

80286

Microprocessor

Department of Computer Science and Engineering
BRAC University

Course ID: CSE - 34I
Course Title: Microprocessors

80186

- ? 80186 is a 16 bit microprocessor with 16 bit data bus and 20 bit address bus similar to 8086.
- ? 80186 has several additional functional chips:
 - ? clock generator
 - ? 2 independent DMA channels
 - ? PIC
 - ? 3 programmable 16-bit timers
- ? more a microcontroller than a microprocessor
- ? used mostly in industrial control applications

80286



Comparison between 8086 and 80286

8086	80286
1. 16 bit microprocessor	1. 16 bit microprocessor
2. Data bus is 16 bit (D0 to D15)	2. Data bus is 16 bit (D0 to D15)
3. Address bus is 20 bit (A0 to A19)	3. Address bus is 24 bit (A0 to A23)
4. Works in two modes (Minimum and Maximum)	4. Works in two modes (Real and Virtual protected Mode)
5. Has Multiplexed buses so the number of pins are 40.	5. Has non-multiplexed buses so the number of pins are 68.
6. Operates between 5 to 10 MHz having standard frequency of 6MHz. So slower than 80286	6. Operates in 8 MHz, 10 MHz & 12.5 MHz . So faster than 8086
7. Does not support memory management and protection	7. High performance microprocessor with memory management and protection

Intel 80286 has 2 operating modes:

Real Address Mode :

- ❑ 80286 is just a fast 8086 --- up to 6 times faster
- ❑ All memory management and protection mechanisms are disabled
- ❑ 286 is object code compatible with 8086

Protected Virtual Address Mode

- ❑ 80286 works with all of its memory management and protection capabilities with the advanced instruction set.
- ❑ it is source code compatible with 8086

Real Address Mode (Real Mode)

- In this mode, the **80286 behaves like a very fast 8086** (up to 6× faster).
- **Addressing is restricted to 20 address lines** (just like 8086).
- Even though the chip physically has **24 address lines** (capable of addressing 16 MB), only **1 MB** of memory is accessible in this mode.
- **Why?** Because Intel wanted backward compatibility with 8086 programs, which expected only 1 MB of address space.
- So, in **real mode**, the extra 4 address lines (A20–A23) are not used.
- The 286 is **object code compatible** with 8086 → meaning old machine code (compiled binaries) runs directly.

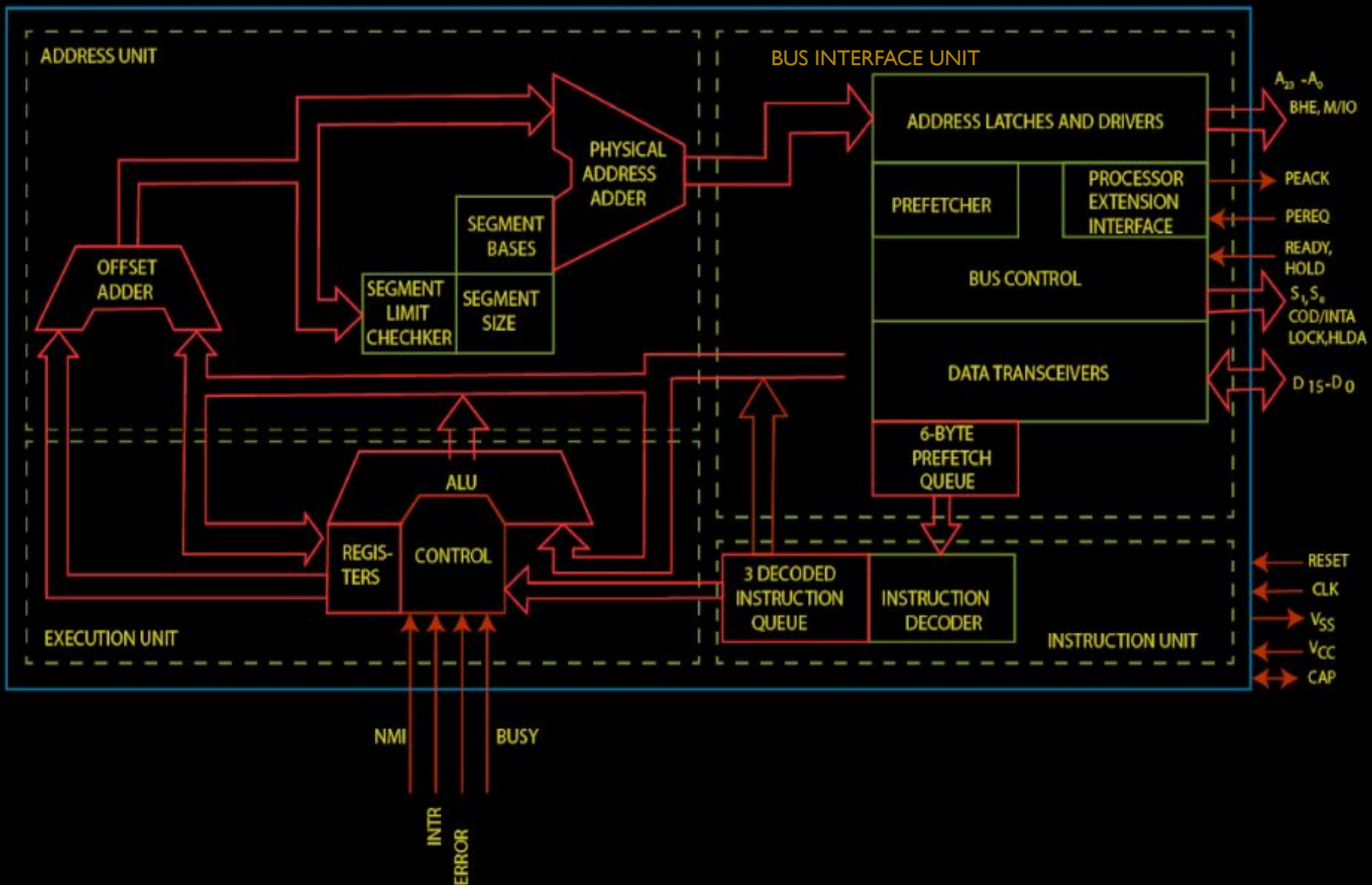
Protected Virtual Address Mode (Protected Mode)

- Here, the **286** unlocks its full hardware capabilities.
 - All **24 address lines** are active → can address **16 MB of physical memory**.
 - Supports **1 GB of virtual memory** through segmentation.
 - Adds **memory protection features** → programs can't overwrite each other's memory.
 - Uses **descriptors and tables (GDT, LDT)** for segment management instead of just CS, DS, ES, SS registers.
 - The 286 is **source code compatible** with 8086 → meaning old assembly programs can be recompiled and run in this mode.
-

