaded into segment register or used by selector test instructions.	

System Descriptor Formats



Type	Defines	Type	Defines
0	Invalid	8	Invalid
1	Available 80286 TSS	9	Available Intel386™ DX TSS
2	LDT	A	Undefined (Intel Reserved)
3	Busy 80286 TSS	B	Busy Intel386™ DX TSS
4	80286 Call Gate	C	Intel386™ DX Call Gate
5	Task Gate (for 80286 or Intel386™ DX Task)	D	Undefined (Intel Reserved)
6	80286 Interrupt Gate	E	Intel386™ DX Interrupt Gate
7	80286 Trap Gate	F	Intel386™ DX Trap Gate

NOTE:
In a maximum-size segment (ie. a segment with G = 1 and segment limit 19...0 = FFFFFH), the lowest 12 bits of the segment base should be zero (ie. segment base 11...000 = 000H).

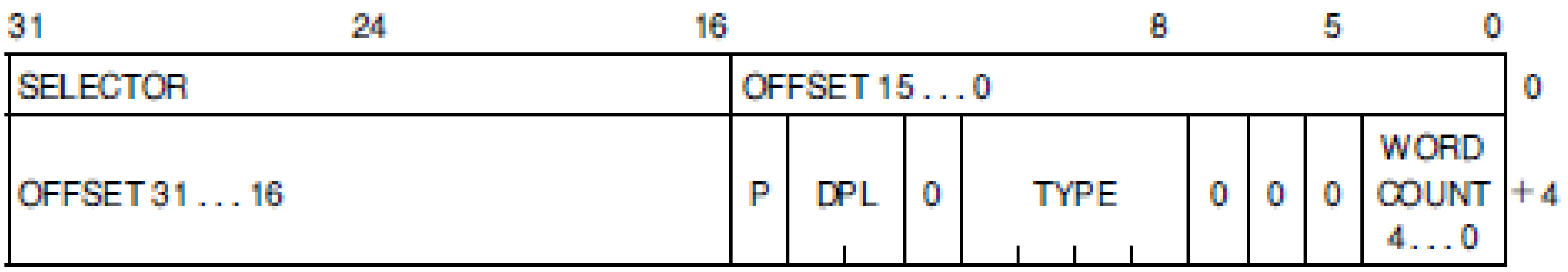
LDT Descriptor (Type 2)

- It describes about LDT.
- It is present in GDT and point to base of the LDT.
- The segment descriptors in LDT are unique to each task.

GATE Descriptor

- It is special type of descriptor used for protection checks.
- They also control the access to entry points.

- **Call Gate (C):** Used to modify the privilege level.
- **Task Gate (5):** Used in Multitasking.
- **Interrupt Gate (E):** Used to specify ISR.
- **Trap Gate (F):** Used for interrupt and exception handling.



Gate Descriptor Fields

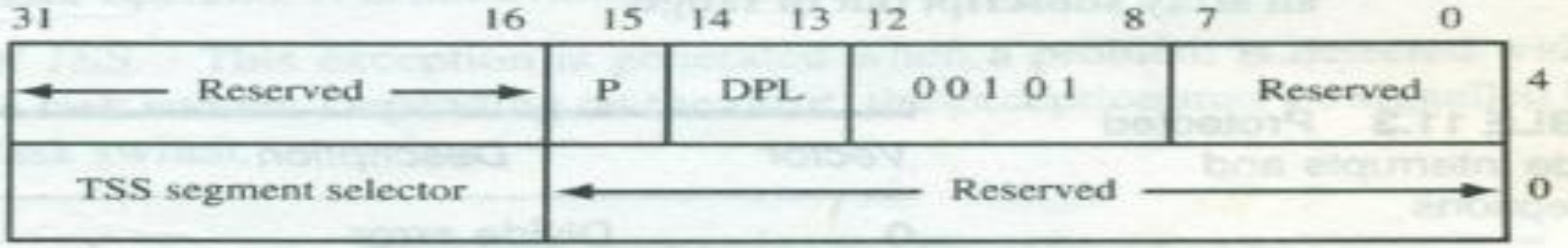
Name	**Value**	**Description**
Type	4	80286 call gate
	5	Task gate (for 80286 or Intel386™ DX task)
	6	80286 interrupt gate
	7	80286 trap gate
	C	Intel386™ DX call gate
	E	Intel386™ DX interrupt gate
	F	Intel386™ DX trap gate
P	0	Descriptor contents are not valid
	1	Descriptor contents are valid

DPL—least privileged level at which a task may access the gate. WORD COUNT 0–31—the number of parameters to copy from caller's stack to the called procedure's stack. The parameters are 32-bit quantities for Intel386™ DX gates, and 16-bit quantities for 80286 gates.

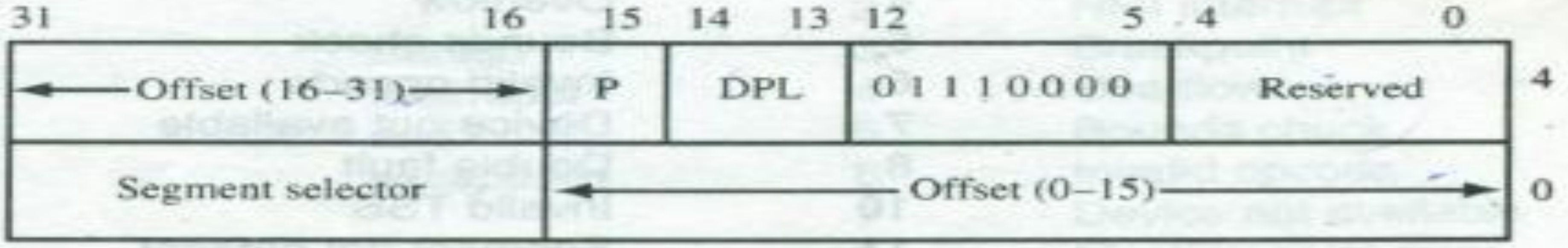
DESTINATION SELECTOR	16-bit selector	Selector to the target code segment or Selector to the target task state segment for task gate
DESTINATION OFFSET	offset 16-bit 80286 32-bit Intel386™ DX	Entry point within the target code segment

- A **Word Count** which **specifies how many parameters are to be copied from the caller's stack to the stack of the called routine**.

