UNIT-1

Introduction

→ Intel introduced its first 4-bit microprocessor 4004 in 1971 and its 8-bit microprocessor 8008 in 1972.

These microprocessors could not survive as general purpose microprocessors due to their design and performance limitations.

→ The launch of the first general purpose 8-bit µp 8080 in 1974 by intel.

→ The µp 8085 followed 8080, with a few more added features to its architecture.

The main limitations of the 8-bit µp were their low speed, low memory addressing capability, limited number of General purpose registers and a less powerful instruction set.

→ In the family of 16-bit microprocessors, intel's 8086 was the first one to be launched in 1978.

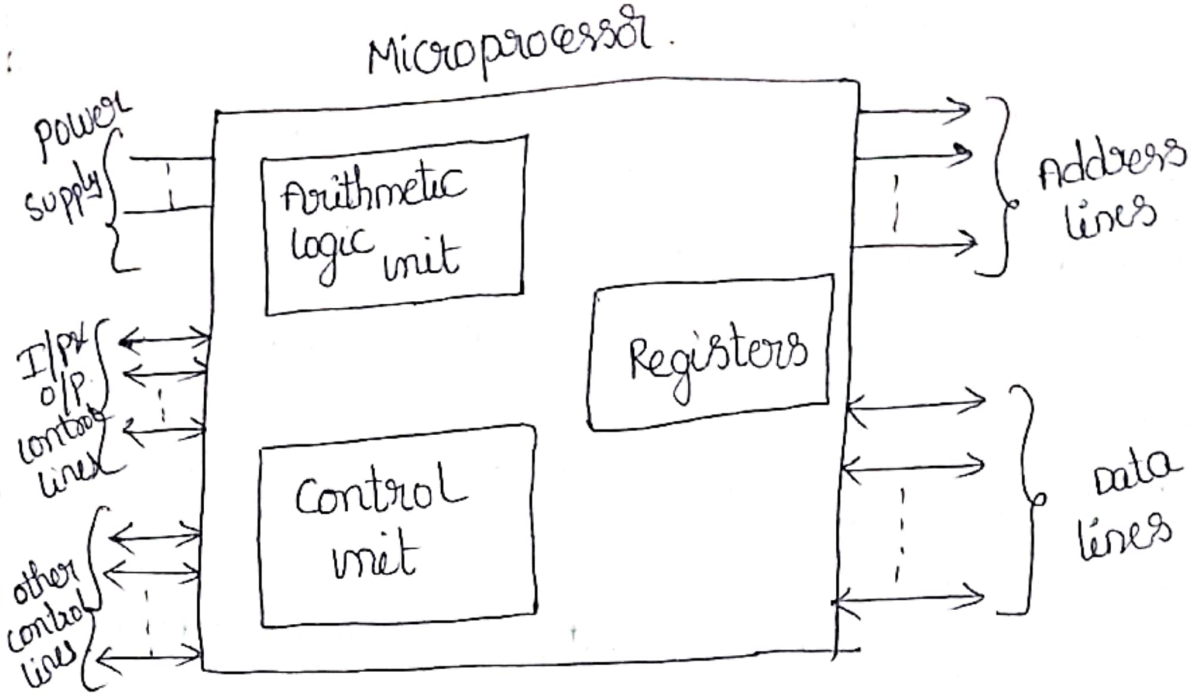
→ The 8086 µp has a much more powerful and high speed over the 8-bit microprocessors

Basic microprocessor Architecture:-

→ A µp is just like a electronic chip It is more powerful than just any other chip.

→ Inside the chip, there is an arithmetic logic unit (ALU). The ALU executes all arithmetic and logic instructions.

→ The control unit generates the external signals.

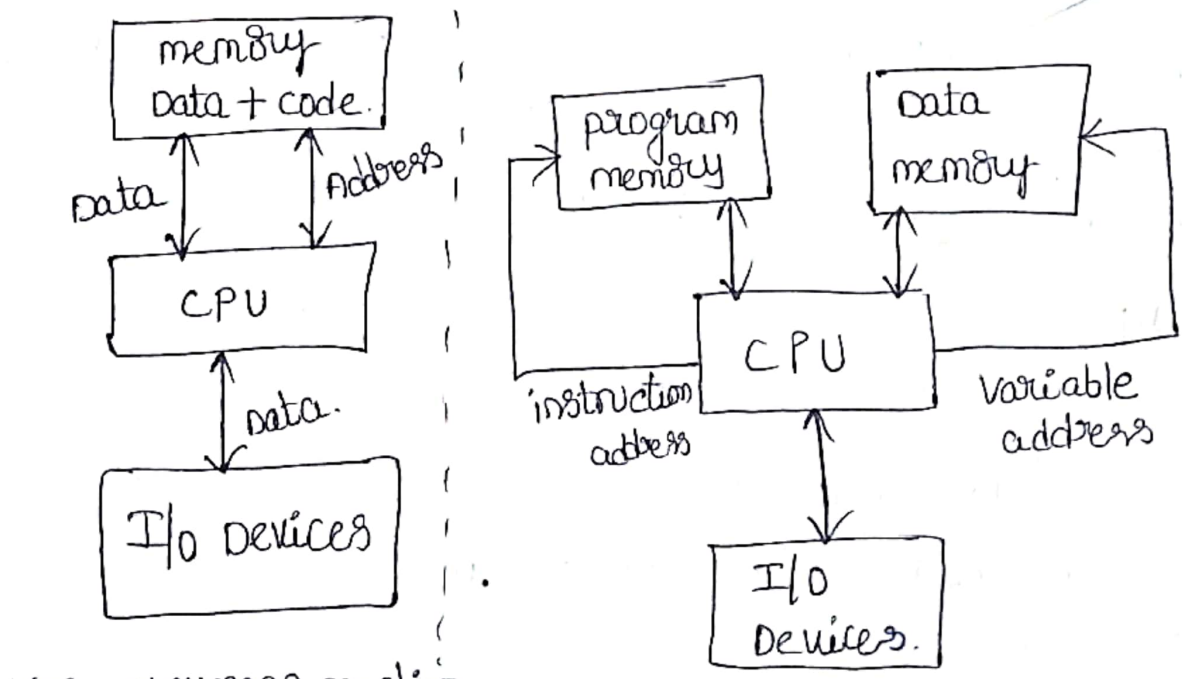


Inside and outside a microprocessor.

→ It controls the operation of the internal on-chip circuitry.

→ The on-chip memory, in the form of registers, is generally very limited.

Harvard and Von-Neumann Architecture :-



Von-Neumann Architecture	Harvard architecture
→ It is ancient computer architecture based on stored program computer concept.	→ It is modern computer architecture based on Harvard mark I relay based model.
→ Same physical memory address is used for instructions and data.	→ Separate physical memory address is used for instructions and data.
→ There is common bus for data and instruction transfer.	→ Separate buses are used for transferring data and instruction.
→ Two clock cycles are required to execute single instruction.	→ An instruction is executed in single cycle.
→ It is cheaper in cost	→ It is costly than Von-Neumann Architecture.
→ cpu can not access instructions and read/write at the same time	→ cpu can access instructions and read/write at the same time.
→ It is used in PC and small computers.	→ It is used in micro controllers and signal processing.
→ pipeling is not possible.	→ pipeling is possible.
→ Simple in design	→ Complex in design.

→ For General purpose computing Von-Neumann Architecture is mostly used as it is easier and cheaper to implement.

→ Harvard architecture is used widely, especially when speed is a major factor, as in use of Embedded systems.

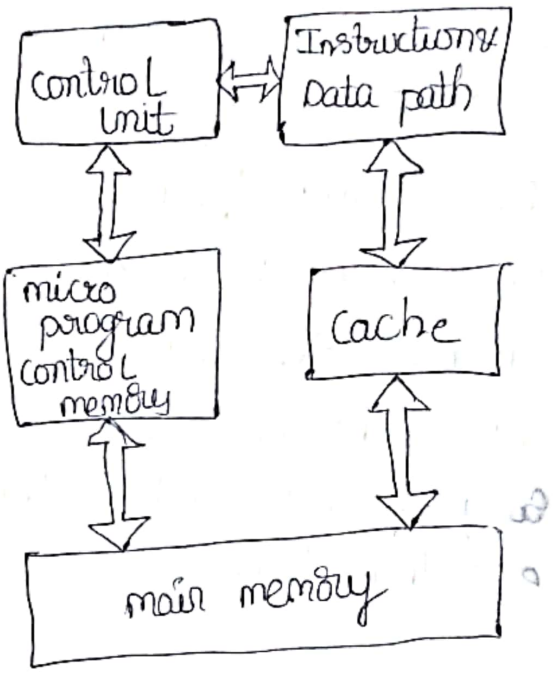
Microprocessor unit versus micro controller unit

microprocessor (μP)	micro controller (μC).
→ A microprocessor requires an external memory for program/ data storage.	→ It has on-chip RAM for data storage, and on-chip ROM for code.
→ It requires more time to access memory.	→ It requires less time to access memory.
→ It contains cpu, timing & control circuit and Register array.	→ It contains μp, on-chip RAM, on-chip ROM, I/o ports, and timers / counters.
→ Requires more hardware	→ Requires less hardware.
→ Used in general purpose applications.	→ microcontrollers have numerous applications many of them are application specific.
→ Single (common) memory for data & code (von-Neumann Architecture)	→ separate memory for data & code (Harvard Architecture)
→ microprocessors have good computing power, and they have higher clock speed to facilitate faster computation.	→ A micro controller clock speed is limited only to a few tens of MHz.
→ Do not have power saving mode.	→ Have a power-saving mode.
→ Used in Personal computers	→ Used in Embedded systems.
→ Less no. of Registers	→ more no. of Register.
→ cost is high	→ cost is Low.

