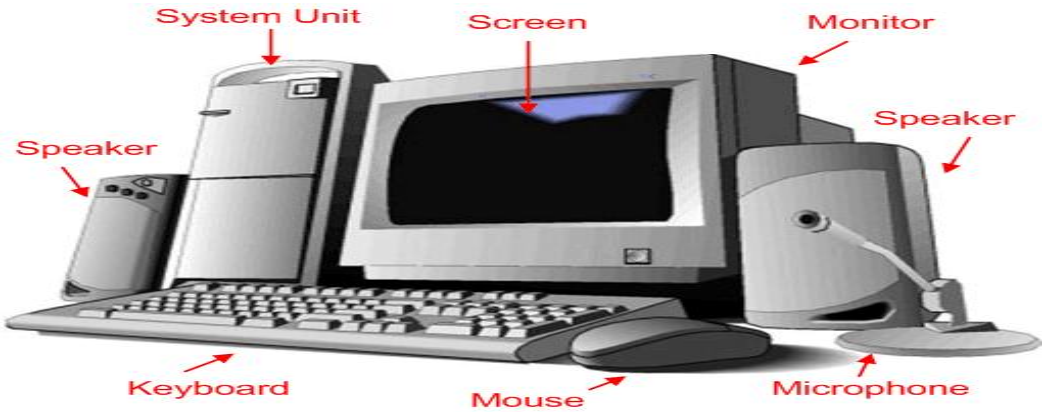
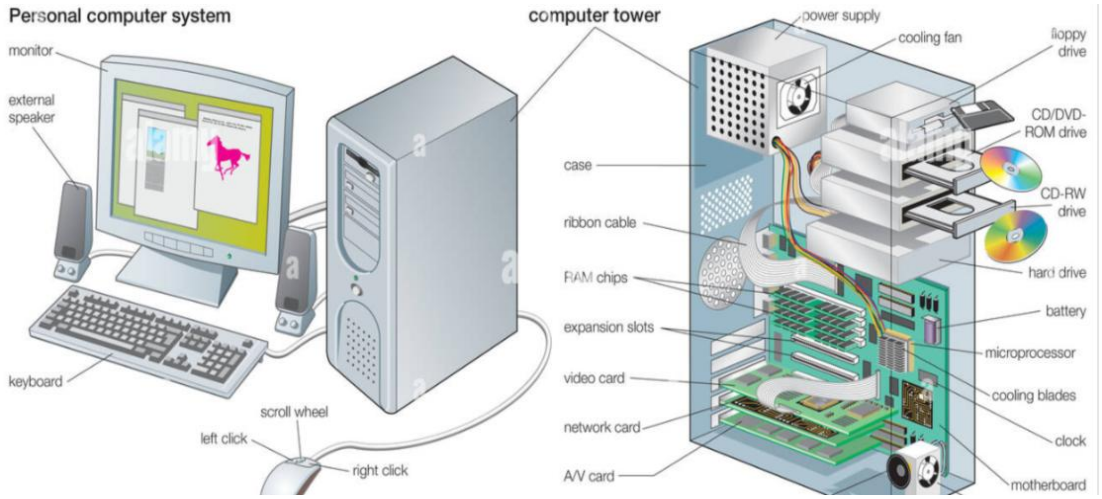
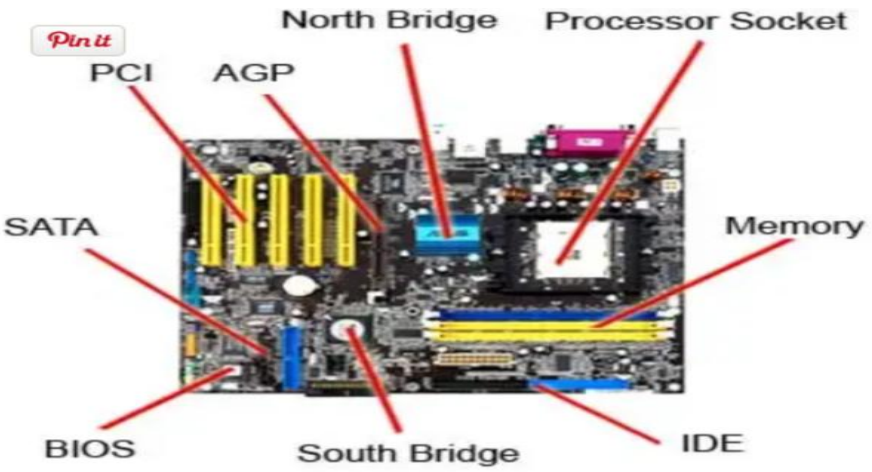
Computer:-

A computer is **an electronic device that manipulates information, or data**. It has the ability to store, retrieve, and process data.