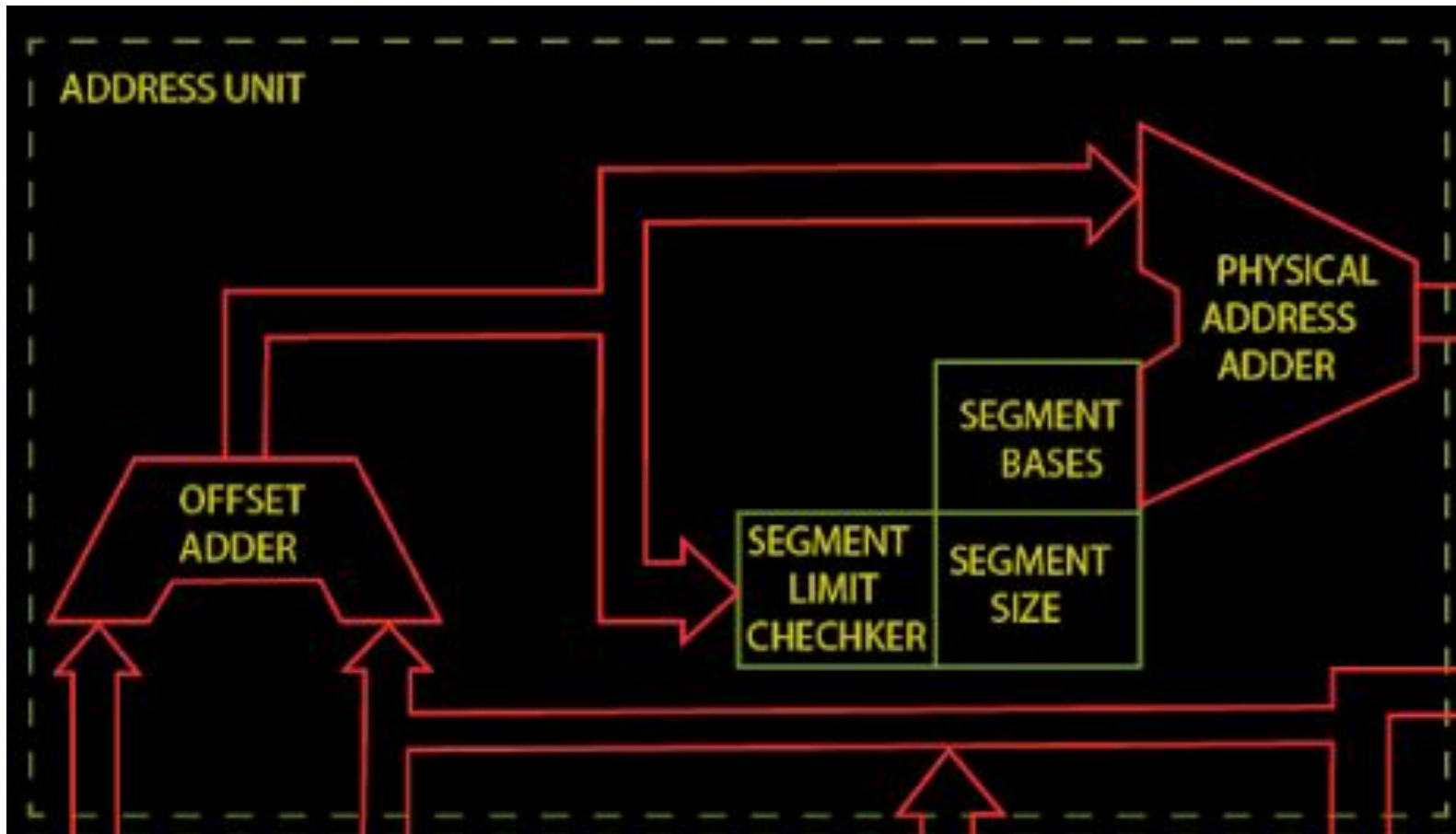


80286

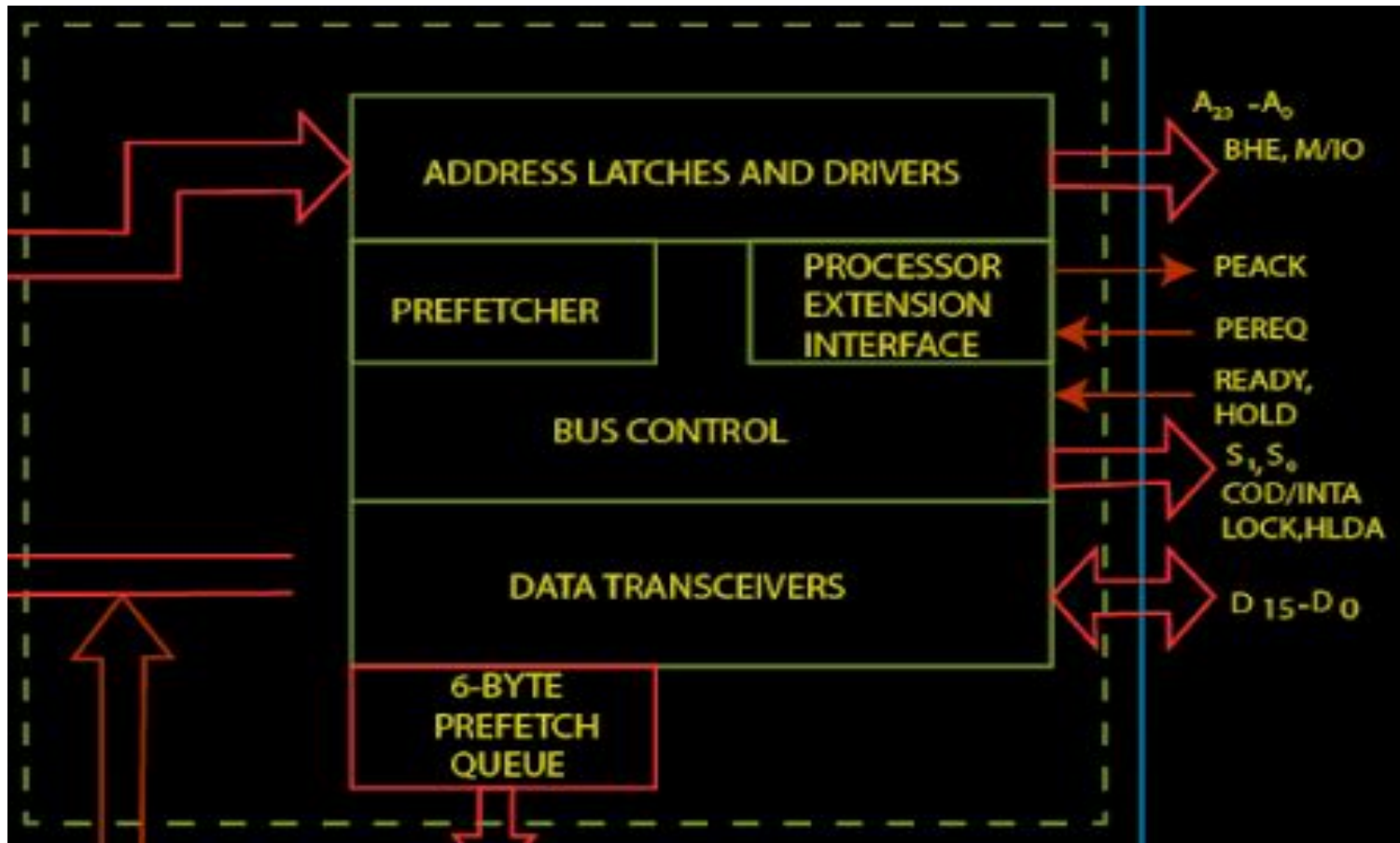
INTERNAL ARCHITECTURE



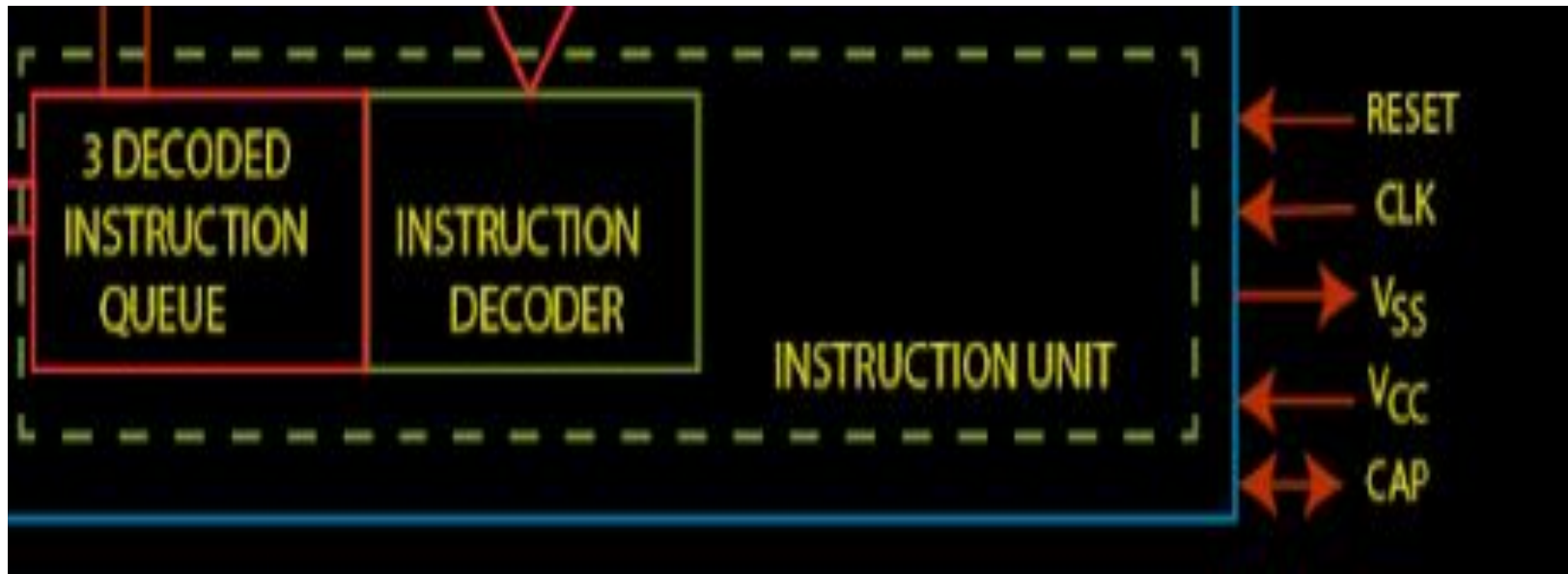
Address unit



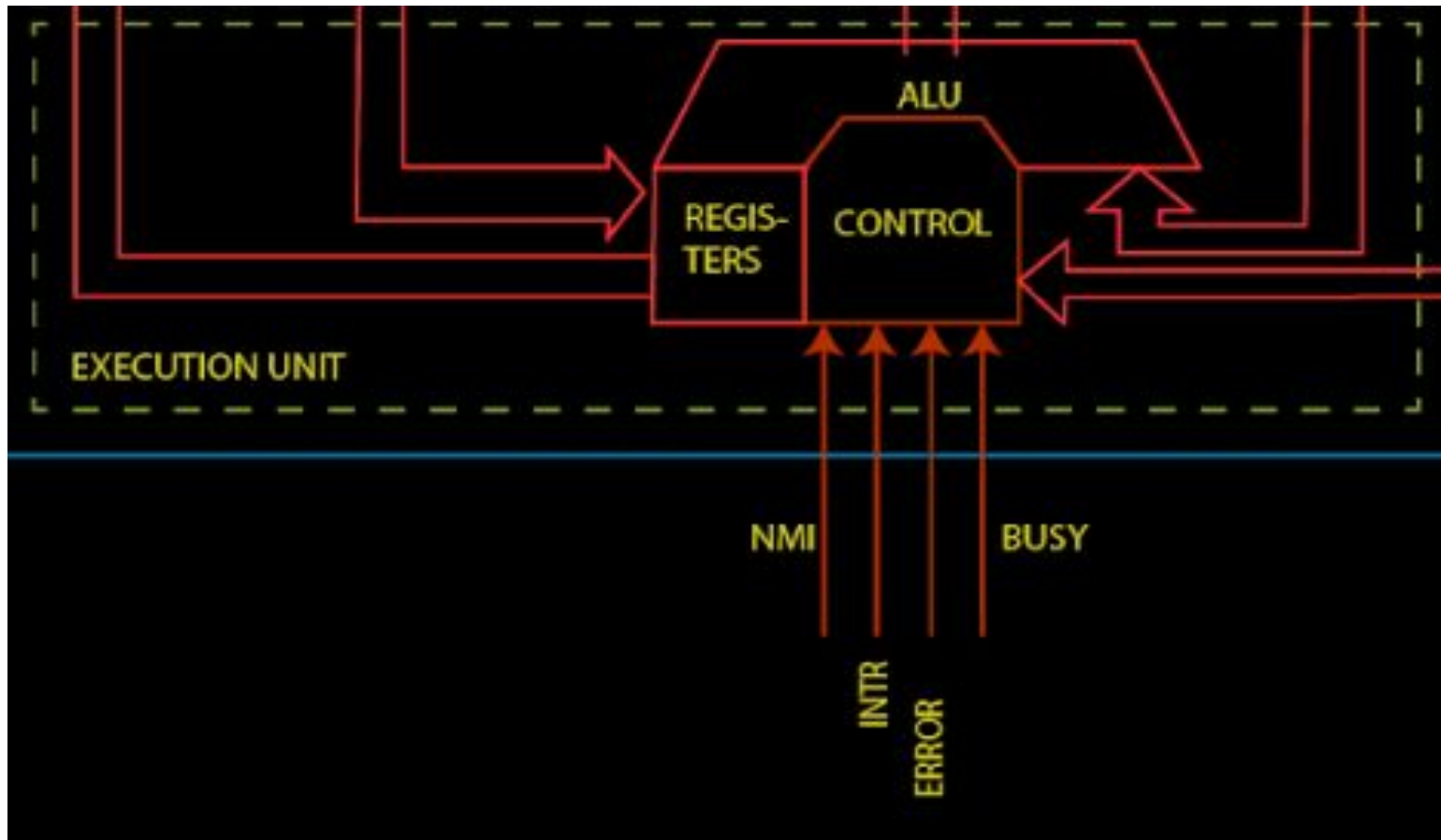
Bus Interface unit



Instruction unit

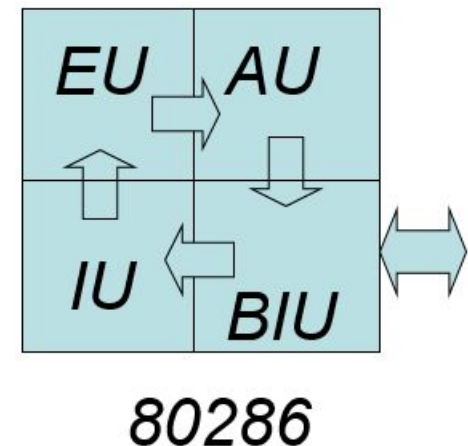


Execution unit



Functional Parts

1. Address unit
2. Bus Interface unit
3. Instruction unit
4. Execution unit



Address Unit

- Calculate the physical addresses of the instruction and data that the CPU want to access
- Address lines derived by this unit may be used to address different peripherals.
- Physical address computed by the address unit is handed over to the **Bus Unit**.

Bus Interface Unit

- ❑ Performs all memory and I/O read and write operations.
- ❑ Take care of communication between CPU and a coprocessor.
- ❑ Transmit the physical address over address bus $A_0 - A_{23}$.
- ❑ Prefetcher module in the bus unit performs this task of prefetching.
- ❑ **Bus controller** controls the prefetcher module.
- ❑ Fetched instructions are arranged in a **6 – byte prefetch queue**.

Instruction Unit

- Receive arranged instructions from 6 byte prefetch queue.
- Instruction decoder decodes up to 3 prefetched instruction and are latched them onto a decoded instruction queue.