CISC and RISC architectures :-

CISC Architecture :-

The main intend of the CISC processor architecture is to complete task by using less number of assembly lines which operates memory banks of a computer directly without making the compiler to perform storing and loading functions.



CISC Architecture

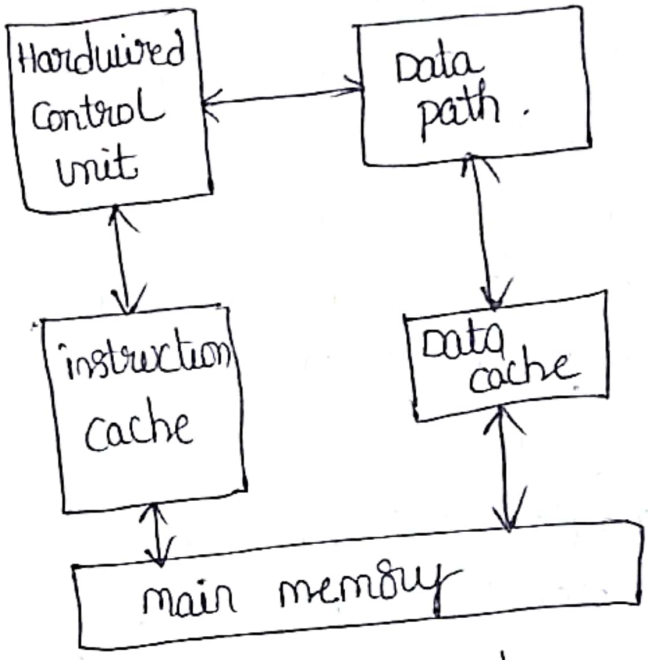
1. Complex instruction, hence complex instruction decoding.
2. instructions are larger than one-word size.
3. instruction may take more than a single clock cycle to get executed.
4. To simplify the computer architecture, CISC supports microprogramming.
5. More data types.
6. Because of the complex instruction set of the CISC the pipelining technique is difficult.

RISC Architecture :-

1. Simpler instruction, hence simple instruction decoding.
2. instruction comes undersize of one word.
3. instruction takes a single clock cycle to get executed.
4. Simple Addressing modes.

5. Pipelining technique of RISC executes multiple stages of instructions simultaneously
6. It simplifies the compiler design by using general purpose registers.

RISC stands for Reduced instruction set computer. In RISC, the instruction set of the computer is simplified to reduce the execution time.



RISC Architecture.

→

CISC	RISC
1. Emphasis on hardware	→ Emphasis on Software.
2. multiple instruction sizes and formats.	→ instruction comes under size of one word.
3. uses less Registers	→ uses more Registers.
4. more addressing modes	→ less addressing modes.
5. pipelining is difficult	→ pipelining is easy.
6. complex instruction decoding	→ Easy instruction decoding
7. Small code size	→ Large code size.
8. instructions take a varying amount of cycle time.	→ Instructions take one cycle time.

8086 Architecture

Features of 8086 :-

1. The 8086 is a 16-bit microprocessor.
2. The 8086 has a 16-bit data bus, so it can read data from or write data to memory and ports either 16 bits or 8 bits at a time.
3. The 8086 has a 20-bit address bus, so it can directly access 2^{20} or 10,48,576 (1mb) memory locations.
4. The 8086 can generate 16-bit I/o address, hence it can access $2^{16} = 65536$ I/o ports.
5. The 8086 provides fourteen 16-bit registers.
6. The 8086 has multiplexed address and data bus which reduces the no. of pins needed, but does slow down the transfer of data.
7. The 8086 requires one phase clock with a 33% duty cycle to provide optimized internal timing. (5MHz)
8. The 8086 is possible to perform bit, byte, word and block operations in 8086.
9. The Intel 8086 is designed to operate in two modes namely the minimum mode and the maximum mode.
10. An interesting feature of the 8086 is that it fetches upto six instruction bytes from memory.
11. The 8086 provides powerful instruction set with the addressing modes: Register, immediate, direct, indirect index or base, relative and implied.

Architecture of 8086:-

It is internally divided into two separate functional units. These are the Bus interface unit (BIU) and the Execution unit (EU)

Bus interface unit:-

The bus interface unit is the 8086's interface to the outside world. It provides a full 16-bit bi-directional data bus and 20-bit address bus. It sends address of the memory or I/O, fetches instructions from memory, read/write the data from port/memory and supports instruction queuing.

To implement these functions the BIU contains the instruction queue, Segment Registers, instruction pointer, address summer and bus control logic.

Instruction queue:-

To speed up program execution, the BIU fetches six instruction bytes from the memory. These pre-fetched instruction bytes are held for the execution unit in a group of registers called queue. with the help of queue it is possible to fetch next instruction when current instruction is in execution.

The queue operates on the principle first in first out (FIFO). so that the execution unit gets the instructions for execution in the order they are fetched. It supports pipelining.