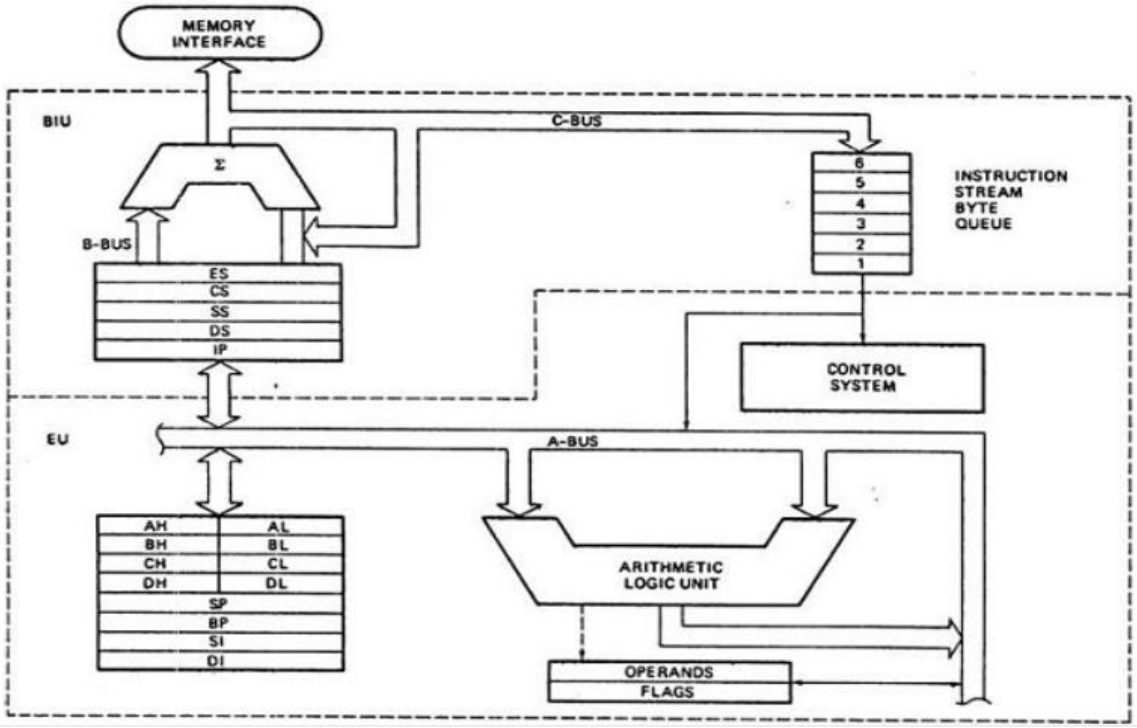




Feature of Microprocessor 8086

- It is upgraded microprocessor from 8085. 8086 has following features:
  - 8086 has 16 bits ALU. [Older version 8085 has 8 bits of ALU.] So, operating speed 8086 have increased significantly well.
  - 8086 has 16 bits of Data bus. So in one machine cycle it can transfer 2 bytes. [Older version 8085 has 8 bits of data bus.]
  - 8086 has 20 bits of Address bus. With 8086 we can interface total memory by 2^{20}= 1MB [Older version 8085 has 16bits Address bus.]
  - 8086 has two internal hardware units. BIU – Bus Interface Unit & EU – Execution Unit.
  - 8086 supports pipelining. So Multiple instructions can be executed in parallel to increase execution speed of microprocessor.
  - Four 16 bits general purpose registers (AX, BX, CX & DX). We can use it with 8 bits also (AH, AL, BH, BL, CH, CL, DH, DL). Two 16 bits Index registers (SI and DI). Two 16 bits stack pointers (SP and BP).
  - 8086 supports memory segmentation. (CS, DS, ES and SS)
  - 16 bit flag register.
  - Two hardware modes : Maximum Mode and Minimum Mode.
  - 8086 provides memory banks. So 1MB is bisected into two bank of 512KB, Lower bank (even) and higher bank (Odd).

ARCHITECTURE OF 8086 MICRO PROCESSOR:



Architecture is Divided into two units

1. **BIU(Bus Interface Unit)**
2. **EU(Execution Unit)**

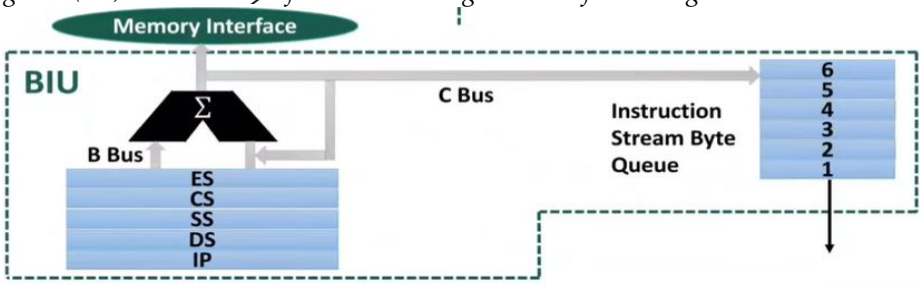
Bus Interface Unit of Microprocessor 8086

- BIU is responsible for establishing communications with external peripheral devices and memory via system bus.
- Main Purpose of BIU serves is as followed
  - It fetches the instructions from Memory
  - It reads data from IO and Memory
  - It writes data into IO and Memory
  - It provides the address relocation facility
- BIU contains three main parts
  - Segment Registers
  - Instruction Queue
  - Instruction Pointer
- Segment Register of BIU
  - The 8086 MP has the capability of addressing 1MB memory, which is divided into 16 local segments.
  - Each segment contains 64KB memory
  - But any instant 8086 works with only four 64KB segments.
- Each segment associated with segment register
  - Code Segment Register CS – 16bits
  - Data Segment Register DS – 16bits
  - Stack Segment Register SS – 16bits
  - Extra Segment Register ES – 16bits
- Instruction Queue
  - BIU prefetches six instruction bytes in advance from memory.
  - The prefetched instructions are stored in a group of high speed registers known instruction queue.
  - This Instruction queue works on FIFO order.
  - BIU and EU works in parallel.
  - The simultaneous operations of BIU and EU is possible only when the EU does not require the system bus.
  - The process of fetching the next instruction in advance while the EU is executing the current instruction is pipelining.
- Segment Register of BIU is used to store the starting address of Memory segment.
- BIU generates 20 bits address using segment register and Offset pointer registers.

  PA = SR X 10H + OP
- Instruction Pointer of BIU
  - IP holds the address of next instruction which is to be executed next.
  - It contains the OFFSET value of next Address.

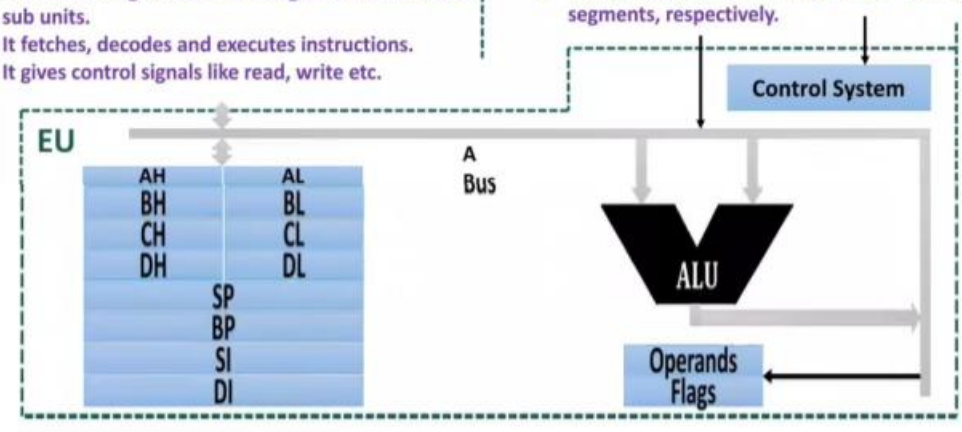
The four segment registers in 8086 are

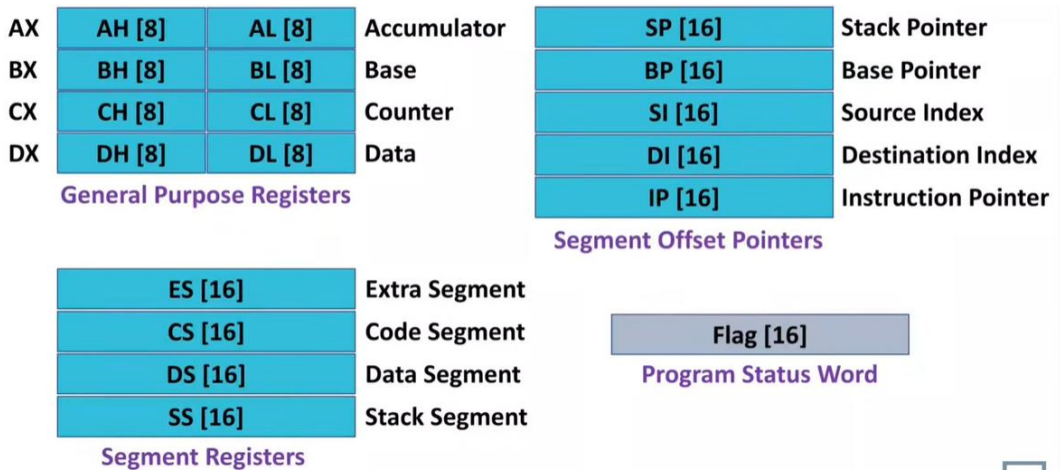
Code segment register (CS) → *defines the starting address of Code segment*
Data segment register (DS) → *defines the starting address of Data segment*
Extra segment register (ES) → *defines the starting address of Extra segment*
Stack segment register (SS) → *defines the starting address of Stack segment*



Execution Unit of Microprocessor 8086

- EU informs BIU from where the next instruction or data to be fetched.
- The EU performs following functions:
  - It picks up the instruction from the instruction queue of BIU.
  - It decode the instructions and then executes the instruction.
  - It updates the status of flag register.
- Control Unit
  - It is controlling and coordinating all the activities of sub units.
  - It fetches, decodes and executes instructions.
  - It gives control signals like read, write etc.
- ALU of EU
  - It performs all the arithmetic and logical operations.
  - Results may be stored in general purpose registers or index registers.
  - It updates Flag register after every instructions.
- General Purpose Registers & Pointer of EU
  - EU have four general purpose registers.
  - These registers are used for accessing data very fast.
  - EU have SP and BP for stack.
  - Index registers are SI and DI for data and Extra segments, respectively.





