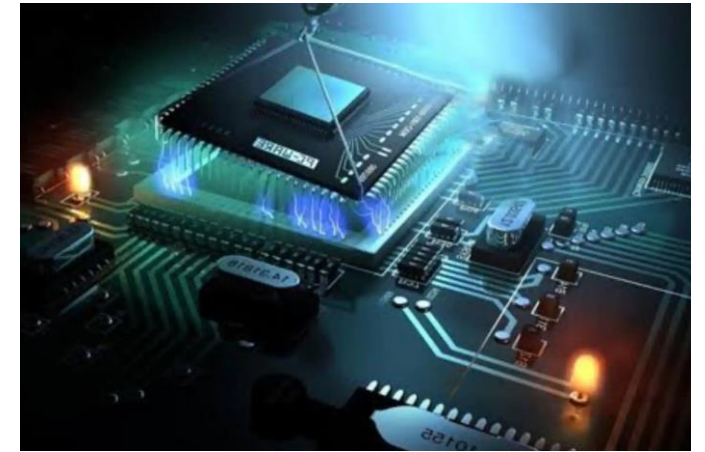
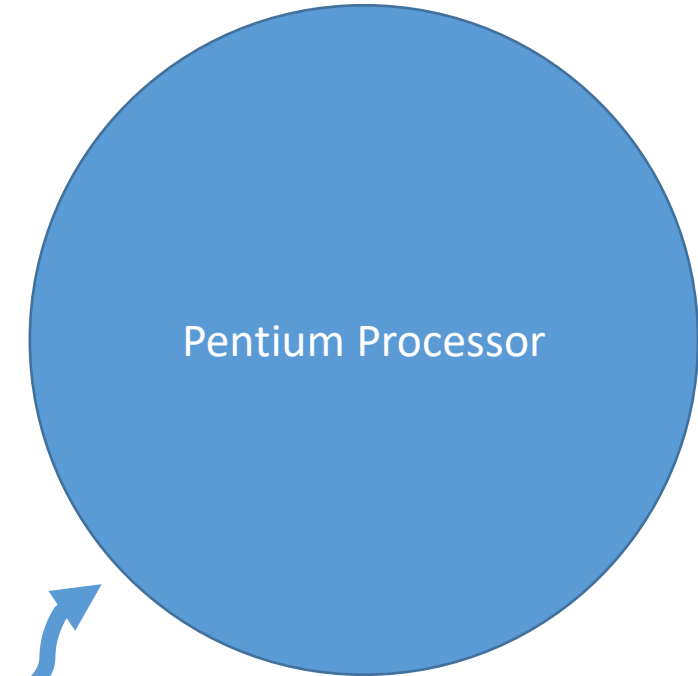
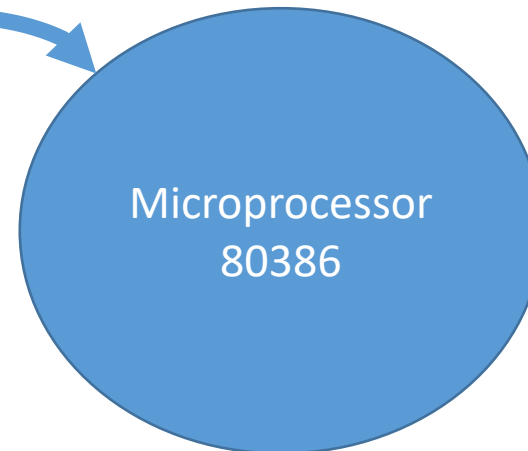
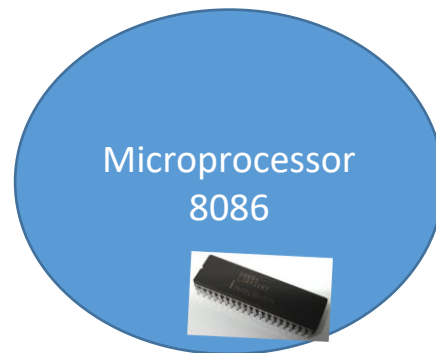
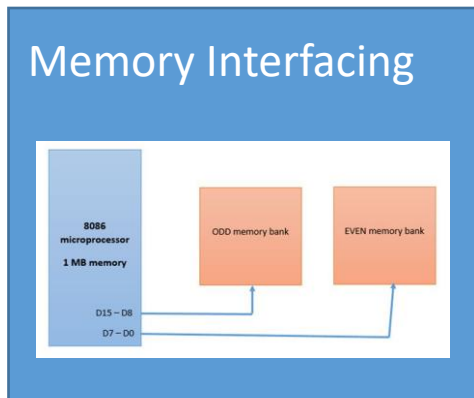


# MICROPROCESSOR

## MODULE 1



# Syllabus



# MODULE 1

## **The Intel Microprocessors 8086 Architecture**

- 1.1 8086CPU Architecture,
- 1.2 Programmer's Model
- 1.3 Functional Pin Diagram
- 1.4 Memory Segmentation
- 1.5 Banking in 8086
- 1.6 Demultiplexing of Address/Data bus
- 1.7 Functioning of 8086 in Minimum mode and Maximum mode
- 1.8 Timing diagrams for Read and Write operations in minimum and maximum mode
- 1.9 Interrupt structure and its servicing

# REVISION OF BASIC CONCEPTS

## ➤ BASIC COMPUTER SYSTEM

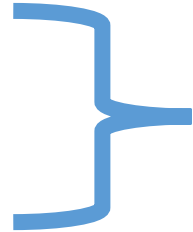
### ➤ MEMORY

### ➤ Instruction Cycle

➤ Fetch

➤ Decode

➤ Execute



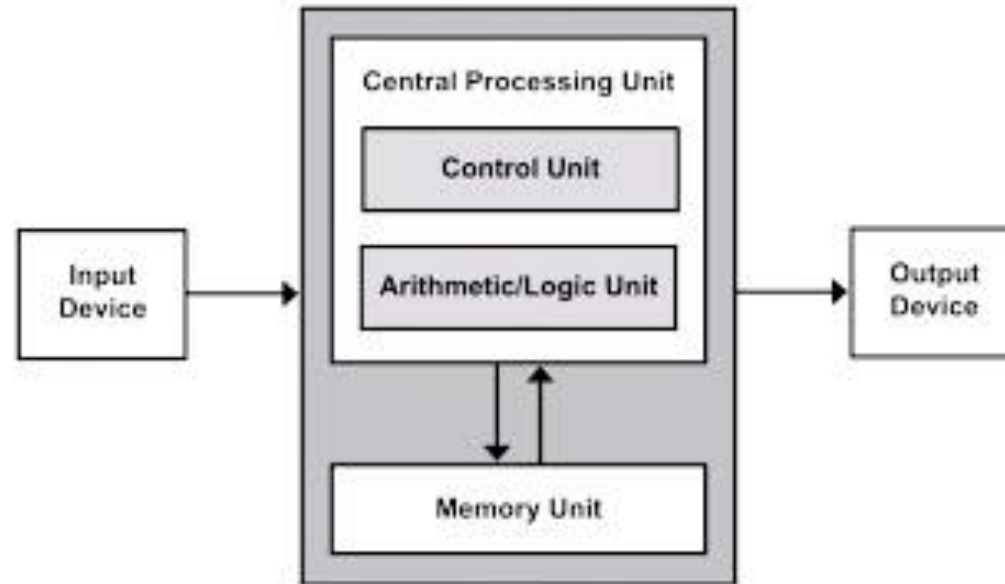
Concept understanding with respect to microprocessor

### ➤ Why Hexa-decimal Number system ?

### ➤ Power of 2

### ➤ System Bus

# BASIC COMPUTER SYSTEM



Block diagram of computer system

# History of Microprocessors

8085

8086

80186

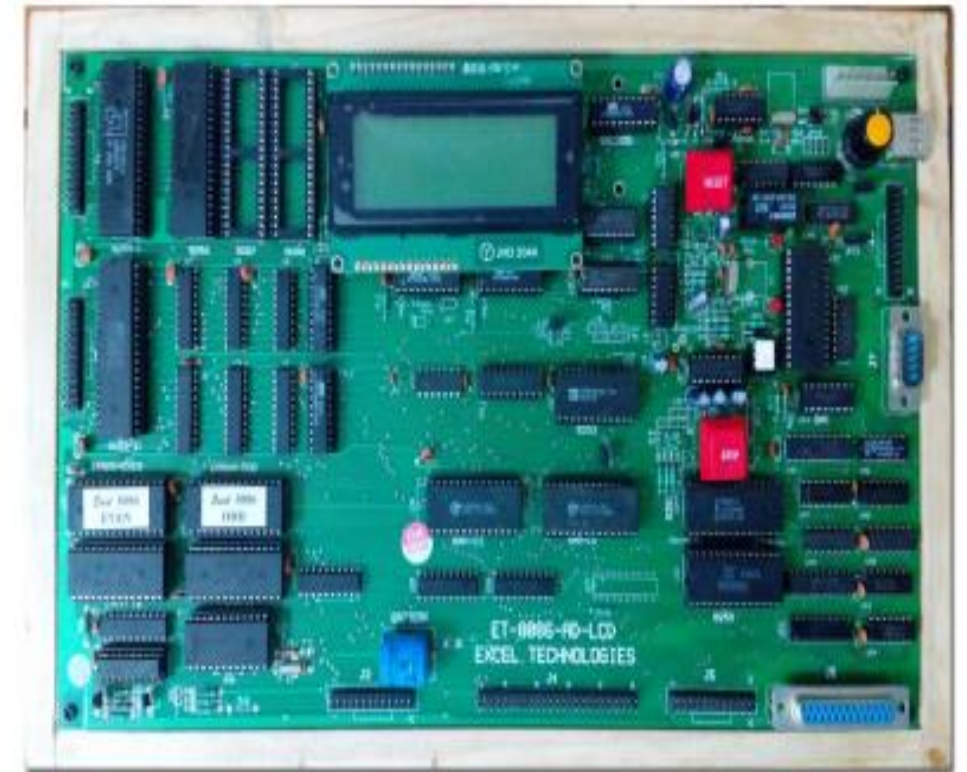
80286

80386

80486

80586 : P1,P2,P3..C2D,Ci7, Ci9

# HOW 8086 PROCESSOR LOOKS LIKE?



# INTRODUCTION TO 8086

- Enhanced version of 8085 designed by Intel in 1976
- It is 16 bit Microprocessor
- It has a 20 bit address bus so can access upto  $2^{20}$  memory locations
- 16 bit data lines
- 64k I/O Ports
- 14, 16 bit Registers
- It has powerful instruction set (multiplication & divisions)
- It supports two modes of operations:
  - Minimum Mode : suitable for single processor
  - Maximum Mode : suitable for multiple processors



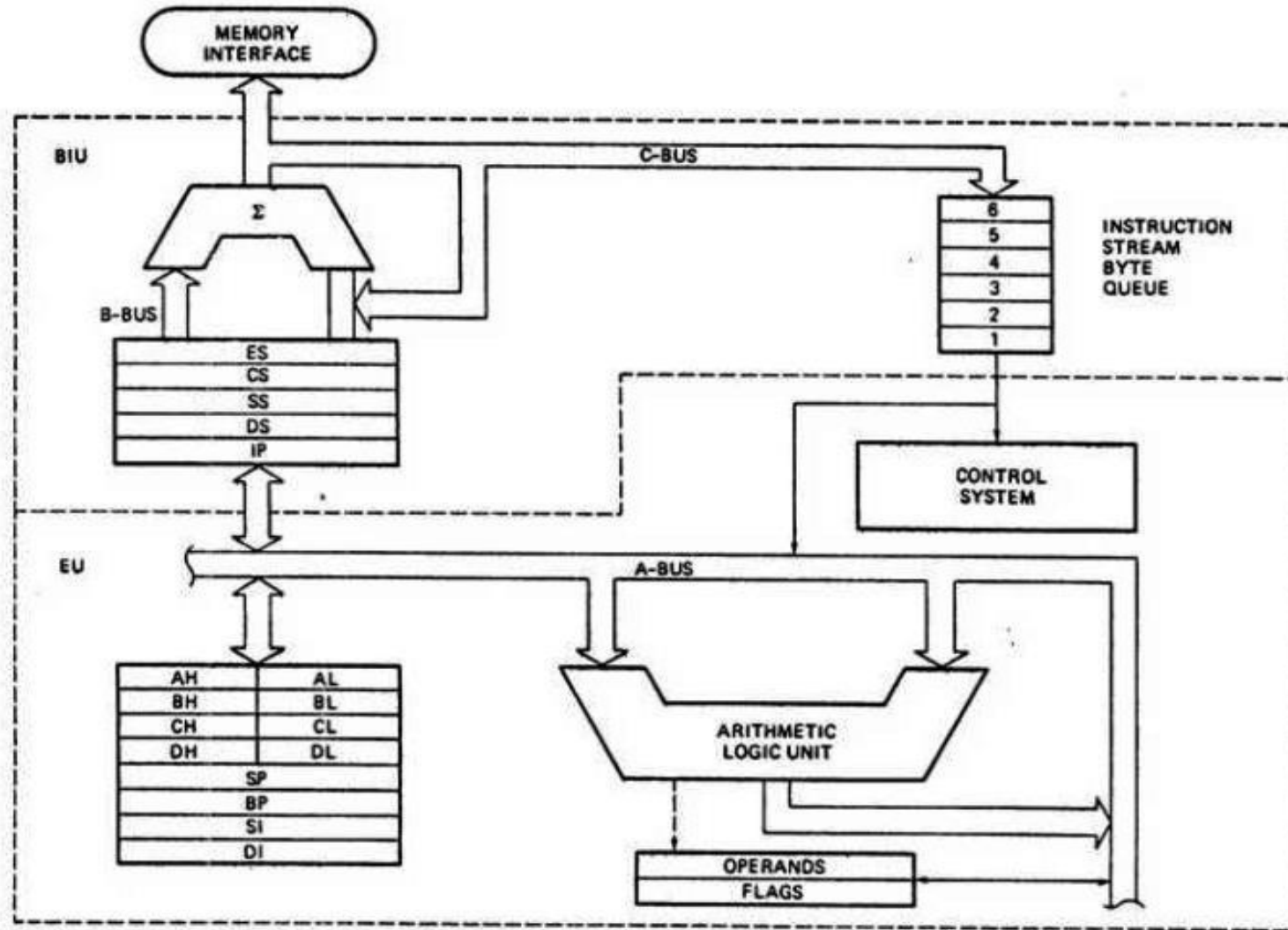
# FEATURES TO 8086

- It has instruction queue capable of storing 6 instructions bytes from memory resulting in faster processing.
- It was the first 16 bit processor having 16 bit ALU, 16 Bit Registers, internal data bus resulting in faster processing
- Available in 3 versions:
  - 5MHz
  - 8MHz
  - 10 MHz
- Two stages of pipelining : (Fetch +Execution)
- Single phase clock with 33% duty cycle
- 256 vectored interrupts.