Block diagram of 8086:-

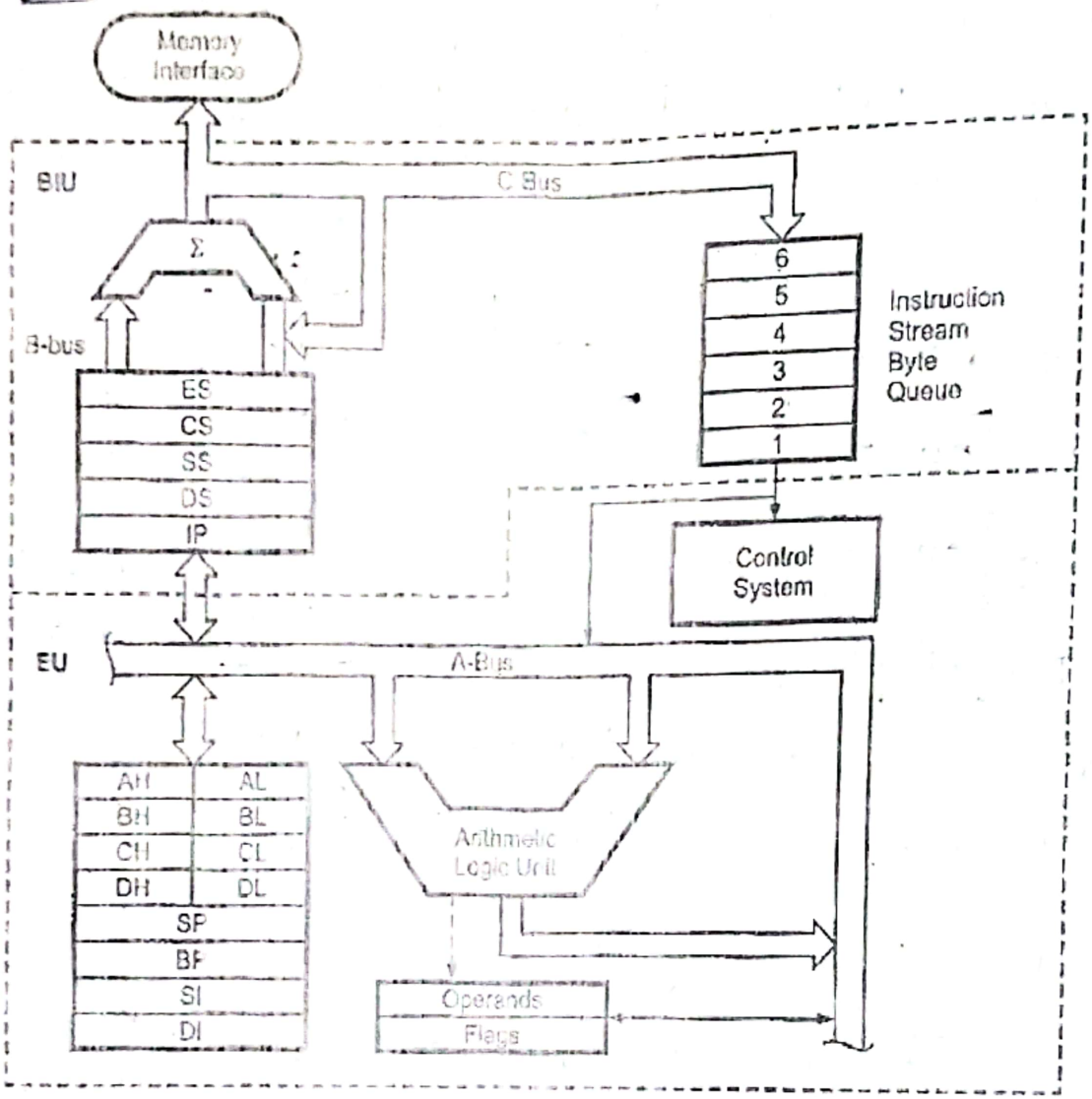


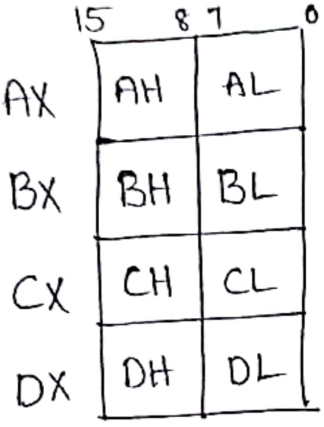
Fig. 1.8 8086 Internal block diagram

Execution unit :-

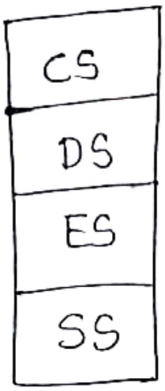
The Execution unit 8086 tells the BIU from where to fetch instructions & data, decodes instructions and executes instructions. It contains control circuitry, instruction decoder, ALU, Register, Flag Register. General purpose Registers and pointers and Index Registers.

Register Organisation of 8086

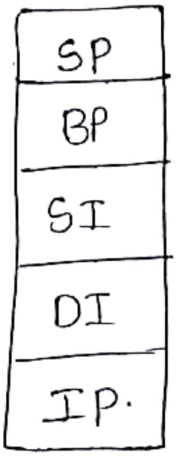
1. 8086 has a powerful set of Registers known as general purpose and special purpose registers. All of them are 16-bit registers.



(a) General purpose Register.



(b) Segment Registers.



(c) Index & pointer Register

(d) Flag Register.

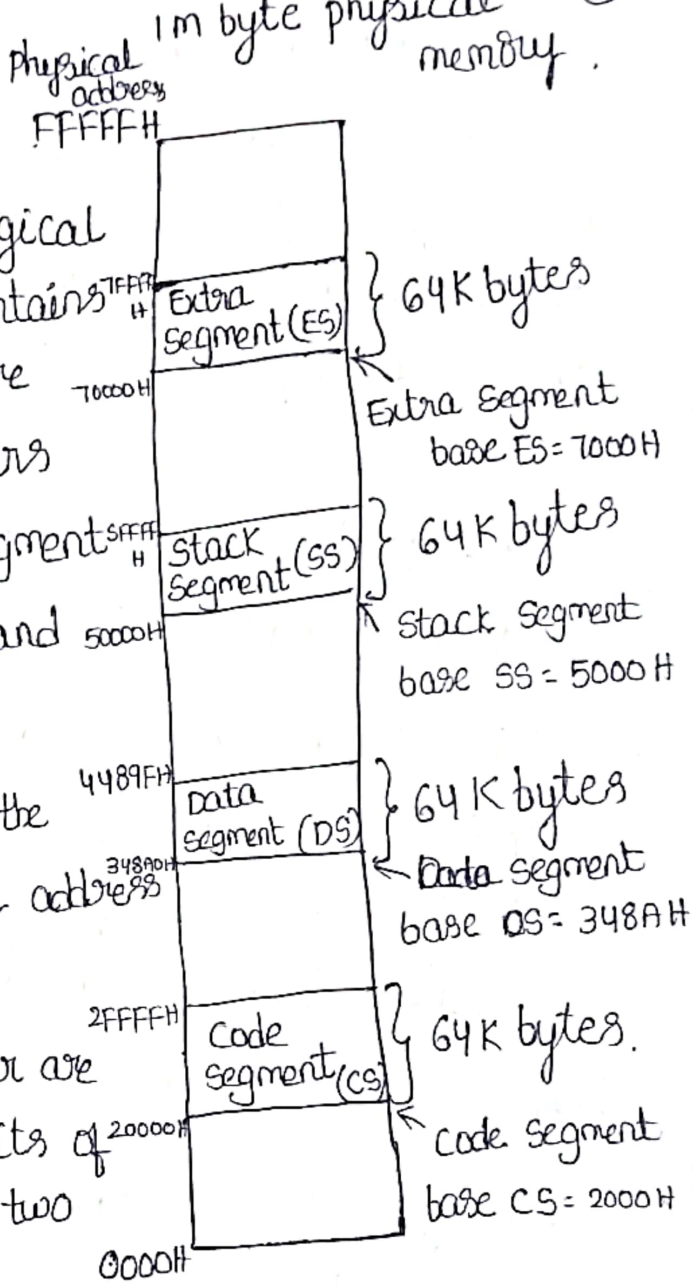
General purpose Registers:-

The 8086 has four 16-bit general purpose Registers labeled AX, BX, CX and DX. Each 16-bit general purpose register can be split into two 8-bit registers.

The general purpose registers are either used for holding data, variables and intermediate results temporarily. They can also be used as a counters or used for storing offset address for some particular addressing modes. AX is used as 16-bit accumulator. BX is also used as offset storage, CX used as a default counter. DX register used as implicit operand or destination in case of a few instructions.

Segment Registers:-

The complete 1 megabyte memory is divided into 4 logical segments. Each segment contains 64K bytes of memory. There are four segment registers code segment (CS), Data segment (DS), Extra segment (ES), and Stack segment (SS).



→ The CS register is holds the upper 16-bits of the starting address of the segment.

→ ES Registers and DS register are used to hold the upper 16-bits of the starting address of the two memory segments which are used for data.

→ The SS register is used for addressing stack segment memory which is used to store stack data. The CPU uses the stack for temporarily storing important data.

The CS, DS, SS and ES segment registers contains the segment addresses for the code, data, stack and Extra segments of memory.

Pointers and Index Registers:-

All Segment Registers are 16-bit wide. But it is necessary to generate 20-bit address (physical address) on the address bus. To get 20-bit physical address one or more pointer or index registers are associated with each segment register.

→ The pointer Registers IP, BP and SP are associated with code, data and Stack segments.

→ The Index Registers DI and SI are used as a general purpose registers as well as for offset storage in case of indexed, base indexed and relative based indexed addressing modes.

How to calculate 20-bit (physical address):-

For generating a physical address from contents of these two Registers, the content of a segment register also called as Segment address is shifted left bit-wise four times and to this result content of an offset Register also called as offset address is added, to produce a 20-bit physical address. For Example segment address is 1005H. and the offset is 5555H. then the physical address is calculated as.

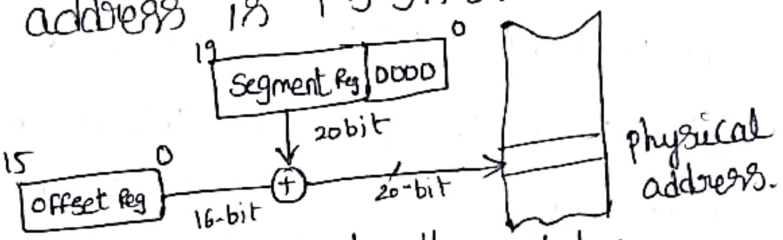
Segment address 1005H
offset address 5555H.

Segment address 10050
(left shifted) → 0001 0000 0000 0101 0000
by
+ 4 bits / multiplied by 10H 0101 0101 0101 0101
OFFset address →

0001 0101 0101 1010 0101
1 5 5 A 5.

Final phaysical address is 155A5.

Flag Registers :-



The flag register is used to indicate the status information (or) condition produced by an instruction execution. The 8086 has 6- status flags and 3- control flags.

U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

conditional flags are CF, PF, AF, ZF, SF, OF.
control flags are DF, IF, TF.

→ Carry Flag (CF) :- It is set to 1, if carry is generated in Addition (or) subtraction.

→ Parity Flag (PF) :- It is set to '1', if the result of an operation has even number of 1's.

→ Auxiliary carry Flag (AF):- It is used in BCD operations. if there is an overflow out of bit 3 carry from lower nibble to higher nibble (D_3 bit to D_4 bit).

→ zero Flag (ZF):- it is set to 1, if the result of an operation is zero.

→ Sign Flag (SF):- It is used with signed numbers only SF = 1 for -ve results and SF = 0 for +ve results.

