80386 Microprocessor

Agenda

- ❑ 80386: Architecture (Block Diagram),
- ❑ Register organization,
- ❑ Memory Access in Protected Mode, Paging (Up to LA to PA)

Limitations of 286

- **16-bit** *ALU*
- **64K segment size**
- cannot be easily switched back and forth between real and protected mode
  - to come back to the real mode from protected mode, you have to switched off the 286

80386 was designed to overcome these limitations of 80186

- **32 bit ALU**
- segment size can be **as large as 4G**
  - a program can have as many as **16K segments.**
  - So, a program has access to **4Gx16K=64TB** of virtual memory
- 386 has a ***virtual 86*** mode which allows easy switching between real and protected modes.

80386 Features

- Alternatively referred to as a **386** or the **i386**
- **386** incorporates **275,000 transistor.**
- 386 was available in clock speeds between **12 and 40MHz.**

Introduction : 80386

- Enhanced version of 80286 microprocessor.
- 32-bit processor i.e **32-bit** data bus, **32-bit** registers, **32-bit** address bus.
- Addressable **physical memory** is **2^{32}=** 4GB.
- 80386 CPU supports 16KB of segments, thus total virtual memory space = 4GB X 16KB = 64TB.
- The 80386 memory manager is similar to the 80286, except the physical addresses generated by the MMU are 32-bit wide instead of 24-bit.
- The concept of **paging** was introduced in 80386.

Introduction : 80386

- ❑386 is capable of performing more than 5 **Million Instructions Per Second** (MIPS).
- ❑ 80386 support 3 operating modes:
  - ✓Real Mode (default)
  - ✓Protected Virtual Address Mode (PVAM)
  - ✓Virtual Mode
- ❑The memory management section of 80386 supports virtual memory, paging and four levels of protection.
- ❑The 80386 includes special hardware for task switching.

versions of 386

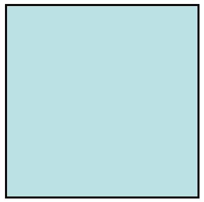
- Two versions were commonly available:
  1) 80386DX
  2) 80386SX

DX vs SX

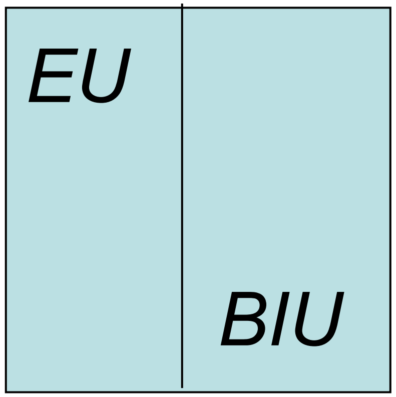
80386DX	**80386SX**
32 bit address bus 32 bit data bus	24 bit address bus 16 bit data bus
Packaged in **132 pin** PGA	**100 pin** flat package
Address **4GB of memory**	**16 MB** of memory

- Both have the **same internal architecture**.(ALU is same)

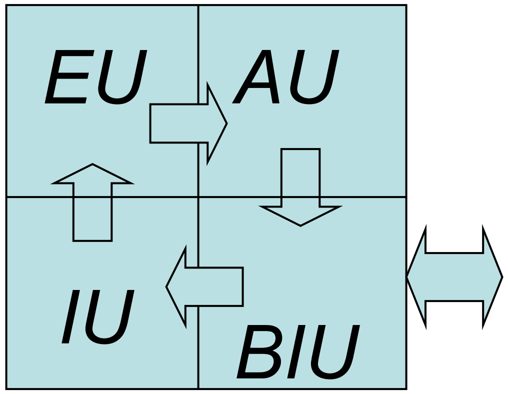
80386
Internal Architecture



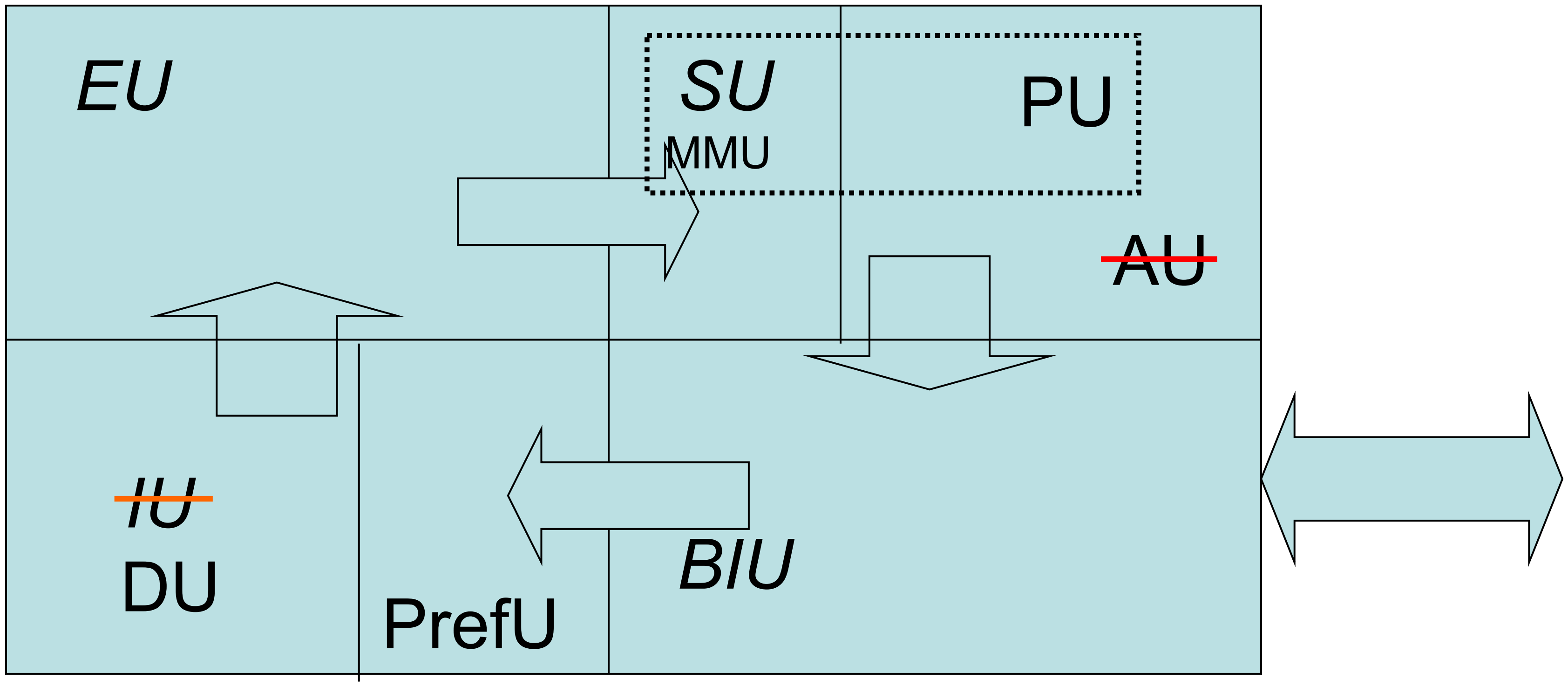
8085



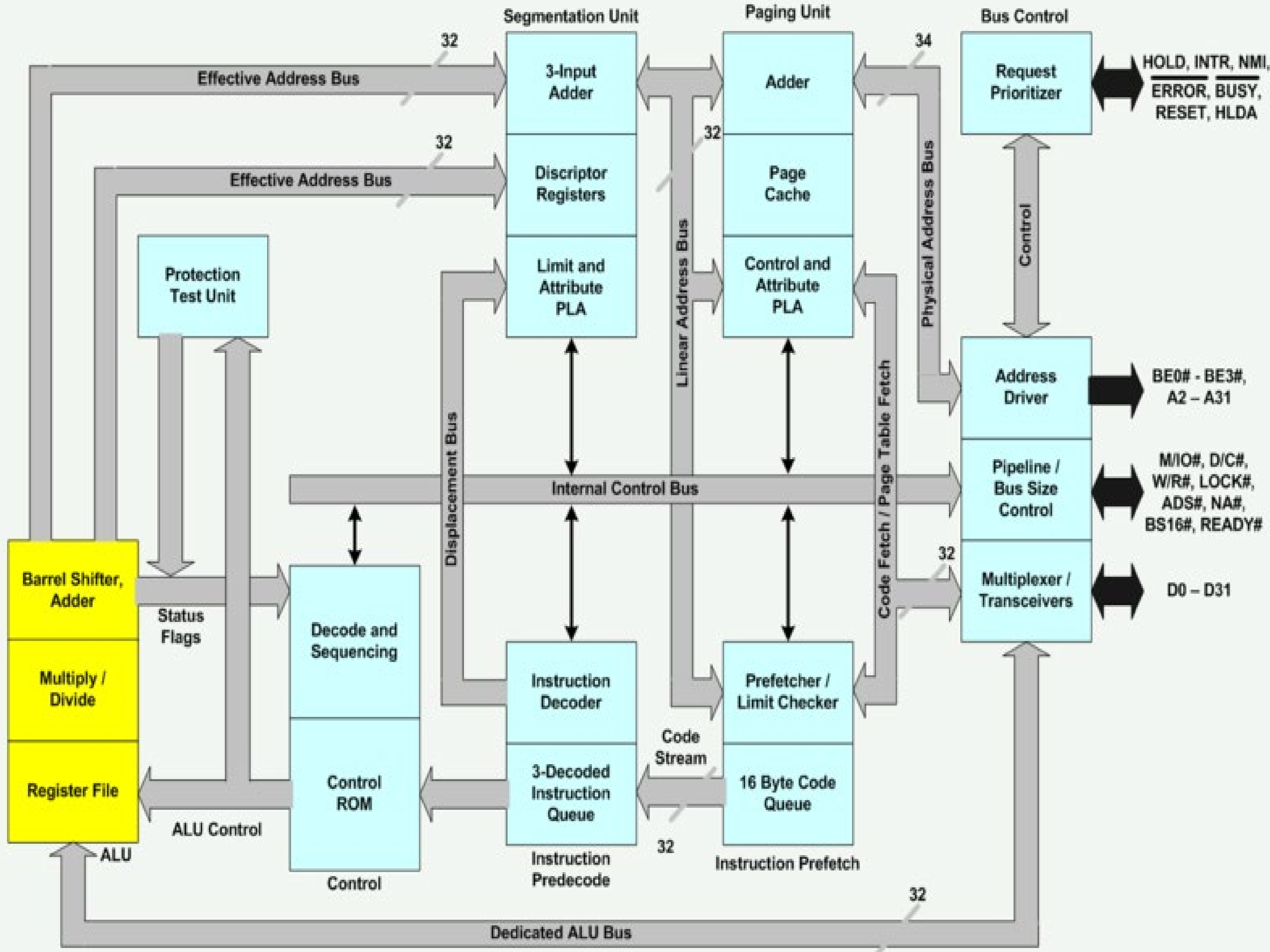
8086



80286



80386


Segmentation Unit
Paging Unit
Bus Control
32
Effective Address Bus
3-Input Adder
Adder
34
Request Prioritizer
HOLD, INTR, NMI, ERROR, BUSY, RESET, HLDA
32
Effective Address Bus
Discriptor Registers
32
Page Cache
Physical Address Bus
Control
Protection Test Unit
Limit and Attribute PLA
Linear Address Bus
Control and Attribute PLA
Address Driver
BE0# - BE3#, A2 – A31
Displacement Bus
Code Fetch / Page Table Fetch
Internal Control Bus
Pipeline / Bus Size Control
M/IO#, D/C#, W/R#, LOCK#, ADS#, NA#, BS16#, READY#
32
Barrel Shifter, Adder
Status Flags
Decode and Sequencing
Multiplexer / Transceivers
D0 – D31
Multiply / Divide
Instruction Decoder
Prefetcher / Limit Checker
Code Stream
Register File
Control ROM
3-Decoded Instruction Queue
16 Byte Code Queue
ALU Control
32
ALU
Control
Instruction Predecode
Instruction Prefetch
32
Dedicated ALU Bus

Architecture of 80386

- The Internal Architecture of 80386 is ***divided into 3 sections:***
  - **i) Central processing unit (CPU)** which is further divided into
    - Execution unit (EU) and
    - Instruction queue and instruction **Decode Unit (DU)**
  - **ii) Memory management unit (MMU)** consists of
    - Segmentation unit(SU) and
    - Paging unit(PU).
  - **iii) Bus interface unit(BIU)**

Architecture of 80386...

