MICROPROCESSOR

CSC 405

Subject Incharge

Dakshata Panchal

Assistant Professor

email: dakshatapanchal@sfit.ac.in

CSC 405 Microprocessor

Module 4

Intel 80386DX Processor

Contents as per syllabus

- Architecture of 80386 microprocessor
- 80386 registers – General purpose Registers, EFLAGS and Control registers
- Real mode, Protected mode, virtual 8086 mode
- 80386 memory management in Protected Mode – Descriptors and selectors, descriptor tables, the memory paging mechanism

Introduction

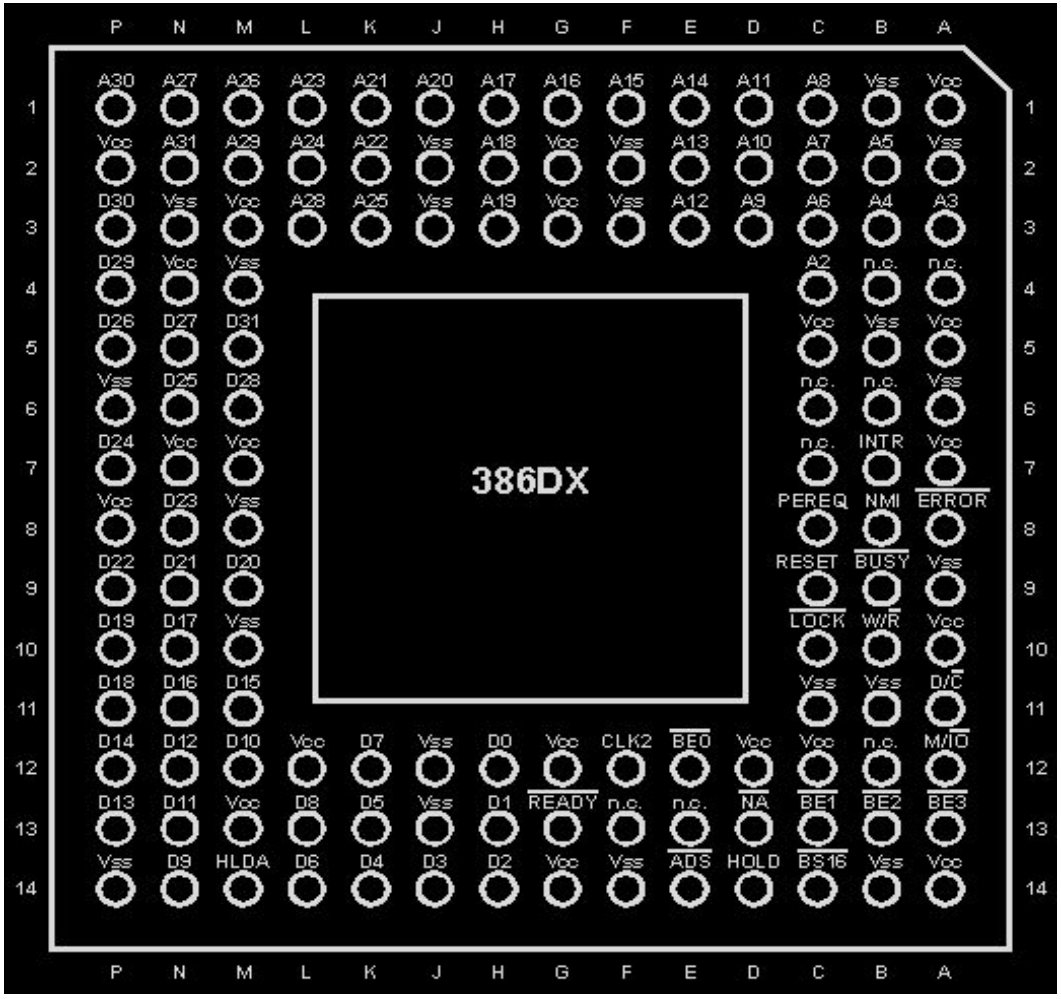
Intel 80386 Processor

- The first 32-bit processor
- Launched in Oct 1985, but was available in market only by Sep 1986.
- Also known as i386 or just 386
- The first version had 275,000 transistors, and worked at a speed of 20 MHz
- It is a 132 pins IC in PGA (Pin Grid Array) package.

Introduction

Intel 80386 Processor

132 pins in PGA (Pin Grid Array)



Introduction

8086 vs 80286 vs 80386 Processor in terms of memory

- **8086**
  - Physical memory = logical memory = 1 MB
  - 20-bit address line = 2^{20} locations = 1 MB
- **80286**
  - Physical memory = 16 MB
  - 24-bit address line = 2^{24} locations = 16 MB
  - Logical memory = 1 GB!
- **80386**
  - Physical memory = 4 GB
  - 32-bit address line = 2^{32} locations = 4 GB
  - Logical memory = 64 TB!

Introduction

What's New in 80386?

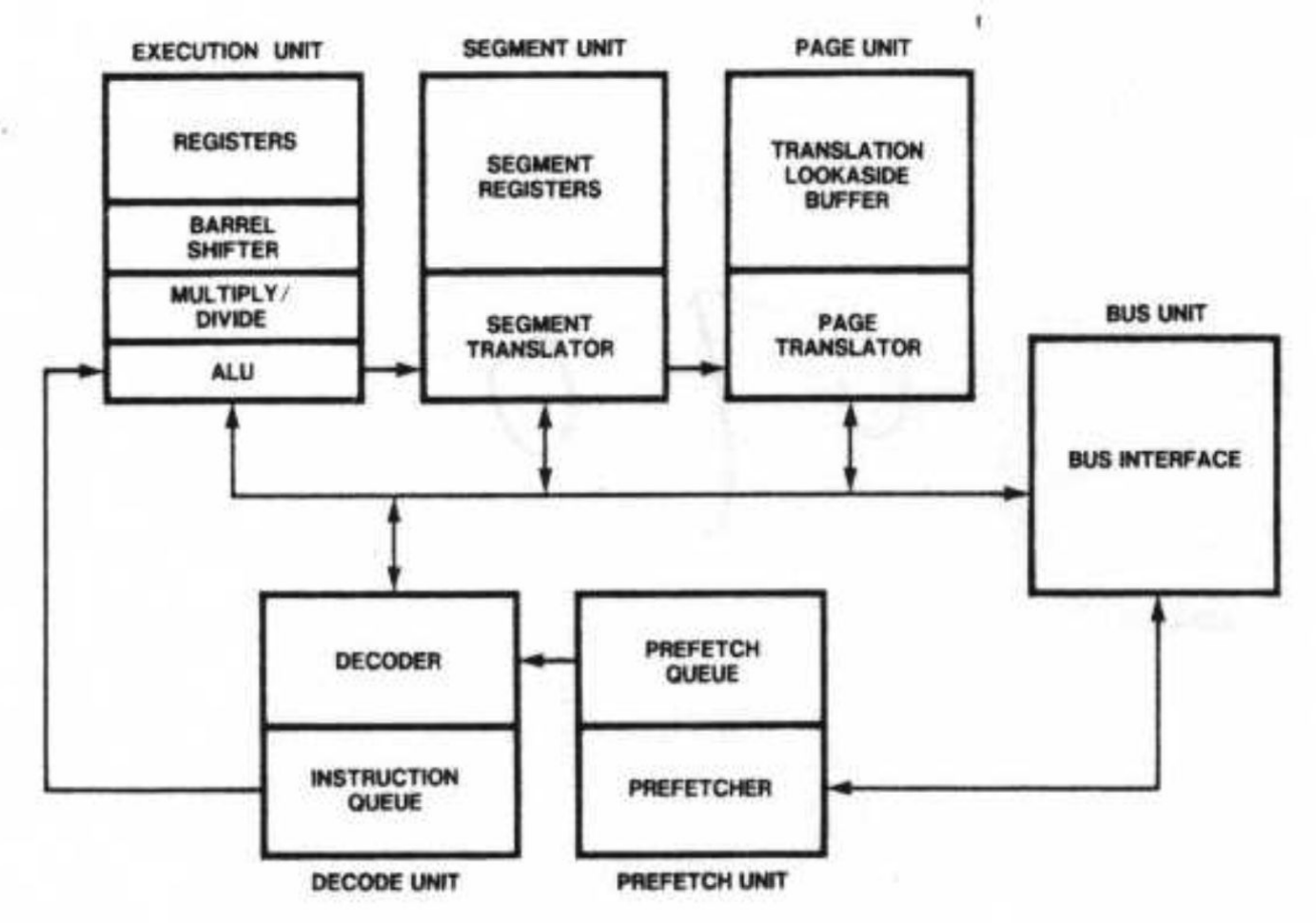
- 132 pins in PGA package
  - 40 pins in DIP package
- 32-bit processor
  - 8086 is 16-bit µP
- Can access 4 GB of physical addresses
  - 8086 can access 1 MB locations
- Allows multitasking
  - 8086 can load only one program at a time.
- Concept of virtual memory
  - 8086 does not have virtual memory

Introduction

Salient Features of 80386 Processor

1) It is a **32 bit Microprocessor**.
2) It has a **32 bit data bus**.
3) It has **4 memory banks**.
4) It has a **32 bit address bus**.
5) It can access **4 GB memory**.
6) It has **3 Pipeline stages**.
   The Pipeline stages are called: **Fetch, Decode, Execute**.
7) It operates on **16 MHz – 33 MHz** frequency.
8) It has **2,75,000 transistors**.
9) It was released in the year **1985**

Architecture of 80386



Architecture of 80386

80386 architecture is divided into 5 independent units.

1. Bus Unit (Bus Interface Unit)
2. Prefetch Unit
3. Decode Unit
4. Execution Unit
5. Memory Unit

Architecture of 80386

1. Bus Unit (Bus Interface Unit)

1) The Bus unit is responsible for **transferring data in and out of the µP.**
2) It is connected to the external memory and I/O devices, using the system bus.
3) It gets requests from Prefetch unit for fetching instructions and from execution unit for transferring data.
4) If both requests occur simultaneously preference is given to execution unit.

Architecture of 80386

2. Prefetch Unit

1) The Pre-fetch unit fetches further instructions in advance to implement pipelining.
2) It fetches the next 16 bytes of the program and stores it into the Prefetch Queue.
3) It refills the queue when at least 4 bytes are empty as 80386 has a 32 bit data bus.
4) During a branch, the instructions in the queue are invalid and hence are discarded

Architecture of 80386

3. Decode Unit

1) 80386 µP has a separate unit for decoding instructions called the Decode Unit.
2) It decodes the next three instructions and keeps them ready in the Decode Queue.
3) The decoded instructions are stored in Micro-Coded form.
4) During a branch, the instructions in the queue are invalid and hence are discarded.

Architecture of 80386

4. Execution Unit

1) Execution Unit performs the main task of executing instructions.
2) Normally, execution requires Arithmetic or Logic operations performed by a 32-bit ALU.
3) It also has dedicated circuits for 32-bit multiplication & division.
4) A 64-bit barrel shifter is also provided for faster shifts during multiplication and division.
5) Operands for the ALU can either be provided in the instruction, or can be taken from memory or could be taken from the 32-bit registers like EAX, EBX etc.
6) Additionally there is a 32-bit Flag register (EFLAGS) giving the Status of the current result.

Architecture of 80386

5. Memory Unit

1) The Memory unit converts Virtual Address (Logical address) to Physical Address.

2) 80386 µP implements 64 Terra bytes of Virtual memory using Segmentation and Paging.

Hence the Memory Unit is sub-divided into Segmentation Unit and Paging Unit.

3) Segmentation is compulsory, while Paging is optional.

4) The Segmentation Unit converts the Logical Address into a Linear Address.

5) The Paging Unit converts the Linear Address into a Physical Address.