→ overflow flag (OF):- It is set to 1, if the result of a signed operation exceeds the register capacity.

→ Trap Flag (TF):- It is used for debugging purpose. If TF = 1, the processor enters the 'single step execution mode' the 8086 gets interrupted automatically at the end of every instruction execution.

→ Directional Flag (DF):- It is used in string operations to select either increment (or) decrement mode for SI and DI registers.

DF = 0 → increment mode and DF = 1 decrement mode.

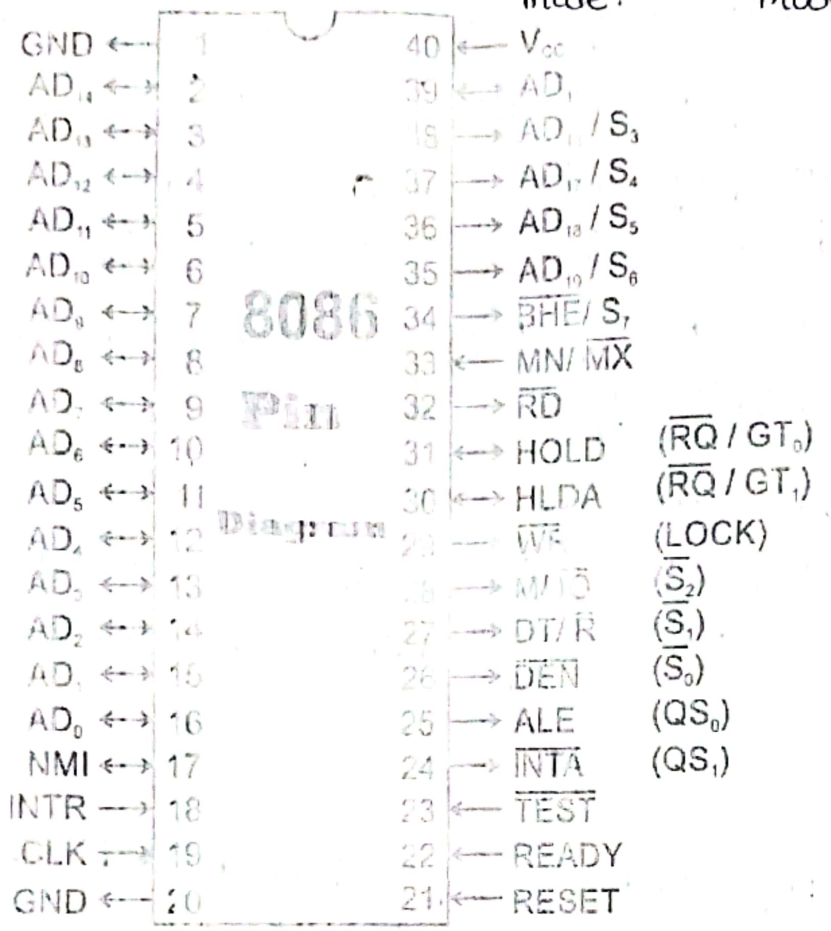
→ interrupt Flag (IF):- if IF = 1, all the maskable interrupts are recognized and processed by the CPU. otherwise, these interrupts are ignored.

Example:-

```
  79 H →  0111 1001
+ 88 H →  1000 1000
          ++++
ADD       1 0000 0001
```

CF = 1
PF = 0
AF = 1
ZF = 0
SF = 0
OF = 0.

Pin diagram of 8086 :-



The microprocessor 8086 is a 16-bit CPU available in three clock rates 5, 8, and 10 MHZ, packaged in a 40 pin Dual line package (DIP) or plastic package.

$AD_{15} - AD_0$:- These are the time-multiplexed memory I/O address and data lines. Address remains on the lines during T_1 state, while the data is available on the data bus during T_2, T_3, T_W and T_4. T_W is a wait state.

$A_{19}|S_6$, $A_{18}|S_5$, $A_{17}|S_4$, $A_{16}|S_3$:- These are the time multiplexed address and status lines. During I/O operations, these lines are low. Status information is available on these lines for T_2, T_3, T_W and T_4.

→ The S_4 and S_3 together indicate enable which segment register is presently being used for memory accesses.

→ The status of the interrupt Enable ~~bit~~ flag bit (displayed on S_5) is updated at the beginning of each clock cycle.

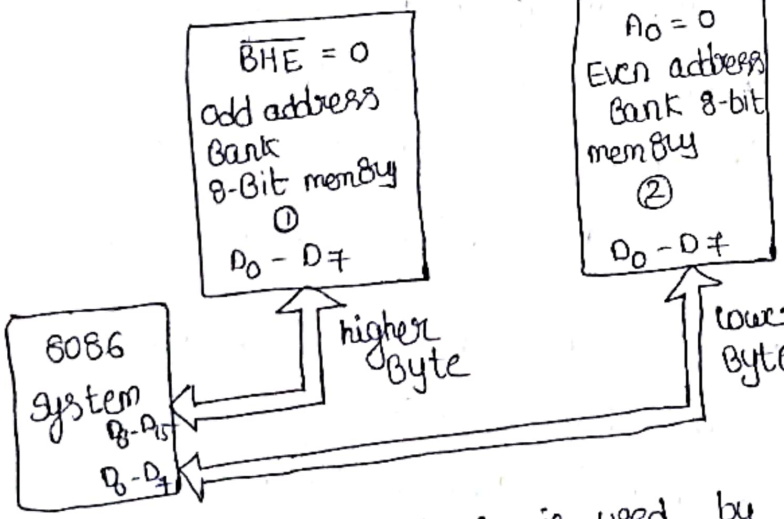
S_4	S_3	indications
0	0	Alternate Data
0	1	stack
1	0	Code & none
1	1	Data.

→ The status line S_6 is always low. The address bits are separated from the status bits using latches controlled by the ALE signal.

→ $\overline{BHE}/S_7$:-

The 1MB physical memory of 8086 is divided into two banks for accessing 16-bit numbers. Each bank size is 512K bytes.

physical memory organisation :-



$\overline{BHE}$	A_0	Bank selected
0	0	Both banks are enabled for 16-bit operation
0	1	High bank is Enabled
1	0	Low bank is enabled
1	1	No banks are Enabled.

→ The status signal S_7 is used by arithmetic co-processor 8087 to determine whether the CPU is 8086 (or) 8088.

→ NMI :- It is a positive edge triggered non maskable interrupt request. This interrupt has highest priority than INTR. It is a vectored interrupt and the vector address of NMI is 0000:0008H.

→ INTR :- It is a level triggered maskable interrupt request. It can be disabled by using software. the processor will get interrupted only if IF = 1, otherwise the processor will not get interrupted even INTR is active. It is a non-vectored interrupt.

→ CLK :- The clock input provides the basic timing for µp and bus control activity. the clock frequencies of different versions of 8086 are 5MHz, 10MHz & 8MHz.

→ Reset :- It clears DS, ES, SS, IP, Flag registers and the instruction queue. (0000H). CS is initalized to FFFFH. when Reset removed, 8086 starts execution from FFFF0H.

→ Ready :- A slow peripheral (or) memory device can be connected to µp through Ready line.
If Ready = 1, I/O device is ready to transfer data
If Ready = 0, the processor waits until it goes to high.

→ $\overline{TEST}$:- The 8086 enters into a wait state after execution of the wait instruction until a low signal on $\overline{TEST}$
If $\overline{TEST}$ = 0, the wait instruction functions as NOP.
if $\overline{TEST}$ = 1, the processor waits until $\overline{TEST}$ = 0.

→ $\overline{RD}$:- Read control signal is active whenever the processor is ready to read the data from memory or I/O.

→ MN/$\overline{MX}$:- 8086 can be operated in two modes.

MN/$\overline{MX}$ = 1 for minimum mode operation.

MN/$\overline{MX}$ = 0 for maximum mode operation.

→ $\overline{INTA}$:- The interrupt acknowledge signal is a response to the INTR.

→ ALE :- This signal provided by 8086 to demultiplex the AD_0-AD_{15} into A_0-A_{15} and D_0-D_{15} using external latches.

→ $\overline{DEN}$:- This signal enable the bi-directional data bus Buffer. (the CPU is ready to send & receive data).

→ DT/$\overline{R}$ (Data transmit | receive).

DT/$\overline{R}$ = 1 8086 is transmitting the data

DT/$\overline{R}$ = 0 8086 is Receiving the data.

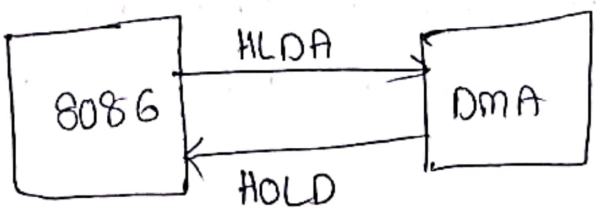
→ M/$\overline{IO}$:- M/$\overline{IO}$ = 1 memory data transfer

M/$\overline{IO}$ = 0 I/o data transfer.