I. Execution unit (EU)

- **Execution unit** has **8 General purpose registers** which are either used for handling data or calculating offset addresses.

I. Execution unit (EU)

- The 80386 has **eight 32-bit general purpose registers** which may be used as *either 8 bit, 16 bit or 32 bit registers*.

- A 32-bit register known as an **extended register**, is represented by the register name with prefix E.
  - ✓ Example : A 32 bit register corresponding to AX is **EAX**

- So the general purpose registers of 386 are
  - ✓ **EAX, EBX, ECX, EDX, EBP, ESP, ESI and EDI**

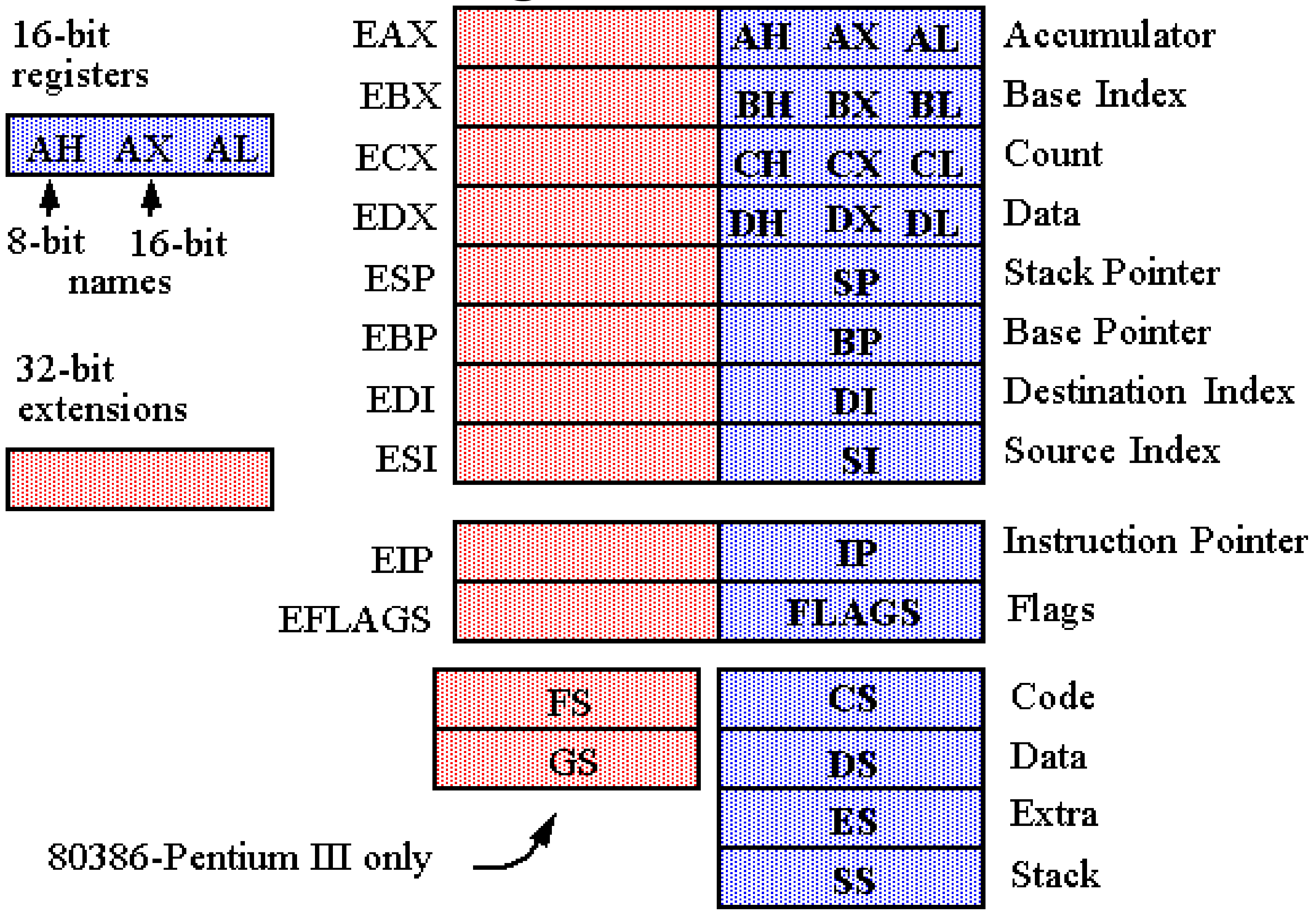
I. Execution unit (EU)

- **BP, SP, SI, DI** represents the lower 16 bit of their 32 bit counterparts, and can be used as independent 16 bit registers.

- The 16 bit flag register is available along with 32 bit counterpart **EFLAGS**.