(i) General purpose registers :

The 8086 has 4- general purpose registers - AX, BX, CX, DX.
All these registers are 16-bit registers used to store 16-bit data
Each register is divided in two 8-bit registers to store 8-bit data
These registers have multiple functions, shown in following Table.

Register	*Name*	*Purpose*
AX	Accumulator	Used to hold the results of some operations like MUL, DIV , Shift, Rotate…etc
BX	Base Index	Used to hold the offset address (Based addressing)
CX	Counter	Used as a counter in LOOP and String instructions
DX	Data Index (or) Extended Accumulator	Used to hold a part of result from MUL & DIV. It is also used to hold the 16-bit I/O port address during I/O operation.

(ii) Index Registers :

- The Index Registers are used to hold the 16-bit offset address of data stored in Data and Extra segments.
- These registers are used in string operations to hold the offset address of source string and destination strings.
  The SI register holds the offset address of source string in Data Segment.
  The DI register holds the offset address of destination string in Extra Segment.
  - Address of source string → DS:[SI]
  - Address of destination string → ES:[DI]

- The directional flag (DF) selects the increment (or) decrement mode for SI and DI registers during String instructions manipulation.

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

(iii) Pointers :

- The Pointers are used to hold the offset address relative to data and stack segments.
- The base pointer (BP) is used to access data in stack segment.
- The stack pointer (SP) is used to hold the address of stack top.

Register	*Name*	*Purpose*
SI	Source Index	It is used in string operations to hold the offset address of source string in Data segment.
DI	Destination Index	It is used in string operations to hold the offset address of destination string in Extra segment.
BP	Base Pointer	It is used to access data the stack segment. It is used to hold the offset address (Based addressing)
SP	Stack Pointer	It is used to hold the address of Stack top

(iv) Flag Register / Program Status Word (PSW)

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.

Conditional Flags / Status Flags :

These flags are Set or Reset according to the condition produced by an instruction execution.
The 6- conditional flags of 8086 are CF, PF, AF, ZF, SF, OF

Control flags:

These flags control the operation of the processor.
The 3- control flags of 8086 are DF, IF, TF.

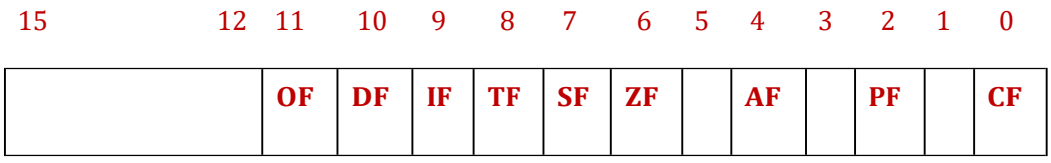


Figure: Flag register of 8086

Carry Flag (CF) : It is set to 1, if a 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.
It is set to 1, if addition of lower nibble generates a carry.

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 registercapacity.

Directional Flag (DF) :

It is used in string operations to select either increment (or) decrement mode for SI and DI registers. DF = 0 selects increment mode and
DF = 1 selects decrement mode.

Interrupt Flag (IF) :

- If IF = 1, all the maskable interrupts are recognized and processed by the CPU.
- Otherwise, the maskable interrupts are ignored and not processed by CPU.

Trap Flag (TF) :

- It is used for debugging purpose
- If TF = 1, the processor enters the 'Single step execution mode' i.e., the 8086 gets interrupted automatically at the end of every instruction execution.

Problem :

The content of registers AL and BL are 79H and 88H respectively. Determine the status of control flags of 8086, after the execution of the instruction ADD AL,BL?

Ans:

```
AL = 79 H   =   0 1 1 1 1 0 0 1   ← AL
BL = 88 H   =   1 0 0 0 1 0 0 0   ← BL
-----------------------------------------
ADD AL, BL  :   0 0 0 0 0 0 0 1   ← AL
-----------------------------------------
```

The status of conditional flags are :