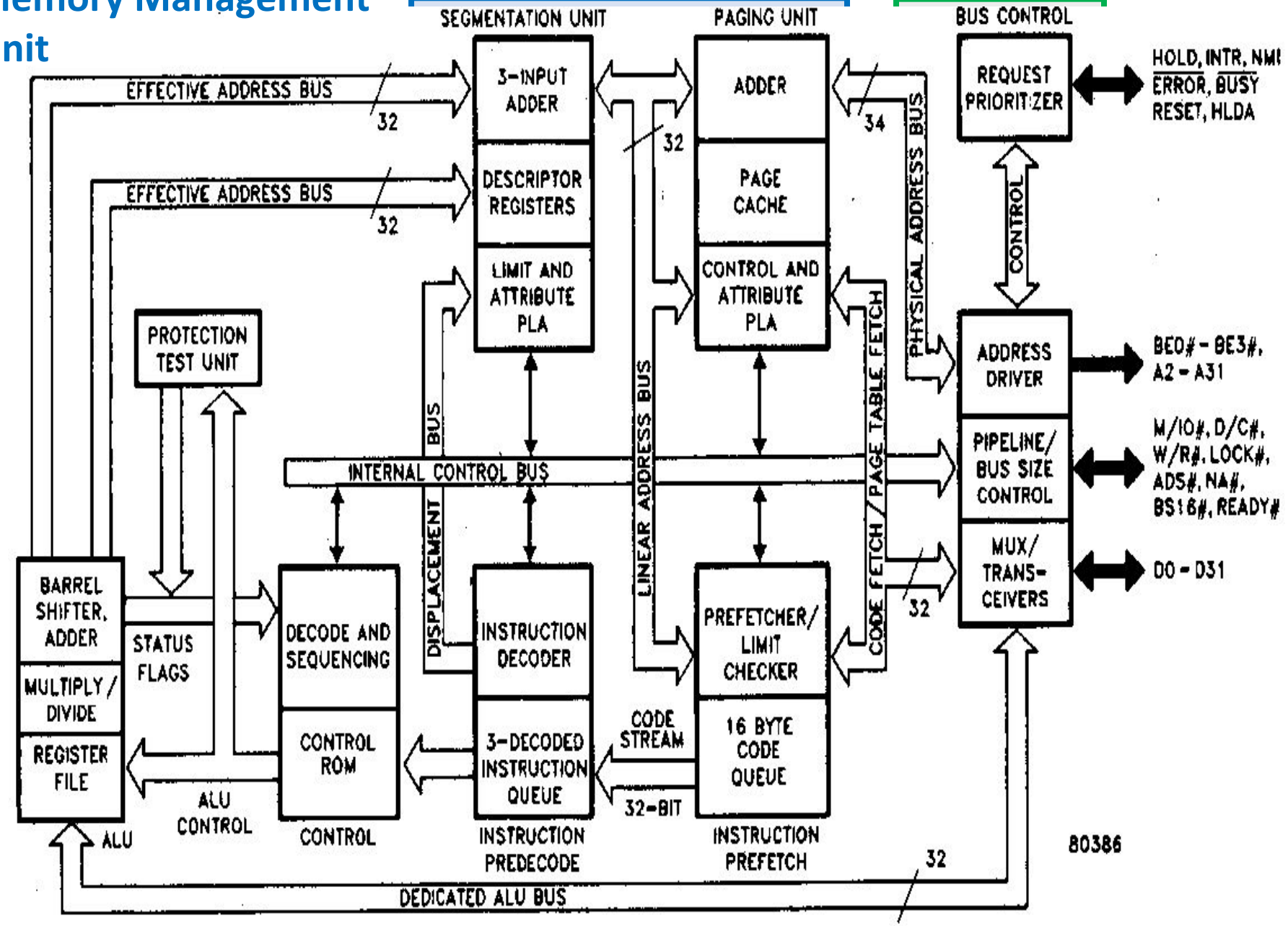
6) If Paging is not used, then the Linear Address itself is the Physical Address.

Detailed Architecture of 80386

The internal architecture of 80386 is divided into three sections:

1. **Central Processing Unit (CPU)**
2. **Memory Management Unit (MMU)**
3. **Bus Interface unit (BIU)**

Memory Management Unit



Central Processing Unit

Detailed Architecture of 80386

Central Processing Unit

- The CPU is further divided into:
  - Instruction Unit
  - Execution Unit
- **Instruction Unit:**
  - It decodes the opcode from 16-byte instruction queue and arranges them into a 3-decoded instruction queue.
  - After decoding , control section derives necessary control signals

Central Processing Unit

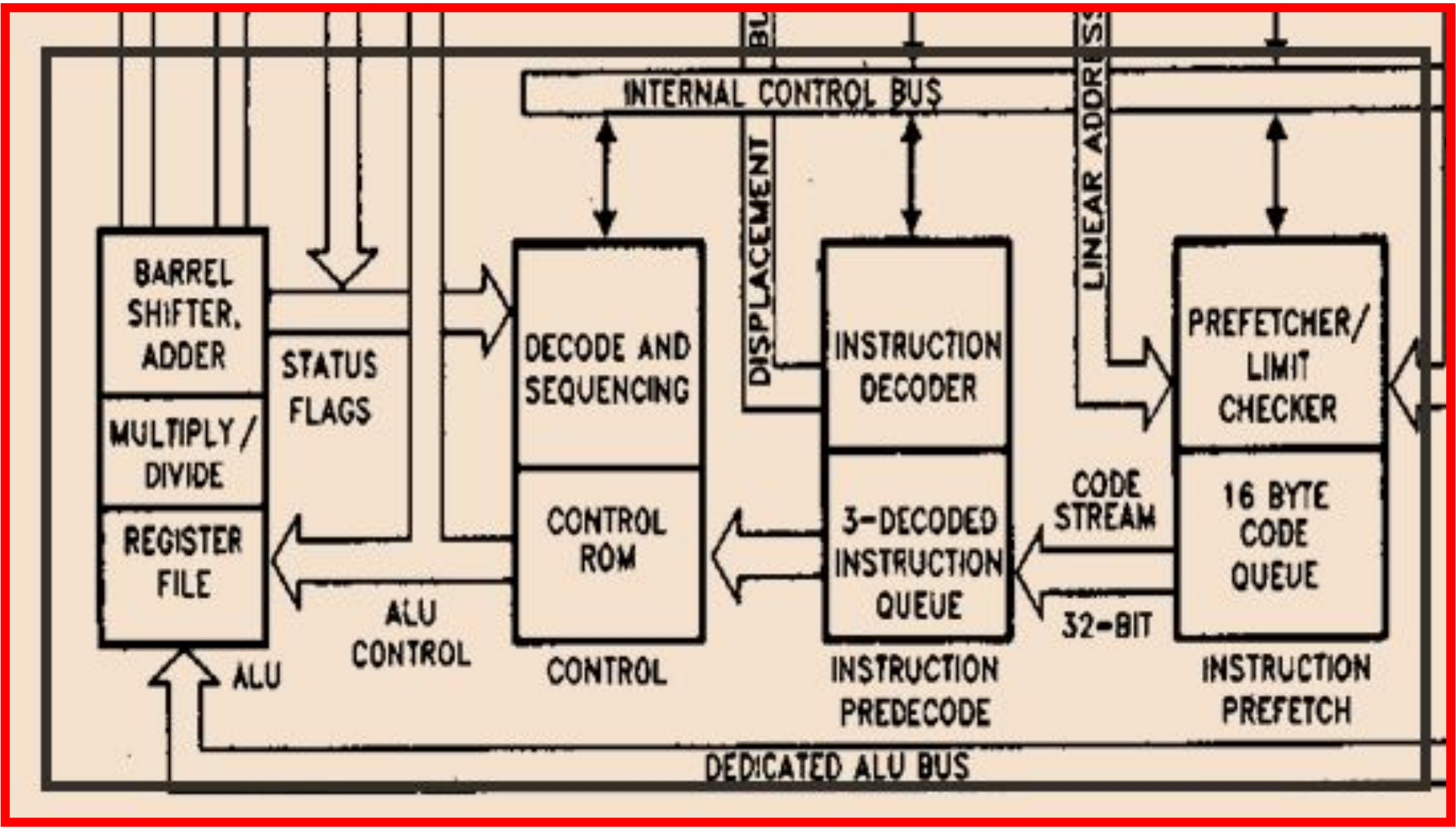
- **Execution Unit:**
  - It has **8 general purpose** and **8 special purpose registers** which handles data or offset addresses.
  - The 64-bit **barrel shifter** do shift, rotate, multiply and divide operations
  - The multiply/divide logic implements the bit-shift-rotate algorithms to reduce time(Even 32bit multiplication is done in 1µs)

Detailed Architecture of 80386

Central Processing Unit

- **Elements of EU**
  - Arithmetic/logic unit (ALU)
    - Performs the operation identified by ADD, SUB, AND, etc.
  - Flags register
    - Holds status and control information
  - General-purpose registers
    - Holds address or data information
  - Control ROM
    - Contains microcode sequences that define operations performed by machine instructions
  - Special multiply, shift, and barrel shift hardware
    - Accelerate multiply, divide, and rotate operations

Detailed Architecture of 80386



Memory Management Unit

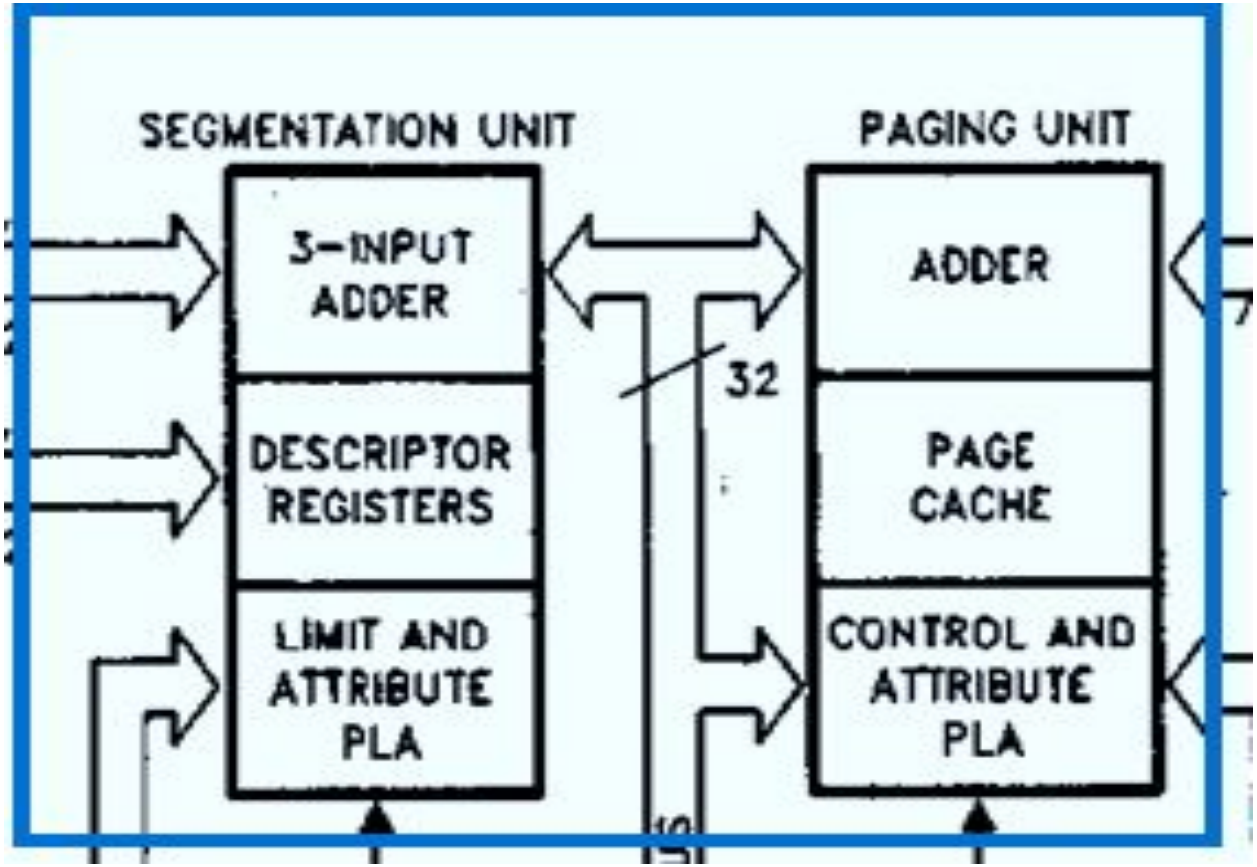
- MMU consists of a segmentation unit and paging unit
- **Segmentation Unit:**
  - It uses two address components - **segment &offset** – for relocability and sharing of code and data
  - It allows a **maximum segment size of 4GB**
  - It provides a 4-level protection mechanism
  - The limit and attribute PLA checks segment limits and attributes at segment level to avoid invalid accesses.

