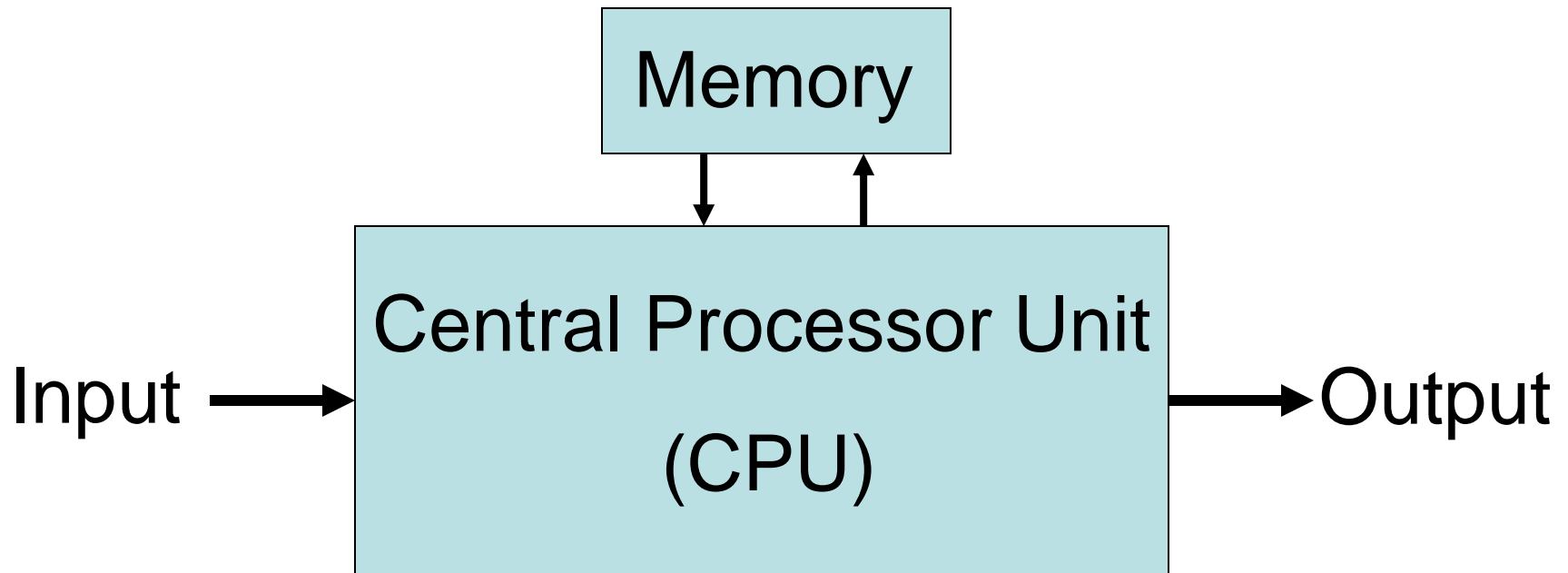


The Functions of a Computer

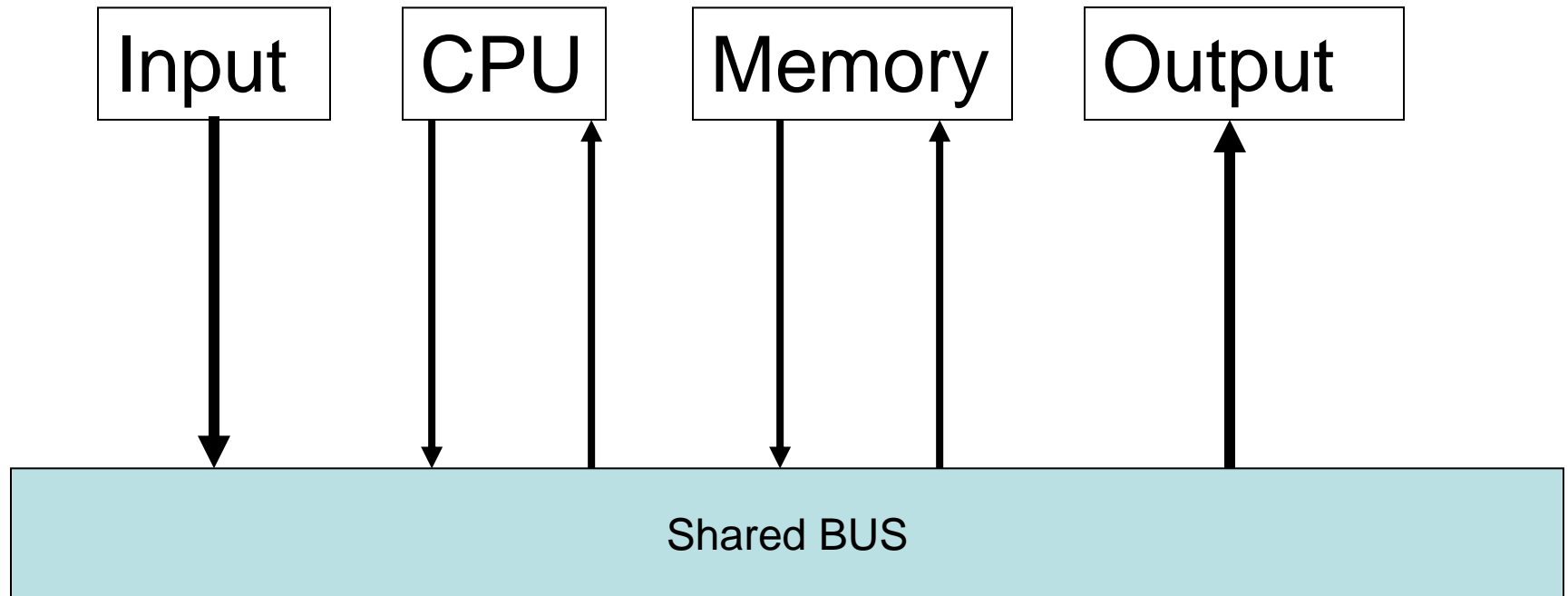
Review

A Typical Computer System

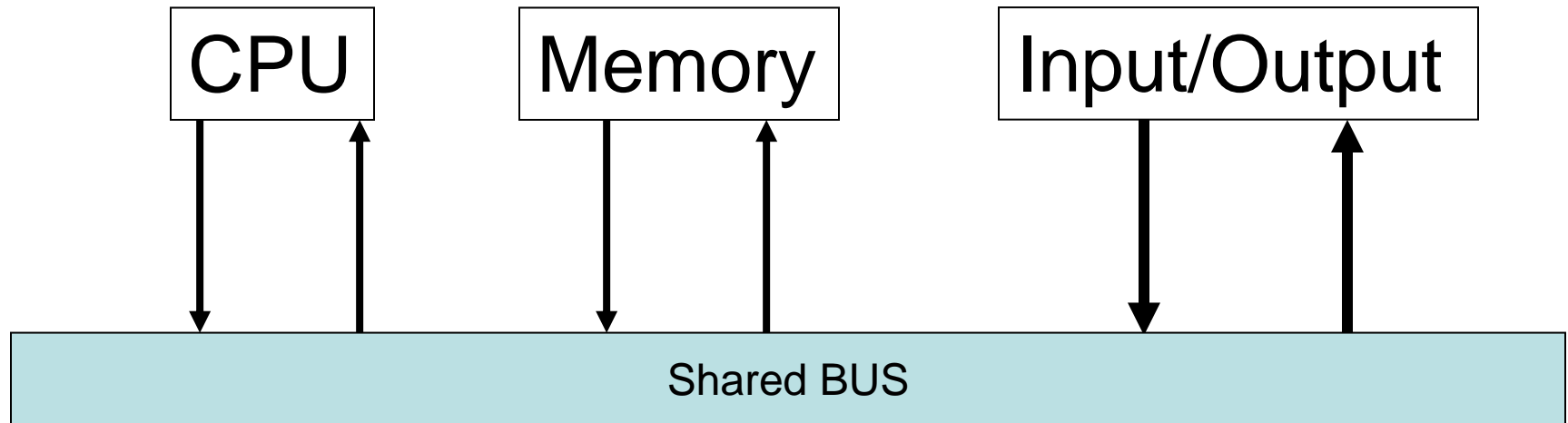
Early days



A Typical Computer System nowadays



A Typical Computer System nowadays



- μ P cannot drive I/O devices by itself
 - For I/O, peripheral chips are used
- 8255 from Intel is such a chip

Memory

- Store ***Instructions*** and ***Data***
 - ***Instructions*** are coded pieces of information that direct the activities of CPU
 - ***Data*** are coded pieces of information that are processed by CPU

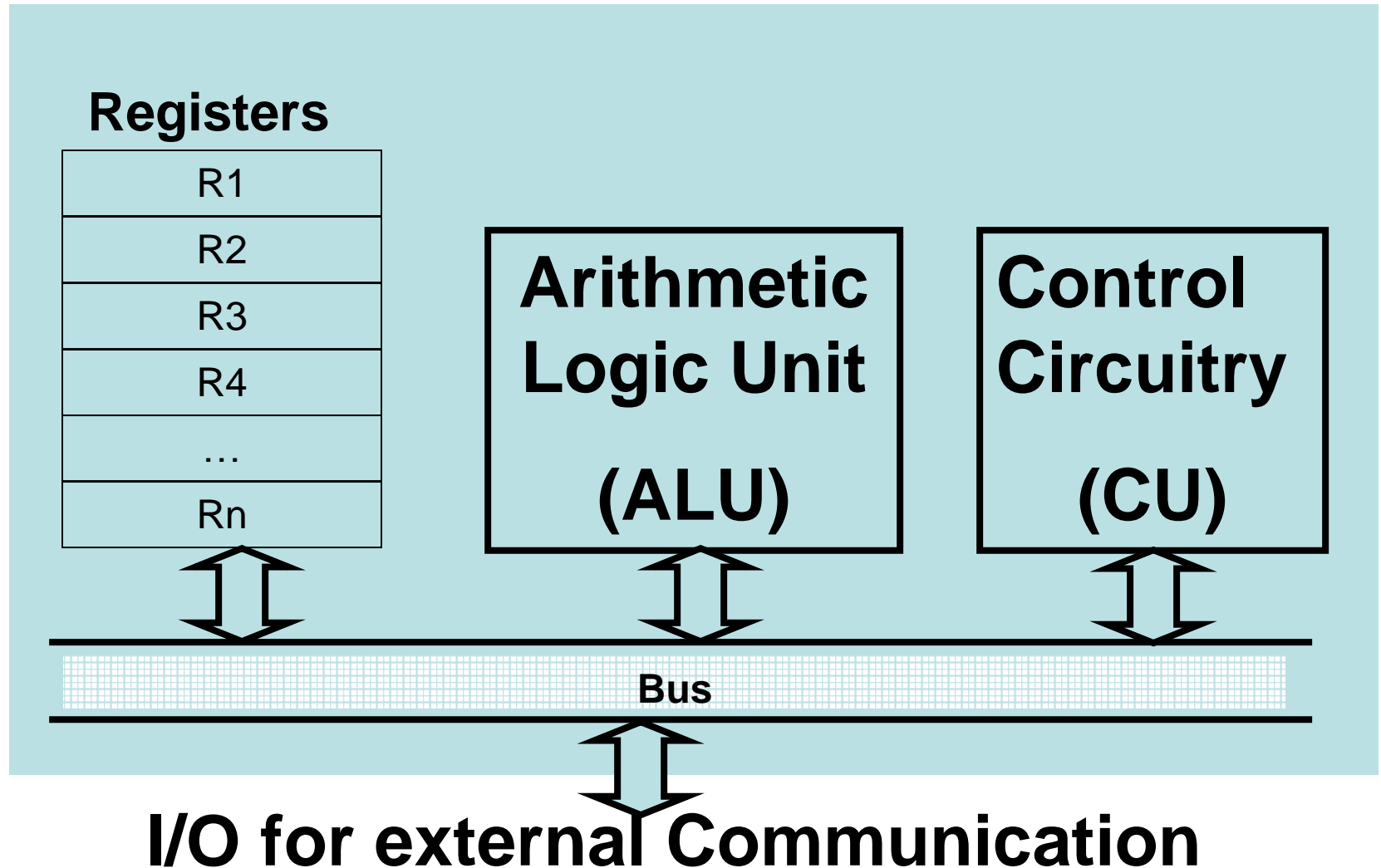
Input Ports

- But often the ***memory*** is not large enough to store the entire ***data*** required for a particular application
- The problem can be resolved by providing the computer with one or more ***Input Ports***
- Also ***Input Ports*** enable Computer to receive information from external equipments

Output Ports

- Computers also require ***Output Ports***
 - To communicate the results of its processing to the outside world
 - To display unit : for human operator
 - To a peripheral device to produce 'hard copy' : Printer
 - To a peripheral storage device : Hard Disk
 - Or the output may constitute process control signals that control the operations of another system : Process control in industry

The Internal Architecture of a CPU



Registers

- Temporary storage unit within the CPU
- Some registers have dedicated uses
- Others are for more general purpose use
- Often in some μP one register is specially known as ***Accumulator***
- Numbers of registers vary from μP to μP

ALU

- As its name implies, **ALU** is that portion of the CPU hardware which performs the ***arithmetic*** and ***logic*** operations on the ***binary data***
- Generally **ALU** contains simply ***an Adder***
 - to perform all arithmetic operations like
 - Subtraction, multiplication, division
 - and all logic operations like
 - AND, OR, NOT

ALU (cont...)

- ALU also contains ***Flag Bits*** which specify certain conditions/status that arise after arithmetic and logic operations

CU

- CU is the primary functional unit within a CPU
- CU maintains the proper sequence of events required for any processing task
 - Using clock inputs
- CU issues the appropriate signals for initiating the proper processing action
 - To units both internal and external to CPU
- Often CU is capable of responding to external signals known as ***interrupt***

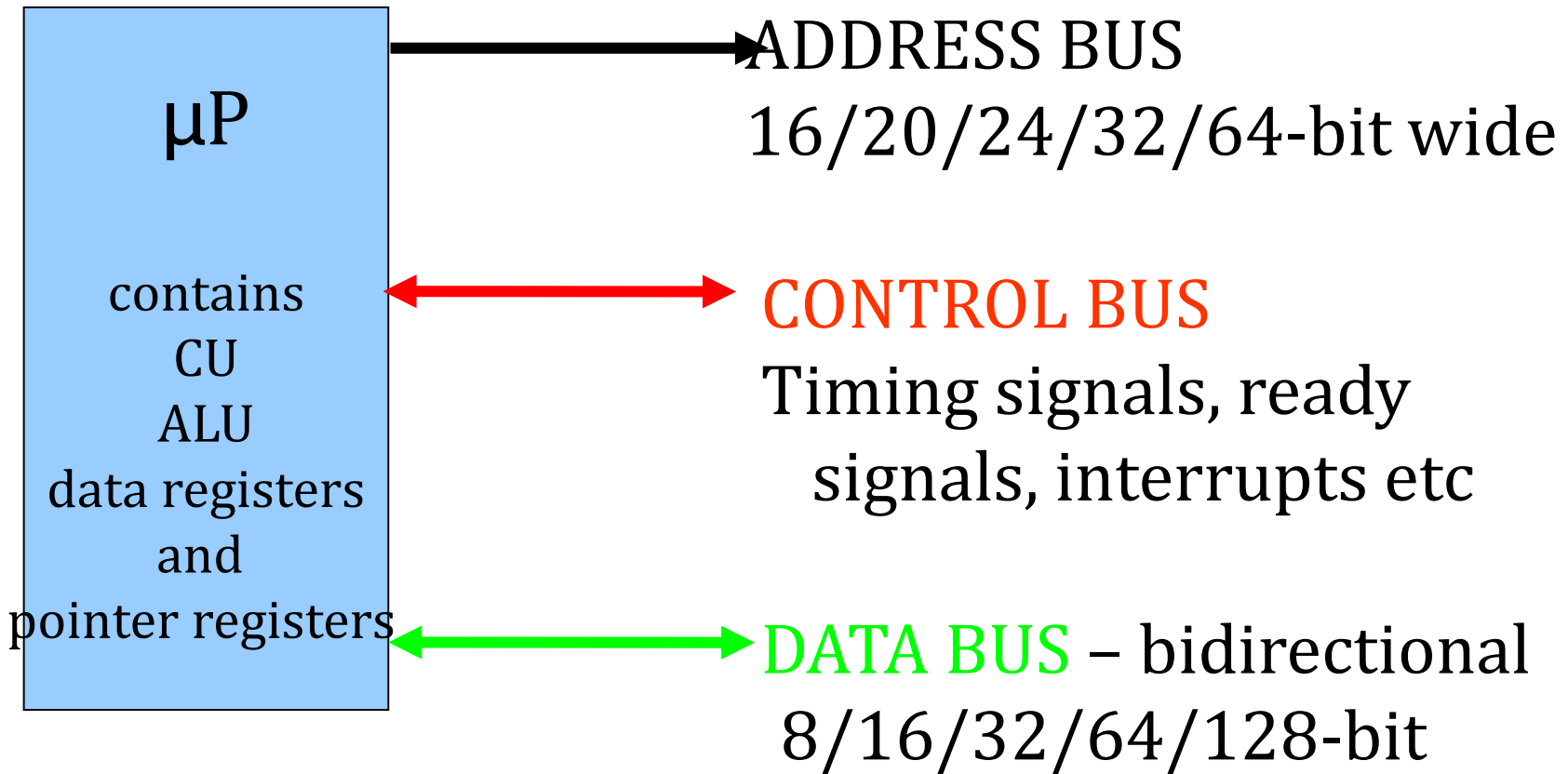
Interrupt

- An ***Interrupt*** request will cause the **CU** to ***temporarily*** interrupt main program execution to serve the interrupting device
- After serving the interrupting device, the **CU** will automatically return to the main program

BUS

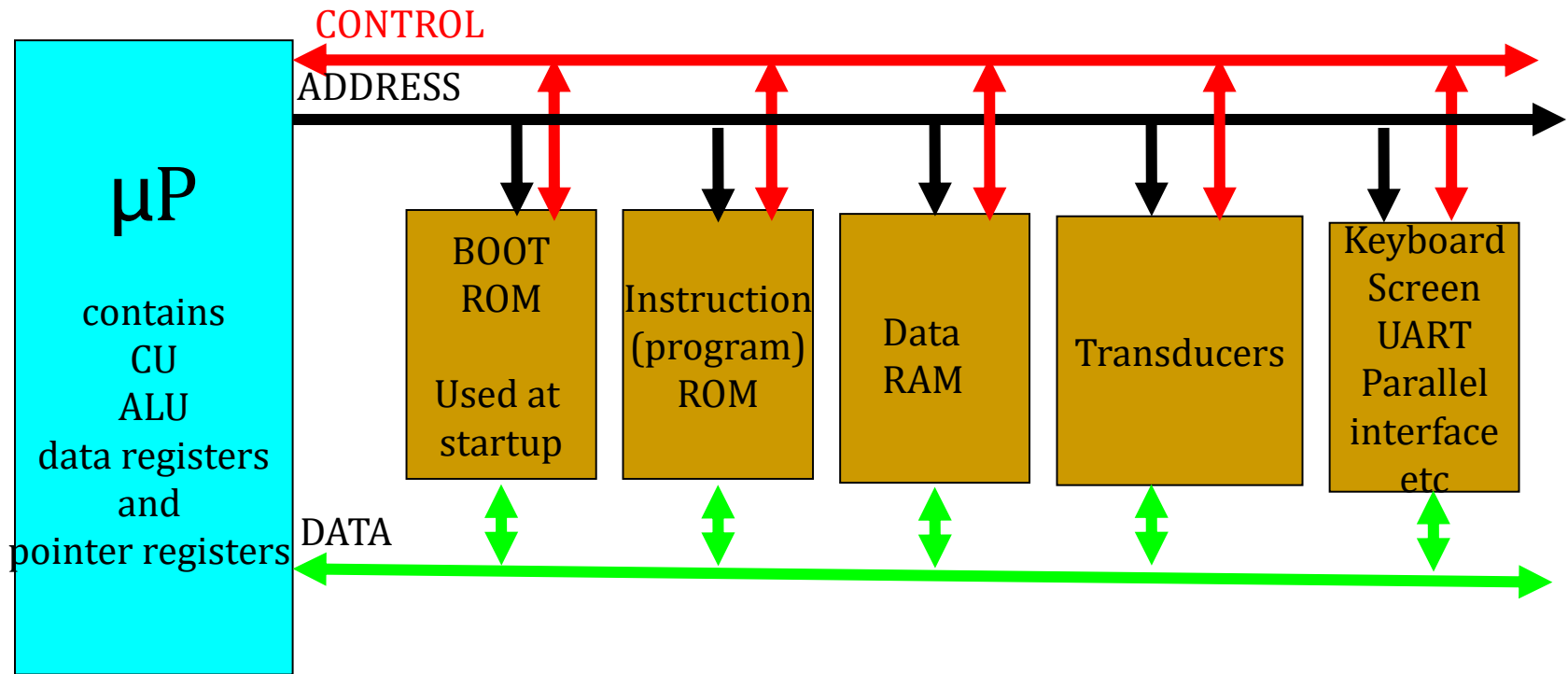
- Set of single or parallel lines grouped together for connecting Registers, ALU and CU with each other
- ***Three types of buses*** we will see in a CPU
 - ***Address Bus***
 - ***Data Bus***
 - ***Control Bus***

Microprocessor – Basic concept



Microprocessor – Basic concept

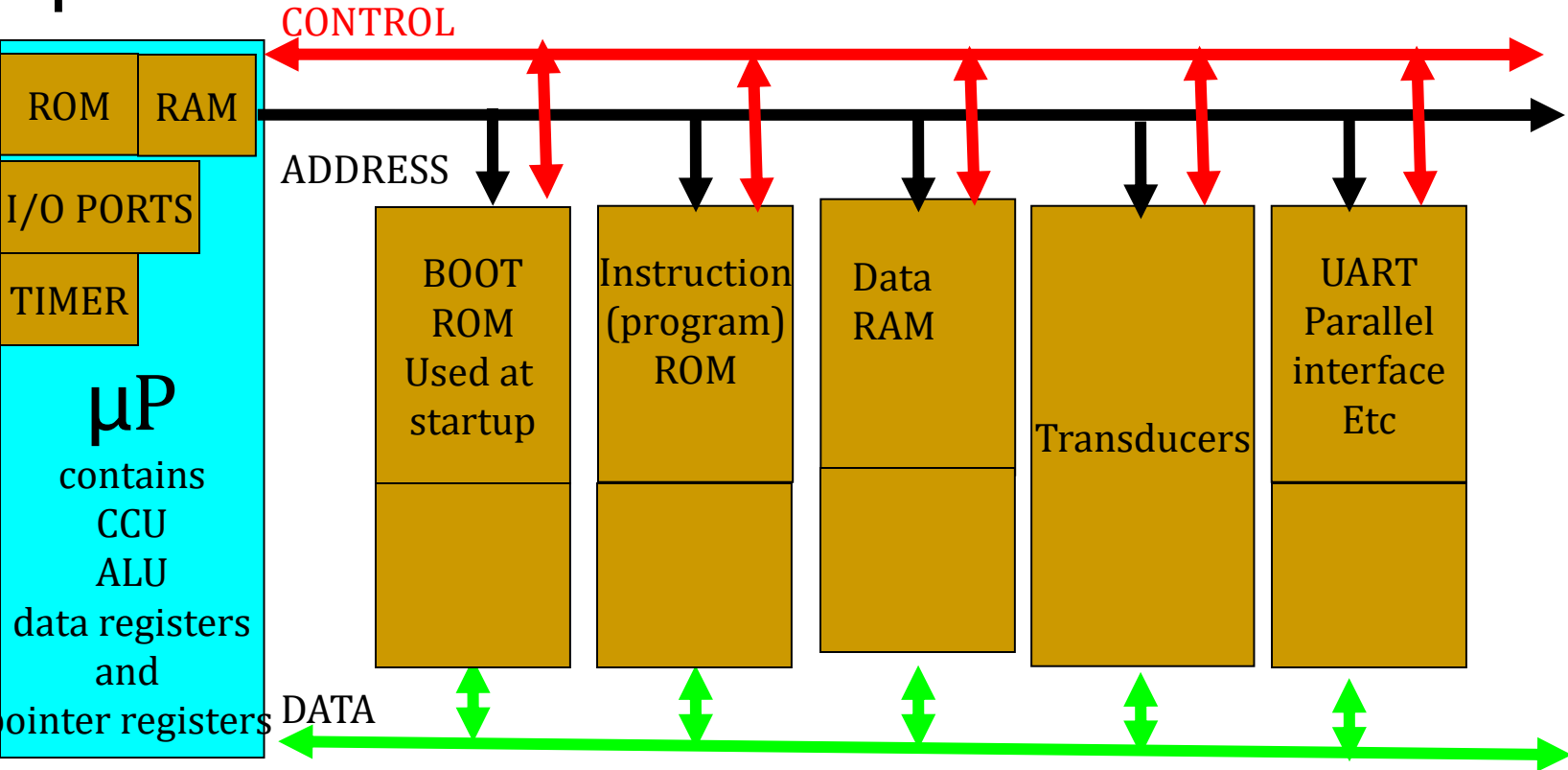
Microprocessor, by-itself, is completely useless
must have external peripherals to interact with outside world



Many chips on mother board

MicroCONTROLLER – Basic concept

MicroPROCESSOR – Basic concept

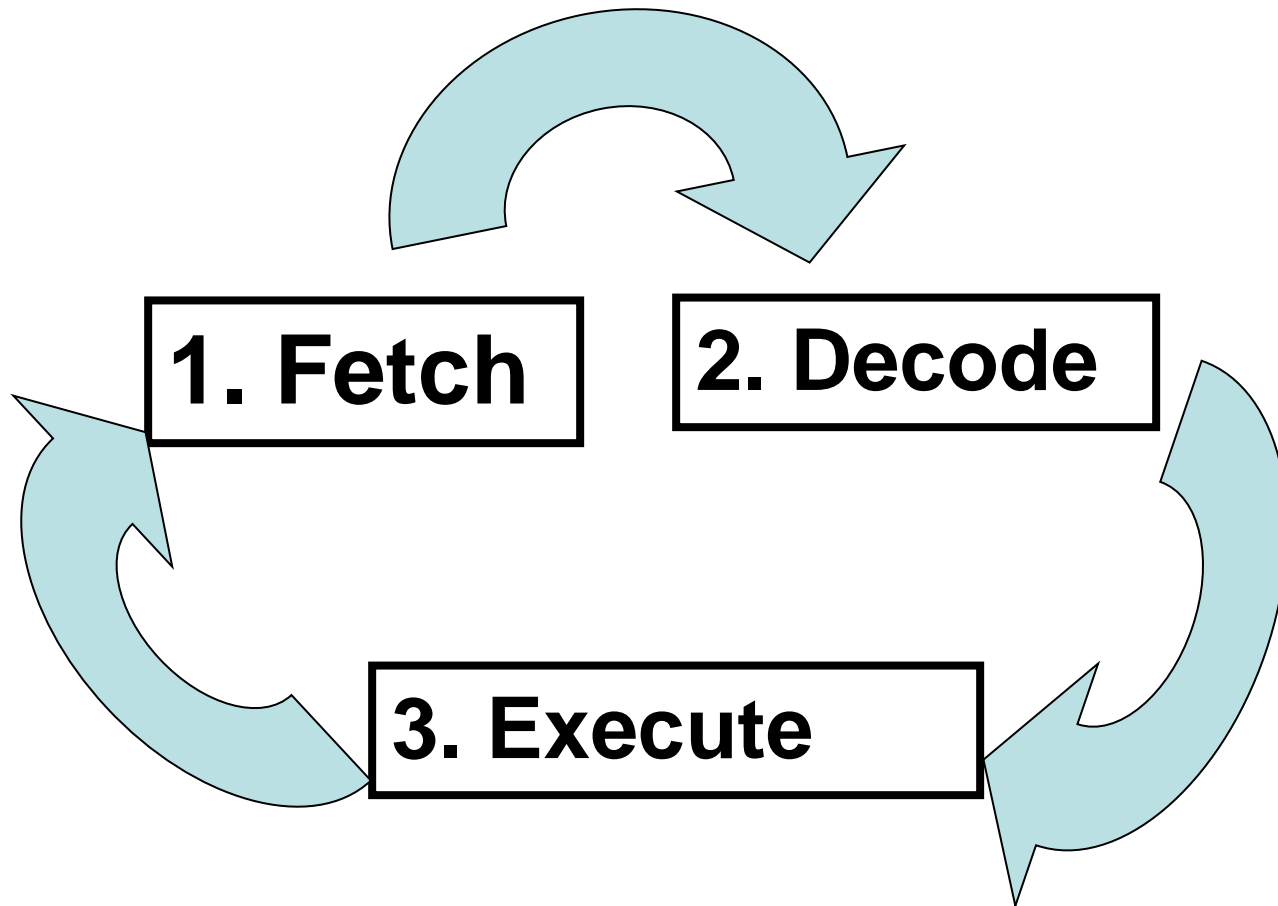


Microcontroller – put a limited amount of most commonly used resources “inside” the chip – a “limited” amount is often “enough” for many applications

Advantages of microCONTROLLER over microPROCESSOR

- Pin count down
- Design time down, Board layout size down
- Upgrade path easier – matching between peripherals for speed
- Cost down – bulk purchases
- Reliability up
- Common software / hardware design environment available from manufacturer

Operation of μ P



Computer Operation (cont...)

- The combined **fetch-decode-execute** operation of a single instruction is known as **Instruction Cycle**
- The portion of a cycle identified with a clearly defined activity is called a **State**
- The interval between pulses of the timing oscillator is known as **Clock Period**
- As a general rule, one or more clock periods are necessary for the completion of a state
- And there are several states in a cycle

What is an *n-bit* processor?

- If a processor can operate on *n-bit* of data simultaneously, then the processor will be called an *n-bit* processor.
- ALU operates on data
- So, ALU size determines the size of the processor
-

Microprocessor

8086

INTRODUCTION

Intel Datasheet

Rafiq 3.1

Hall

Brey

8086 Features

- First 16 bit processor from Intel
 - introduced in 1978
 - produced from 1978 to 1990s
- CPU clock speed 5MHz to 10MHz
- Packaged in 40-pin DIP
- 8 general purpose register
- prefetches up to 6 instructions from memory and queues them within the chip to speed up execution

8086 Features...

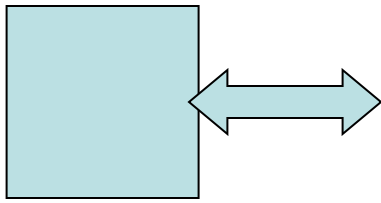
- 16 bit data bus
- 20 bit address bus
 - can address up to 1MB memory
 - Intel has 2 peculiarities in accessing memory
 - memory must be divided in to 2 banks
 - instead of a flat memory, it uses segmented memory
- 21 bit control bus
- most of the lines/pins are used for more than one purpose
 - hence they are multiplexed

8086 INTERNAL ARCHITECTURE

Rafiq 3.2

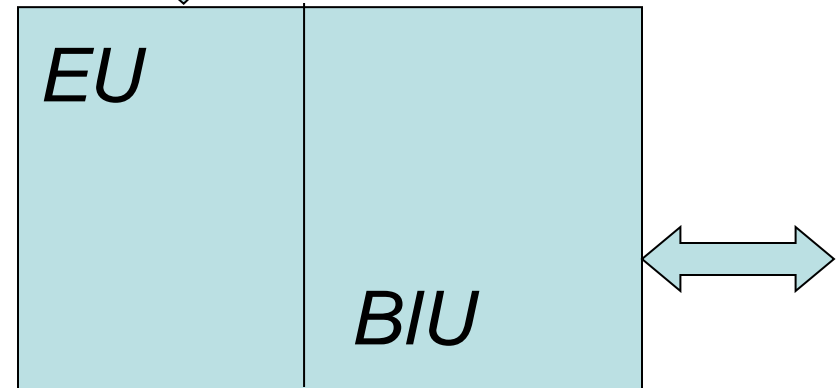
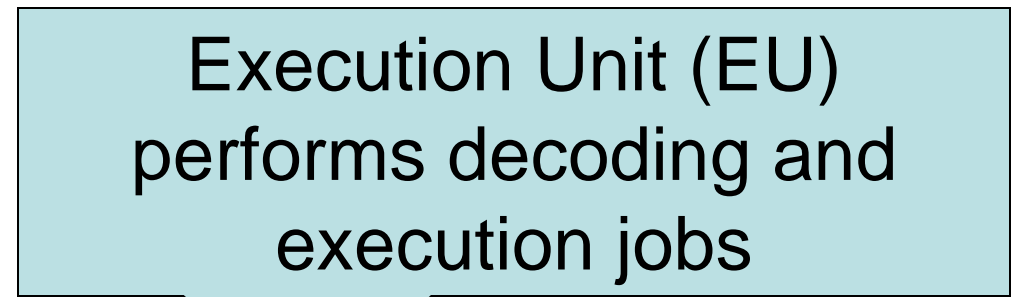
Hall ch 2 p/29

Brey 2.1



8085

Before a program is written or instruction investigated, internal configuration of the microprocessor must be known.



8086

Bus Interface Unit (BIU)
performs the fetching job

8086 INTERNAL ARCHITECTURE

EU registers

	15	8 7	0	
AX	AH		AL	Accumulator
BX	BH		BL	Base Register
CX	CH		CL	Count Register
DX	DH		DL	Data Register
	SP			Stack Pointer
	BP			Base Pointer
	SI			Source Index Register
	DI			Destination Index Register
	FLAGS			FLAGS Register

Some registers are general-purpose or multipurpose registers while some have special purposes

AX (ACCUMULATOR)

- A multipurpose Register
- The accumulator is also used for some special purposes such as multiplication, division, some of the adjustment instructions and IN/OUT I/O operations.
- Can be used as 16-bit register (AX), or as either of two 8-bit registers (AH and AL).

BX (BASE INDEX)

- A multipurpose Register
- Also used to hold offset address of a location in the memory.
- Can be used as 16-bit register (BX), or as either of two 8-bit registers (BH and BL).

CX (count)

- A multipurpose Register
- Also holds the count for various instructions that use a count
 - repeated string instructions REP, REPE, REPNE use CX
 - LOOP instruction uses CX
 - shift and rotate instructions use CL
- Can be used as 16-bit register (CX), or as either of two 8-bit registers (CH and CL).

DX (data)

- A multipurpose Register
- Also has some special use
 - holds a part of the result after multiplication
 - or part of the dividend before a division.
- Can be used as 16-bit register (DX), or as either of two 8-bit registers (DH and DL).

SP (Stack Pointer)

- Intel categorized it as a multipurpose Register
- But it is assigned a special function
 - it points to an area of memory called the stack.
 - The stack memory is a LIFO data structure.

BP (Base Pointer)

- A multipurpose Register
- Can also point to a memory location for memory data transfers.

DI (Destination index)

- A multipurpose Register
- Often addresses string destination data for the string instruction.