→ $\overline{WR}$:- $\overline{WR}$ is low 8086 is writing data into memory & an I/o device.

→ HOLD input, HLDA output :-

HOLD = 1 → the another master (DMA) is requesting to take over the system bus. on receiving HOLD signal processor outputs HLDA signal high as an acknowledgment.



→ $\overline{QS_1}$, $\overline{QS_0}$ (output) :- These two output signals reflect the status of the instruction queue.

$\overline{QS_1}$	$\overline{QS_0}$	status
0	0	No operation (idle)
0	1	First byte of an opcode
1	0	Queue is empty
1	1	Subsequent byte of an opcode

→ $\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ (output).

These three status signal indicates the type of transfer to be take place.

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	machine cycle
0	0	0	Interrupt Acknowledge
0	0	1	I/O Read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Instruction fetch
1	0	1	memory read
1	1	0	memory write
1	1	1	Inactive.

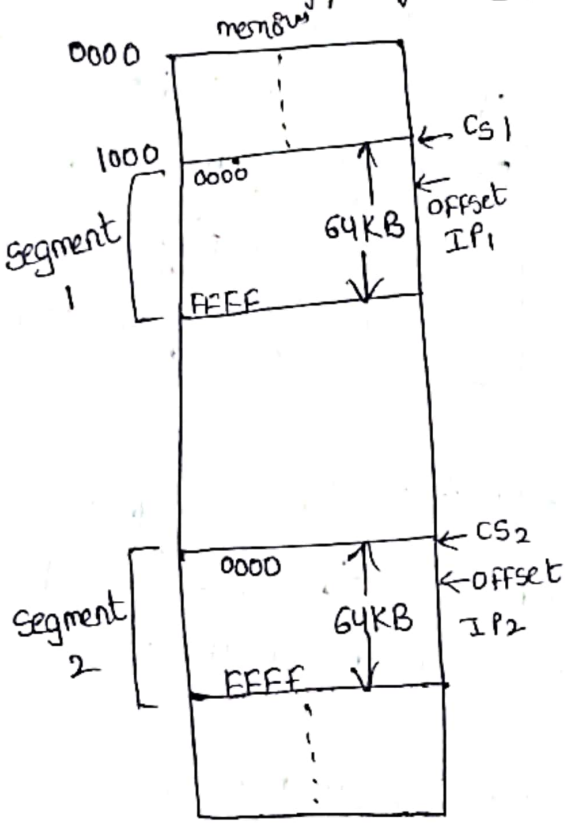
→ $\overline{LOCK}$:- This signal indicates the bus is not to be used by another processor.

→ $\overline{RQ}/\overline{GT_1}$ and $\overline{RQ}/\overline{GT_0}$:- In the maximum mode, HOLD and HLDA pins are replaced by $\overline{RQ}$ (Bus Request) $\overline{GT_0}$ (Bus Grant) $\overline{RQ}/\overline{GT_0}$ has higher priority than $\overline{RQ}/\overline{GT_1}$

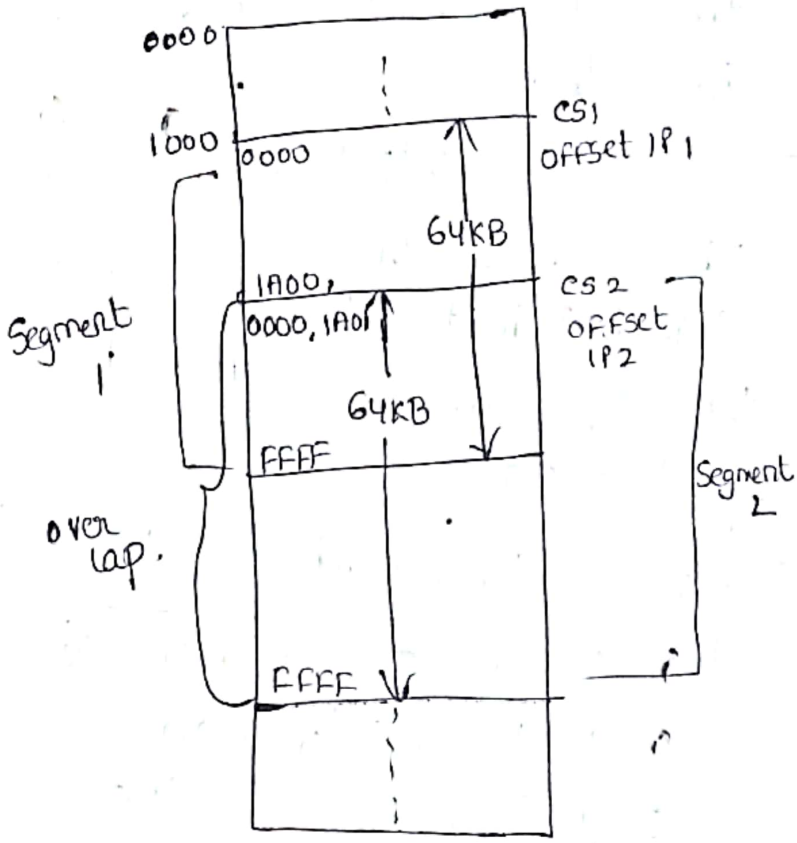
memory Segmentation :-

The memory in an 8086/8088 based system is organised as segmented memory. In this, scheme the 1MB of memory (physically) can be divided into 16 segments. each of 64K bytes size. The addresses of the segments may be assigned as 0000H to F000H. The offset address values are from 0000H to FFFFH. So that the physical addresses range from 00000H to FFFFFH

In the above said case, the segments are called non-overlapping segments.



Non-overlapping segments.



overlapping segments.

In some cases, however the segments may be overlapping. Suppose a segment starts at a particular address and its maximum size is 64K Bytes. But if another segment starts before this 64K bytes locations of the first segment the two segments are said to be overlapping segments. The area of memory from the start of the second segment to the possible end of the first segment is called an overlapped segment area.

In the overlapped area locations physical address

$= CS_1 + IP_1 = CS_2 + IP_2$.

Same physical address generated from two different sets of segment and offset addresses.

Advantages :- 16-bit size, data and code protection.

1.6. MINIMUM MODE CONFIGURATION OF 8086 :

- The min. mode operation is selected by connecting the pin MN/$\overline{MX}$ to + 5 V
- The 8086 is operated in minimum mode in simple systems with a single CPU
- In min. mode operation, all the control signals are generated by CPU
- It is least expensive way to operate and the operation is similar to 8085A.
- Min. Mode signals → $\overline{INTA}$, ALE, $\overline{DEN}$, DT/$\overline{R}$, M/$\overline{IO}$, $\overline{WR}$, HLDA, HOLD

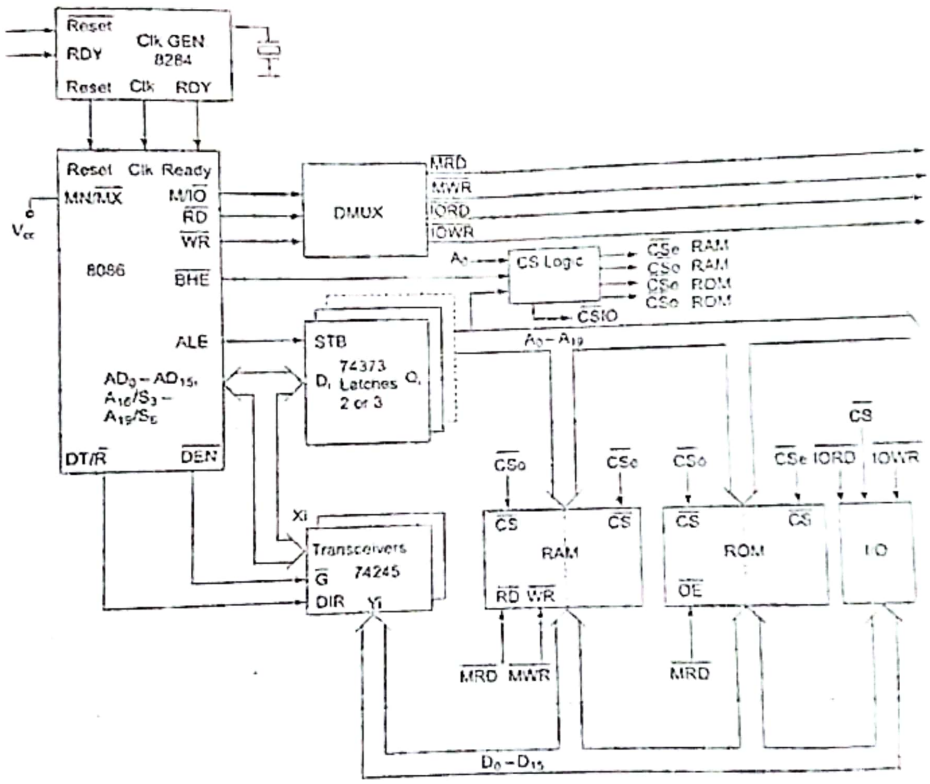


Fig.1.13 *Minimum Mode 8086 System*

Clock generator :