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

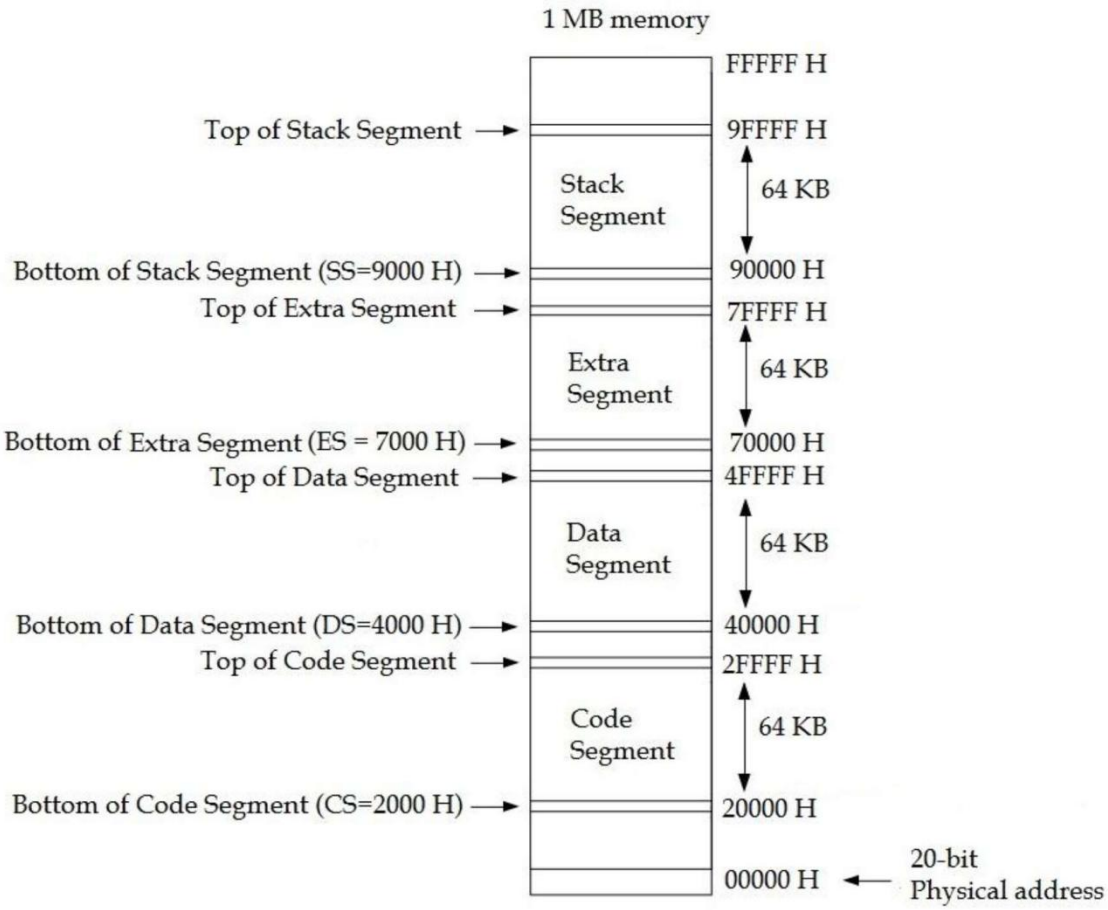
1.1. MEMORY SEGMENTATION

The 8086 uses memory segmentation. In this scheme, the complete 1 MB physical memory is divided into number of logical segments. The size of each segment is 64 KB in size and is addressed by one of the segment registers.

The 64 KB logical segment can be located anywhere in 1 MB memory, but the segment will always start at an address with ZERO's in lowest 4-bits.

Note that the 8086 does not work the whole 1 MB memory at any given time. However it works only with four 64 KB segments within the whole 1MB memory.

The four segment registers in BIU define the starting addresses of the four memory segments with which the 8086 is working at that instant of time. The segment registers in BIU are used to hold the upper 16-bits of starting address of logical segments. The BIU inserts ZERO's for lower 4-bits of 20-bit starting address.



Code Segment :

- The Code segment is used to store the program instruction codes.
- The Code Segment register (CS) is used to hold the upper 16-bits of starting address of the Code segment. The BIU inserts ZERO's for lower 4-bits of 20-bit starting address.
- For example, if CS=2000H then the Code segment starts at 20000H.

Data Segment :

- The Data segment is used to store data variables and constants of the program.
- The Data Segment register (DS) is used to hold the upper 16-bits of starting address of the Data segment.
- Data are accessed from Data segment by an Offset address.
- The SI, DI, BX, BP registers are used to store the offset address for data segment

Extra Segment :

- The Extra segment is an additional data segment.
- It is used in String operations to store the destination string.
- The Extra Segment register (ES) is used to hold the upper 16-bits of starting address of the Extra segment.
- The SI, DI, BX, BP registers are used to store the offset address for data segment

Stack Segment :

- The Stack segment defines the area of memory used for stack.
- Stack is a section of memory where the data are accessed in LIFO manner
- The stack memory is used to store data, address and status information. It is used by CPU to store return address during the execution of procedures and ISRs.
- The Stack Segment register (SS) is used to hold the upper 16-bits of starting address of the Stack segment.
- The Stack Pointer (SP) register holds the address of Stack top.

Addressing a location within a segment (20-bit Physical address calculation) :

The BIU has a Physical Address Generation Circuit.
It generates the 20-bit physical address by adding Segment base to the Offset address.

The segment register defines the segment base address and
a location within a segment can be addressed by *16-bit offset address* as shown in Figure.

For example, if CS= 2000H and IP= 5678H then

Segment base = CS*10 = 20000 H
Offset address = IP = 5678 H

20-bit Physical address = Segment base + Offset = 25678 H

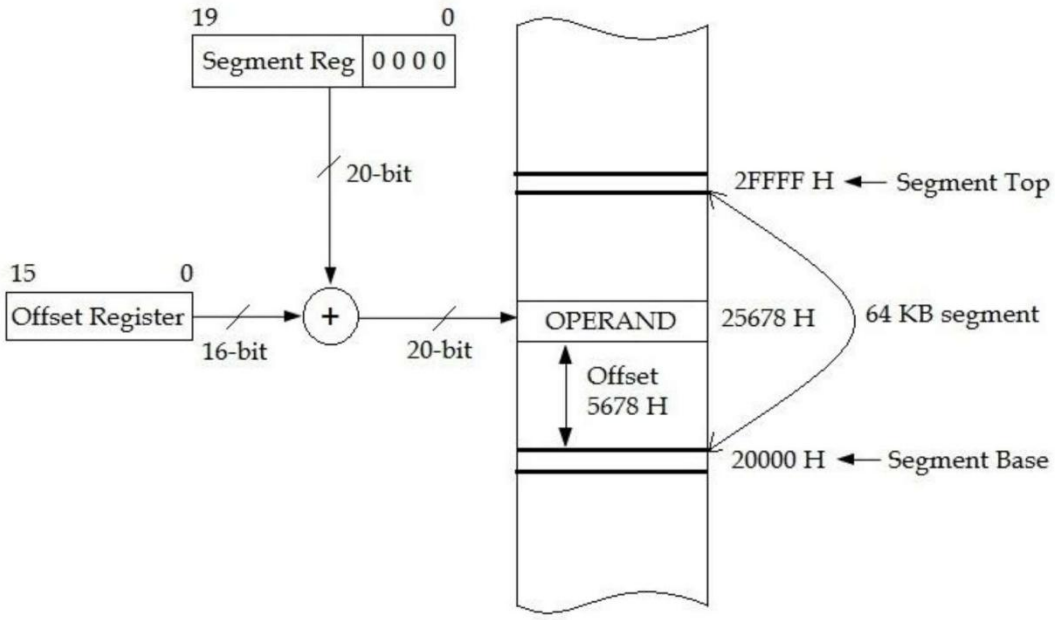


Figure : 20-bit Physical address calculation

The segment register holds the upper 16-bits of starting address of logical segments. The BIU inserts ZERO's for lower 4-bits of 20-bit starting address.

The Offset address is defined as the distance of operand from the Starting address of the Segment. The registers used to store the 16-bit offset addresses for various segments are given in Table.

Segment	*Segment Register*	*Offset Register*
Code Segment	CS	IP
Data Segment	DS	SI (DI/BX/BP)
Extra Segment	ES	DI (SI/BX/BP)
Stack Segment	SS	BP /SP

Advantages of Memory Segmentation :

- Allows to access 1 MB memory with 16-bit address.
- Allows Separate memory areas for program, data and stack
- Provides data and code protection
- Provision for relocation of programs and data
- Provides a powerful memory management mechanism
- Allows the processor to access the data from memory easily and fastly, which increases the speed of operation

1.2. REGISTER ORGANIZATION OF 8086

The Intel 8086 has a powerful set of registers as shown in figure.
The description of all registers is given in section 1.2