- Output of the decoding circuit drives a control circuit in the Execution unit.

Execution unit

- **EU** executes the instructions received from the decoded instruction queue sequentially.
- Contains Register Bank.
- contains one additional special register called **Machine status word (MSW)** register --- lower 4 bits are only used.
- ALU is the heart of execution unit.
- After execution ALU sends the result either over data bus or back to the register bank.

Register organization of 80286

- The 80286 CPU contains the **same set of registers, as in 8086** .

- Eight 16-bit general purpose registers.
- Four 16 bit segment registers.
- One Flag register.
- One Instruction pointer.

plus

- one new 16-bit machine status word (MSW) register

16-BIT
REGISTER
NAME

Special
Register
Functions

BYTE
ADDRESSABLE
(16-BIT
REGISTER
NAMES
SHOWN)

7	07	0
AX	AH	AL
DX	DH	DL
CX	CH	CL
BX	BH	BL
BP		
SI		
DI		
SP		

MULTIPLY/DIVIDE
I/O INSTRUCTION

LOOP/SHIFT/REPEAT COUNT

BASE REGISTERS

INDEX REGISTERS

STACK POINTER

15 0
GENERAL
REGISTERS

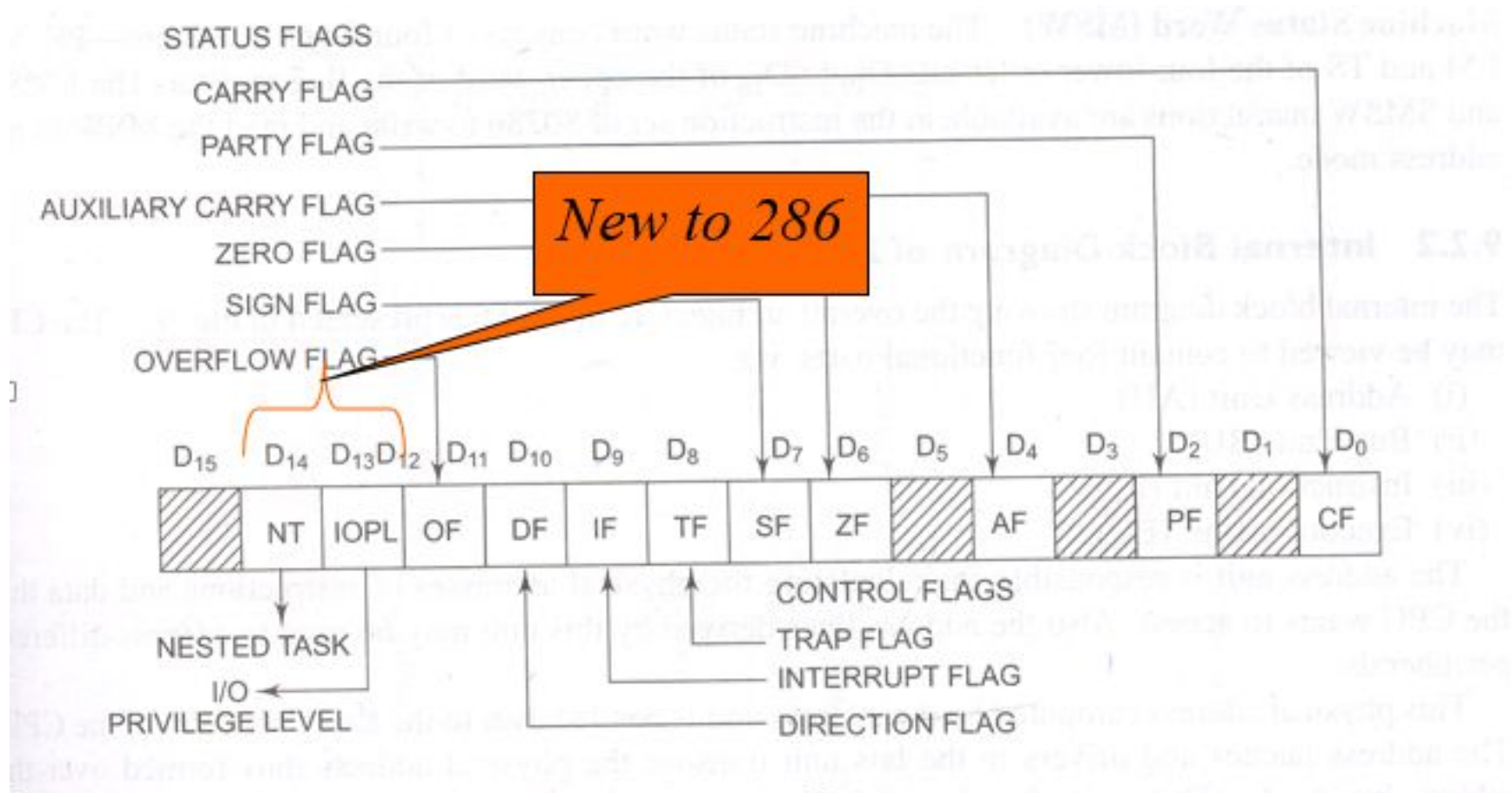
15	0	CS	CODE SEGMENT SELECTION
		DS	DATA SEGMENT SELECTION
		SS	STACK SEGMENT SELECTION
		ES	EXTRA SEGMENT SELECTION