I. Execution unit (EU)

Registers of 80386



Registers of 80386

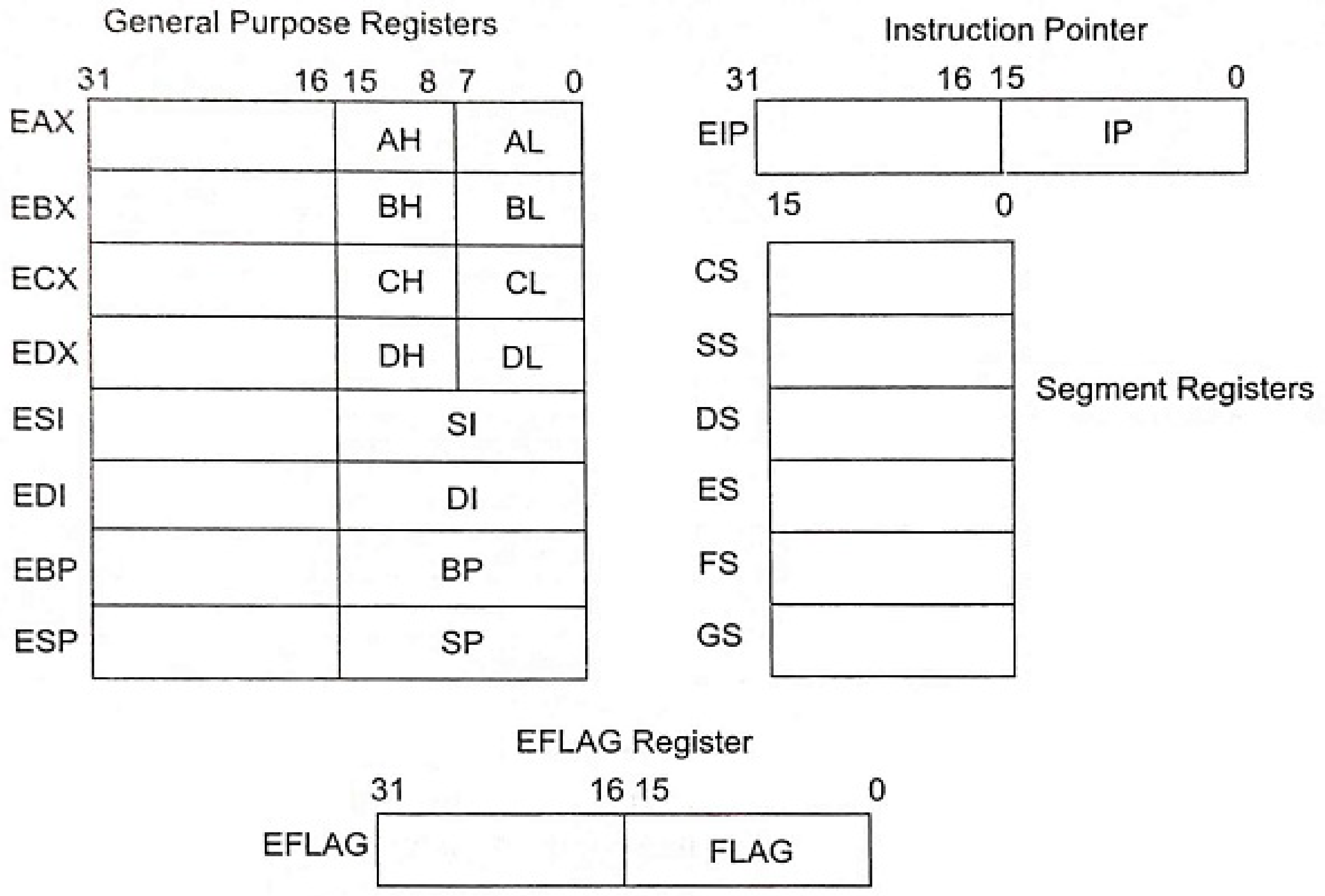


Fig. 11.33 *Registers of 80386*

Registers of 80386

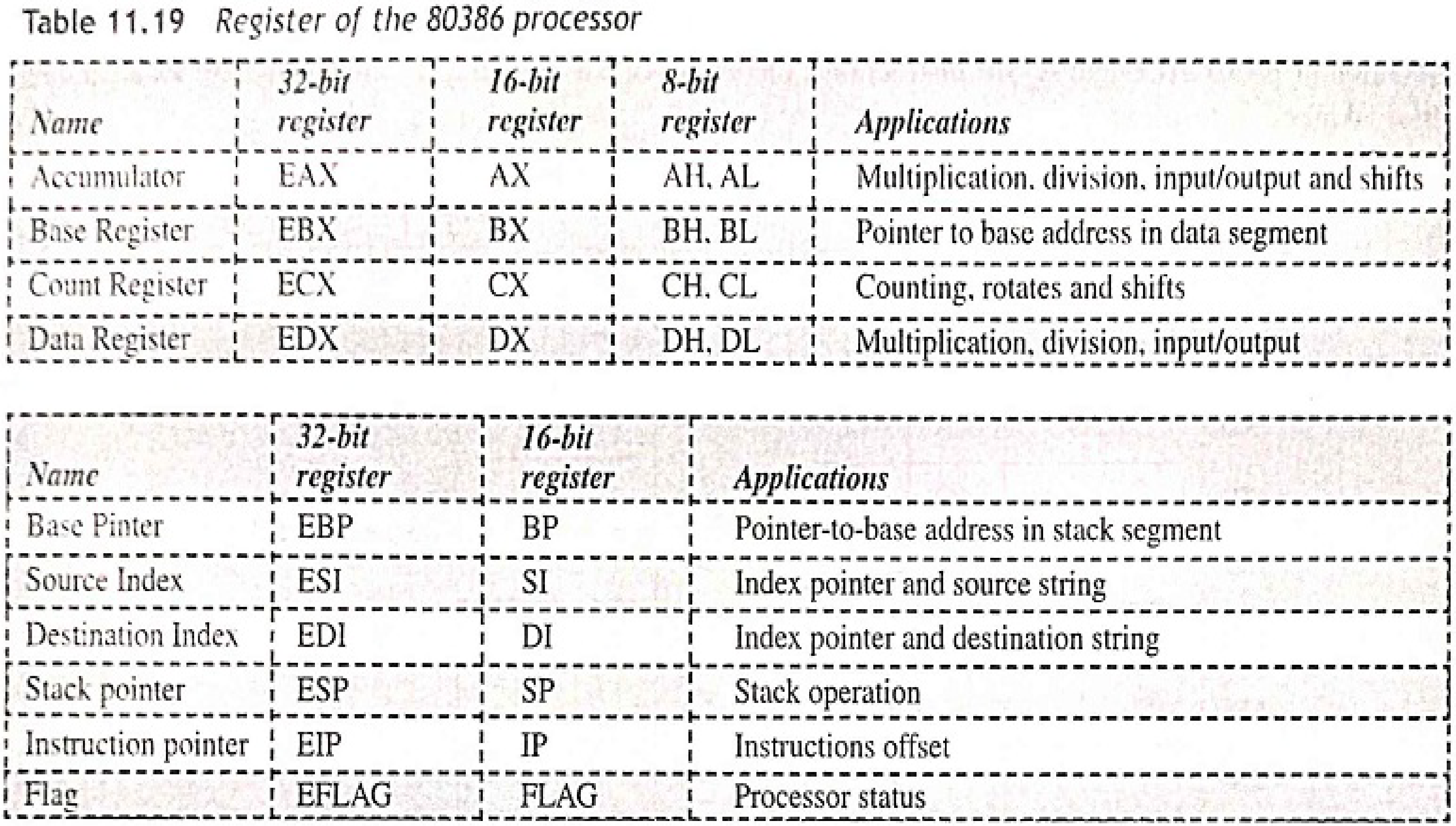


Table 11.19 *Register of the 80386 processor*

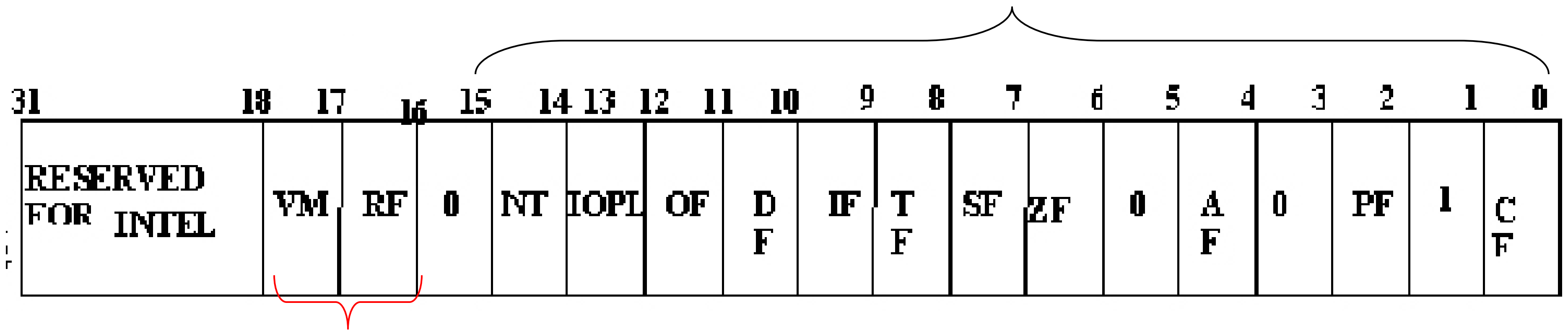
Name	*32-bit register*	*16-bit register*	*8-bit register*	*Applications*
Accumulator	EAX	AX	AH, AL	Multiplication, division, input/output and shifts
Base Register	EBX	BX	BH, BL	Pointer to base address in data segment
Count Register	ECX	CX	CH, CL	Counting, rotates and shifts
Data Register	EDX	DX	DH, DL	Multiplication, division, input/output

Name	*32-bit register*	*16-bit register*	*Applications*
Base Pinter	EBP	BP	Pointer-to-base address in stack segment
Source Index	ESI	SI	Index pointer and source string
Destination Index	EDI	DI	Index pointer and destination string
Stack pointer	ESP	SP	Stack operation
Instruction pointer	EIP	IP	Instructions offset
Flag	EFLAG	FLAG	Processor status

Flag Register

- The **Flag register** of **80386 is a 32 bit register**.
- Out of the 32 bits, Intel has reserved bits *D18 to D31, D5 and D3 and set to 0*
- while *D1 is always set at 1.*
- **Two extra new flags** are added to the 80286 flag to derive the flag register of 80386.
- They are **VM and RF flags.**

Flag Register...



New flags for 386

Flag Register of 80386

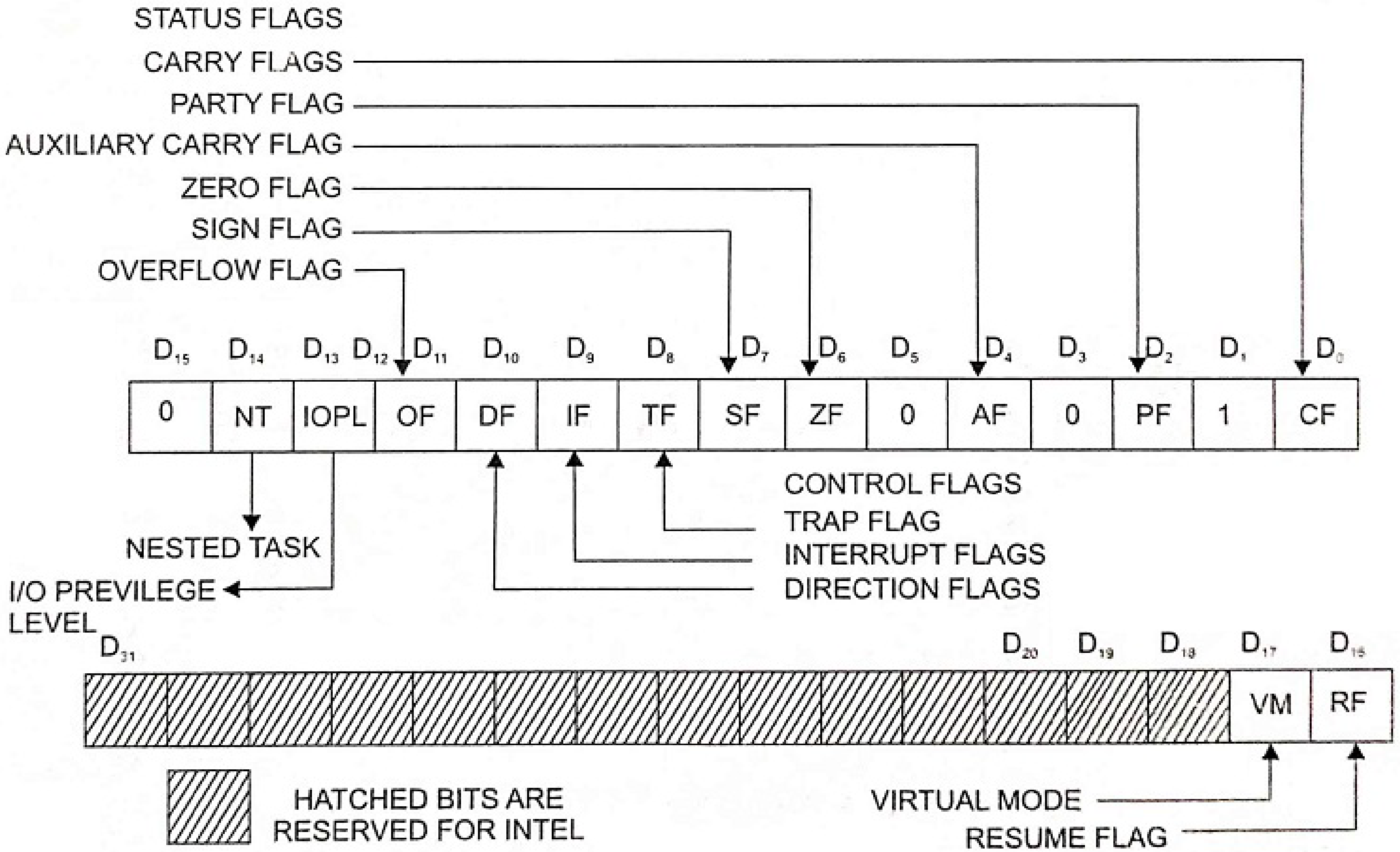


Fig. 11.35 *FLAG register of the 80386*

VM - *Virtual Mode Flag*

- If **this flag is set to 1**, the 80386 enters the **virtual 8086 mode** within the protection mode.
- When **VM bit is 0**, 386 operates in **protected mode.**
- When this flag is **cleared or reset**, the 80386 operates in **real address mode**.

RF- Resume Flag

- This flag is used with the **debug register breakpoints.**
- If **RF=1**, *386 ignores debug faults*
- If **RF=0,** 386 takes another debug exception to service **debug faults**
- This flag is automatically reset after execution of instructions except IRET and POPE.

Architecture of 80386...

Instruction unit(IU)

- The Instruction unit decodes the op-code bytes received from the **16-byte instruction code queue** and arranges them in a ***3-instruction decoded instruction*** queue.

- After decoding them pass it to the **control section** for deriving the necessary control signals.

Architecture of 80386...
execution unit

- The **barrel shifter** increases the **speed of all shift and rotate operations.**

- The *multiply / divide logic implements* the bit-shift-rotate algorithms to complete the operations in minimum time.

Architecture of 80386...

Segmentation unit

- **Segmentation unit** allows the use of two address components, viz. **segment and offset** for relocate ability and sharing of code and data.

- **Segmentation Unit(SU)** allows segments of **size 4GB at max.**

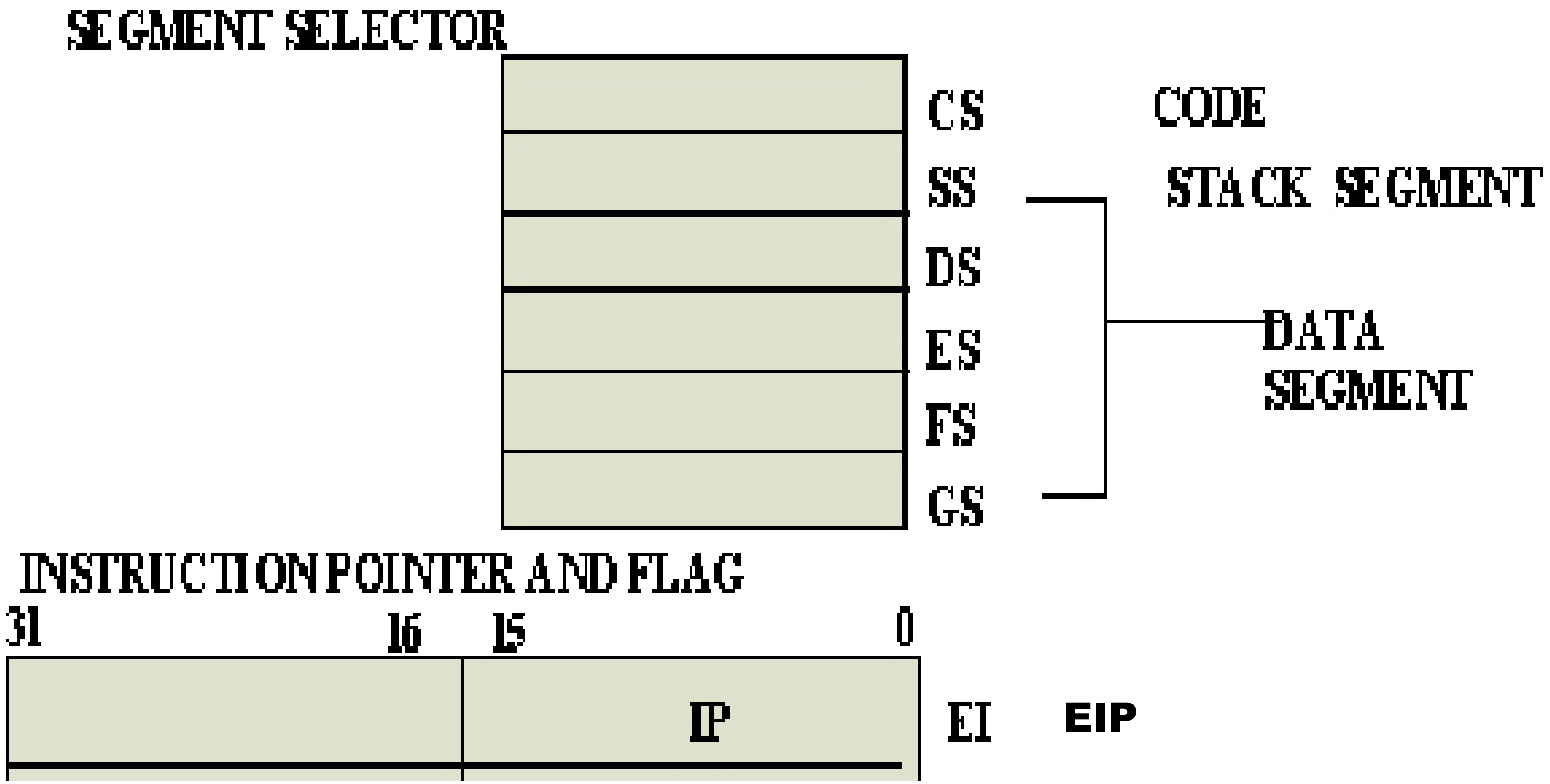
- The **Segmentation unit** provides a **4 level(00.01.10.11)** protection mechanism for protecting and isolating the system code and data from those of the application program.

- *Besides 4 segment registers* of 286, 386 has **2 more segment registers**
- The **six segment registers** available in 80386 are *CS, SS, DS, ES, FS and GS.*

  FS: all-Function Segment

  GS: General-purpose Segment.