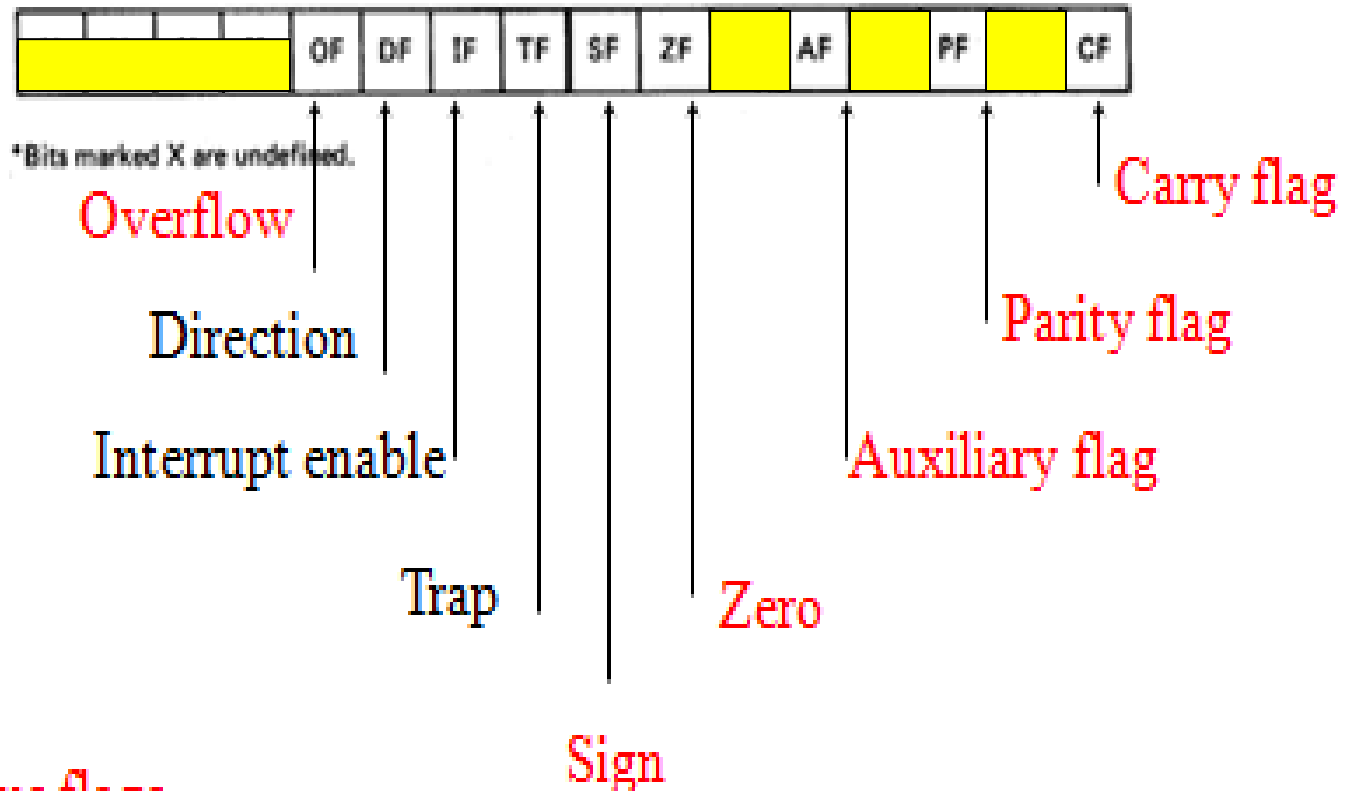
SI (Source index)

- A multipurpose Register
- Often used to the address source string data for the string instructions.

FLAGS

- A special purpose register
- Status flags Indicate the condition of the microprocessor
- Some of the flags are also used to control the microprocessor.
- Most arithmetic and logic operations change the status flags
- Status flags never change for any data transfer or program control operation.

Flags Register

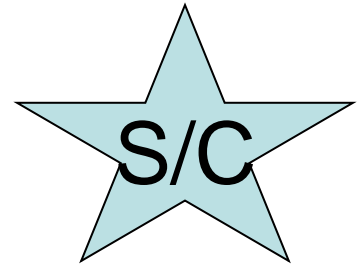


6 are status flags

3 are control flag

Undefined/unused flags

CF Carry Flag



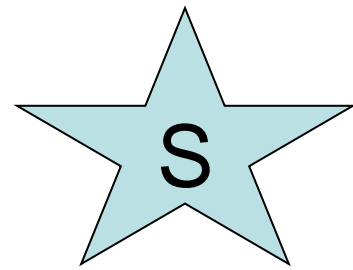
- It holds the carry after addition or the borrow after subtraction.
- The carry flag also indicates error conditions, as dictated by some programs and procedures.
- Also can be set, cleared and inverted with the STC, CLC or CMC instructions respectively.

PF Parity Flag



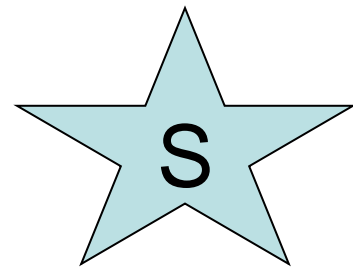
- **Parity** is the count of ones in accumulator AL or AX
 - expressed as even or odd.
 - Logic 0 for odd parity; logic 1 for even parity.
 - if a number contains no one bits, it has even parity
- Set by most instructions if the least significant eight bits of the destination operand contain an even number of 1 bits.
- Today this is seldom used, initially implemented to check data during communication. Today this is mostly done through external hardware rather than the μ P.

AF Auxiliary Flag



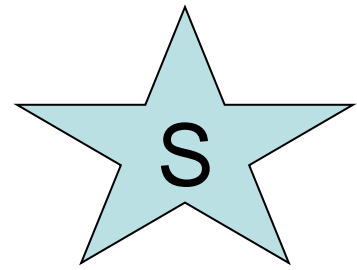
- The auxiliary carry holds the carry (half-carry) after addition or the borrow after subtraction between bits positions 3 and 4
- This highly specialized flag is used by DAA or DAS instructions during BCD operations.
- Otherwise this flag is not used by any other instruction or μP

ZF Zero Flag



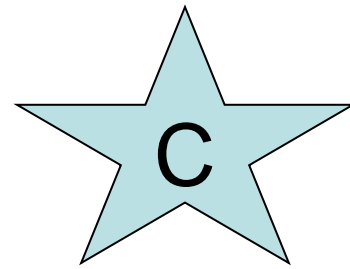
- The zero flag shows whether the result of an arithmetic or logical operation is zero or not.
 - When $Z = 1$, the result is zero.
 - When $Z = 0$, the result was non-zero.

SF Sign Flag



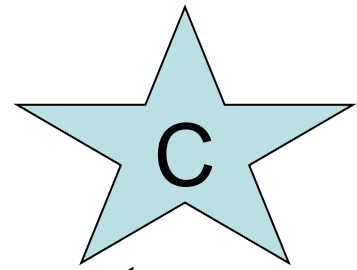
- The sign flag holds the arithmetic sign after an arithmetic or a logical operation.
- If $S = 1$ the sign bit is set and the result is negative.
- If $S = 0$, the sign bit is not set and the result is positive.

TF Trap / Trace Flag



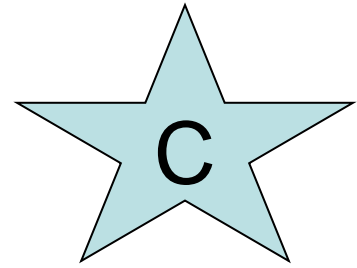
- It enables trapping through an on-chip debugging facility.
- When set to 1 the μP interrupts operation based on values set in the debugging register and the control registers.
- On being set it allows single-step through programs i.e., executes exactly one instruction and generates an internal exception

TF Trap / Trace Flag...



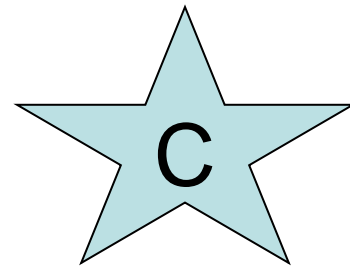
- Unlike other control flags it cannot be set or clear directly
 - indirect procedure:
 - by calling INT 1
- OR
- pushing flag register in stack using PUSHF
 - popping in any general purpose register, say AX
 - manipulating the T-bit with AND/OR operation
 - pushing it in the stack
 - popping to flag register using POPF

IF Interrupt Flag



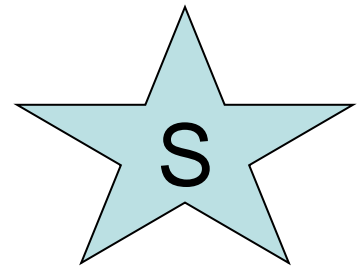
- The interrupt flag controls the operations of the INTR (Interrupt request) input pin.
 - If $I = 1$, the INTR pin is enabled;
 - if $I = 0$, the INTR pin is disabled.
- can be set by the STI (set I flag) instruction
- can be clear/reset by CLI (Clear I flag) instruction

DF Direction Flag



- Another control flag
- The flag selects either the increment or decrement mode for DI and/or SI registers during string instructions.
- If D=1 the registers are automatically decremented;
- if D =0 the registers are automatically incremented.
- set with the STD (set direction) instruction
- cleared with the CLD (clear direction) instruction.

OF Overflow Flag



- Overflow occurs when signed numbers are added or subtracted.
- An overflow indicates that the result has exceeded the capacity of the machine.
- For example if a 7FH (+127) is added using a 8 bit addition to a 01H (+1) the result is 80H(-128). The result represents an overflow condition indicated by the overflow flag for the signed addition.
- Most arithmetic instructions set this flag to indicate that the result was in error

BIU Registers

CS	CODE Segment
DS	DATA Segment
ES	EXTRA Segment
SS	STACK Segment
IP	INSTRUCTION POINTER

Segment Registers

- Generate memory addresses when combined with other registers in the microprocessor.
- Following is a list of each segment register, along with its function in the system.

CS Code Segment Register

- The code segment is a section of memory that holds the code (programs and procedures) used by the microprocessor.
- The code segment (**CS**) register defines the starting address of the section of memory holding code.
- The code segment size is limited to max 64KB

DS Data Segment Register

- The data segment contains most data used by a program.
- The data segment (**DS**) register defines the starting address of the section of memory holding data.
- The data segment size is limited to max 64KB
- Data are accessed in the data segment by an offset address or the contents of other registers that hold the offset address.

ES Extra Segment Register

- **ES (extra)** is an additional data segment used by some of the string instructions to hold destination data.
- The extra segment (**ES**) register defines the starting address of the section of memory used as extra segment
- Can be used as an extra/additional segment for code or data
- Max size of the segment is limited to 64KB

SS Stack Segment Register

- The stack segment defines the area of memory used for the stack.
- The stack segment (**SS**) register defines the starting address of the section of memory used as stack
- The stack entry point is determined by the stack segment and stack pointer registers.
- The BP registers also addresses data within the stack segment.

IP Instruction Pointer

- Special–Purpose Registers
- The Instruction pointer, which points to the next instruction in a program, is used by the microprocessor to find the next sequential instruction in a program located within the code segment.
- The instruction pointer can be modified with a jump or a call instruction.

8086 signals

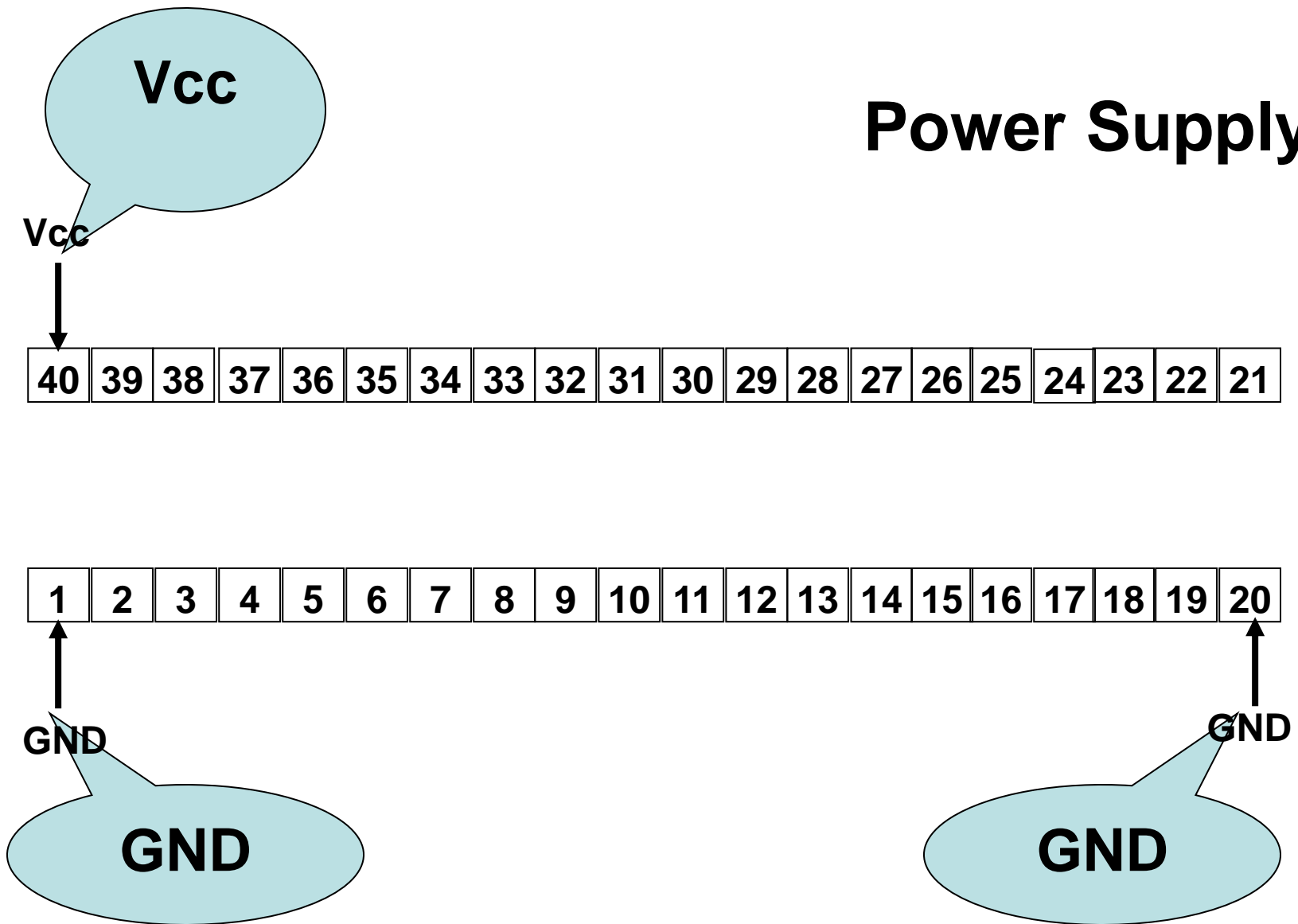
Brey

9.1

Rafiq

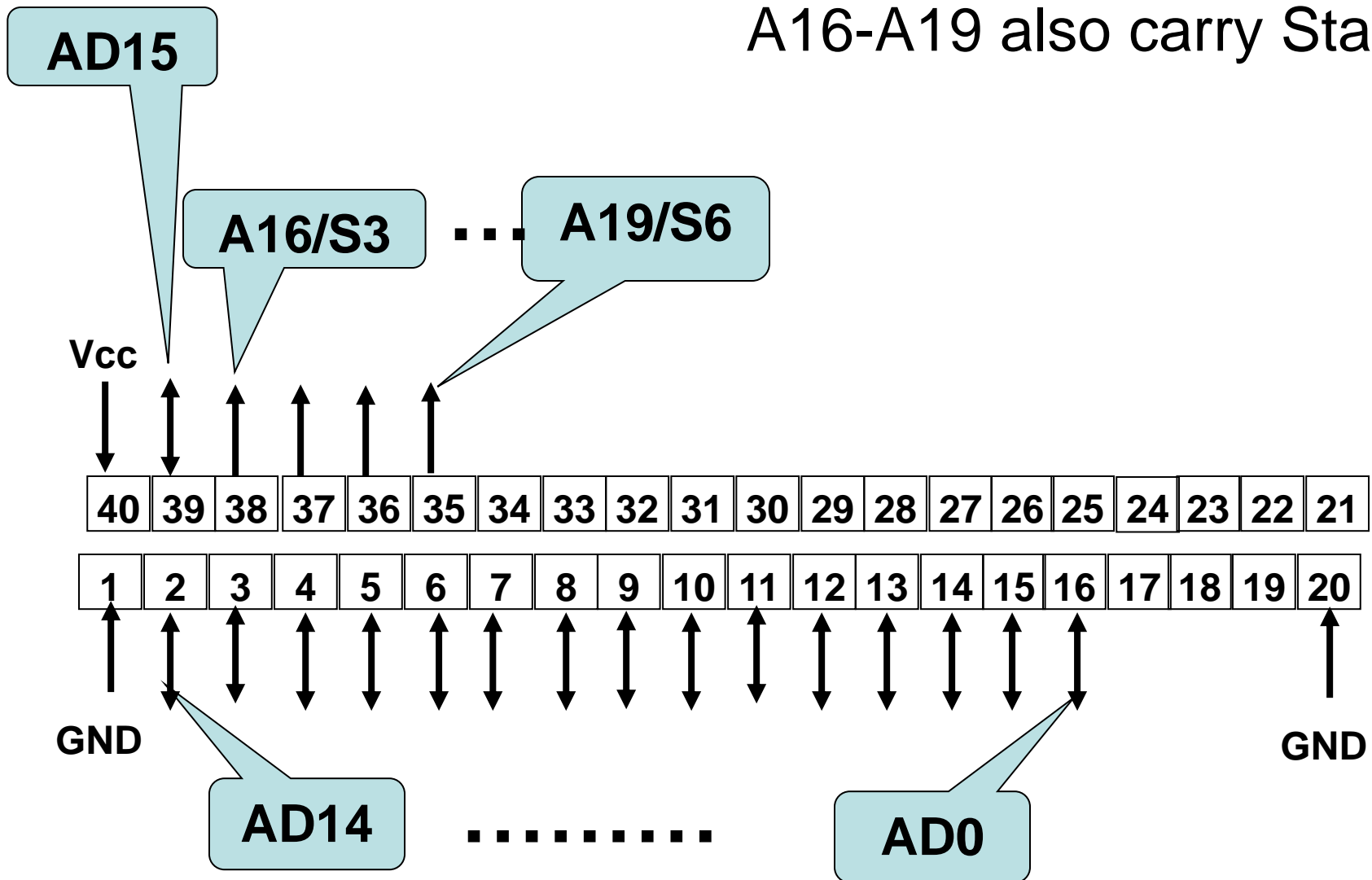
3.8.1

Power Supply

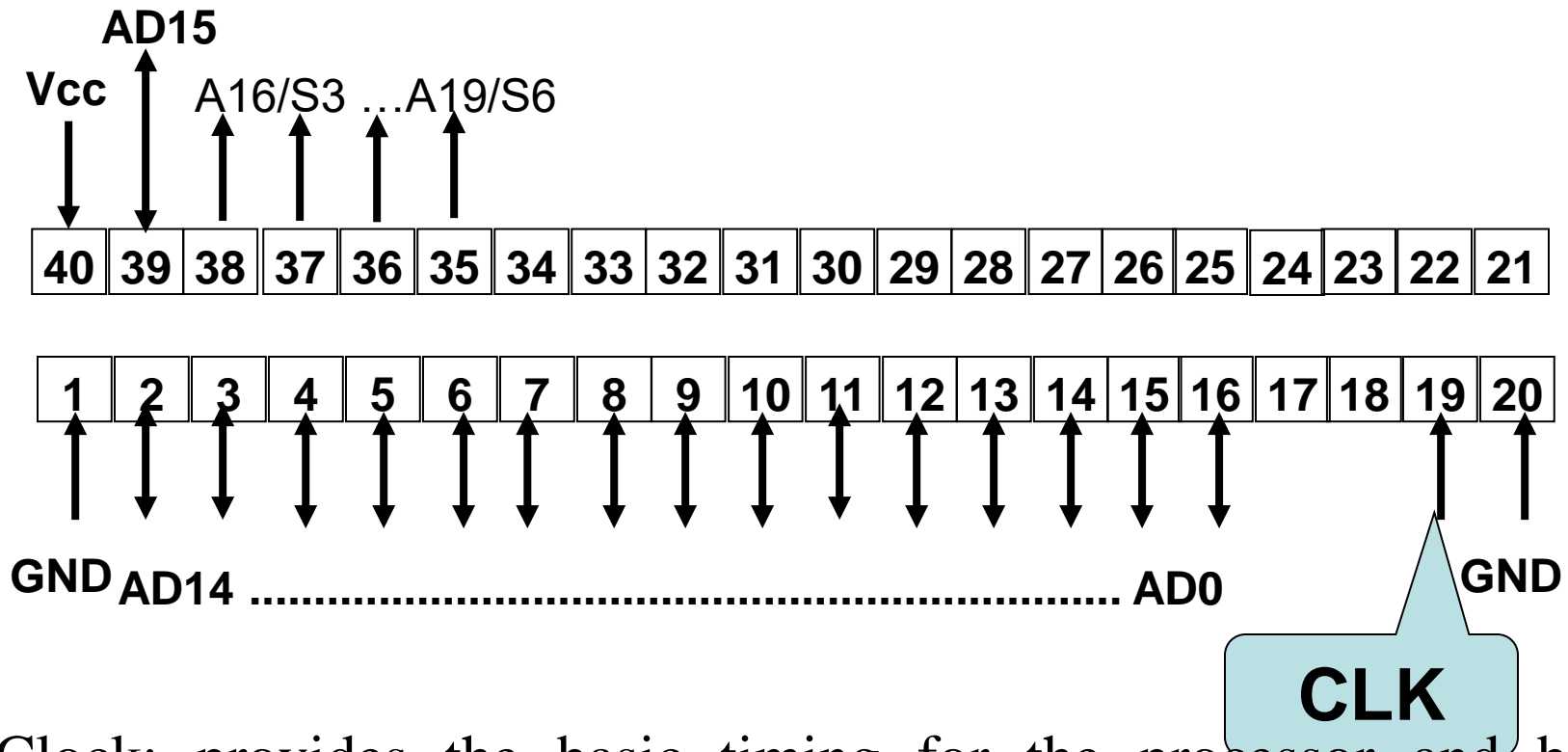


Address/Data Bus

A16-A19 also carry Status

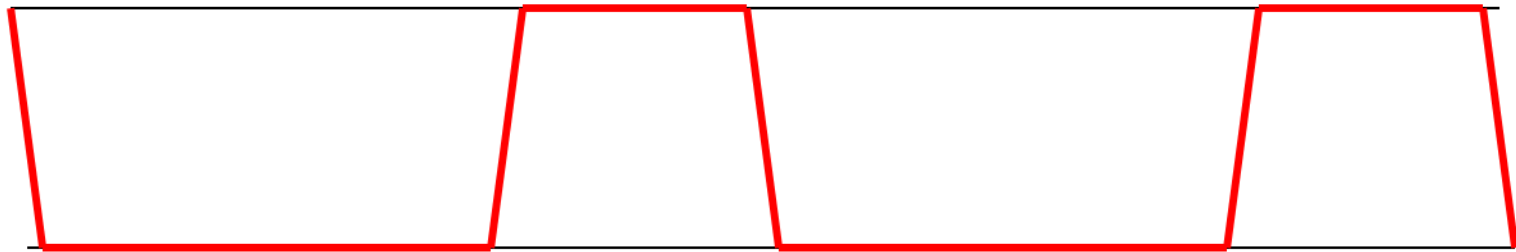


Control Signals --- Clock



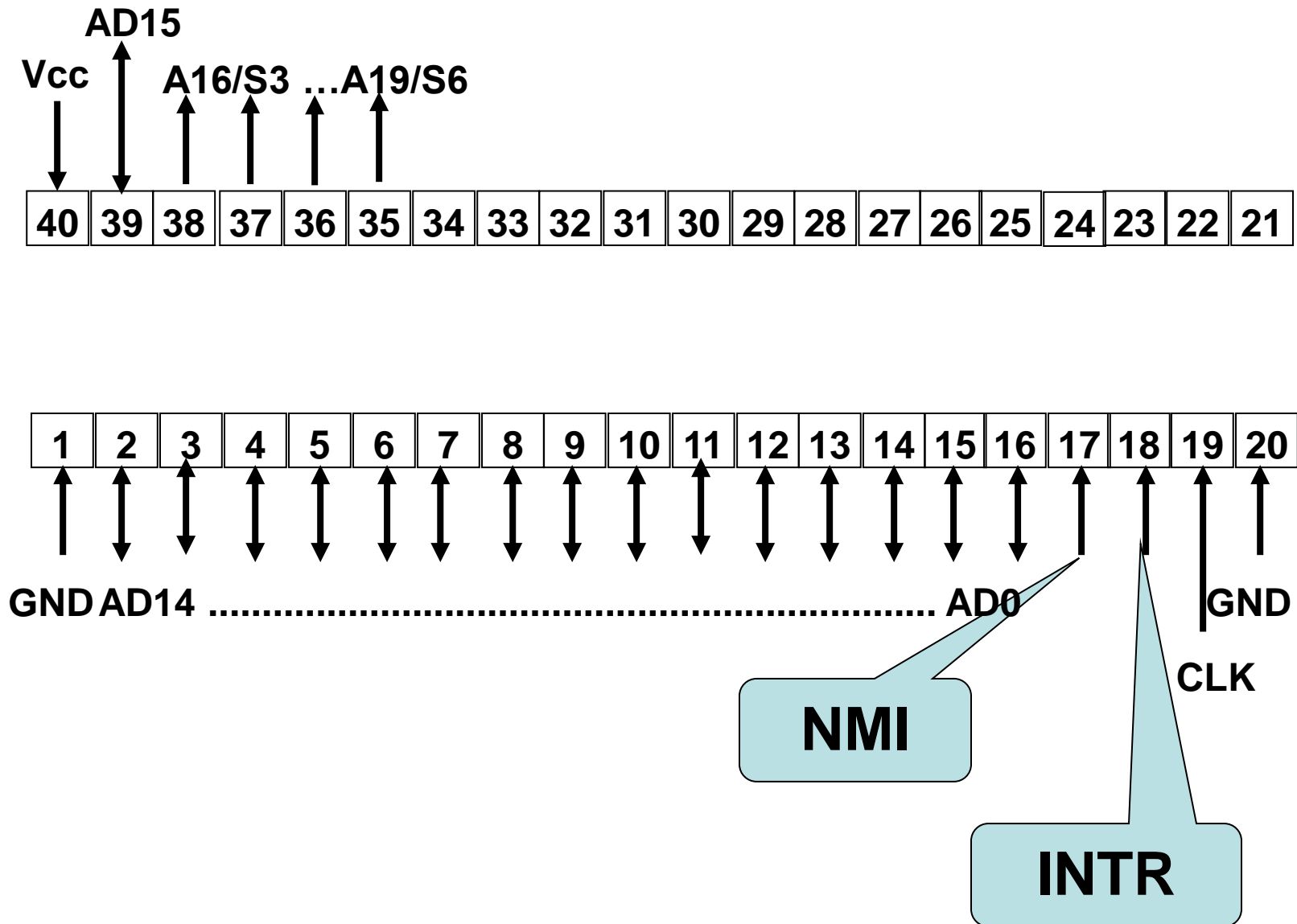
Clock: provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing. 8086 is found to operate in 5-, 8- and 10-Mhz.

System Clock



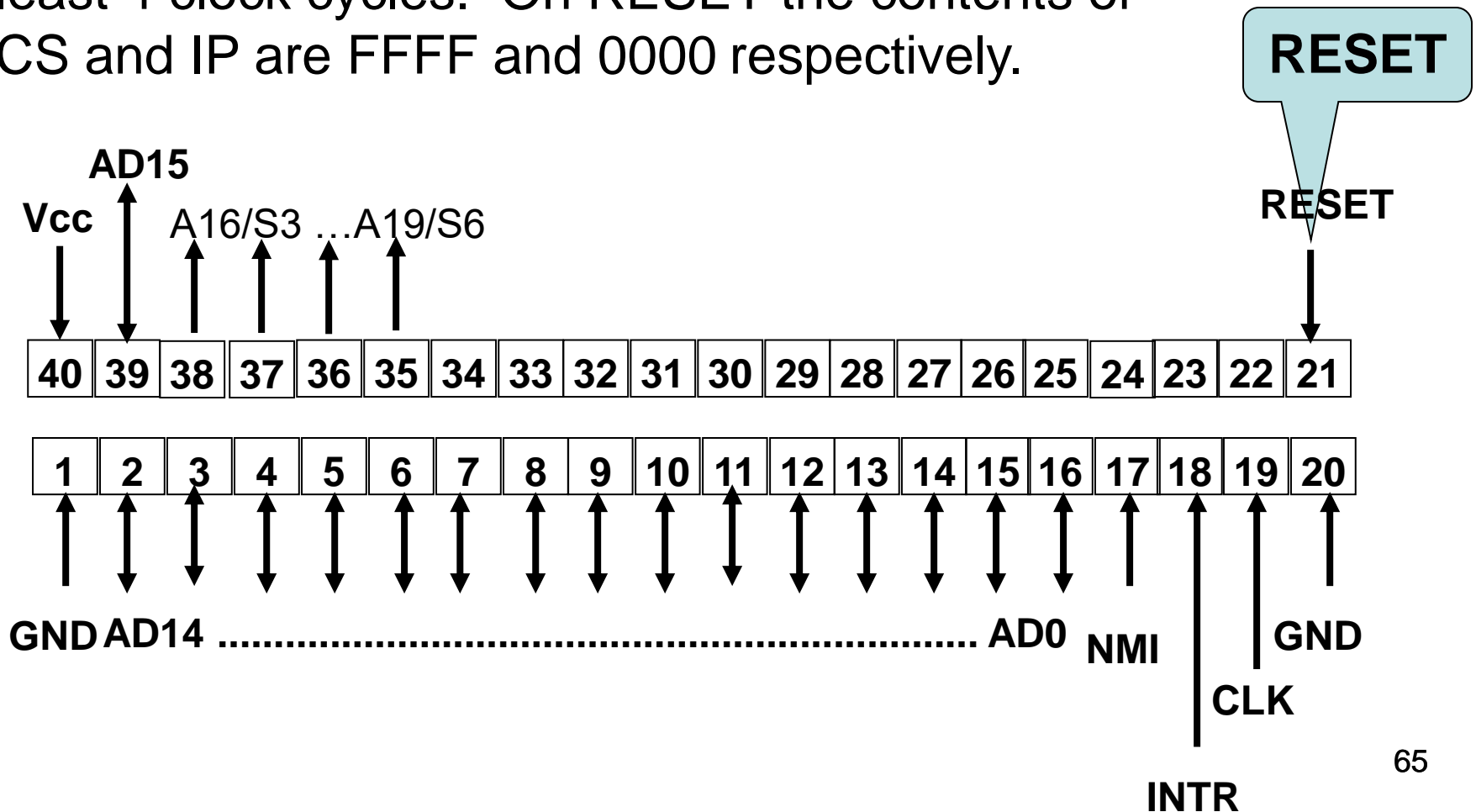
33% Duty Cycle

Control Signals ---Interrupts



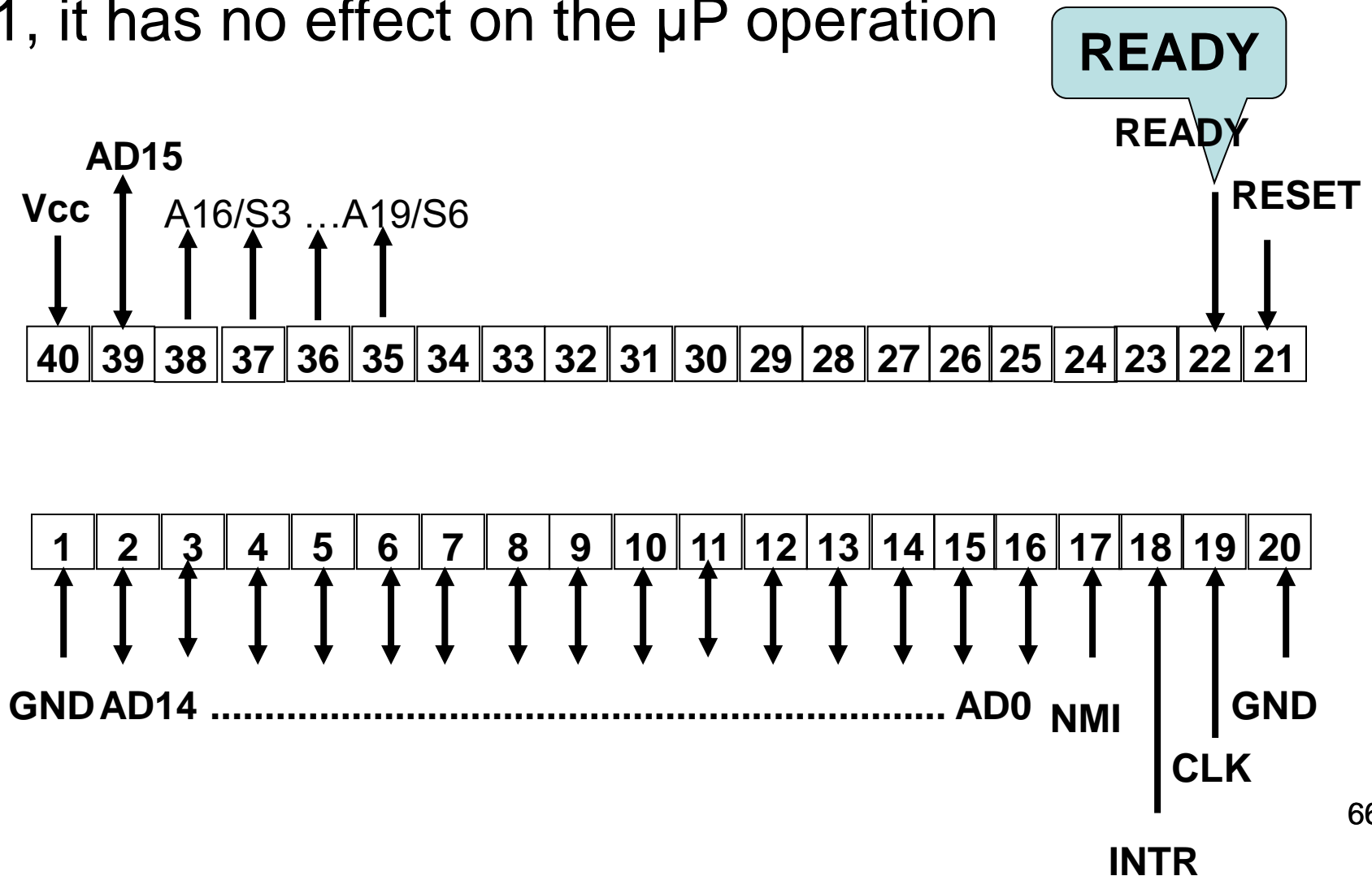
Control Signals --- RESET

causes the processor to immediately terminates its present activity. The signal must be HIGH for at least 4 clock cycles. On RESET the contents of CS and IP are FFFF and 0000 respectively.