- It generates a CLK of frequency 5 MHz
- It provides the basic timing for µp and bus control activity
- It also synchronizes some external signals with the system clock

Address Latches :

- Latches are used for de-multiplexing of $AD_0 - AD_{15}$, $A_{16}/S_3 - A_{19}/S_6$ and $\overline{BHE}$ / S_7
- Latches are used for the separation of address and data lines
- Address latch enable (ALE) signal is used to enable the Latches

Transceivers/ Bi-directional data buffers

- The Bi-directional data buffers (transmitters/receivers) are used to maintain proper signal quality. i.e., to increase the fan-out of the system.
- The $\overline{DEN}$ signal is used to enable the bi-directional data buffers
- The DT/$\overline{R}$ signal controls the flow of data through data buffers

MINIMUM MODE SIGNALS:

$\overline{INTA}$ (pin-24) :

- The interrupt acknowledge signal is a response to the INTR.
- It indicates the recognition of an Interrupt Request.

ALE (pin-25) :

- The address latch enable signal is used to indicate the presence of valid address information on multiplexed bus (AD_0 - AD_{15})
- This signal is used to enable the Address Latches.
- The address latches are used to separate the address and data lines.(demultiplexing)

$\overline{DEN}$ (pin-26) :

- The data buffers enable signal is used to enable the bi-directional data bus buffers.
- The data bus buffers or transceivers (transmitters/receivers) are used to maintain proper signal quality.

DT/$\overline{R}$ (pin-27) :

- The data transmit / receive signal is used to indicates the direction of data flow through the data bus buffers.

  DT/$\overline{R}$ = 1 for transmitting data

  DT/$\overline{R}$ = 0 for receiving data

M/$\overline{IO}$ (pin-28) :

It selects either Memory or I/O operation

M/$\overline{IO}$= 1 selects memory operation

M/$\overline{IO}$= 0 selects I/O operation

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

$\overline{WR}$ (pin-29) :

This control signal is active whenever the processor is writing data to Memory or I/O

HOLD & HLDA (pins 31 & 30):

- These two signals are used in DMA operation
- The HOLD input indicates the processor that other bus master (DMA controller) is requesting for the use of system bus.
- When HOLD =1, the µp stops the normal program execution and places address, data & control buses at high-impedance state and sends HLDA signal to the DMA controller.
- The HLDA signal indicates that the processor has accepted the HOLD request and the gain control of the system bus is transferred to the DMA controller.
- During HLDA =1 , the DMA controller is the master of the system bus.
- After removal of HOLD request, the HLDA becomes Low.

Timing diagrams for Min. mode

- The working of microprocessor based system can be explained with the help of Timing diagrams.
- The timing diagrams provide the information about the various conditions of the signal while a machine cycle is executed.

T- State : It is one cycle of the clock (clock period)

Machine cycle / Bus cycle :

- The group of T-States required for a basic bus operation is called as Machine cycle
- The microprocessor uses bus cycles for memory and I/O operations

  Opcode fetch cycle
  Memory Read cycle
  Memory Write cycle
  I/O Read cycle
  I/O Write cycle

Instruction cycle:

- The time required for the microprocessor to fetch, decode & execute an instruction is called an Instruction cycle.
- An instruction cycle consists of one (or) more bus cycles.

The timing diagram for Memory Read cycle in Minimum mode is shown in figure.

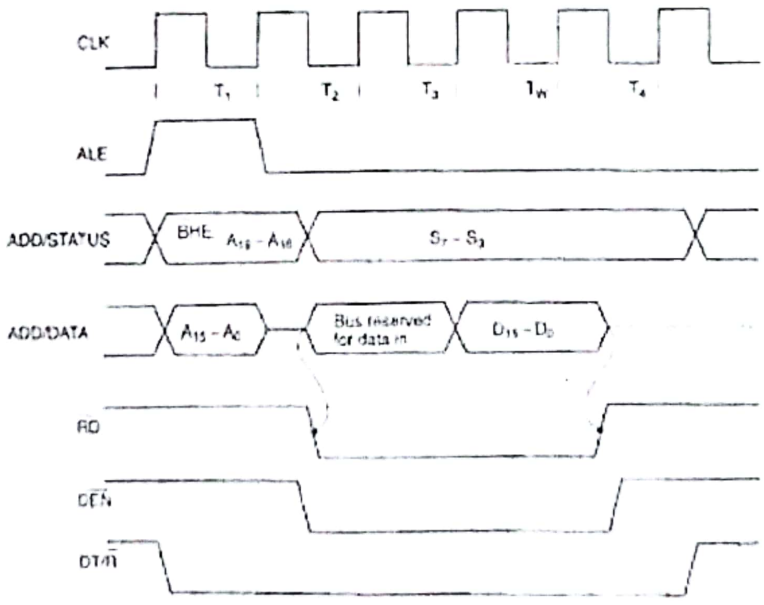


Fig 1.9(a) *Read Cycle Timing Diagram for Minimum Mode*

During the time period T_1

- The micro processor sends out 20 bit address.
- Enables ALE Signal
- Issues $M/\overline{IO} = 1$ for memory operation
- Issues $DT/\overline{R} = 0$ for data reception

During the time period T_2

- The address information is removed from multiplexed bus and status signals are placed. However A0-A19 remain available as they were latched during T_1
- Enables $\overline{RD}$ signal
- Enables $\overline{DEN}$ signal

During the time period T_3

- The microprocessor checks the READY line. If READY =1, the μP reads the data from data bus. Otherwise, a WAIT state T_W is inserted

During the time period T_4

- All the control signals are deactivated to start the next cycle.
- $\overline{DEN}$, $DT/\overline{R}$, $M/\overline{IO}$ and $\overline{RD}$ signals are deactivated

1.7. MAXIMUM MODE CONFIGURATION OF 8086 :

- The max. mode operation is selected by connecting the pin MN/MX to GND
- The 8086 is operated in max. mode in multi-processor system with more than one processor.
- **In max. mode, the control signals are generated by external Bus-controller 8288.**
- The maximum mode operation is used only when the system contains arithmetic co-processor such as 8087 numeric co-processor.
- Max. Mode signals → QS_1, QS_0, $\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$, $\overline{LOCK}$, $RQ/\overline{GT_1}$, $RQ/\overline{GT_0}$

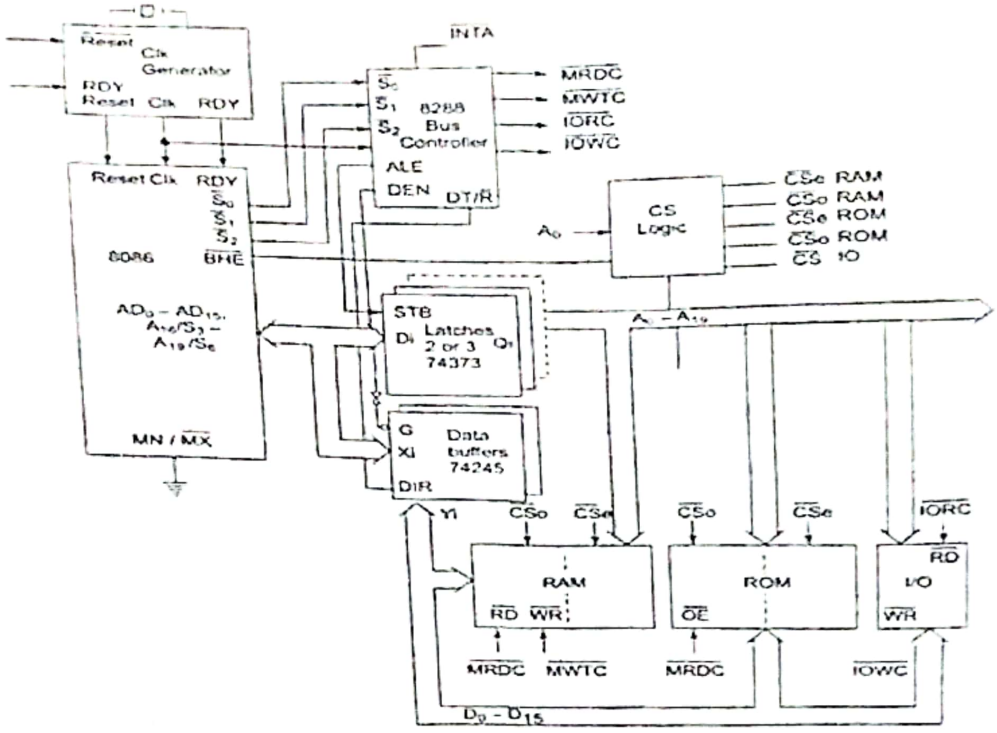


Fig. 1.15 *Maximum Mode 8086 System*

MAXIMUM MODE SIGNALS:

QS_1 & QS_0 (pins-24&25):

The Queue status pins indicate the status of Instruction Queue of 8086 processor.