- The CS and SS are the code and the stack segment registers respectively, while DS, ES,FS, GS are 4 data segment registers.
- A 16 bit instruction pointer IP is available along with 32 bit counterpart EIP.


SEGMENT SELECTOR
CS
SS
DS
ES
FS
GS
CODE
STACK SEGMENT
DATA SEGMENT
INSTRUCTION POINTER AND FLAG
31
16
15
0
IP
EI
EIP

Architecture of 80386...

Paging unit

- The Paging unit organizes the **physical memory** in terms of pages of **4kbytes** size each.
- Paging unit works under the control of the **segmentation unit**, i.e. The virtual ***each segment*** *is further divided into* ***pages.***
- memory is also organizes in terms of segments and pages by the **memory management unit**.
- **Paging unit** converts *linear addresses into physical addresses.*

Architecture of 80386...

Paging unit

- The control and attribute PLA checks the privileges at the page level.
-
- Each of the pages maintains the paging information of the task.
- The limit and attribute **PLA(Privileged Level Attribute)** checks segment limits and attributes at segment level to **avoid invalid accesses** to *code and data* in the memory segments.

Architecture of 80386...

Bus control unit

- The **Bus control unit** has a prioritize to resolve the priority of the various bus requests.
- This controls the access of the bus.
- The address driver drives the bus enable and address signal **A2 – A31**. *(A0,A1---to support 4 chips(1gb----4x1-4gb+)*
- The pipeline and dynamic bus sizing unit handle the related control signals.
- The data buffers interface the internal data bus with the system bus.

Hidden Registers/ Program invisible registers/ Special Registers

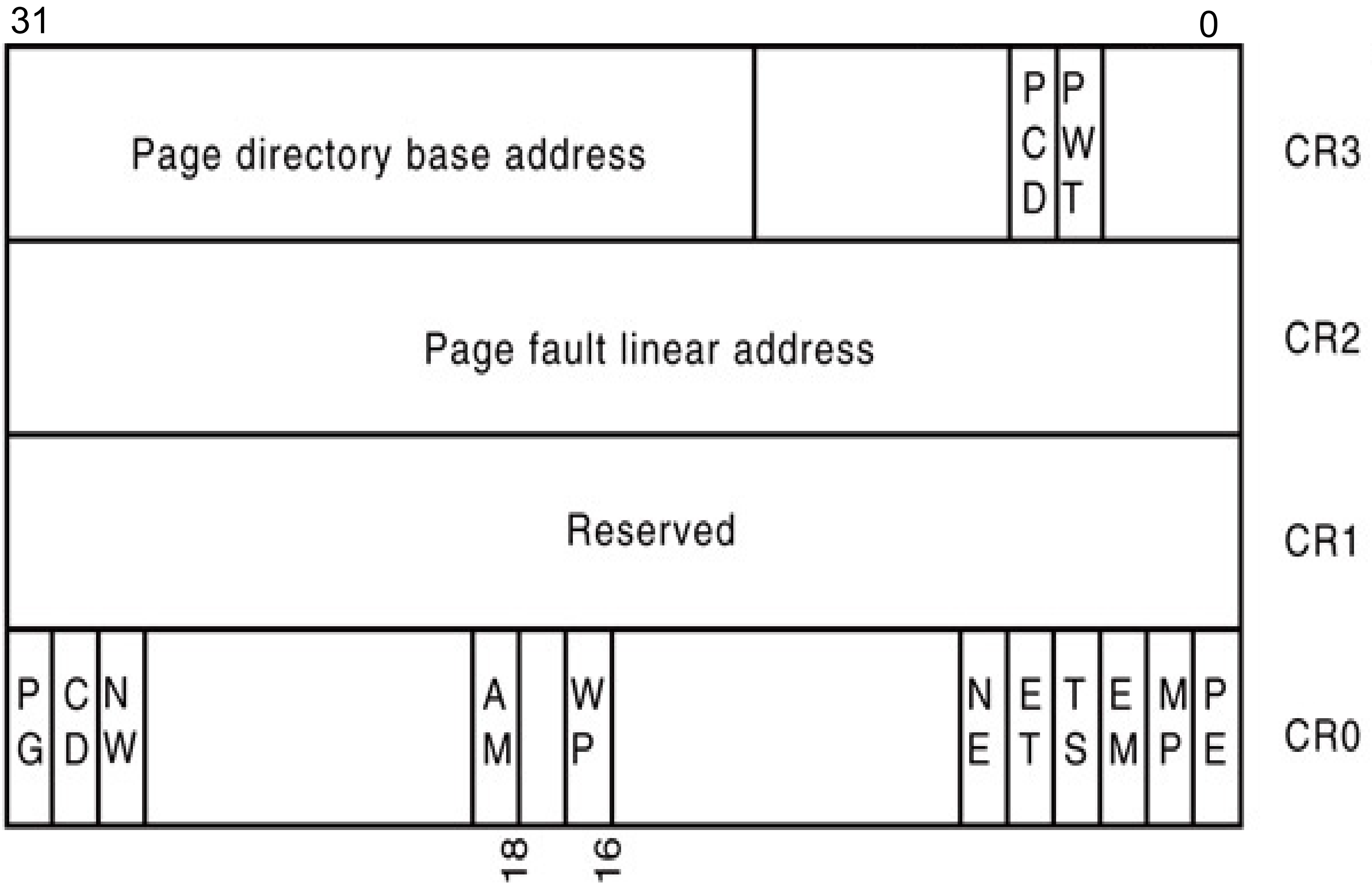
Segment Descriptor Registers

- **The six segment registers (CS, DS, ES, FS, GS & SS) have corresponding** six 73 bit descriptor registers.
- Each of them contains **32 bit base address**, **32 bit base limit** and **9 bit attributes(**73 bit).
- These are *automatically loaded* when the corresponding segments are loaded with selectors.

Control Registers

- The 80386 has four **32 bit control registers CR0, CR1, CR2 and CR3** to hold global machine status independent of the executed task.
- *CR1 is not used in 386*. It is reserved for future use.
- Load and store instructions are available to access these registers.

386 control registers



Debug Registers

- Intel has provide a set of **8 debug registers** for *hardware debugging*.
- Out of these **eight registers DR0 to DR7**, two registers *DR4 and DR5 are Intel reserved*.

Debug Registers

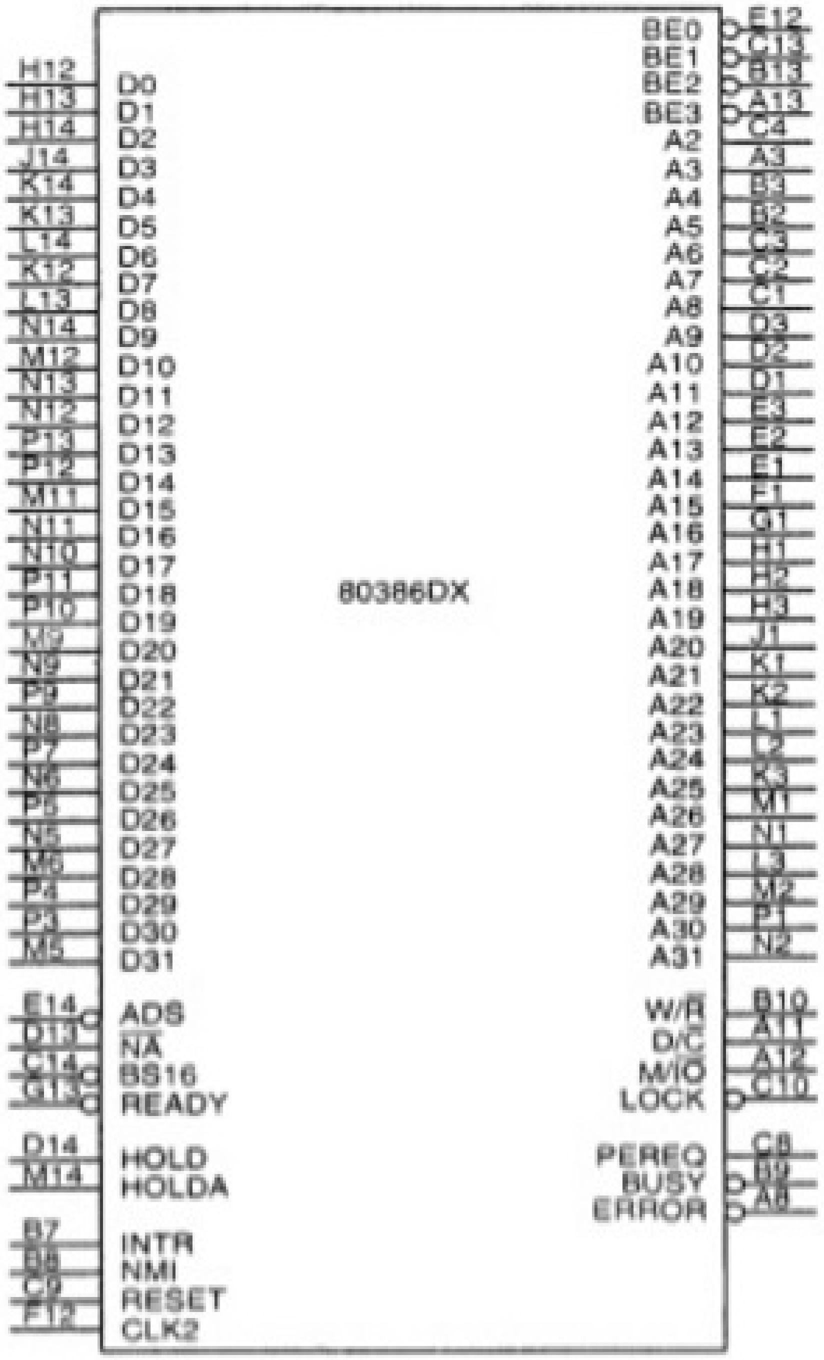
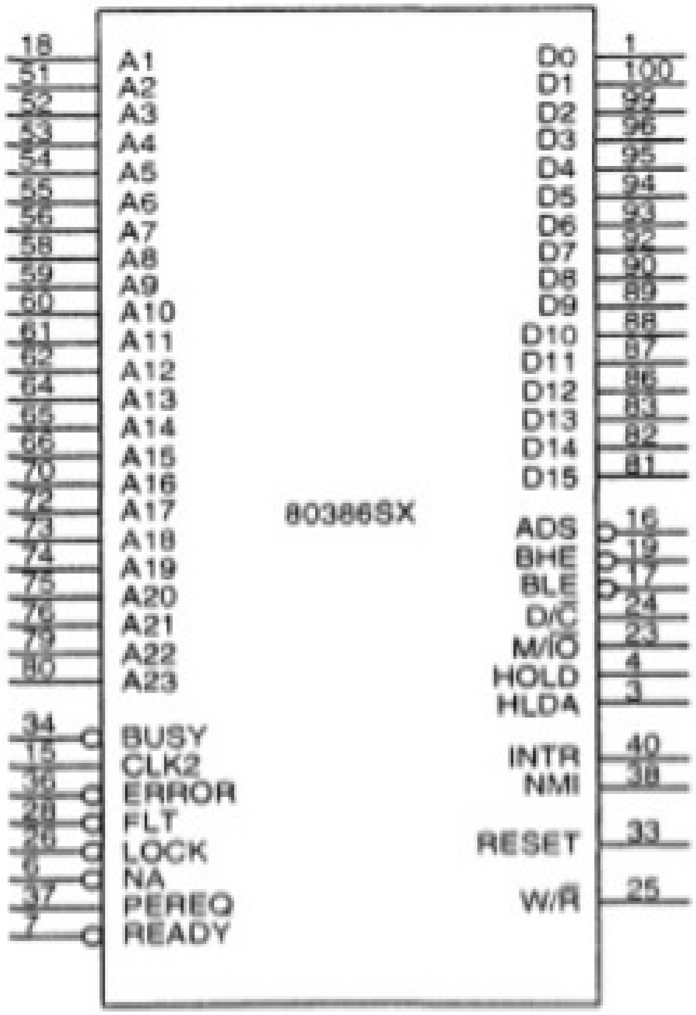
breakpoint control info	DR7
breakpoint status	DR6
RESERVED	DR5
RESERVED	DR4
Linear breakpoint address 3	DR3
Linear breakpoint address 2	DR2
Linear breakpoint address 1	DR1
Linear breakpoint address 0	DR0