SEGMENT REGISTERS

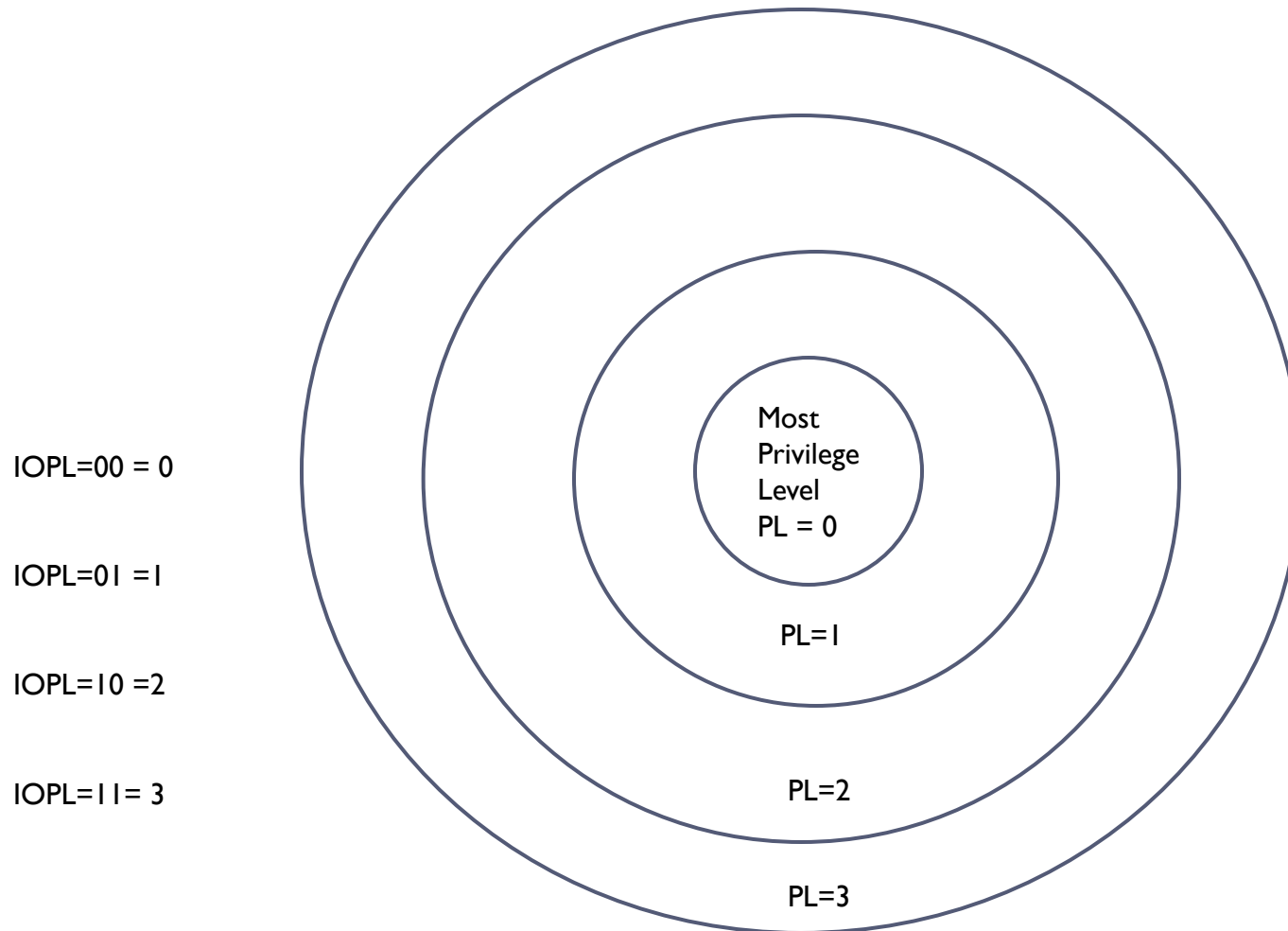
15	0	F	STATUS WORD
		IP	INSTRUCTION POINTER

STATUS AND CONTROL
REGISTERS

Flag Register



IOPL – Input Output Privilege Level flags (bit D12 and D13)



This mechanism **protects hardware** from direct access by untrusted user programs.

Normally:

PL = 0 → Most privileged (Kernel, Operating System).

PL = 3 → Least privileged (User programs).

- **OS kernel (PL=0)** has I/O privileges.
- **User apps (PL=3)** are blocked from I/O unless the OS explicitly allows it.

IOPL values (bits D12–D13)

- 00 → IOPL = 0 → Only code at **PL=0** can run I/O instructions.
- 01 → IOPL = 1 → Code at **PL=0 or PL=1** can run I/O.
- 10 → IOPL = 2 → Code at **PL=0,1,2** can run I/O.
- 11 → IOPL = 3 → Code at **all privilege levels (PL=0–3)** can run I/O.

IOPL – Input Output Privilege Level flags (bit D12 and D13)

- IOPL is used in protected mode operation to select the privilege level for I/O devices. If the current privilege level is higher or more trusted than the IOPL, I/O executed without hindrance.
- If the current privilege level is lower than the IOPL, an interrupt occurs, causing execution to suspend.
- Note that IOPL 00 is the highest or more trusted; and IOPL 11 is the lowest or least trusted.

-
- The CPU checks the **current Privilege Level (CPL)** of the running code against the **IOPL value** in FLAGS.
 - If **$CPL \leq IOPL$** , then I/O instructions are allowed.
 - If **$CPL > IOPL$** , the CPU blocks I/O operations and raises a **protection fault** (General Protection Exception).

NT – Nested task flag (bit D14)

- When set, it indicates that one system task has invoked another through a **CALL** instruction as opposed to a **JMP**.
- For multitasking this can be manipulated to our advantage

The **NT flag** is used in **task switching** (protected mode multitasking with Task State Segments, TSS).

When set (NT=1):

- It indicates that the current task was **invoked by another task** using a **CALL instruction** (task call).
- This means the CPU keeps track of the “calling task,” so when the current task finishes, it can **return control back** properly.

When clear (NT=0):

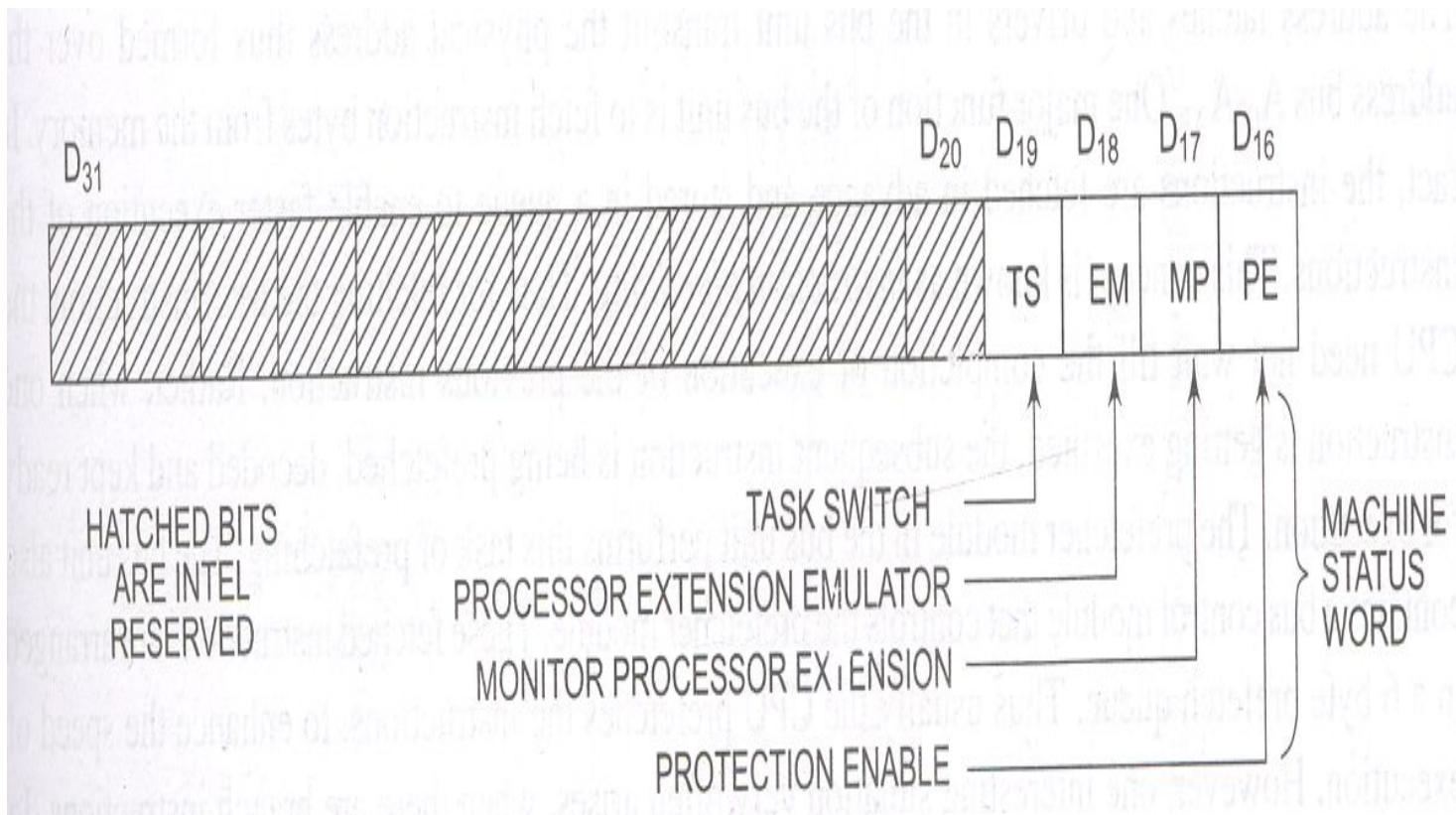
- The current task was not started by a task call, but maybe by a direct task switch (like a **JMP**).
- There's no “nested return” expected.