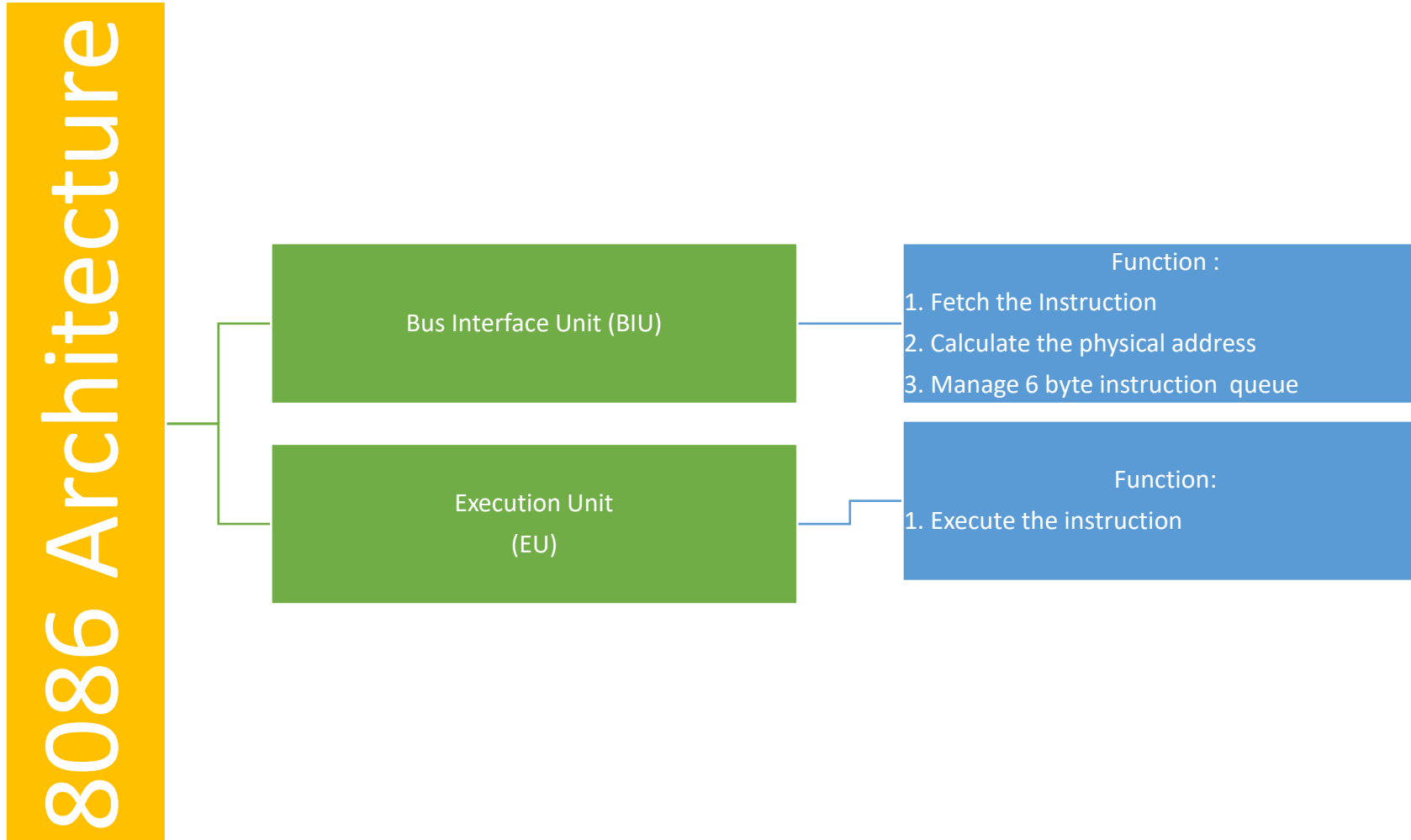
# COMPARISON WITH 8085

Parameter	8085	8086
size	8 bit	16 bit
Address Bus	16 bit	20 bit
Memory	64KB	1MB
Instruction	No Instruction Queue	Instruction Queue of 6 instructions
Pipelining	Does not support	Supports 2 stages of Pipelining
I/O	$2^8$ - 256 I/O's	$2^{16}$ - 65,356 I/O's
Cost	Low	High

# ARCHITECTURE OF 8086



# WHY 8086 HAS TWO PARTS?



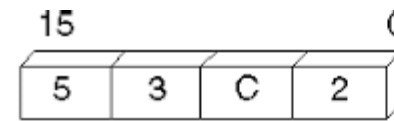
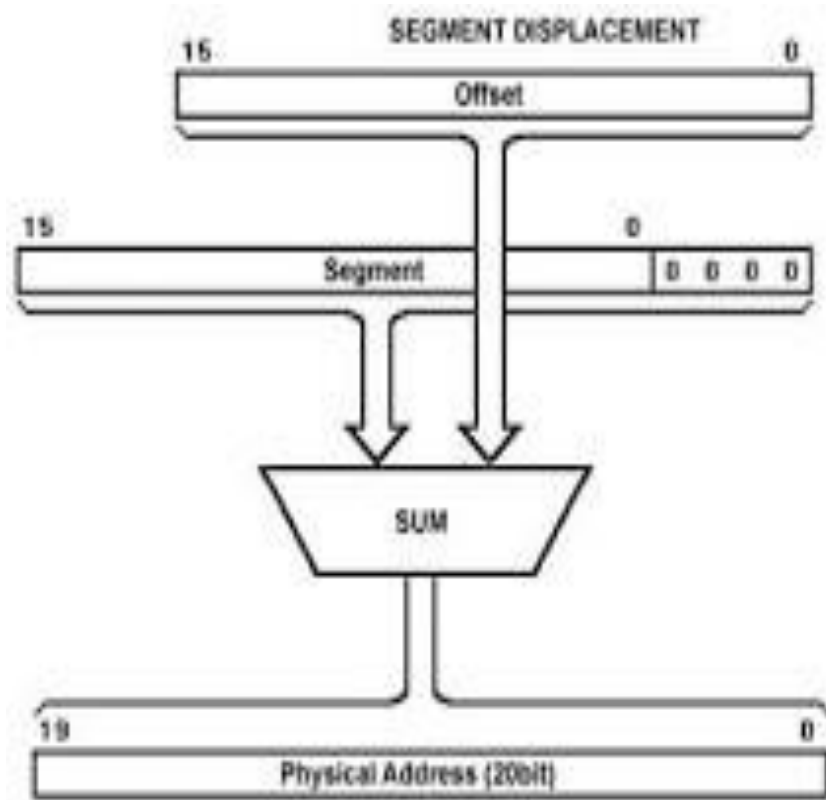
# BUS INTERFACE UNIT (BIU)

- It generates 20 bits of physical address for memory access.
- Fetch instruction from memory
- Transfer data to and from the memory and I/O
- Manages pipelining using 6 byte instruction queue

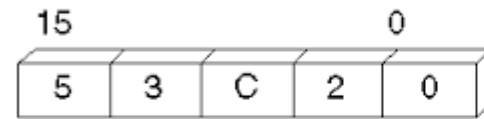
Main Components :

1. Segment Register
2. Address generation circuit
3. IP
4. Prefetch Queue

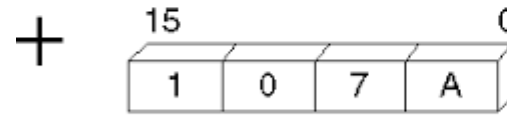
# PHYSICAL ADDRESS CALCULATION



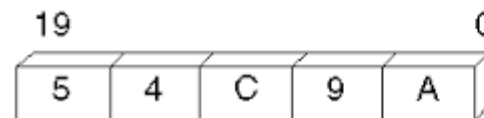
16-bit segment register...



...shifted left 4 bits



...plus 16-bit offset



...equals 20-bit physical address

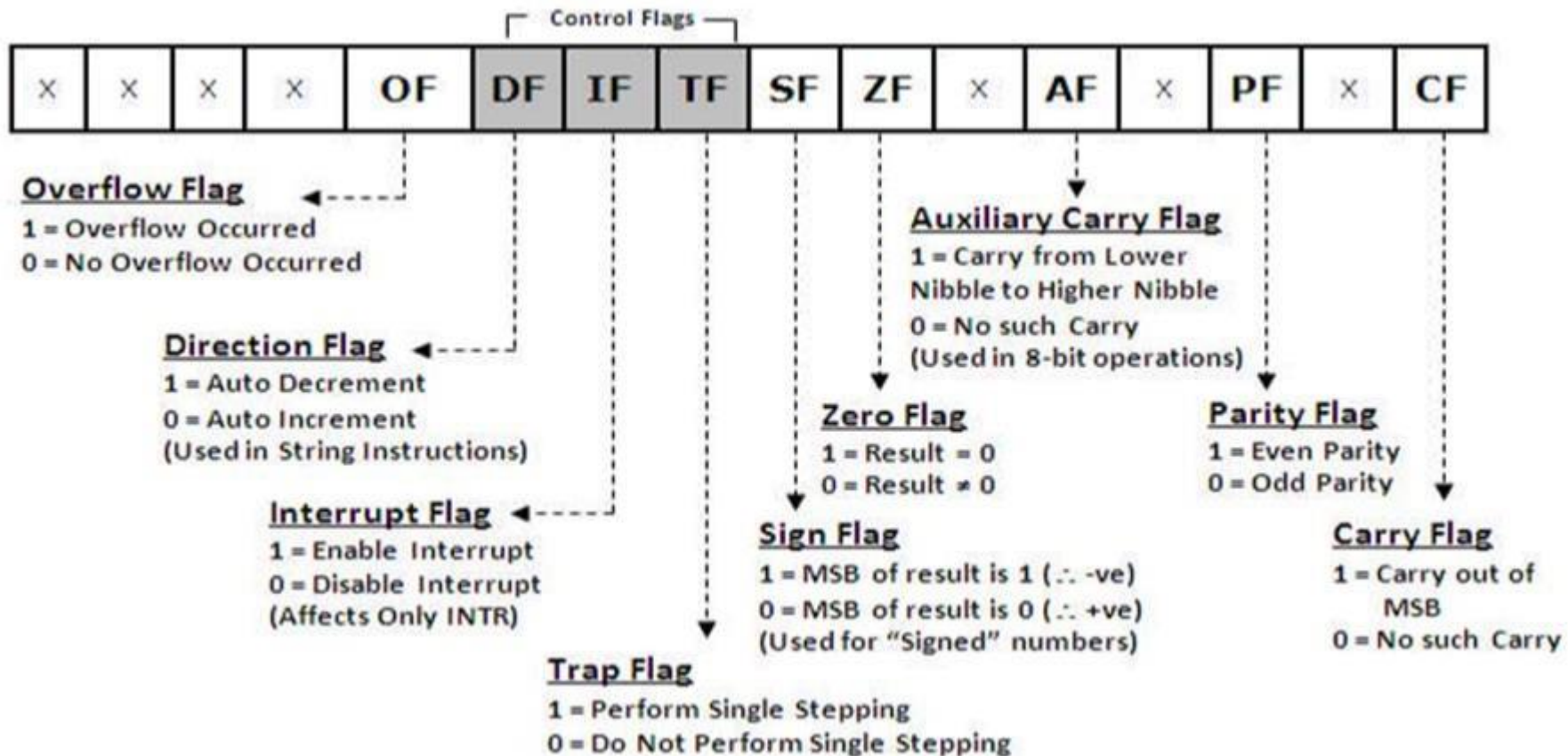
# EXECUTION UNIT (EU)

- Decode and execute instructions
- Perform arithmetic logic and internal data transfer operation
- Send request signal to BIU to access the external module
- It operates with respect to 'T states' (clock cycle)

Main components :

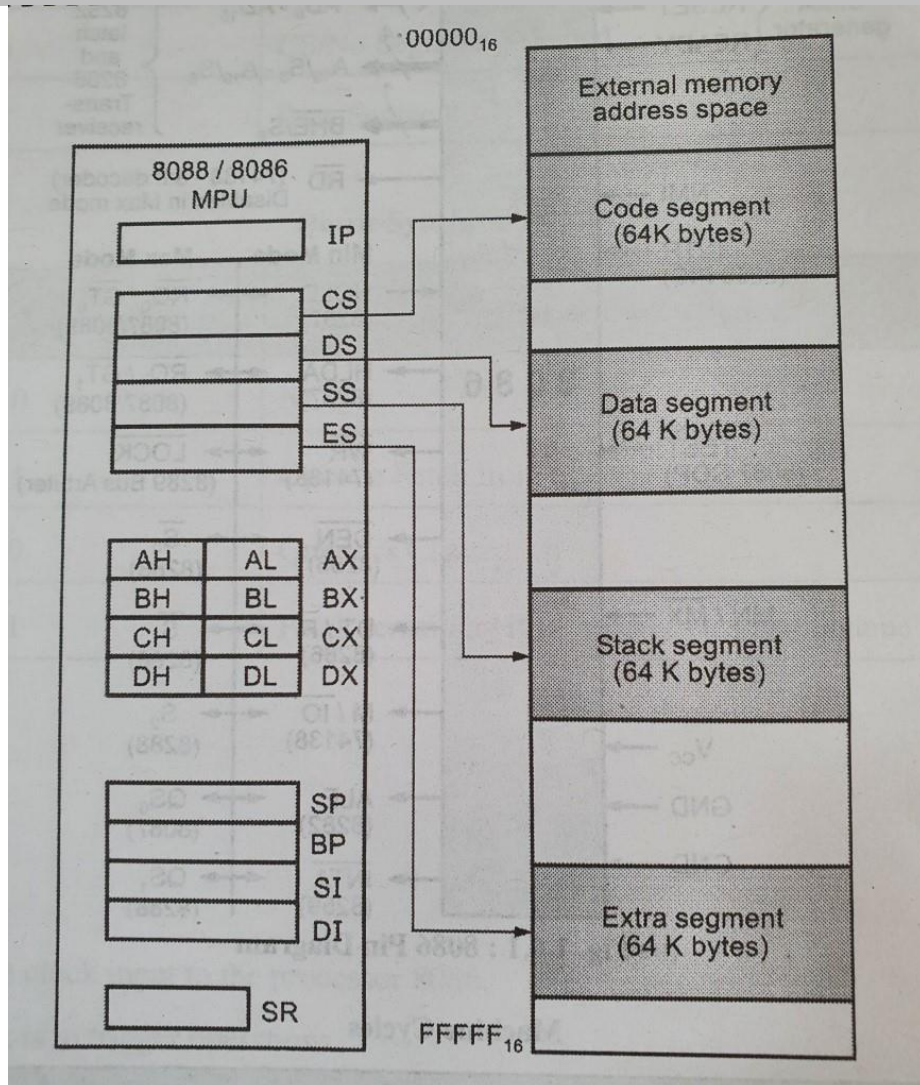
1. General purpose register
2. Special purpose register
3. ALU
4. Operand
5. Flag register

