Microprocessor (Arch. & Programming - 8bit-8085)

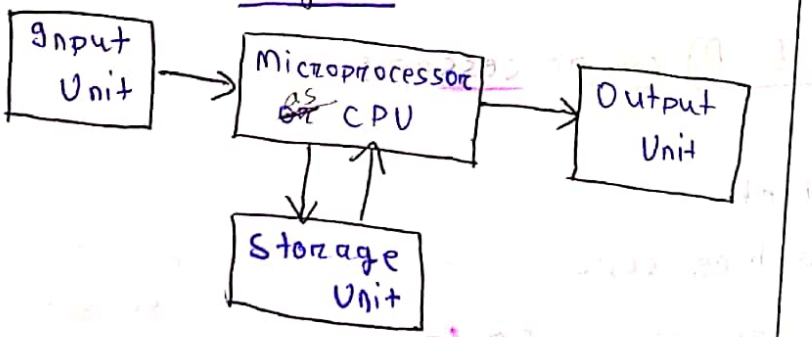
Microprocessor :-

⇒ Microprocessor is a programmable electronic chip or sillicon IC (Integrated Circuit).

⇒ Microprocessor is a small processor or computer processor which incorporates the functions of computer's central processing unit (CPU) on the single IC chip.

⇒ Basically microprocessor contain → multipurpose
→ digital service
→ sillicon IC
→ reprogramming or reprogrammable

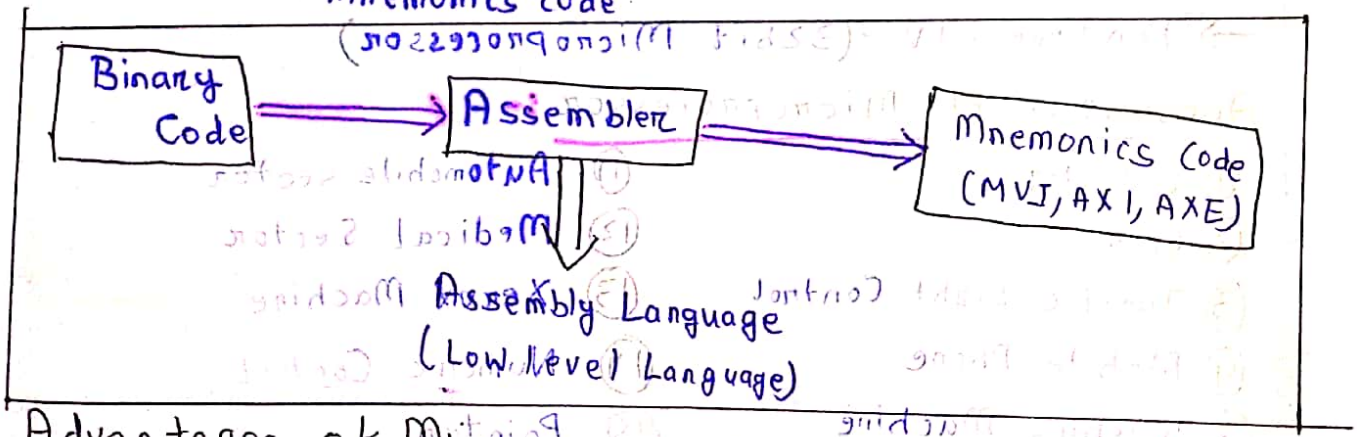
Diagram



⇓

Binary Code (0,1) & MVI, Mnemonics Code (AX I, AXE)

⇒ Microprocessor programming concept is in two ways → binary code
→ mnemonics code



Advantages of Microprocessor :-

→ Speed
→ Data Movement
→ Complex Mathematics
→ Small in Size

Speed: The rate of speed is becoming more to use of microprocessor

Data Movement: Data can be easily, transfer from one place to another to use microprocessor

Complex Mathematics: More complex mathematical function will be done by the microprocessor.

Small in Size: It shape of size is small.