Machine Status Word Register

Consist of four flags

- PE,
- MP,
- EM and
- TS

Machine Status Word...



□ PE - Protection enable

Protection enable flag places the 80286 in protected mode, if set. This can only be cleared by resetting the CPU.

□ MP – Monitor processor extension

□ Flag allows WAIT instruction to generate a processor extension.

EM – Emulate processor extension flag:

if set , causes a processor extension absent exception and permits the emulation of processor extension by CPU.

TS – Task switch: if set, this flag allows task switching and if $TS=0$ there is no task switching. Task switching means the operating system switches from one task to another.

80286 ADDRESS TRANSLATION-S EGMENTATION

Operating Modes of 80286

- ? 80286 operates in either the
 - ? **real** or
 - ? **protected** mode.
- ? **Real mode operation** allows addressing of only the first 1M byte of memory space—even in Pentium or Core2 microprocessor.
- ? the first 1M byte of memory is called the **real memory , conventional memory , or DOS memory** system

Why is this Real Mode there?

? Protected Mode cannot be entered directly

To work in protected mode, various system tables are required. They are:

- 1) Global Descriptor Table
- 2) Local Descriptor Table
- 3) Page Tables

Segments and Offsets

- ? All real mode memory addresses must consist of a segment address plus an offset address.
- ? **segment address** defines the beginning address of any 64K-byte memory segment
- ? **offset address** selects any location within the 64K byte memory segment

Protected Mode:

- ? In real mode, PE=0
- ? When PE=1, 80286 enters into protected mode
- ? Then it continues to work in protected mode till the time processor is turned off
- ? Thus, after entering into protected mode, PE bit cannot be made 0.
- ? In protected mode, the calculation of physical address is different than that of real mode.

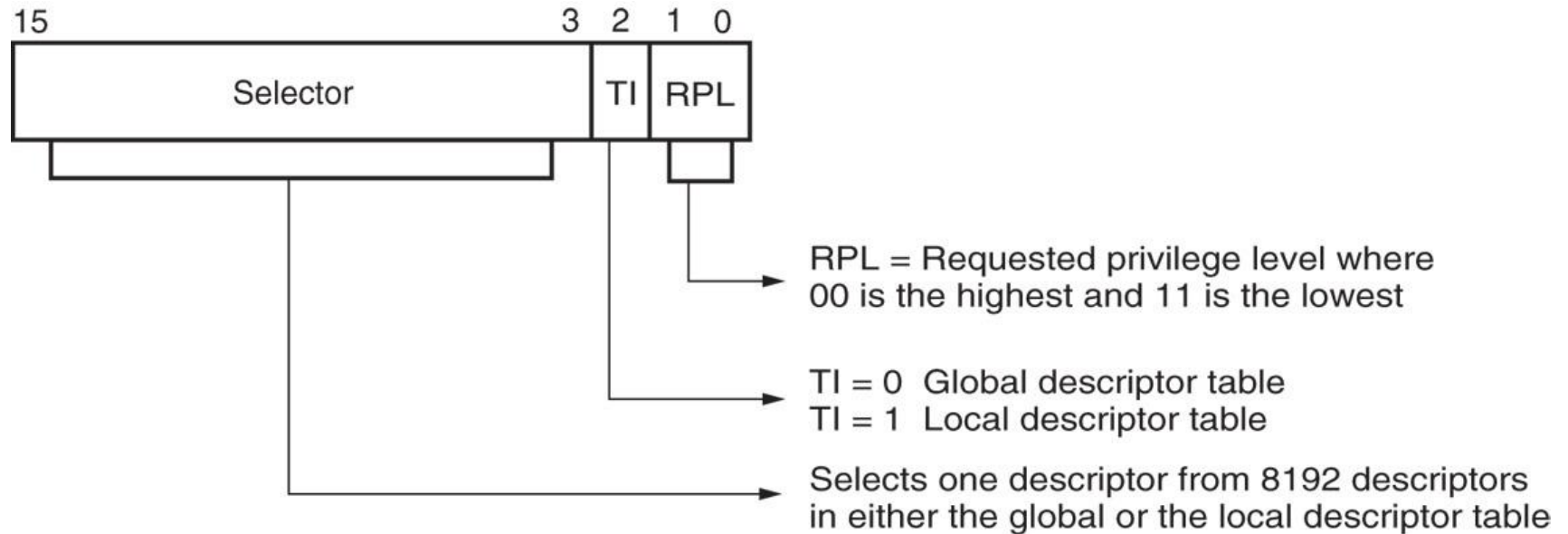
Protected Mode: Virtual Address

- ? Virtual address is the combination of segment address and offset address.
- ? Segment address tells which segment will be accessed. So, from segment address, the number of segments are to be known.
- ? Offset address tells the location within the segment and from it the maximum size of the segment is known

Descriptor

- ? There are mainly the LDT and GDT
- ? **Global descriptors** contain segment definitions that apply to all programs.
- ? **Local descriptors** are usually unique to an application.
 - ? a global descriptor might be called a **system descriptor**, and local descriptor an **application descriptor**

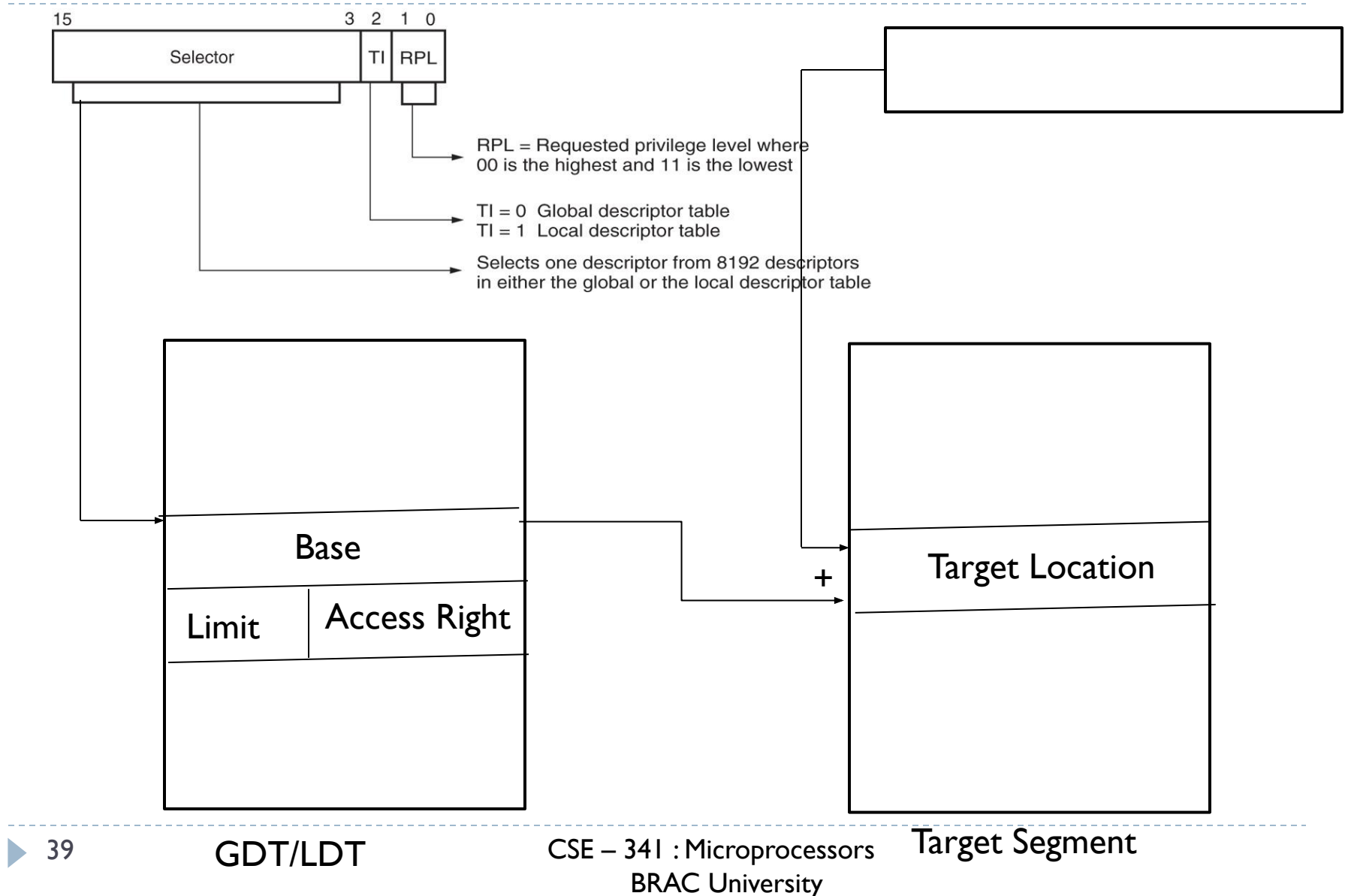
Contents of segment register



Segment descriptors are each 8k bytes

80286 Address Translation- Segmentation

OFFSET(16)