# FLAG REGISTER



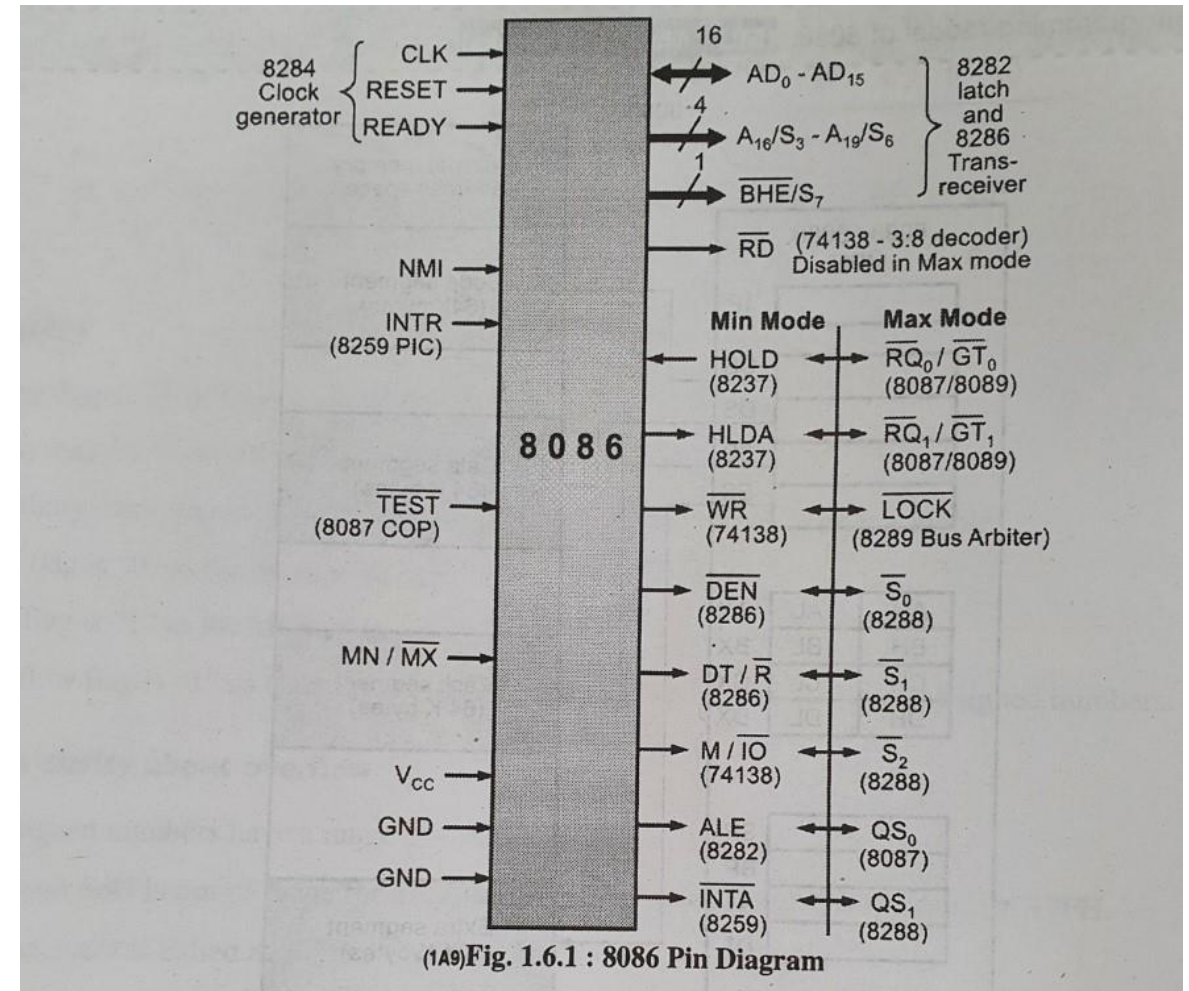
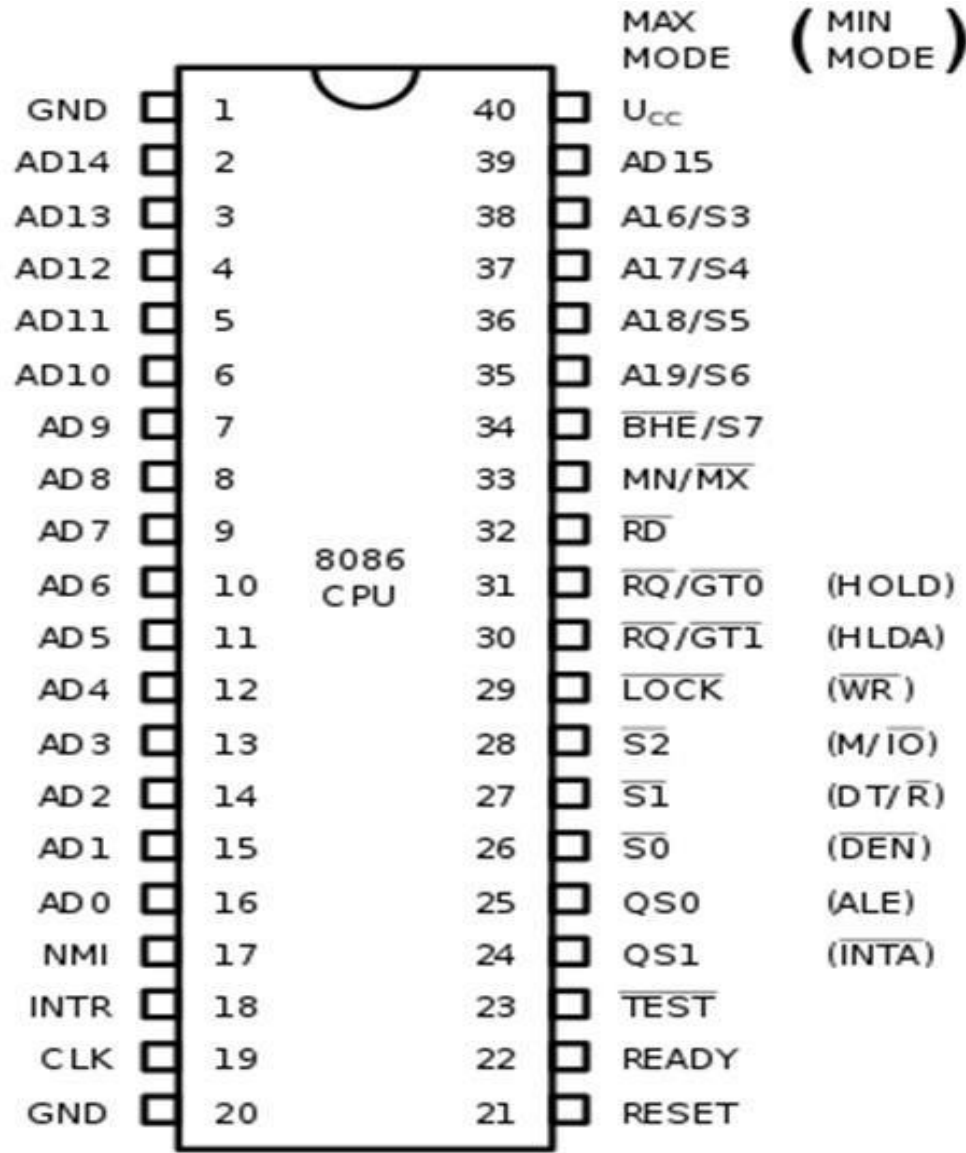


# PROGRAMMERS MODEL



- Software model
- All registers visible to the programmer
  - All GPR's
  - All Segment registers
  - All off set registers (special purpose reg)
  - Flag registers

# PIN DIAGRAM

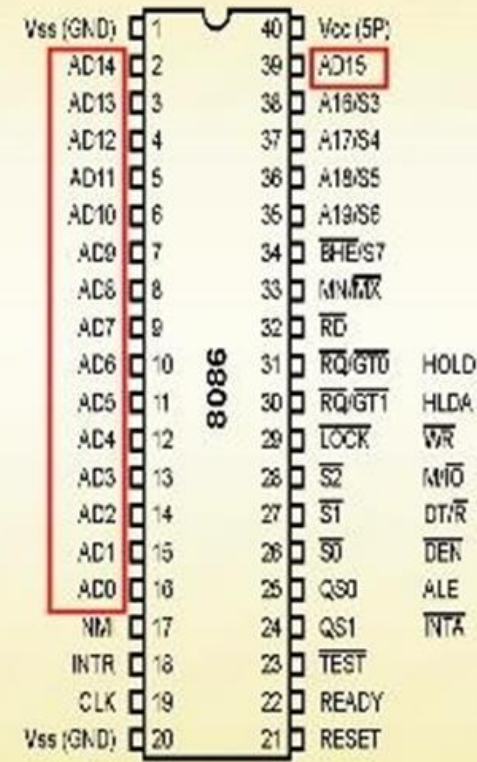


# PIN DIAGRAM

## $AD_0 - AD_{15}$

Pin 16-2, 39 (Bi-directional)

- These lines are multiplexed bi-directional address/data bus.
- During  $T_1$ , they carry lower order 16-bit address.
- In the remaining clock cycles, they carry 16-bit data.
- $AD_0 - AD_7$  carry lower order byte of data.
- $AD_8 - AD_{15}$  carry higher order byte of data.



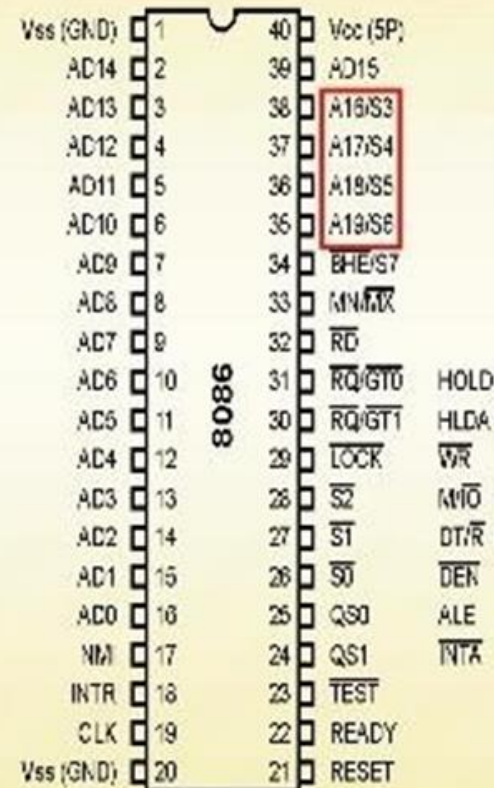


# PIN DIAGRAM

$A_{19}/S_6, A_{18}/S_5, A_{17}/S_4, A_{16}/S_3$

Pin 35-38 (Unidirectional)

- These lines are multiplexed unidirectional address and status bus.
- During  $T_1$ , they carry higher order 4-bit address.
- In the remaining clock cycles, they carry status signals.



# PIN DIAGRAM

S4	S3	FUNCTION
0	0	Extra segment access
0	1	Stack segment access
1	0	Code segment access
1	1	Data segment access

**S5** : Carries the status of 'IF' flag

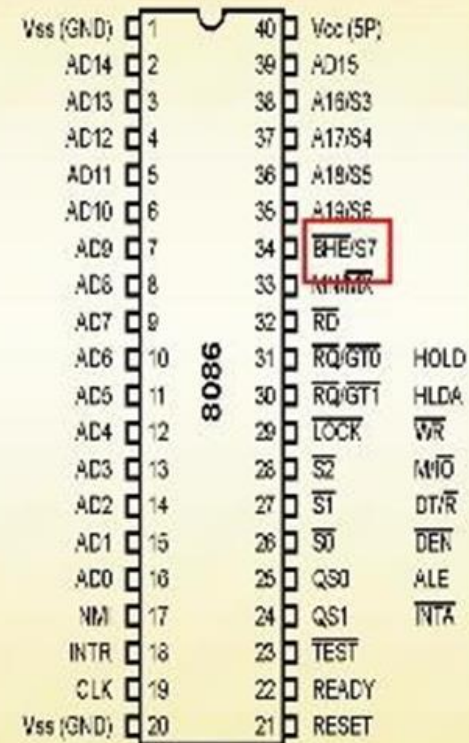
**S6** : Indicates Whether 8086 is in-charge or other processor in the configuration is in-charge of the system bus. Low when 8086 is in charge of the system bus

# PIN DIAGRAM

**$\overline{\text{BHE}} / \text{S}_7$**

Pin 34 (Output)

- BHE stands for Bus High Enable.
- BHE signal is used to indicate the transfer of data over higher order data bus ( $D_8 - D_{15}$ ).
- 8-bit I/O devices use this signal.
- It is multiplexed with status pin  $\text{S}_7$ .

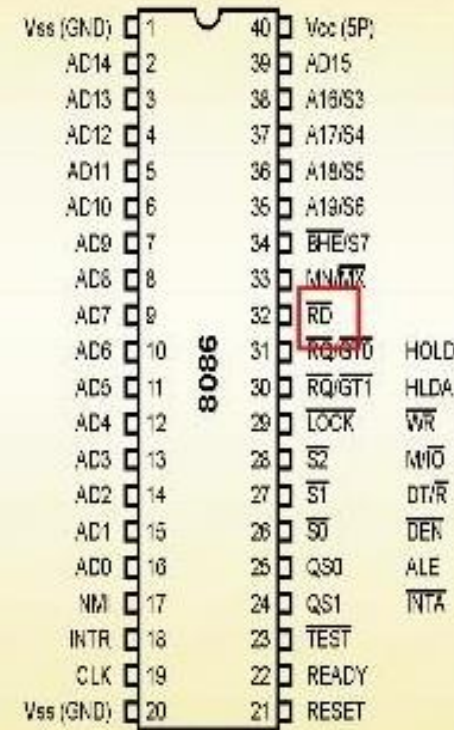


# PIN DIAGRAM

## $\overline{RD}$ (Read)

Pin 32 (Output)

- It is a read signal used for read operation.
- It is an output signal.
- It is an active low signal.



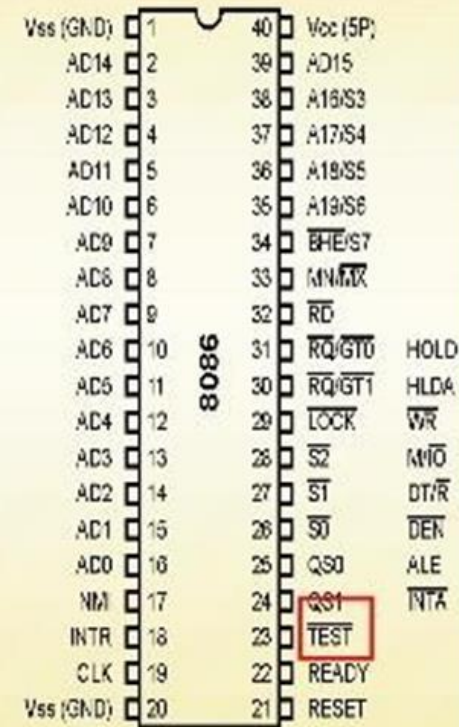


# PIN DIAGRAM

## TEST

Pin 23 (Input)

- It is used to test the status of math co-processor 8087.
- The BUSY pin of 8087 is connected to this pin of 8086.
- If low, execution continues else microprocessor is in wait state.