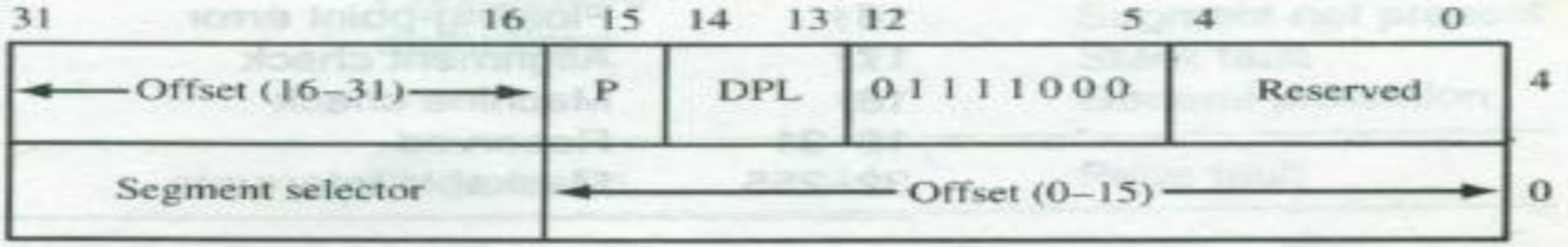
- The **Word Count** field is **only used by call gates** when there is a change in the privilege level, <u>other types of gates ignore the word count field.</u>



(a) Task gate



(b) Interrupt gate



(c) Trap gate

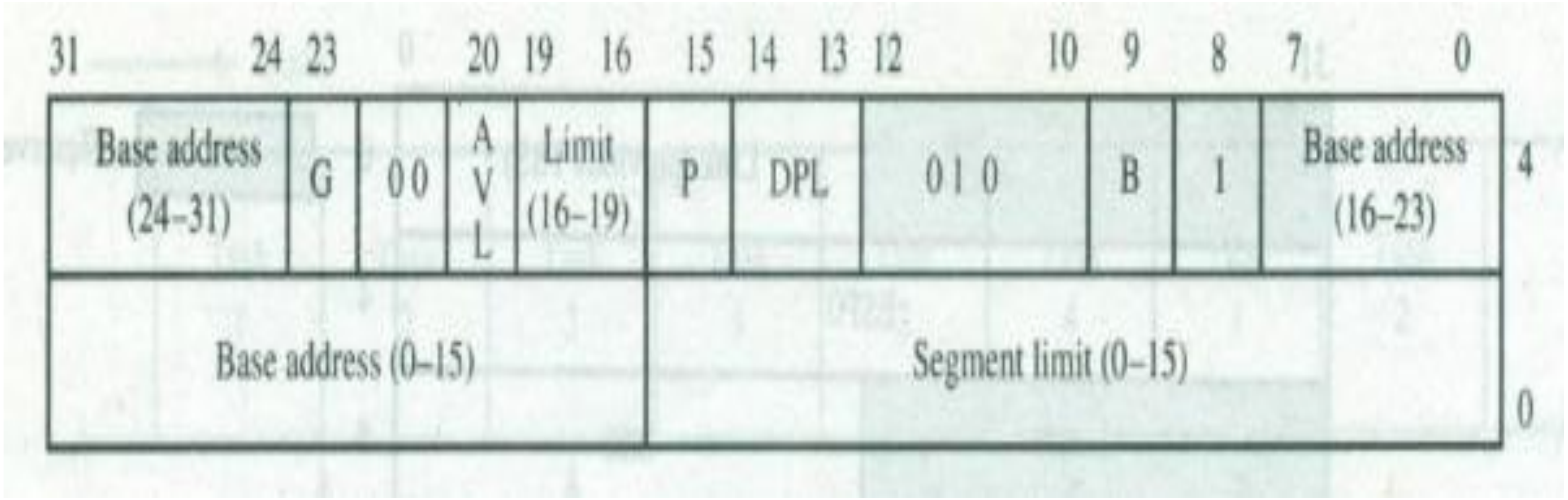
- **Interrupt and Trap Gates** use the destination selector and destination offset fields of the gate descriptor as a pointer to the start of the interrupt or trap handler routines.

- The difference between interrupt gates and trap gates is that the **interrupt gate disables interrupts (resets the IF bit) while the trap gate does not.**

Task State Segment (TSS) Descriptor

- In multitasking the task segment is addressed with the help of TSS descriptor.
- It contain the information of location, size and Privilege level of TSS.
- It appears only in GDT