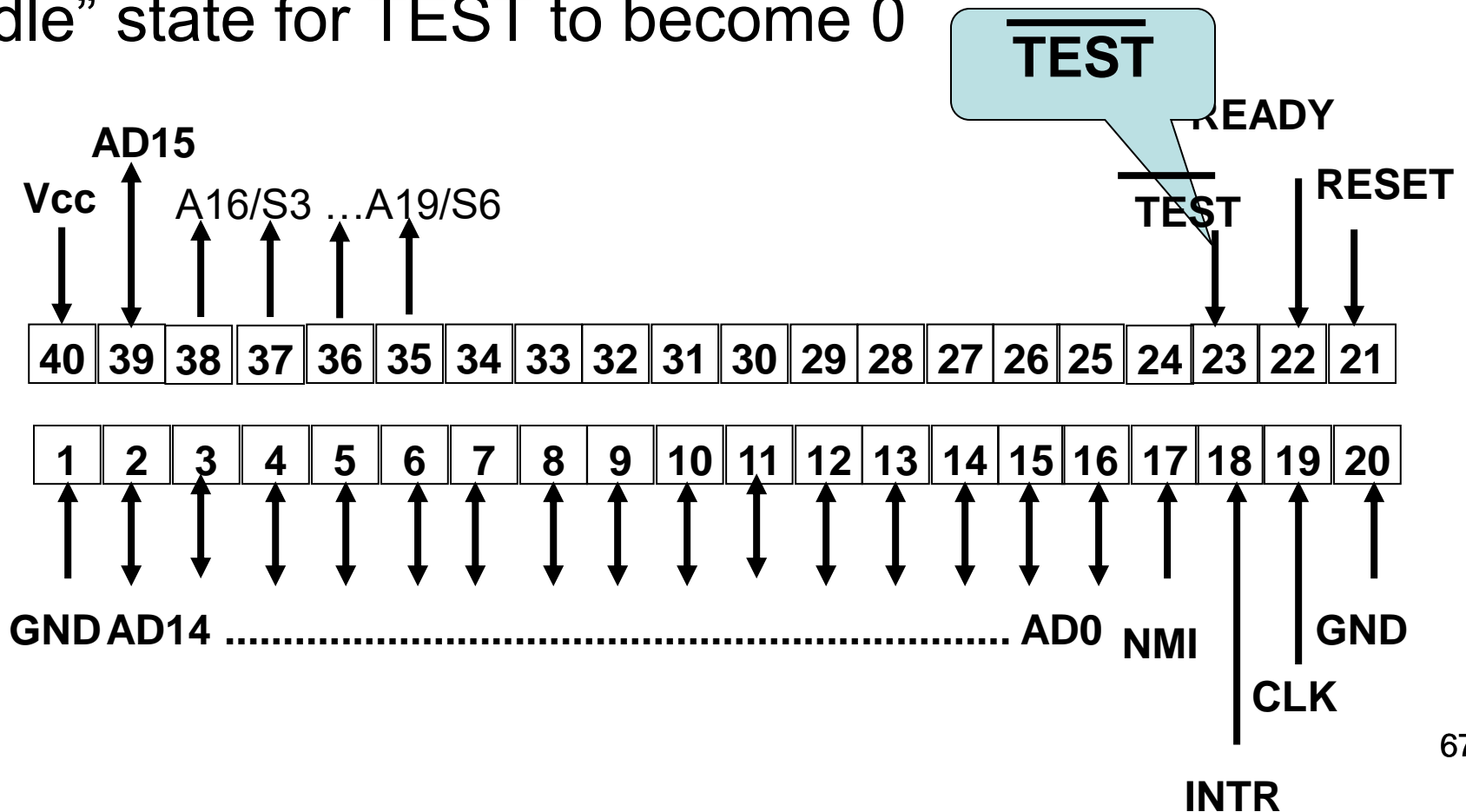
Control Signals --- READY

If READY is 0, μ P enters into WAIT states remain idles
if 1, it has no effect on the μ P operation

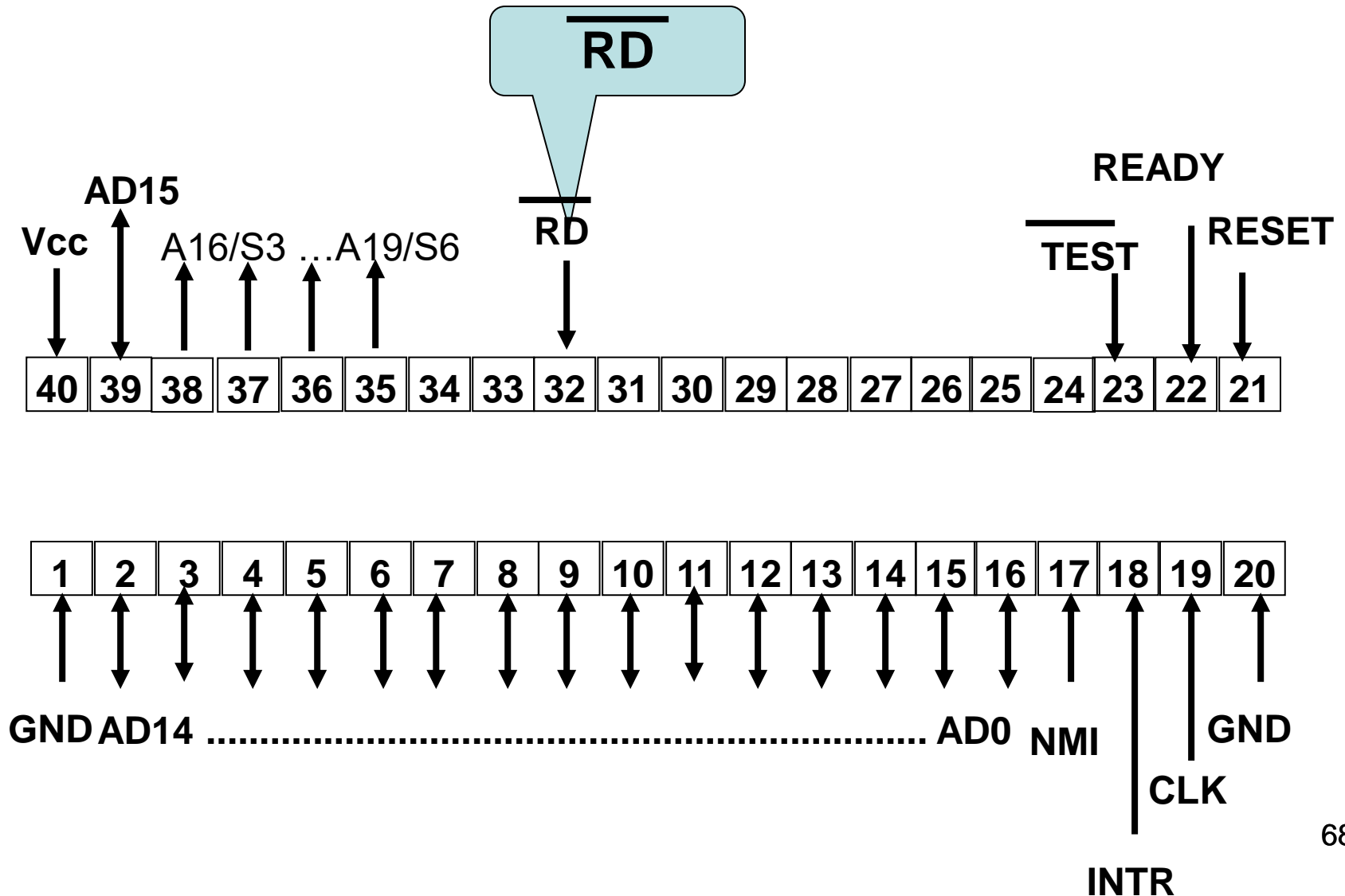


Control Signals --- TEST

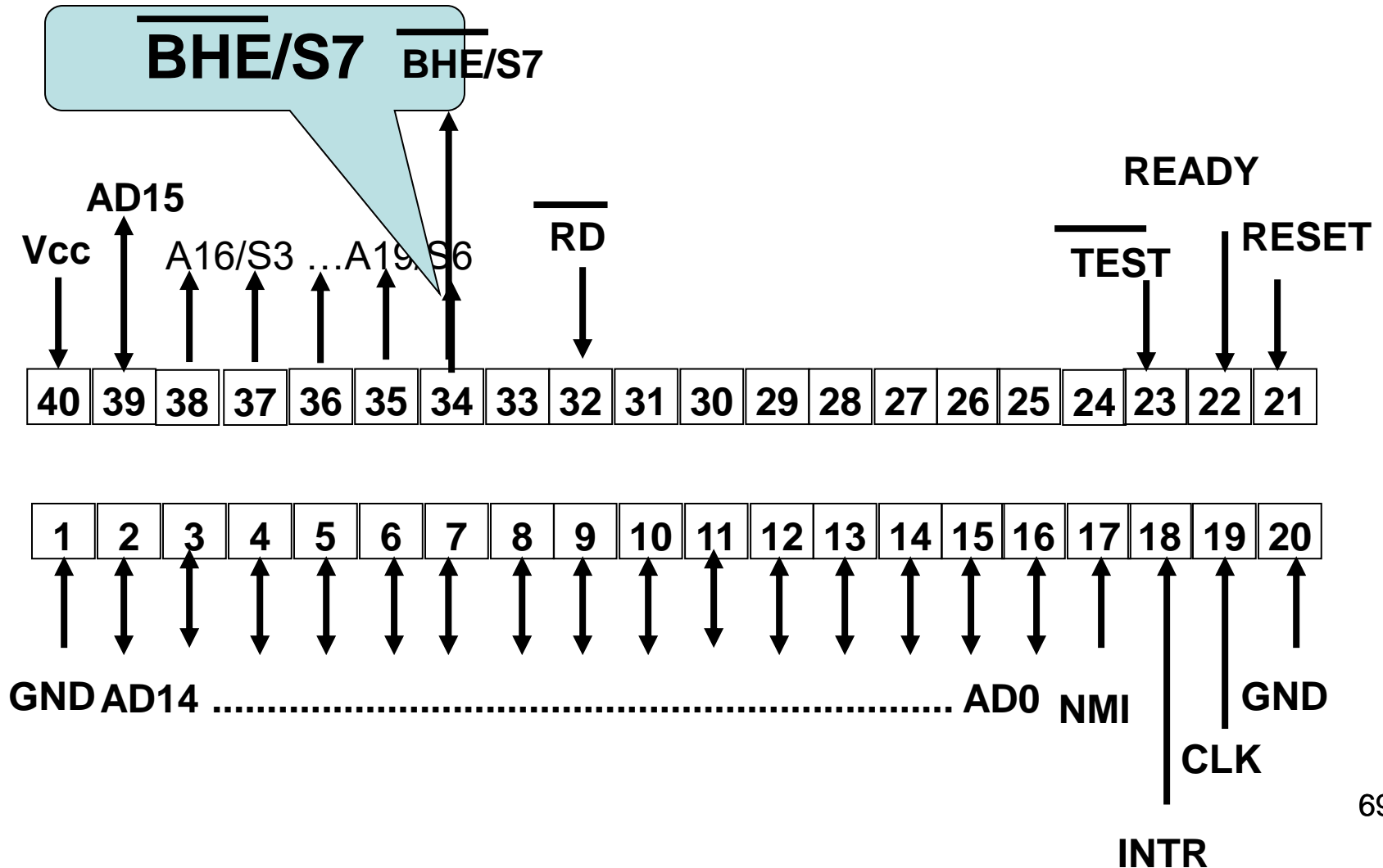
used by WAIT instruction. If the input is LOW WAIT instruction functions as NOP and execution of the program continues, otherwise processor waits in an “idle” state for TEST to become 0



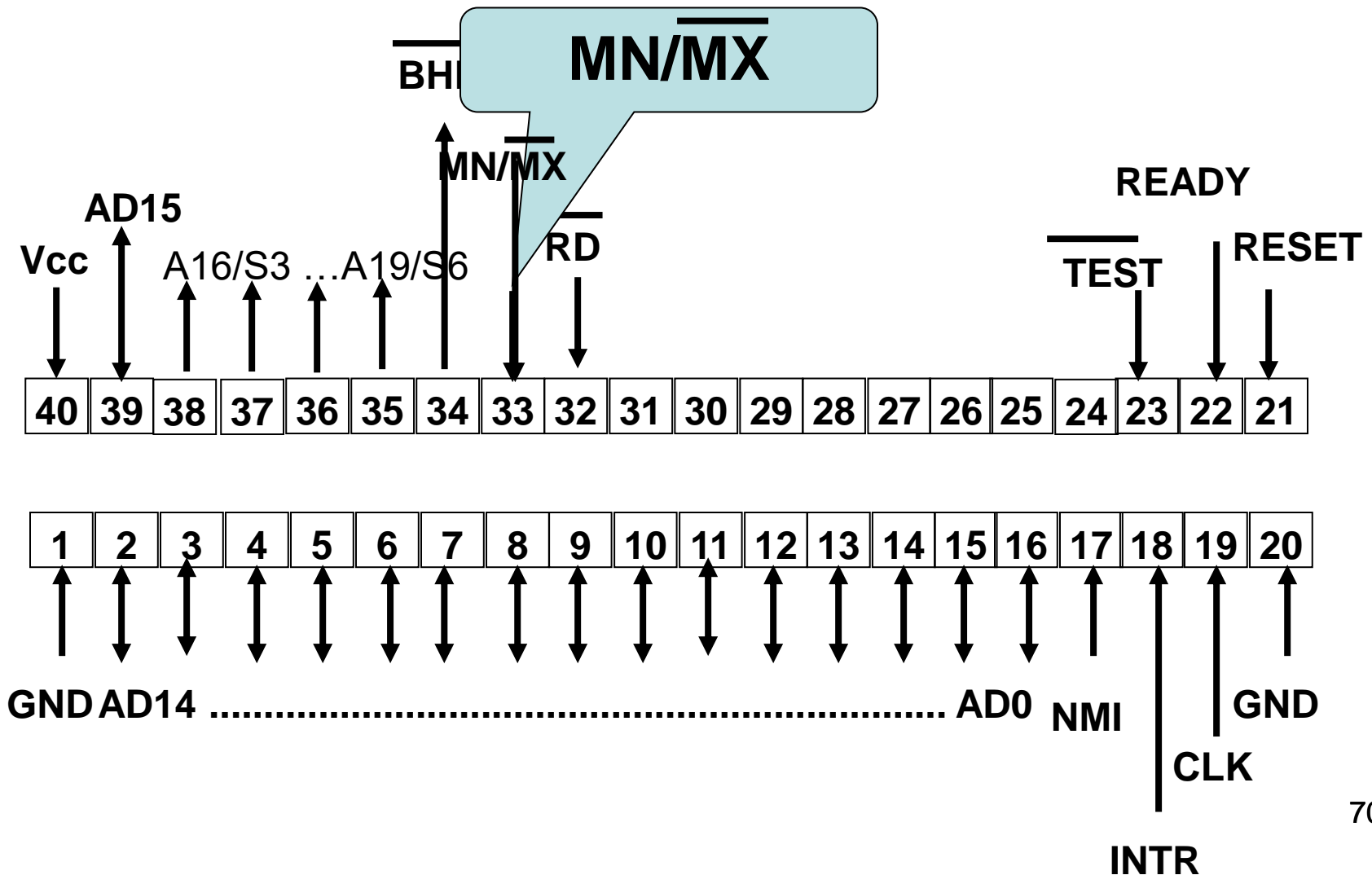
Control Signals --- RD



Control Signals --- BHE/S7



Control Signals --- MN/MX



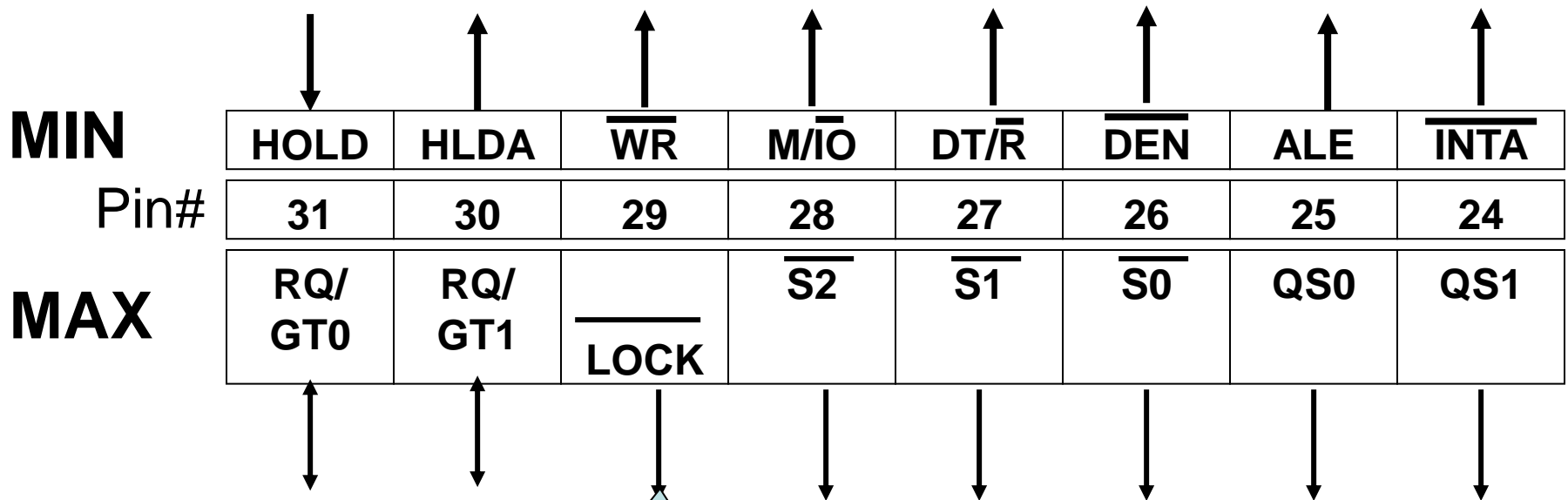
Modes of operation of 8086

- **Minimum mode**

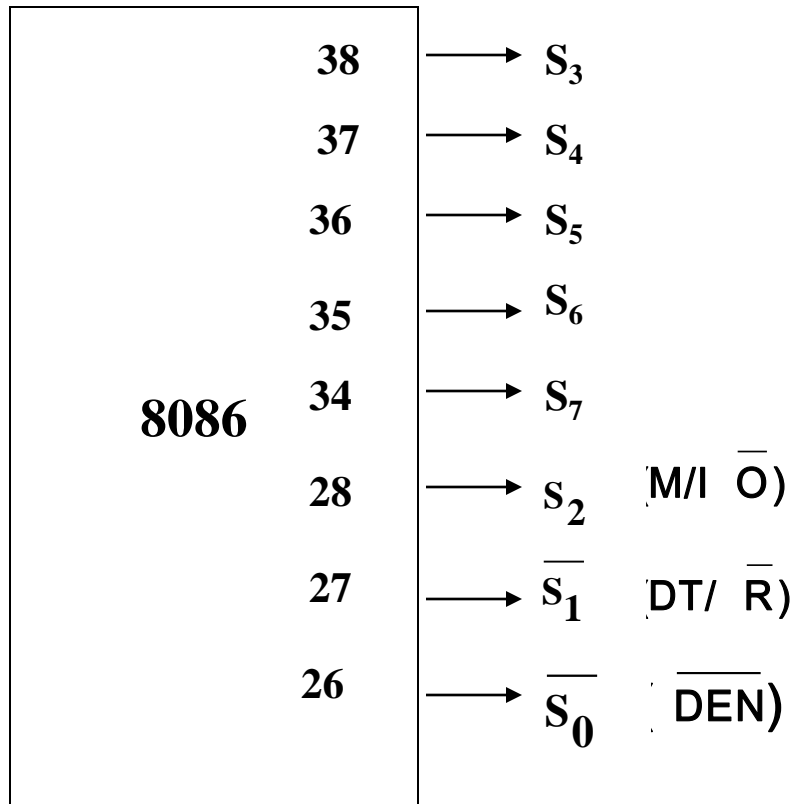
- single processor system,
- only one 8086 in the system
- generally small systems
- set by applying '1' in $\overline{MN}/\overline{MX}$ pin

- **Maximum mode**

- Multi processor system,
- more than one 8086 in the system
- generally large systems
- set by applying '0' in $\overline{MN}/\overline{MX}$ pin



Status Pins S_0 - S_7



Bus cycle status

$\overline{s_2}$	$\overline{s_1}$	$\overline{s_0}$	
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code Access
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Inactive

This information indicates which relocation register is presently being used for data accessing.

S4	S3	Function
----	----	----------

0	0	ES
---	---	----

0	1	SS
---	---	----

1	0	CS or no segment
---	---	---------------------

1	1	DS
---	---	----

S5	indicates the status of I-flag
----	-----------------------------------

S6	low indicates that μ P is on the bus tristates it during HLDA
----	---

S7	spare status bit, always 1
----	-------------------------------

Instruction Queue Status

QS1 QS0

0	0	No operation
0	1	First byte of op code from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

8086 Pin diagram

Maximum Mode

Minimum Mode

GND	1	40	VCC	
AD14	2	39	AD ₁₅	
AD13	3	38	A ₁₆ /S ₃	
AD12	4	37	A ₁₇ /S ₄	
AD11	5	36	A ₁₈ /S ₅	
AD10	6	35	A ₁₉ /S ₆	
AD9	7	34	$\overline{\text{BHE}}/\text{S}_7$	
AD8	8	33	MN/ $\overline{\text{MX}}$	
AD7	9	32	$\overline{\text{RD}}$	
AD6	10	31	$\overline{\text{RQ}}/\overline{\text{GT}}_0$	(HOLD)
AD5	11	30	$\overline{\text{RQ}}/\overline{\text{GT}}_1$	(HLDA)
AD4	12	29	$\overline{\text{LOCK}}$	($\overline{\text{WR}}$)
AD3	13	28	$\overline{\text{S}}_2$	($\overline{\text{MIO}}$)
AD2	14	27	$\overline{\text{S}}_1$	($\overline{\text{DT/R}}$)
AD1	15	26	$\overline{\text{S}}_0$	($\overline{\text{DEN}}$)
AD0	16	25	QS ₀	(ALE)
NMI	17	24	QS ₁	($\overline{\text{INTA}}$)
INTR	18	23	$\overline{\text{TEST}}$	
CLK	19	22	READY	
GND	20	21	RESET	

8086

8086

Memory Organization

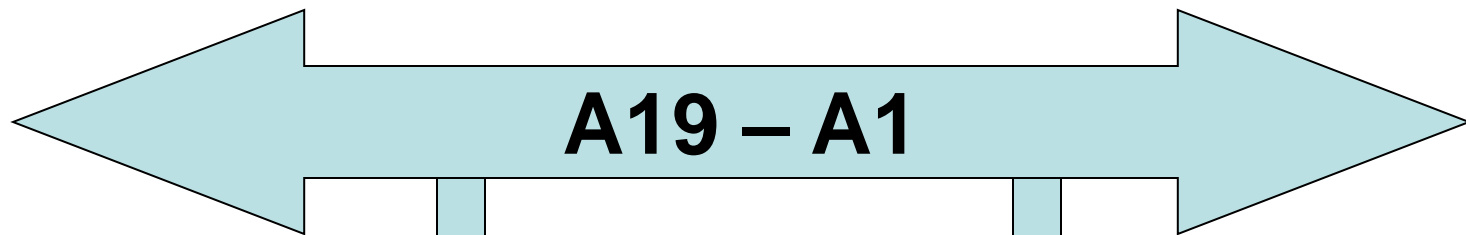
Brey	Ch 2
Rafiq	Ch 3
Hall	Ch 2

Memory Organization of 8086

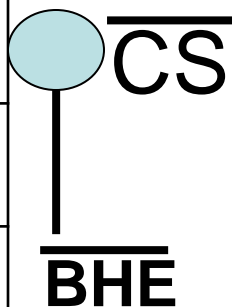
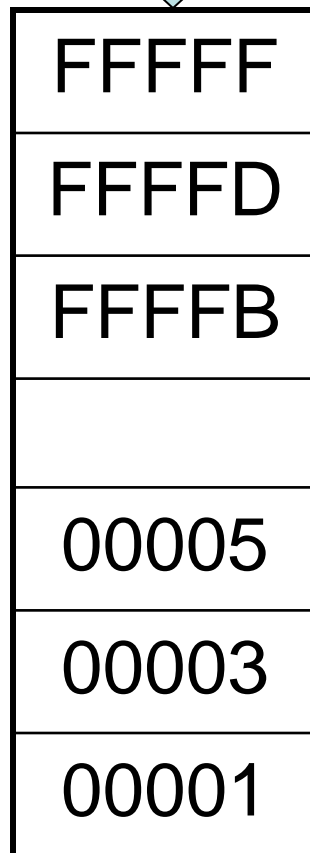
- 8086 has 20-bit address line to address up to 1MB memory space which is organized as a linear array with address from 00000H to FFFFFH
- Physically memory is organized as
 - a high or odd bank and
 - a low or even bankof 512KB addressed in parallel.

Memory Organization of 8086...

- Byte data with even addresses is transferred on the $D_7 - D_0$ bus lines while odd addressed data is transferred on the $D_{15} - D_8$ bus lines.

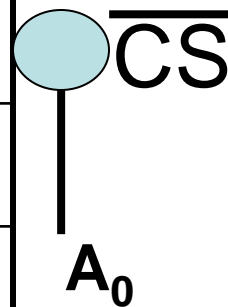
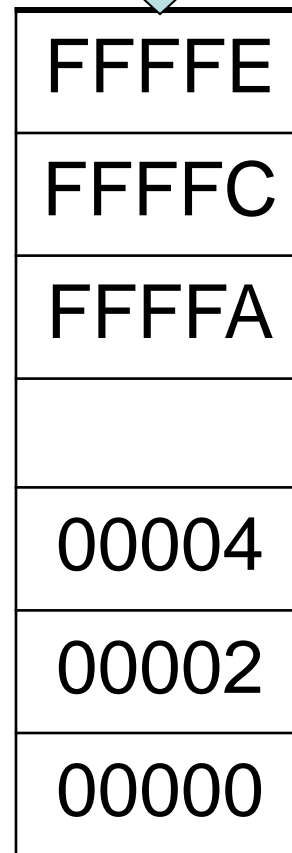


high
or odd
bank



D15 – D8

low or
even
bank



D7 – D0

Memory Organization of 8086...

- 8086 can access either
 - an odd byte location, or
 - an even byte location or
 - both even and odd byte locationsby using two signals **$\overline{\text{BHE}}$** and **A_0**

Memory Organization of 8086...

$\overline{\text{BHE}}$	A0	Accessed bank	Data bits
0	0	Both banks	All 16 bits (D0-D15)
0	1	Odd or high bank	High order 8 bits (D8-D15)
1	0	Even or low bank	Low order 8 bits (D0-D7)
1	1	None	---

Memory Access by 8086

- if the given address is even, reads 16-bit at a time
- if the given address is odd, reads only odd byte
- If 16-bit data is stored from odd address,
 - it needs two cycles to read or write data from/to memory

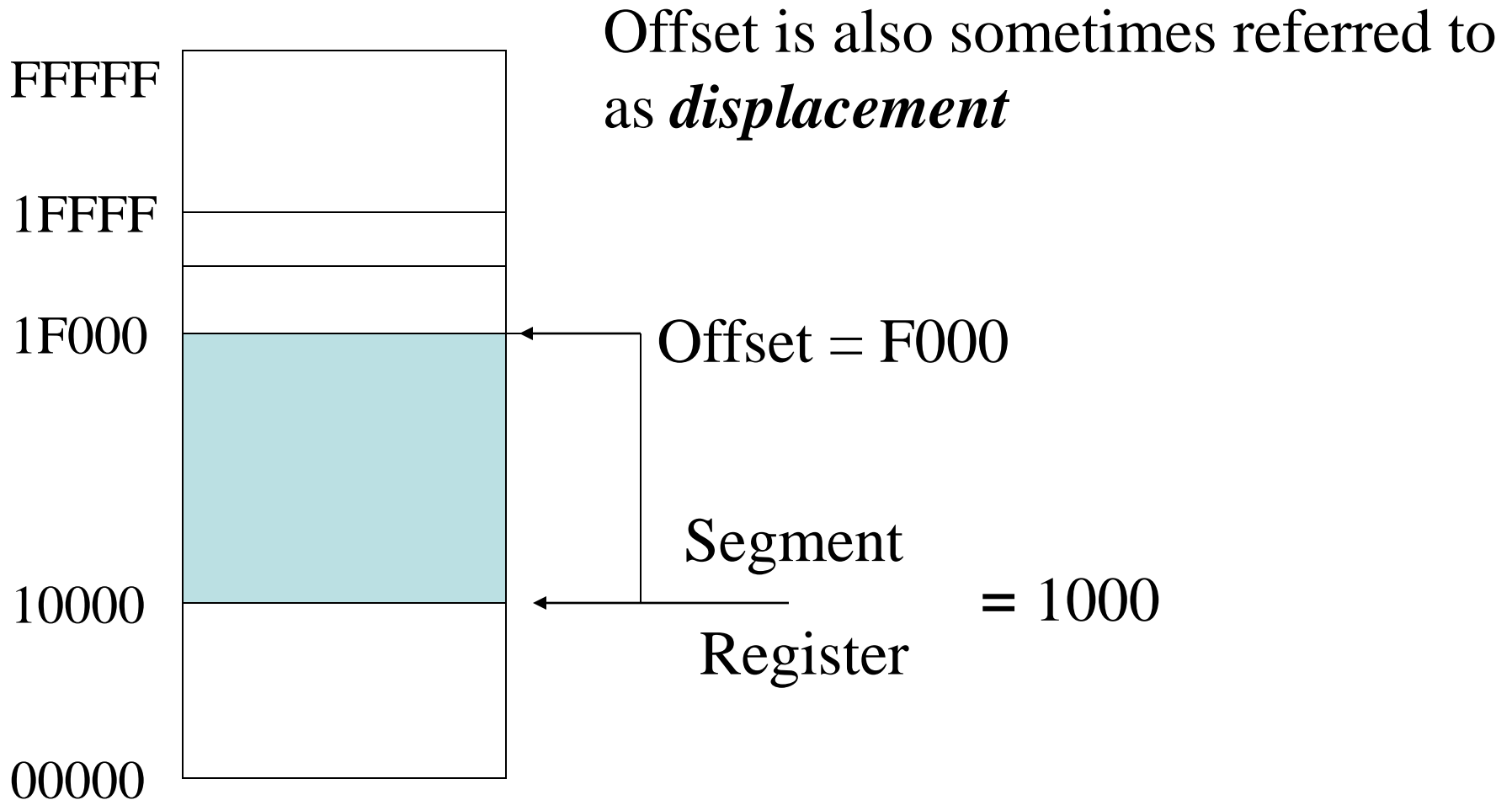
8086 Memory Addressing

- As said earlier Intel do not view this 1MB memory as a flat address from 00000h to FFFFFh
- Rather they use a concept known as **Segment:Offset** concept
- The entire 1M memory is divided into **segments** of max^m size 64KB
- Memory segments can touch or even overlap if 64K bytes of memory are not required for a segment.

Segment and Offsets

- Combination of a **segment address** and an **offset address** access a memory location.
- The **segment address** located in one segment register defines the beginning address of any 64K-byte memory segment.
- The **offset address** is also held in a register and selects any location within the 64K byte memory segment.

8086 Memory Addressing...



8086 Memory Addressing...

- The **segment register** in the previous example contained 1000H yet it addresses a segment starting at location 10000H.
- the segment register is always **appended** with a **0H** on its rightmost end.
- Because of the append, segments can begin only at 16-byte boundary in the memory system.
 - This 16-byte boundary is called a **paragraph**.

8086 Memory Addressing...

- A memory location pointed by a segment address of 1000H and offset address of 2000H is written as 1000:2000.
- What is the actual/physical memory location for 1000:2000?

8086 Memory Addressing...

Calculation of physical address from
segment : offset pair 1000:2000

1. Append 0H to the right of
segment value

1 0 0 0 **0**

2. Add offset value with the
appended segment value

2 0 0 0

3. The result is the physical
address for the given
segment : offset pair

1 2 0 0 0

Default Segment and Offset Registers

- 8086 has a set of rules that apply to segments whenever memory is addressed.
- These rules define the segment and offset register combination

Default Segment and Offset Register combinations...

Segment	Offset	Purpose
CS	IP	Code / Instruction address
DS	BX, DI, SI, 8-bit or 16-bit number	Data address
SS	SP or BP	Stack address
ES	SI	String Source address

8086 Memory Addressing...

Advantages of segmentation

- Segment and offset addressing scheme seems complicated.
- But, it has its advantages as well.
 - Allows **Relocation** of
 - ✓ **program** and
 - ✓ **data**

8086 Memory Addressing...

- This is ideal for use in general-purpose computer system in which not all machines contain the same memory areas.
- The structure of the personal computer memory structure is different from machine to machine requiring relocateable software and data.

Segment and Offset Addressing Scheme Allows Relocation

- A **relocatable program** is one that can be placed into any area of memory and **executed without any change.**
- **Relocatable data** are data that can be placed in any area of memory and used **without any change to the program.**

8086 Memory Addressing...

- Because memory is addressed within a segment by an offset address, the memory segment can be moved to any place in the memory system without changing any of the offset addresses.
- Only the contents of the segment register must be changed to address the program in the new area of memory.

Watch out!!!

- 8086 produces a 20-bit external or effective or physical address from the 32-bit segment:offset pair.
- As a result, each external address can be referred to by $2^{12} = 4096$ different segment:offset pairs.

8086 Addressing Modes

Rafiq

3.3

Brey

ch 3

Categories of addressing modes

Addressing modes of 8086 can be divided into the following categories:

1. addressing Data
2. addressing Program codes in memory
3. addressing Stack in memory
4. addressing I/O
5. Implied addressing

1. addressing Data

- I. Immediate addressing
- II. Direct addressing
- III. Register [direct] addressing
- IV. Register indirect addressing
- V. Base-plus-index addressing
- VI. Register relative addressing
- VII. Base-relative-plus-index addressing

1. addressing Data (contd...)

i) Immediate addressing

- Data is immediately given in the instruction

`MOV AL, 90`

1. addressing Data (contd...)

ii) Direct addressing

- Data address is directly given in the instruction

```
MOV AX, [12345]  
MOV BX, DATA
```

1. addressing Data (contd...)

iii) Register [direct] addressing

- data is in a register (here BX register contains the data)

MOV AX, BX

1. addressing Data (contd...)

iv) Register indirect addressing

- Register supplies the address of the required data

MOV CX, [BX]

1. addressing Data (contd...)

v) Base-plus-index addressing

MOV DX, [BX+DI]

- Base register is either BX or BP
- Index register is either DI or SI

1. addressing Data (contd...)

vi) Register relative addressing

MOV AX, [BX+1000]

- Register can be a base (BX, BP) or an index register (DI, SI)
- Mainly suitable to address array data

1. addressing Data (contd...)

vii) Base-relative-plus-index addressing

MOV AX, [BX+DI+10]

- suitable for array addressing

2. addressing Program codes in memory

- used with JMP and CALL instructions
- 3 distinct forms:
 - direct
 - indirect
 - relative

addressing Program codes ... direct

- address is directly given in the instruction

JMP 1000:1000	JMP doagain
CALL 1000:1000	CALL doagain

- often known as *far* jump or *far* call

addressing Program codes ... indirect

- address can be obtained from
 - a) any GP registers (AX,BX,CX,DX,SP,BP,DI,SI)
 - b) any relative registers ([BP],[BX],[DI],[SI])
 - c) any relative register with displacement

JMP AX

CALL BX

JMP TABLE[BX]

CALL SUBPROG[BX]

addressing Program codes ... relative

- means relative to instruction pointer (IP)
 JMP [20]
- skips over the next 20 bytes of memory that follow the JMP instruction
- [20] is known as *displacement* which could be 1 byte or 2 byte long
- 1 byte of displacement is known as *short* jump or call
- 2 byte of displacement is known as *near* jump or call

3. addressing Stack in memory

- PUSH and POP instructions are used to move data to and from stack.
- CALL also uses the stack to hold return address for procedure

4. addressing I/O

- IN and OUT instructions are used to address I/O ports
- could be *direct addressing*
IN AL, 05
- or *indirect addressing*
OUT DX, AL ; DX contains the
address of I/O port
- only DX can be used to point a I/O port

5. Implied addressing

CLC ; clear carry flag

- NO explicit address is given with the instruction
- implied within the instruction itself

8086 INSTRUCTION SET

Instruction Set

- You have studied it in CSE214. So we will skip it here.
- Includes equivalents of 8085 instruction set plus many new ones
- Has approx 117 different instructions
 - with about 300 different op codes

Instruction Set (cont...)