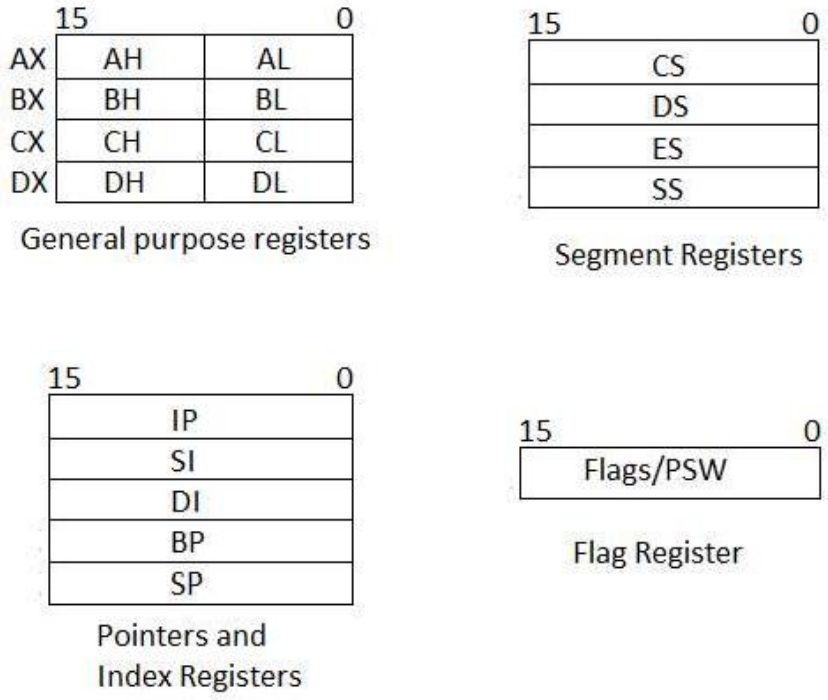


Figure: Register organisation of 8086

1.3. PIN CONFIGURATION OF 8086 :

The pin diagram of 8086 is shown in Figure.
The 8086 consists of 40-pins.
Among these, 21 are multiplexed pins to reduce the number of pins

AD_0 – AD_{15}
A_{16}/S_3 - A_{19}/S_6 and
$\overline{BHE}$ / S_7

The 8086 issues two different sets of signals (Pins 24 to 31) in Min. mode and Max. mode.

Min. Mode → $\overline{INTA}$, ALE, $\overline{DEN}$, DT/$\overline{R}$, M/$\overline{IO}$, $\overline{WR}$, HLDA, HOLD

Max. Mode → QS1, QS0, $\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$, $\overline{WR}$, RQ/$\overline{GT_1}$, RQ/$\overline{GT_0}$

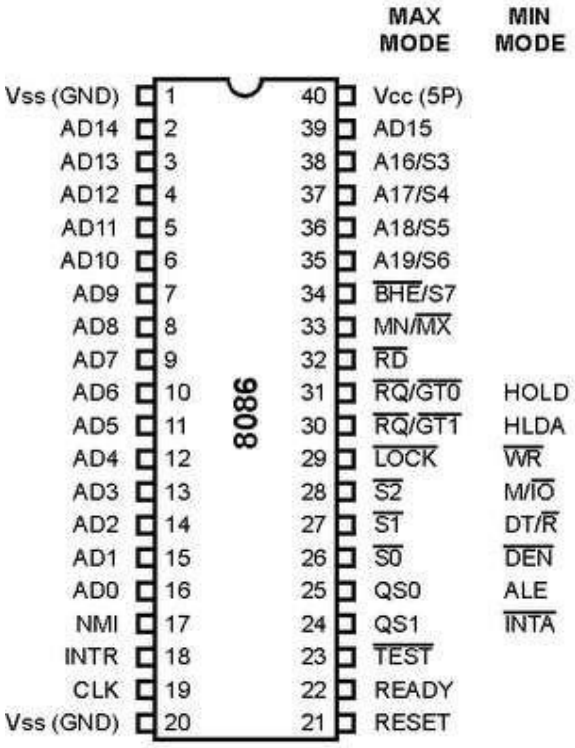


Figure: Pin diagram of 8086

AD_0 - AD_{15} (Address / Data lines) :

- The lower order 16-address lines (A_0 - A_{15}) of 8086 are multiplexed with 16- data lines (D_0 - D_{15}).
- During T_1 state of Bus-cycle, the 8086 sends out address on these lines and during later part of Bus-cycle this multiplexed bus is used as Data bus.
- All the above multiplexed pins are at high-impedance state during DMA operation.

A_{16}/ S_3 - A_{19}/ S_6 (Address/Status lines) :

- The higher order 4-address lines (A_{16} - A_{19}) are multiplexed with Status lines (S_3 - S_6).
- The **S_3 and S_4** shows which segment is accessed during current bus cycle.
- The S_5 indicates the status of IF flag.
- The S_6 is always logic 0 and it is not used.
- All the above multiplexed pins are at high-impedance state during DMA operation.

S4 S3	*Segment*
0 0	Code Segment
0 1	Data Segment
1 0	No Segment
1 1	Extra Segment

$\overline{BHE}$ / S_7 :

- The 1 MB Physical memory of 8086 is divided into two banks **for accessing 16-bitnumbers**. Each bank size is 512 K bytes.
- The Bus High Enable $\overline{BHE}$ signal is used to enable the higher order data bus (D_8 - D_{15}) connected to HIGH BANK.
- The status signal S_7 is used by arithmetic co-processor 8087 to determine whether the CPU is 8086 (or) 8088.

$\overline{BHE}$	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

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.
- It is a vectored interrupt and the vector address of NMI is 0000:0008 H.

.

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 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 5 MHz, 10 MHz, 8 MHz.

RESET :

- It forces all the registers to a predefined values and microprocessor gets reset.
- When Reset is active DS, ES, SS, IP and FLAG registers are initialized to 0000H and CS is initialized to FFFF H.
- After Reset, the processor starts execution from FFFF0 H.

READY:

- A slow peripheral (or) memory device can be connected to microprocessor through READY line. It is used to insert wait states in bus cycles as needed to interface with slow memory & I/O
- It is used by the MPU to sense whether the peripheral is ready to transfer data or not

  If READY =1, peripheral is ready to transfer data.

  If READY=0, the processor WAITS until it goes to HIGH

$\overline{TEST}$:

- This input is examined by WAIT instruction.
- The 8086 enters into WAIT state after the execution of WAIT instruction.

  If $\overline{TEST}$ = 0, the WAIT instruction functions as NOP

  If $\overline{TEST}$ =1, the processor waits until $\overline{TEST}$ = 0.
- If the co-processor has finished its work then it makes $\overline{TEST}$= 0.

$\overline{RD}$:

The **Read Control Signal** is active whenever the processor is ready to read data from Memory (or) I/O.

MN/ $\overline{MX}$:

OPERATING MODES OF 8086/8088:

- The 8086 can be operated in two modes – Minimum mode and Maximum mode.
- The pin MN/$\overline{MX}$ is used to select the Min. (or) Max. mode of operation.

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

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

  - The pins (24-31) issues two different set of signals – for minimum & maximum mode operations. *These pins are described below:*

Note : The Min.mode pins and Max.mode pins are described in sections 1.6.and 1.7.