Memory Management Unit

- **Paging Unit**
  - It organizes physical memory in terms of pages of 4KB size
  - It works under the control of segmentation unit
  - It converts linear addresses into physical addresses
  - The control and attribute PLA checks privileges at page level.

Detailed Architecture of 80386

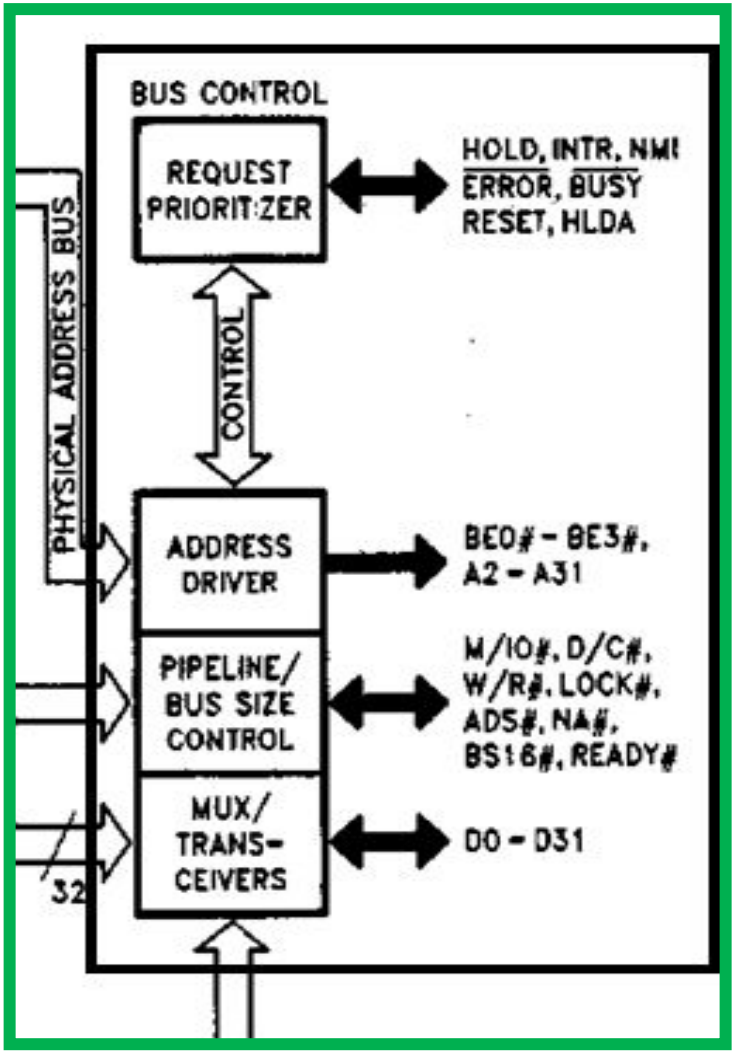
Memory Management Unit



Detailed Architecture of 80386

Bus Interface Unit

- It has a **Prioritizer** to resolve the priority of various bus requests.
- **Address driver** drives the byte enable and address signals A_2 – A_{31}.
- **Pipeline**/bus size unit handles the control signals for pipelining and dynamic bus sizing units
- **Data buffers** interface the internal data bus with system bus



80386 Register Overview

- The Intel386 DX has 32 register resources in the following categories:
  - General Purpose Registers
  - Segment Registers
  - Instruction Pointer and Flags
  - Control Registers
  - System Address Registers
  - Debug Registers
  - Test Registers

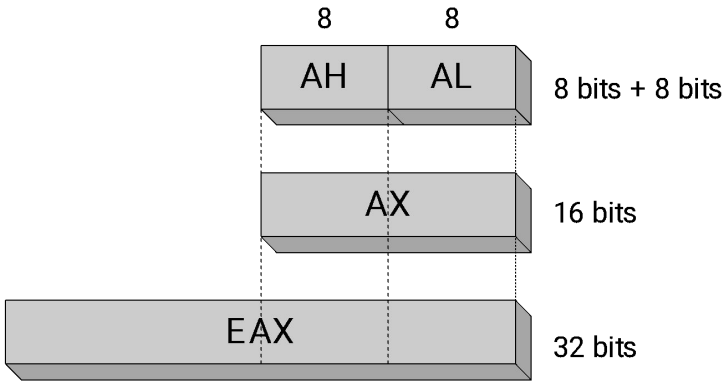
80386 registers – General purpose Registers

- hold data or address values.
- support data of 8, 16, 32 and 64 bits.
- 32-bit registers : EAX, EBX, ECX, EDX, ESI, EDI, EBP, and ESP.
- The least significant 16 bits can be accessed as in 8086 as AX, BX, CX, DX, SI, DI, BP, and SP.

80386 registers – General purpose Registers

- When accessed as a 16-bit operand, the upper 16 bits are neither used nor changed.
- 8-bit operations can be performed with AL, BL, CL and DL.
- The higher bytes are AH, BH, CH and DH
- The individual byte accessibility offers flexibility for data operations.

80386 registers – General purpose Registers



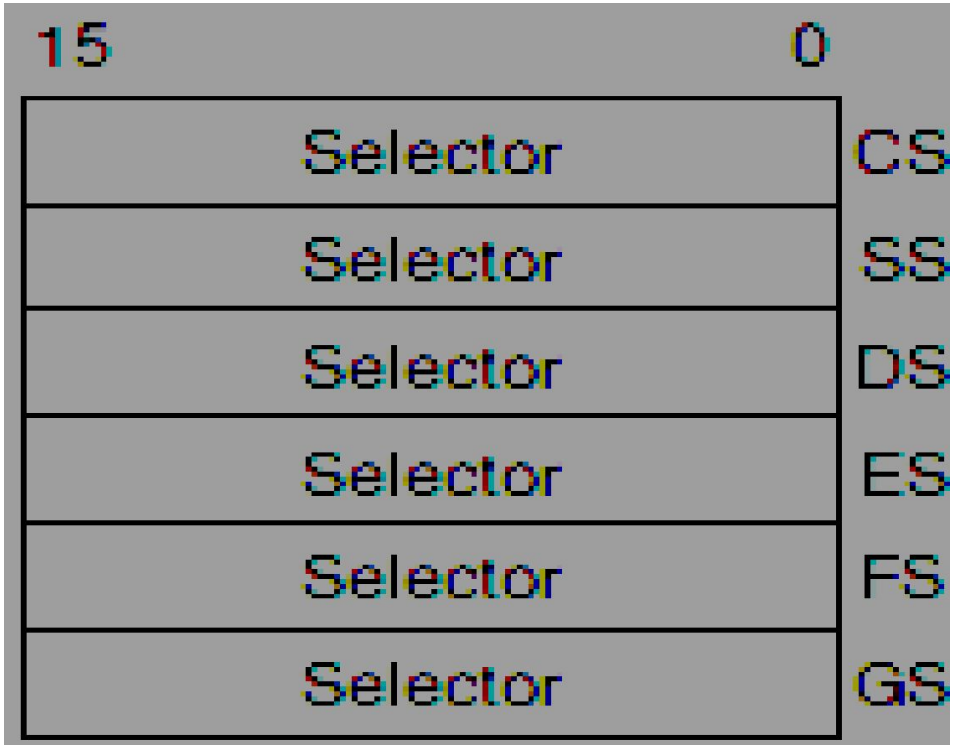
32-bit	16-bit	8-bit (high)	8-bit (low)
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL

32-bit	16-bit
ESI	SI
EDI	DI
EBP	BP
ESP	SP

80386 registers – Segment Registers

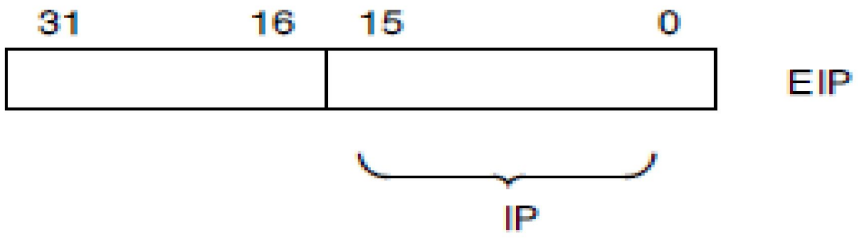
- The segment registers
  - CS indicates the current code segment
  - SS indicates the current stack segment
  - DS, ES, FS and GS indicate four current data segments.
- On any data reference the DS-pointed data segment is assumed by default.
- In order to access any **other data segment**, an **override directive** is used, e.g., FS:[1050]

80386 registers – Segment Registers



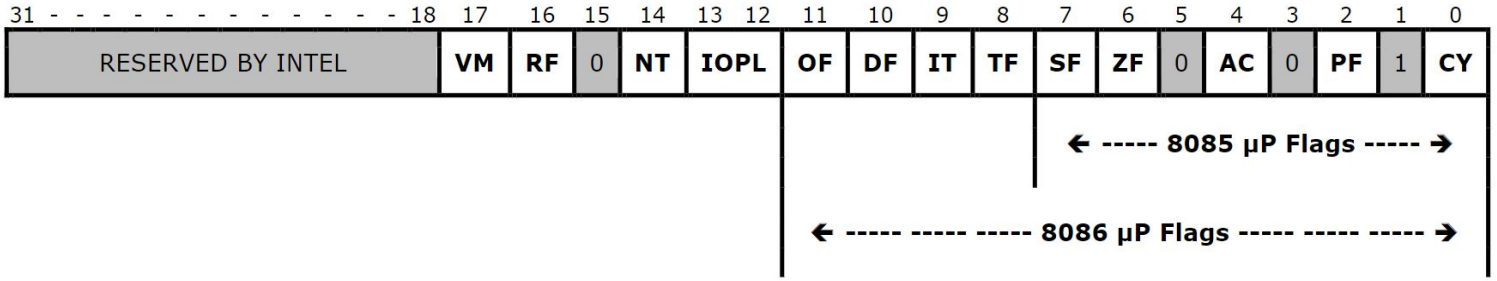
80386 registers – Instruction Pointer

- It is a 32-bit register named EIP.
- EIP holds offset of next instruction to be executed.
- The offset is always relative to code segment (CS).
- The lower 16 bits of EIP contain the 16-bit instruction pointer named IP, which is used by 16-bit addressing.



80386 registers – Flag Register

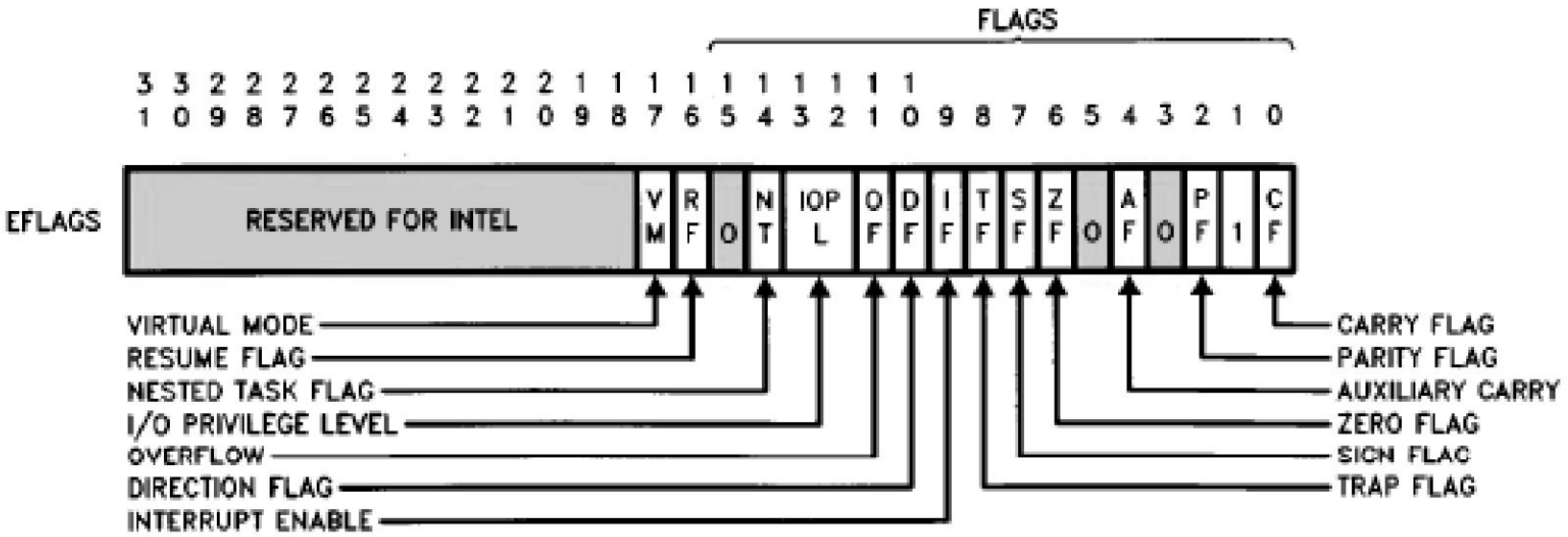
Flag Register of 80386 (EFLAGS – 32 bits)