PROTECTED MODE MEMORY ADDRESSING

- ? **Protected mode** is where Windows operates.
- ? In place of a segment address, the segment register contains a **selector** that selects a **descriptor** from a **descriptor table**.
- ? The **descriptor** describes the memory segment's location, length, and access rights.

- ? Descriptors are chosen from the descriptor table by the segment register.
 - ? register contains a 13-bit selector field, a table selector bit, and requested privilege level field
- ? The **TI bit** selects either the global or the local descriptor table.
- ? **Requested Privilege Level (RPL)** requests the access privilege level of a memory segment.
 - ? If privilege levels are violated, system normally indicates an application or privilege level violation

Selectors and Descriptors

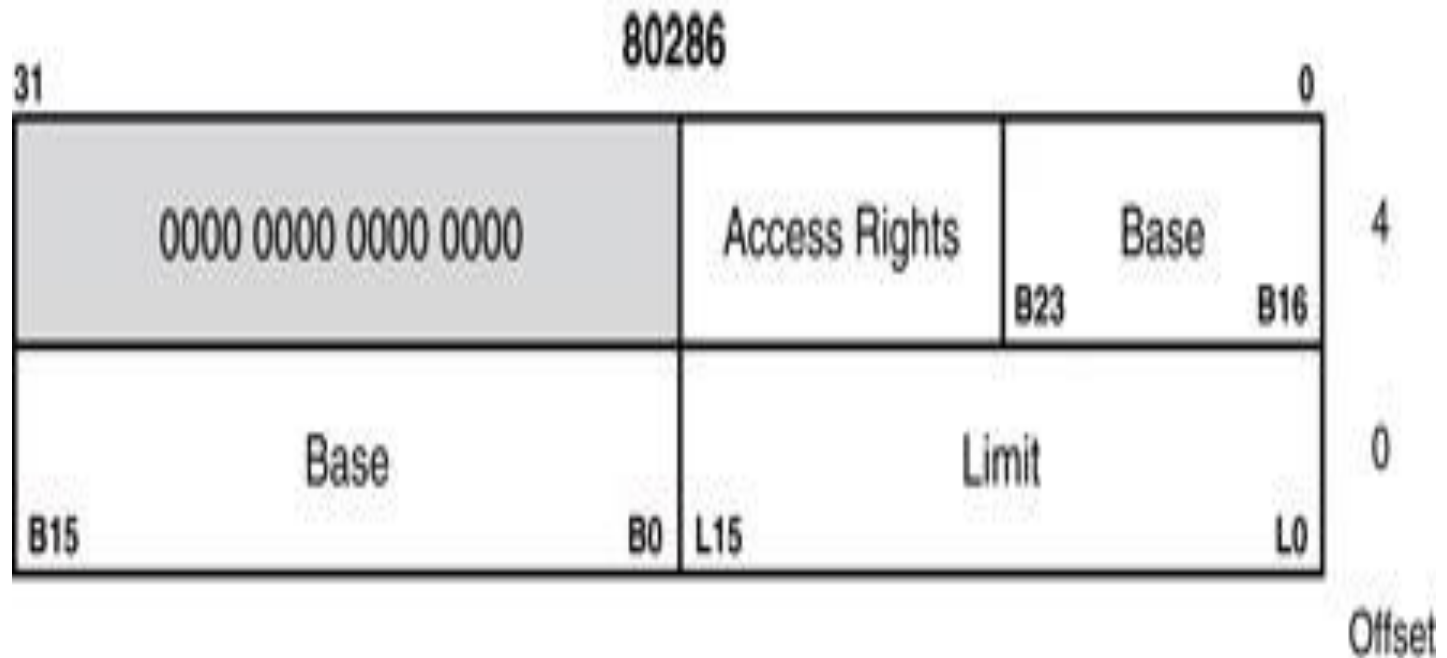
- ? The **selector**, located in the segment register, selects one of 8192 **descriptors** from one of two tables of descriptors.
- ? **Descriptor** describes the location, length, and access rights of the segment of memory.
- ? In protected mode, this segment number can address any memory location in the system.
- ? Indirectly, the segment register still selects a memory segment, but not directly as in real mode.

- ? **Global descriptors** contain segment definitions that apply to all programs.
- ? **Local descriptors** are usually unique to an application.
 - ? a global descriptor might be called a **system descriptor**, and local descriptor an **application descriptor**
- ? The global descriptor table's base address is stored in GDTR
- ? The local descriptor table's base address is stored in LDTR
- ? The two *privileged instructions* LGDT and LLDT loads the GDTR and LDTR.

80286

Descriptor

The 80286 descriptors



The 80286 descriptors

The 80286 descriptor has 3 parts:

1) Limit:

16-bit limit or size allows memory segment lengths of max^m 64K bytes.

2)Base:

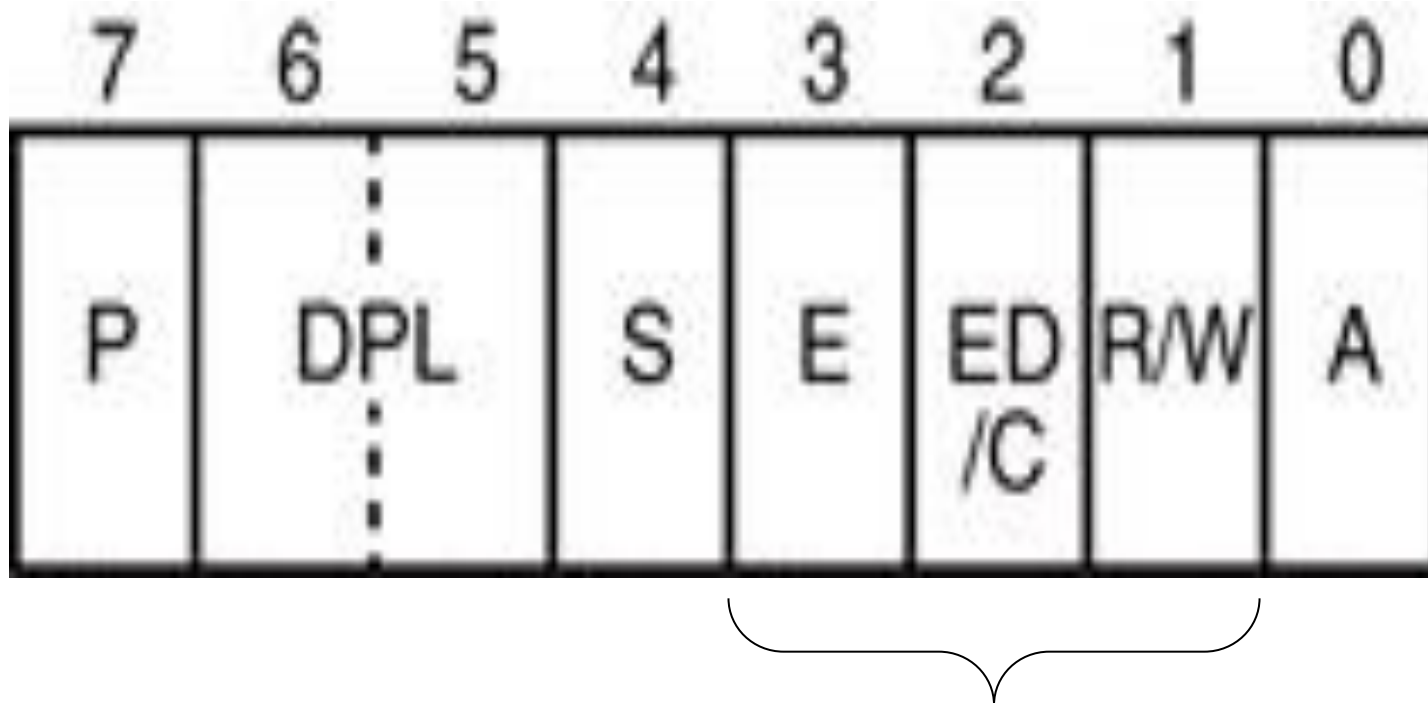
- ? The **base address** of the descriptor indicates the starting location of the memory segment.
- ? the paragraph boundary limitation is removed in protected mode
- ? segments may begin at any address