Minimum mode:

- The min. mode operation is selected by connecting the pin MN/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

Maximum mode:

- The max. mode operation is selected by connecting the pin MN/MX to GND
- The 8086 is operated in maximum 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/$\overline{GT_0}$

1.4. MINIMUM MODE CONFIGURATION OF 8086 :

- The min. mode operation is selected by connecting the pin MN/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

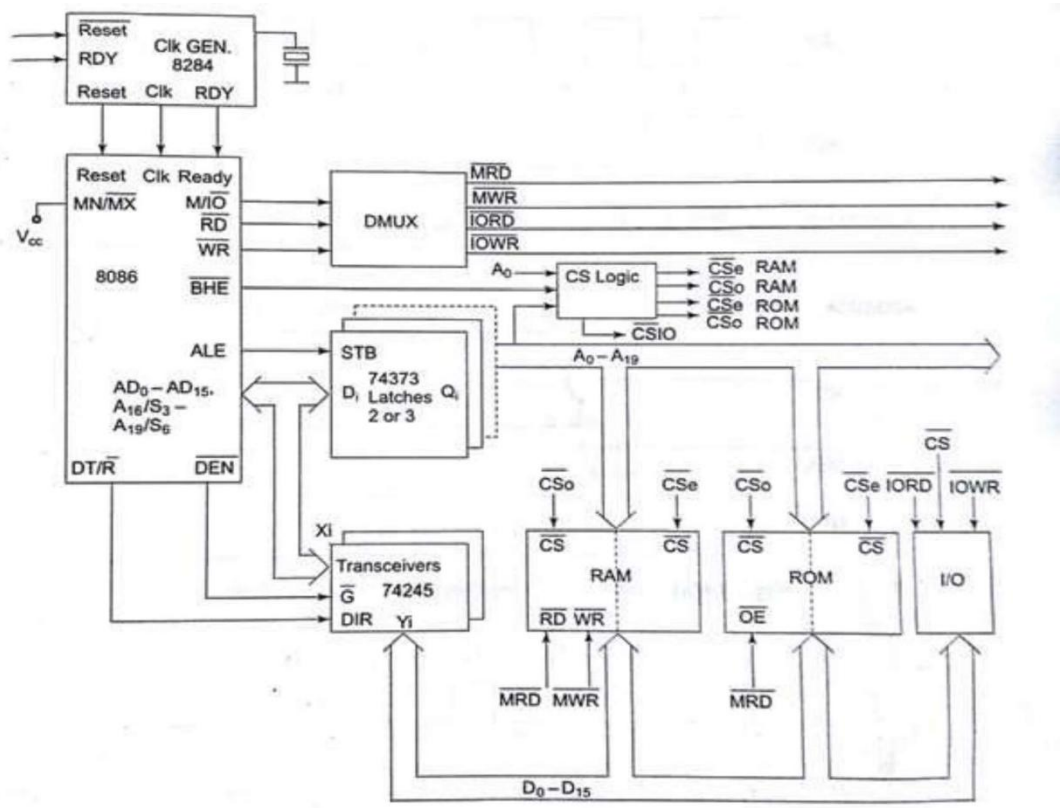


Fig.1.13 *Minimum Mode 8086 System*

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:

$\overline{\text{INTA}}$ (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)

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

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

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

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

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

It selects either Memory or I/O operation

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

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

M/IO	RD	WR	Operation
1	0	1	Memory Read
1	1	0	Memory Write
0	0	1	I/O Read
1	1	0	I/O Write

$\overline{\text{WR}}$ (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 μ 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.

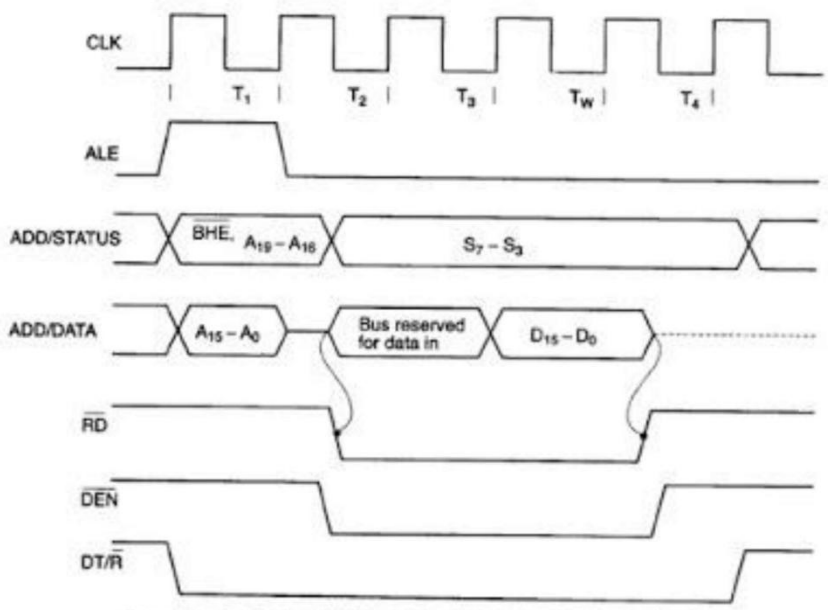


Fig. 1.9(a) *Read Cycle Timing Diagram for Minimum Mode*

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 areplaced. However A0-A19 remain available as they were latched during T_1
- Enables $\overline{RD}$ signal
- Enables $\overline{DEN}$ signal

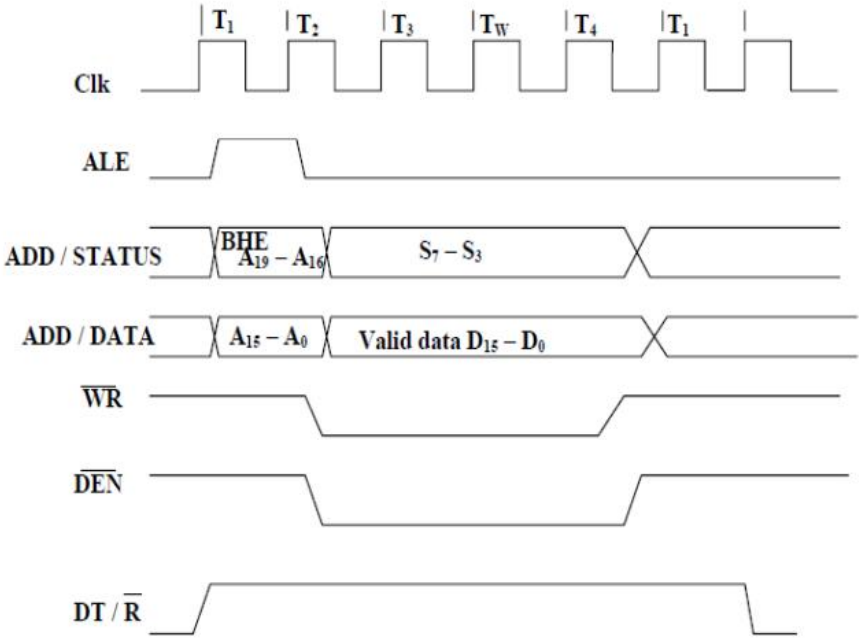
During the time period T_3

- 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

The timing diagram for Memory Write cycle in Minimum mode:



1.5. 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, $\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$, $\overline{LOCK}$, $RQ/\overline{GT_1}$, $RQ/\overline{GT_0}$