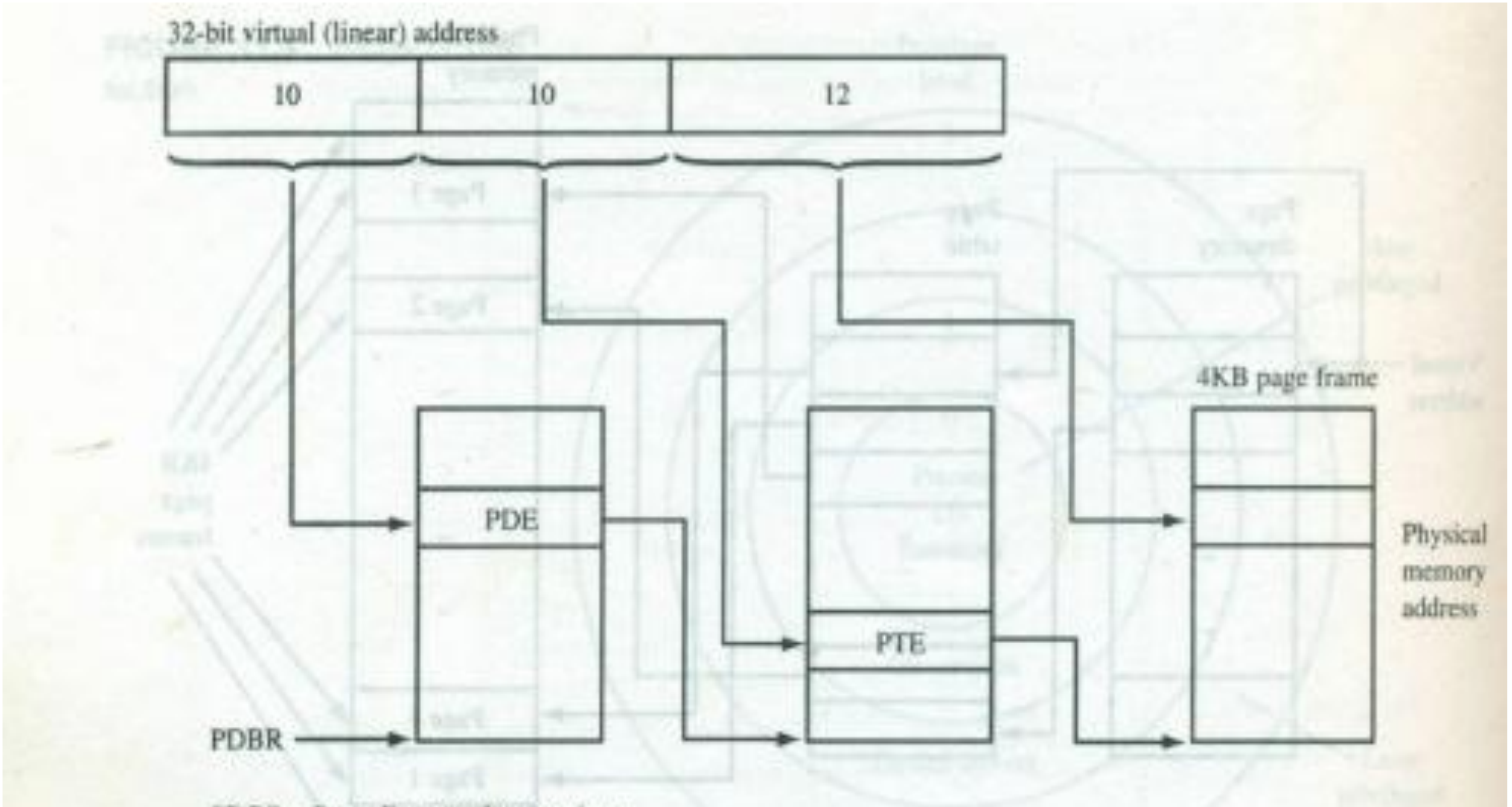
TSS Descriptor



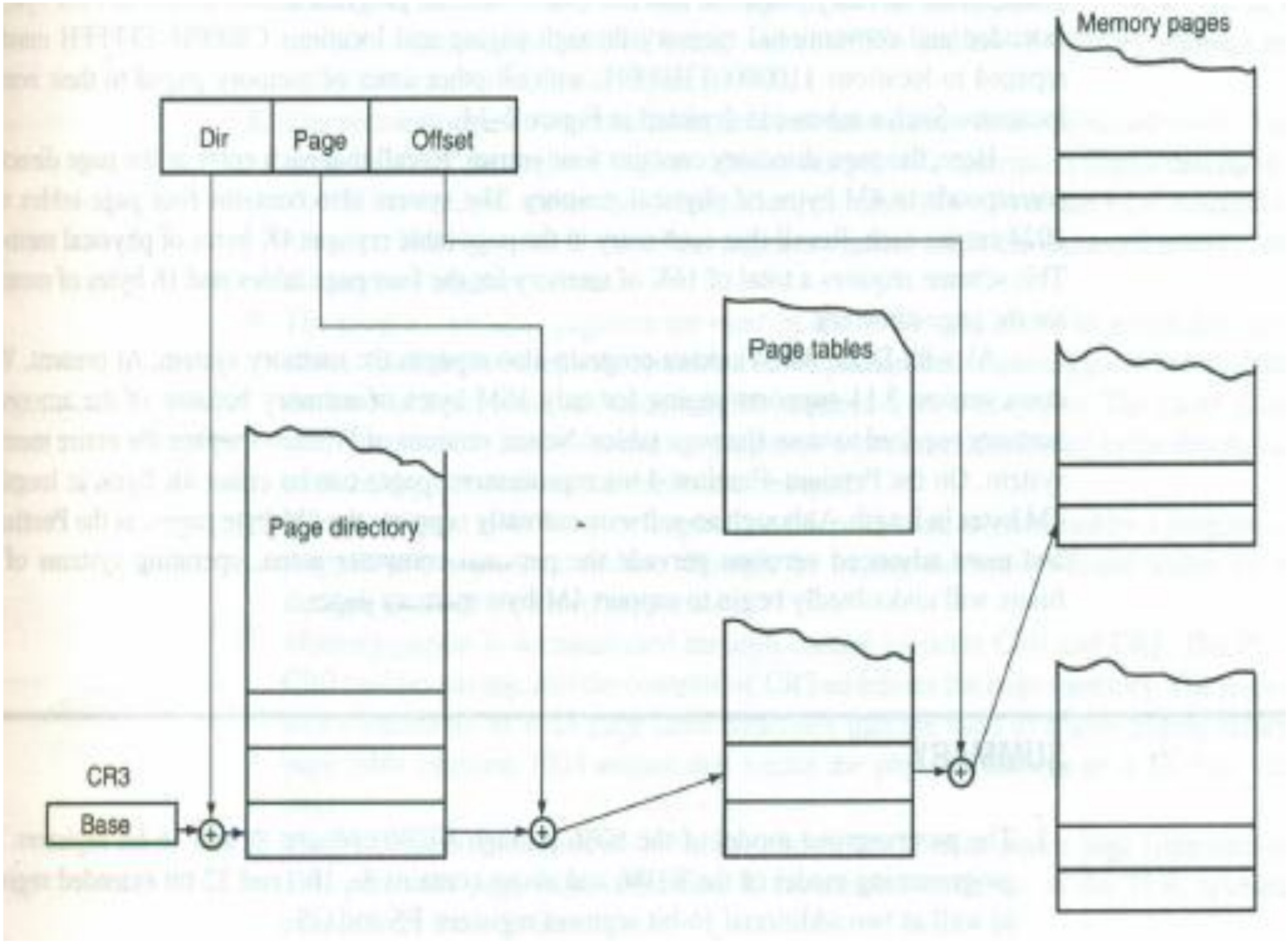
The **B bit controls the size of the stack pointer register**. If B=1, ESP will be used to point stack. And If B=0, SP will be used to point stack.