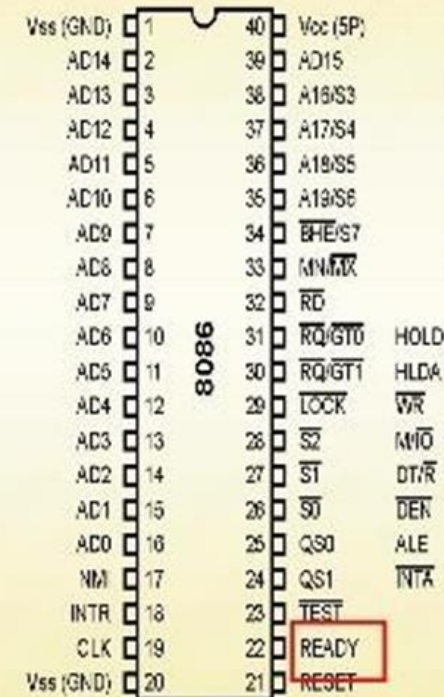


# PIN DIAGRAM

## READY

Pin 22 (Input)

- This is an acknowledgement signal from slower I/O devices or memory.
- It is an active high signal.
- When high, it indicates that the device is ready to transfer data.
- When low, then microprocessor is in wait state.

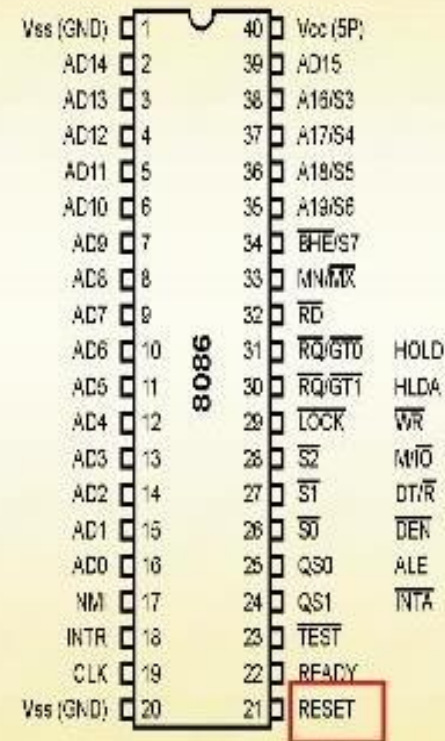


# PIN DIAGRAM

## RESET

Pin 21 (Input)

- It is a system reset.
- It is an active high signal.
- When high, microprocessor enters into reset state and terminates the current activity.
- It must be active for at least four clock cycles to reset the microprocessor.

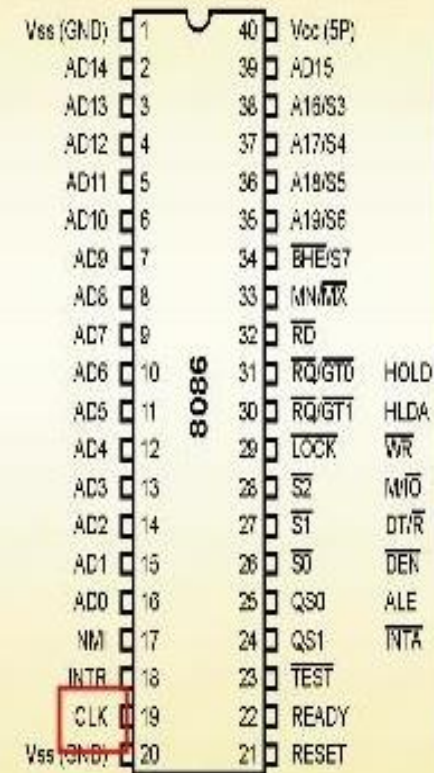


# PIN DIAGRAM

## CLK

Pin 19 (Input)

- This clock input provides the basic timing for processor operation.
- It is symmetric square wave with 33% duty cycle.
- The range of frequency of different versions is 5 MHz, 8 MHz and 10 MHz.



# PIN DIAGRAM

## INTR

Pin 18 (Input)

- It is an interrupt request signal.
- It is active high.
- It is level triggered.



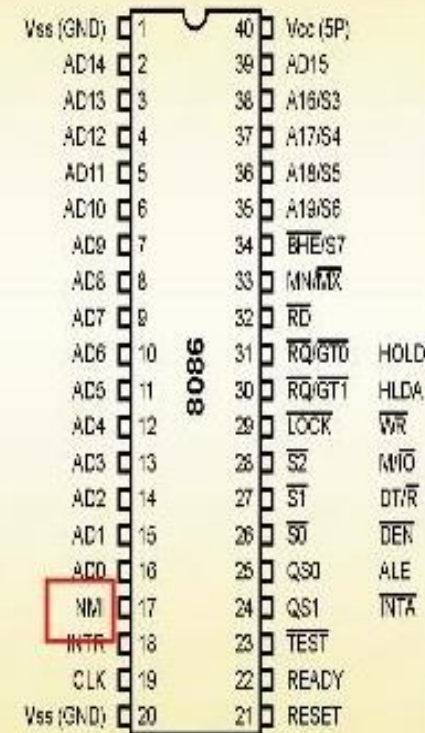


# PIN DIAGRAM

## NMI

Pin 17 (Input)

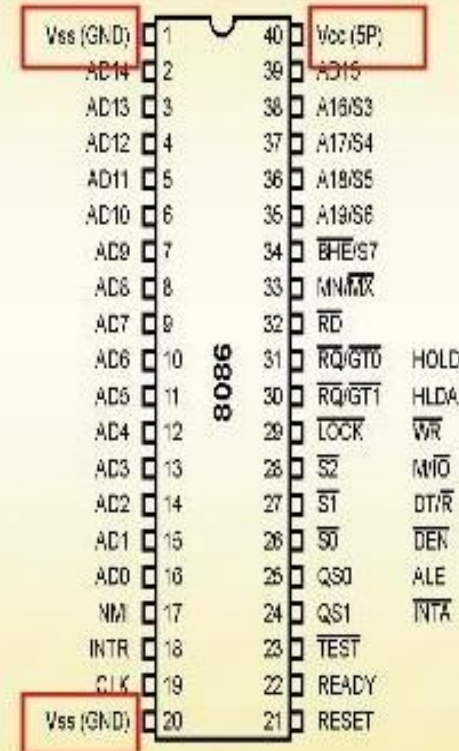
- It is a non-maskable interrupt signal.
- It is an active high.
- It is an edge triggered interrupt.



# PIN DIAGRAM

## $V_{CC}$ and $V_{SS}$ Pin 40 and Pin 20 (Input)

- $V_{CC}$  is power supply signal.
- +5V DC is supplied through this pin.
- $V_{SS}$  is ground signal.

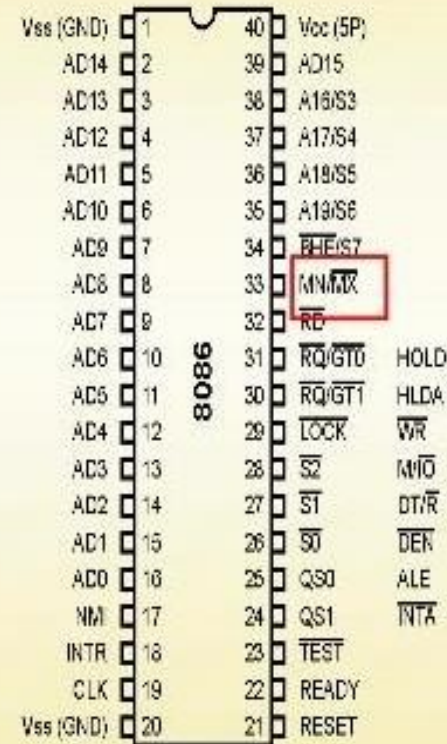


# PIN DIAGRAM

**MN /  $\overline{\text{MX}}$**

Pin 33 (Input)

- 8086 works in two modes:
  - Minimum Mode
  - Maximum Mode
- If  $\text{MN}/\overline{\text{MX}}$  is high, it works in minimum mode.
- If  $\text{MN}/\overline{\text{MX}}$  is low, it works in maximum mode.



# PIN DIAGRAM

## **Pin Description for Minimum Mode**



# PIN DIAGRAM

## HOLD

Pin 31 (Input)

- When DMA controller needs to use address/data bus, it sends a request to the CPU through this pin.
- It is an active high signal.
- When microprocessor receives HOLD signal, it issues HLDA signal to the DMA controller.

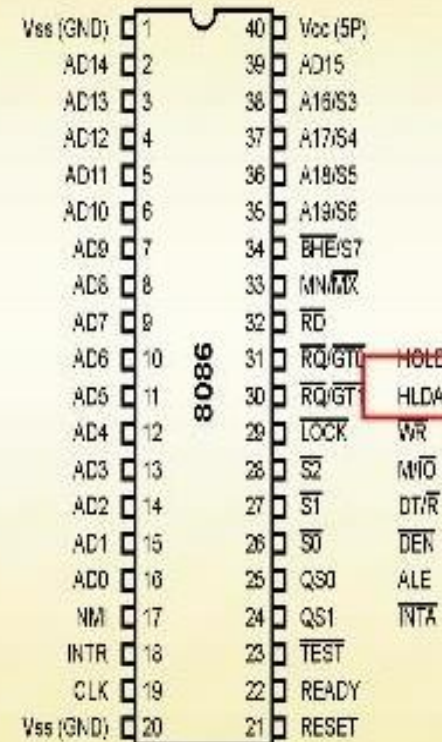


# PIN DIAGRAM

## HLDA

Pin 30 (Output)

- It is a Hold Acknowledge signal.
- It is issued after receiving the HOLD signal.
- It is an active high signal.

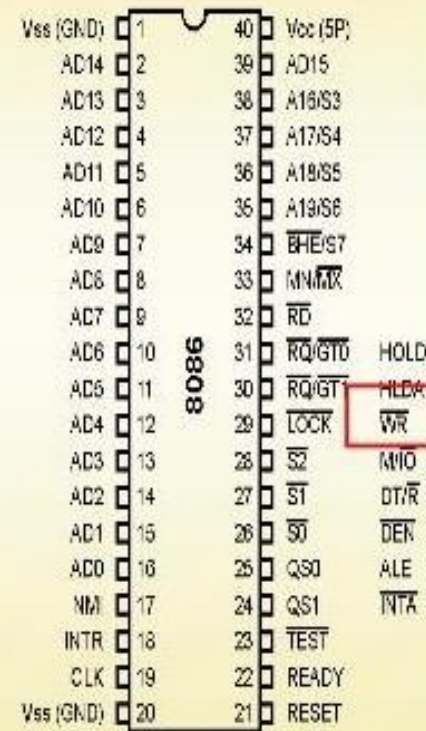


# PIN DIAGRAM

**$\overline{WR}$**

Pin 29 (Output)

- It is a Write signal.
- It is used to write data in memory or output device depending on the status of M/IO signal.
- It is an active low signal.

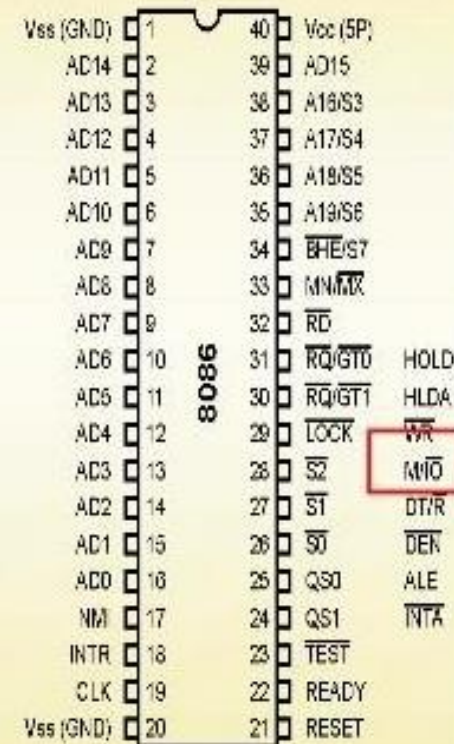


# PIN DIAGRAM

**M /  $\overline{\text{IO}}$**

**Pin 28 (Output)**

- This signal is issued by the microprocessor to distinguish memory access from I/O access.
- When it is high, memory is accessed.
- When it is low, I/O devices are accessed.



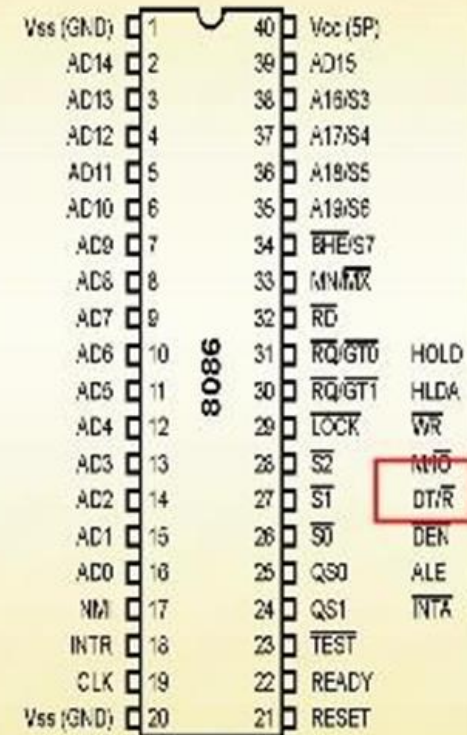


# PIN DIAGRAM

## DT / $\bar{R}$

Pin 27 (Output)

- This is a Data Transmit/Receive signal.
- It decides the direction of data flow through the transceiver.
- When it is high, data is transmitted out.
- When it is low, data is received in.

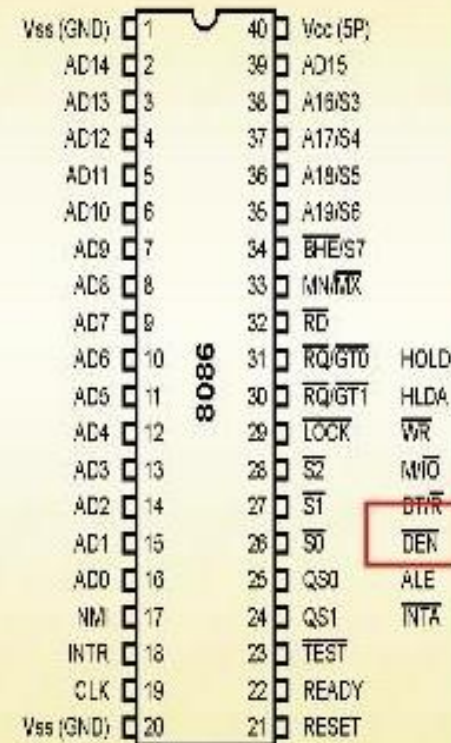


# PIN DIAGRAM

## DEN

Pin 26 (Output)

- This is a Data Enable signal.
- This signal is used to enable the transceiver 8286.
- Transceiver is used to separate the data from the address/data bus.
- It is an active low signal.