- Instructions contain
 - no operand, **CLC, CLD, CLI, NOP**
 - single operand, **INC AX** and
 - two operands **MOV AX, BX**
- Except for string instructions which involve array operations, 8086 does **not permit memory-to-memory** operations

8086 Instruction Format

Rafiq 3.5

Could be 1 byte to 6 bytes in length in machine code

Byte

1

2

3

4

5

6

Opcode	Extension of opcode	Displacement		Immediate data	
		Low byte	High byte	low byte	high byte

8086 Instruction Format

- **1 byte** Mostly no operand instructions
 - NOP, POPF, PUSHF, INC AX
- **2 byte** most of the instructions are 2 byte long
 - MOV AX, BX
 - MOV AL, 90
- **3 byte**
 - MOV AX, 1234

8086 Instruction Format...

- **4 byte**
 - CMP BX, 0504 81 FB 04 05
 - MOV [500],AL 88 06 00 05
 - ADD CX,10 81 C1 10 00
- **5 byte** inter segment CALL or JMP
 - JMP 2000:3000, EA 0030 0020
 - CALL 1000:2000
- **6 byte** immediate data to memory
 - MOV START, 1234
 - MOV [1234], 1234

8086 INTERRUPT

✓ Brey

Ch12 Interrupts

✓ Rafiq

3.9 Interrupts

✓ Hall

Ch 8

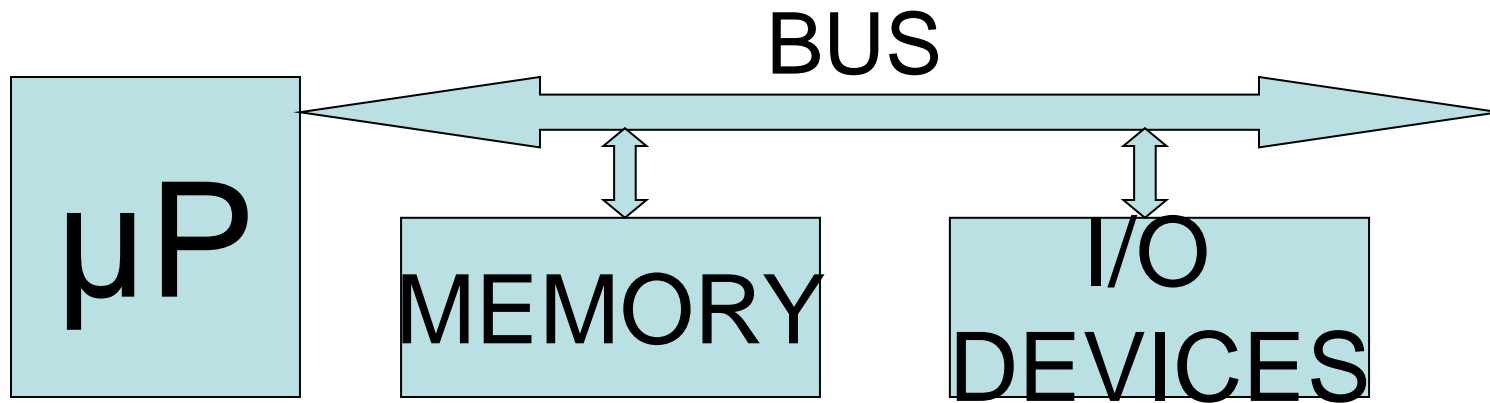
What is an interrupt?

- capability to suspend the execution of running program and execution of another program to fulfill specific requirement upon request
- after finishing the second program, automatically return to the first program and start execution from where it was left

Why interrupt capability is necessary?

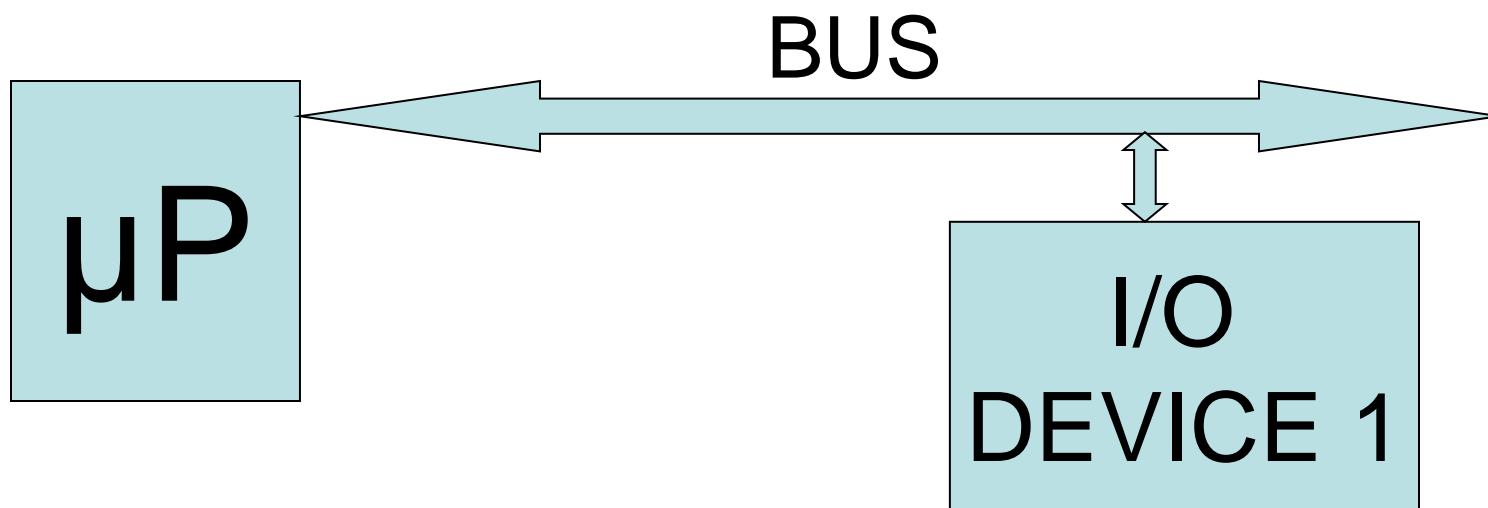
- To understand this we will have to review how $\mu\text{P}/\mu\text{C}$ communicate with the outside world.

Basic block diagram of a μ C



First type of communication

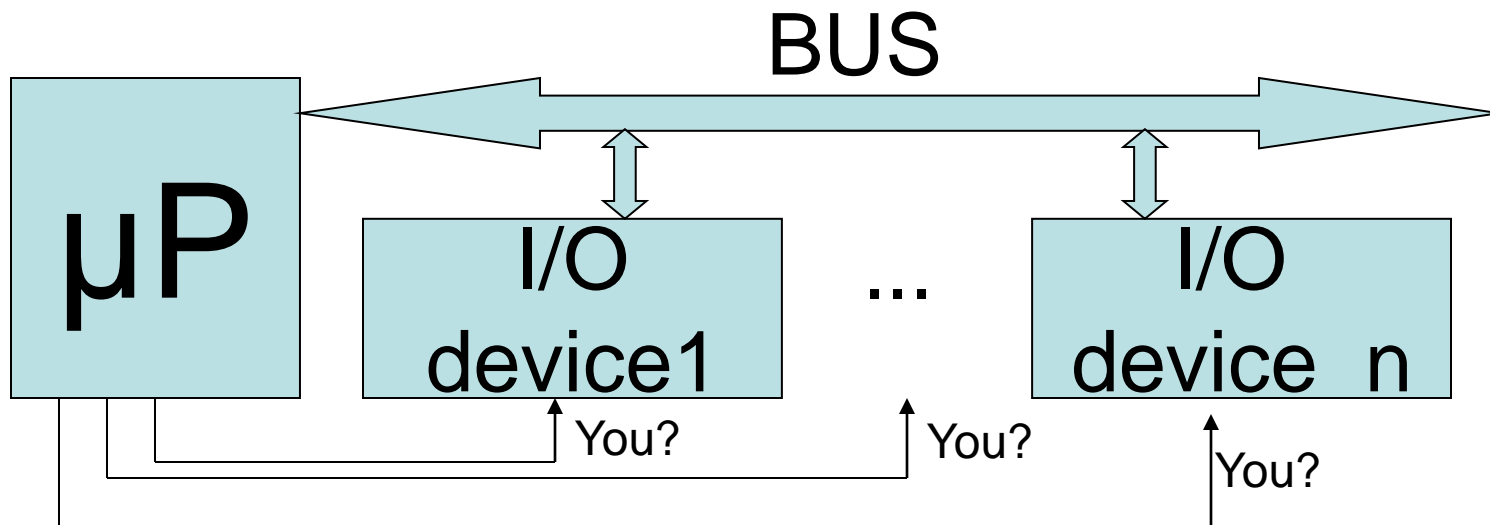
DEDICATED



Single I/O device

Second type of communication

POLLED I/O or PROGRAMMED I/O

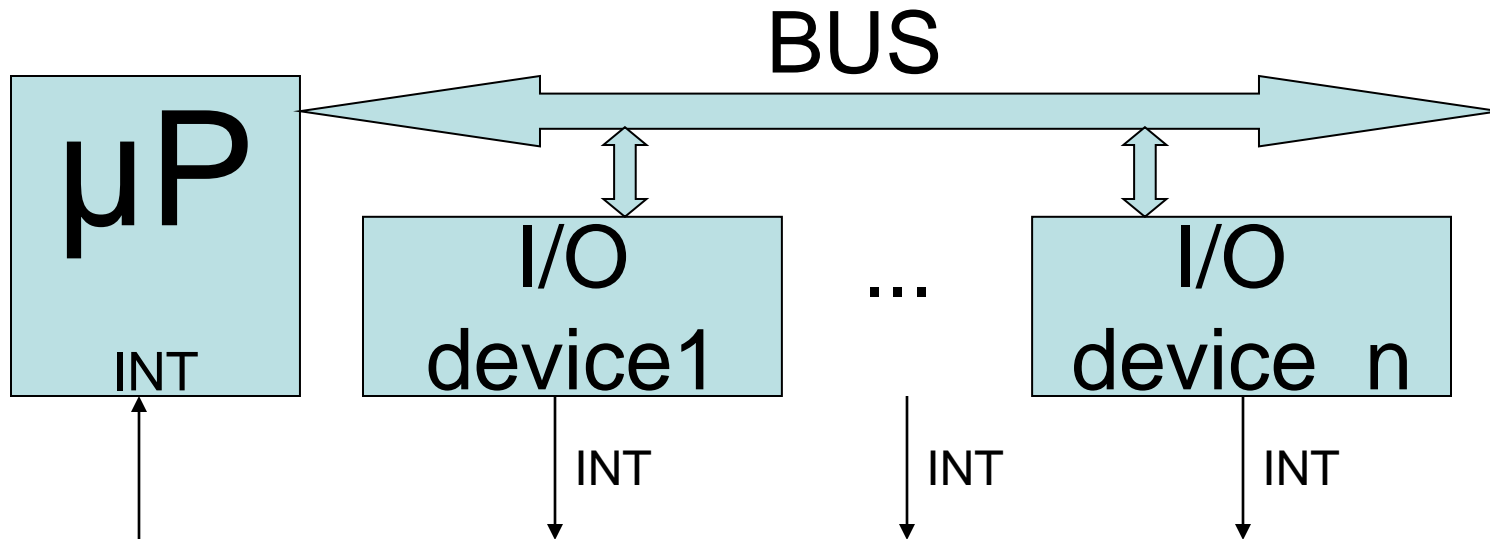


Disadvantage

- not fast enough
- waste too much microprocessor time

3RD Type of communication

INTERRUPTED I/O



Interrupts are particularly useful when I/O devices are slow

- Most microprocessors allow normal program execution to be interrupted by **some external signal** or by a **special instruction in the program**.
- In response to an interrupt, the microprocessor stops executing its current program and calls a procedure which “services” the interrupt.
- A special instruction --- IRET --- at the end of interrupt-service procedure returns execution to the interrupted main program.

8086 interrupts can be classified into two types:

1) **Predefined interrupt**

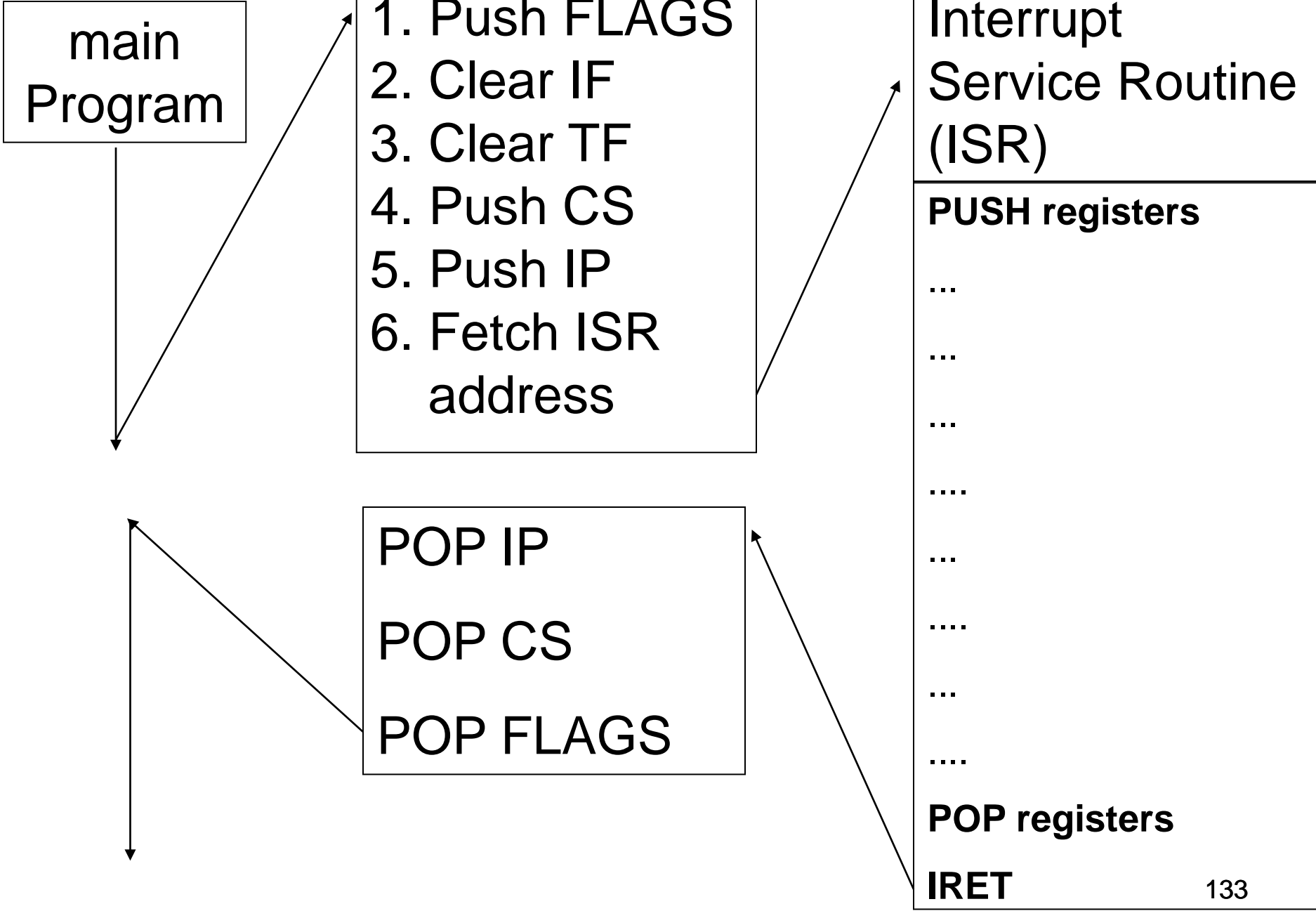
some error condition produced by execution of an instruction, e.g., trying to divide some number by zero.

2) **User defined interrupt**

- i) ***hardware interrupt***
 - An ***external signal*** applied to INTR pin
- ii) ***software interrupt***
 - execution of interrupt instruction ***INT***

- At the **end of each instruction cycle**, 8086 checks to see if any interrupts have been requested.
- If yes, then 8086 responds to the interrupt by stepping through the following series of major actions:

1. It decremented SP by 2 and pushes ***Flag register*** on the stack.
2. It disables 8086 **INTR** input by clearing **IF flag** in Flag register
3. It resets the **TF (trap) flag** in Flag register
4. It decremented SP again by 2 and pushes current **CS** contents on the stack.
5. It decremented SP again by 2 and pushes current **IP** contents on the stack.
6. It does an indirect far jump to the start of the procedure written to respond to the interrupt.



- How does 8086 get to Interrupt Service Routine?
 - Simple. It loads its CS and IP registers with the address of ISR.
 - So, the next instruction to be executed is the first instruction of ISR

- How does 8086 get the address of Interrupt Service Routine (ISR)?
 - It goes to **specified memory location** to fetch **four consecutive bytes**
 - higher two bytes to be used as CS
 - lower two bytes to be used as IP

- How does 8086 get the address of that **specified memory location**?
 - In an 8086 system, the first 1Kbytes of memory, from 00000 to 003FF, is set aside as a **Table** for storing the starting addresses of interrupt service routines.
 - Since 4 bytes are required to store **CS** and **IP** values for each ISR, the **Table** can hold the starting addresses for up to 256 ISRs.

- The starting address of an ISR is often called
 - the ***interrupt vector*** or
 - the ***interrupt pointer***.
- So the Table is referred to as
 - interrupt-vector table*** or
 - interrupt-pointer table***.

- Next slide shows how the 256 interrupt vectors are arranged in the table in memory
- Note that
 - the **IP** value is put in as the **low word** of the vector
 - **CS** as **high word** of the vector
- Each double word interrupt vector is identified by a number from 0 to 255
- INTEL calls this number the ***TYPE*** of the interrupt

AVAILABLE FOR USER		}	3FFH	TYPE 255
(224)			080H	...
RESERVED (27)		}		TYPE 31
			014H	...
Predefined/ Dedicated/Internal Interrupts Pointers (5)		}		TYPE 4
			010H	INTO OVERFLOW
			00CH	TYPE 3 INT
			008H	TYPE 2 NON-MASKABLE
			004H	TYPE 1 SINGLE STEP
				TYPE 0
CS Base Address		}	000H	DIVIDE ERROR
IP Offset				

139

- How does 8086 get the address of a particular ISR?
 - In an 8086 system, each “interrupter” has an id#
 - 8086 treat this id# as interruption type#
 - after receiving INTR signal, 8086 sends an INTA signal
 - after receiving INTA signal, interrupter releases it's id#, i.e., type# of the interruption.

contd...

- 8086 multiplies this id# or type# by 4 to produced the desired address in the ***vector table***
- 8086 reads 4 bytes of memory starting from this address to get the starting address of ISR
- lower 2 byte is loaded in to IP
- higher 2 bytes to CS

What happens if two or more interrupts occur at the same time?

- higher priority interrupts will be served first

Priorities of 8086 interrupts

Interrupt Type	Priority
DIVIDE ERROR, INT n, INT0	HIGHEST
NMI	
INTR	
SINGLE STEP	
	LOWEST

Delay Loop Calculation

Hall ch 4

2nd Edition

page 91

- Each instruction takes a certain number of clock cycles to execute
- MOV register to register [mov ax,bx] for example requires 2 clock cycles
- If you are running a μP , for example, with a 10 MHz clock,
 - then each clock cycles takes $1/(10 \text{ MHz})$ or $0.1\mu\text{s}$
- An instruction which takes 2 clock cycles, then will take $2 \times 0.1\mu\text{s}$ or $0.2 \mu\text{s}$ to execute

- The basic principle is to execute
 - an instruction or
 - series of instructions
 - over and over
 - until the desired time has elapsed.
- Following is a program that might be used to do this:

```
        MOV CX, n  
abrar:  NOP  
        LOOP abrar
```

Calculation

1. Calculate the number of clock cycles needed to produce the desired delay
 - Example: Say μP is of 10 MHz and you want a delay of 1ms.
 - Calculation: a) each clock cycle = $1/(10\text{MHz})$
 $= 0.1\ \mu\text{s}$
b) to have, say, 1 ms delay you have to wait $1\text{ms}/0.1\ \mu\text{s} = 10,000$ clock cycles

Calculation

2. Calculate the number of clock cycles required by the *overhead*, C_o

- MOV CX, n is the overhead
- It requires 4 clock cycles

3. Determine how many clock cycles are required for the loop, C_L

- abar: NOP

LOOP abar

is the body of the loop

- 3 clock cycles is required for NOP
- 17 clock cycles is required if it jumps back to abar
- 5 to exit the loop

Calculation

4. Total number of cycles required for the execution of the example code C_T

$$C_T = C_O + \{C_T \times n - 12\}$$

$$10,000 = 4 + \{20n - 12\}$$

$$n = (10000 - 4 + 12) / 20 = 500.8$$

5. Let n be 501

- Precise timing is a little bit difficult

80186

- 80186 contains 8086 processor and 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

INTRODUCTION

Salient features of 80286

- High performance microprocessor with memory management and protection
 - 80286 is the first member of the family of advanced microprocessors with built-in/on-chip memory management and protection abilities primarily designed for multi-user/multitasking systems
- Available in 8 MHz, 10 MHz & 12.5 MHz clock frequencies

Salient features of 80286

bus and memory sizes

cont...

- The 80286 CPU, with its 24-bit address bus is able to address 16MB of physical memory.
- 1GB of virtual memory for each task

Microprocessor	Data bus width	Address bus width	Memory size
8086	16	20	1M
80186	16	20	1M
80286	16	24	16M

Salient features of 80286 Operating Modes

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 8086⁵

Salient features of 80286

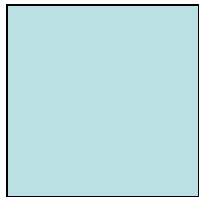
cont...

- 286 includes special instructions to support operating system.
 - for example, one instruction can
 - i) ends the current task
 - ii) save its states
 - iii) switch to a new task
 - iv) load its states and
 - v) begin executing the new task
- housed in 68-pin package

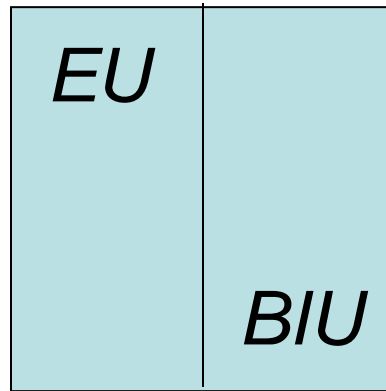


80286

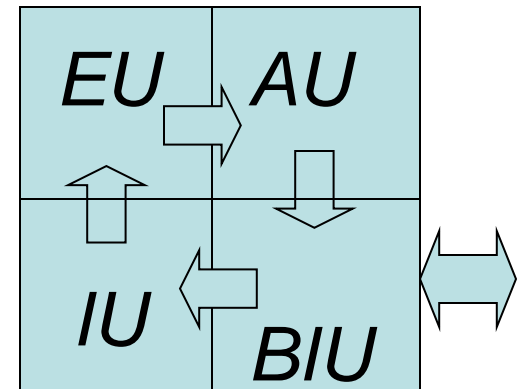
INTERNAL ARCHITECTURE



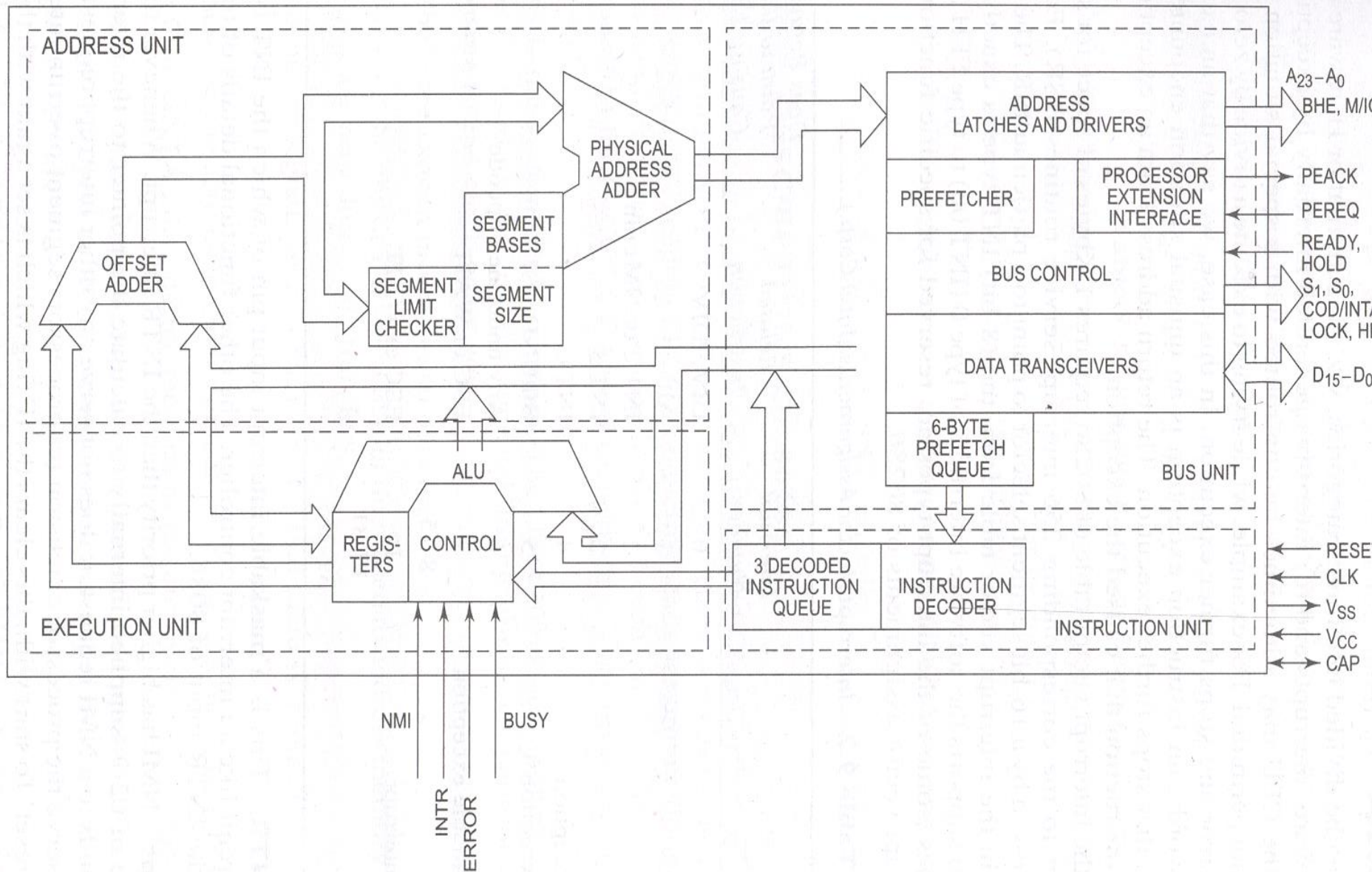
8085



8086



80286



Functional Parts

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

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**.

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

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			

Special
Register
Functions

MULTIPLY/DIVIDE
I/O INSTRUCTION

LOOP/SHIFT/REPEAT COUNT

BASE REGISTERS

INDEX REGISTERS

STACK POINTER

GENERAL
REGISTERS

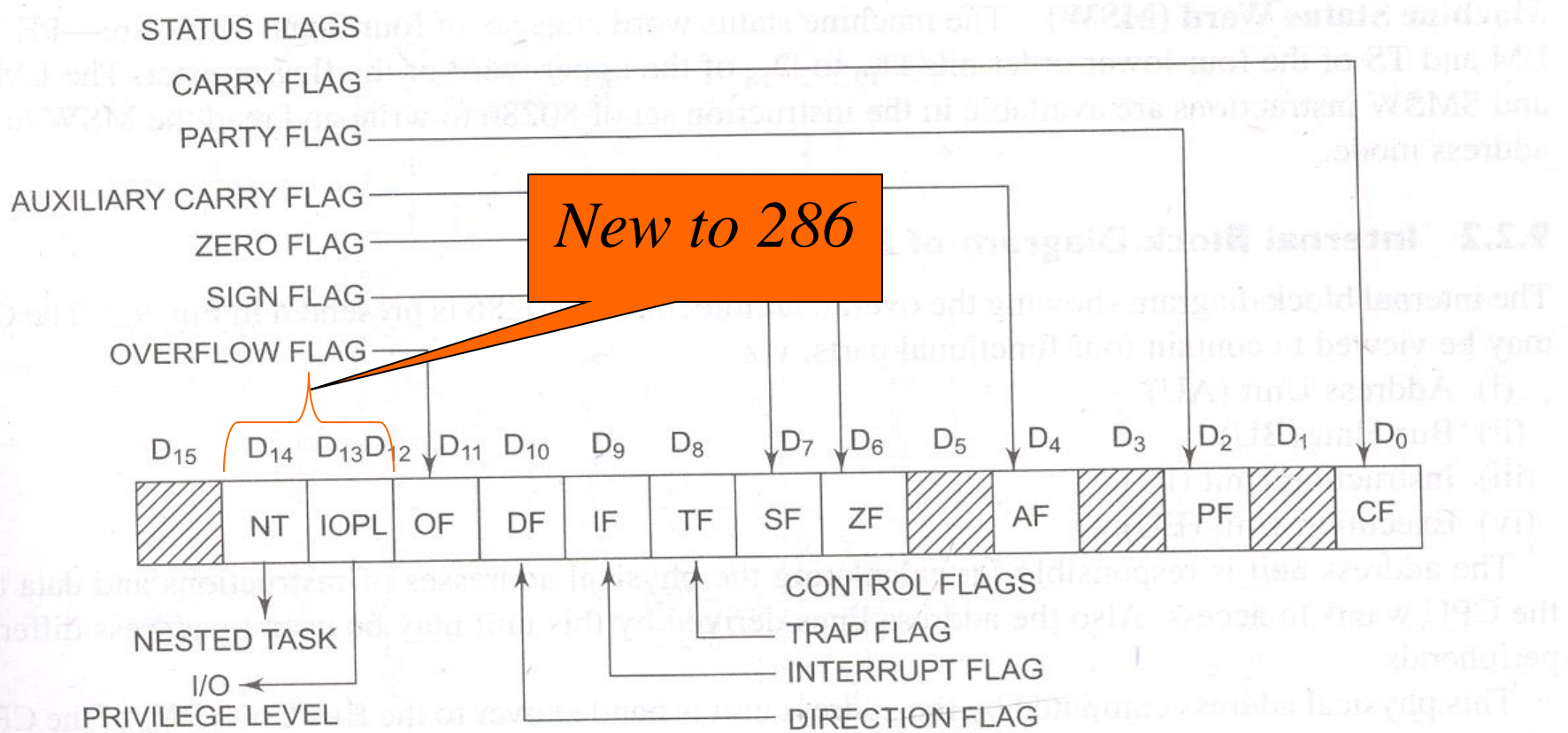
15	0		
CS		CODE SEGMENT SELECTION	
DS		DATA SEGMENT SELECTION	
SS		STACK SEGMENT SELECTION	
ES		EXTRA SEGMENT SELECTION	

SEGMENT REGISTERS

15	0		
F		STATUS WORD	
IP		INSTRUCTION POINTER	

STATUS AND CONTROL
REGISTERS

Flag Register



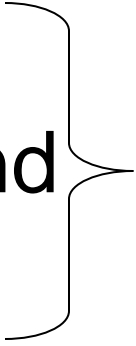
➤ IOPL – Input Output Privilege Level flags (bit D12 and D13)

- 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 IOPL is lower than the current privilege level, 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.

➤ NT – Nested task flag (bit D14)

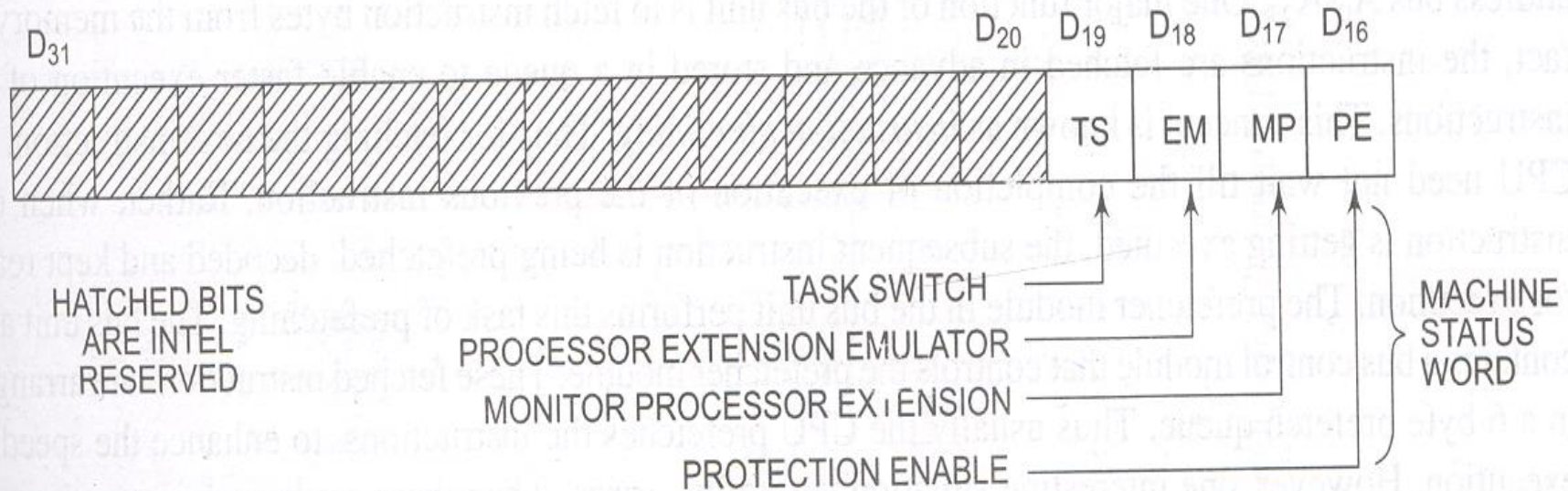
- When set, it indicates that one system task has invoked another through a CALL instruction as opposed to a JMP.
- For multitasking this can be manipulated to our advantage

Machine Status Word Register

- Consist of four flags
 - PE,
 - MP,
 - EM and
 - TS

are for the most part used to indicate whether a processor extension (co-processor) is present in the system or not
- **LMSW & SMSW** instruction are available in the instruction set of 80286 to write and read the MSW in real address mode.

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 indicates the next instruction using extension will generate exception 7, permitting the CPU to test whether the current processor extension is for current task.

80286 signals

- 68 pins are there, instead of 40 in 8086
- data and address bus are de-multiplexed
- mostly like 8086 with some differences

80286 signals...

- some new signals are provided for special jobs like connecting co-processor with 286
 - 4 pins are provided to interface 286 with a co-processor
 - they are
 - PEREQ [Processor Extension **REQ**uest]
 - PEACK [Processor Extension **ACK**nowledge]
 - BUSY
 - ERROR

80286 signals...

- $\overline{\text{PEREQ}}$ [Processor Extension REQuest]
 - Input signal to μP
 - Asserted by co-processor to tell μP to perform data transfer to or from memory for it

80286 signals...

- PEACK [Processor Extension ACKnowledge]
 - Output signal
 - Used to let the co-processor know that the data transfer has started
 - Data transfer are done through μP so that the co-processor can use the protection and virtual memory capability

80286 signals...

- $\overline{\text{BUSY}}$
 - Input signal
 - Same as $\overline{\text{TEST}}$ of 8086
 - Enters in a WAIT loop till μP finds a *high* from co-processor

80286 signals...

- ERROR
 - Input signal
 - If a co-processor finds some error during processing, it will let μP know through this line

80286

Memory Organization

- Same as 8086
- Uses odd and even banks

Addressing Modes

- Same as 8086

INSTRUCTION SET

- Same as 8086 with some additional instructions

Additional Instructions of Intel 80286

Sl no	Instruction	Purpose
1.	CLTS	Clear the task – switched bit
2.	LDGT	Load global descriptor table register
3.	SGDT	Store global descriptor table register
4.	LIDT	Load interrupt descriptor table register
5.	SIDT	Store interrupt descriptor table register
6.	LLDT	Load local descriptor table register
7.	SLDT	Store local descriptor table register
8.	LMSW	Load machine status register
9.	SMSW	Store machine status register

Sl no	Instruction	Purpose
10.	LAR	Load access rights
11.	LSL	Load segment limit
12.	SAR	Store access right
13.	ARPL	Adjust requested privilege level
14.	VERR	Verify a read access
15.	VERW	Verify a write access

➤ CLTS

- The **clear task – switched flag** instruction clears the TS (Task - switched) flag bit to a logic 0.

➤ LAR

- The **load access rights** Instruction reads the segment descriptor and place a copy of the access rights byte into a 16 bit register.