Page Translation

Paging

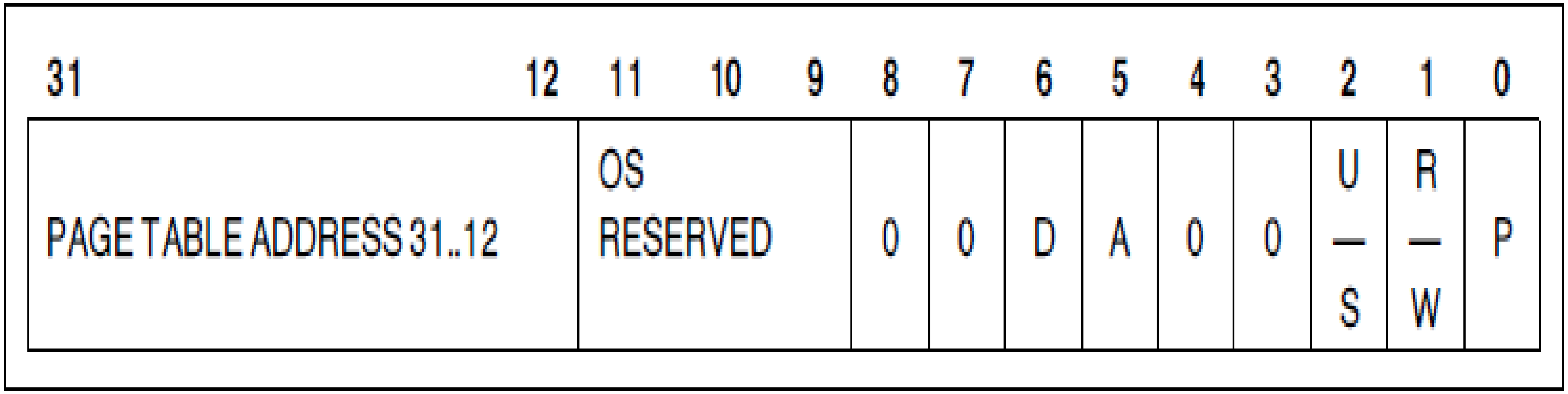


PDBR – Page directory base register
PDE – Page directory entry
PTE – Page table entry


Dir
Page
Offset
Memory pages
Page tables
Page directory
CR3
Base

PDE (Page Directory Entry)

- The page directory have **1024 directory entries** of 4 bytes each.
- Each page directory entry addresses a page table that contains 1024 entries.



31 … 12	11 10 9	8	7	6	5	4	3	2	1	0
PAGE TABLE ADDRESS 31..12	OS RESERVED	0	0	D	A	0	0	U — S	R — W	P

Figure 4-20. Page Directory Entry (Points to Page Table)

- Each Page Directory Entry contains the

    1. Address of the next level of tables

    2. Page Tables and information about the page table.

- The **upper 10 bits of the linear address** (A22±A31) **are used as an index** to select the correct Page Directory Entry.

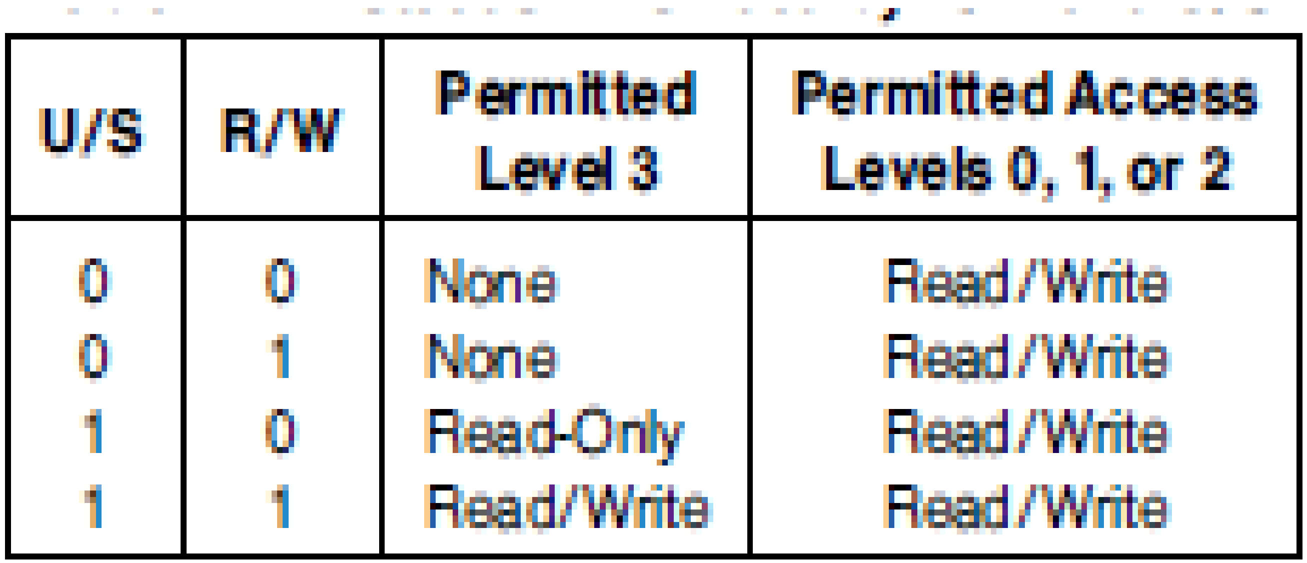
- **D** : The D (Dirty) bit 6 is set to 1 before a write to an address covered by that page table entry occurs.