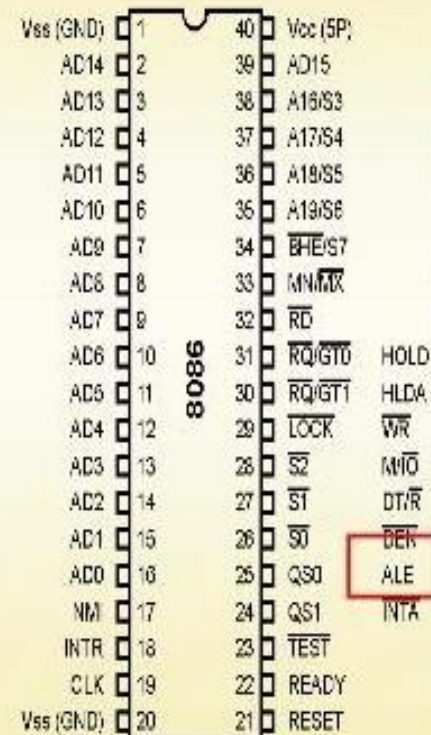


# PIN DIAGRAM

## ALE

Pin 25 (Output)

- This is an Address Latch Enable signal.
- It indicates that valid address is available on bus  $AD_0 - AD_{15}$ .
- It is an active high signal and remains high during  $T_1$  state.
- It is connected to enable pin of latch 8282.

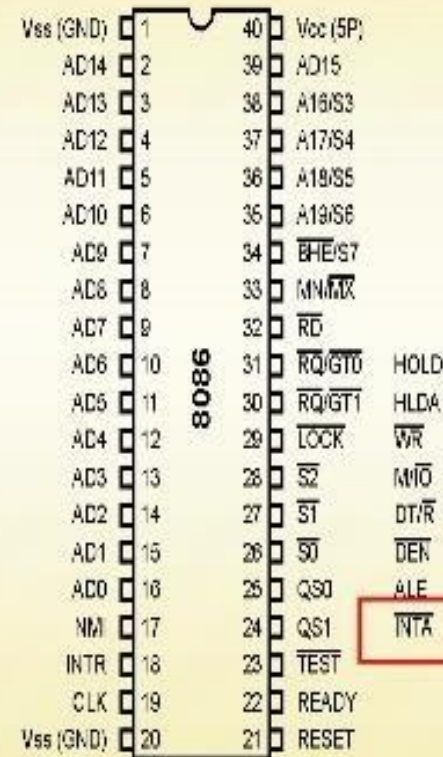


# PIN DIAGRAM

## $\overline{\text{INTA}}$

Pin 24 (Output)

- This is an interrupt acknowledge signal.
- When microprocessor receives INTR signal, it acknowledges the interrupt by generating this signal.
- It is an active low signal.





# PIN DIAGRAM

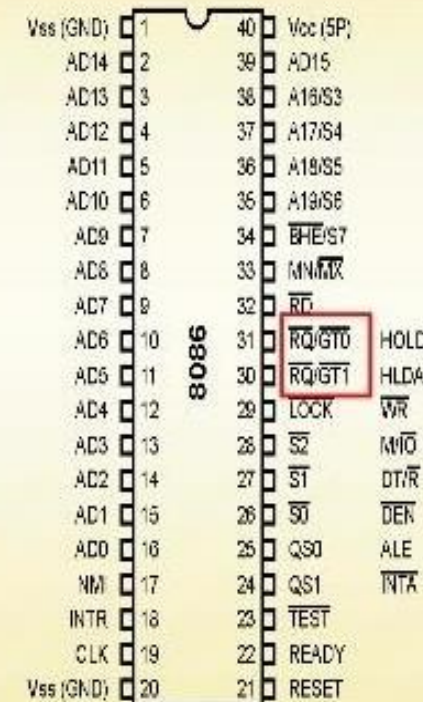
## **Pin Description for Maximum Mode**

# PIN DIAGRAM

## $\overline{RQ/GT_1}$ and $\overline{RQ/GT_0}$

Pin 30 and 31 (Bi-directional)

- These are Request/Grant pins.
- Other processors request the CPU through these lines to release the system bus.
- After receiving the request, CPU sends acknowledge signal on the same lines.
- $\overline{RQ/GT_0}$  has higher priority than  $\overline{RQ/GT_1}$ .

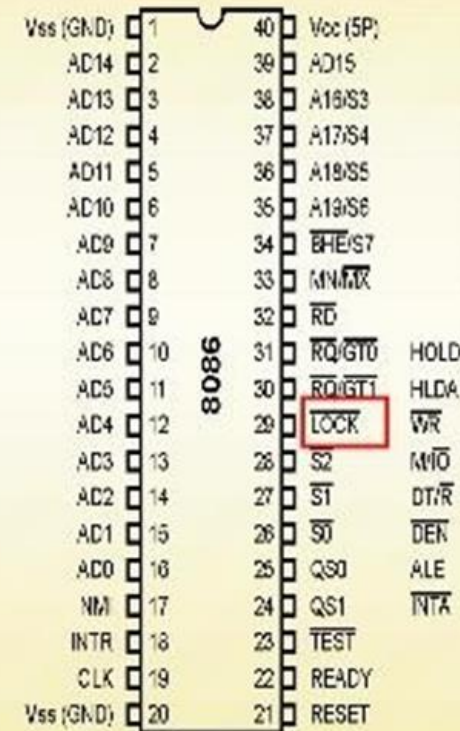


# PIN DIAGRAM

## LOCK

Pin 29 (Output)

- This signal indicates that other processors should not ask CPU to relinquish the system bus.
- When it goes low, all interrupts are masked and HOLD request is not granted.
- This pin is activated by using LOCK prefix on any instruction.

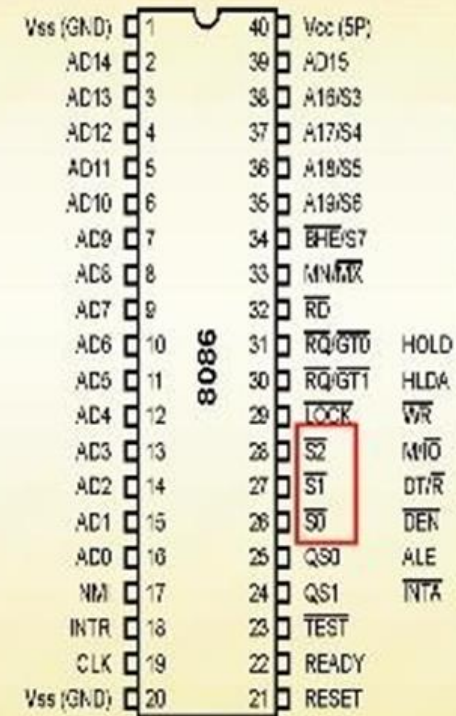


# PIN DIAGRAM

$\overline{S_0}, \overline{S_1}, \overline{S_2}$

Pin 26, 27, 28 (Output)

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Status
0	0	0	Interrupt Acknowledge
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	Halt
1	0	0	Opcode Fetch
1	0	1	Memory Read
1	1	0	Memory Write
1	1	1	Passive





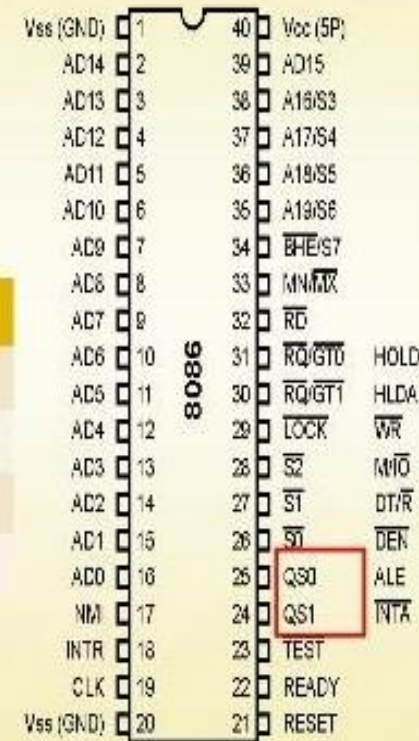
# PIN DIAGRAM

## $QS_1$ and $QS_0$

Pin 24 and 25 (Output)

- These pins provide the status of instruction queue.

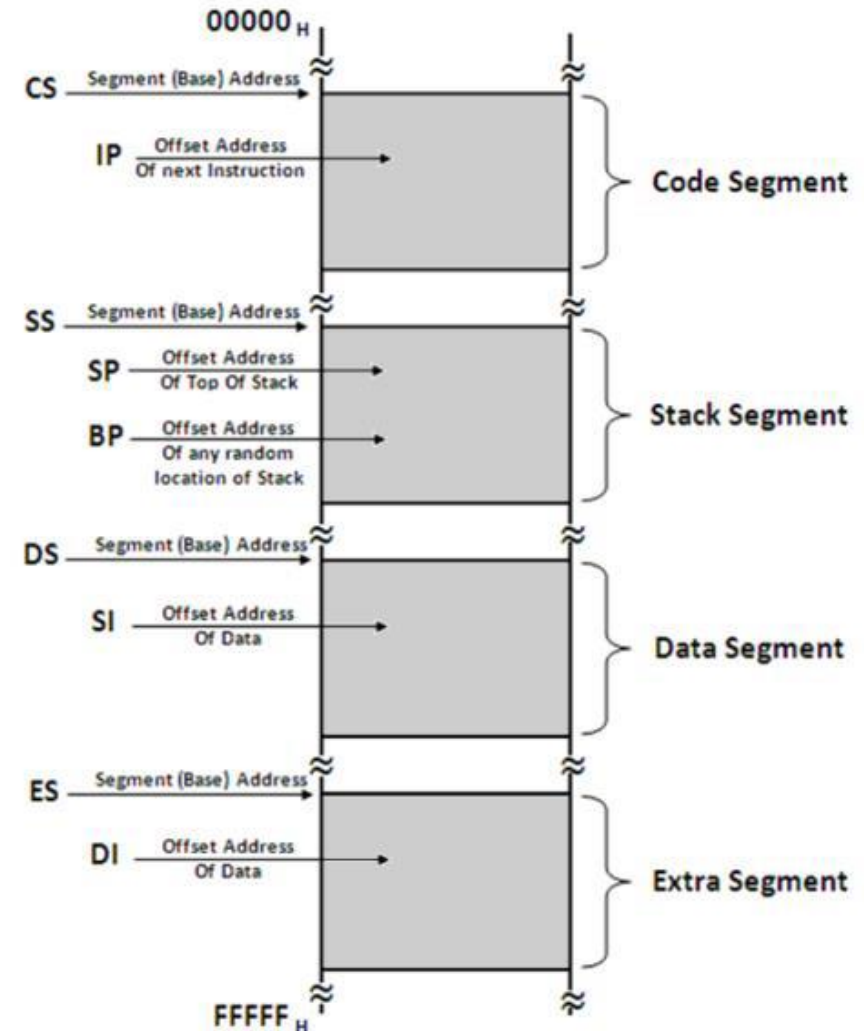
$QS_1$	$QS_0$	Status
0	0	No operation
0	1	1 <sup>st</sup> byte of opcode from queue
1	0	Empty queue
1	1	Subsequent byte from queue



# MEMORY SEGMENTATION

- **What is Segmentation ?**

**Segmentation** is the process in which the main memory of the computer is divided into different segments/partitions and each segment/partition has its own base address.



# MEMORY SEGMENTATION

## Advantages of Segmentation

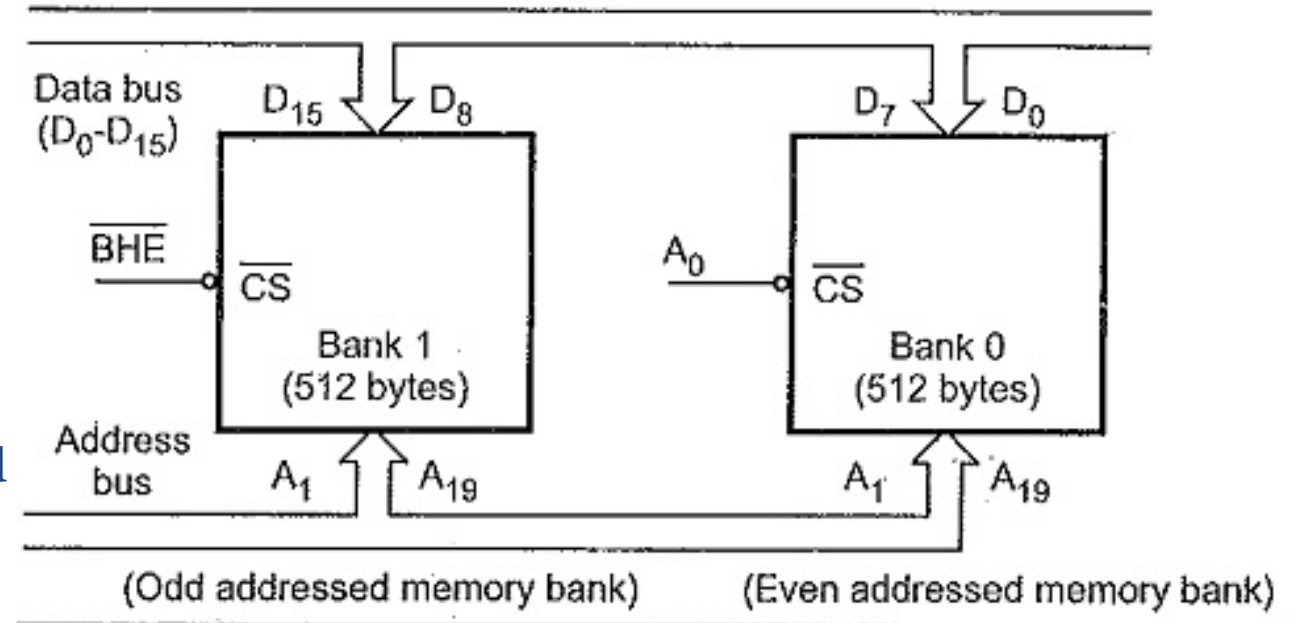
- With the help of memory segmentation we are able to work with registers having only 16-bits.
- The data and the code can be stored separately allowing for more flexibility.
- Also due to segmentation the logical address range is from 0000H to FFFFH, the code can be loaded at any location in the memory.



# MEMORY BANKING

## What is Memory Banking ?

- The memory address space is equally divided into two parts(banks).
- One of the banks contain even addresses called **Even bank** and the other contain odd addresses called **Odd bank**.
- Even bank always stores lower byte so Even bank is also called **Lower bank**(LB) and Odd bank is also called a **Higher bank**(HB).