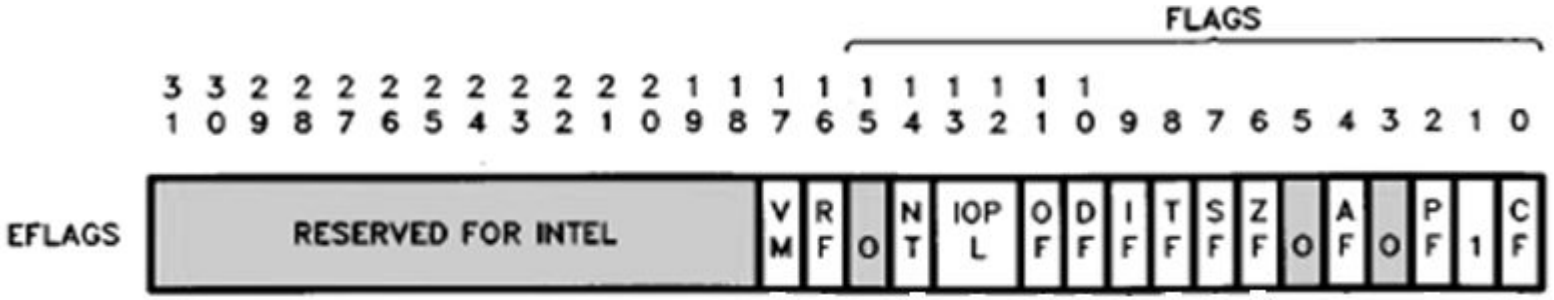
As seen from the above diagram, the lower 12 bits (11 ... 0) of EFLAGS are same as those in 8086. These are the only flags available when µP is in Real Mode.

The additional 5 flags are only available once µP enters Protected mode by making PE bit = 1, in CR0 register.

80386 registers – Flag Register

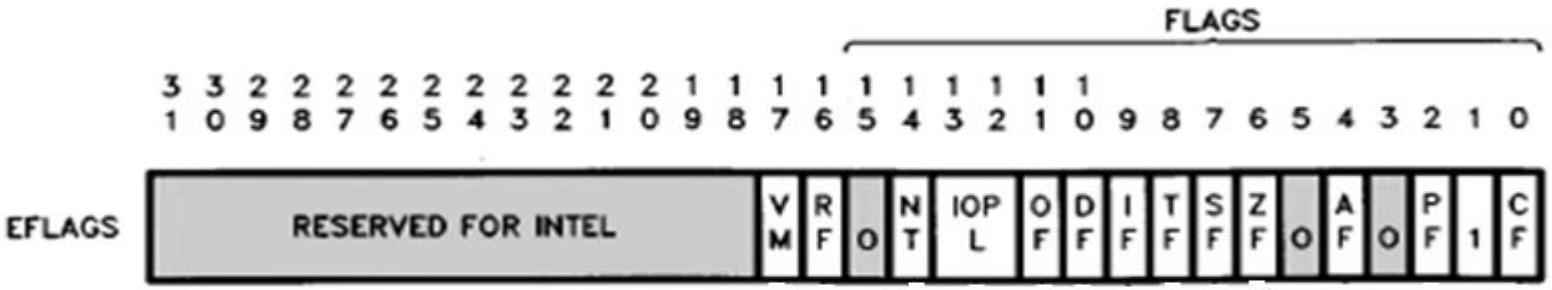


80386 registers – Flag Register



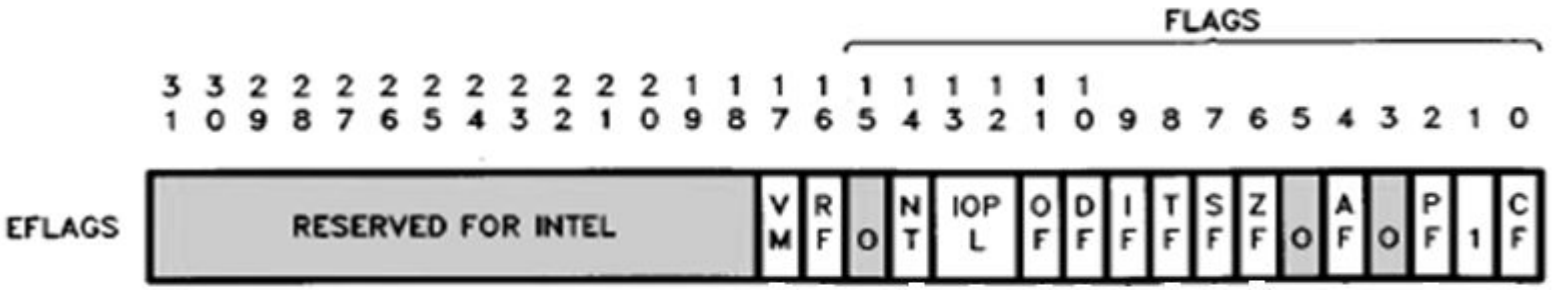
- **Bit 17 (VM Bit, Virtual Mode):**
  - VM bit is set to work in Virtual 8086 mode
- **Bit 16 (RF Bit, Resume Flag):**
  - RF flag is used with debug register breakpoints.
  - When RF is set, **debug fault need to be ignored** on the next instruction.
  - RF is then **automatically reset** after every instruction

80386 registers – Flag Register



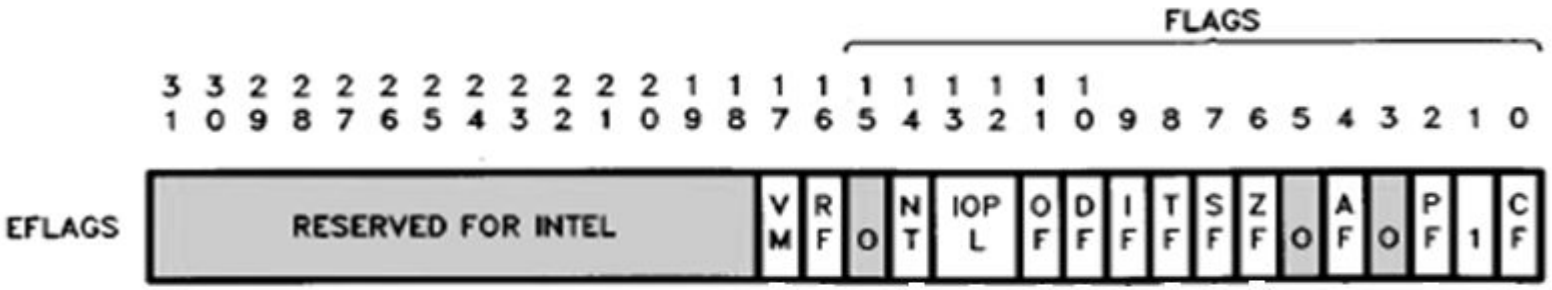
- **Bit 15 : Reserved**
- **Bit 14 (NT Bit, Nested Task):**
  - This flag applies to Protected Mode.
  - NT is set to indicate that the execution of this task is **nested within another task.**
  - If set, it indicates that the current nested task's Task State Segment (TSS) has a valid back link to the previous task's TSS.

80386 registers – Flag Register



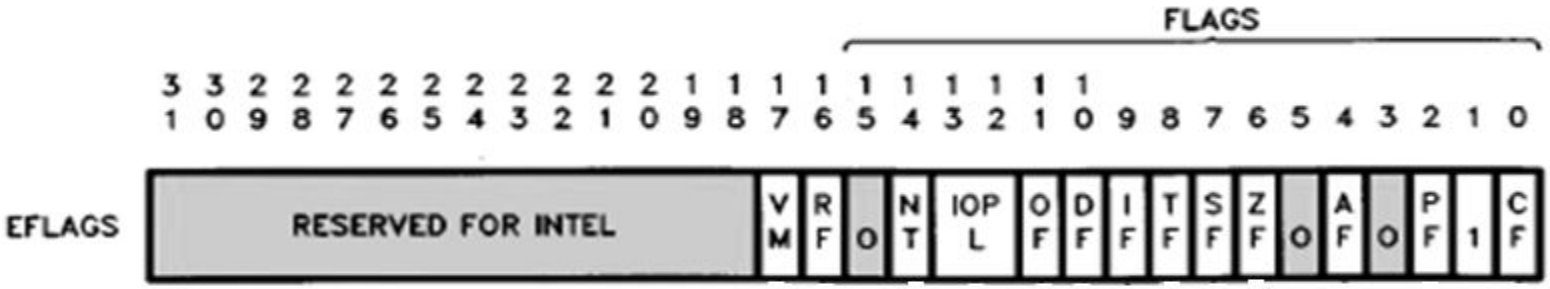
- **Bit 13,12 (IOPL Bit, Input/output Privilege):**
  - maximum CPL (current privilege level) value permitted to execute I/O instructions without generating an exception 13 fault or consulting the I/O Permission Bitmap.

80386 registers – Flag Register



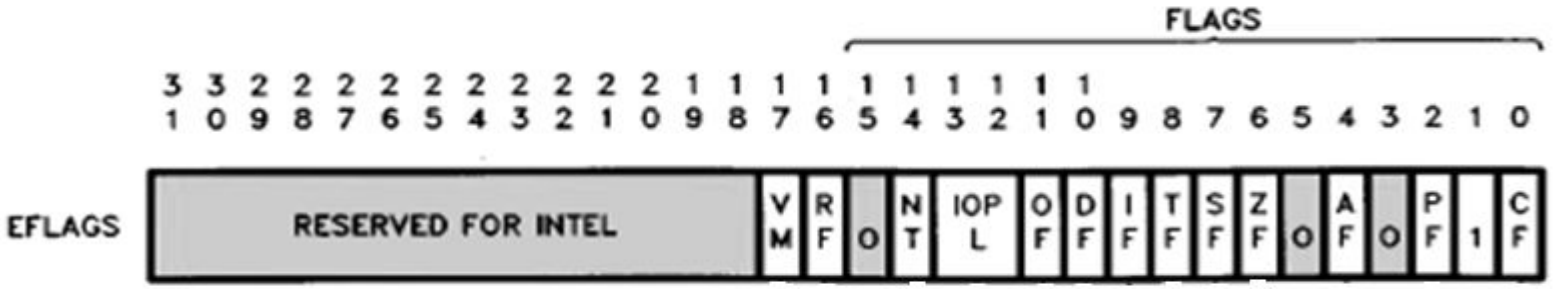
- **Bit 11 (OF Bit, Overflow Flag):**
  - OF is set if the operation resulted in a signed overflow.
- **Bit 10 (DF Bit, Direction Flag):**
  - DF defines whether ESI/EDI post-decrement or post-increment during the string instructions.
  - Post-increment occurs if DF is reset.
  - Post-decrement occurs if DF is set.

80386 registers – Flag Register



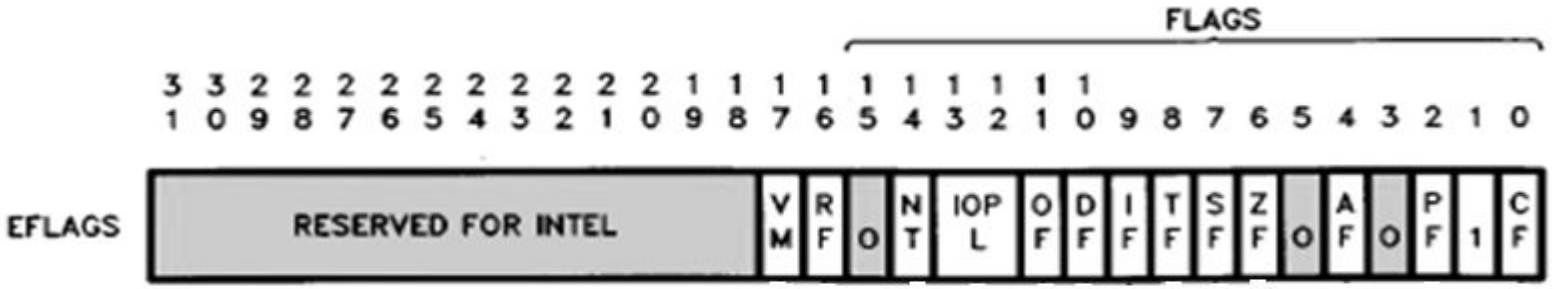
- **Bit 9 (IF Bit, Interrupt Enable Flag):**
  - When IF =1 the processor allows recognition of external interrupts on INTR pin
- **Bit 8 (TF Bit, Trap Enable Flag):**
  - When TF =1 the processor enables the single step mode for debugging.
- **Bit 7 (SF Bit, Sign Flag):**
  - SF is set if the high-order bit of the result is set, it is reset otherwise.