QS_1	QS_0	Queue Status
0	0	No operation
0	1	First byte of Opcode
1	0	Queue is empty
1	1	Next byte of Opcode

$\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$ (pins- 26,27&28):

These status signals indicate the function of the current bus-cycle. i.e., the type of operation being carried out by the processor.

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	Bus cycle
0	0	0	INTA
0	0	1	I/O Read
0	1	0	I/O Write
0	1	1	HALT
1	0	0	Op-code fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	INACTIVE

$\overline{LOCK}$: (pin-29)

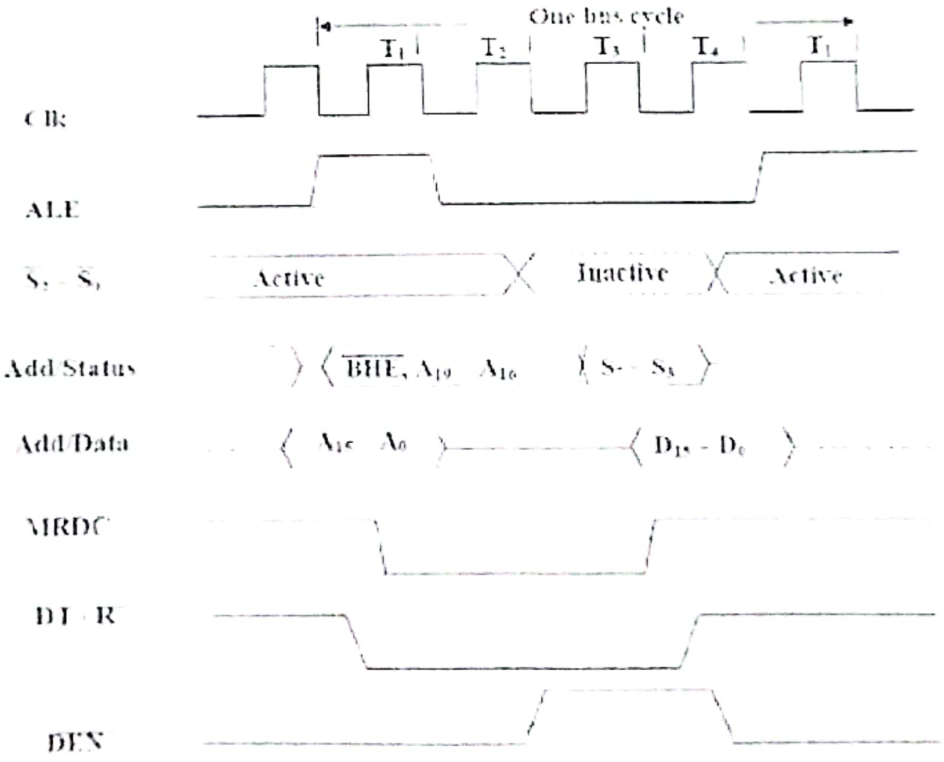
- This pin indicates that the processor is executing a LOCK prefixed instruction and the System bus is not to be used by another bus-master.
- This pin is used in multi-processor system to prevent other Bus masters from taking the control of System bus during the execution of a critical instruction.

$RQ/\overline{GT_1}$, $RQ/\overline{GT_0}$ (pins -30 & 31) :

- The Request/Grant pins are used to request a DMA action during Maximum mode operation.
- These are bidirectional and used to request and grant the DMA operation.

Timing diagrams for Max. mode

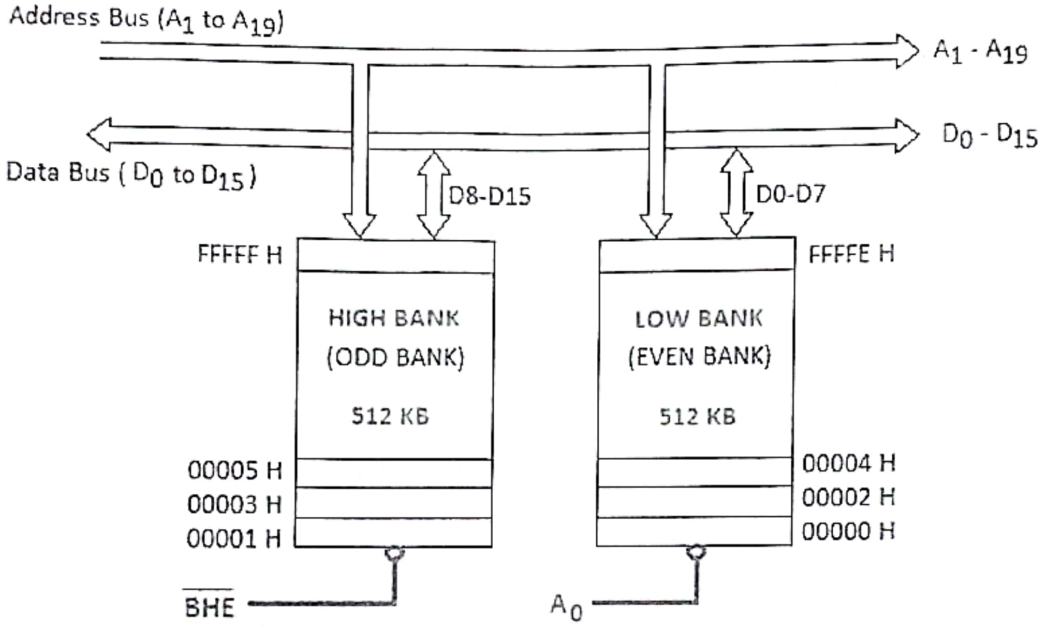
Op-code fetch cycle
Memory Read cycle
Memory Write cycle
I/O Read cycle
I/O Write cycle



Memory Read Timing in Maximum Mode

1.8. PHYSICAL MEMORY ORGANIZATION AND MEMORY BANKS ACCESSING

- The Physical memory of 8086 is divided into TWO banks **for accessing 16-bit numbers**. Each bank size is 512 K bytes.
- The data bus (D_0 - D_7) is connected to LOW bank /EVEN bank.
- The data bus (D_8 - D_{15}) is connected to HIGH bank/ ODD bank.
- Both Banks are enabled at a time for 16-bit operations



- Address lines (A_1-A_{19}) are used to select a location within the bank
- The LOW bank is selected by A_0
- The HIGH bank is selected by $\overline{BHE}$ signal.
- Both Banks are enabled at a time for 16-bit operations.
- Note that the address must be EVEN for 16-bit access.

$\overline{BHE}$ A_0	*Bank Selected*
0 0	Both banks are enabled for 16-bit operation
0 1	HIGH bank is enabled
1 0	LOW bank is enabled
1 1	No banks are enabled

1.9. INTERRUPTS OF 8086 & INTERRUPT VECTOR TABLE

- Interrupt is an event that causes the µp to stop the normal program execution.
- The µp services the interrupt by executing a subroutine called Interrupt Service Routine
- After executing ISR, the control is transferred back again to the main program.

There are 3 sources of interrupts for 8086

Hardware Interrupts	→	signal applied to NMI, INTR pins
Software Interrupts	→	by executing INT instructions
Error in program execution	→	Divide by Zero, Overflow, etc

Hardware Interrupts :

These interrupts occur as signals on the external pins of the microprocessor. 8086 has two pins to accept hardware interrupts, NMI and INTR.

NMI (Non Maskable Interrupt)

- It is a positive going edge triggered Non-Maskable Interrupt
- It cannot be disabled by using software.
- This interrupt has highest priority than INTR.
- On receiving an interrupt on NMI line, the microprocessor executes INT 2
- The µp obtains the ISR address from location 2 x 4 = 00008H from the IVT

INTR (Interrupt Request)

- It is a **level triggered** maskable Interrupt Request.
- It can be disabled by using software i.e., the processor will get interrupted only if IF=1. Otherwise the processor will not get interrupted even INTR is active.
- It is masked by making IF = 0 by software through CLI instruction.
  It is unmasked by making IF = 1 by software through STI instruction.
- It is a non-vectored interrupt. On receiving an interrupt on INTR line, the µp issues INTA pulse to the interrupting device. Then the interrupting device sends the vector number 'N' to the processor. Now the µp multiplies N x 4 and goes to the corresponding location in the IVT to obtain the ISR address.