- **A** : The A (Accessed) bit 5, is set by the Intel386 DX for both types of entries before a read or write access occurs to an address covered by the entry.

- **U/S and R/W** : These bits are used to provide User/ Supervisor and Read/Write protection for individual pages.

- **P** : The P (Present) bit 0 indicates if a Page Directory or Page Table entry can be used in address translation.
  - If P = 1 the entry can be used for address translation.

- **User which corresponds to level 3** of the segmentation based protection, and **supervisor which encompasses all of the other protection levels (0, 1, 2).**



U/S	R/W	Permitted Level 3	Permitted Access Levels 0, 1, or 2
0	0	None	Read/Write
0	1	None	Read/Write
1	0	Read-Only	Read/Write
1	1	Read/Write	Read/Write

PTE (Page Table Entry)

- Each Page Table is 4K bytes and holds up to 1024 Page Table Entries.
- Page Table Entries have the starting address of the page frame and statistical information about the page.

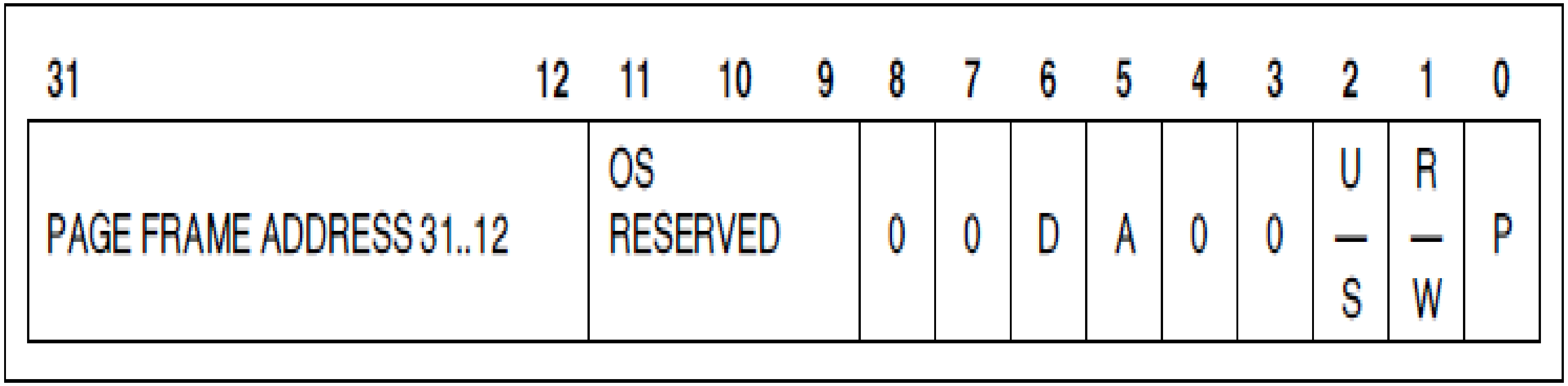


Figure 4-21. Page Table Entry (Points to Page)

- **The 20 upper bit page frame address is concatenated with the lower 12 bits of the linear address to form the physical address.**

- Page tables can be shared between tasks and swapped to disks.

Translation Look-aside Buffer

- The Intel386 DX keeps a cache of the most recently accessed pages, this cache is called the **Translation Look-aside Buffer (TLB).**

- The 32-entry TLB coupled with a 4K page size, results in **coverage of 128K bytes of memory addresses.**