# MEMORY BANKING

- **Why Memory Banking is Done in 8086 ?**

**8086** has 20-bit addressing model for **memory** access. Each address represents a single byte - however, the natural word size of **8086** is 2 bytes (16-bit Data Bus), so you need a way to read two bytes at the same time - hence, two **banks**. One **bank** has all the odd bytes, the other all the even bytes in a word.

# MEMORY BANKING

$\overline{\text{BHE}}$	A0	Operation
0	0	Read or Write 16 bit from both banks
0	1	Read or Write 8 bits from higher bank
1	0	Read or Write 8 bits from lower bank
1	1	No Operation (Processor is Idle)

# MEMORY BANKING

Examples : 8bit operation on even bank

Eg. MOV BL, [2000H]

MOV BH, [2001H]

MOV BX, [2000H] (aligned operation)

MOV BX, [20001H] (misaligned operation)

# 8282-8BIT(OCTAL) LATCH

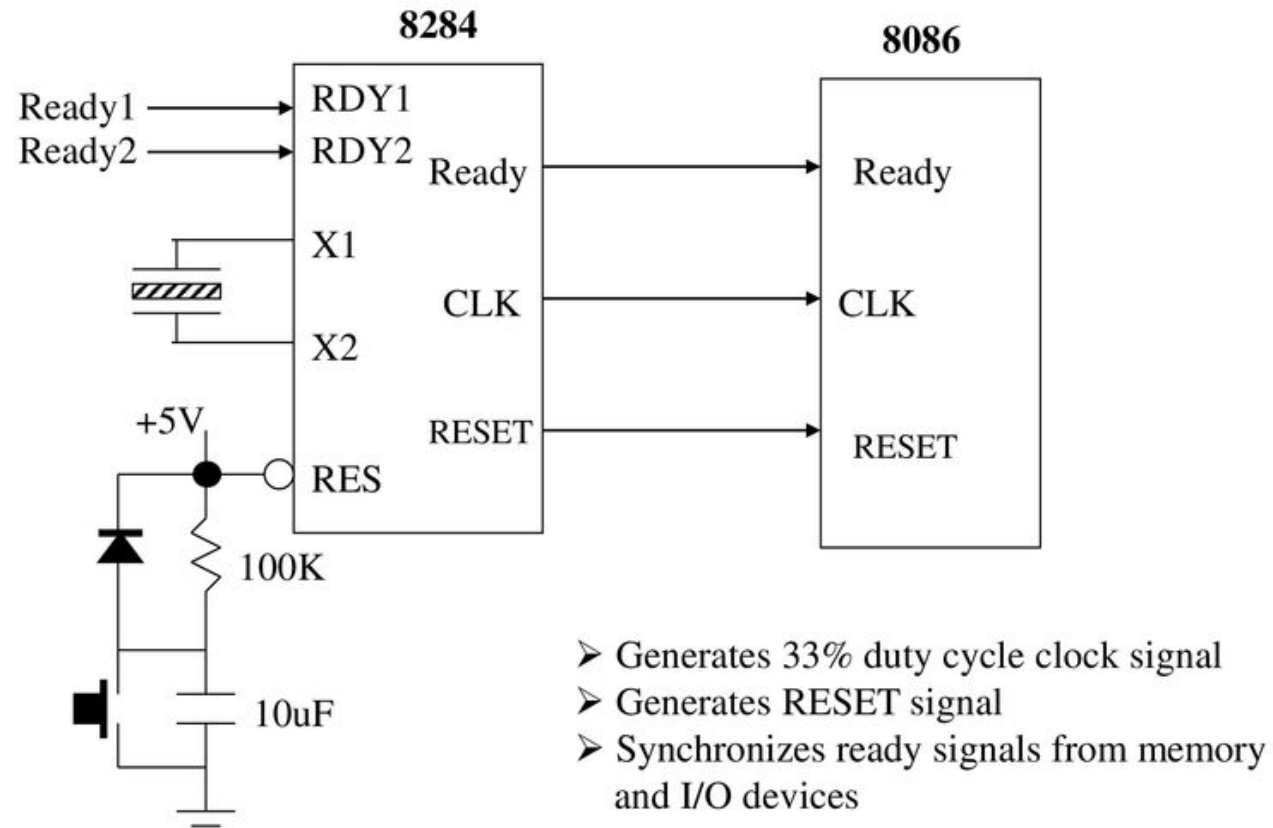
1. 8282 is an 8-bit latch.
2. In 8086, the address bus is multiplexed with the data bus and status bits.
3. 8282 is used to latch the address from this bus.
4. The ALE signal is connected to STB of 8282.
5. When STB (ALE) is high, the input is latched and transferred to the output. Hence address is latched.
6. When STB is low, the input is discarded. Hence, data is not latched. The previously latched address remains at the output.
7. As totally 21 bits are to be latched (A19-A0, and BHE ), 3 latches are required, each latch being 8 bit.

# 8286- 8BIT DATA TRANS-RECEIVER

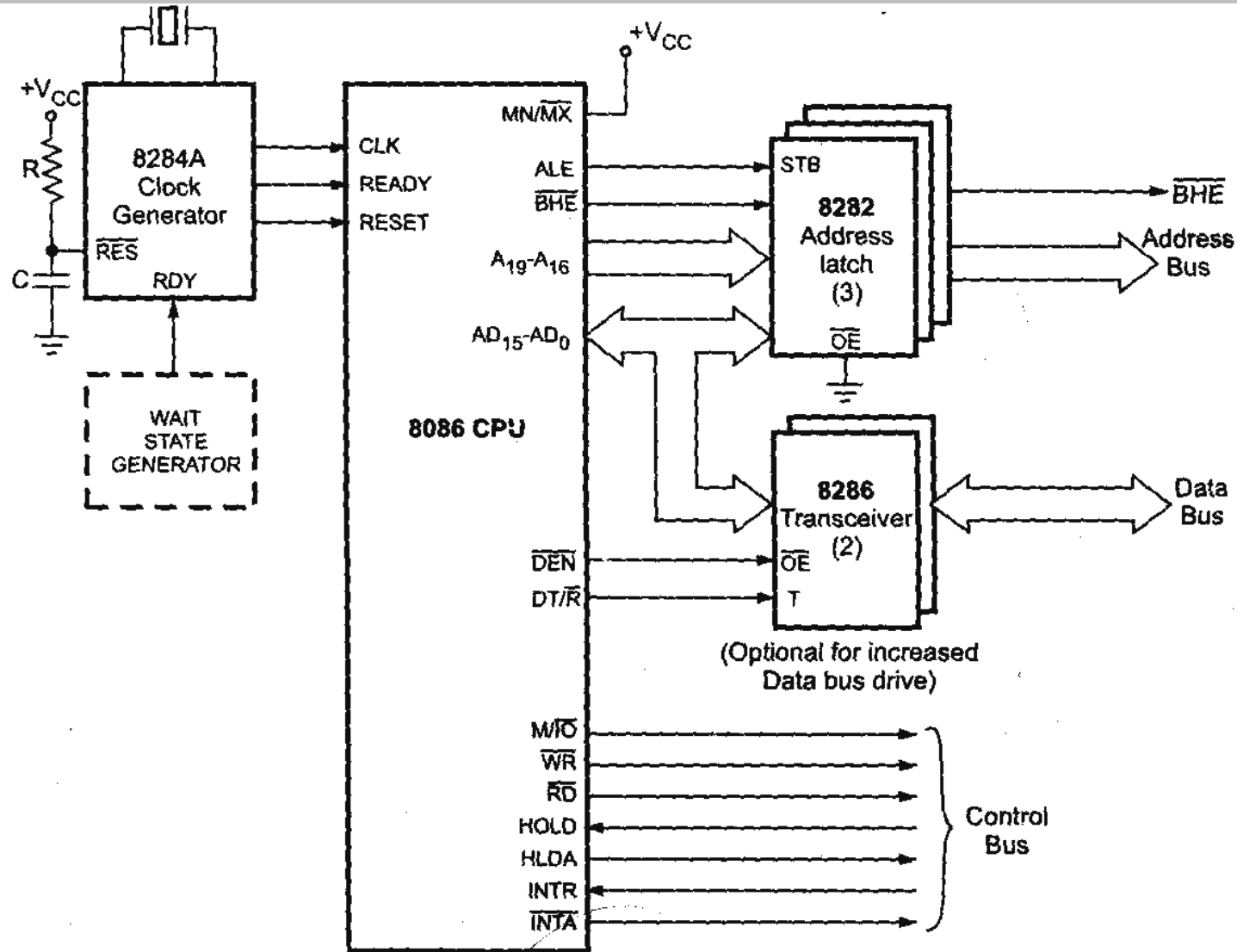
1. 8286 is an 8-bit Trans-receiver.
2. It acts as a bi-directional buffer, and increases the driving capacity of the data bus.
3. It is enabled when OE is low.
4. T controls the direction of data.
  - If T=1: data is transmitted.
  - If T=0: data is received.
5. As the data bus is 16 bits, 2 trans-receivers are required.
6. Its main function is to prevent address and allow data to be transferred on the data bus.



# 8284 – CLOCK GENERATOR



# MINIMUM MODE CONFIGURATION



# MINIMUM MODE CONFIGURATION

## What is Minimum Mode ?

- Programmed when MN/MX# is High
- Clock, Reset, Ready generated by 8284
- 8086 generates RD#, WR#, M/IO#, ALE, DEN#, DT/R#, HOLD, HLDA
- IOR#, IOW#, MEMR#, MEMWR# are generated with external decoder (74 LS 138)
- Single processor in the system and that is 8086, does not support additional processors

# MINIMUM MODE CONFIGURATION

$\overline{DEN}$	$DT/\overline{R}$	Action
1	X	Transceiver is disabled
0	0	Receive data
0	1	Transmit data

$M/\overline{IO}$	$\overline{RD}$	$WR$	Operation
1	0	1	Memory Read
1	1	0	Memory Write
0	0	1	I/O Read
0	1	0	I/O Write

# MINIMUM MODE CONFIGURATION

## Basic Terms Related to Timing Diagram

- T-State =  $1/\text{Clock Frequency}$
- Machine or Bus Cycle = Time to Execute One Read/Write Cycle
- Instruction Cycle = Total Time Required to Fetch and Execute one Instruction (Combination of Bus Cycles)