➤ LSL

- The **load segment limit** instruction Loads a user – specified register with the segment limit.

➤ VERR

- The **verify for read access** instruction verifies that a segment can be read.

➤ VERW

- The **verify for write access** instruction is used to verify that a segment can be written.

➤ ARPL

- The **Adjust request privilege level** instruction is used to test a selector so that the privilege level of the requested selector is not violated.

INTERRUPT

- Same as 8086 but defines more dedicated/internal interrupts
 - Uses from type 5 to type 13 and type 16

80286

Memory Management

80286 Memory Addressing

- Memory management is supported by a hardware unit called **Memory management unit**.
- Intel's 80286 is the first CPU to incorporate the *Integrated memory management unit*.

80286 Memory Addressing

- Function of memory management unit :
 1. To ensure smooth execution of the program by
 - **SWAPPING IN** data from secondary memory to physical memory
 - **SWAPPING OUT** data from physical memory to secondary memory
 2. Important aspect of memory management is **Data Protection** or **unauthorized access prevention** done with the help of segmented memory

REAL MODE MEMORY ADDRESSING

- 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 4 or Core2 microprocessor.
 - the first 1M byte of memory is called the **real memory, conventional memory,** or **DOS memory** system

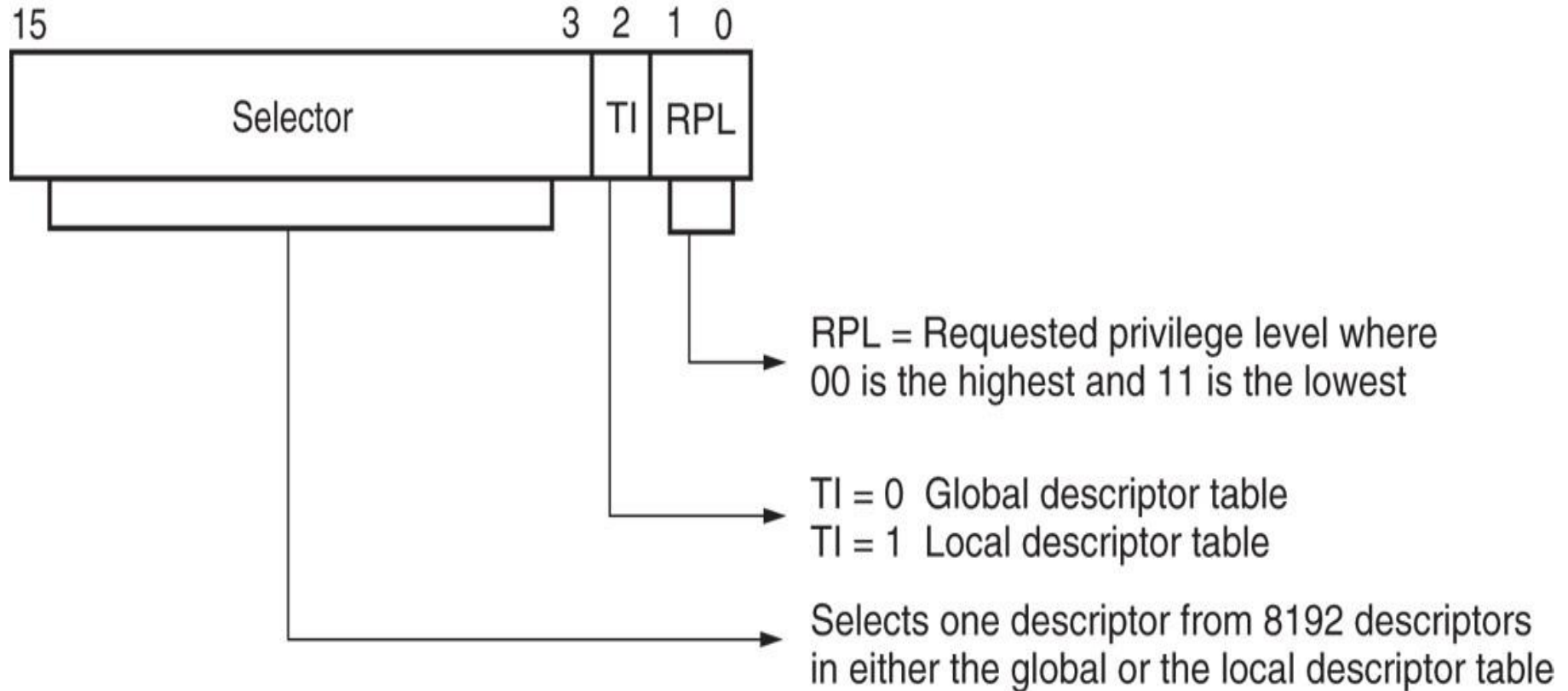
Segments and Offsets

- 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 MEMORY ADDRESSING

- Allows access to data and programs located within & above the first 1M byte of memory.
- **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.

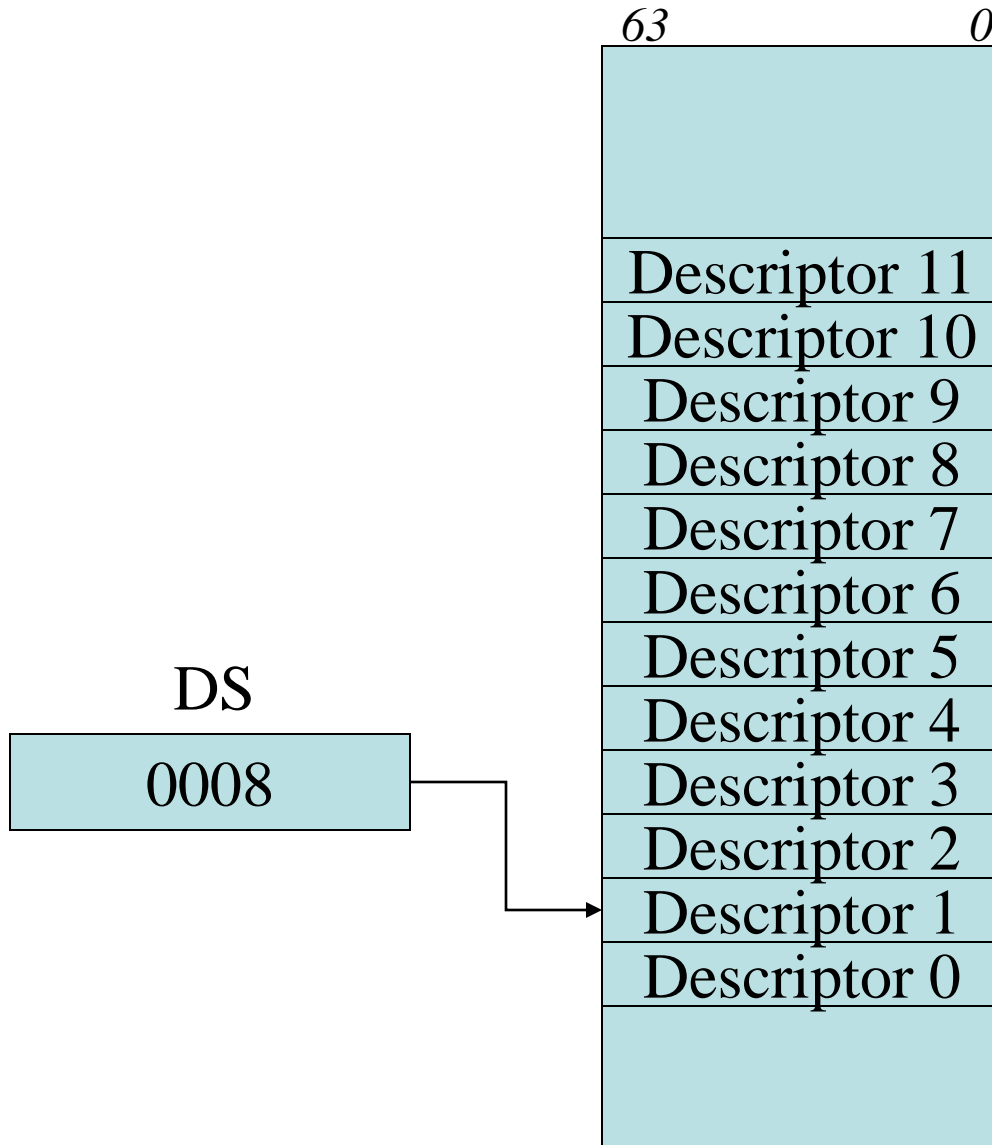
Contents of segment register



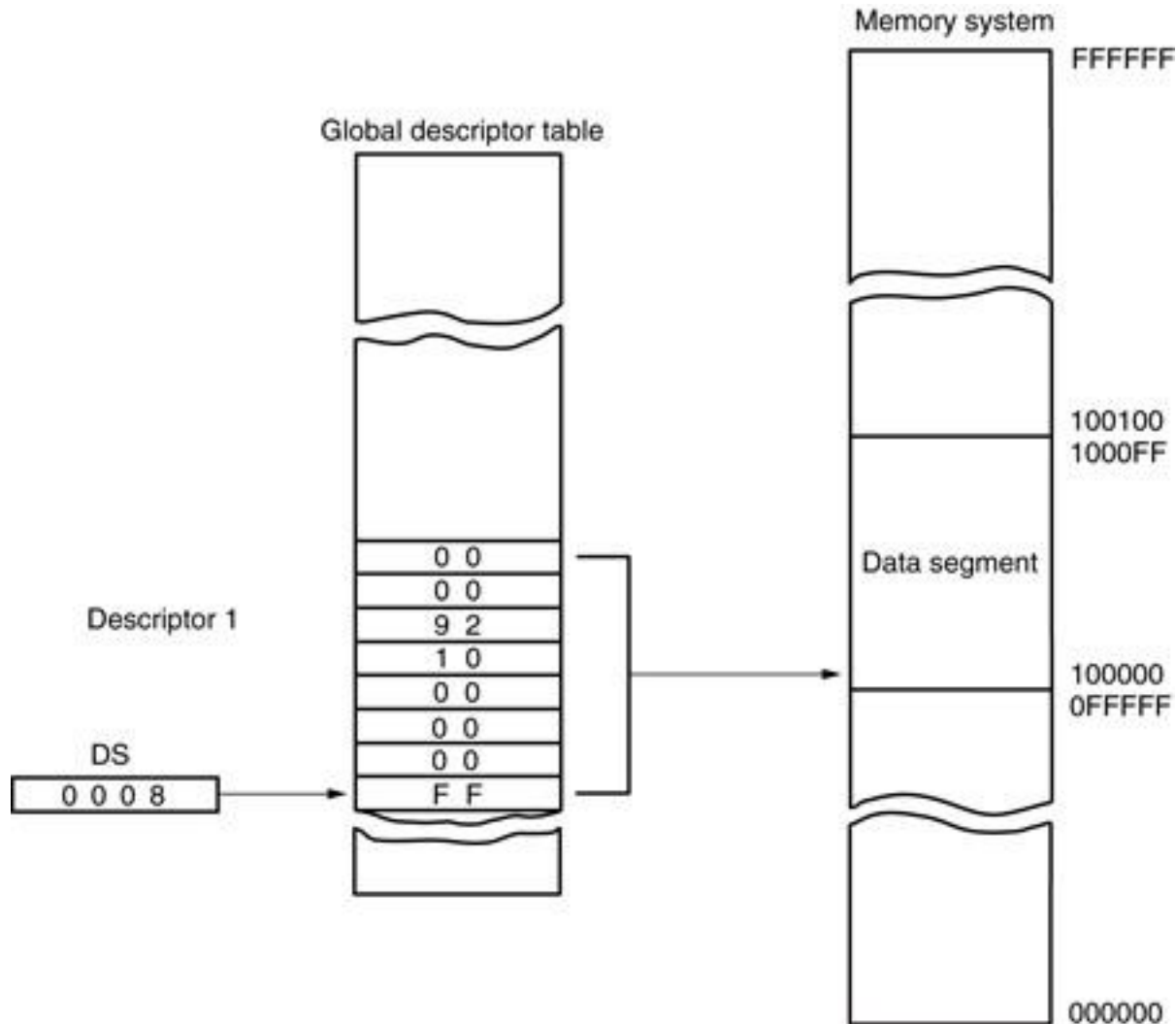
Since segment descriptors are each 8 bytes, the last three bits of the selector is zero, in which one of them is used for LDT/GDT access.

- 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

- Following slide shows how the segment register, containing a selector, chooses a descriptor from the global descriptor table.
- The entry in the global descriptor table selects a segment in the memory system.
- Descriptor zero is called the null descriptor, must contain all zeros, and may not be used for accessing memory.



Using the DS register to select a description from the global descriptor table. In this example, the DS register accesses memory locations 00100000H–001000FFH as a data segment.



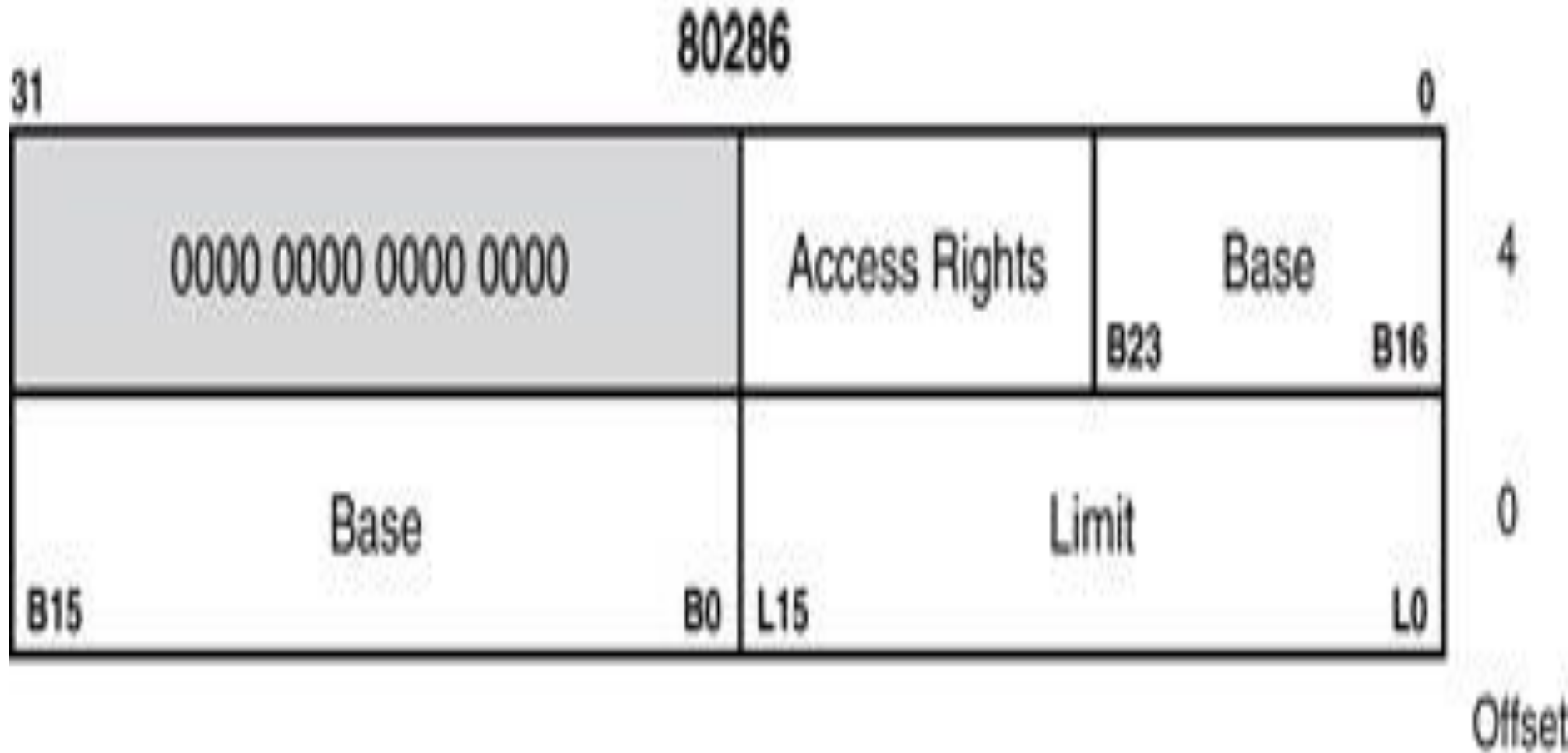
Selectors and Descriptors

- 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.

- Following slide shows the format of a descriptor for the 80286
 - each descriptor is 8 bytes in length
 - global and local descriptor tables are a maximum of 64K bytes in length

The 80286 descriptors



The 80286 descriptors

Reserve

2 Bytes

1 Byte

Base Address

24 bits

Length

16 bits

0000 0000 0000 0000

Access Rights

MSB

B23

LSB

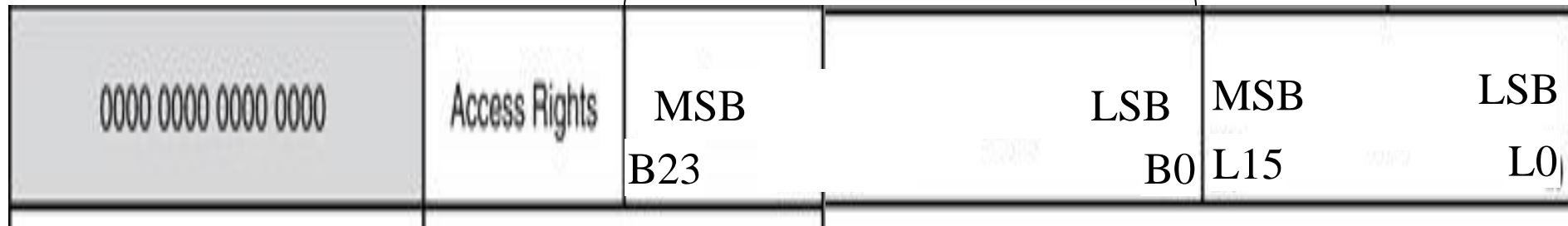
B0

MSB

L15

LSB

L0



The 80286 descriptors

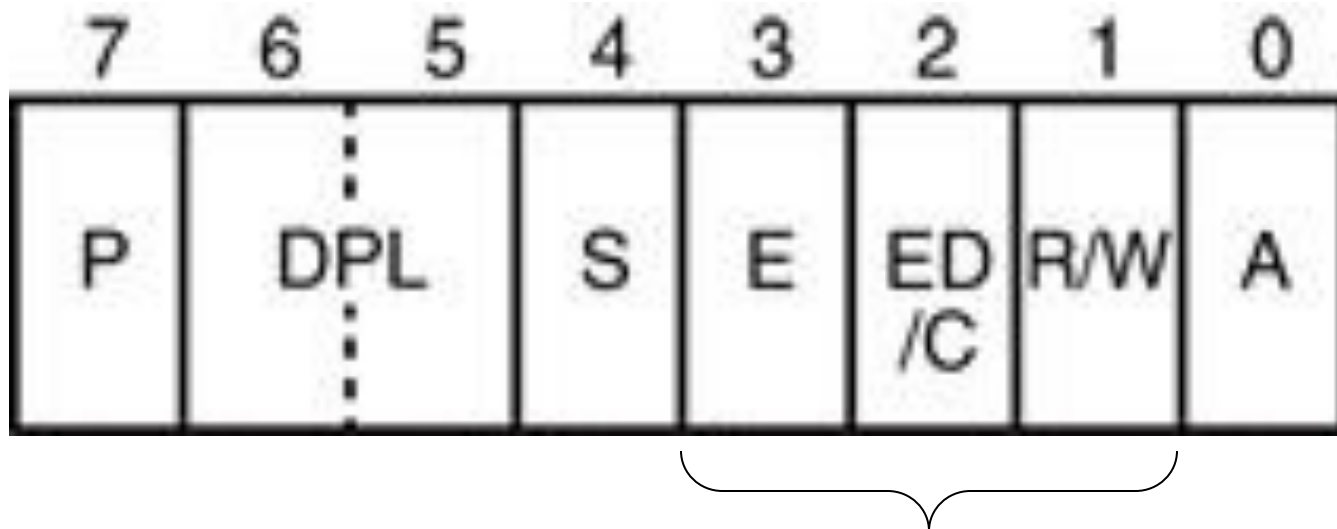
Reserve		ARB	Base Address	Length
00	00	8 bit	$B_{23} \dots B_0$	$L_{19} \dots L_0$
MSB				LSB

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

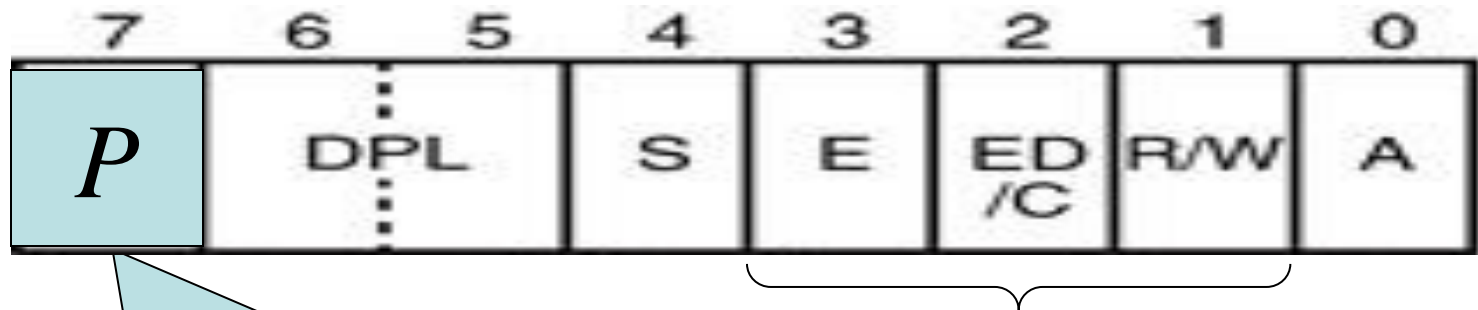
- The **access rights byte** controls access to the protected mode segment.
 - describes segment function in the system and allows complete control over the segment
 - if the segment is a data segment, the direction of growth is specified
- If the segment grows beyond its limit, the operating system is interrupted, indicating a general protection fault.
- You can specify whether a data segment can be written or is write-protected.

Access Right Byte

Note: Some of the letters used here to describe the bits vary from Intel documentation



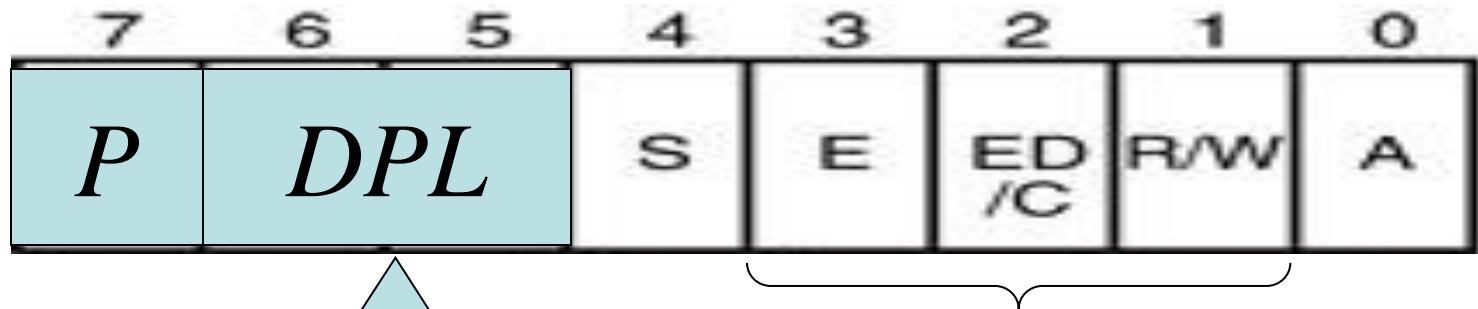
Access Right Byte --- Bit 7 Present?



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

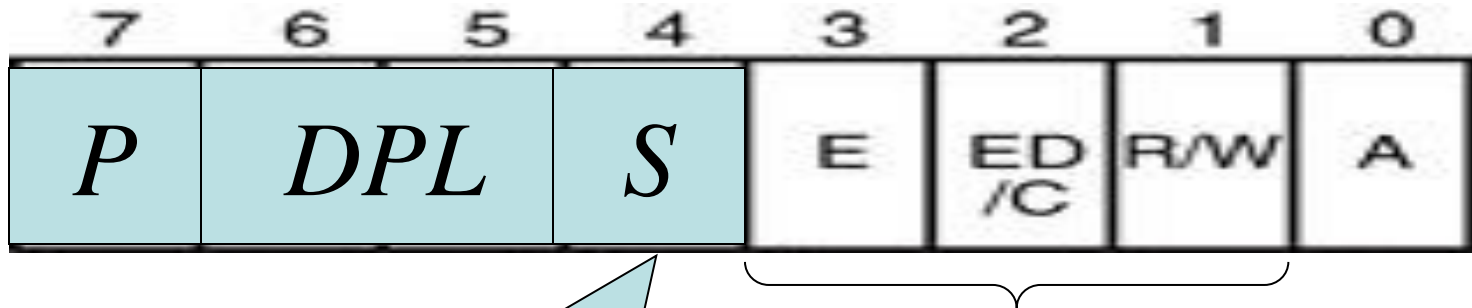
P=1 *Valid Descriptor info, Segment is mapped into physical memory*

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



*Sets the Descriptor privilege level
necessary for protection*

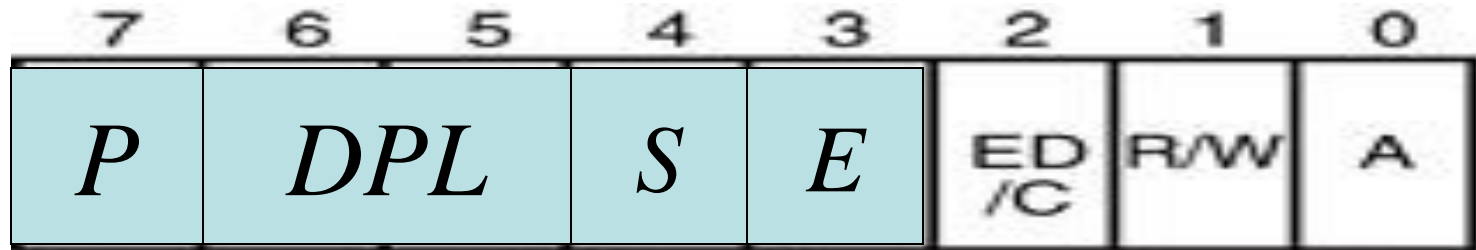
Access Right Byte --- Bit 4 Descriptor Type



S=0 *System descriptor*

S=1 *Application Descriptor*

Access Right Byte --- Bit 3 Executable?



E=0 *No Data/Stack Segment*

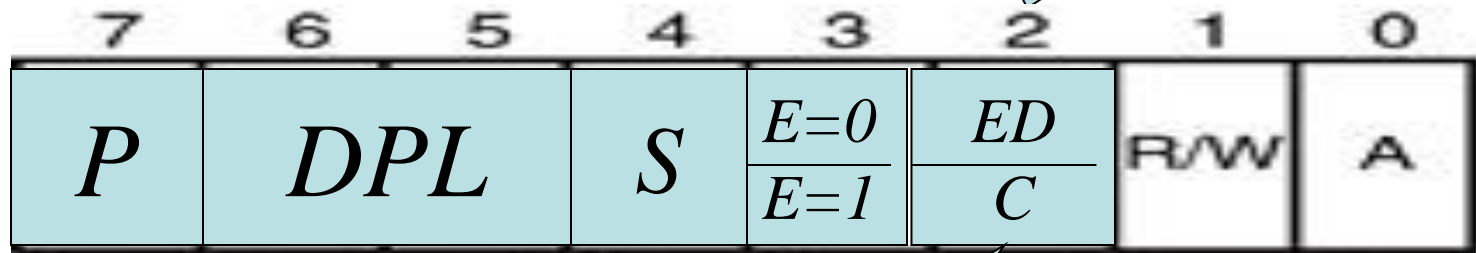
E=1 *YES Code Segment*

ED=0 Segment expands upward (Data segment)

ED=1 Segment expands downward (Stack Segment)

Access Right Byte

Bit 2



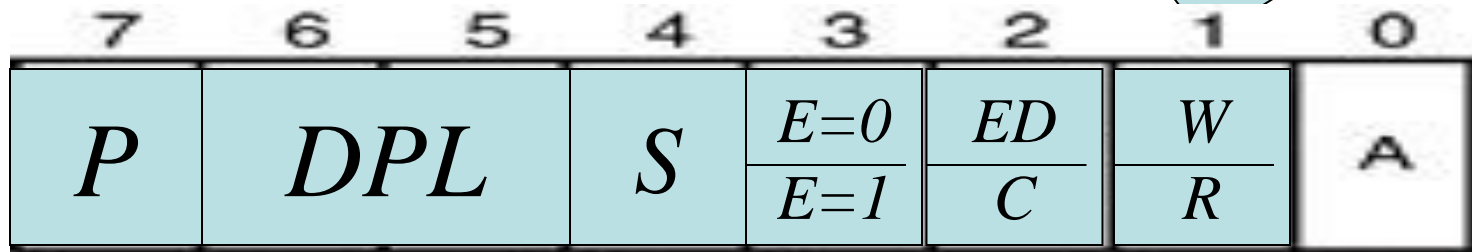
C=0 Ignore DPL

C=1 Abide by DPL

Access Right Byte --- Bit 1

$W=0$ Data segment not writable

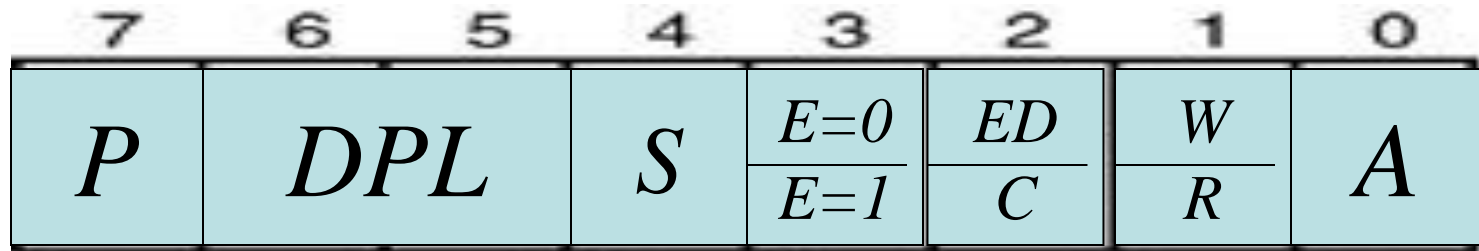
$W=1$ Data segment writable



$R=0$ 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*

Did You Note!!

- There is an 100 % degradation in Memory access time – because every memory access is two accesses now, one for getting the base address and another for actually accessing the data.
- solution ?????
- Along with the segment registers, keep a shadow registers which stores additional necessary information.

Program-Invisible Registers

- Global and local descriptor tables are found in the memory system.
- To access & specify the table addresses, 80286 contain program-invisible registers.
 - not directly addressed by software
- Each segment register contains a program-invisible portion used in the protected mode.
 - often called cache memory because cache is any memory that stores information

Segment Registers

CS

DS

ES

SS

Descriptor Cache

Base Address	Limit	Access

TR

LDTR

Base Address	Limit	Access

Descriptor Table addresses

GDTR	Base Address	Limit
IDTR		

- When a new segment number is placed in a segment register, the microprocessor accesses a descriptor table and loads the descriptor into the program-invisible portion of the segment register.
 - held there and used to access the memory segment until the segment number is changed
- This allows the microprocessor to repeatedly access a memory segment without referring to the descriptor table.
 - hence the term *cache*

- The GDTR (**global descriptor table register**) and IDTR (**interrupt descriptor table register**) contain the base address of the descriptor table and its limit.
 - when protected mode operation desired, address of the global descriptor table and its limit are loaded into the GDTR
- The location of the local descriptor table is selected from the global descriptor table.
 - one of the global descriptors is set up to address the local descriptor table

- To access the local descriptor table, the LDTR (**local descriptor table register**) is loaded with a selector.
 - selector accesses global descriptor table, & loads local descriptor table address, limit, & access rights into the cache portion of the LDTR
- The TR (task register) holds a selector, which accesses a descriptor that defines a task.
 - a task is most often a procedure or application
- Allows multitasking systems to switch tasks to another in a simple and orderly fashion.

GDTR – Global descriptor table register.

The base address of the descriptor table and its limit. The limit of each descriptor table is 16 bits because the maximum length of the table is 64K bytes.

When protected mode operation is desired the values are loaded in the GDTR .

IDTR – Interrupt descriptor table register.

The base address of the descriptor table and its limit. The limit of each descriptor table is 16 bits because the maximum length of the table is 64K bytes.

When protected mode operation is desired the values of the IDTR must also be initialized. We will deal with this later.

LDTR – Local descriptor table register.

The LDTR does not hold the direct address of the local descriptor table. Instead it holds a **selector** just as the segment registers.

The selector selects a descriptor in the global address table which points to the location of the local descriptor table. The base address of the LDT is then loaded in the LDTR.

TR – Task Register

Holds a selector which accesses a descriptor that defines a task. Task can be procedure or application program. The descriptor for the task is stored in the GDT so access can be controlled. Task register allows fast context switching helping in tasks that are often used or in multi-programming.

80286 Protection

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.

Updating Segment registers

- Segment registers (DS, ES, and SS) are updated by normal MOV instructions.
 - MOV AX, 0x10 ; MOV DS, AX
- The above command is successful if and only if the descriptor stored at the offset 0x10 in the descriptor table has a privilege level numerically greater than or equal to the CPL.
- A process with CPL = 3 cannot load the segment descriptor of CPL ≤ 2 , and hence cannot access the segments.

Updating segment registers

- The code segment register is updated by normal jump/call operations.
 - `jmp 0x20:0x1000`
 - This updates the CS by 0x20, provided the descriptor stored at offset 0x20 has a privilege level numerically greater than or equal to CPL
- Other modes of updating CS register
 - Numerically higher to lower Privilege Levels using CALL gates – useful for system calls.
 - Any privilege level to any other privilege level using task switch.

80386

References

- Hall
 - Rafiq
- } Hall / Rafiq presents a concise treatment
-
- Brey
- the materials are scattered throughout the book in different chapters

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 switch off the 286

386 was designed to overcome these limitations

- 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 $4\text{G} \times 16\text{K} = 64\text{TB}$ 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**
- Intel introduced the first 32-bit chip, 80386, in October 1985 as an upgrade to the 80286 processor
- Intel stopped producing 386 since September 2007.
- 386 incorporates 275,000 transistor
- 386 was capable of performing more than five million instructions every second ([MIPS](#))
- 386 was available in clock speeds between 12 and 40MHz.

versions of 386

- Two versions were commonly available:
 - 1) 80386DX
 - 2) 80386SX
- The original 386 processor was renamed as 80386DX or 386DX after introducing 386SX
- 80386SX was introduced in 1988 as a low cost alternative to the original 386.

versions of 386...

- 80386SX was developed after the DX for application that didn't require the full 32-bit capabilities.
- It is found that many PCs use the same basic mother board design as the 80286.
- Most application need less than the 16MB of memory, so the SX is popular and less costly version of the 80386 microprocessor.

versions of 386...

- The 80386SX lacked a math coprocessor but still featured the 32-bit architecture and built-in multitasking.
- The chip was available in clock speeds of 16MHz, 20MHz, 25MHz, and 33MHz.

versions of 386...

- The **80386SL**, more commonly known as the **386SL**, was a version of 386DX first manufactured in 1990
 - featuring low power consumption and
 - was used mainly in portable computers.