TLB

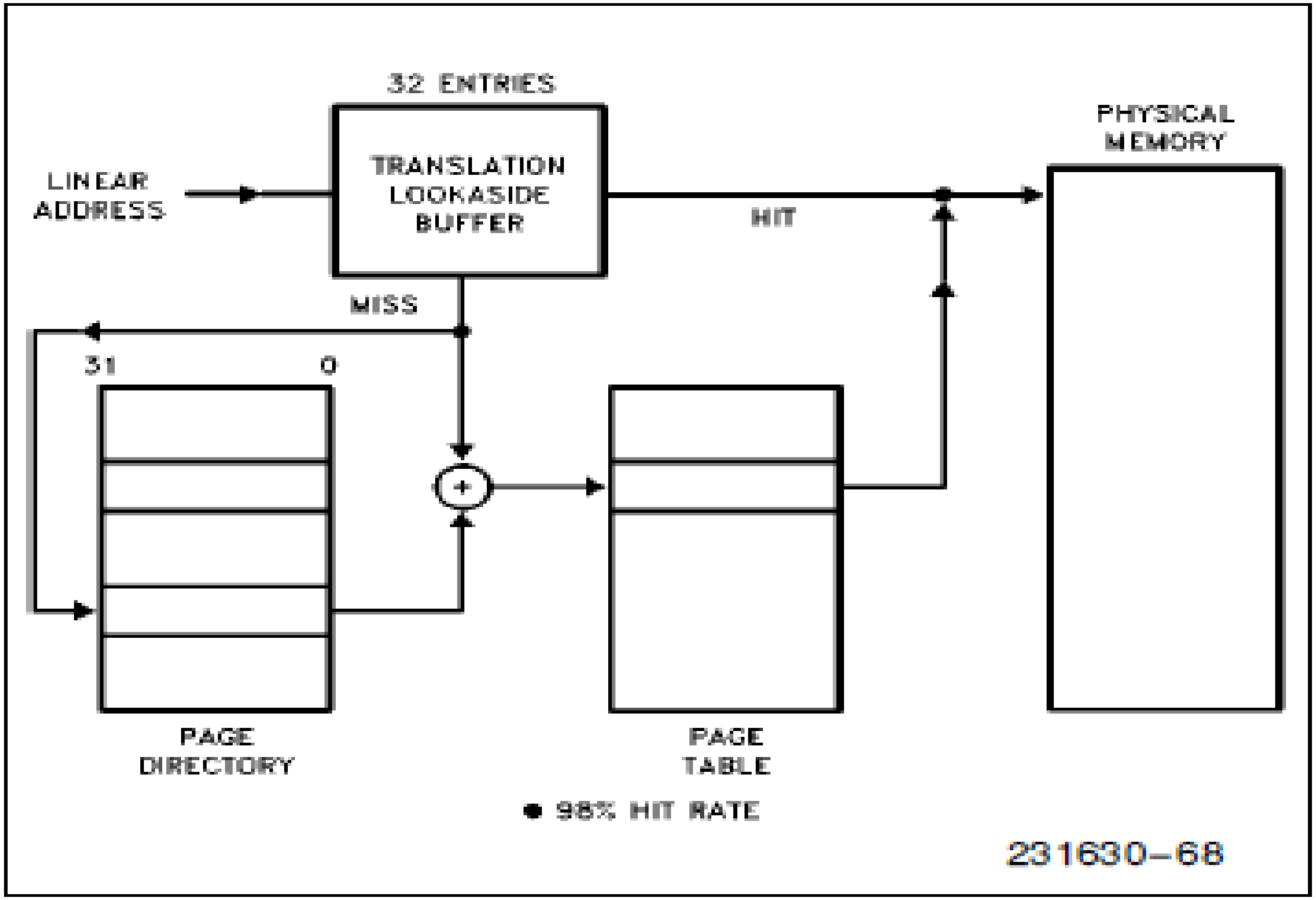


Figure 4-22. Translation Lookaside Buffer

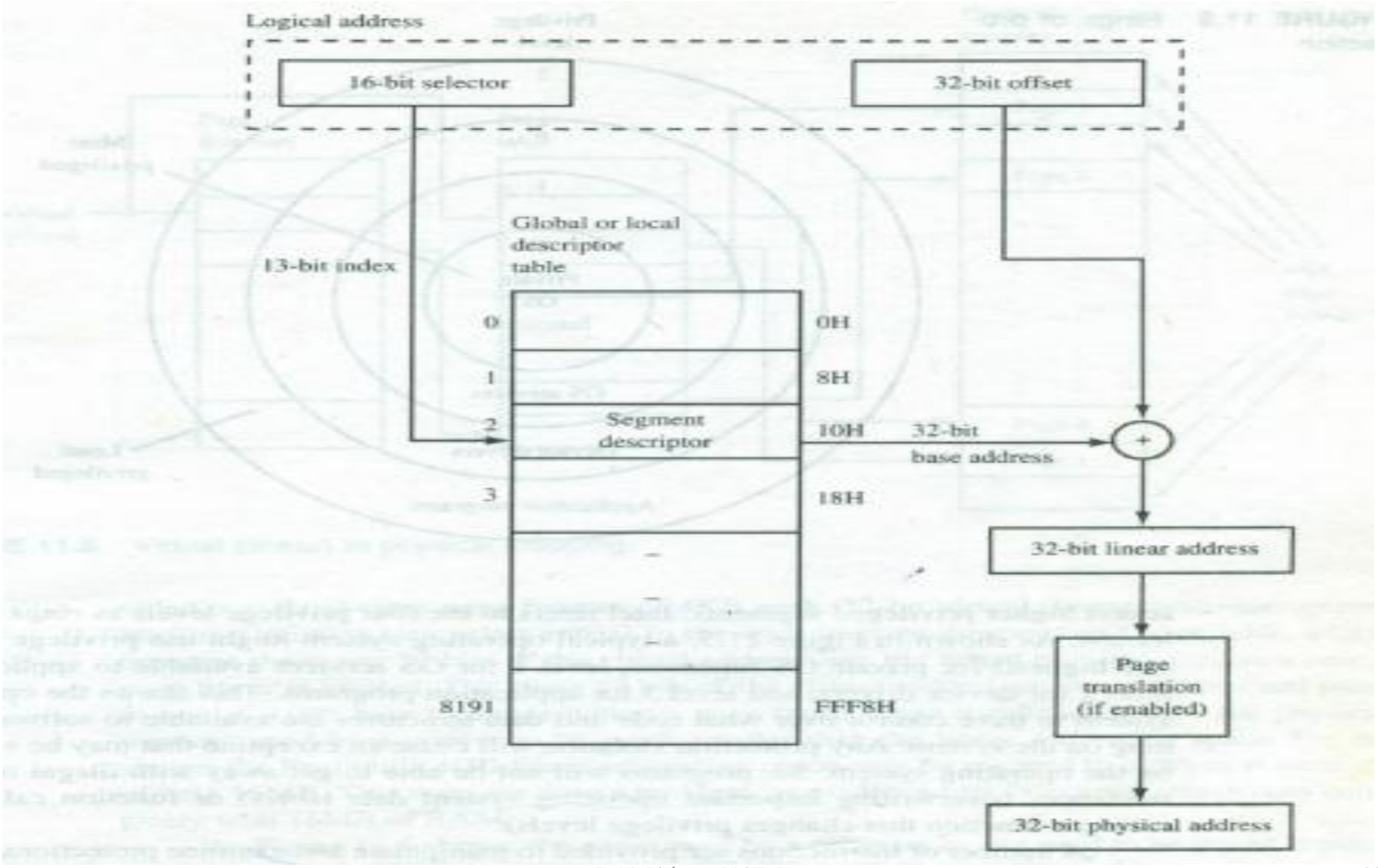
- The paging unit hardware receives a 32-bit linear address from the segmentation unit.