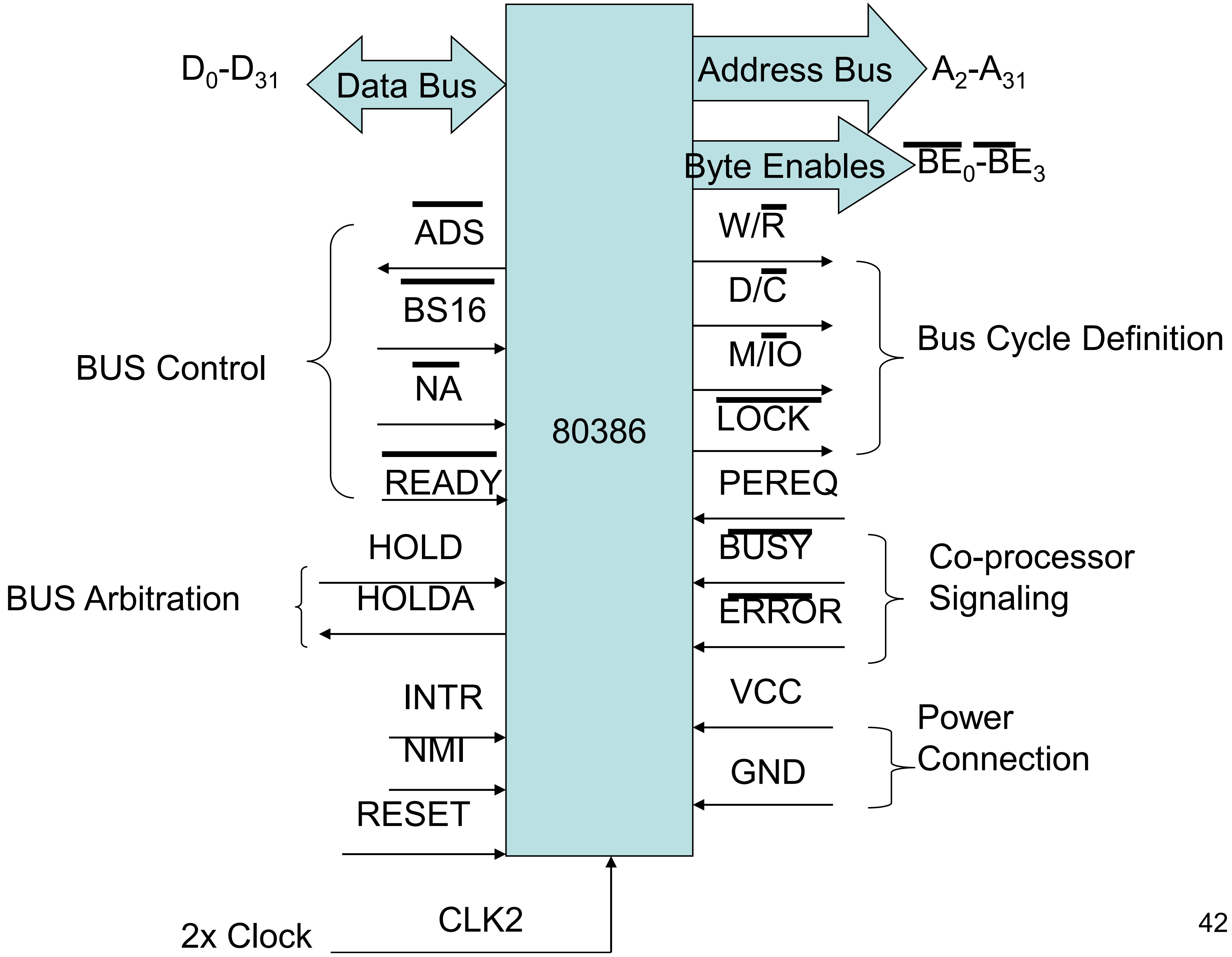
31 0

Test Registers

- **Two test register** are provided by 80386 for page caching namely ***test control*** *and* ***test status register.***

386 Pin Diagram


80386DX
H12 D0
H13 D1
H14 D2
J14 D3
K14 D4
K13 D5
L14 D6
K12 D7
L13 D8
N14 D9
M12 D10
N13 D11
N12 D12
P13 D13
P12 D14
M11 D15
N11 D16
N10 D17
P11 D18
P10 D19
M9 D20
N9 D21
P9 D22
N8 D23
P7 D24
N6 D25
P5 D26
N5 D27
M6 D28
P4 D29
P3 D30
M5 D31
E14 ADS
D13 NA
C14 BS16
G13 READY
D14 HOLD
M14 HOLDA
B7 INTR
B8 NMI
C9 RESET
F12 CLK2
BE0 E12
BE1 C13
BE2 B13
BE3 A13
A2 C4
A3 A3
A4 B3
A5 B2
A6 C3
A7 C2
A8 C1
A9 D3
A10 D2
A11 D1
A12 E3
A13 E2
A14 E1
A15 F1
A16 G1
A17 H1
A18 H2
A19 H3
A20 J1
A21 K1
A22 K2
A23 L1
A24 L2
A25 K3
A26 M1
A27 N1
A28 L3
A29 M2
A30 P1
A31 N2
W/R B10
D/C A11
M/IO A12
LOCK C10
PEREQ C8
BUSY B9
ERROR A8

80386SX
18 A1
51 A2
52 A3
53 A4
54 A5
55 A6
56 A7
58 A8
59 A9
60 A10
61 A11
62 A12
64 A13
65 A14
66 A15
70 A16
72 A17
73 A18
74 A19
75 A20
76 A21
79 A22
80 A23
34 BUSY
15 CLK2
36 ERROR
28 FLT
26 LOCK
6 NA
37 PEREQ
7 READY
D0 1
D1 100
D2 99
D3 96
D4 95
D5 94
D6 93
D7 92
D8 90
D9 89
D10 88
D11 87
D12 86
D13 83
D14 82
D15 81
ADS 16
BHE 19
BLE 17
D/C 24
M/IO 23
HOLD 4
HLDA 3
INTR 40
NMI 38
RESET 33
W/R 25


D_0-D_{31}
Data Bus
Address Bus
A_2-A_{31}
Byte Enables
$\overline{BE_0}$-$\overline{BE_3}$
$\overline{ADS}$
$\overline{BS16}$
$\overline{NA}$
$\overline{READY}$
BUS Control
HOLD
HOLDA
BUS Arbitration
INTR
NMI
RESET
80386
W/$\overline{R}$
D/$\overline{C}$
M/$\overline{IO}$
$\overline{LOCK}$
Bus Cycle Definition
PEREQ
$\overline{BUSY}$
$\overline{ERROR}$
Co-processor Signaling
VCC
GND
Power Connection
CLK2
2x Clock

Address Bus

- Address bus consists of **A2 to A31 address lines** and <u>*BE0 to BE3* **byte/bank enable lines**</u>
- No **A0 & A1 address lines** are available in 386
  - they are internally **decoded to produce BE0 to BE3 signals**
- Memory are arranged in **4 Banks**
- BE0-BE3 allow 386 to transfer ***byte***, ***word*** and ***double word***

I/O Ports Signals...

- 386 direct port structure is an extension of 286 port structure to include 32-bit ports

- 32-bit I/O ports can be constructed by connecting 8-bit I/O port devices --- such as 8255 --- in parallel

I/O Ports Signals...

- 386 ***IN*** or ***OUT*** instruction followed by 8-bit port address can be used to address 256 8-bit port or 128 16-bit port or 64 32-bit ports.
- Using DX register to hold port address 386 can address 64K 8-bit port,32K 16-bit port or 16K 32-bit ports
- I/O port addresses 00F8H through 00FFH are reserved by Intel
- Co-processors are at 800000F8H through 800000FFH.

Bus control Signals

- **ADS : ADDRESS STATUS** indicates that a valid bus cycle definition and address
- **BS16 : BUS SIZE 16** input allows direct connection of 32-bit and 16-bit data buses
- **NA : NEXT ADDRESS** is used to request address pipelining by an external hardware

Co-processor Signals

- ✓ **PEREQ:PROCESSOR EXTENSION REQUEST**
  - ▪ indicates that the processor extension has data to be transferred by the 386 DX.
- ✓ **BUSY :**
  - ▪ indicates that co-processor is still busy probably still executing the previous instruction
- ✓ **ERROR :**
  - ✓ signals an error condition from a processor extension

Clock Signals...

- ✓ **CLK2** :
  - provides the basic timing for 386.
  - applied clock is internally divided by 2 to produce clock signals which actually drives 386 operations
  - for 33 MHz operation then, a 66 MHz clock is applied to CLK2

BIST

- 386 contains a large amount of ***Built-In Self Test (BIST)*** circuitry
- if ***BUSY*** is held **low,** while ***RESET*** is held **low,** 386 automatically enters into BIST
- if 386 passes all tests, a ***signature*** of all 0’s will be put in EAX register

Memory Organization

- Memory is organized in 4 banks

ADDRESSING MODES

ADDRESSING MODES

- Same as 286 with one new mode

  <u>Scale Indexed Addressing Mode</u>

- *Scaled index mode* is efficient for accessing arrays or structures.

- In case of all those modes, the 80386 can now have ***<u>32-bit immediate or 32- bit register operands or displacements.</u>***

- In case of scaled modes, any of the index register values can be multiplied by a valid scale factor to obtain the displacement.

- The valid scale factor are **1, 2, 4 and 8.**

ADDRESSING MODES...
different scaled modes

- **Scaled Indexed Mode**: Contents of an index register are multiplied by a scale factor that may be added further to get the operand offset.

  MOV AX, 50[2*EDI]

- **Based Scaled Indexed Mode**: Contents of the an index register are multiplied by a scale factor and then added to base register to obtain the offset.

  MOV AX, [2*EDI+ECX]

ADDRESSING MODES... different scaled modes

- **Based Scaled Indexed Mode with Displacement:** The Contents of the an index register are multiplied by a scaling factor and the result is added to a base register and a displacement to get the offset of an operand.

```
MOV [EAX+2*EDI+100H], CX
```

386 modes of operation

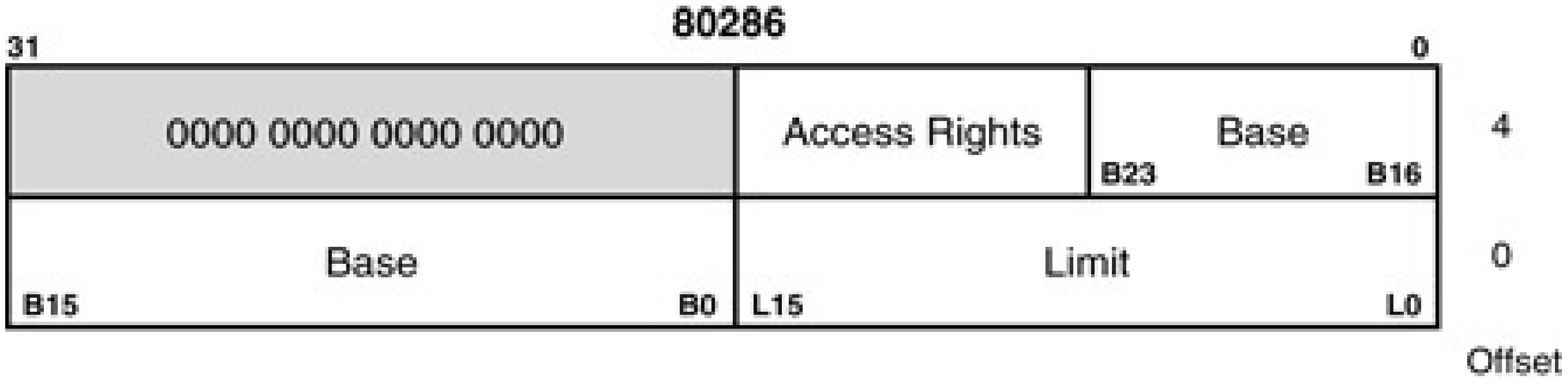
386 modes of operation

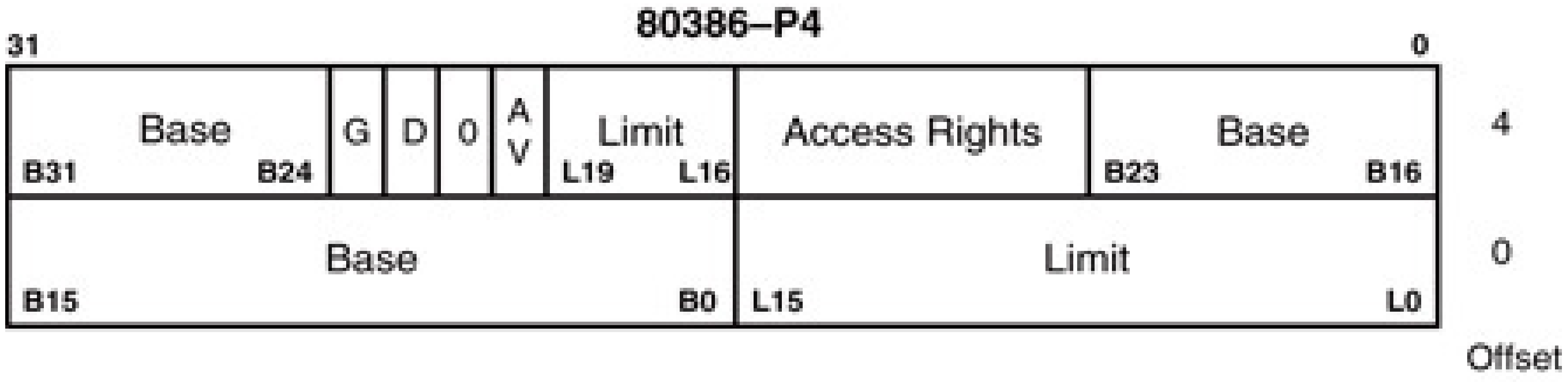
- There are 3 modes of operations:
  - **Real mode**
    - we have already discussed it
  - **Protected mode**
    - we have already discussed it --- same as 286. Only difference is in descriptor description
  - **virtual 8086 mode**