80386 registers – Flag Register



- **Bit 6 (ZF bit, Zero Flag):**
  - ZF is set if all bits of the result are 0.
- **Bit 4 (AF Bit, Auxiliary Carry Flag):**
  - It is used to simplify the addition and subtraction of packed BCD numbers.
  - AF is set if the operation resulted in a carry out of bit 3 (addition) or a borrow into bit 3 (subtraction). Otherwise AF is reset.
  - AF is only for bit 3.

80386 registers – Flag Register

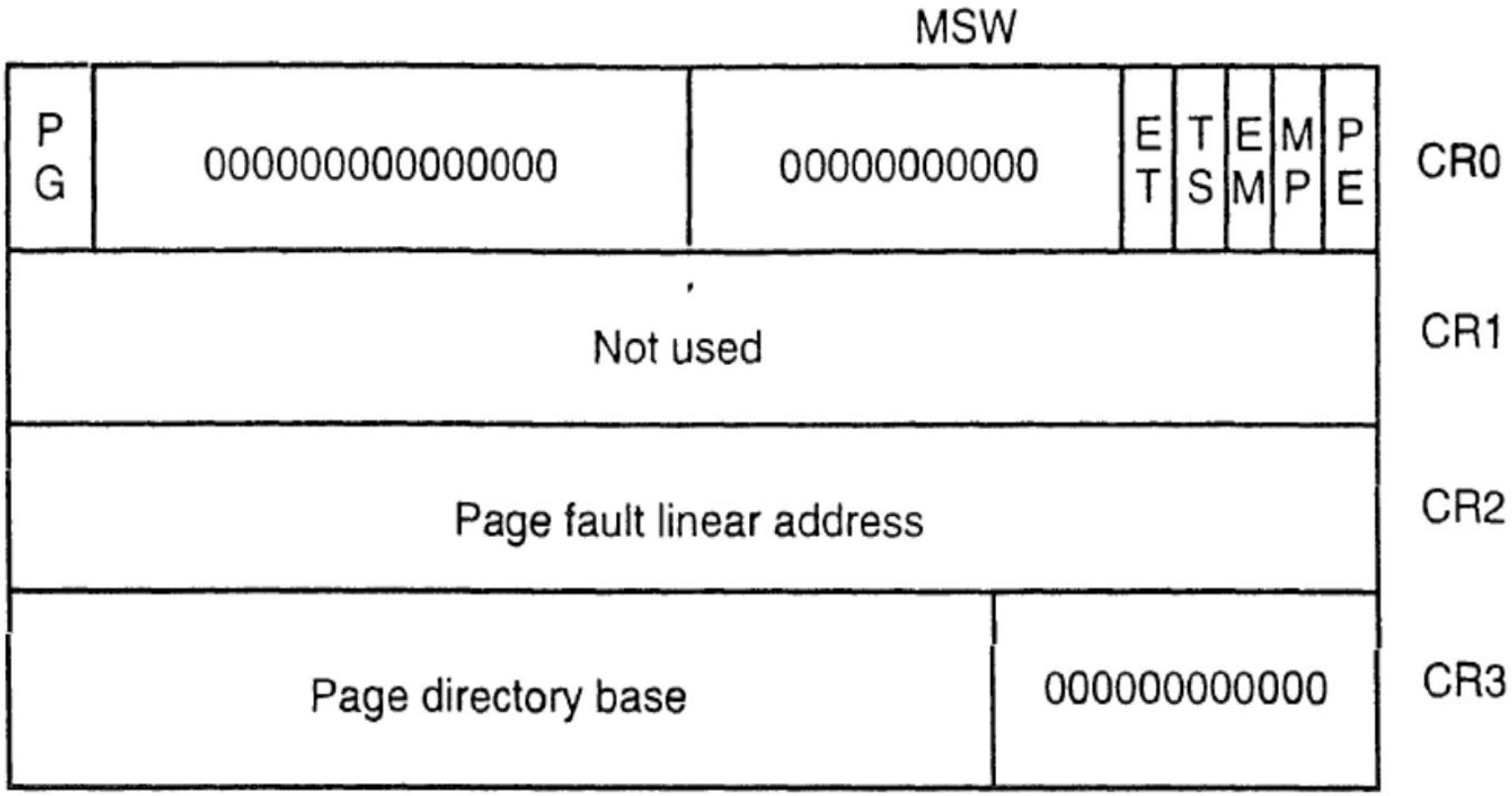


- **Bit 2 (PF Bit, Parity Flag):**
  - PF is set for even parity.
- **Bit 0 (CF Bit, Carry Flag):**
  - CF is set for 8-, 16- or 32-bit operations if it results in a carry out of (addition), or a borrow into (subtraction) the high-order bit.

80386 Control Registers

- Intel386 DX has 4 control registers(CR0, CR1, CR2 and CR3) of 32 bits to hold machine state of a global nature
- These registers along with System Address Registers hold machine state that affects all tasks in the system
- To access Control Registers, load and store instructions are defined

80386 Control Registers



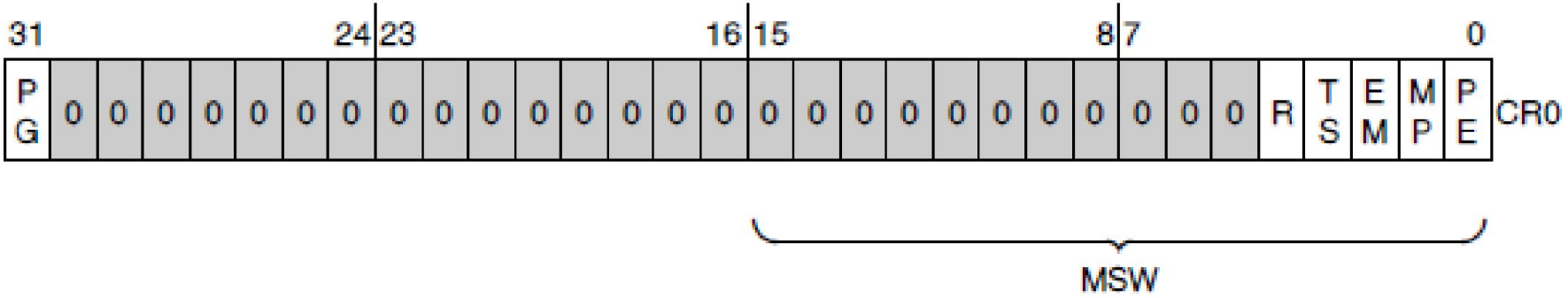
80386 Control Registers

CR0 : Machine Control Register

- CR0 contains 6 defined bits for control and status purposes.
- The low-order 16 bits of CR0 is defined as Machine Status Word
- To operate only on the low-order **16-bits** of CR0, **LMSW and SMSW** instructions are used.
- For **32-bit** operations the system should use **MOV CR0, Reg** instruction.

80386 Control Registers

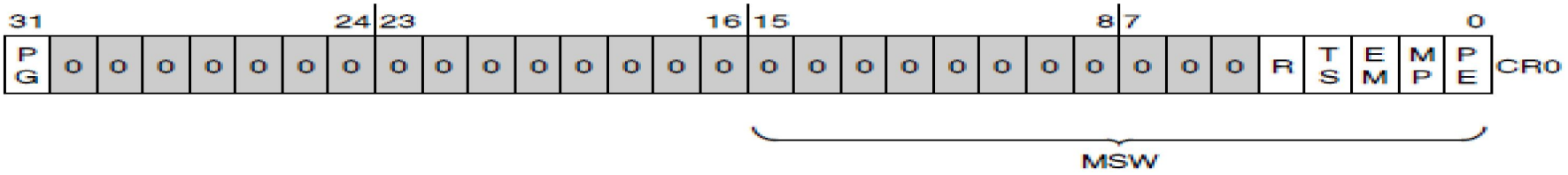
CR0 : Machine Control Register



- **Bit 31 (PG Bit, Paging Enable) :** The PG bit is set to enable the on-chip paging unit.
- **Bit 4 (Reserved) :** This bit is reserved by Intel.

80386 Control Registers

CR0 : Machine Control Register



Bit 3 (TS Bit, Task Switched) : TS is automatically set whenever a task switch operation is performed.

Bit 2 (EM Bit, Emulate Coprocessor) :

Bit 1 (MP Bit, Monitor Coprocessor) :

EM	MP	Interpretation
0	0	Numerics instructions are passed to coprocessor; WAIT ignores TS
0	1	Numerics instructions are passed to coprocessor; WAIT tests TS
1	0	Numerics instructions trap to emulator; WAIT ignores TS
1	1	Numerics instructions trap to emulator; WAIT tests TS

80386 Control Registers

CR0 : Machine Control Register

Bit 0 (PE Bit, Protection Enable) :

- PE =1, enable the Protected Mode.
- PE =0, processor operates in Real Mode.

80386 Control Registers

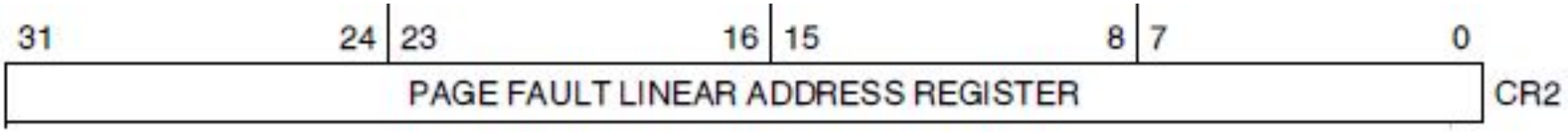
CR1 : Reserved

CR1 is reserved for use in future Intel processors

80386 Control Registers

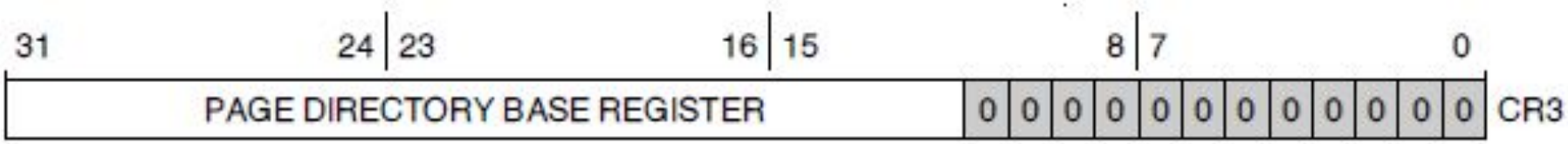
CR2 : Page Fault Linear Address

CR2 holds the 32-bit linear address that caused the last page fault detected.



80386 Control Registers

CR3 : Page Directory Base Address



- CR3 contains the physical base address of the page directory table.
- The Intel386 DX page directory table is always page-aligned (4 Kbyte-aligned).
- Thus the lowest twelve bits of CR3 are ignored.
- A task switch through a TSS invalidates all page table entries in paging unit cache.

80386 Modes of Operation

- **Real address mode**
- **Protected virtual address mode**
- **Virtual 8086 mode**

80386 Modes of Operation

Real address mode

It is the **default mode selected when 80386 is reset**. In this mode, 80386 µP simply behaves as a **fast 8086 machine**. All registers are just like 8086. Even the **memory used is only 1 MB**, just like in 8086. Physical address calculation is also like in 8086:

Physical Address = Base Address x 10H + Offset Address

Eg: Segment address (Base address) = 5142H and offset address = 0006H then the Physical address will be = 51426H

This mode is basically used to run the monitor program (BIOS) on reset.
Once the required registers are initialized, we can switch to Protected mode by making PE bit = 1 in CR0.