- The **upper 20 linear address bits are compared with all 32 entries in the TLB to determine if there is a match.**

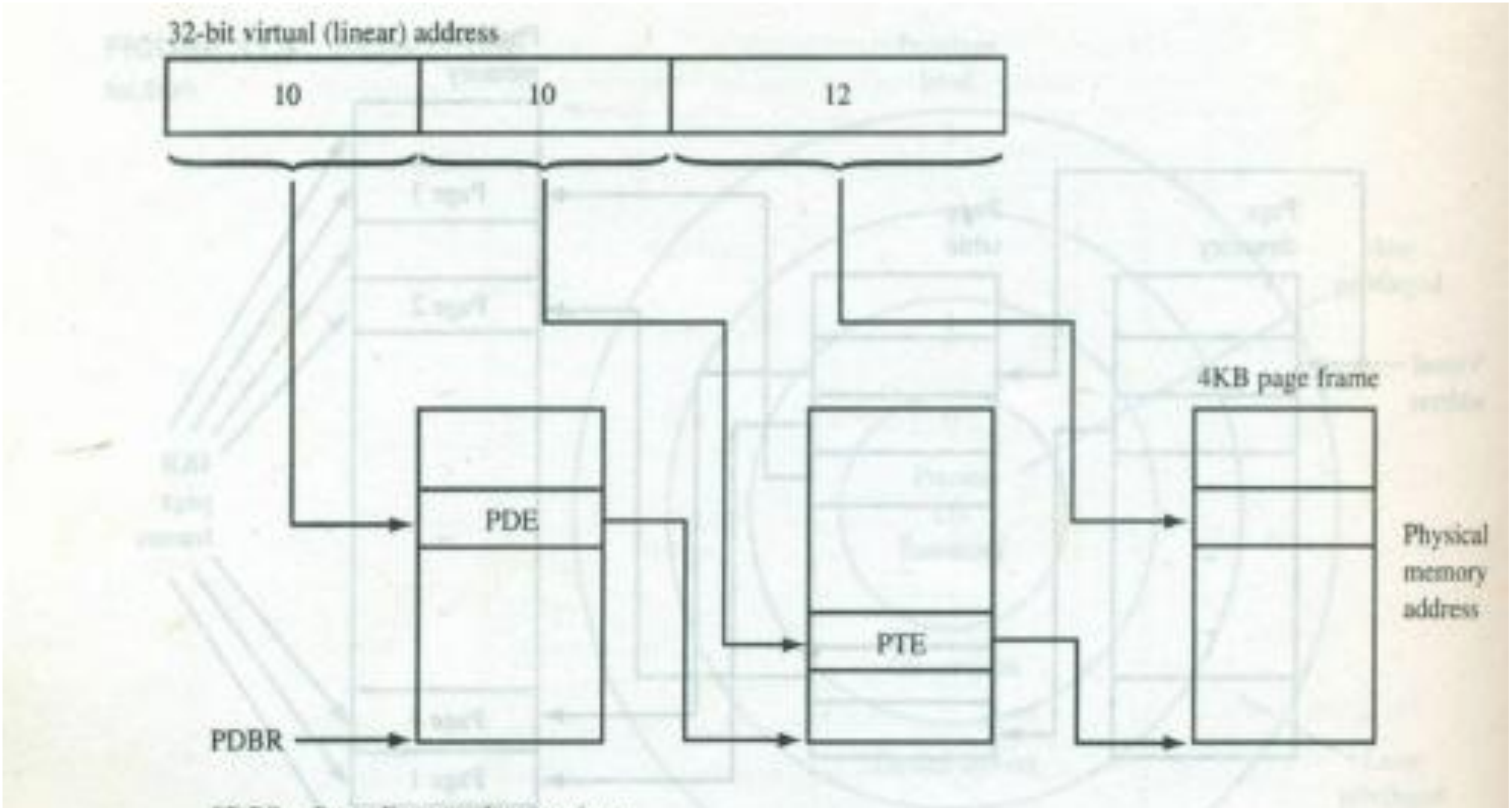
- **If there is a match (i.e. a TLB hit),** then the 32-bit physical address is calculated and will be placed on the address bus.