3) Access rights byte:

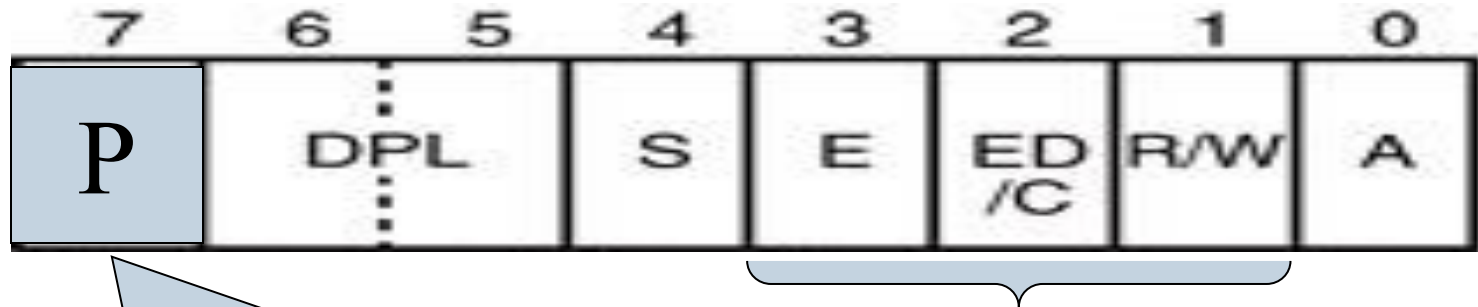
It controls access to the protected mode segment.

- ? describes segment function in the system and allows complete control over the segment
- ? if the segment is a data segment, the direction of growth is specified

Access Right Byte



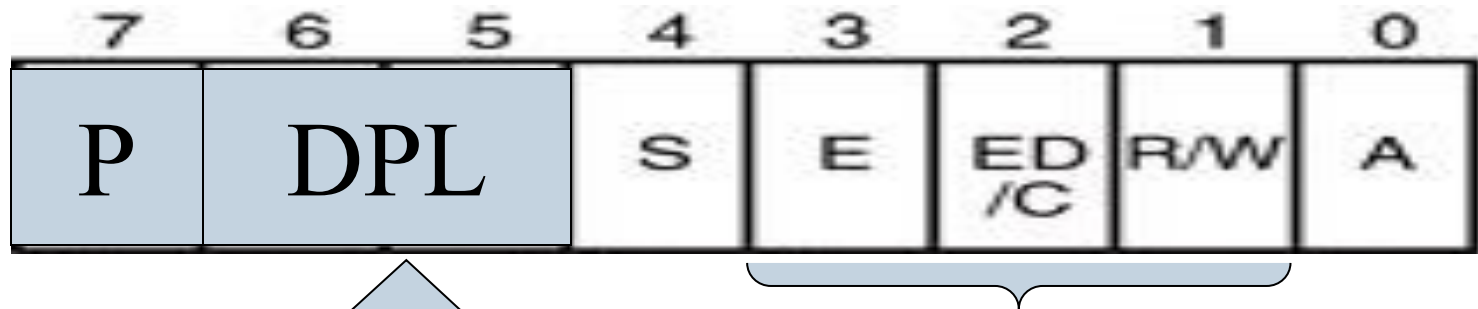
Access Right Byte --- Bit 7 Present?



P=1 Valid Descriptor info, Segment is mapped into physical memory

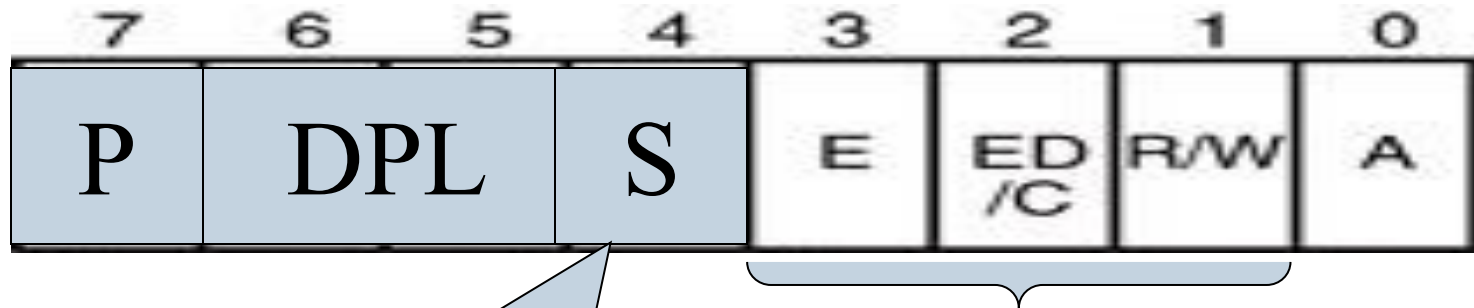
P=0 Descriptor is undefined, no mapping to physical memory exists

Access Right Byte --- Bit 5,6 Privilege Level



Sets the Descriptor privilege level necessary for protection

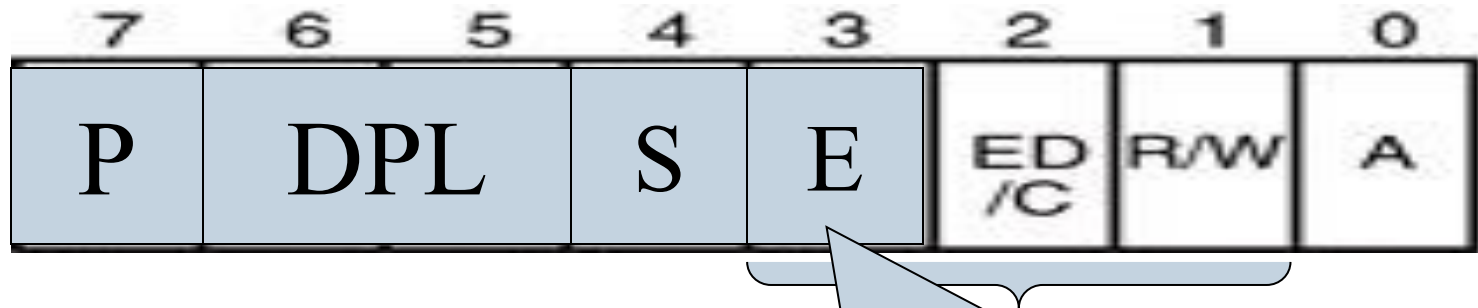
Access Right Byte --- Bit 4 Descriptor Type



S=0 System descriptor

S=1 Application Descriptor

Access Right Byte --- Bit 3 Executable?



E=0 No Data/Stack Segment

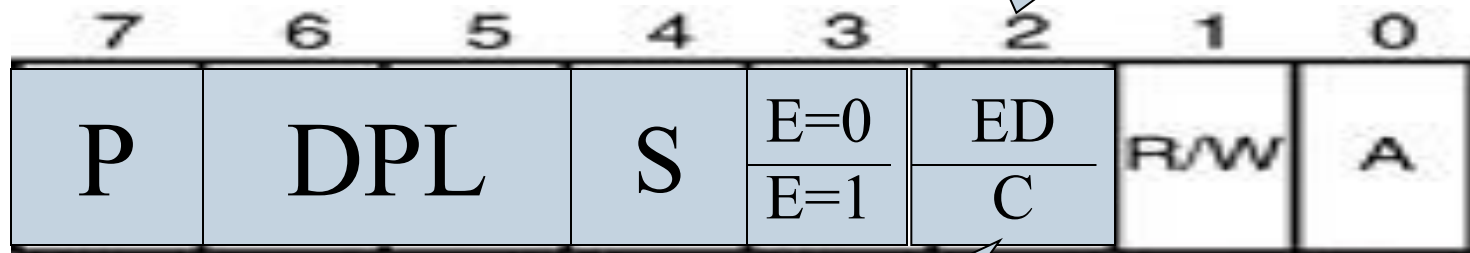
E=1 YES Code Segment

ED=0 Segment expands upward (Data segment)

ED=1 Segment expands downward (Stack Segment)