Protected mode

- same as 286
- Only difference is in
  - ✓descriptor description
  - ✓optional use of ***page***
- If the **Paging Unit(PU) is disabled**, then linear address produced by segment unit is used as **physical address,**
- Otherwise, the **PU** converts the linear address into page address.
- The paging mechanism allows handling of large *segments of memory in terms of* **pages of 4Kbyte size.**

Descriptor


80286
31
0
0000 0000 0000 0000
Access Rights
Base
B23
B16
4
Base
B15
B0
Limit
L15
L0
0
Offset


80386–P4
31
0
Base
B31
B24
G
D
0
A
V
Limit
L19
L16
Access Rights
Base
B23
B16
4
Base
B15
B0
Limit
L15
L0
0
Offset

Granularity (G) bit

- *When G=0, segments can be 1 byte to 1MB in length.*
- *When G=1, segments can be 4KB to 4GB in length.*

Default Operation Size (D-bit)

- when this bit is cleared, D = 0, operands contained within this segment are assumed to be 16 bits in size.

- When it is set, D = 1, operands are assumed to be 32-bits.

Example of D bit uses

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

- 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)

O bit(Reserved by Intel)

- It neither can be defined nor can be used by user.
- Bit must be zero (0) for compatibility with future processors

Available (Av) bit

- The AVL (available) field specifies whether the descriptor is available for user or it is for use by operating system
- Av=0 not available for user, used by OS
- Av=1 available for user

3. Virtual 86 mode

- In its protected mode of operation, 386 provides a virtual 8086 operating environment to execute the 8086 programs.
- The real mode can also used to execute the 8086 programs along with the capabilities of 386, like protection and a few additional instructions.

Protection Levels in 80386

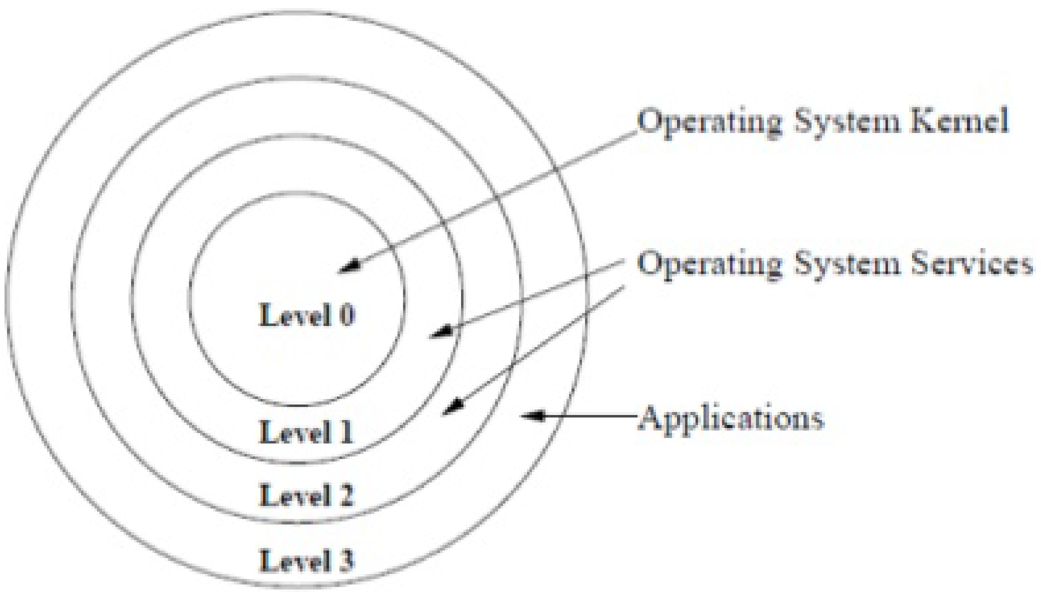
- The 80386 supports **four protection** which isolates and protect user programs for each other and the operating system during a multi-tasking operating system.
- The protection level avoid the use of privileged instructions, I/O instructions and access to segments and segment descriptors.
- The privilege protection mechanism of 80386 is integrated in On- chip **Memory Management Unit** which also gives protection to pages, when paging is enable.

Privilege protection

- 80386 protection mechanism
  - Memory management
  - Privilege protection
- 4 privilege level protection
  - PL0 (highest)
  - PL1
  - PL2
  - PL3(lowest)
- A numerically Smaller PL means a Higher privilege.

00	Kernel Level
01	OS Services
10	OS Extension
11	Application Level

Protection Rings



Virtual 86 mode...

- The address forming mechanism in virtual 8086 mode is exactly identical with that of 8086 real mode.
- In virtual mode, 8086 can address 1Mbytes of physical memory that may be anywhere in the 4Gbytes address space of the protected mode of 386.

Virtual 86 mode...

- Like 386 real mode, the addresses in virtual 8086 mode lie within 1Mbytes of memory.

- In virtual mode, the paging mechanism and protection capabilities are available at the service of the programmers.

- The 386 supports multiprogramming, hence more than one programmer may be use the CPU at a time.

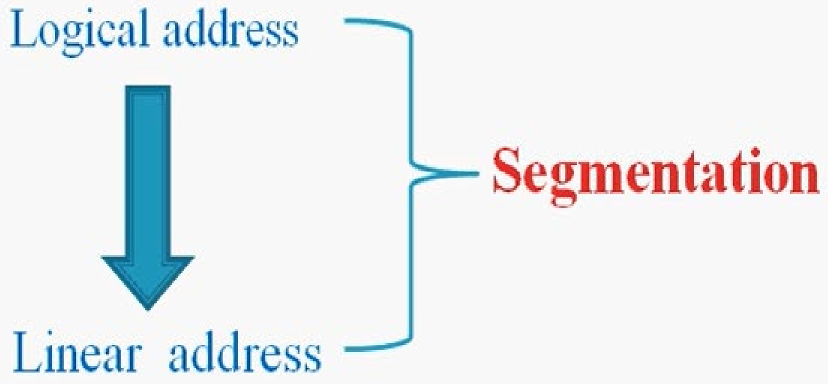
Paging

Paging

- The physical memory is divided into equal sized block called as **page** and the process is called as **paging**.

- In paging, address generated by segment address translation is not valid .*Linear address generated by segment address translation is converted into **physical address*** . The process is called as **“Page translation”.**

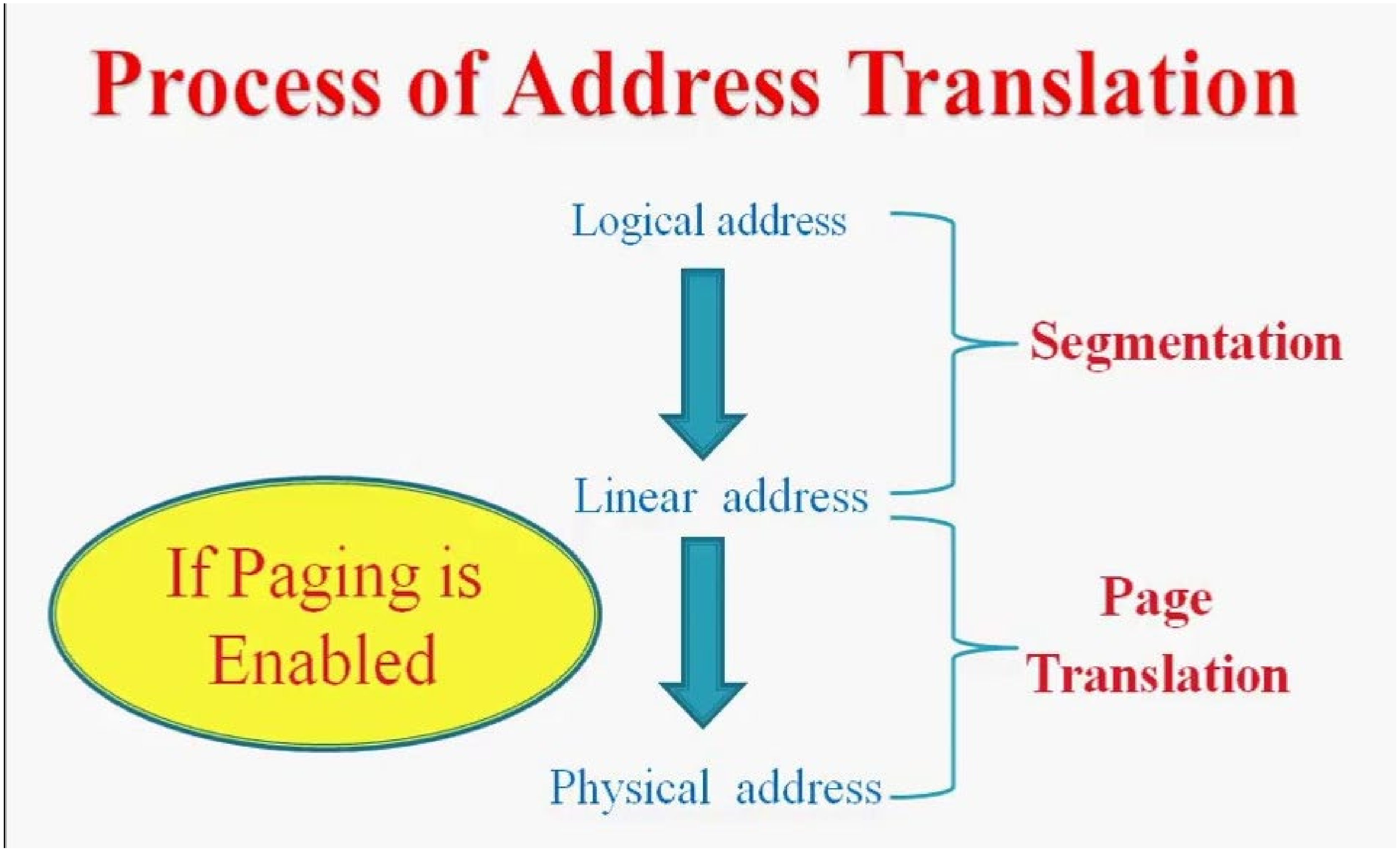
Process of Address Translation



If Paging not Enabled

Linear Address = Physical Address

Why Paging ?