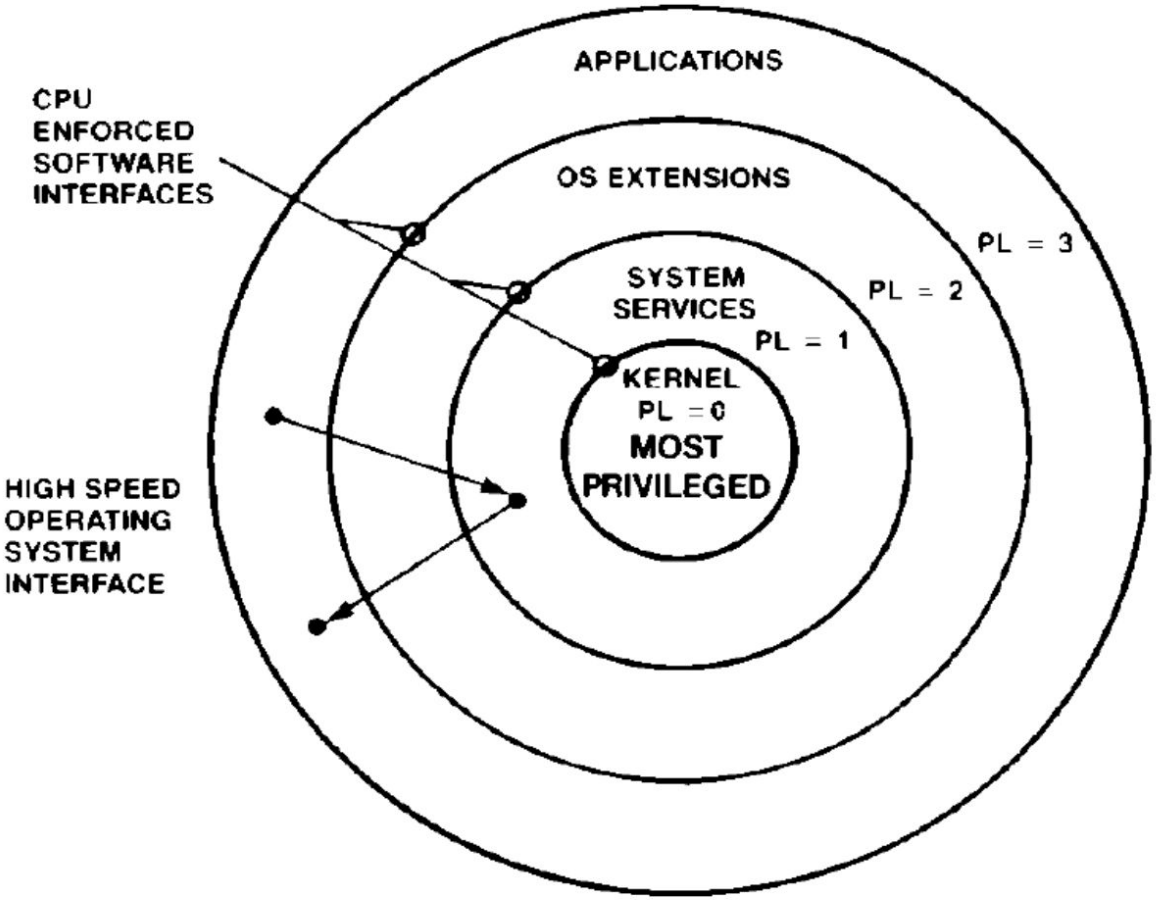
80386 Modes of Operation

Protected virtual address mode

80386 µP provides a very advanced mode of operations called the Protected Mode.
In Protected Mode, 80386 µP provides dedicated hardware to prevent user programs from affecting other user programs and also safeguards the Operating System from being affected by user programs.
There are Four Privilege Levels, assigned to programs and data to define their privileges.

80386 Modes of Operation

Protected virtual address mode



80386 Modes of Operation

Protected virtual address mode

Level 0: **This level is assigned to the Operating System Kernel (Main part of the Operating System).**

It is the most privileged level.

Any program at this level can access all the data at any Privilege Level, whereas a data at this Privilege Level can only be accessed by a program at Privilege Level 0.

Level 1: **This level is assigned to the System Services such as File Handling, Device Drivers.**

Any program at this level can access the data at any Privilege Level which is lower than this level (numerically higher), whereas a data at this Privilege Level can only be accessed by a program at Privilege Level 0 or Privilege Level 1.

80386 Modes of Operation

Protected virtual address mode

Level 2: **This level is assigned to the Custom Extensions of the OS.**
It is the 3rd most privileged level.
Any program at this level can access the data at any Privilege Level which is lower than this level (numerically higher), whereas a data at this Privilege Level can only be accessed by a program at Privilege Level 0, 1 or 2.

Level 3: **This level is assigned to all the User Application and Programs.**
It is the least privileged level.
Any program at this level can normally access the data at Privilege Level 0, whereas a data at this Privilege Level can only be accessed by a program at any Privilege Level 0...3.

80386 Modes of Operation

Virtual 8086 mode

- In its protected mode of operation, 80386DX provides a virtual 8086 operating environment to execute the 8086 programs.
- Processor can switch from PVAM to V86 by setting VM bit in the EFLAG register to logic 1.
- The real mode can also be used to execute the 8086 programs along with the capabilities of 80386, like protection and a few additional instructions.
- Once the 80386 enters the protected mode from the real mode, it cannot return back to the real mode without a reset operation.
- Thus, the virtual 8086 mode of operation of 80386, offers an advantage of executing 8086 programs while in protected mode.

80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode

80386 transforms logical addresses into physical address two steps:

Segment translation: a logical address is converted to a linear address.

Page translation: a linear address is converted to a physical address.(optional)

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

80386 memory management in Protected Mode

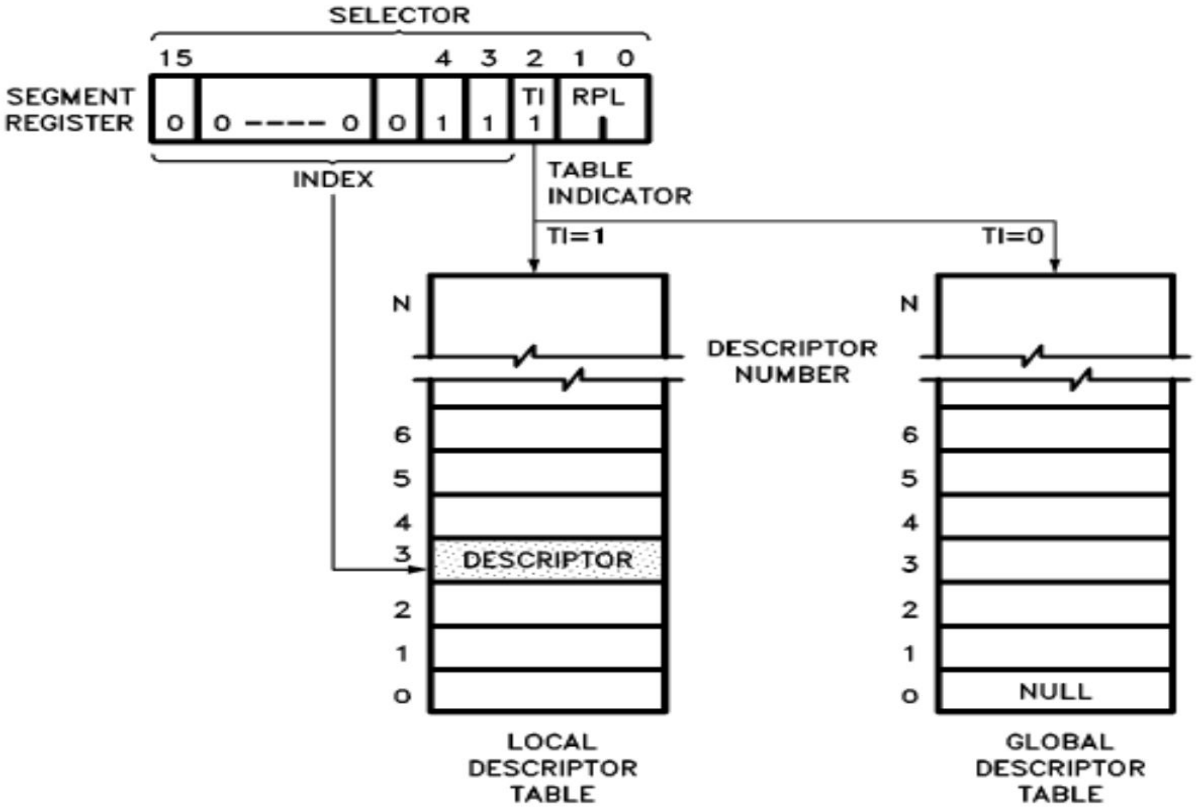
Address Translation mechanism in Protected Mode : Segmentation

VIRTUAL AND PHYSICAL ADDRESS SPACE

1) Physical Memory is the total memory that can be directly connected to the CPU using its address bus.
2) **Since 80386 µP has a 32 bit address bus, the total physical memory that can be connected is 2^{32} = 4GB.**
3) Virtual Memory is the total memory space that can be addressed by the CPU registers.
4) **Virtual address is basically the combination of a 16-bit segment register and a 32 bit offset register.**
5) In Real Mode, the segment register gives the starting address of the segment.
6) But in Protected Mode, the segment register just gives a selector which selects a Descriptor for the segment.

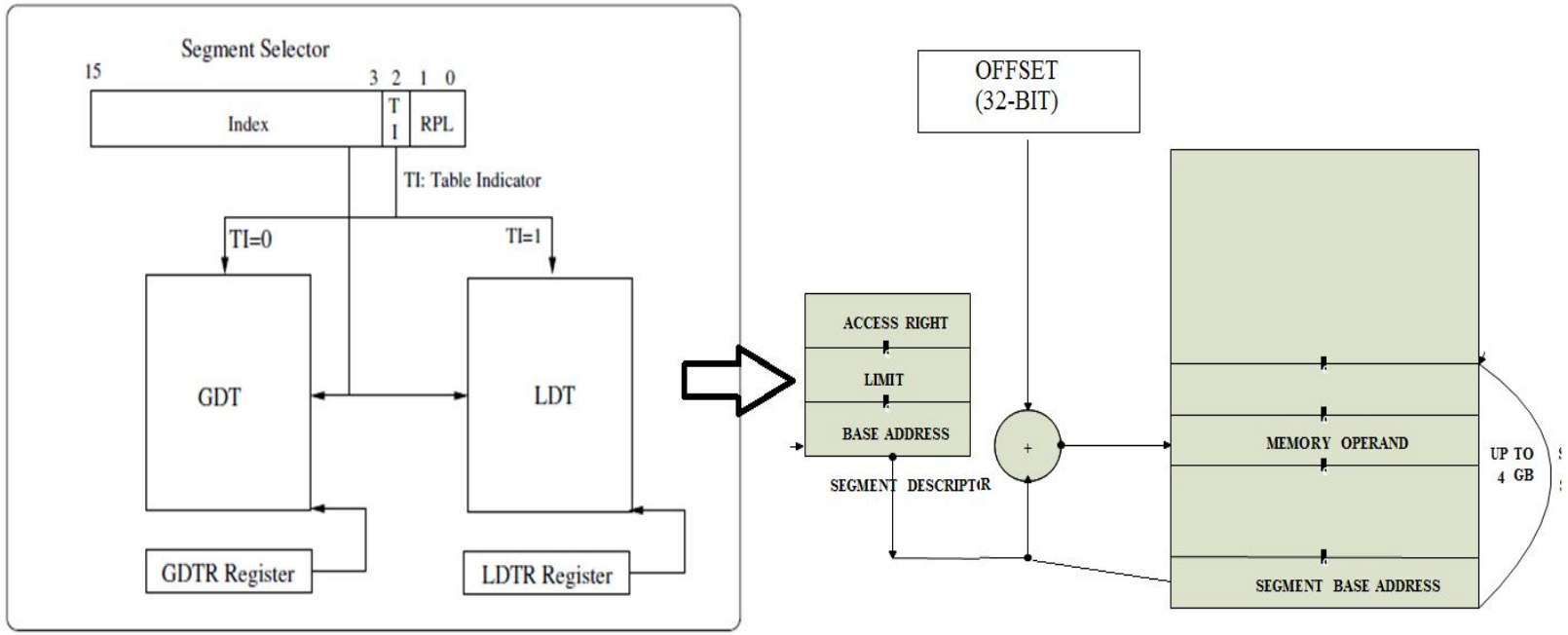
80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode : Segmentation



80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode : Segmentation



80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode : Segmentation

1) Though the selector is of 16-bits, only 14 bits are used as two bits give the Privilege Level used for protection as seen above. Each selector value corresponds to a different segment. Hence there can be max 2^{14} segments.
2) The locations within the segment are identified by their offset addresses. Since offset addresses are 32 bit, each segment can be max 2^{32} = 4GB. Hence the max total Virtual memory that can be accessed = Max number of segments x Max size of each segment **= 2^{14} x 2^{32} = 2^{46} = 2^{6} x 2^{40} = 64 x 1TB = 64 Terra Bytes.**
3) Out of 14 bits of the selector, one bit is used as a table identifier (TI)
   if TI = 1 then use LDT, if TI = 0 then use GDT.
4) So basically there are 2^{13} (i.e. 8K) Descriptors in LDT and 8K Descriptors in GDT.
5) This means the total 64TB Virtual space is divided into 32TB of Global space and 32 TB of Local Space.