Access Right Byte

Bit 2



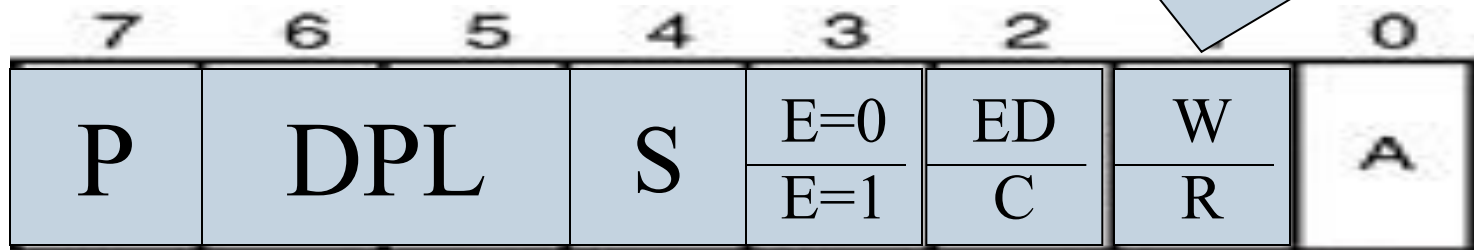
C=1 Ignore DPL

C=0 Abide by DPL

Access Right Byte --- Bit 1

W=0 Data segment not writable

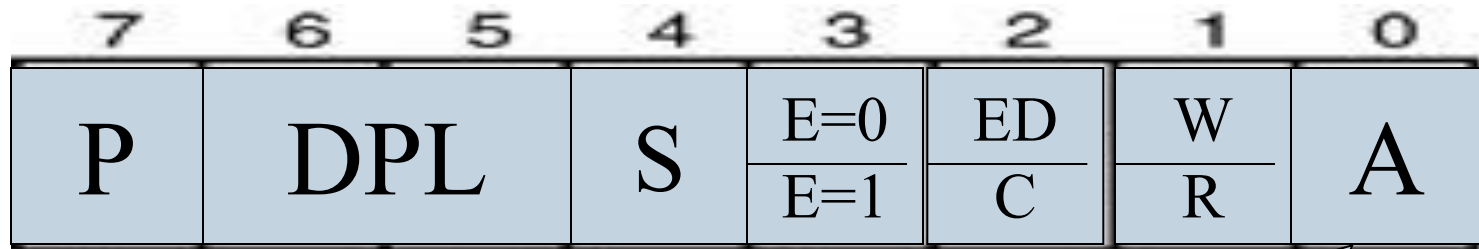
W=1 Data segment writable



R=0 Code Segment execute only, not readable

R=1 Code Segment both executable & readable

Access Right Byte --- Bit 0 Accessed?



A=0

Segment not accessed

A=1

Segment has been accessed

80286 Protection



Protection Mechanism

There are three checks in protection mechanism:

- ? Limit Check
- ? Type Check
- ? Privilege Check

Limit Check

? Offset is compared with the limit

? For 8 bit data,

$\text{Offset} \leq \text{Limit}$

? For 16 bit data,

$\text{Offset} \leq \text{Limit} - 1$

Type Check

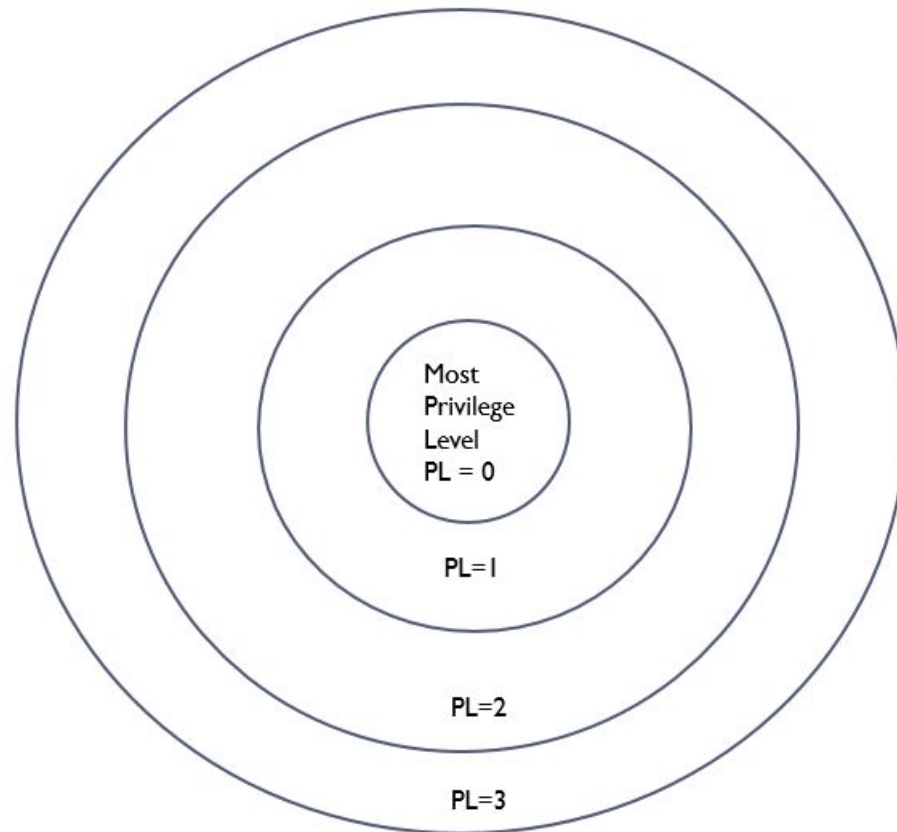
S E ED W

Or

S E C R

DESCRIPTOR BITS

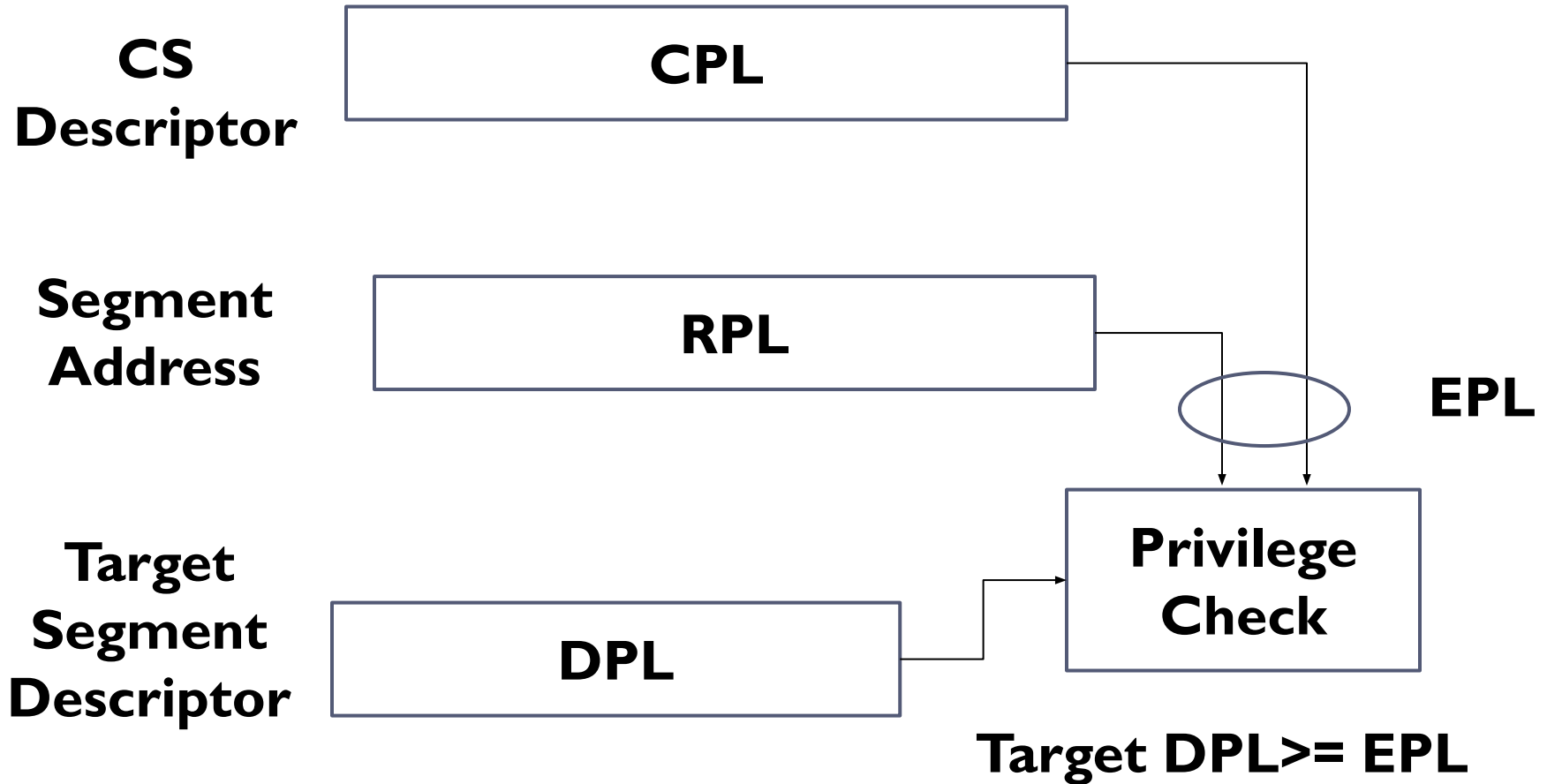
Privilege Check



Privilege Check

- ? Descriptor Privilege Level (DPL) – It is the privilege level of the segment to be accessed. It is there in the target segment descriptor.
- ? Request Privilege Level (RPL) – It is the privilege level of the program to be executed.
- ? Current Privilege Level (CPL) – It is there in the code segment register that is currently running.
- ? Effective Privilege Level (EPL) – Max (CPL, RPL)

Privilege Check



Privilege levels and Protection

- ? Every segment has an associated privilege level and hence any code segment will have an associated privilege level.
- ? The CPL (Current Privilege Level) of a process is the privilege level of the code segment, the code stored in which, it is executing.
- ? A process can access segments that have privilege levels numerically greater than or equal to (less privileged than) its CPL.

Thank You !! and