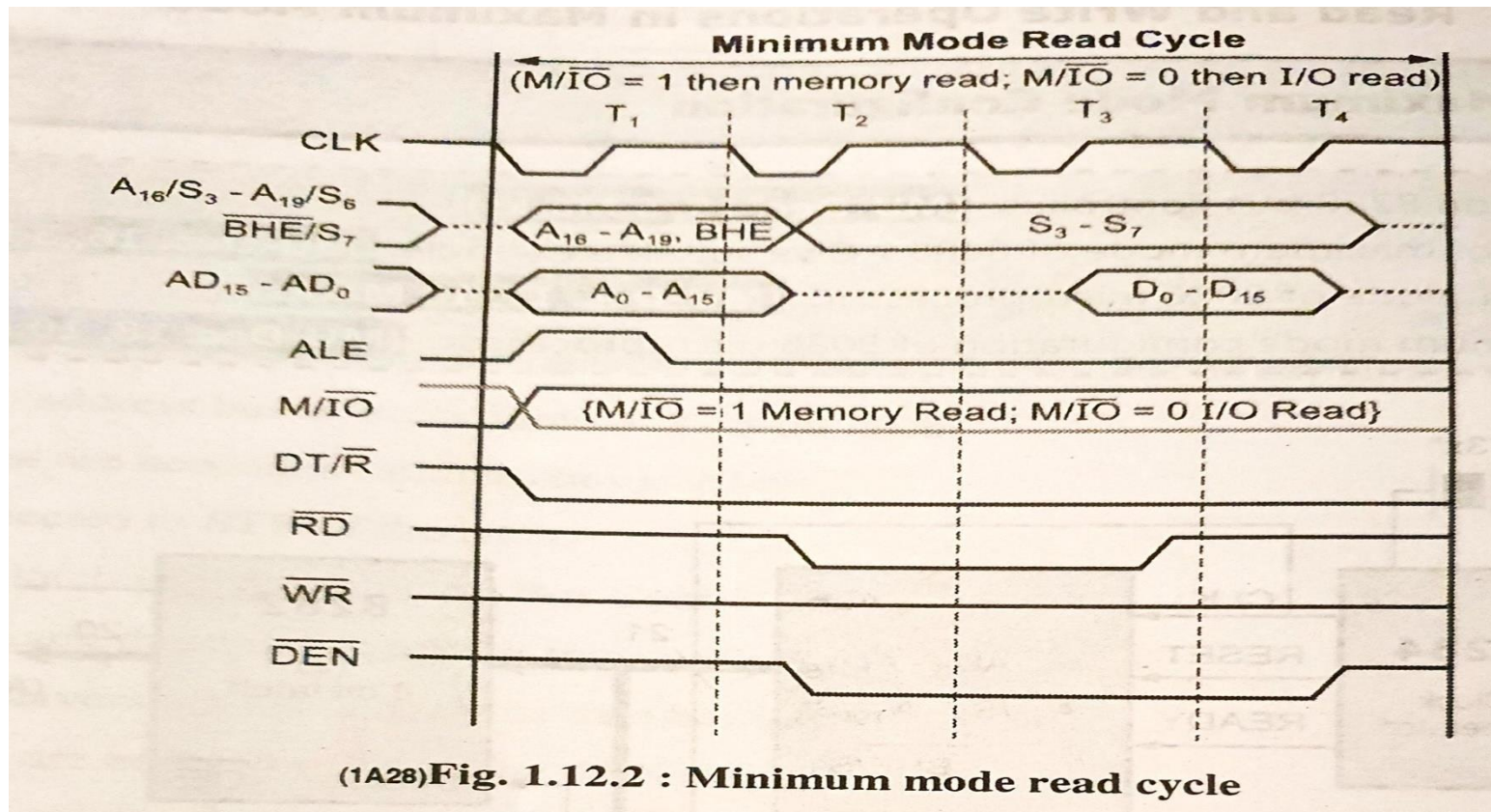


# MINIMUM MODE CONFIGURATION

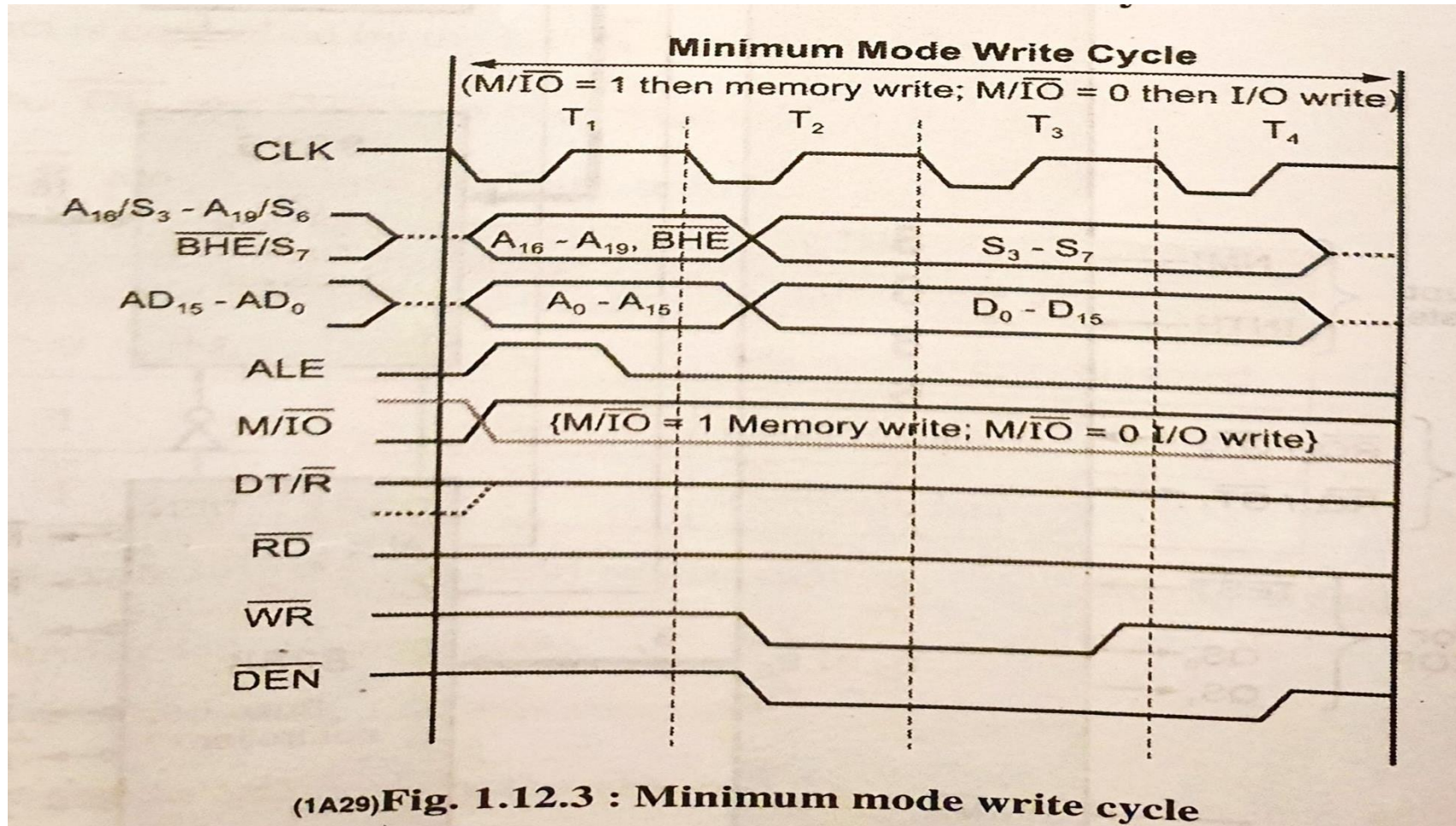
## Significance of Clock

- 8086 Operates at 5MHz, 8MHz & 10MHz
- Read/Write Cycle in 8086 is of 4-Clock Cycles
- At 5Mhz, one T-State =  $1/5\text{MHz} = 200 \text{ nsec}$ , So Read/Write Cycle =  $4 * 200 = 800 \text{ nsec}$
- At 8MHz, Read/Write Cycle =  $4 * 125 = 500 \text{ nsec}$
- At 10MHz, Read/Write Cycle =  $4 * 100 = 400 \text{ nsec}$

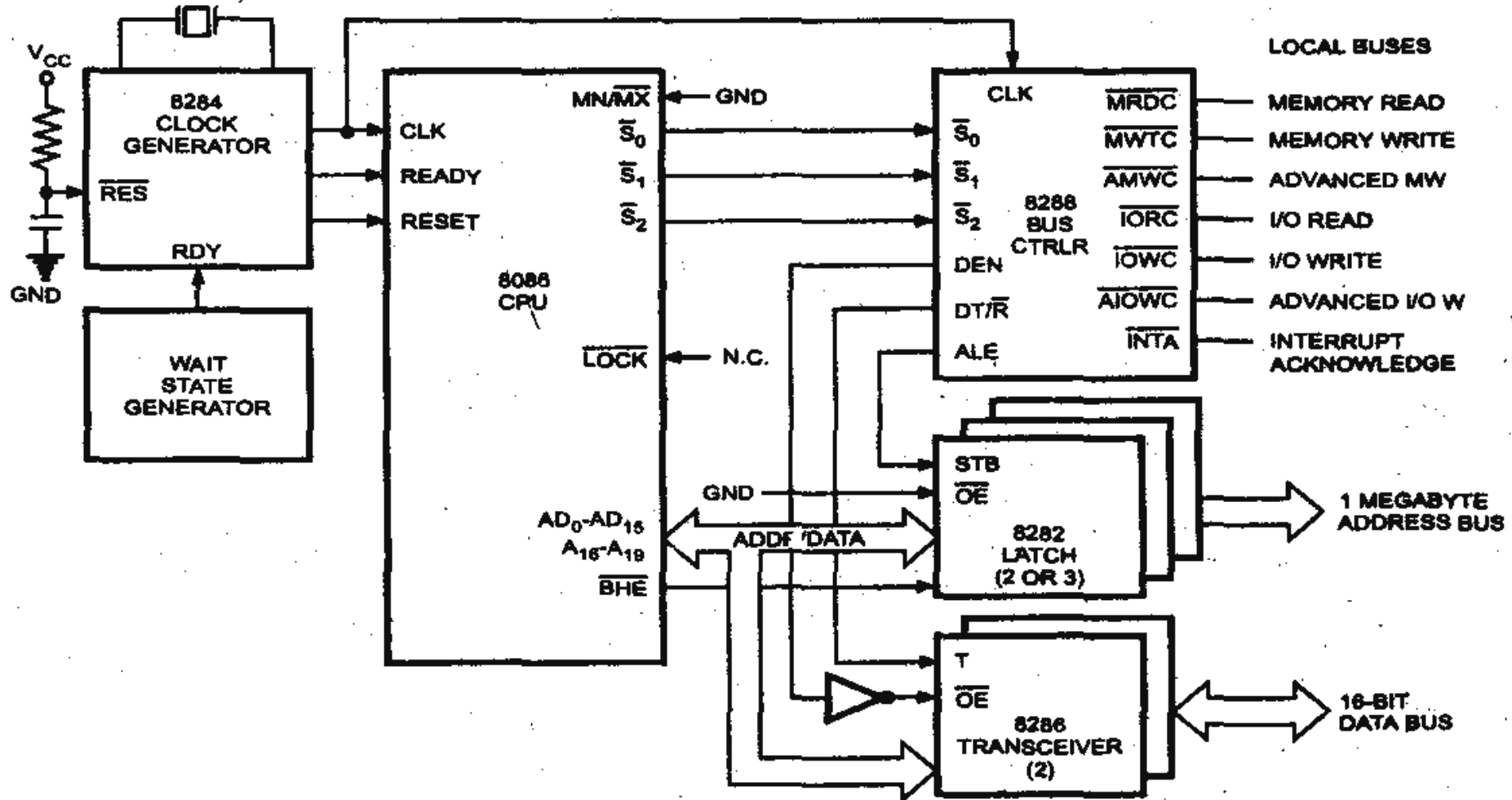
# MINIMUM MODE CONFIGURATION



# MINIMUM MODE CONFIGURATION



# MAXIMUM MODE CONFIGURATION



# MAXIMUM MODE CONFIGURATION

1. MN/MX=0
2. We can connect more processors to 8086 (8087/8089)
3. Clock provide by 8284 clock generator
4. The most significant part of maximum mode circuit is the 8288 bus controller.
5. Address from the address bus is latched into 8282 8 bit latch.
  1. Three latches are needed as address bus is 20 bit.
  2. ALE is connected to STB of the latch
  3. The ALE for this latch is given is given by 8288 bus controller.
6. The data bus is driven through 8286 8 bit transreceiver
  1. Two transreciever is needed, as the data bus is 16 bit.
  2. The transreciever are enabled through the DEN signal
  3. The direction of the data is controlled by DT/R signal.
  4. DEN is connected to OE and DT/R is connected to T
7. Control signals for all the operations are generated by decoding s2, s1 and s0 signals.

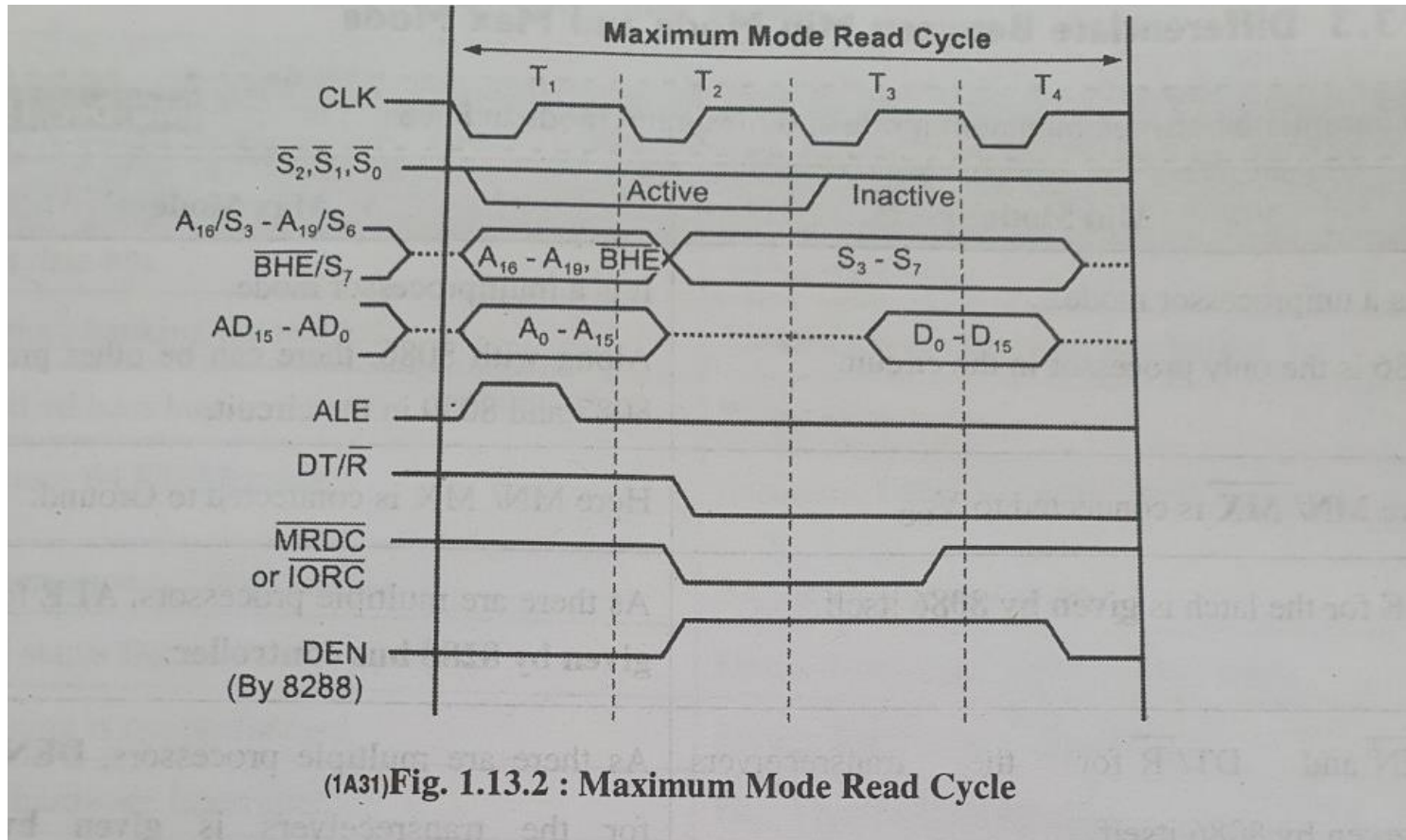


# MAXIMUM MODE CONFIGURATION

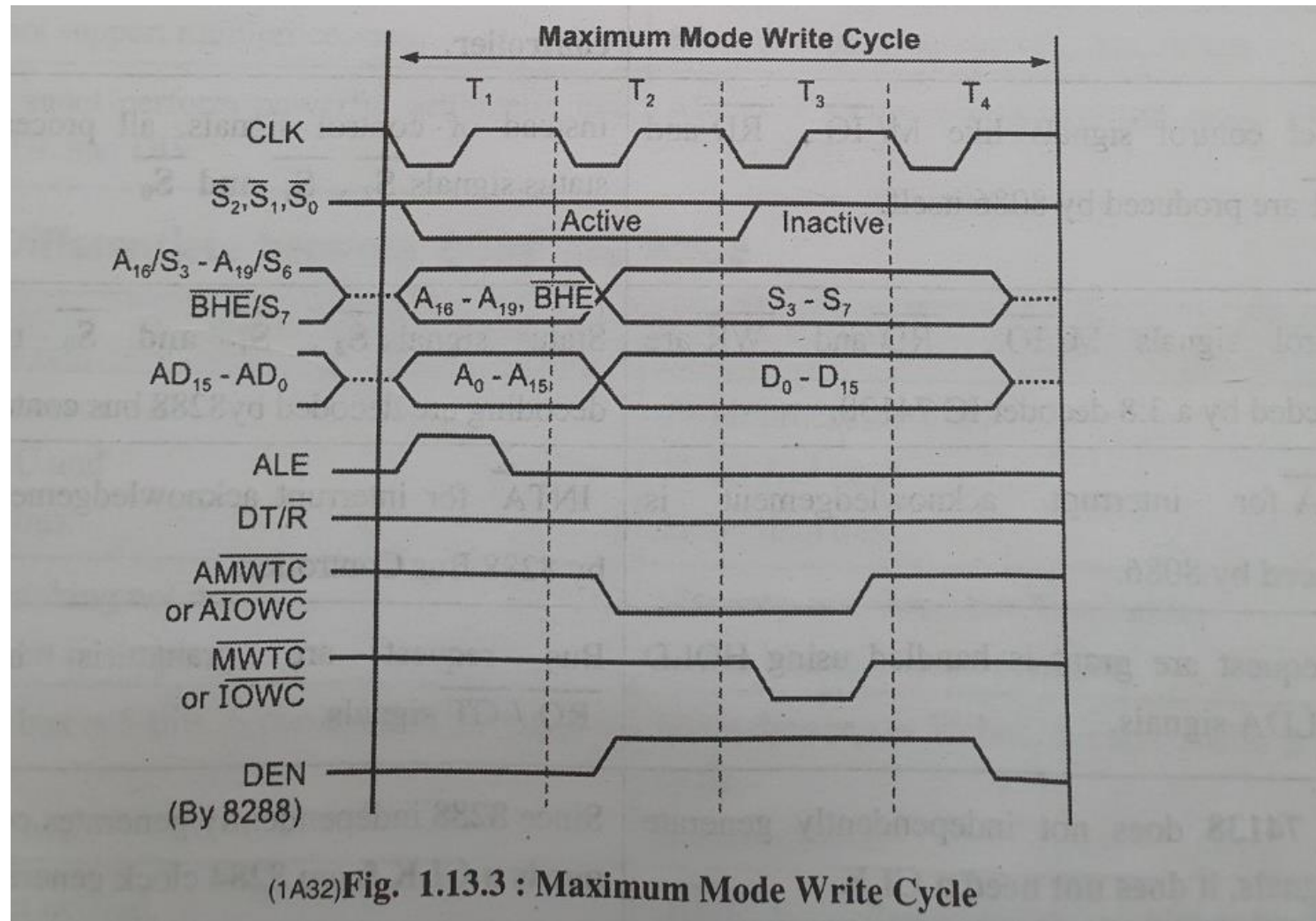
DEN (of 8288)	DT/R	Action
0	X	Transeceiver is disabled
1	0	Receive data
1	1	Transmit data