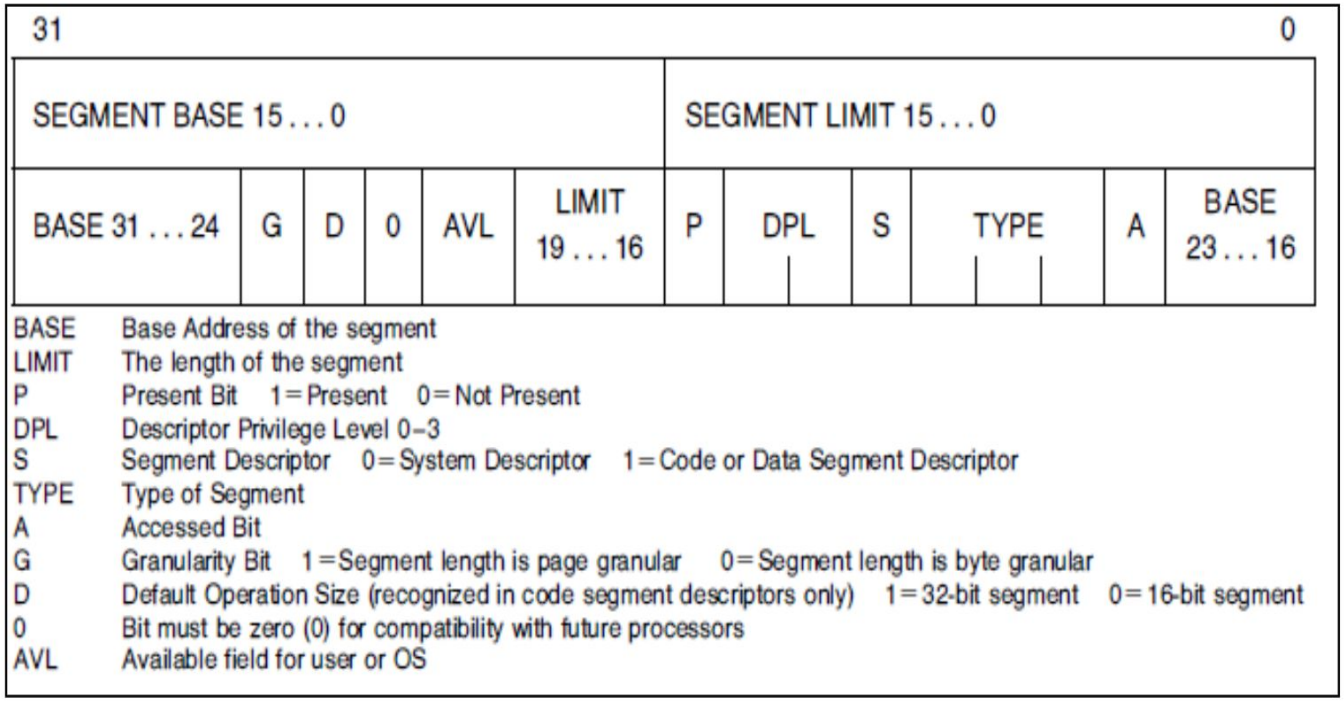
80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode : Segmentation

6) The total 48 bit address having 16 bit segment address and 32 bit offset address is called Virtual address. It is converted into a 32 bit physical address using two translations: Segment translation (compulsory) and page translation (optional).
7) Segment translation converts 48-bit Virtual address into 32 bit Linear Address which is further converted into a 32 bit Physical Address by Page translation.

Segment Descriptor

Basic structure of a segment descriptor



80386 memory management in Protected Mode

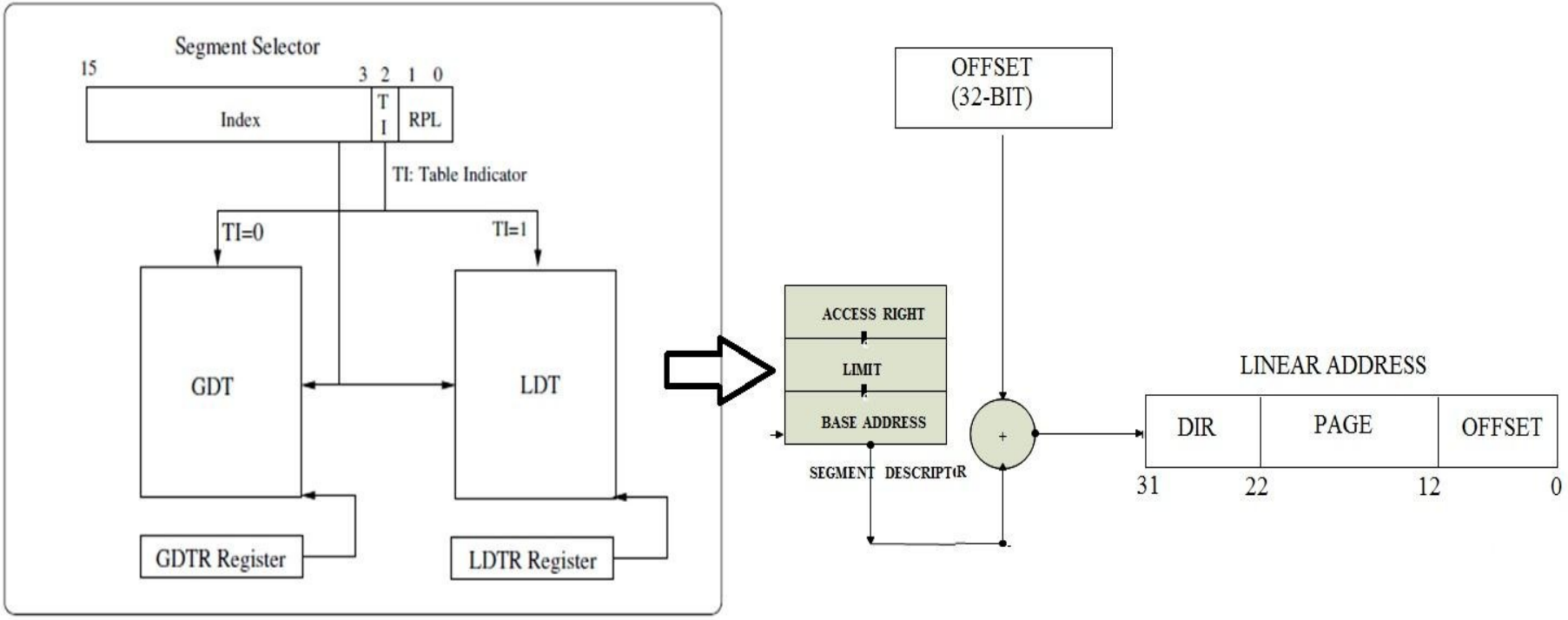
Address Translation mechanism in Protected Mode : Paging

- In paging concept, each segment is composed of numerous pages each of 4KB.
- The base address of each page is stored in a page table.
- There will be many such page tables.
- The base address of page table and other details (page descriptor) will be in a page directory.
- The base address of current page directory will be in CR3 register

80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode : Paging

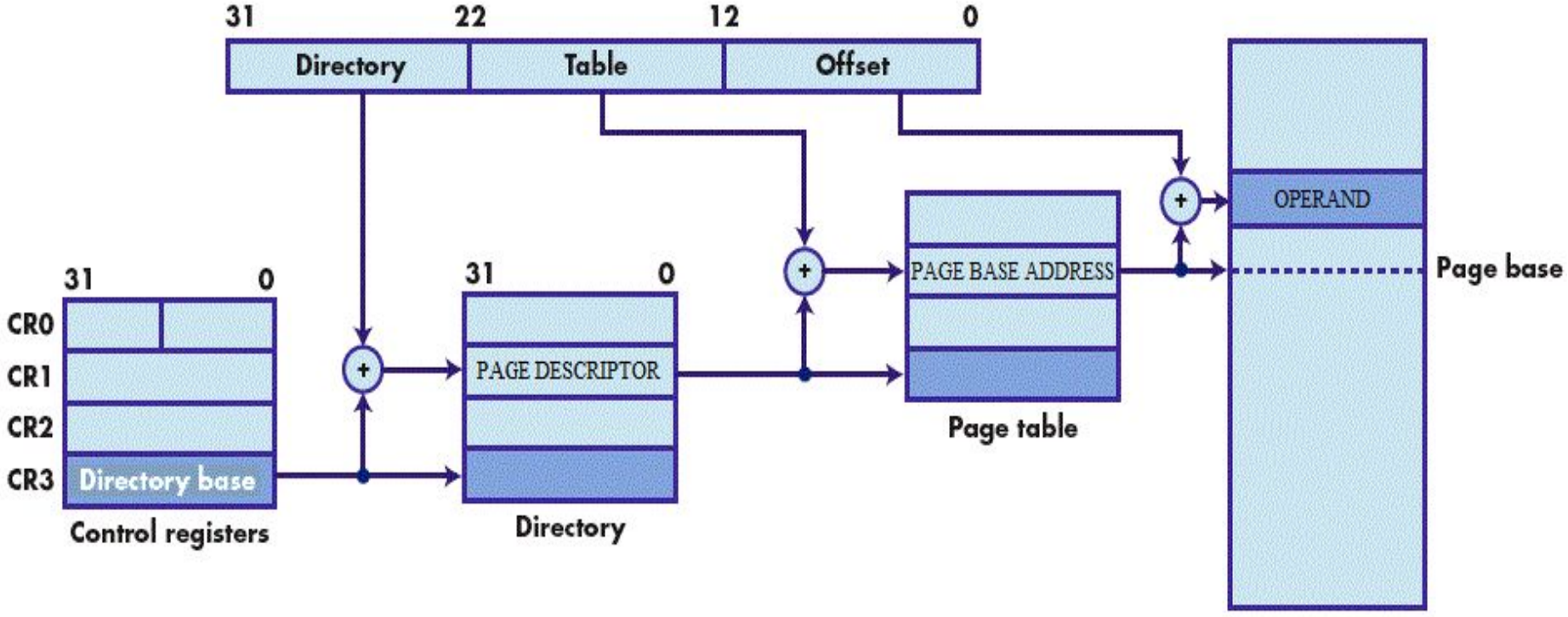
Using segmentation, a 32-bit linear address is generated



80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode : Paging

Using this linear address, an operand from a page is selected as shown below:



80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode : Paging

- CR3 register contains the base address of current page directory. Using DIR in linear address, a corresponding page descriptor is selected.
- Page descriptor contains base address of page table. The PAGE field of linear address is used to select the corresponding page table entry.
- The page table entry contains the base address of the page we are looking for. The OFFSET field in the linear address is used to locate the operand in physical memory.

80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode : Paging

- In paging concept, each segment is composed of numerous pages each of 4KB.
- The base address of each page is stored in a page table. There will be many such page tables.
- The base address of page table and other details (page descriptor) will be in a page directory.
- The base address of current page directory will be in CR3 register

80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode : Paging ---Page Directory

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

- It is at the most 4KB in size and allows upto 1024 entries are allowed.
- The upper 10 bits of the linear address are used as an index to corresponding page directory entry
- Page directory entry points to page tables.

80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode : Paging ---Page Table

- Each Page Table is 4KB and holds up to 1024 Page Table Entries(PTE).
- PTEs contain the starting address of the page frame and statistical information about the page.
- Upper 20 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.