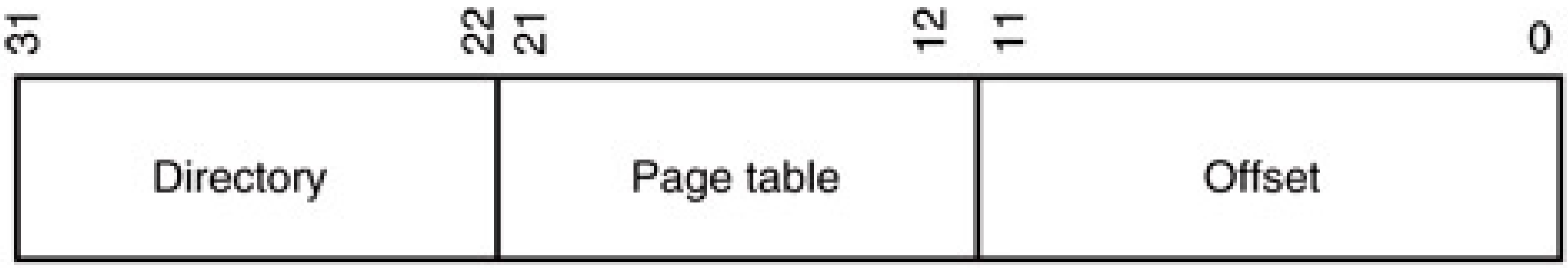

Process of Address Translation
Logical address
Linear address
Physical address
Segmentation
Page Translation
If Paging is Enabled

Why Paging ?

- The protected mode segmentation and virtual memory scheme of 286 and 386 are essentially the same,
- Only difference is 386 segment can be as large as 4G, instead of 64K as in 286
- Swapping in and out of 4G to and from physical memory requires too long time, so paging mechanism was developed.
- Thus, swapping 4K pages is faster.

How Paging mechanism works?

- The components of paging mechanism :
  - ✓ Page Directory
  - ✓ Page Table
  - ✓ Page Frame or offset (4KB page)
- The 32-bit linear address, as generated by software, is broken into three sections that are used to access the operand in the memory



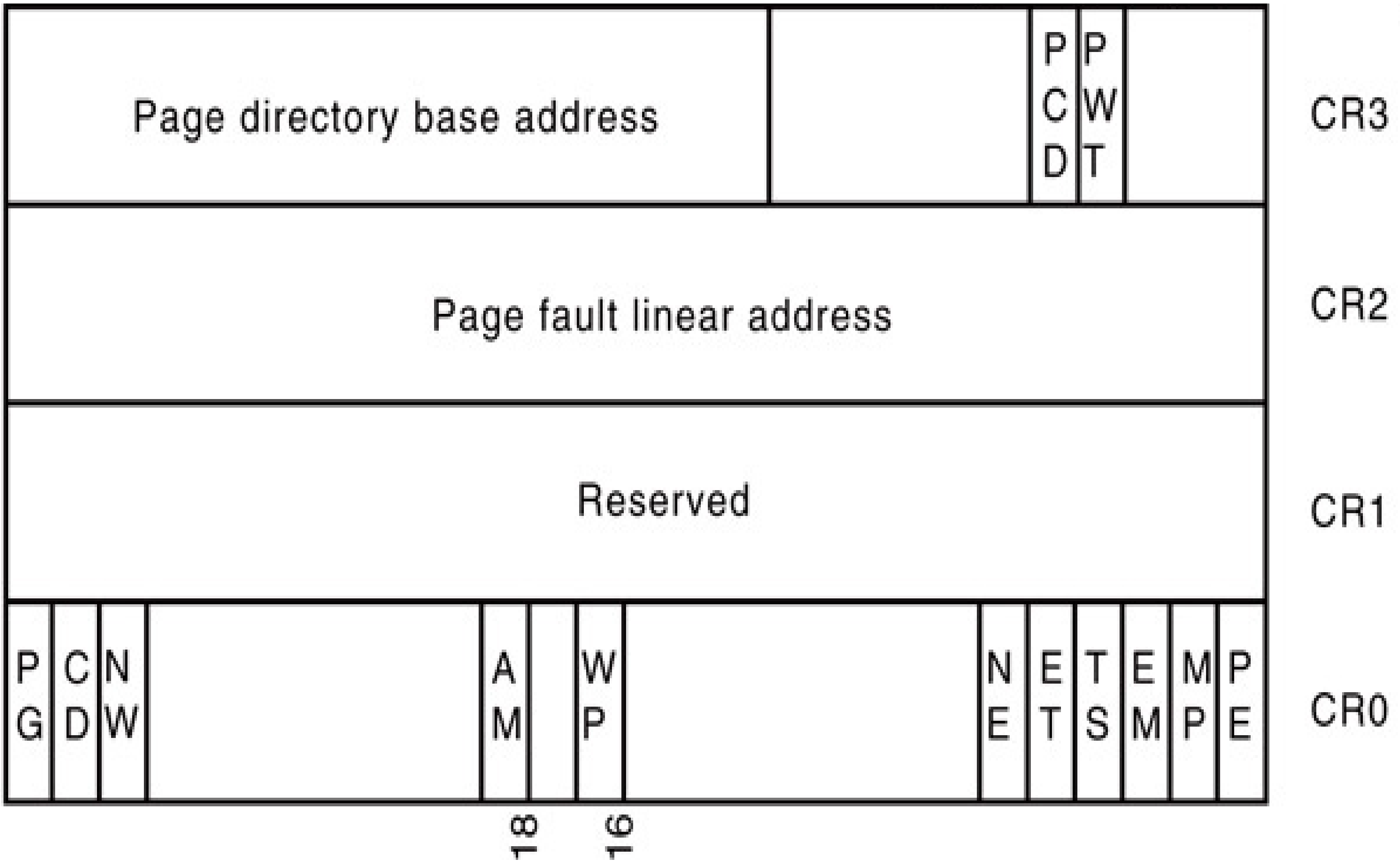
How Paging mechanism works?

- ❑**4G memory** is *divided into **1M pages of 4K***.
- ❑The *starting addresses of these 1M pages* are kept in **two-level tables**.
  - ✓First level table, known as ***Page Directory***, contains 1K entries(0—1023) for the second level table, known as ***Page Tables***
  - ✓Each ***Page Table*** contains the ***starting addresses*** of 1K pages(0---1023)

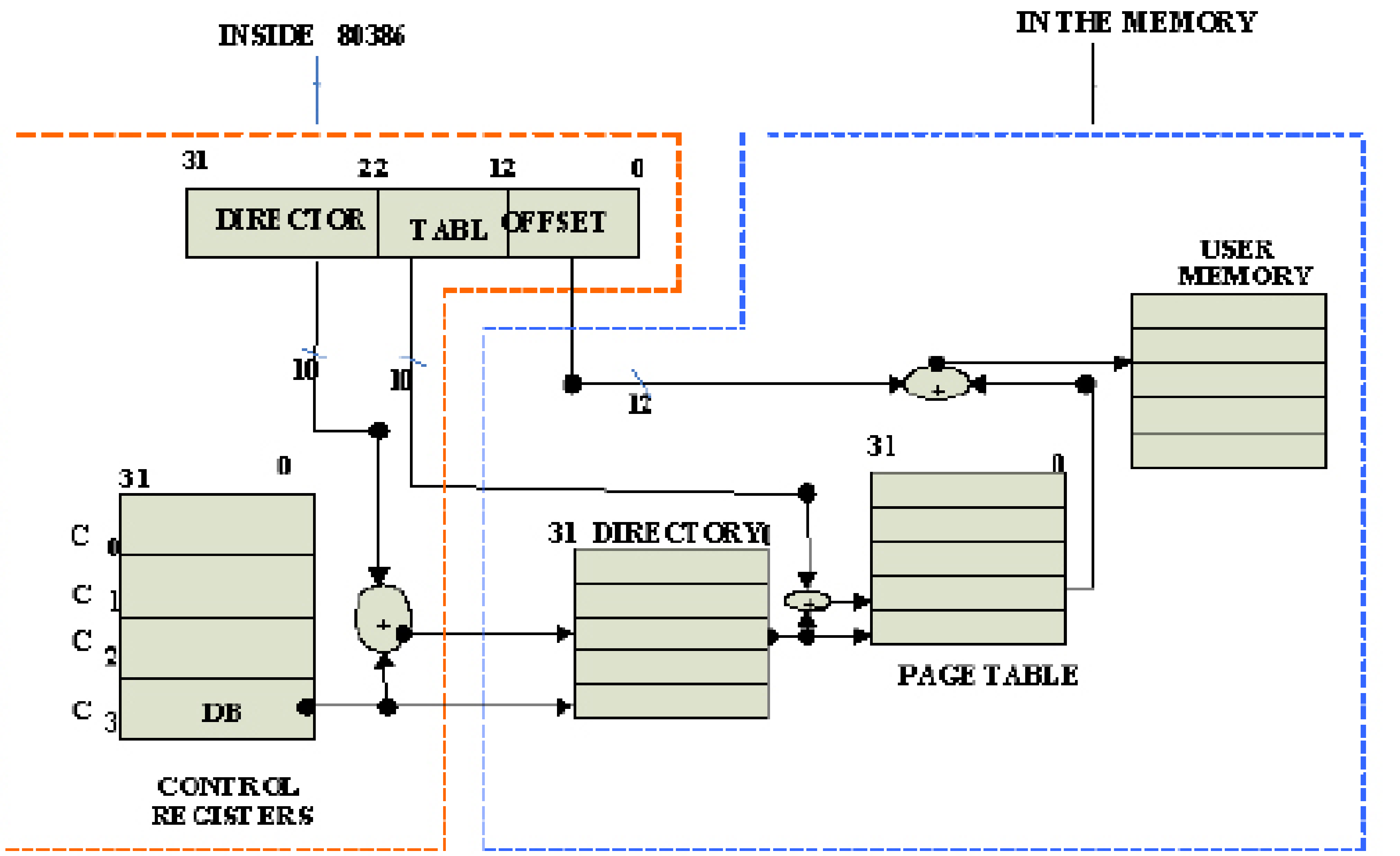
How Paging mechanism works?...
Page Directory

- There is only one ***Page Directory*** in the system
- Starting address of ***Page Directory*** is in CR3
- The paging unit is controlled by the contents of the microprocessor’s Control Registers(CR).
- 386 has ***4*** *<u>control registers</u>* **CR0 through CR3**.
- *<u>CR1 of 386</u>* are ***<u>kept reserved by Intel.</u>***