S2	S1	S0	Processor state
0	0	0	Int Ack
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Instruction Fetch
1	0	1	Memory Read
1	1	0	Memory Write
1	1	1	Inactive


# TIMING DIAGRAM



# TIMING DIAGRAM

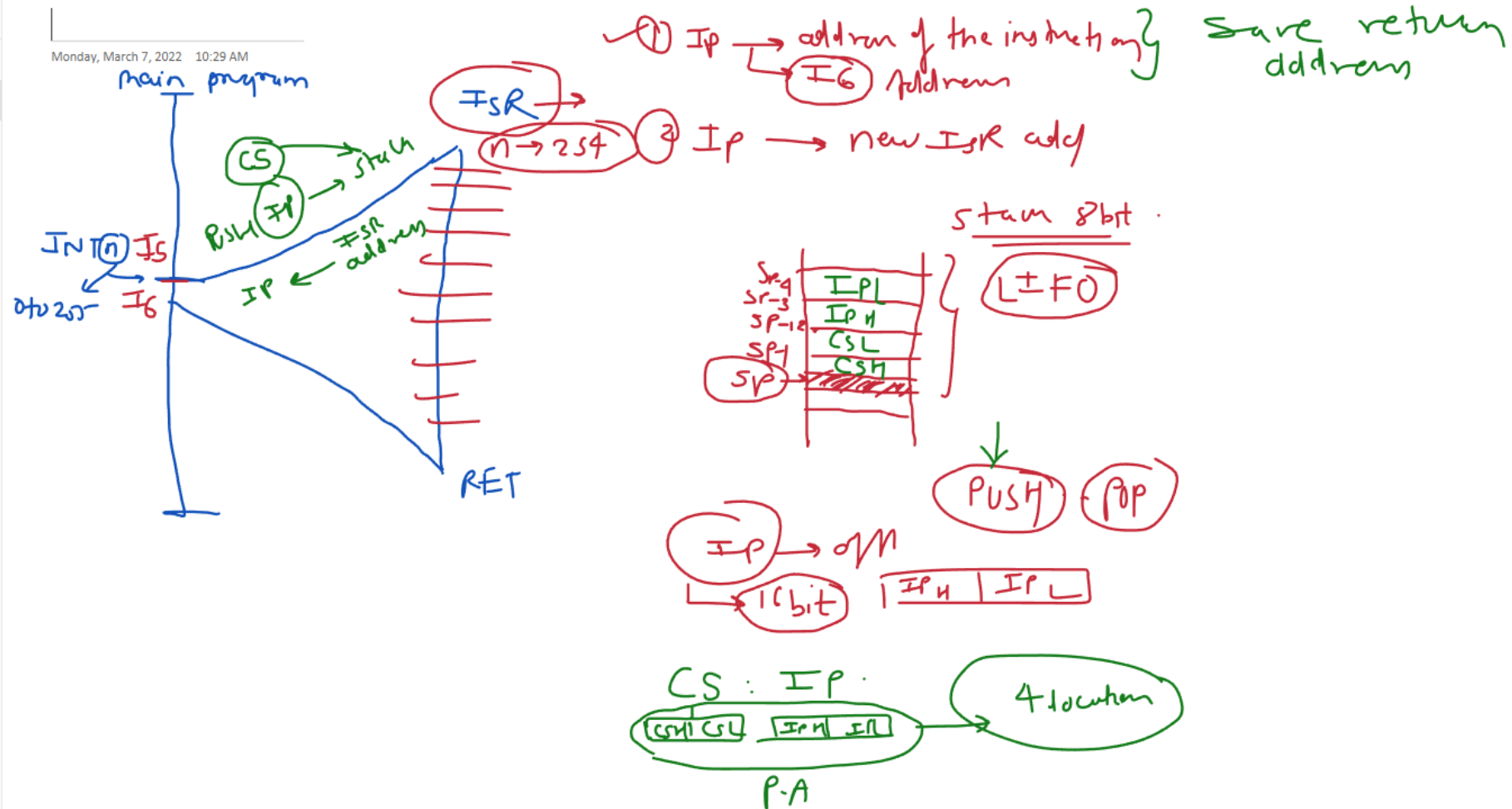


# 8086 INTERRUPTS

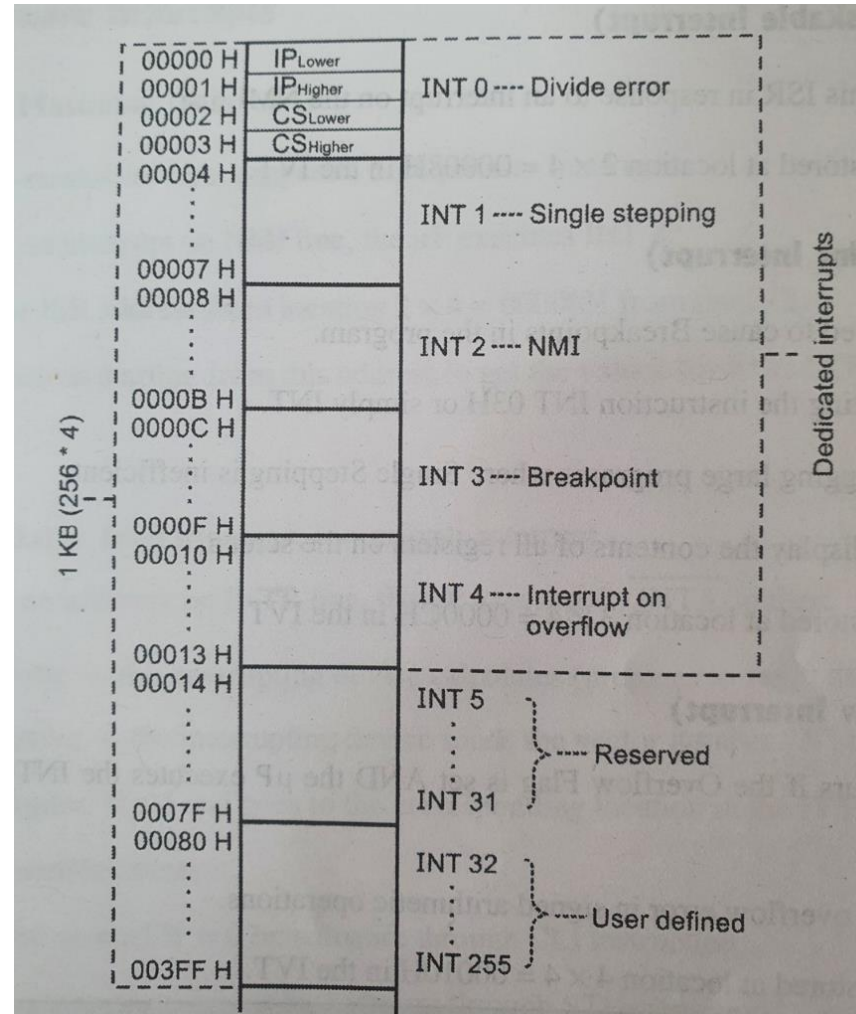
- An Interrupt is a special condition that arise during the working of the microprocessor.
- Microprocessor services it by executing a subroutine call Interrupt service routine.
- Every activity initiated through interrupt.
- External signal (hardware interrupts)
  - NMI (Vector interrupt)
  - INTR
- Special instructions (Software interrupts)
  - INT 'n' 

```
graph LR; A[➤ INT 'n'] --> B[Instructions]; A --> C[0 to 255]
```
  - All 256 interrupts can be invoked by software.

# 8086 INTERRUPTS



# 8086 INTERRUPTS





# 8086 INTERRUPTS

- The IVT contains ISR address for the 256 interrupts.
- Each ISR address is stored as CS and IP.
- As each ISR address is of 4 bytes (2-CS and 2-IP), each ISR address requires 4 locations to be stored.
- There are 256 interrupts: INT 0 ... INT 255 the total size of the IVT is  $256 \times 4 = 1\text{KB}$ .
- The first 1KB of memory, address 00000 H ... 003FF H, are reserved for the IVT.
- Whenever an interrupt INT N occurs,  $\mu\text{P}$  does  $N \times 4$  to get values of IP and CS from the IVT and
- hence perform the ISR.

# 8086 INTERRUPTS

- **Dedicated Interrupts: (INT 0 ... INT 4)**

- INT 0 (Divide Error)

- This interrupt occurs whenever there is division error
    - i.e. when the result of a division is too large to be stored.
    - This condition normally occurs when the divisor is very small as compared to the dividend or the divisor is zero.
    - Its ISR address is stored at location  $0 \times 4 = 00000H$  in the IVT.

- INT 1 (Single Step)

- The  $\mu P$  executes this interrupt after every instruction if the TF is set.
    - It puts  $\mu P$  in Single Stepping Mode i.e. the  $\mu P$  pauses after executing every instruction.
    - This is very useful during debugging.
    - Its ISR generally displays contents of all registers.
    - Its ISR address is stored at location  $1 \times 4 = 00004H$  in the IVT.

- INT 2 (Non Maskable Interrupt)

- The  $\mu P$  executes this ISR in response to an interrupt on the NMI line.
    - Its ISR address is stored at location  $2 \times 4 = 00008H$  in the IVT.

# 8086 INTERRUPTS

- **Dedicated Interrupts: (INT 0 ... INT 4)**

## INT 3 (Breakpoint Interrupt)

- This interrupt is used to cause Breakpoints in the program.
- It is caused by writing the instruction INT 03H or simply INT.
- It is useful in debugging large programs where Single Stepping is inefficient.
- Its ISR is used to display the contents of all registers on the screen.
- Its ISR address is stored at location  $3 \times 4 = 0000\text{CH}$  in the IVT.

## INT 4 (Overflow Interrupt)

- This interrupt occurs if the Overflow Flag is set AND the  $\mu\text{P}$  executes the INTO instruction
- (Interrupt on overflow).
- It is used to detect overflow error in signed arithmetic operations.
- Its ISR address is stored at location  $4 \times 4 = 00010\text{H}$  in the IVT.

# 8086 INTERRUPTS

## Reserved Interrupts: (INT 5 ... INT 31)

. These levels are reserved by INTEL to be used in higher processors like 80386, Pentium etc. They are not available to the user.

## User defined Interrupts : (INT 32 ... INT 255)

- These are user defined, software interrupts.
- ISRs for these interrupts are written by the users to service various user defined conditions.
- These interrupts are invoked by writing the instruction INT n.
- Its ISR address is obtained by the  $\mu P$  from location  $n \times 4$  in the IVT.

# 8086 INTERRUPTS

## HARDWARE INTERRUPTS

### NMI (Non Maskable Interrupt)

- This is a non-maskable, edge triggered, high priority interrupt.
- On receiving an interrupt on NMI line, the  $\mu P$  executes INT 2.
- $\mu P$  obtains the ISR address from location  $2 \times 4 = 00008H$  from the IVT.
- It reads 4 locations starting from this address to get the values for IP and CS, to execute the ISR.

### INTR

- This is a maskable, level triggered, low priority interrupt.
- On receiving an interrupt on INTR line, the  $\mu P$  executes 2 INTA pulses.
- 1st INTA pulse --- the interrupting device calculates (prepares to send) the vector number.
- 2nd INTA pulse --- the interrupting device sends the vector number "N" to the  $\mu P$ .
- Now  $\mu P$  multiplies  $N \times 4$  and goes to the corresponding location in the IVT to obtain the ISR address.
- INTR is a maskable interrupt.
- It is masked by making  $IF = 0$  by software through CLI instruction.
- It is unmasked by making  $IF = 1$  by software through STI instruction.

# HOW 8086 RESPOND TO INTERRUPTS

## Response to any interrupt --- INT N

i) The  $\mu$ P will PUSH Flag register into the Stack.

SS:[SP-1], SS:[SP-2]  $\longleftarrow$  Flag -----SP SP - 2

ii) Clear IF and TF in the Flag register and thus disables INTR interrupt.

IF 0, TF 0

iii) PUSH CS into the Stack.

SS:[SP-1], SS:[SP-2]  $\longleftarrow$  CS -----SP - 2

iv) PUSH IP into the Stack.

SS:[SP-1], SS:[SP-2]  $\longleftarrow$  IP-----SP SP - 2

v) Load new IP from the IVT

IP [N x 4], [N x 4 + 1]

vi) Load new CS from the IVT

IP [N x 4 + 2], [N x 4 + 3]

Since CS and IP get new values, control shifts to the address of the ISR and the ISR thus begins. At the end of the ISR the microprocessor encounters the IRET instruction and returns to the main program in the following steps.



# HOW 8086 RESPOND TO INTERRUPTS

## Response to IRET instruction

i) The  $\mu$ P will restore IP from the stack

IP SS:[SP], SS:[SP+1]

SP SP + 2

ii) The  $\mu$ P will restore CS from the stack

CS SS:[SP], SS:[SP+1]

SP SP + 2

iii) The  $\mu$ P will restore FLAG register from the stack

Flag SS:[SP], SS:[SP+1]

SP SP + 2

# 8086 INTERRUPTS

## Interrupt Priorities

Interrupt	Priority	
	(Simultaneous occurrence)	(To interrupt another ISR)
Divide Error, INT n, INTO	1 (Highest)	Can interrupt any ISR
NMI	2	
INTR	3	Cannot interrupt an ISR (IF, TF $\leftarrow$ 0)
Single Stepping	4(Lowest)	

# 8086 INTERRUPTS

Priority in 8086 interrupts is of two types:

## 1. Simultaneous Occurrence:

- When more than one interrupts occur simultaneously then, all s/w interrupts except single stepping, get the highest priority.
- This is followed by NMI. Next is INTR. Finally, the lowest priority is of the single stepping interrupt.
- Eg: Assume the  $\mu P$  is executing a DIV instruction that causes a division error and simultaneously INTR occurs.
- Here INT 0 (Division error) will be serviced first i.e. its ISR will be executed, as it has higher priority, and then INTR will be serviced.

## 2. Ability to interrupt another ISR:

- Since software interrupts (INT N) are non-maskable, they can interrupt any ISR. NMI is also non-maskable hence it can also interrupt any ISR.
- But INTR and Single stepping cannot interrupt another ISR as both are disabled before  $\mu P$  enters an ISR by IF 0 and TF 0.