80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode : Paging ---Page Table

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

- **P(Present)Bit**: indicates if the entry can be used in address translation. P-bit of the currently executed page is always high.
- **A (Accessed) Bit**: It is set before any access to the page.
- **D (Dirty) bit**: It is set before a write operation to the page is carried out. The D bit is undefined for PDEs.

80386 memory management in Protected Mode

Address Translation mechanism in Protected Mode : Paging ---Page Table

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

- **OS Reserved Bits:** They are defined by the operating system software.
- **U/S (User/Supervisor)Bit and R/W (Read/Write) Bit:** They are used to provide protection. They are decoded as :

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

80386 Protection Mechanism

- 80386 DX has four levels of protection which isolate and protect user programs from each other and the operating system Uses:

1) Segment Level Protection

2) Page level Protection mechanism

- Before memory cycle, each memory reference is checked by the hardware to verify that it satisfies the protection criteria

80386 Protection Mechanism

- Has five aspects
  - Type Checking
  - Limit Checking
  - Restriction of addressable domain
  - Restriction of procedure entry point
  - Restriction of Instruction Set.

80386 Protection Mechanism

- **Type Checking:**
  - Type field of the descriptor specifies
    1. Type of descriptor (system/non-system)
    2. Intended usage of the segment.

    Eg. If the segment is read only segment then its accessed is limited to only reading purpose

80386 Protection Mechanism

Limit Checking:

- To prevent program from addressing outside the segments.
- It interprets limit field depending on the setting of the G (granularity) bit, which specifies whether limit value counts 1 byte or 4 Kb.

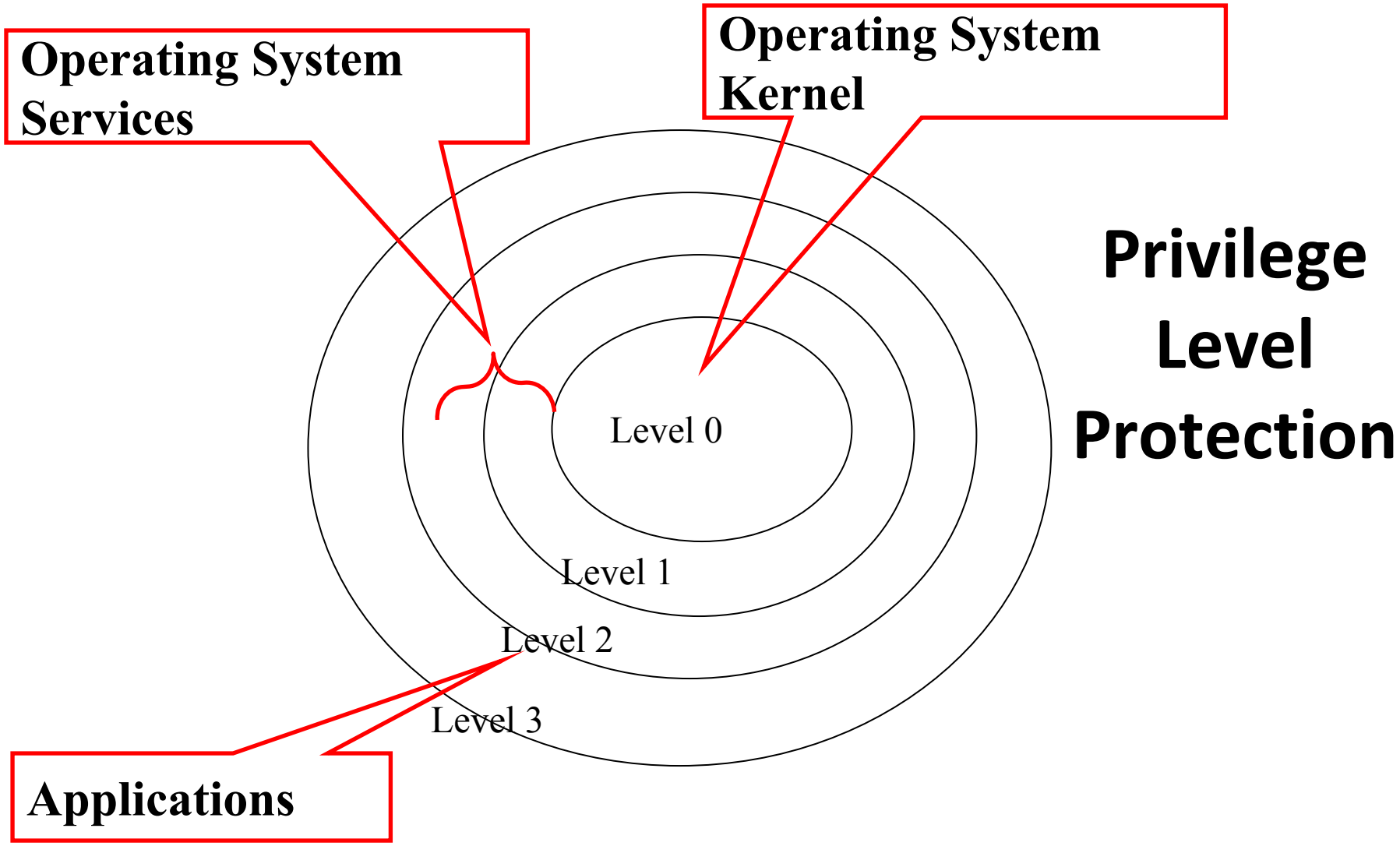
8-bit Instruction	Offset <=Limit
16-bit Instruction	Offset <=Limit-1
32-bit Instruction	Offset <=Limit-3

- If granularity bit is 0 then the limit is same as the limit but if granularity bit is 1 then limit= limit * 4KB, then it will be compared because the actual size limit is limit x 4KB

Privilege Level Protection

- The privilege levels (PL) are numbered 0 through 3.
- Level 0 is the most privileged or trusted level.

80386 Protection Mechanism



80386 Protection Mechanism

- There are 3 different types of privilege level entering into the privilege level checks:
  - **Current Privilege Level (CPL)**
  - **Descriptor Privilege Level (DPL)**
  - **Requestor Privilege Level (RPL)**
  - **Effective Privilege Level (EPL**

Protection Mechanism

CPL – Current Privilege Level is the privilege level of the currently executing program or task. It is stored in CS and SS segment registers

DPL - Descriptor privilege level is the privilege level of a segment. It is stored in the DPL field of the segment descriptor

RPL - Requested privilege level is an override privilege level use to specify the target segment.

EPL - EPL is defined as

EPL = max { RPL,CPL } (numerically)

Thus the task becomes less privileged

Protection Mechanism

Current Privilege Level(CPL)

- Also called **Task Privilege Level**
- It specifies privilege level of **currently executing task**
- A task's CPL can **only be changed by** control transfers through **gate descriptors** to a code segment with a different privilege level.
- E.g. an **application program** running at **PL = 3** may call an **OS routine** at **PL = 1 (via a gate)** which would cause the task's CPL to be set to 1 **until the OS routine is finished.**

Protection Mechanism

Current Privilege Level(CPL)

- **Normally, CPL = DPL** of the segment that the processor is currently executing.
- **CPL changes** as control is transferred to segments **with differing DPLs.**

Protection Mechanism

Descriptor Privilege Level (DPL)

- It is the PL of the object which is being attempted to be accessed by the current task
- It is PL of target segment and is contained in the descriptor of the segment

Protection Mechanism

Requestor Privilege Level (RPL)

- RPL is the **two least significant bits of selector**
- RPL is used **to establish a less trusted privilege level than CPL** for the use of a segment and this level is called the task's **Effective Privilege Level** (EPL).
- EPL is defined as

  EPL = max { RPL,CPL } (numerically)

  - Thus the task becomes less privileged
- E.g. If RPL = 2 and CPL = 1, EPL = 2 ? task becomes less privileged

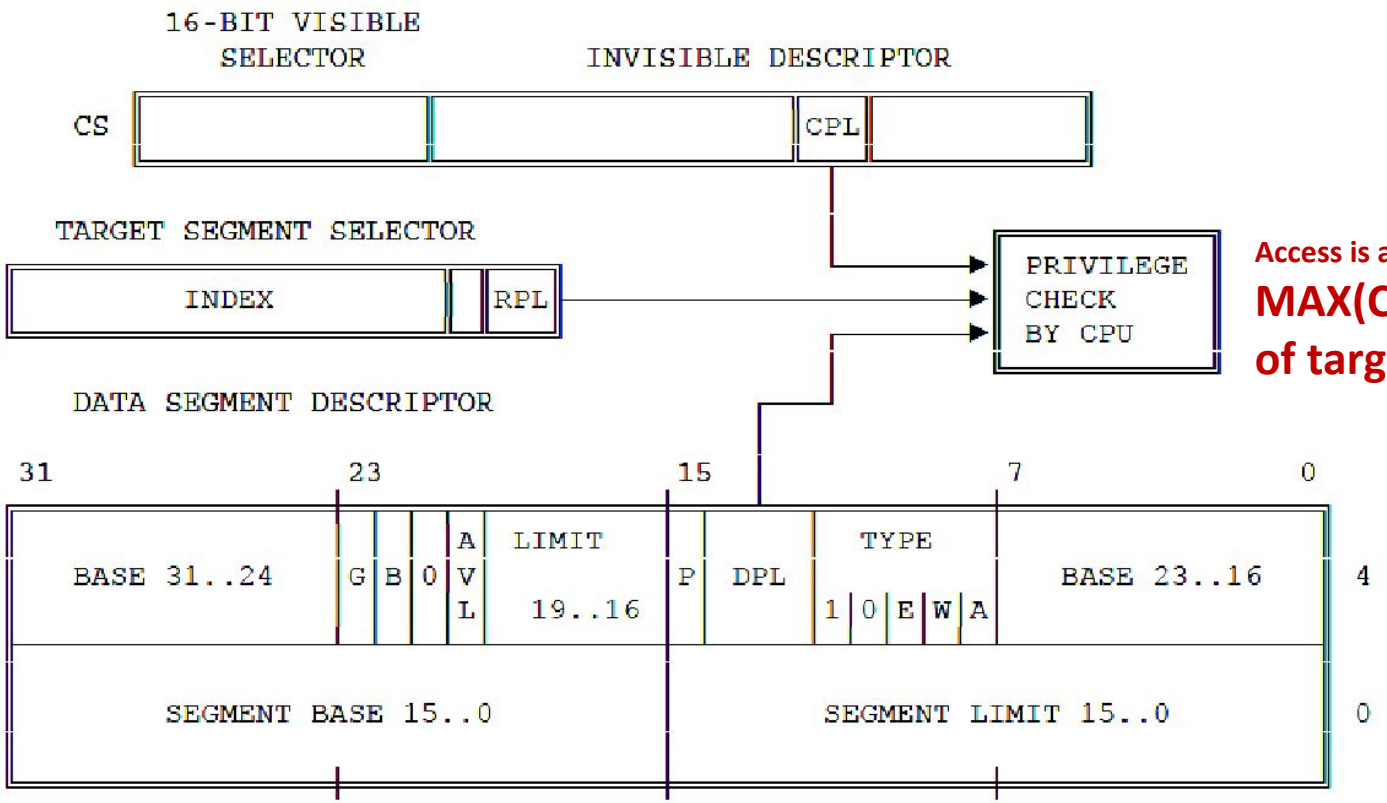
Protection Mechanism

Restricting Access to Data

- Assume that a task needs data from data segment.
- The privilege levels are checked at the time a selector for the target segment is loaded into the data segment register.
- Three privilege levels enter into privilege checking mechanism
  - CPL
  - RPL of the selector of target segment
  - DPL of the descriptor of the target segment

Protection Mechanism

Restricting Access to Data



Access is allowed only if

MAX(CPL,RPL) <= DPL of target

CPL - CURRENT PRIVILEGE LEVEL
RPL - REQUESTOR'S PRIVILEGE LEVEL
DPL - DESCRIPTOR PRIVILEGE LEVEL

Protection Mechanism

Restricting Access to Data

Sr.No.	DPL	CPL	RPL	Access
1	2	0	1	Valid
2	3	1	2	Valid
3	1	1	0	?
4	1	2	0	?
5	2	2	3	?

- A procedure can only access the data that is at the same or less privilege level (not numerically)
- Instructions may load a data-segment register only if **DPL(data) ≥ max {CPL(proc),RPL}** numerically