DX

vs

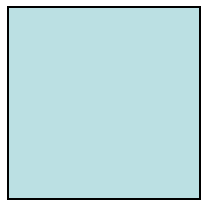
SX

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

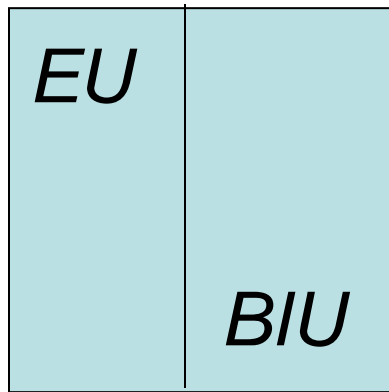
- Both have the same internal architecture.
- Lower cost package and the ease of interfacing to 8-bit and 16-bit memory and peripherals make SX suitable for use in low cost systems.

80386

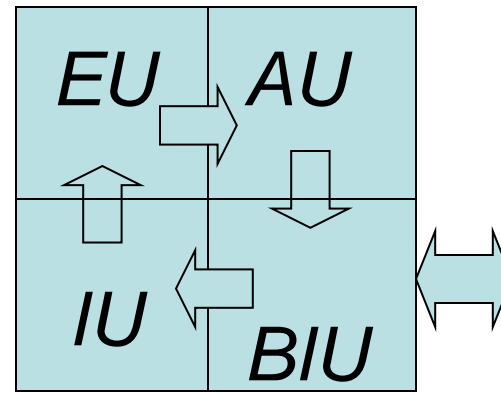
Internal Architecture



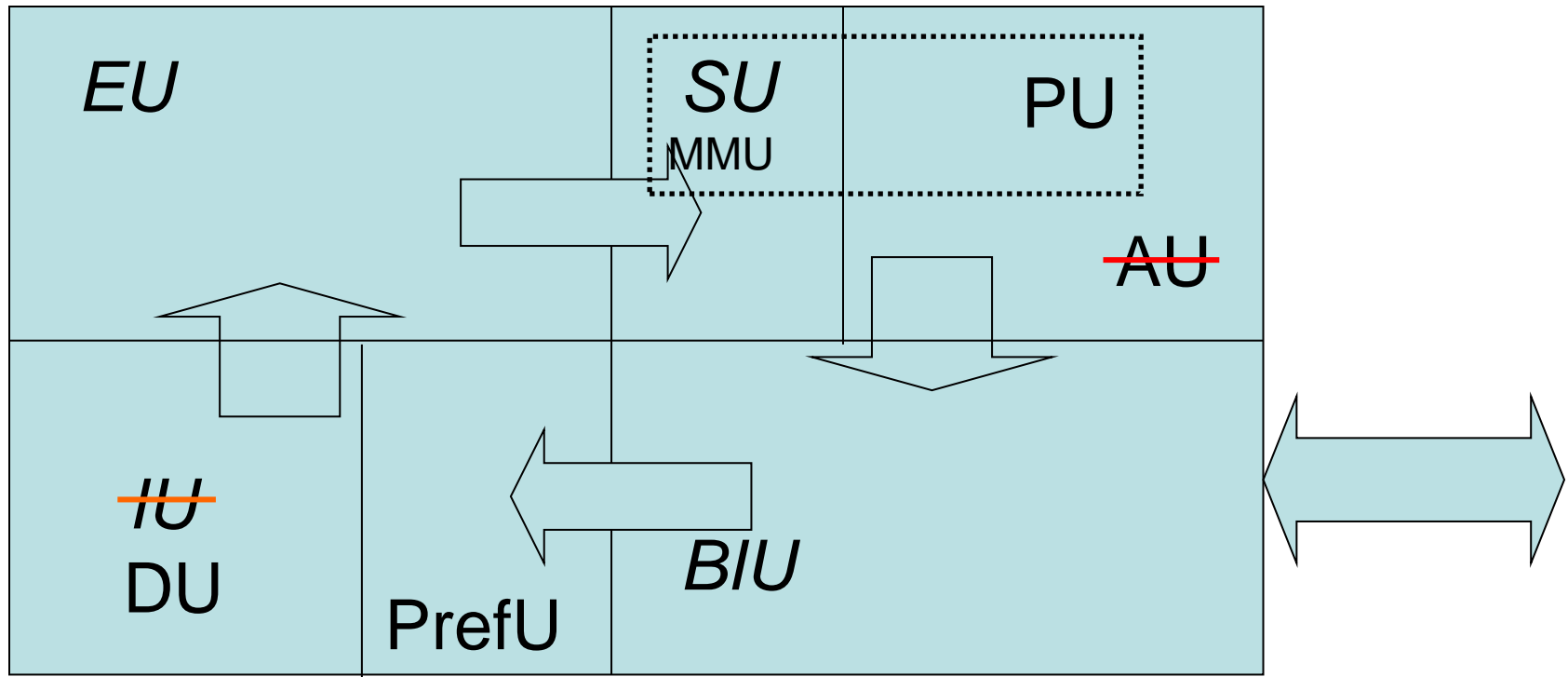
8085



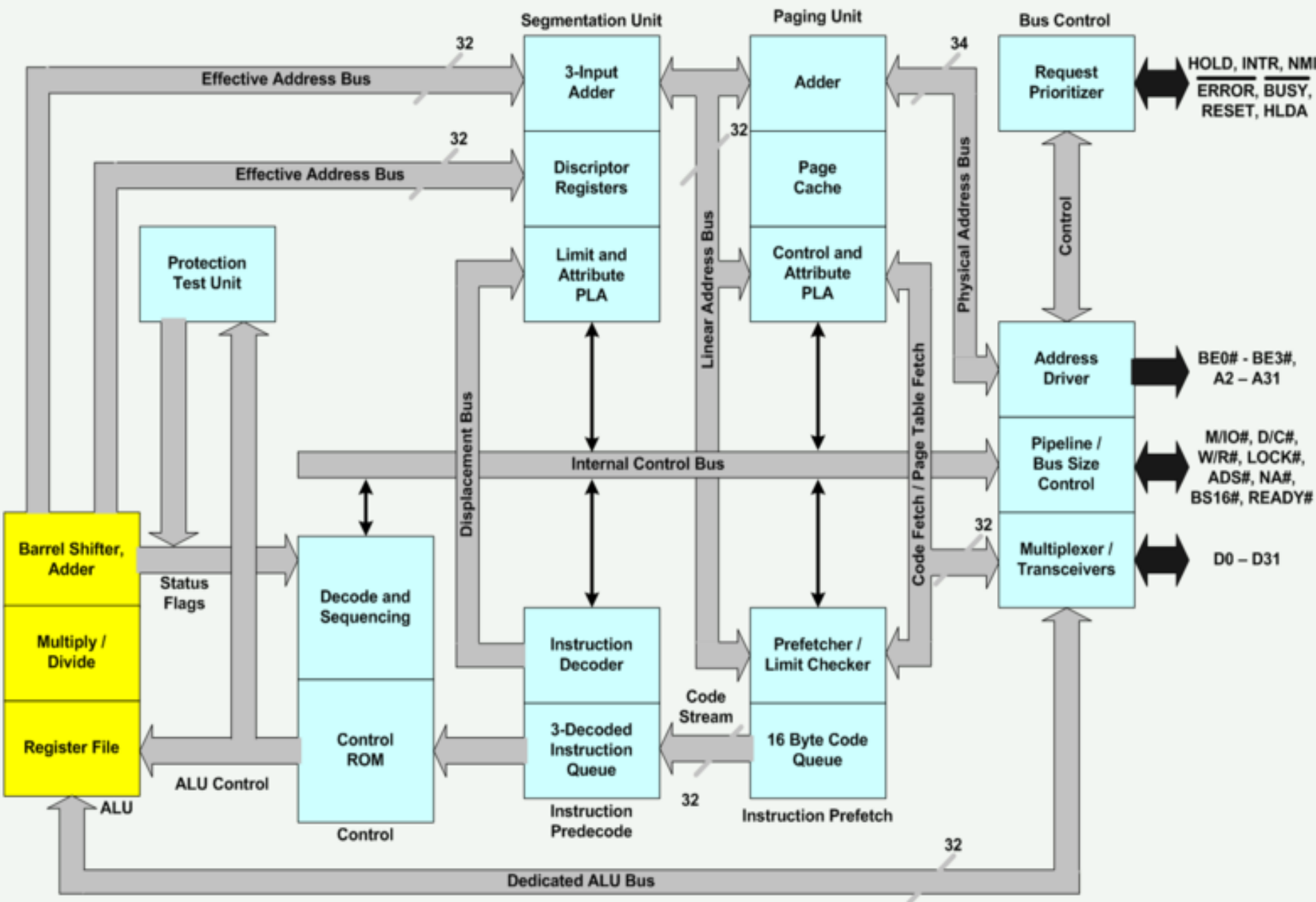
8086



80286



80386



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 unit (IU)
 - **ii) Memory management unit (MMU)** consists of
 - Segmentation unit and
 - Paging unit.
 - **iii) Bus interface unit(BIU)**

Architecture of 80386...

Execution unit

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

- **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**

- **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.**

GENERAL DATA AND ADDRESS

31 16 15 0

	AX	EAX
	BX	EBX
	CX	ECX
	DX	EDX
	SI	ESI
	DI	EDI
	BP	EBP
	SP	ESP

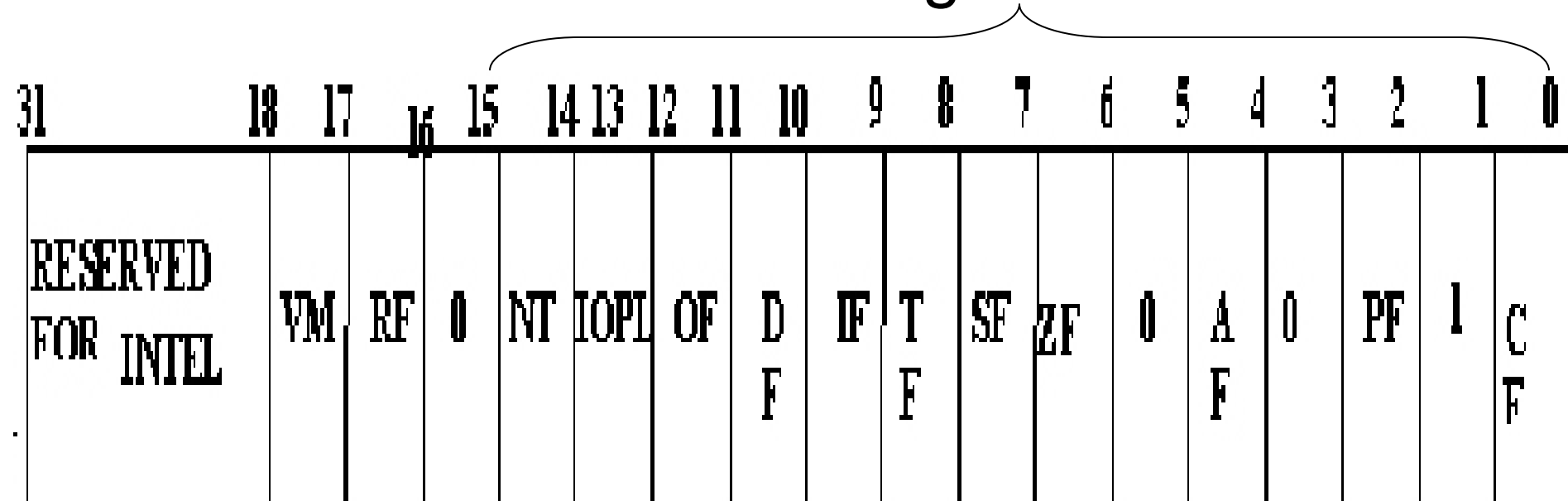
	FLAG	EFLAGS
--	------	--------

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...

Old flags of 286



New flags for 386

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
- This is to be set only when the 80386 is in protected mode.
- This bit can be set using IRET instruction or any task switch operation only in the protected mode.

RF- Resume Flag

- If $RF=1$, 386 ignores debug faults
 - does not take another exception so that an instruction can be restarted after a normal debug exception.
- If $RF=0$, 386 takes another debug exception to service debug faults

RF- Resume Flag...

- This flag is used with the debug register breakpoints.
- It is checked at the starting of every instruction cycle and if it is set, any debug fault is ignored during the instruction cycle.
- The RF is automatically reset after successful execution of every instruction, except for IRET and POPF instructions.

RF- Resume Flag...

- Also, it is not automatically cleared after the successful execution of JMP, CALL and INT instruction causing a task switch.
- These instruction are used to set the RF to the value specified by the memory data available at the stack.

Architecture of 80386...

Instruction unit

- 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...

Instruction unit

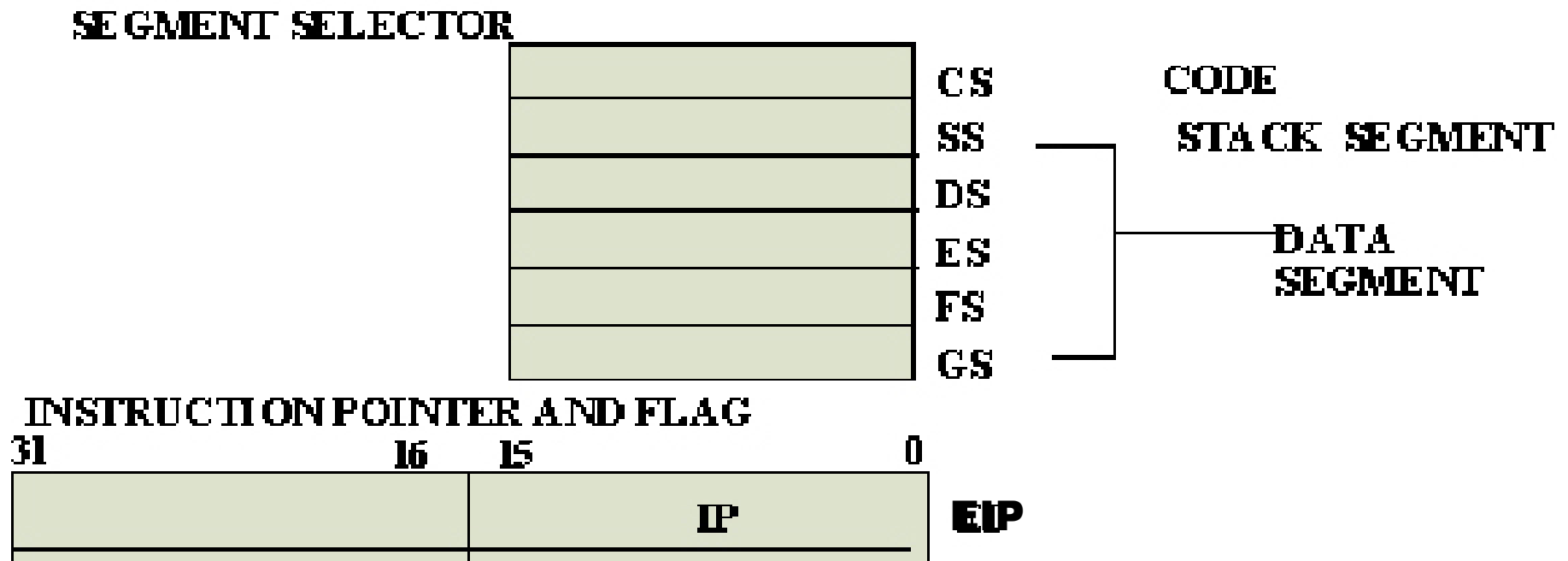
- The barrel shifter increases the speed of all shift and rotate operations.
- The multiply / divide logic implements the bit-shift-rotate algorithms to complete the operations in minimum time.
- Even 32- bit multiplications can be executed within one microsecond by the multiply / divide logic.

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 allows segments of size 4GB at max.
- The Segmentation unit provides a 4 level 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**.
- 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.



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. each segment is further divided into pages.
- The virtual 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 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 prioritizer 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.
- 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

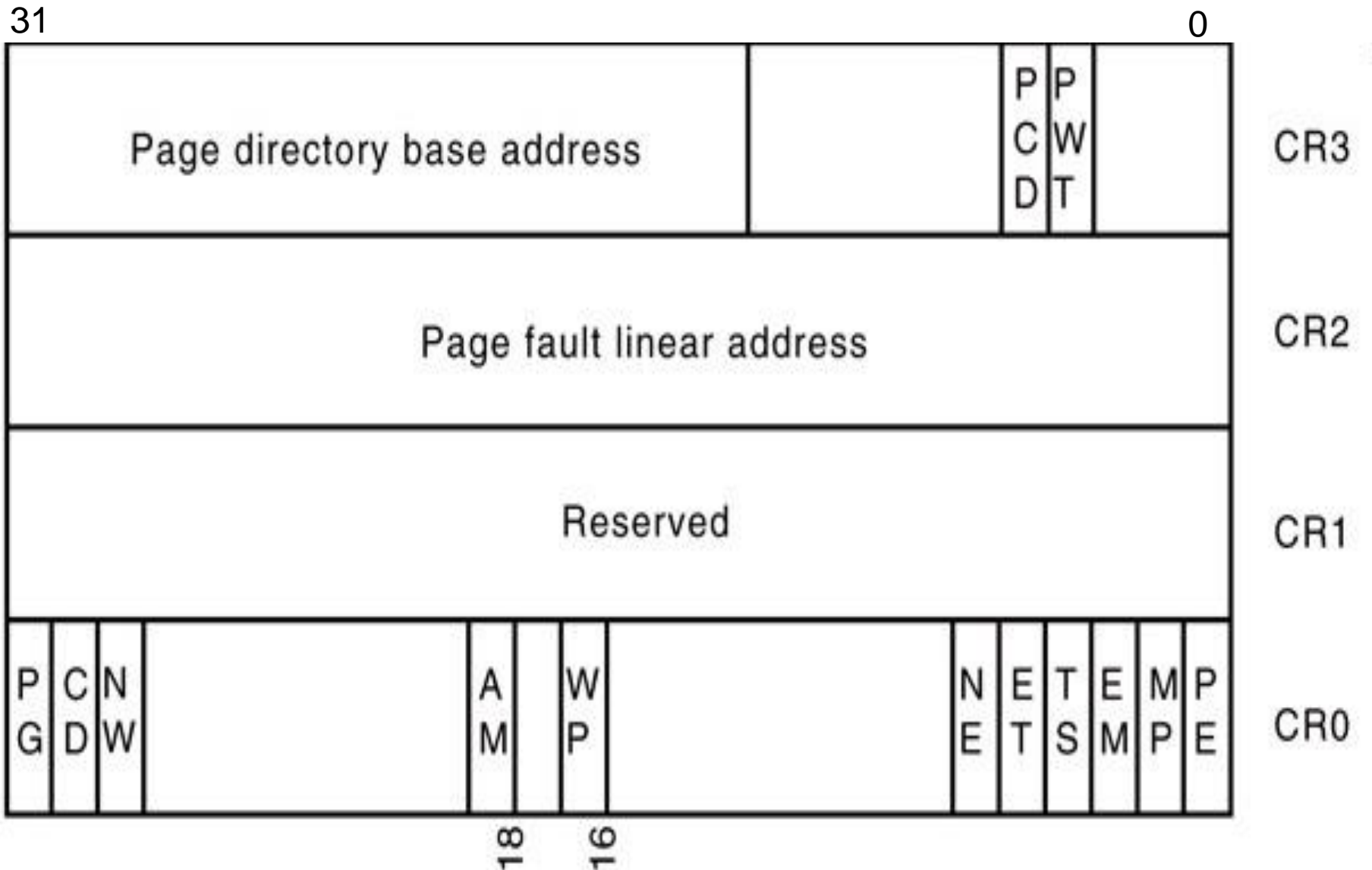
Segment Descriptor Registers

- **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.**
- **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



System Address Registers

- **The 386 supports four types of descriptor table:**
 - **global descriptor table (GDT),**
 - **local descriptor table (LDT),**
 - **interrupt descriptor table (IDT), and**
 - **task state segment descriptor (TSS).**
- **Four special registers are defined to hold the base address of these tables**
 - **global descriptor table Register (GDTR),**
 - **local descriptor table Register (LDTR),**
 - **interrupt descriptor table Register (IDTR), and**
 - **task state segment descriptor Register (TR).**

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.
- **The initial four registers DR0 to DR3** store four program controllable breakpoint addresses, while DR6 and DR7 respectively hold breakpoint status and breakpoint control information.

Debug Registers

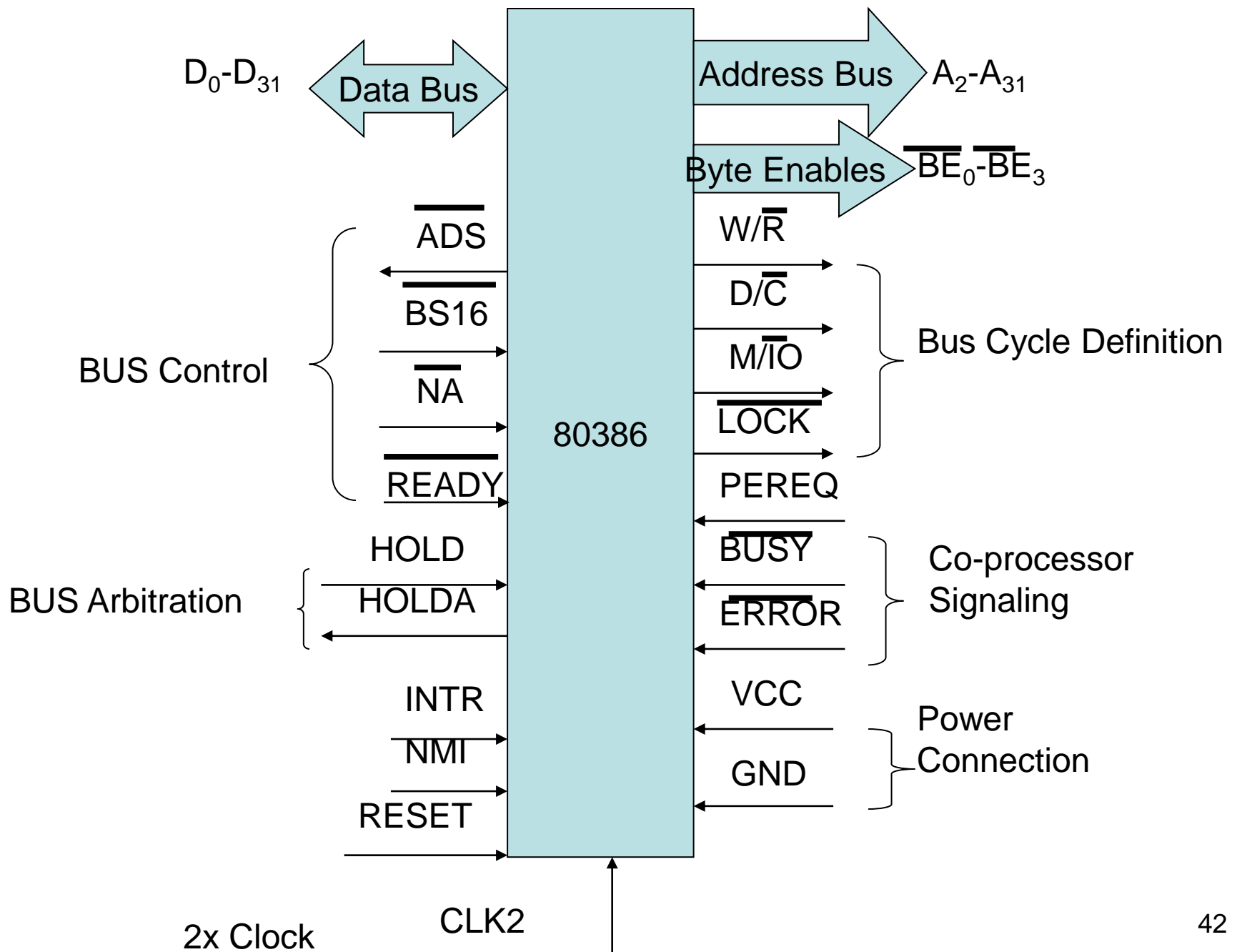
breakpoint control info	DR7
breakpoint status	DR6
RESERVED	DR5
RESERVED	DR4
Linear breakpoint address 3	DR3
Linear breakpoint address 2	DR2
Linear breakpoint address 1	DR1
Linear breakpoint address 0	DR0

Test Registers

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

386 Signals

[Hall & Rafiq]



Signals...Address Bus

- Address bus consists of A2 to A31 address lines and BE0 to BE3 byte/bank enable lines
- 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***

Signals... (I/O Ports)

- 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

Signals... (I/O Ports)

- 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.

Signals...Bus control

- **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

Signals...Co-processor Signaling

- **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

Signals...Clock

- **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
- In 2^{20} ***CLK2*** cycles it can test about 60% of its internal circuitry
- 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**
 - scale indexed addressing mode
 - scaled index mode is efficient for accessing arrays or structures
- The 80386 supports overall eleven addressing modes to facilitate efficient execution of higher level language programs.
- In case of all those modes, the 80386 can now have 32-bit immediate or 32- bit register operands or displacements.

ADDRESSING MODES

- The 80386 has a family of scaled modes.
- 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.
- The different scaled modes are as follows.

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.
- ***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, [EDI+2*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

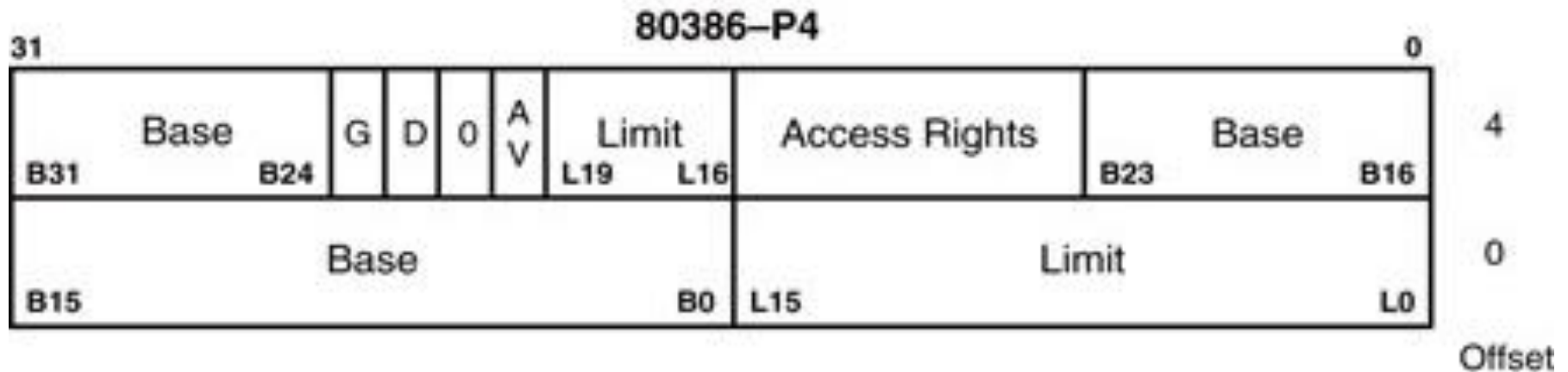
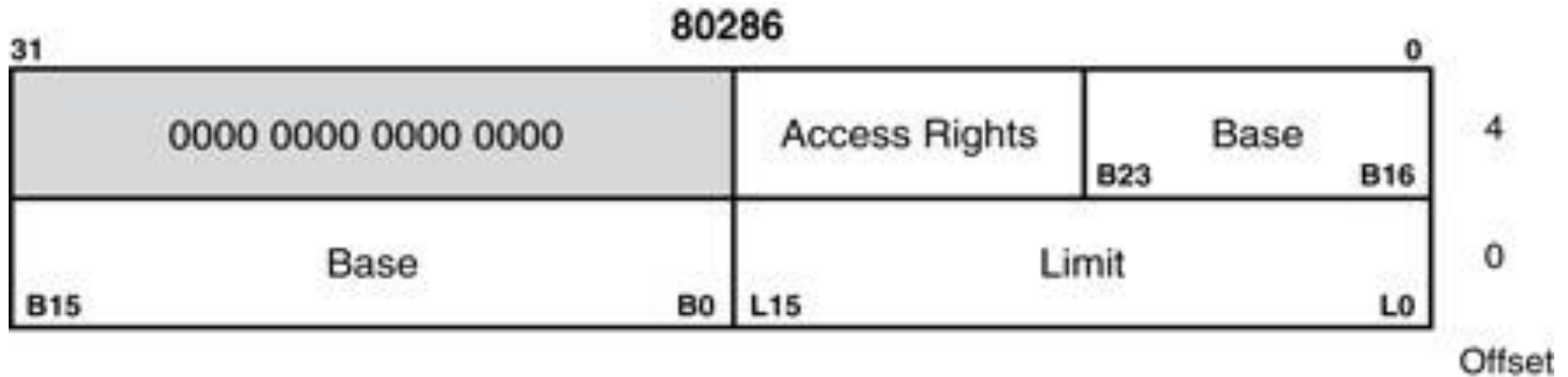
386 modes of operation

- 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 is disabled, then linear address produced by segment unit is used as physical address
- otherwise the paging unit 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



Granularity (G) bit

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

Default 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 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)₇₁

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

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 be used to execute the 8086 programs along with the capabilities of 386, like protection and a few additional instructions.
- Once the 386 enters the protected mode from the real mode, it cannot return back to the real mode without a reset operation.

Virtual 86 mode...

- Thus, the virtual 8086 mode of operation of 386, offers an advantage of executing 8086 programs while in protected 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

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
- swapping 4K pages is faster

How Paging mechanism works?

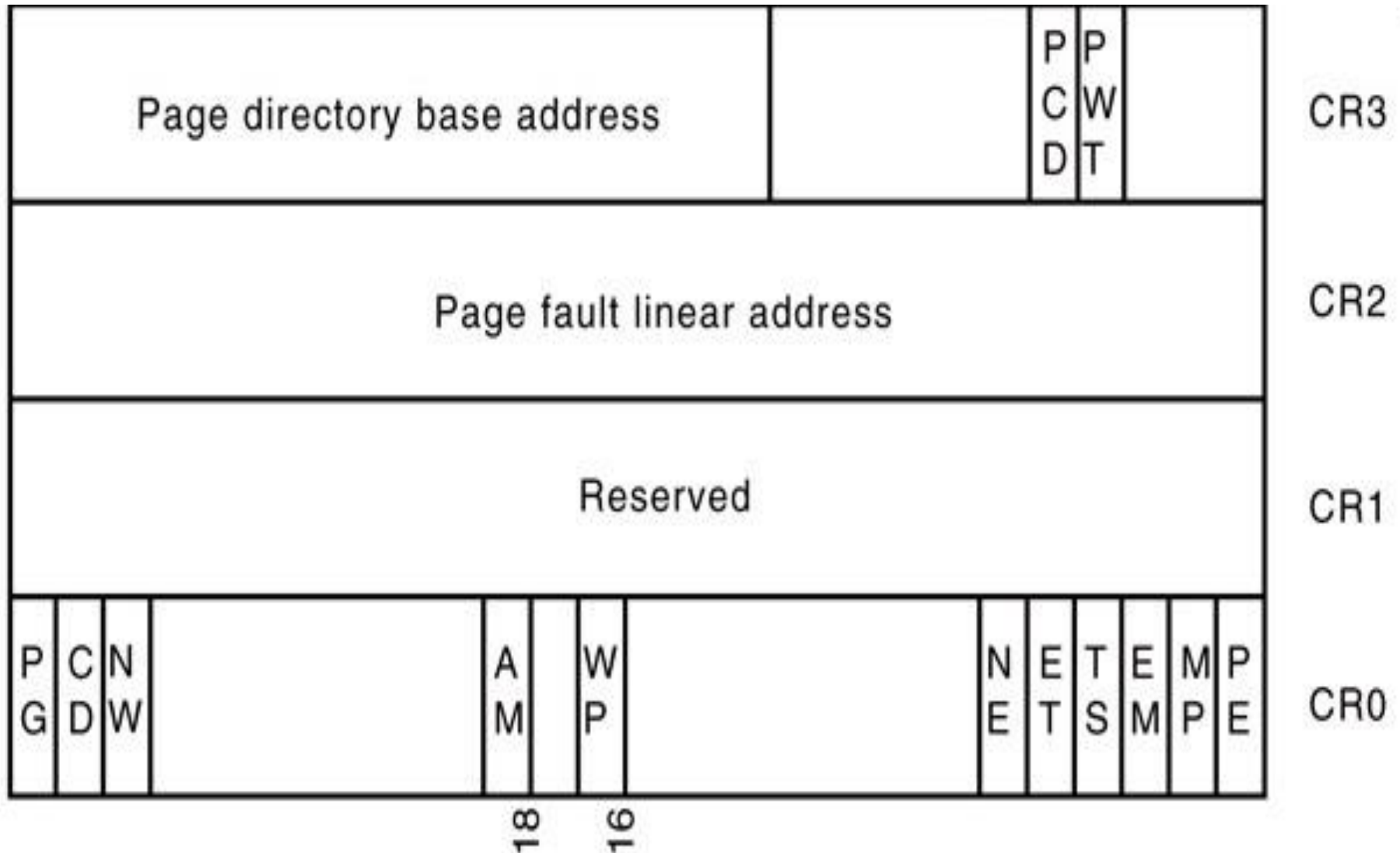
- 4G memory is divided into 1M pages of 4K
- The starting addresses of these 1M pages are kept in two-level tables
- First level table, known as ***Page Directory***, contains 1K entries for the second level table, known as ***Page Tables***
- Each ***Page Table*** contains the ***starting addresses***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

How Paging mechanism works?...