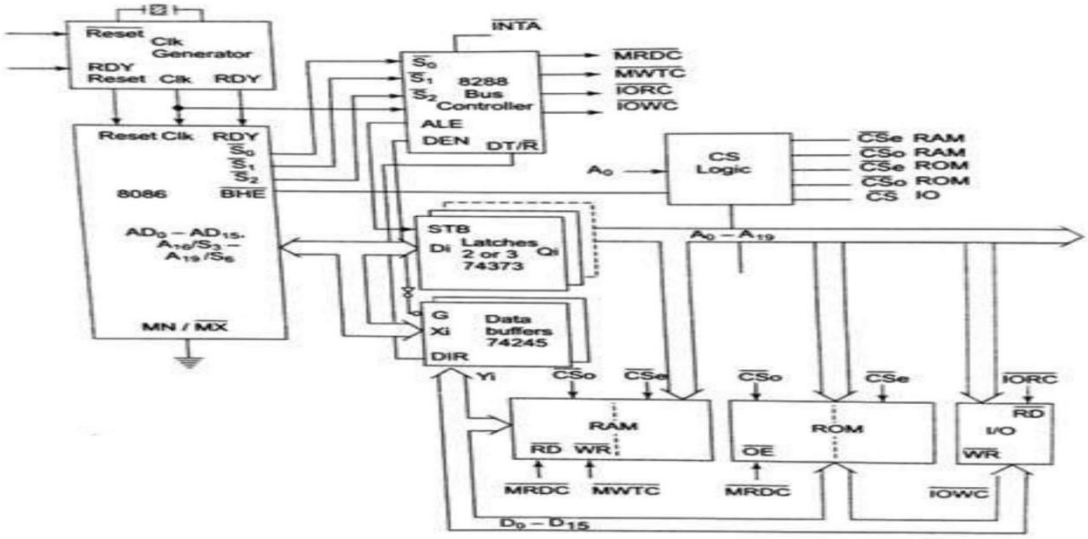


Fig. 1.15 *Maximum Mode 8086 System*

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)

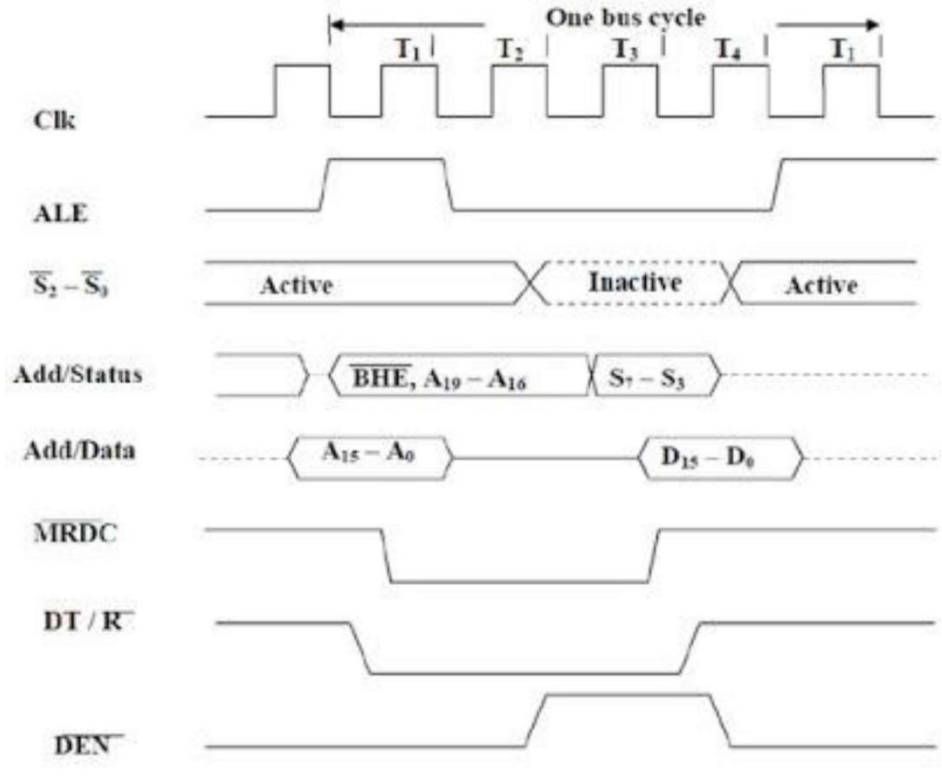
- This pin indicates that the processor is executing a LOCK prefixed instruction and the System bus is not to be used by another bus-master.
- This pin is used in multi-processor system to prevent other Bus masters from taking the control of System bus during the execution of a critical instruction.

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

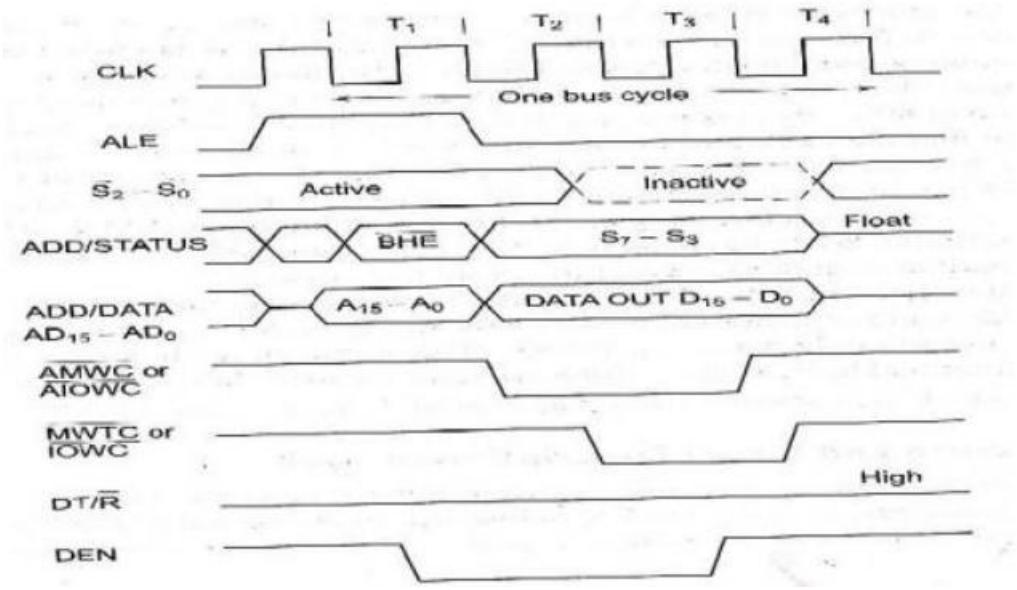
- The Request/Grant pins are used to request a DMA action during Maximum mode operation.
- These are bidirectional and used to request and grant the DMA operation.