Combining Segment and Page Translation

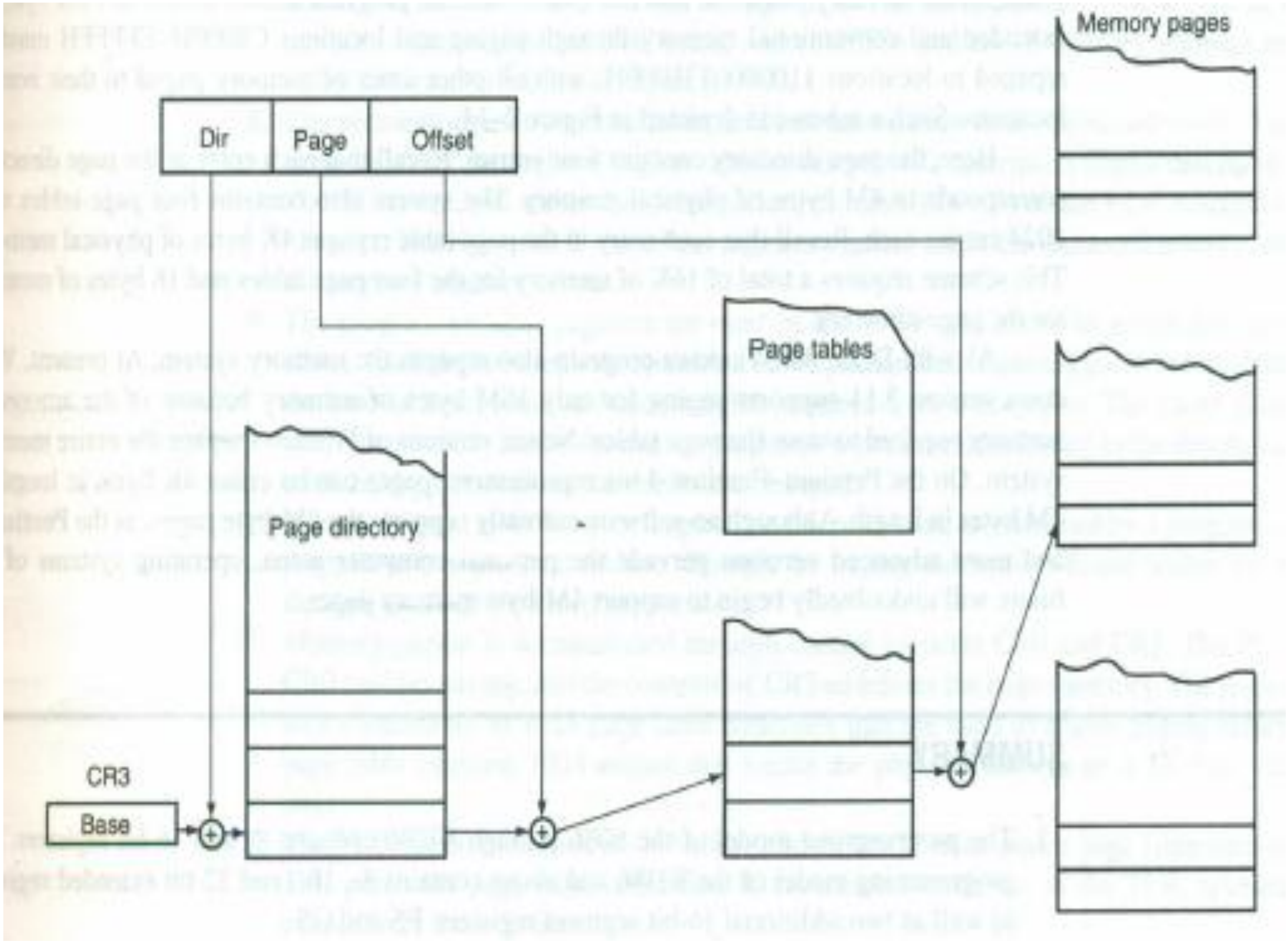
Segmentation in 80386DX



Paging



PDBR – Page directory base register
PDE – Page directory entry
PTE – Page table entry


Dir
Page
Offset
Memory pages
Page tables
Page directory
CR3
Base

Virtual Memory in 80386

- In a system, at any point minimum **16,384 descriptors (8192 of GDT + 8192 of LDT)** will be present.
- Each descriptor from system can **address a memory of minimum 1 Byte and Maximum 4 GB.**
- So the virtual memory supported by the system will be 64 TB. **(16,384 descriptors * 4 GB= 64 TB)**

Demand Paging

- **Demand Paging** follows that pages should only be brought into memory if the executing process demands them.

- This is often referred to as **Lazy Evaluation** as only those pages demanded by the process are swapped from secondary storage to main memory.

- The new terms we will be learning in this section is:
  - **Demand Paging**
  - **Swapping**
  - **Virtual Memory / Virtualization**

Steps involved in Demand Paging

1. Determining Memory Requirement
2. Allocating Memory
3. Saving the Contents of Reallocated Memory
4. Remapping a Page Memory
5. Restoring Reallocated Pages

1. Determining Memory Requirement

- Theoretically, a program gives best performance when it is completely loaded into primary memory, but practically it is not possible.
- Majority of memory is used by OS, Memory resident Programs and other user’s programs, or memory may not be available.
- So as per requirement of new task, memory will be made free by swapping out data to secondary memory.

2. Allocating Memory

- System will check **A bit (Bit 5) from PTE** of every page, and accordingly pages will be allocated.
- If accessed bit has not be set by the system, meaning is that 4 KB page frame is never references by the processor.
- If required, System will use **LRU algorithm.**

3. Saving the Contents of Reallocated Memory

- Contents from page frame will be copied to secondary memory.
- During copying, **D bit (bit 6) from PTE** will be checked.

4. Remapping a Page Memory

- After completing copy from the page, **P bit from PTE will be set again**, as new valid entries are added to page of new task.
- Also entries from TLB will be updated accordingly.

5. Restoring Reallocated Pages

- Data may be restored to pages as per system requirement.

Demand Paging and Virtual Memory

1. Creates the illusion of nearly infinite memory.
2. Achieved by “Swapping” pages by physical memory.
3. Processor requests page marked not present.
4. Page is chosen for reuse.
5. Contents of page are swapped out.
6. Page is readdressed.
7. Program is restarted.

System Instructions

1. **Verification of pointer parameters :**

   i. ARPL — Adjust RPL
   ii. LAR — Load Access Rights
   iii. LSL — Load Segment Limit
   iv. VERR — Verify for Reading
   v. VERW — Verify for Writing

2. Addressing descriptor tables :

i. LLDT — Load LDT Register
ii. SLDT — Store LDT Register
iii. LGDT — Load GDT Register
iv. SGDT — Store GDT Register

3. Multitasking:

i. LTR — Load Task Register
ii. STR — Store Task Register

4. Coprocessing and Multiprocessing):

i. CLTS — Clear Task-Switched Flag
ii. ESC — Escape instructions
iii. WAIT — Wait until Coprocessor not

5. Input and Output:

i. IN — Input
ii. OUT — Output
iii. INS — Input String
iv. OUTS — Output String

6. Interrupt control:

i. CLI — Clear Interrupt-Enable Flag
ii. STI — Set Interrupt-Enable Flag
iii. LIDT — Load IDT Register
iv. SIDT — Store IDT Register

7. Debugging :

i. MOV — Move to and from debug registers

8. TLB testing:

i. MOV — Move to and from test registers

9. System Control:

i. SMSW — Set MSW

ii. LMSW — Load MSW

iii. HLT — Halt Processor

iv. MOV — Move to and from control registers