Page Directory

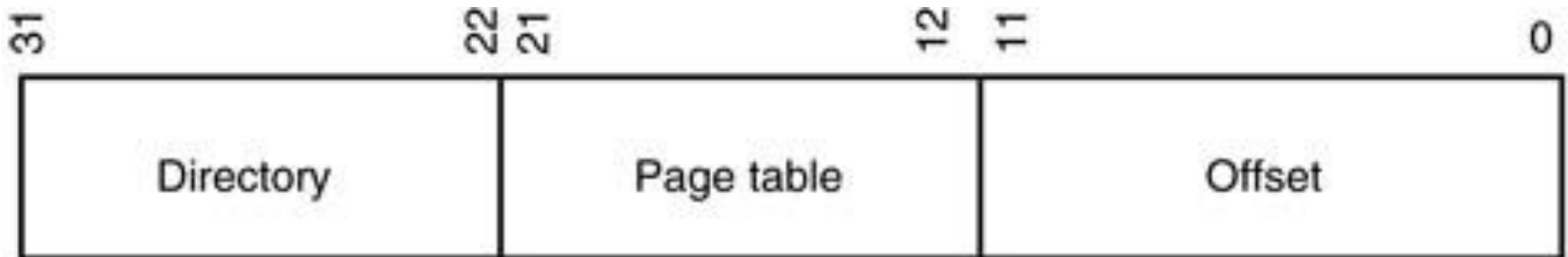
- There is only one ***Page Directory*** in the system
- Starting address of ***Page Directory*** is in CR3
- The paging unit is controlled by the contents of the microprocessor's control registers.
- 386 has 4 control registers CR0 through CR3.
- CR1 of 386 are kept reserved by Intel

386 control registers

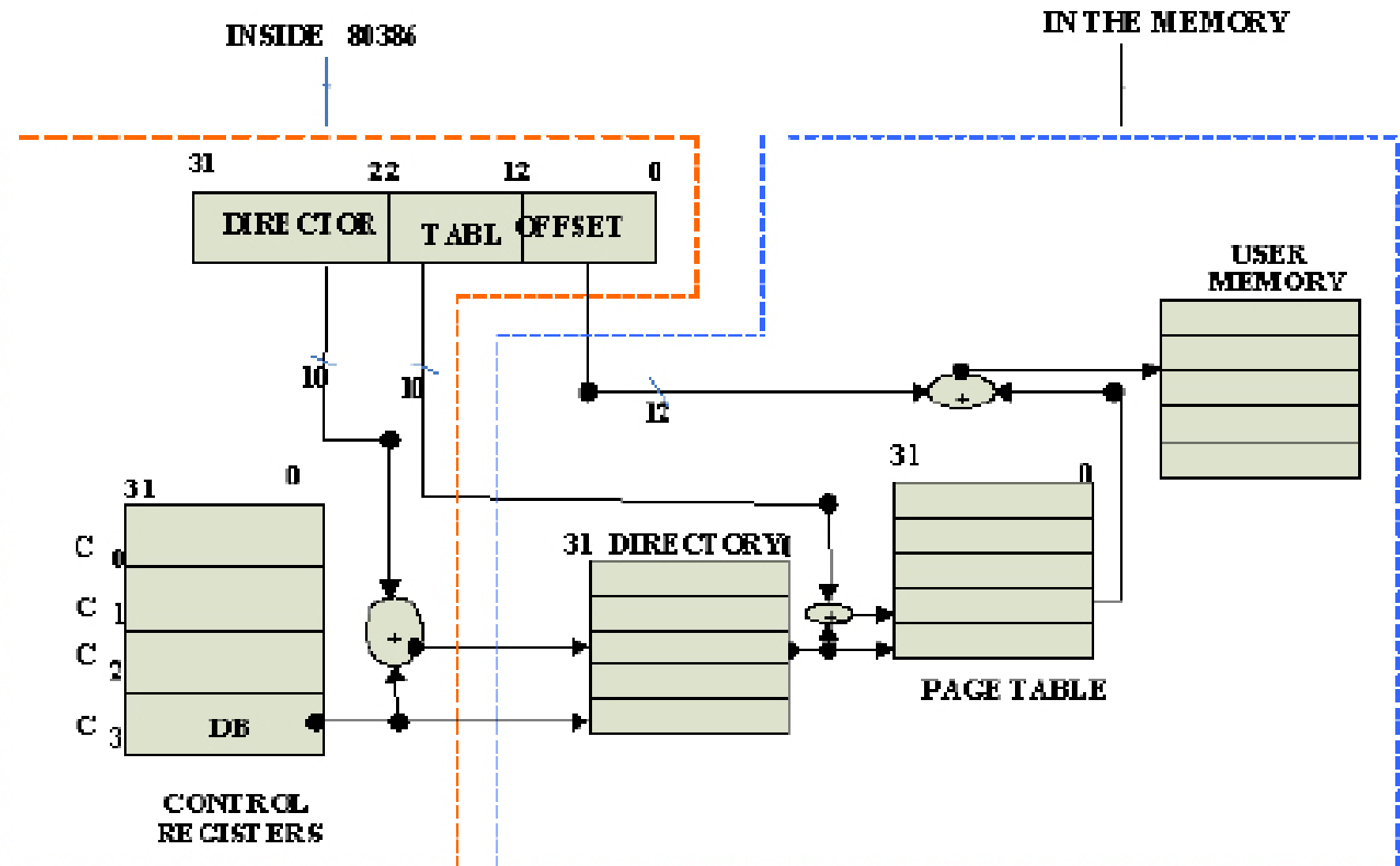


How Paging mechanism works?...

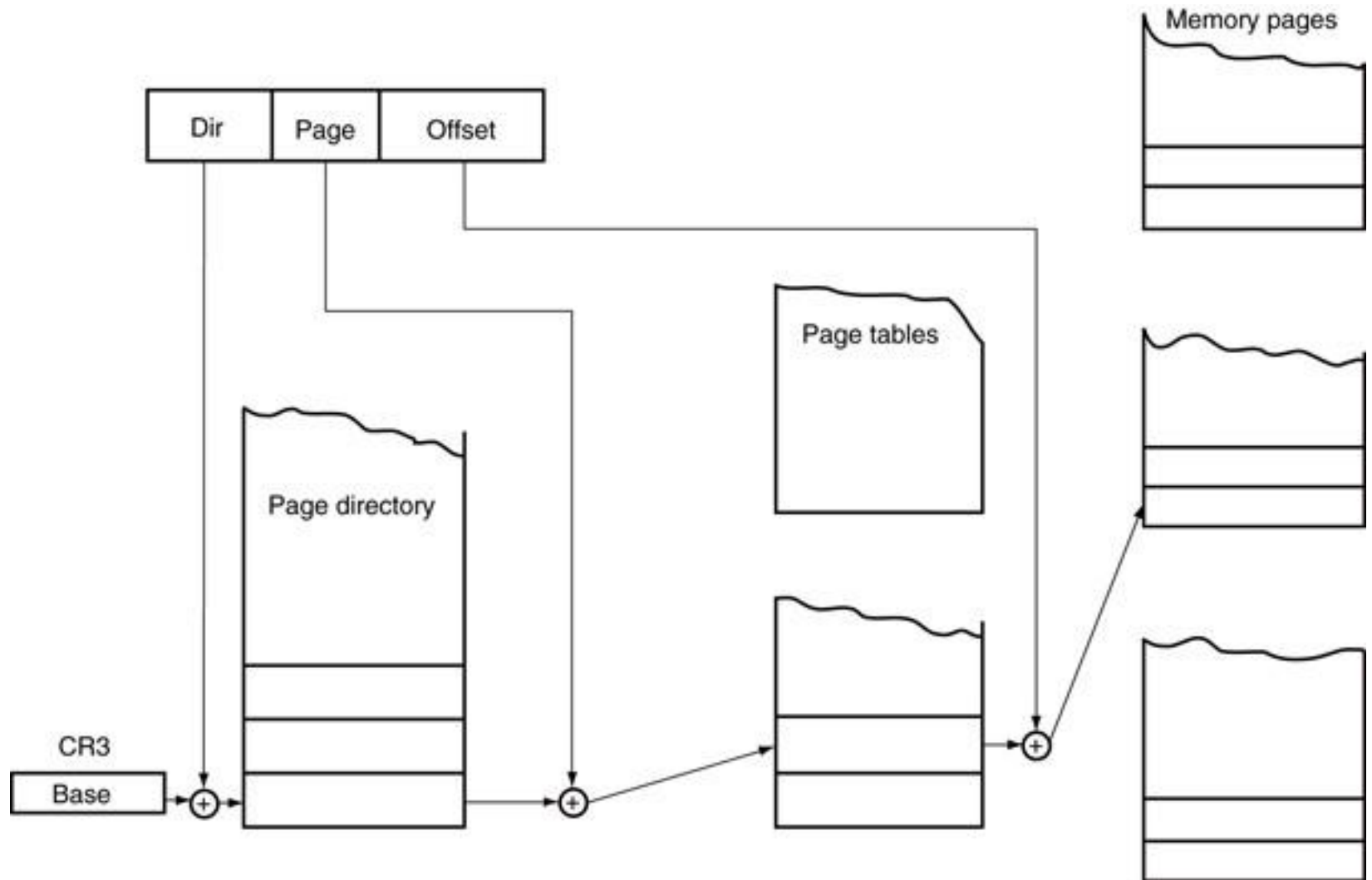
- The 32-bit linear address, as generated by software, is broken into three sections that are used to access the operand in the memory



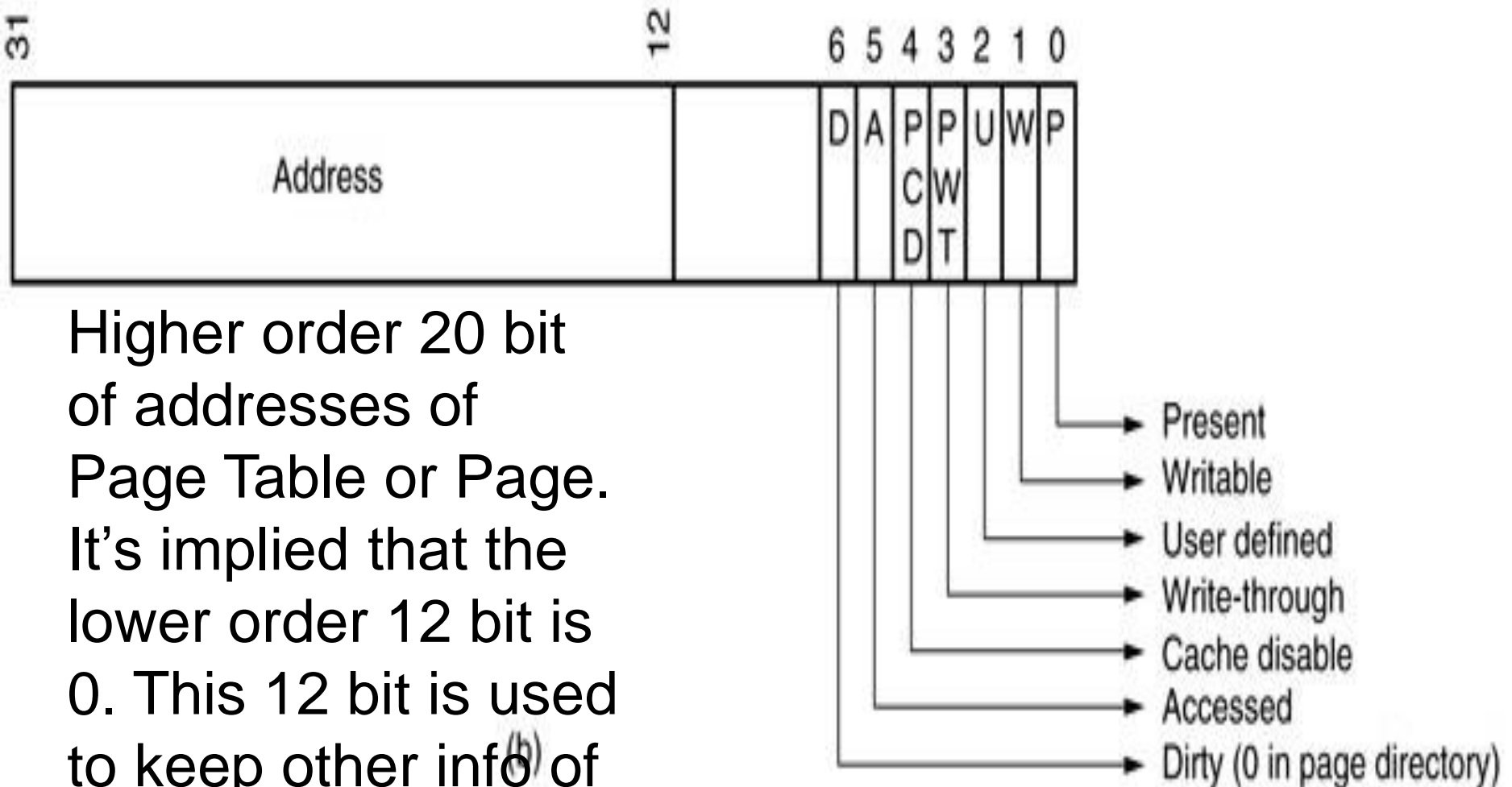
The paging mechanism of 386



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 (level 0, 1, and 2) 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.

80486

386 + 387 = 486

- only new idea here is the introduction of **8K cache**
 - used for both data and instructions

Pentium

Pentium

- was expected in '92
- Introduced in '93
- Still 32 bit processor but has
 - 64 bit data bus
 - 32 bit address bus
- Pentium signals an improvement to the architecture found in 486

Pentium improvements

- Improved cache structure
 - Re-organized to **two caches** that are each 8K bytes in size
 - One for caching **data**
 - Another for caching **instructions**
- Wider data bus width
 - Increased from 32 bit to **64 bits**

Pentium improvements...

- **Faster** numeric processor
 - Operates about 5 times faster than the 486 numeric processor
- **A dual integer processor**
 - Often allows **two instructions per clock**
- Branch prediction logic
- MMX instructions a later introduction to Pentium

PIPELINING in Pentium

- Pentium has two pipelines
 - **U pipeline**
 - U-pipeline can execute any Pentium instruction
 - **V pipeline.**
 - V-pipeline only executes *simple* instructions.
- Each pipeline has 5 stages
 - i. Pre-fetch
 - ii. Instruction Decode
 - iii. Address Generation
 - iv. Execute, Cache, and ALU Access
 - v. Write back

PIPELINING in Pentium...

pipeline stages

- **Pre-fetch:** Instructions are fetched from the instruction cache and aligned in prefetch buffers for decoding.
- **Instruction Decode:** Instructions are decoded into the Pentium's internal instruction format. Branch prediction also takes place at this stage.

PIPELINING in Pentium...

pipeline stages...

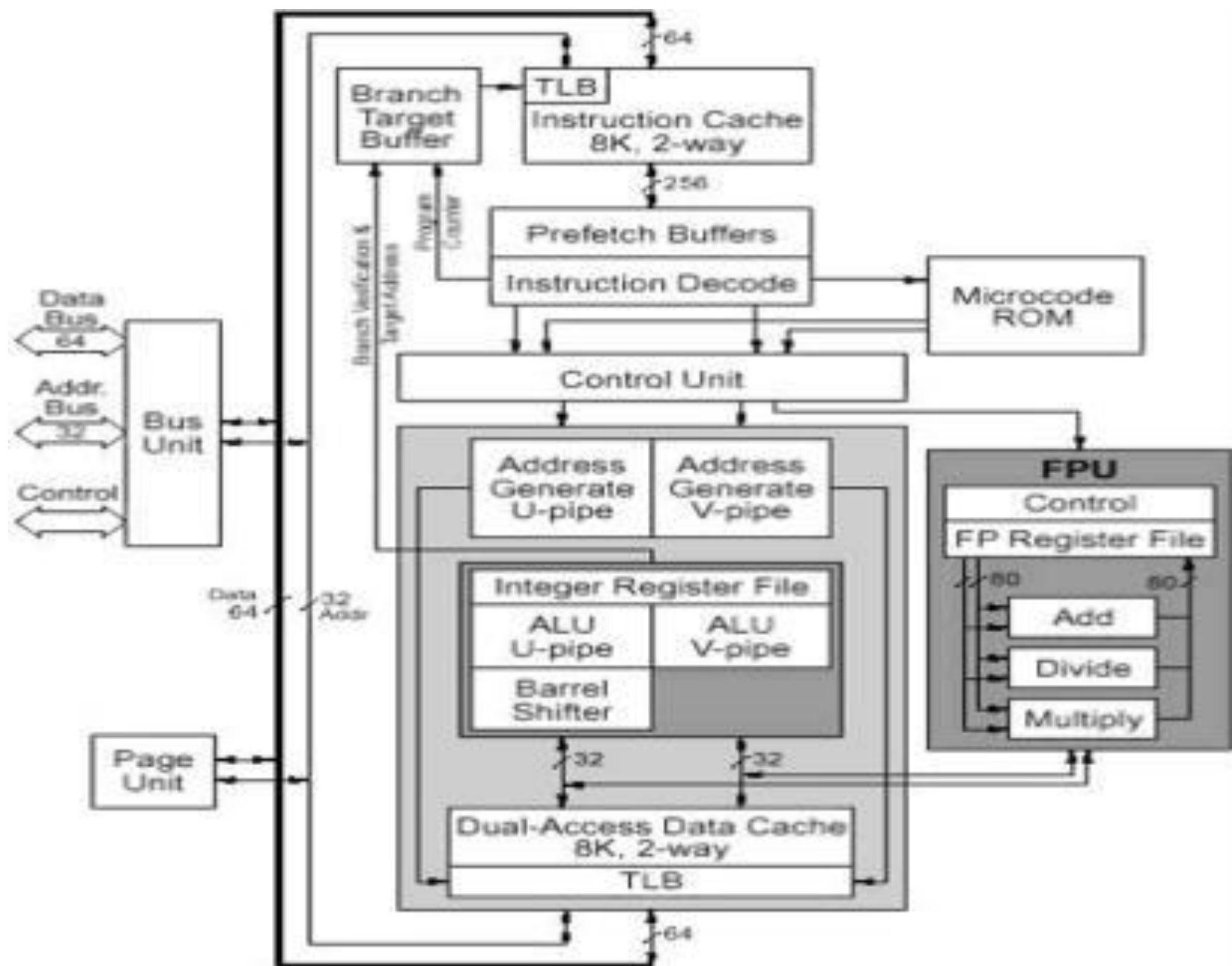
- **Address Generation:** Address computations take place at this stage.
- **Execute, Cache, and ALU Access :** The integer hardware executes the instruction.
- **Write-back:** The results of the computation are written back to the register file.

superscalar

- Any Processor capable of parallel instruction execution of multiple instructions is known as **superscalar processor**.
- As, in Pentium, there are two execution lines
--- U-line and V-line,
–so Pentium is a ***Super Scalar Processor***

Pentium Architecture

- Now we will look into Pentium's internal Block diagram



Floating Point Unit (FPU)

- has 8 80-bit general purpose floating point registers, ST(0) through ST(7)
- has 8-stage pipeline
 - First 5 stages are identical to U/V pipelines
 - 2 additional execution stages
 - First execution stage (X1 stage)
 - Second execution stage (X2 stage)
- In these two stages FPU reads the data from data cache and executes the floating point computation
- 1 additional error reporting stage

Inquire Cycles

- used to maintain cache coherency in a multiprocessor system
- Pentium watches the system bus in a multiprocessor system
 - known as ***bus snooping***
- If it detects a memory read/write operation by another CPU, it runs an internal inquire cycle to determine whether the address on the bus is stored in one of its internal cache
 - If so, the cache needs to be updated
- To implement bus snooping, address bus needs to be bidirectional

Pentium Memory System

- arranged in **8 banks**
 - each bank stores **1 byte of data with *parity bit***
 - BE0 to BE7 selects the banks
- new feature added to Pentium is its capability to generate and check **parity** for address bus

Branch Prediction

- Branch instructions occur on the average 15% to 25% of time
- They change the normal sequential control flow of the program and may stall the pipelined execution in Pentium
- Performance increases by 25%

Cache structure

- Problem found in 486
 - Data intensive programs quickly filled up cache with data
 - Leaving little room for instruction
- Soln in Pentium
 - Separate cache for data and instructions

MMX

- Multi Media Extension
 - Introduced mainly for applications like
 - Video
 - 2D/3D graphics
 - Image processing
 - Speech synthesis
 - Telephony etc
- They have their own data formats and instruction sets

Superscalar Architecture

- Pentium is organized with 3 execution units:
 - One FPU
 - U-pipeline
 - V-pipeline
- So can execute 3 instructions simultaneously

Special Registers

- Brey 18.2 7th Edition

Special Registers

- **Segment Descriptor Registers**
 - These are not available for programmers, rather they are internally used to store the descriptor info like physical base address of segment, its limit and attributes.
 - Six segment registers have corresponding six descriptor registers
 - These are automatically loaded when the corresponding segments are loaded with selectors

Special Registers (cont...)

- **System Address Registers (32 bit)**
 - 4 special registers are defined to refer to the descriptor tables
 - 4 types of descriptor tables are
 - GDT (Global Descriptor table)
 - LDT (Local Descriptor table)
 - IDT (Interrupt Descriptor table)
 - TSS (Task State Segment Descriptor)
 - Corresponding registers are named as GDTR, LDTR, IDTR and TR respectively

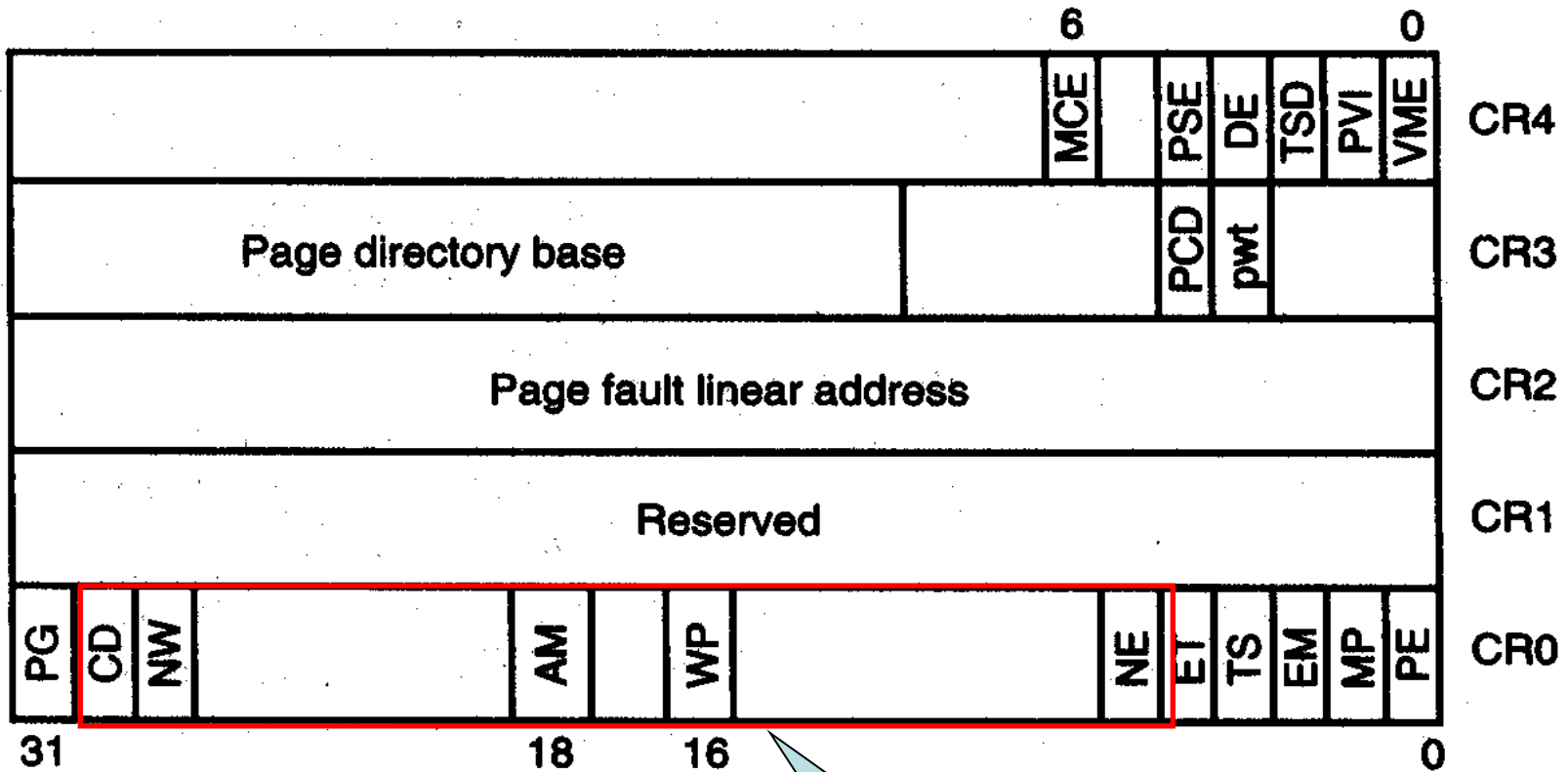
Special Registers (cont...)

- **Debug and Test Registers (DR0-DR7)**
 - 8 32-bit registers for hardware debugging
 - DR0 to DR3 store four program controllable breakpoint addresses
 - DR4 and DR5 are reserved
 - DR6 holds breakpoint status and
 - DR7 holds breakpoint control info
- Two more Test Registers (TR6,TR7), namely Test Control and Test Status Register are provided for page caching

Special Registers (cont...)

- **Control Registers**

- 4 32-bit control registers, CR0-CR4, to hold global machine status independent of the executed task
- Load and store instructions are available to access these registers
- CR1 is reserved
- CR4 is new to Pentium



Control Register Bits

Control Register CR0 Bits

- **CD (Cache disable)** controls the internal cache.
- If $CD = 1$, the cache will not fill with new data for cache misses, but it will continue to function for cache hits.
- If $CD = 0$, misses will cause the cache to fill with new data.

cache writing policy

- **write-through** : writing is done synchronously both to the cache and to the backing storage
- **write-back (or write-behind)** : initially writing is done only to the cache. The writing to backing storage is only done when this data is evicted from cache

Control Register Bits

Control Register CR0 Bits

- **NW (Not write-through)** selects the mode of operation for the data cache.
- If $NW = 1$, the data cache is inhibited from cache write-through.
- **AM (Alignment mask)** enables alignment checking when set. Note that alignment checking occurs only for protected mode operation when the user is at privilege level 3.

Control Register Bits

- **WP (Write protect)** protects user level pages against supervisor level write operations. When $WP = 1$, the supervisor can write to user level segments.
- **NE (Numeric error)** enables standard numeric coprocessor error detection. If $NE = 1$, the FERR pin becomes active for a numeric coprocessor error. If $NE = 0$, any coprocessor error is ignored.
- **VME (Virtual mode extension)** enables support for the virtual interrupt flag in protected mode. If $VME = 0$, virtual interrupt support is disabled.

Control Register Bits

- **PVI (Protected mode virtual interrupt)** enables support for the virtual interrupt flag in protected mode.
- **TSD (Time stamp disable)** controls the RDTSC instruction.
- **DE (Debugging extension)** enables *I/O* breakpoint debugging extensions when set.
- **PSE (Page size extension)** enables 4M-byte memory pages when set.
- **MCE (Machine check enable)** enables the machine checking interrupt.

New Flags in Pentium

- **AC (Alignment Check Flag)**
- Same as AM bit in CR0

New Flags in Pentium

- **VIF (Virtual Interrupt Flag)**
 - This is a copy of the interrupt flag bit
- **VIP (Virtual Interrupt Pending)**
 - provides info about a virtual mode interrupt.
 - This is used in multitasking environments to provide the OS with virtual interrupt flags and interrupt pending info

New Flags in Pentium (cont...)

- **ID (Identification)**
 - indicates that the processor support **CPUID** instruction
 - **CPUID** provides the system with info about the processor, such as its version number, manufacturer

Built-in Self Test (BIST)

- The built-in self-test (BIST) is accessed on power-up by placing a logic 1 on INIT while the RESET pin changes from 1 to 0.
- The BIST tests 70 percent of the internal structure of the Pentium in approximately 150μs
- Upon completion of the BIST, the Pentium reports the outcome in register EAX.
- If EAX = 0, the BIST passed and the Pentium is ready for operation.
- If EAX contains any other Value, the Pentium has malfunctioned and is faulty.

Pentium Memory Management

- Many of the features of earlier 386 & 486 are unchanged in Pentium
- The main change is
 - in the paging unit and
 - a new **system memory-management mode**

Paging Unit

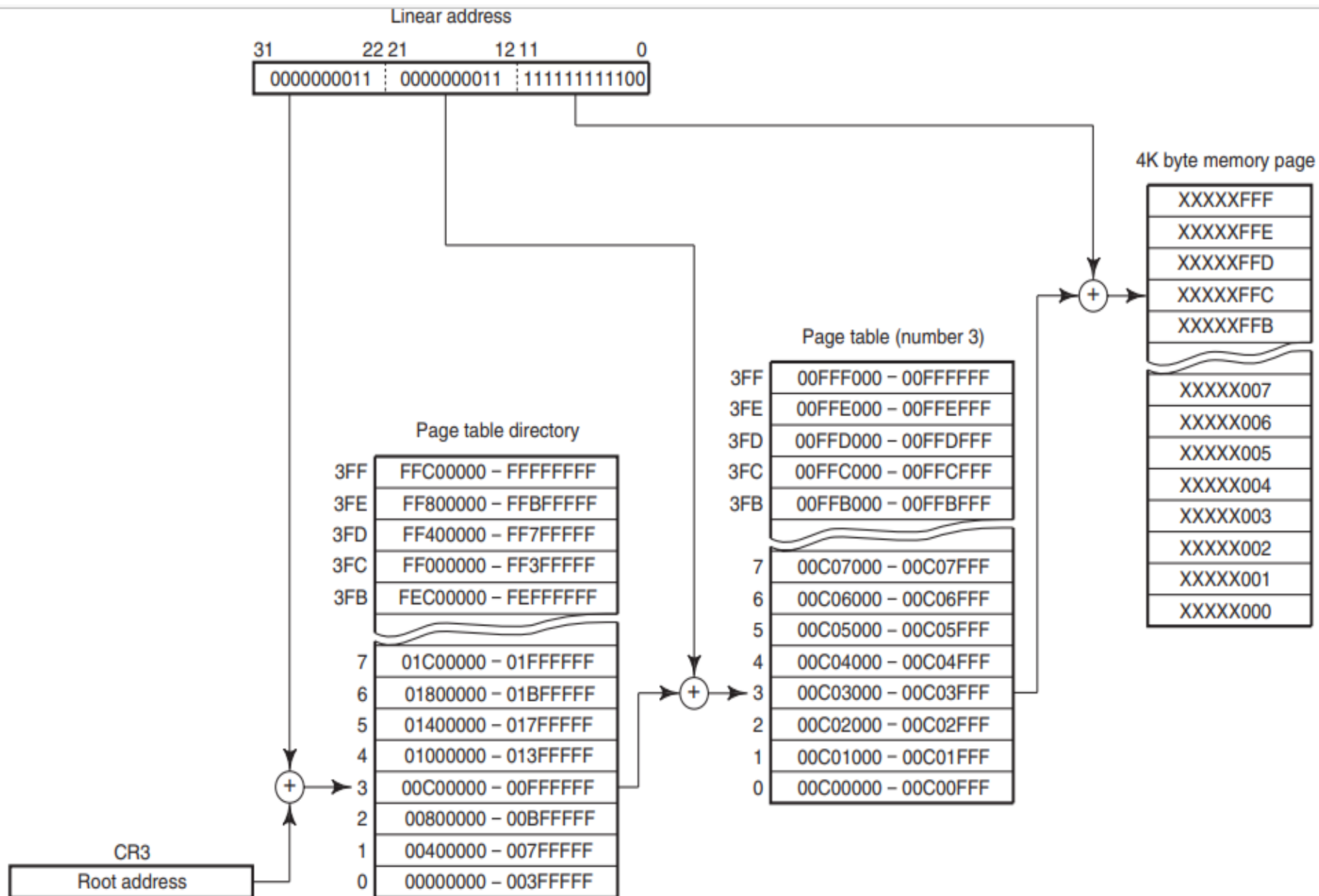
- The paging mechanism functions with older **4KB memory pages**
or
- With a **new** extension available to Pentium with **4MB memory pages**

Paging Unit (cont...)

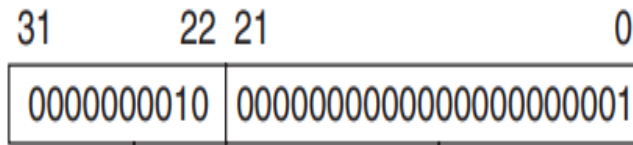
- The size of the paging table structure can become large in a system that contains a large memory
- For a 386 system with 4G memory, slightly over 4M is required just for the page tables
- In Pentium with new 4M paging feature, this is drastically reduces to just a single page table
- The new 4M page size is selected by the PSE bit in CR4

Paging Unit (cont...)

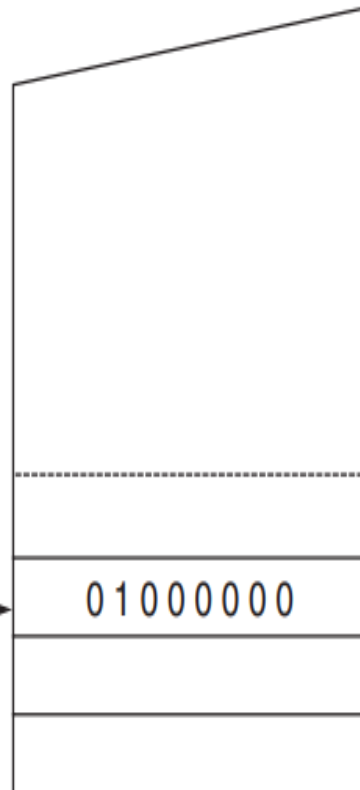
- Explain the differences between 4K & 4M paging scheme with diagram



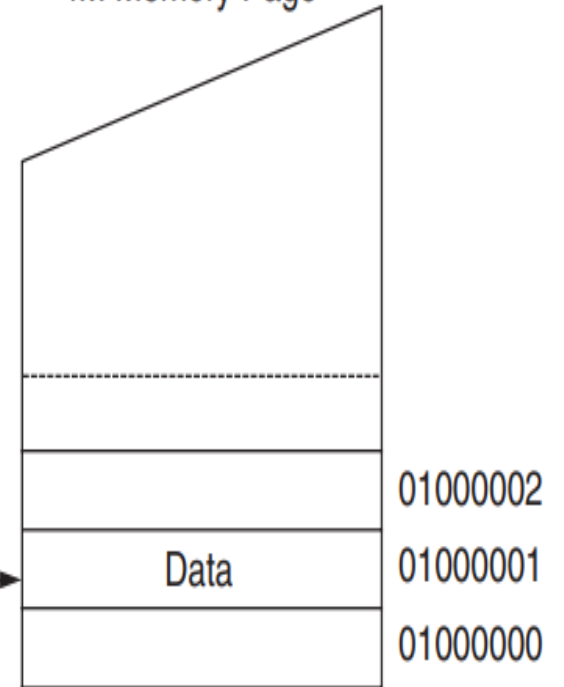
Linear Address



Page Directory



4M Memory Page



CR3



+

+

Data

Memory Management Mode

- System Memory Management Mode (SMM) is the fourth mode of operation for Pentium besides the other three modes of operation of earlier processors viz.,
 - Real mode
 - Protected mode
 - Virtual Mode

Memory Management Mode (cont...)

- SMM is not intended to be used
 - as an application or
 - a system level feature
- It is intended for high-level system functions such as
 - Power management
 - Security etc

Memory Management Mode (cont...)

- Access to SMM is accomplished via a new external hardware interrupt applied to the SMI# pin on the Pentium
- When the SMM interrupt is activated, the processor begins executing system level software in an area of memory called the *system management RAM* or *SMRAM*.
- The SMI interrupt disables all other interrupts normally handled by user applications and the operating system.

Memory Management Mode (cont...)

- A return from the SMM interrupt is accomplished with a new instruction, RSM.
- RSM returns from the memory-management mode interrupt and returns to the interrupted program at the point of the interruption.
- The SMM interrupt calls the software, initially stored at memory location 38000H, using CS = 3000H and EIP = 8000H.
- This initial state can be changed using a jump to any location within the first 1 M byte of memory.

Memory Management Mode (cont...)

- An environment similar to real mode memory addressing is entered by the management mode interrupt, but different because instead of being able to address the first 1 M of memory, SMM mode allows the Pentium to treat the memory system as a flat 4G- byte system.
- In addition to executing software that begins at location 38000H, the SMM interrupt also stores the state of the Pentium in what is called a *dump record*.
- The dump record is stored at memory locations 3FFA8H through 3FFFFH, with an area at locations 3FE00H through 3FEF7H reserved by Intel.

NEW PENTIUM INSTRUCTIONS

<u>Instruction</u>	<u>Function</u>
CMPXCHG8B	Compare and exchange eight bytes
CPUID	Return the CPU identification code
RDTSC	Read time stamp counter
RDMSR	Read model specific register
WRMSR	Write model specific register
RSM	Return from system management interrupt

Pentium Pins and Signals

Brey

Pentium Pins and Signals

- Packaged in 273 PGA
 - Data Bus 64 pins
 - Address Bus 29 pins
 - Control Bus 75 pins
 - Vcc+Ground 99 pins
 - No Connection (NC) 6 pins

Data Bus

- **D₆₃ through D₀** **(Data Bus)**
- **BIDIRECTIONAL**
- These signals make up the Pentium's 64-bit bidirectional data bus.
- can transfer **byte, word, double-word** and **quad-word**
- The actual data lines in use during a particular bus cycle are indicated by the BE# outputs.

Address Bus

- **A3 through A31 (Address Lines)**
- **output/input**
 - These 29 address lines, together with the byte(bus) enable outputs BE_0 - BE_7 , form the Pentium's 32-bit address bus.
 - A memory space of 4 GB is possible, along with 65536 I/O ports.
- The address lines are used as **input** during an **inquire cycle** to read an address *into* the Pentium, for examination by the internal cache.

Address Bus (cont...)