Software Interrupts :

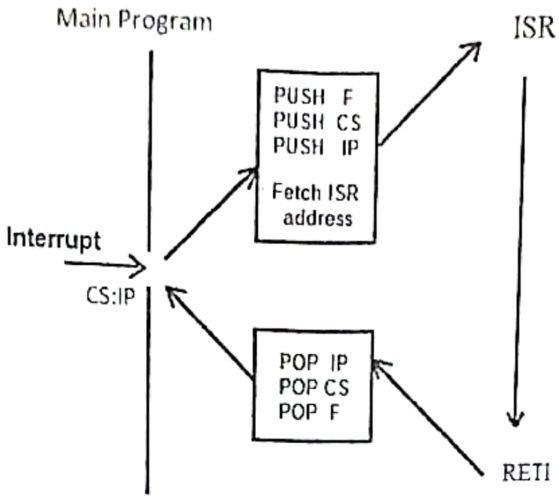
These interrupts are caused by writing the software interrupt instruction **INT N** where 'N' can be any value from 0 to 255 (00H to FFH). Hence all 256 interrupts can be invoked by software.

Error conditions :

The 8086 is interrupted when some special conditions occur while executing certain instructions in the program.

Example: Divide error, Overflow etc

Processing of Interrupts



When an interrupt is enabled, the 8086 µp performs the following actions

- ✓ It completes the execution of current instruction
- ✓ It pushes the Flag register (PSW) on to the stack
- ✓ The contents of CS and IP are pushed to stack **(return address)**
- ✓ It issues an Interrupt acknowledge (INTA)
- ✓ It computes the vector address from the type of interrupt
- ✓ The IP & CS are loaded with ISR address, which is available in Interrupt Vector Table
- ✓ Then the control is transferred to ISR and starts to execute the interrupt service routine at that address.

The code to handle an interrupt is called an *interrupt handler* or *Interrupt Service Routine* (ISR). An interrupt service routine must always finish with the special instruction IRET *(return from interrupt)*, which performs the following actions.

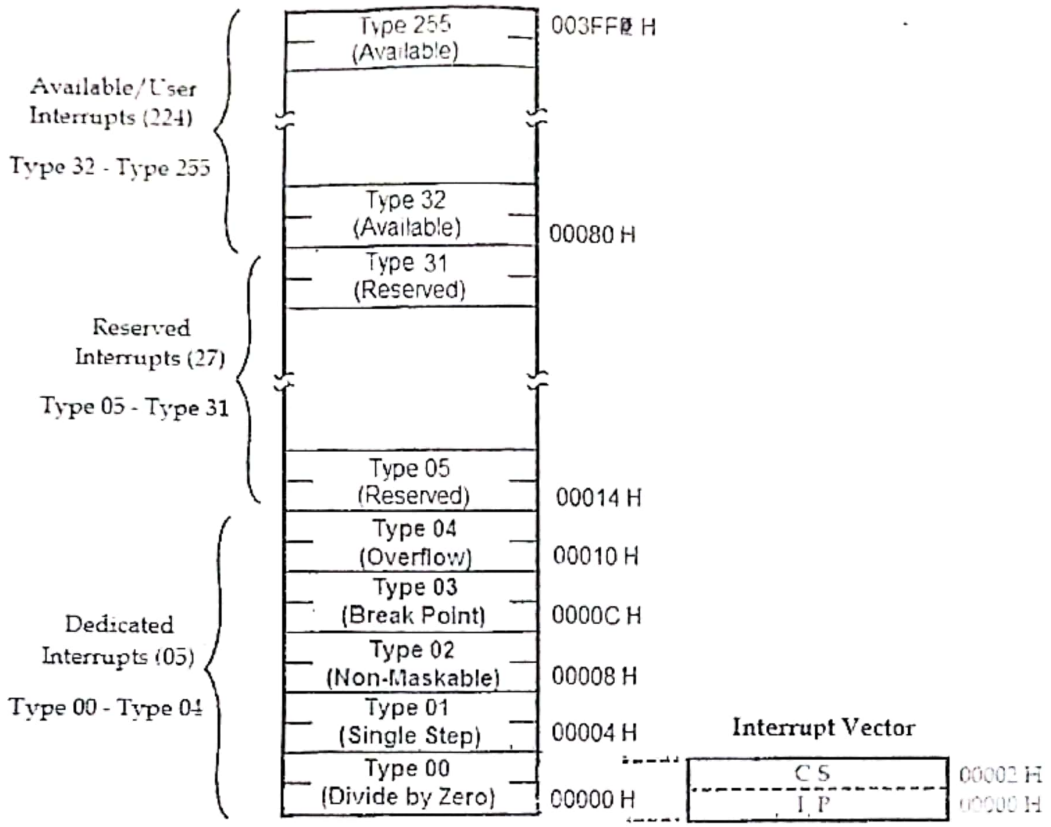
- ✓ The IP and CS are loaded with **return address** from stack
- ✓ The Flag register is popped from the stack
- ✓ The program execution returns to main line, where it was interrupted.

Interrupt Vector Table:

When the interrupt is enabled, the present IP and CS are pushed on to the stack and loaded with the address of ISR, which is available in Interrupt vector table.

Interrupt Vector → It is a memory block which contains address of ISR
The size of Interrupt vector is 4-bytes (to store CS and IP of ISR)

Interrupt Vector Table → It is a memory block which contains interrupt vectors
Since there are 256 types of interrupts, there are 256 interrupt vectors
Hence, the size of Interrupt Vector Table is 4*256 = 1024 bytes



8086 Interrupt Vector Table

In 8086 interrupt system, the first 1 KB memory from 00000 H to 003FF H is reserved for storing the starting addresses of ISRs. This block of memory is called as IVT.

- The Interrupt vector table contains 256 interrupt vectors
- **The interrupt vector contains IP and CS values of ISR**
- The address of interrupt vector can be calculated by multiplying the TYPE with 4
- *For example,* The interrupt vector address for Type-02 interrupt = 02 H*4= **00008 H**

- **The 8086 uses 256 types of interrupts – Type 00 to Type 255**
  These interrupts are classified as
  (A) Dedicated Interrupts (Type 00 to Type 04) : 05
  (B) Reserved Interrupts (Type 05 to Type 31) : 27
  (C) Available/User Interrupts (Type 32 to Type 255) : 224

(A) The dedicated interrupts (Type 0 – Type 4) are

(i) INT 0 : Divide Error
(ii) INT 1 : Single step execution (TRAP)
(iii) INT 2 : Non-Maskable Interrupt (NMI)
(iv) INT 3 : Break Point
(v) INT 4 : Overflow

(i) **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: Type 0 x 4 = 00000H in the IVT.

(ii) **INT 1 (Single Step)-**

- The µp executes this interrupt after every instruction if the TF =1
- It puts µp in single stepping mode i.e. the microprocessor will get interrupted after executing every instruction. This is very useful during debugging.
- This ISR generally displays contents of all registers.
- Its ISR address is stored at location: Type 1 x 4 = 00004H in the IVT.

(iii) **INT 2 (Non Maskable Interrupt)-**

- The microprocessor executes this ISR in response to an interrupt on the NMI (Non mask-able Interrupt) line.
- Its ISR address is stored at location: Type 2 x 4 = 00008H in the IVT.

(iv) **INT 3 (Breakpoint Interrupt)-**

- This interrupt is used to cause breakpoints in the program.
- It is useful in debugging large programs
- This ISR generally displays contents of all registers
- Its ISR address is stored at location: Type 3 x 4 = 0000CH in the IVT

(v) **INT 4 (Overflow Interrupt)**

- This interrupt occurs if the overflow flag is set
- It is used to detect overflow error in signed arithmetic operations.
- Its ISR address is stored at location: Type 4 x 4 = 00010H in the IVT

(B) Reserved interrupts (Type 5 to Type 31)

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

(C) Available interrupts (Type 32 to Type 225)

- 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 microprocessor from location N x 4 in the IVT

Problem:

The contents of memory location 0000:008C are given below

0000:008C → 12, 34, 56, 78, 90, 92

(a) What is the interrupt vector address for Type-23H interrupt?

(b) Find the address of ISR corresponding to INT 23H

(c) For which type of interrupt, the interrupt vector address is 0000:00C8H

Sol: (a) Interrupt vector address = Type * 4

For Type-23H interrupt, Interrupt vector address = 23*4 = 8C H

Interrupt vector address for Type-23H is 0000:008C H

(b) The address of ISR for Type-23H is available at 0000:008C H

ISR IP is available at memory 0000:008C H = 3412 H

ISR CS is available at memory 0000:008E H = 7856 H

Address of ISR = CS: IP = 7856:1234

(c) Interrupt vector address = Type * 4

00C8 = Type * 4

Type = 00C8 / 4 = 32H

Hence the Type of Interrupt is 32H