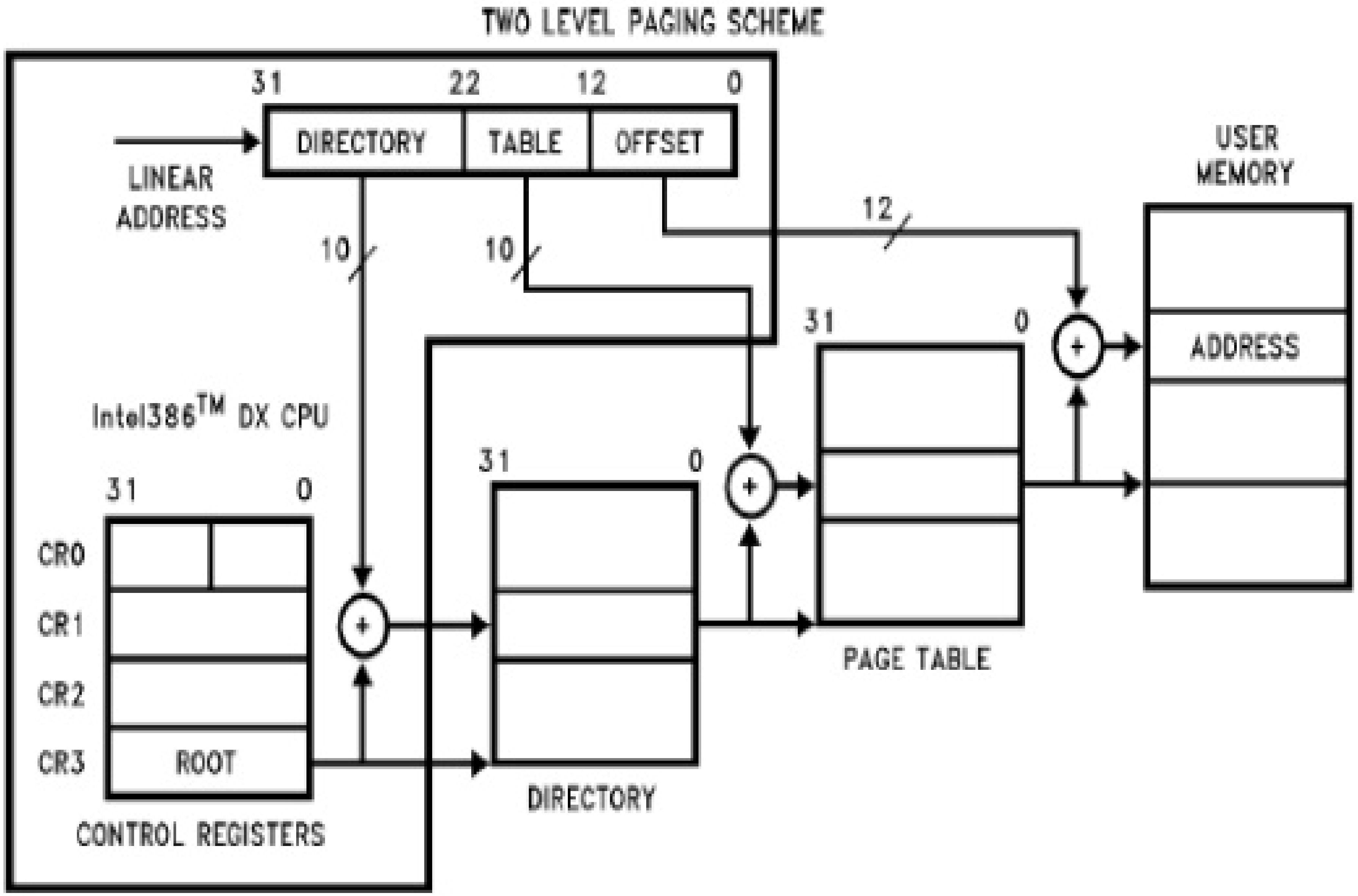
386 control registers



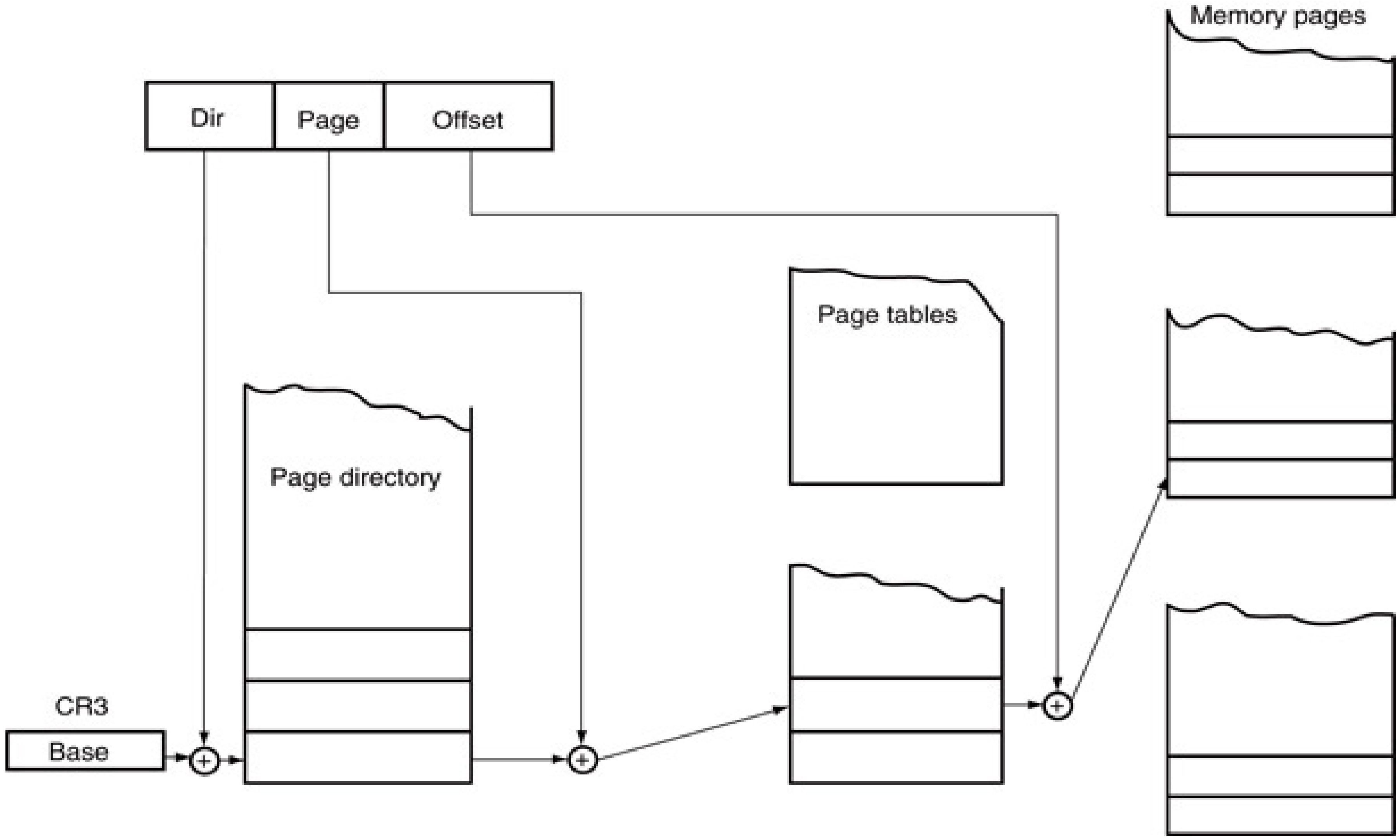
The paging mechanism of 386



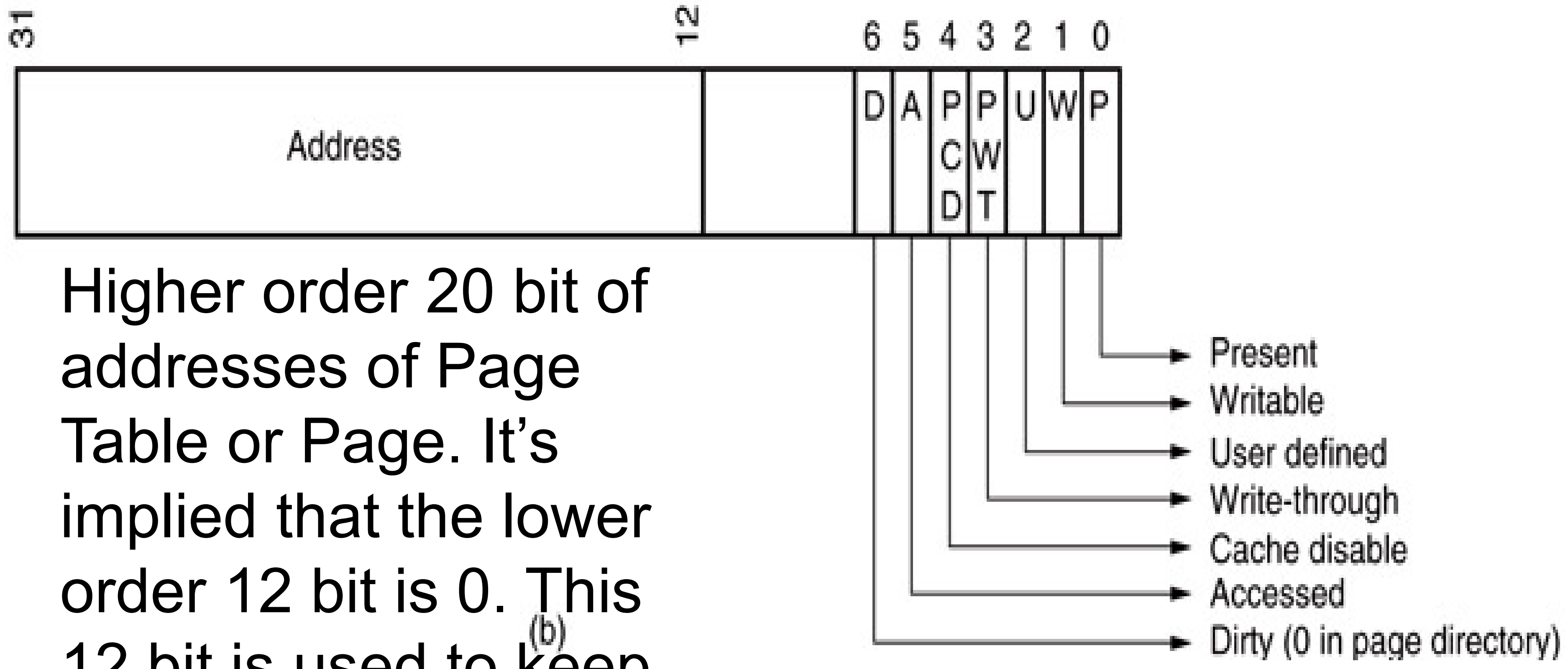
The paging mechanism of 386



The paging mechanism of 386



Page Directory or Page Table entry



Higher order 20 bit of addresses of Page Table or Page. It's implied that the lower order 12 bit is 0. This 12 bit is used to keep other info of the entry

P bit

- The P bit of the entries indicate, whether the entry (**page table/page**) is **present** in the memory
  - If P=1, the entry is in the memory
  - If P=0, the entry is not in the memory
- The P bit of the currently executed page is always high.

R/W and U/S bits

- The **Read/Write (R/W) bit** allows a page to be marked as read only or read/write
- **U/S bit** is used to specify one of two privilege levels, **user or supervisor**
  - **0** specifies an user level or **lowest level** of privilege
  - **1** specifies a supervisor level or **higher level** privilege
- **User/Supervisor (U/S) and R/W bits** can be used in place of or in addition to the protection mechanism.

D and A bits

- The **D bit (Dirty bit)** is set *before a write operation to the page is carried out.*
  - The D-bit is undefined for page directory entries.

- The **accessed bit A** is set by 80386 before any access to the page.
- The OS can periodically check and reset this bit to determine how often the page is being used

Translation Look-aside Buffer (TLB)

- As said before, *Page Directory and Page Tables are located in memory*

- *To **avoid** having to **read memory twice*** in order to get access to *required memory operand*, 386 maintains a special ***cache*** called ***TLB***

- ***TLB*** is a ***four-way set-associative cache*** that *holds the page addresses* of *32 most recently used pages*

Overhead for paging

- ***Directory Table*** occupies 4KB
  - 1K entry X 4 bytes per entry = 4KB
- ***Page Tables*** occupy 4MB
  - 1M page entry X 4 bytes per entry = 4MB
- This represents a considerable amounts of memory

Instruction Set

- Some new instructions are there,
  - But we will not study those

Interrupt

- Some new interrupts are added in 386
- Type 0 to Type 4 --- used in 8086
- Type 5 to Type 7 --- introduced during 186
- Type 8 to Type 14 --- new in 386

Interrupt …

- **Type 8 --- Double fault**
  - activated whenever two separate interrupts occur during the same instruction
- **Type 9 --- co-processor segment overrun**
  - Occurs if co-processor op-code memory operand extends beyond offset address
- **Type 10 --- Invalid task state segment**
  - Occurs if the TSS is invalid. Mostly this is caused because TSS is not initialized.

Interrupt …

- **Type 11 --- segment not present**
  - Occurs when P-bit in a descriptor indicates that the segment is invalid
- **Type 12 ---stack fault**
  - Occurs if the stack segment is not present (P=0) or if the limit is exceeded
- **Type 13 --- general protection fault**
  - Occurs for protection violation
- **Type 14 --- page fault**
  - Occurs for any page fault memory or code access.

80386 summary

- This 80386 is a 32bit processor that supports, 8bit/32bit data operands.
- The 80386 instruction set is upward compatible with all its predecessors.
- The 80386 can run 8086 applications under protected mode in its virtual 8086 mode of operation.
- With the 32 bit address bus, the 80386 can address up to 4Gbytes of physical memory. The physical memory is organized in terms of segments of 4Gbytes at maximum.
- The 80386 CPU supports 16K number of segments and thus the total virtual space of 4Gbytes * 16K = 64 Terabytes.

80386 summary

- The memory management section of 80386 supports the virtual memory, paging and four levels of protection, maintaining full compatibility with 80286.
- The 80386 offers a set of 8 debug registers DR0-DR7 for hardware debugging and control. The 80386 has on-chip address translation cache.
- The concept of paging is introduced in 80386 that enables it to organize the available physical memory in terms of pages of size 4Kbytes each, under the segmented memory.
- The 80386 can be supported by 80387 for mathematical data processing.