- **BE₀# through BE₇#** **output**
- **Byte(Bank) Enable**
- These, together with A3 through A31, make up the 32-bit address output by the Pentium.
 - Each byte enable is used to control a different 8-bit portion of the processor's 64-bit data bus.
 - BE_i enables Bank_i

Control Signals

Data Related

- **DP0 through DP7** **Data Parity**
- **BIDIRECTIONAL**
- During memory write operation, CPU sends 64-bit data to memory along with **even** parity bit for each byte of data.
- CPU also stores these parity bits in a separate parity data memory bank.
- DP0 represents the parity of the lower byte ($D_0 - D_7$).
- During the future memory read operation for the same data, stored parity is compared with new calculated parity.
- If there is any mismatch, the **PCHK#** pin is asserted.

Data Related (cont...)

- **PCHK# (Data Parity Check)**
- **Output**
- This goes low if the Pentium detects a parity error on the data bus.
- External hardware must take the appropriate action to handle the error.

Address Related Control Signals

- **A20M# (Address 20 Mask) input**
- This is used to force the Pentium to limit addressable memory to 1MB, to emulate the memory space of the 8086.
- A20M# may only be active in real mode.

Address Related Control Signals...

- **AP (Address Parity)** **Bidirectional**
- This is used to indicate the **even parity** of address lines A_3 through A_{31} .
- AP is used as an **output** when the Pentium is outputting an address, and
- as an **input** during an inquire cycle.

Address Related Control Signals...

- **APCHK# (Address Parity Check)**
- **output**
- This goes low if the Pentium discovers a parity error on the address lines.
- External circuitry is responsible for taking the appropriate action if a parity error is encountered.

Address Related Control Signals...

- **ADS# (address data strobe) output**
- Indicates a valid memory/IO address in the bus

Address Related Control Signals...

- **AHOLD (address hold) input**
- causes to hold the address and AP signals for the next clock cycle.

Timing Signal

- **CLK (Clock)** **Input**
- This is the clock input to the processor.
- CLK must be stable (running at the desired frequency) within 150 milliseconds of power on.

Timing Signal (cont...)

- **PICCLK (Programmable Interrupt Controller Clock) input**
- This controls the serial data rate in the internal APIC interrupt controller.
- ***STPCLK# (Stop Clock) input***
- When low; this causes the Pentium to stop its internal clock (thus reducing the power consumed).

Interrupts

- **INTR (Interrupt Request) input**
- This, when high, causes the Pentium to initiate interrupt processing
- An 8-bit vector number is read on the lower byte of the data bus to select an interrupt service routine.
- INTR is ignored if the interrupt enable bit in the flag's register is clear.

Interrupts...

- **NMI (Non-Maskable Interrupt)**
- **Input**
- The Pentium responds to this rising edge input by issuing interrupt vector 2.
- No external interrupt acknowledge cycles are generated.

- **RESET (Processor Reset) Input**
- This causes the Pentium to initialize its registers to known states, invalidate the code and data cache, and fetch its first instruction from address FFFFFFFF0H.
- RESET must be active for at least one millisecond after power on to ensure proper operation.

- **INIT (Initialization) Input**
- INIT is a rising edge sensitive input that causes the processor to be initialized in the same way as a RESET,
 - except that the internal registers and cache are left unchanged.

Bus Cycle Definition Group

- **D/C# (Data/Code) output**
- This output indicates that the current bus cycle is accessing code (D/C is low) or data (D/C is high).
- **LOCK# (Bus Lock) output**
- This goes low to indicate that the current bus cycle is locked and may not be interrupted by another bus master.

Bus Cycle Definition Group

- **M/IO# (Memory/Input-Output) output**
- This indicates the type of bus cycle currently starting. If M/IO# is high, a memory cycle is beginning; otherwise an I/O operation is performed.
- **W/R# (Write/Read) output**
- This is used to indicate if the current bus cycle is a read operation (W/R# is low) or a write operation (W/R# is high).

Bus Arbitration Group

- **BOFF# (Backoff)** **input**
- This causes the processor to terminate any bus cycle currently in progress and tri-state its busses.
- Execution of the interrupted bus cycle is restarted when BOFF# goes high.

Bus Arbitration Group

- **BREQ (Bus Request) output**
- This goes low when the Pentium has a pending bus cycle ready to begin.
- In a multiprocessor system, BREQ may be used to select a processor competing for the system bus.

Bus Arbitration Group

- **HOLD (HOLD Bus)** **input**
- If HOLD is high when sampled, the Pentium tri-states its bus signals and activates HLDA.
- HOLD may be used by another processor that wishes to become the bus master.

Bus Arbitration Group

- **HLDA (Bus Hold Acknowledge) output**
- The HLDA goes high (as a result of a HOLD request) to indicate that the Pentium has been placed in a hold state.
- If the code and data cache contain current instructions and operands, execution continues with no bus activity, only cache accesses.

Burst Control Group

- **BRDY# (Burst Ready)** **input**
- During a read cycle, this input indicates that data is available on the data bus.
- For write cycles, BRDY# informs the processor that the output data has been stored.
- BRDY# is used for both memory and I/O operations.
- If BRDY is not low when sampled, the Pentium inserts extra clock cycles into the current cycle (*wait* states) to provide extra time for the data transfer.

Bus Control Group

- **ADS# (Address Data Strobe) output**
- becomes active whenever Pentium has issued a valid memory or I/O address. [Brey]
- The ADS#, when low, indicates the beginning of a new bus cycle.
- Signals that define the new bus cycle are valid when ADS# is active.
- These signals include the address bus and byte enables, AP, CACHE#, LOCK#, M/IO#, W/R#, D/C#, SCYC, PWT, and PCD.

Bus Control Group (cont...)

- **BUSCHK# (Bus Check) input**
- This input, when low, indicates to the Pentium that there was a problem with the last bus cycle and was unsuccessful.
- The processor may perform a machine check exception to recover.

Bus Control Group (cont...)

- **NA# (Next Address)** **input**
- This, when *low*, indicates that the external memory system is read to accept a new bus cycle.

Cache Related

- **AHOLD (Address Hold) input**
- causes the Pentium to hold the address and Address Parity (AP) signals for the next clock. [Brey]
- This is used to place the Pentium's address bus into a high impedance state so that an inquire cycle can be run.
- The address used during the inquire cycle is read in by the Pentium when AHOLD is active.

Cache Related

- **CACHE# (Cacheability) output**
- This output indicates whether the data associated with the current bus cycle is being read from or written to the internal cache.
- Indicates that the current Pentium cycle can cache data. [Brey]

Cache Related

- **EADS# (External Address Strobe) input**
- EADS# is used to indicate that an external address may be read by the address bus during an inquire cycle.

Cache Related

- **FLUSH# (Flush Cache)** **input**
- This, when low, causes the Pentium to write-back all modified data lines in its internal code and data cache.

Cache Related

- **KEN# (Cache Enable)** **input**
- A zero on this enables caching of data being read into the processor.
- If **KEN#** is high when sampled, no caching will occur.

Cache Related

- **EWBE# (External Write Buffer Empty)**
- **input**
- This, when low, indicates that the Pentium may proceed with the next cache write-through operation.

Cache Related

- **HIT# (Inquire Cycle Cache Hit/Miss) output**
 - This indicates a cache hit (when low) as the result of an inquire cycle.
- **HITM (Hit/Miss to a Modified Cache Line) output**
 - This indicates that a modified line in the cache was hit as the result of an inquire cycle.

Cache Related

- **INV (Invalidation Request) input**
- During an inquire cycle, INV is used to determine what happens to a cache line during a hit.
- If INV is high, the cache line is invalidated.
- If INV is low, the line is marked shared.

Cache Related

- **PCD (Page Cacheability Disable) output**
- This indicates the state of CR3's PCD (page cache disable) bit.
- It is used *to control* cacheability on a page-by-page basis.

Cache Related

- **PWT (Page Write-through)** **output**
- This indicates the status of the cache's write-through paging bit in CR3.
- **WB/WT# (Write-back/Write-through)** **input**
- This defines the current cache line update protocol as write-back or write-through.
- Cache lines may be defined on a line-by-line basis with WB/WT#.

Branch Related Signals

- **BT0 through BT3 (Branch Trace) output**
- BT0 through BT2 output the lower three bits (A0 through A2) of the target address of the currently executing branch instruction. BT3 indicates the operand size of the current instruction (0 for 16-bit, 1 for 32-bit).
- These outputs become valid during a branch trace special message cycle. [Brey]₈₅

Branch Related Signals

- **IBT (Instruction Branch Taken) output**
- This goes high for one clock cycle whenever the processor takes a branch (a JNZ that jumps, for example).

Dual Processor Related Signals

- **D/P# (Dual/Primary) output**
- This output indicates the processor type in a dual-processing system.
- When low, the processor is the primary processor.
- The dual processor is indicated when high.

Dual Processor Related Signals

- **CPUTYP (CPU Type) input**
- This input is used to specify the processor type in a dual-processor system.
- When low, CPUTYP specifies the primary processor.
- The dual processor is indicated when CPUTYP is high.

Dual Processor Related Signals

- **DPEN# (Dual Processing Enable) output/input**
- This signal is an output on the dual processor and an input on the primary processor in a dual-processing system.
- During reset, DPEN# is used to initiate dual processing.

Floating Point Related

- **FERR# (Floating-Point Error) output**
- This indicates that the processor's FPU generated an error.
- FERR# is included to maintain compatibility with the error-handling mechanism of MSDOS.

Floating Point Related

- **IGNNE# (Ignore Numeric Exception) input**
- A zero on this input allows the processor to continue executing floating-point instructions, even if an error is generated.

Others

- ***FRCMC# (Functional Redundancy Checking Master/Checker) input***
- This is sampled during a RESET operation to determine whether the Pentium is operating as a master (when high) or a checker (when low).
- Two Pentiums may be connected in such a way that one is the master and the other is the checker.
- The checker checks every operation performed by the master to guarantee correct execution.
- A master/checker pair provides a measure of reliability to critical systems, such as flight control computers.

Others

- **IERR# (Internal Error) output**
- This, when low, indicates that a parity or redundancy check error has occurred.
- Parity errors may cause the Pentium to enter shutdown mode

Others

- **IU (U-Pipeline Instruction Complete)** **output**
 - This goes high for one clock cycle each time an instruction completes in the U pipeline.
- **IV (V-Pipeline Instruction Complete)** **output**
 - This goes high for one clock cycle when an instruction completes in the V pipeline.

Others

- **PEN# (Parity Enable) input**
- If this is low during the same cycle a parity error is detected, the Pentium will save a copy of the address and control signals in an internal machine check register.

Others

- **PRDY (Probe Mode Ready) output**
- This is used for debugging purposes.
- It indicates that the processor has stopped normal execution and is entering a debugging mode called **probe mode**, where it may execute debugging instructions.
- PRDY goes high in response to activity on R/S#.

Others

- **SCYC (Split Cycle Indication) output**
- This goes high to indicate that two or more locked bus cycles are performing a misaligned transfer.
- A misaligned transfer involves a 16- or 32-bit operand that is stored at a starting address that is not a multiple of four, or a 64-bit operand that does not begin at an address that is a multiple of eight.

Others

- **SMI# (System Management Interrupt)** **input**
- This negative edge sensitive input is used to generate a system management interrupt.
- System management is used to perform special functions, such as power management.

Others

- ***SMIACT# (System Management Interrupt Active) output***
- The SMIACT# output goes low in response to a SMI request, and remains low until the processor leaves system management mode.

Others

- **PM/BP0- PM/BPI, BP2 and BP3 (performance monitoring/breakpoint) output**
- The BP (breakpoint) outputs are associated with a set of internal registers called **debug** registers.
- There are eight debug registers.
- The first four (DR0 through DR3) are used to store the memory or I/O address of a program *breakpoint*.
- A breakpoint is a predefined address that the programmer chooses to help determine how a program executes.

PM/BP0- PM/BP1, BP2 and BP3 (performance monitoring/breakpoint)....

- For example, a breakpoint may be set to address 1000H to see if the program ever reads or writes to that address.
- The breakpoint outputs go high when the breakpoint address in its respective debug register matches a program-generated address.
- Two PM (performance monitoring) outputs are multiplexed with the lower two BP outputs.

PM/BP0- PM/BP1, BP2 and BP3 (performance monitoring/breakpoint)....

- These signals are active after a reset and indicate the status of two internal performance monitoring counters.
- Setting the DE (debug extensions) bit in CR4 causes these two outputs to change to BP0 and BP1.
- We will examine the operation of the debug and control registers later.

- The following five inputs are all associated with the Intel debug port.
- **TCK (Test Clock) input**
- **TDI (Test Data Input) input**
- **TDO (Test Data Output) output**
- **TMS (Test Mode Select) input**
- **TRST# (Test Reset) input**

- **TCK (Test Clock)** **input**
 - This is used to clock data into and out of the Pentium when performing a specific test procedure called boundary scan (IEEE Standard 1149.1).
- **TDI (Test Data Input)** **input**
 - This allows serial test data to be input into the Pentium. Data is clocked in on the rising edge of TCK.
- **TDO (Test Data Output)** **output**
 - Serial test information is clocked out of TDO on the falling edge of TCK.

- **TMS (Test Mode Select)** **input**
 - This is used to control the sequencing of boundary scan tests.
- **TRST# (Test Reset)** **input**
 - This, when low, resets the test controller logic.

Pentium Pro

18.5 Brey 7th Edition

INTRODUCTION

- Pentium Pro microprocessor is packaged in an immense 387-pin PGA (pin grid array).
- Pentium Pro is available in two versions.
 - One version contains a 256K level 2 cache;
 - the other contains a 512K level 2 cache.
- The notable difference in the pin-out of the Pentium Pro when compared to the earlier Pentium is that there are provisions for a 36-bit address bus, which allows access to 64G bytes of memory.
- This is meant for future use because no system today contains anywhere near that amount of memory.

Pentium Pro (cont...)

- Pentium Pro requires a single +3.3V power supply for operation.
- The power supply current is a maximum of 9.9 A for the 150 MHz version of the Pentium Pro, which also has a maximum power dissipation of 26.7 W.
- A good heat sink with considerable airflow is required to keep the Pentium Pro cool.
- As with the Pentium, the Pentium Pro contains multiple Vcc and Vss connections that must all be connected for proper operation.
- The Pentium Pro contains VccP pins (primary Vcc) that connect to +3.1 V, VccS (secondary Vcc) pins that connect to +3.3V, and Vcc5 (standard Vcc) pins that connect to +5.0V.
- There are also some pins that are labeled N/C (no connection) and must not be connected.

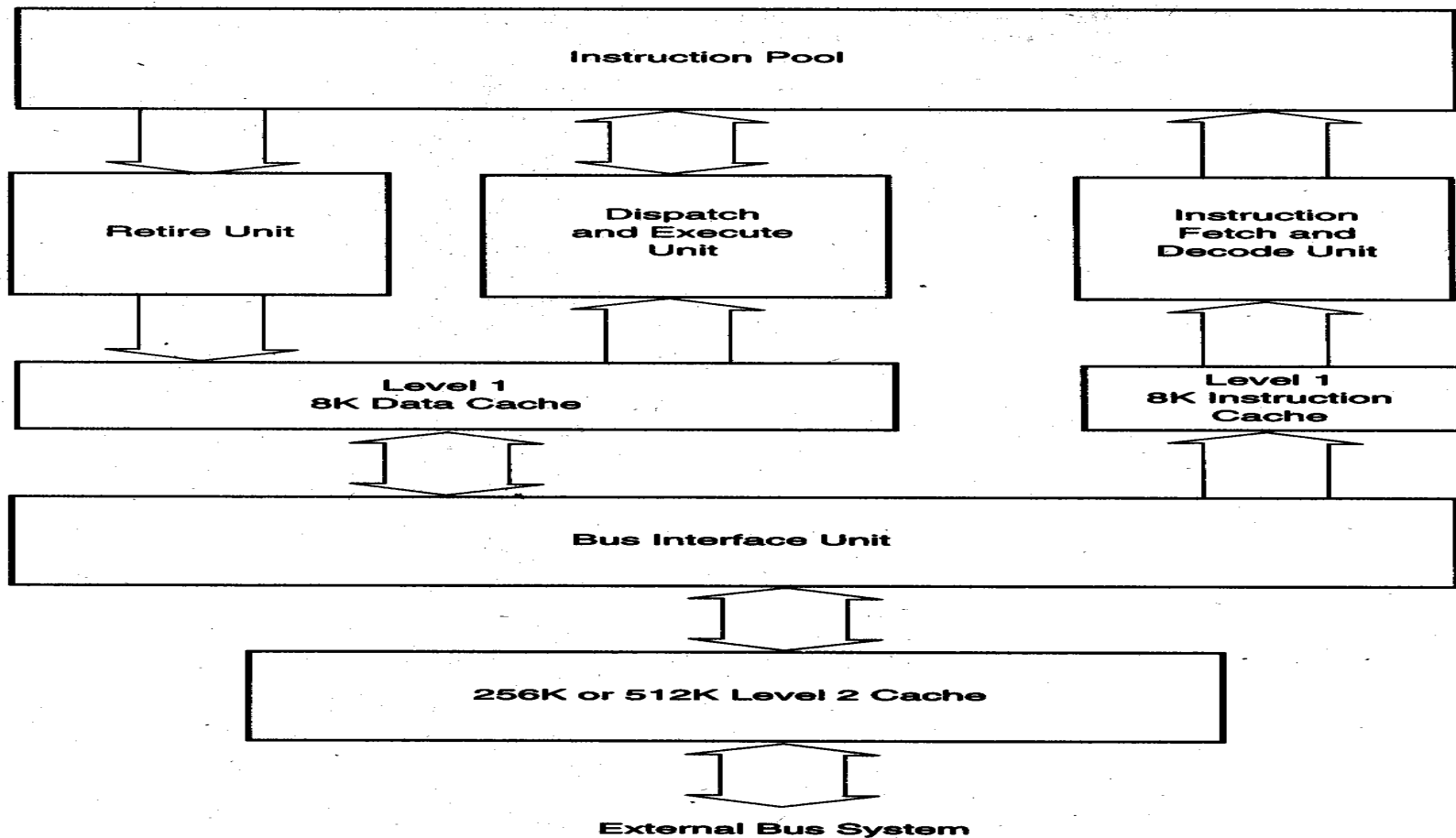
Pentium Pro (cont...)

- Each Pentium Pro output pin is capable of providing an ample 48.0 mA of current at a logic 0 level.
- This represents a considerable increase in drive current compared to the 2.0 mA available on earlier microprocessor output pins.
- Each input pin represents a small load requiring only 15 μ A of current.
- Because of the 48.0 mA of drive current available on each output, only an extremely large system requires bus buffers.

Pentium Pro (cont...)

Internal Structure of the Pentium Pro

- The Pentium Pro is structured differently than earlier microprocessors.



Pentium Pro (cont...)

- The system buses, which communicate to the memory and I/O, connect to an internal level 2 cache that is often on the main board in most other microprocessor systems.
- The level 2 cache in the Pentium Pro is either 256KB or 512KB.
- The integration of the level 2 cache speeds processing and reduces the number of components in a system.
- The bus interface unit (**BIU**) controls the access to the system buses through the level 2 cache, as it does in most other microprocessors.
- Again, the difference is that the level 2 cache is integrated. The BIU generates the memory address and control signals and passes and fetches data or instructions to either a level 1 data cache or a level 1 instruction cache.
- Each of these are presently 8KB in size

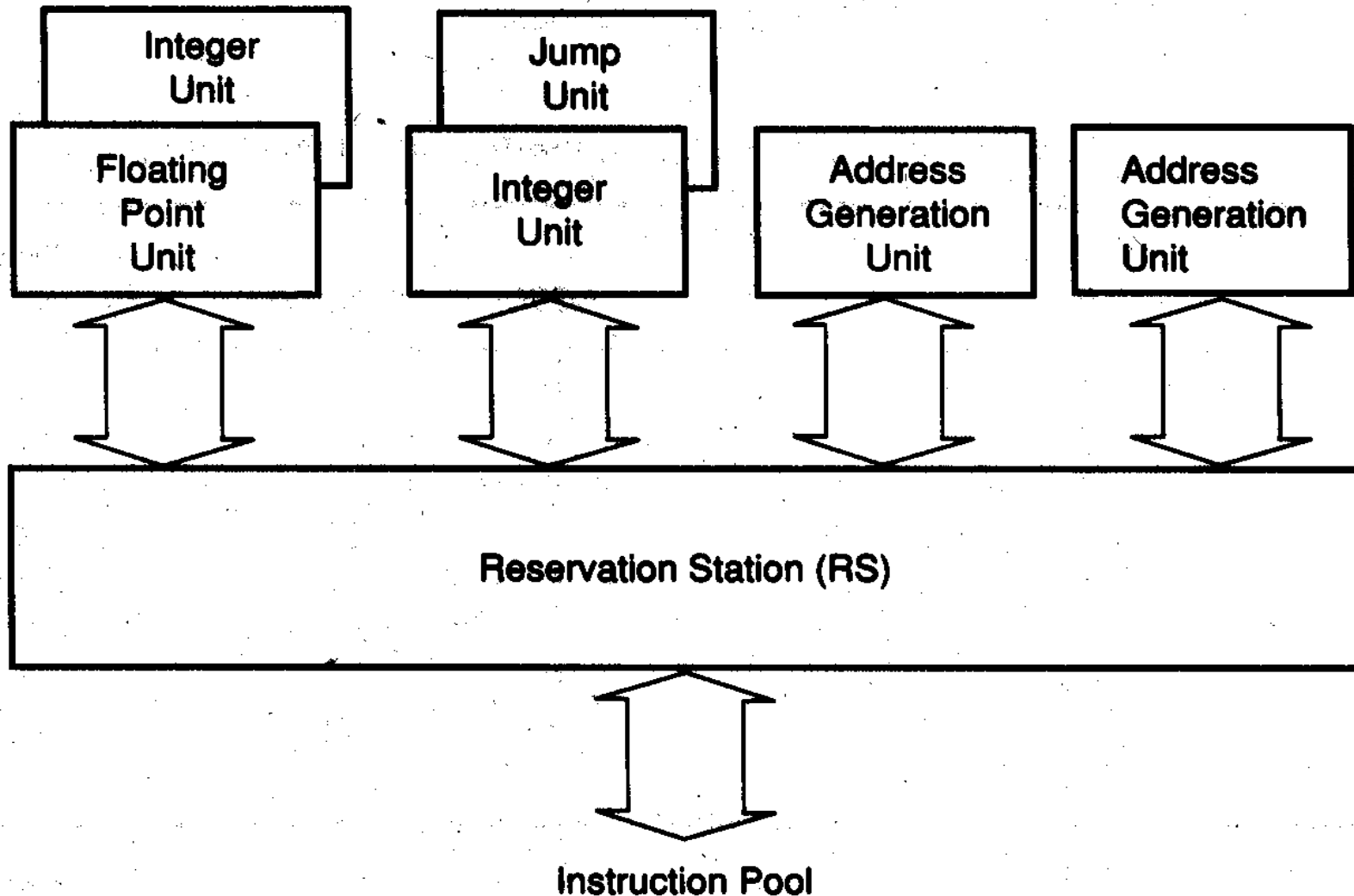
Pentium Pro (cont...)

- The instruction cache is connected to the instruction fetch and decode unit (**IFDU**).
- Although not shown, the IFDU contains three separate instruction decoders that decode three instructions simultaneously.
- Once decoded, the outputs of the three decoders are passed to the **instruction pool**, where they remain until the **dispatch and execution unit** or the **retire unit** obtains them.
- Also included within the IFDU is a branch prediction logic section, which looks ahead in code sequences that contain conditional jump instructions.

Pentium Pro (cont...)

- Once decoded, instructions are passed to the **instruction pool**, where they are held for processing.
- The instruction pool is a content-addressable memory, but Intel never states its size in the literature.
- The **dispatch and execute unit (DEU)** retrieves decoded instructions from the instruction pool, when they are complete, and executes them.
- The internal structure of the DEU is in next slide.
- The **Retire Unit (RU)** checks the instruction pool and removes decoded instructions that have been executed.
- The RU can remove three decoded instructions per clock pulse.

Internal structure of DEU



Pentium Pro (cont...)

- Notice that the DEU contains three instruction execution units:
 - two for processing integer instructions, and
 - one for floating-point instructions.
- This means that the Pentium Pro can process two integer instructions and one floating-point instruction simultaneously.
- The DEU also contains a jump execution unit and address generation units
- The reservation station (RS) can schedule up to five events for execution and can process four simultaneously.

Pentium Pro (cont...)

The Memory System

- The memory system for the Pentium Pro microprocessor is 4G bytes in size
- But access to an area between 4G and 64G is made possible by additional address signals A32-A35.
- Note that the additional memory is enabled with bit position 5 of CR4 and is accessible only when 2M paging is enabled.
- Note that 2M paging is new to the Pentium Pro to allow memory above 4G to be accessed.

Pentium Pro (cont...)

The Memory System

- New to Pro is a built-in error correction circuit (ECC)
- Can correct one-bit error and detect two-bit error
- Extra 8 bits are needed for this purpose
- So memory now becomes 72 bit wide to store 64 bit data
 - Extra 8 bits contain the error correction codes
- More reliable than older parity scheme
- Additional cost is required

Pentium Pro (cont...)

SPECIAL PENTIUM PRO FEATURES

- In Pro some additional features and changes to the control register set have occurred.
- CR4 has two new control bits that are added to the control register array
 - 5th bit PAE
 - 7th bit PGE
- PAE Page address extension enables address lines A35-A32 whenever a special new addressing mode, controlled by PSE, is enabled for the Pentium Pro.
- PGE controls the new, larger 64G addressing mode whenever it is set along with PAE and PSE.

Pentium II

Pentium II

- Extension to Pro architecture with some differences
 - Internal cache in P II has been moved out of the chip
 - P II is not available as a single chip
 - Rather is available on a small plug-in circuit board along with level 2 (L 2) cache chip
- Various versions are available
 - Celeron is a version without L2 cache
 - Xeon is enhanced by having up to 2M L2 cache

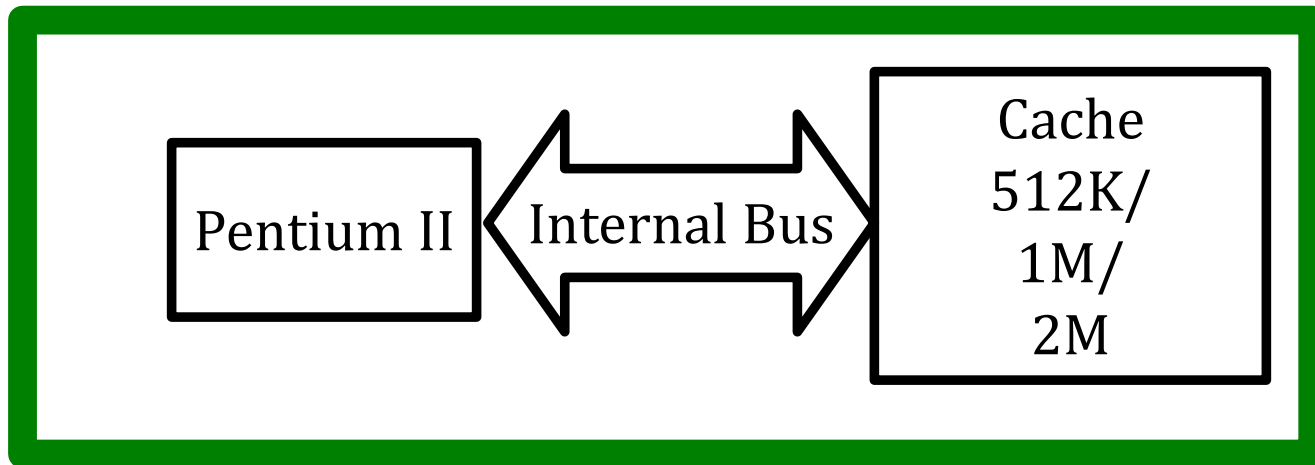
Pentium II

- Extension to Pro architecture with some differences
 - Internal cache in P II has been moved out of the chip
 - P II is not available as a single chip
 - Rather is available on a small plug-in circuit board along with level 2 (L 2) cache chip

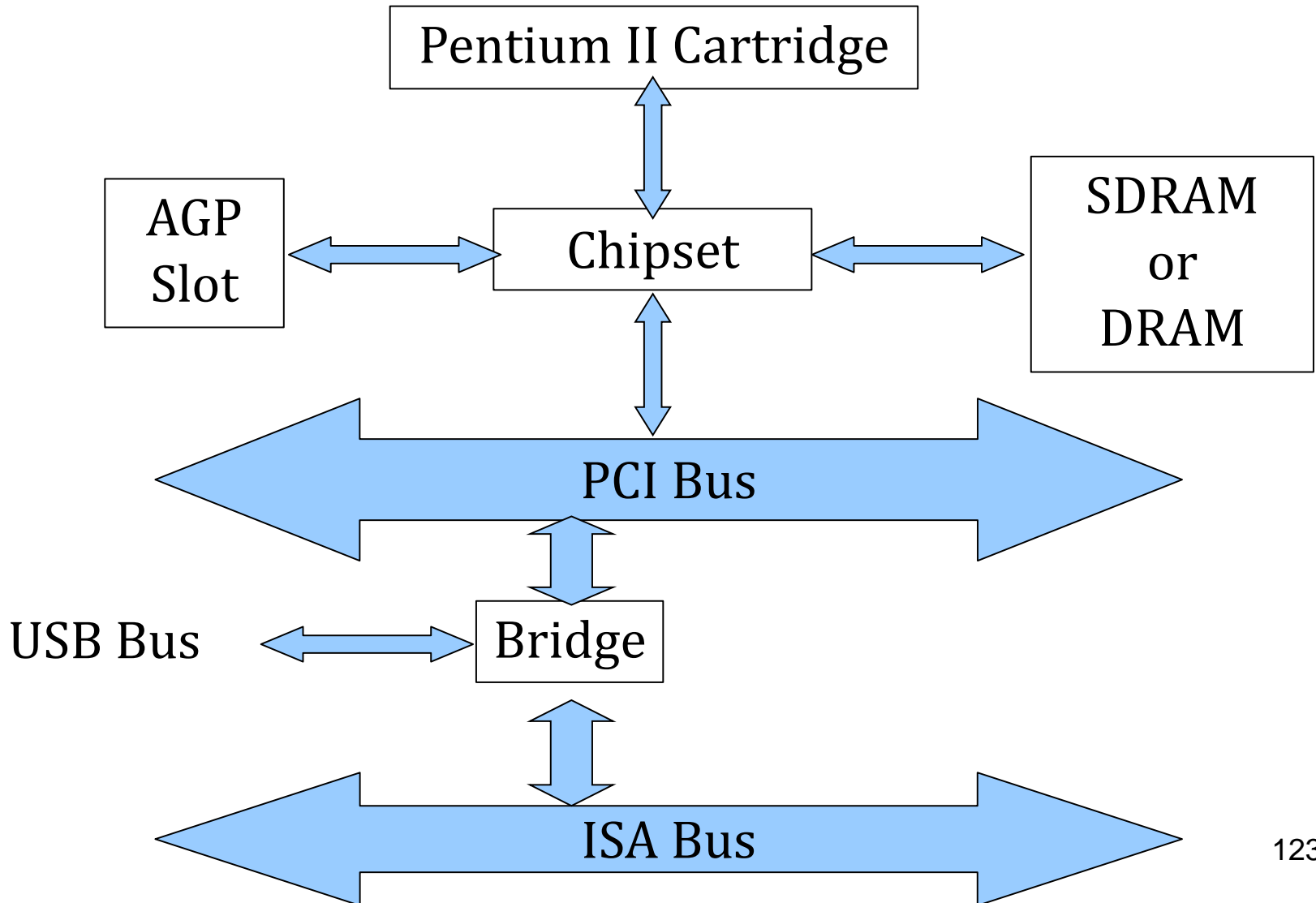
Pentium II (cont...)

Pentium II Internal Architecture

Pentium II Cartridge



A Typical Pentium II System



Pentium II (cont...)

- L2 cache is no longer inside the μ P IC
 - But placed very close to μ P IC
- This changes make the μ P less expensive

Pentium II (cont...)

- Various versions of P II are available
 - **Standard P II**
 - L2 cache operates at half the processor speed
 - **Celeron**: does not contain L2 cache in the cartridge
 - Rather it is in the main board
 - L2 operate at μ P speed
 - **Xeon**: contain up to 2M (512K/1M/2M) L2 cache operates at processor speed

Pentium II (cont...)

- Early P II requires
 - 5.0 V
 - 3.3 V and
 - variable voltage power supply for operation
 - may vary from 3.5v to as low as 1.8v
- Requires 8.4 to 14.2A depending on operating frequency

Pentium II (cont...)

Memory system

- 36 bit address
- 64 bit data
- RAM used has an access time of 8 ns to 10 ns
- Also include ECC
- Though not used by P II system, parity checking is available

Pentium II (cont...)

Memory system

- Transfers between PII and memory system are controlled by the chipset
- In fact, chipset controls PII, which is a departure from the traditional use of processor

Pentium II (cont...)

Memory map of a PII based system

- Conventional Memory 0 – 1M
 - Application Area 0 – 640K
 - System Area 640K – 1M
- Main Memory 1M – 1G
 - Optional ISA Memory 15M – 16M
 - Remapped AGP Data
- PCI Memory 1G – 4G
 - AGP Aperture Texture and Instructions
 - PCI Access to AGP Frame Buffer
 - PCI Access to AGP Registers
- For future expansion 4G – 64G

Pentium III

Pentium III

- Based on Pro architecture, not on P II
- Like P II, P III is packaged in a cartridge instead of IC chip
- But Coppermine is packaged in an IC with 370 pins
- Coppermine also contains an internal cache

Pentium III

- Improved version of PII, but based on Pro architecture, not on P II
- Two version of P III available
 1. packaged in a slot 1 cartridge instead of IC chip like PII with a non-blocking 512K cache running at half speed of processor
 2. Packaged in 370-pins IC, known as Coppermine, with 256K advanced transfer cache within the IC and running at processor speed
- It has been observed that, increasing cache size from 256K to 512K improves the performance by only a few percent

Pentium III (cont...)

- Chipset is different from P II
- Coppermine increases the bus speed to either 100MHz or 133MHz
- Bus speed cannot be increased arbitrarily due to radiation problem

Pentium III (cont...)

- Various versions of P III are also available like PII
 - Standard P III
 - Celeron PIII uses 66MHz bus speed
 - Xeon PIII allows larger cache for server applications

Pentium IV

Pentium IV

- Based on Pro architecture, not on PII or PIII
- Released initially in Nov'00 with 1.3GHz speed
- now available with speed more then 3GHz
- Available in two IC packages using 0.18 micron
 - 423-pin PGA
 - 478-pin FC-PGA2
- More recent versions use 0.13 or 0.09 micron technology
- It uses physically smaller transistors
 - Making it much smaller and faster than P III
- Uses 100MHz bus speed

Pentium IV (cont...)

- memory Interface
 - Typically uses Intel 850 chipset
 - 850 provides a dual-pipe memory bus with processor
- Each pipe interfaced to a 32-bit wide section of memory
- Two pipes functions together to comprise the 64-bit data bus

8255

8086

80286

80386

Pentium

8255

- Port A, B, C
- Mode of operation
- Configuration
 - Mode set control word
 - Bit set/reset control word

8086

- Internal architecture
 - Registers
 - General registers
 - Segment registers
 - Flags
 - Memory addressing
 - Addressing modes
 - Pins and signals
 - Min and max mode of operation
 - Interrupts
 - Dedicated/pre-defined

80286

- Internal architecture
 - Registers
 - General registers
 - Segment registers
 - Flags
 - Modes of operation
 - Real mode
 - Protected mode
 - Memory addressing
 - Selectors and Segment descriptors
 - How protection is ensured

80386

- Internal architecture
 - General registers
 - Segment registers
 - Flags
 - special registers
 - Control, debug, test regs
- Modes of operation
 - Real mode
 - Protected mode
 - Memory addressing
 - Selectors and Segment descriptors
 - Paging
 - Virtual 86 mode

Pentium

- Pentium
 - Memory system
 - Superscaler architecture
 - Branch Prediction
 - Cache structure
 - Memory management
 - Paging