Timing diagrams for Max. mode

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

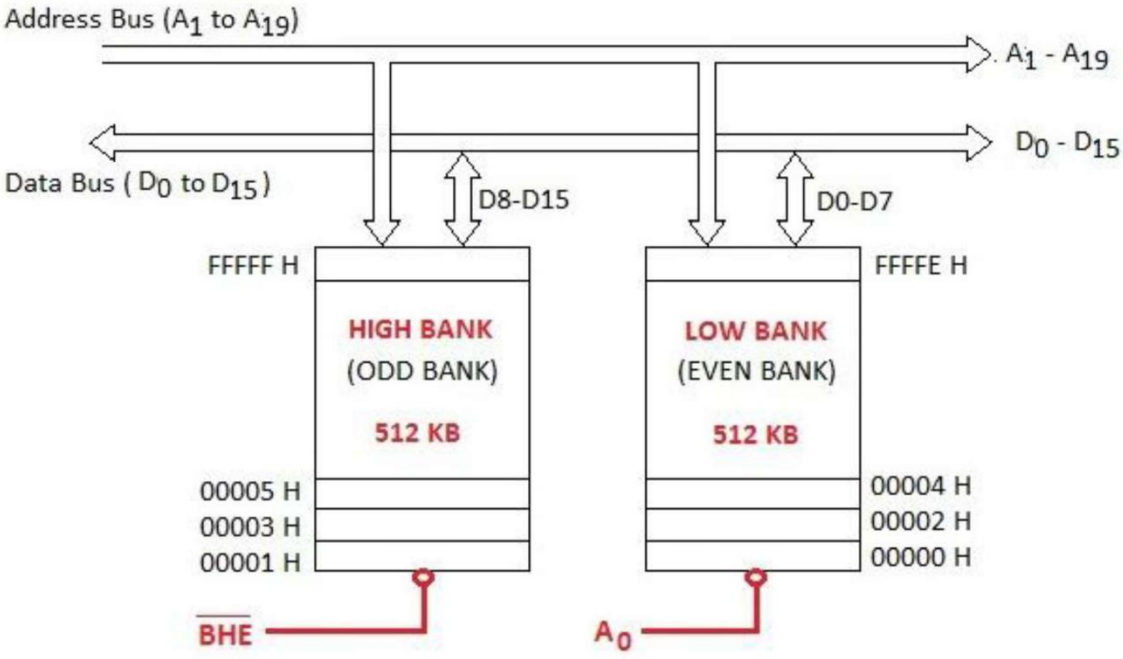


Memory Read Timing in Maximum Mode



1.6. 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

1.7. 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 4
- 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 :

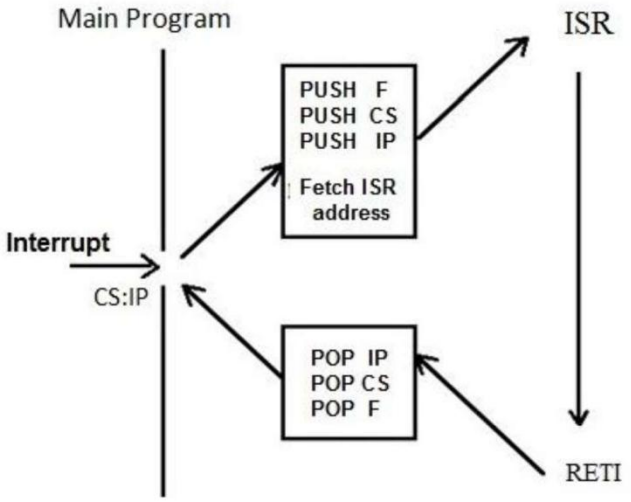
These interrupts are caused by writing the software interrupt instruction **INT N** where 'N' can be any value from 0 to 255 (00H to FFH). Hence all 256 interrupts can be invoked by software.

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.

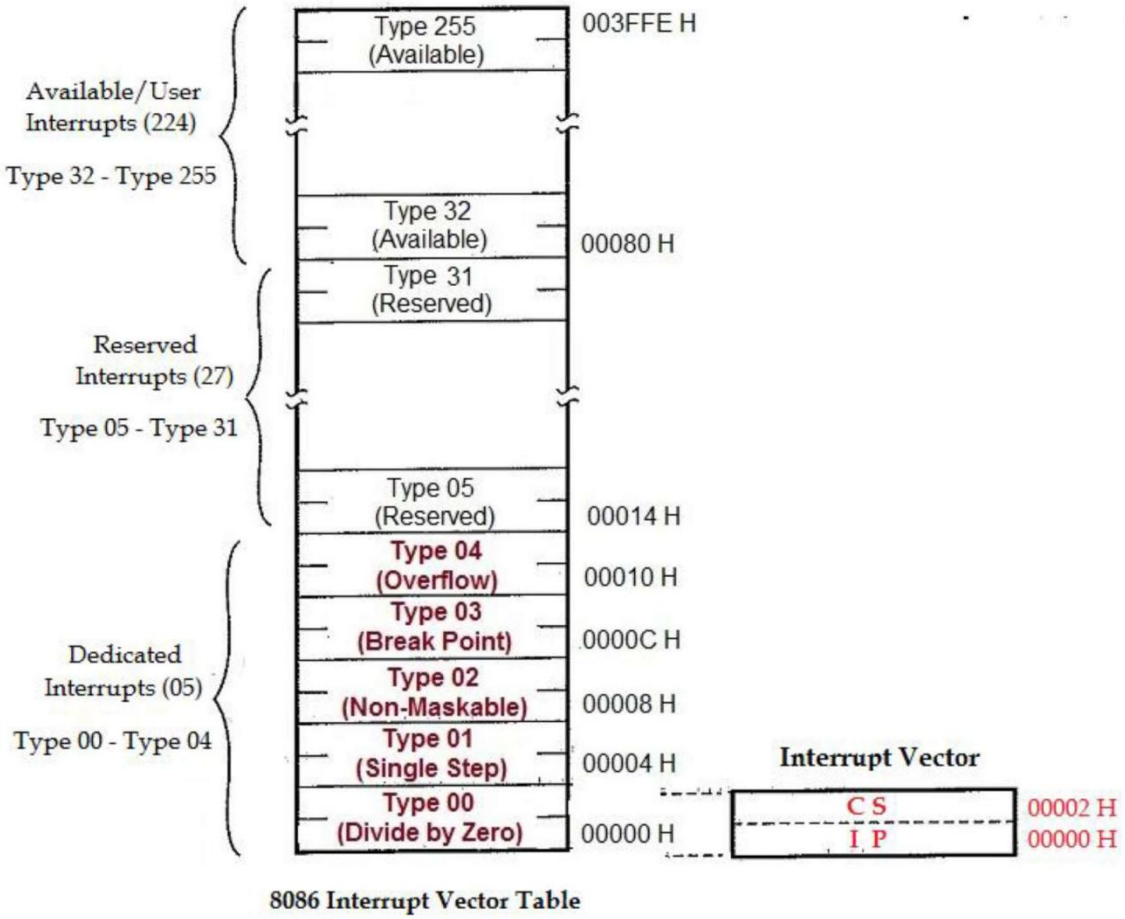
Interrupt Vector Table:

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 = **00000**H 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 = **00004**H 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 = **00008**H 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 = **0000C**H 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 = **0001**0H 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 HInterrupt vector address for Type-23H is
0000:008C H

(b) The address of ISR for Type-23H is available at
0000:008C H ISR IPis available at memory
0000:008C H = 3412 HISR CSis available at memory 0000:008E H = 7856 HAddress of ISR =

(c) Interrupt vector address = Type * 4
00C8 = Type * 4
Type = 00C8 / 4 = 32H
Hence the Type of Interrupt is 32H