Disadvantages of Microprocessor:-

→ Expensive
→ It get overhitted
→ Using only machine code

Example of Microprocessor:-

→ 8085 – (8 bit Microprocessor)
→ 8086 – (16 bit Microprocessor)
→ 80286 – (16 bit Microprocessor)
→ Pentinum - IV - (32bit Microprocessor)

Application of Microprocessor:-

① Calculator
② Game
③ Traffic Light Control
④ Mobile Phone
⑤ Washing Machine
⑥ Mini Oven
⑦ Refrigerator
⑧ Laptop/PC
⑨ Digital Watch
⑩ Television
⑪ Automobile sector
⑫ Medical Sector
⑬ Xerox Machine
⑭ Numeric Control
⑮ Pointer

GENERATION OF MICROPROCESSOR :-

(1) Small Scale Integration (SSI)

→ It was introduced in the year of 1961,
→ And it also uses the (10-20) transistor,
→ Best example of small scale integration is inverter,

(2) Medium Scale Integration (MSI)

→ It was introduced in the year of 1996.
→ And it also uses the (21-100) transistor,
→ Best example of medium scale integration is multiplexer, decoder, encoder.

(3) LARGE SCALE INTEGRATION (LSI)

→ It was introduced in the year of 1971,
→ And it also uses the 1000 Transistor,
→ Best example of Large Scale Integration is 4004- (4 bit Microprocessor), 8085 (8-bit Microprocessor)

(4) VERY LARGE SCALE INTEGRATION (VLSI)

→ It was introduced in the year of 1990.
→ And it also uses the 10,000 Transistor,
→ Best example of Very Large Scale Integration is 8086, 80286 (16 bit microprocessor)

(5) ULTRA LARGE SCALE INTEGRATION (VLSI)

→ It was introduced in the year of 2000,
→ It also uses the 10,000 transistor,
→ Best example of VLSI in Pentinum IV (32-bit Microprocessor).

(6) Graphic s Scale Integration (GSI)

→ It was introduced in the year of 2002.
→ And it also uses the 10,00,000 transistor
→ Best example of Graphics Scale Integration is TMS 320 (Trans Manufacturing Scale).

Evolution Of Microprocessor :-

Sl. No.	Processor	Year	Address Bus Width	Data Bus Width
01	4004	1971	10 bit	4bit
02	8008	1972	14 bit	8 bit
03	8080	1973	16 bit	8 bit
04	8085	1975	16 bit	8 bit
05	8086	1978	20bit	16 bit
06	8088	1979	20bit	16 bit
07	80286	1982	24bit	16 bit
08	80386	1985	32 bit	32 bit
09	80486	1989	32 bit	32 bit
10	Pentinum	1993	32bit	64 bit
11	Pentinum-pro	1995	36 bit	64 bit
12	Pentinum-II	1997	36 bit	64 bit
13	Pentinum-III	1999	36bit	64bit
14	Pentinum-IV	2000	36bit	64bit

(1) The processor 4004 was introduced in the year 1971. Its Address bus width is 10 bit and Data bus width is 4 bit.

(2) The processor 8008 was introduced in the year 1972. Its Address bus width is 14 bit and Data bus width is 8 bit.

(3) The processor 8080 was introduced in the year 1973, its Address bus width is 16 bit and Data bus width is 8 bit.

(4) The processor 8085 was introduced in the year 1975, Its Address bus width is 16 bit and Data bus width is 8 bit.

(5) The processor 8086 was introduced in the year of 1978. Its address bus width is 20 bit and data bus width is 16 bit.

(6) The processor 8088 was introduced in the year of 1979, Its address bus width is 20 bit and Data Bus width is 16 bit.

(7) The processor 80256 was introduced in the year of 1982. Its Address bus width is 24 bit and data bus width is 32 bit.

(8) The processor 80386 was introduced in the year of 1985. Its Address bus width is 32 bit and data bus width is 32 bit.

(9) The processor 80486 was introduced in the year of 1989. Its Address bus width is 32 bit and data bus width is 32 bit.

(10) The processor pentinum was introduced in the year of 1993. Its Address bus width is 32 bit and data bus width is 64 bit.

(11) The processor pentinum-pro was introduced in the year of 1995, Its Address bus width is 36 bit and Data Bus width is 64 bit.

(12) The processor pentinum-II was introduced in the year of 1997. Its Address bus width is 36 bit and data bus width is 64 bit.

(13) The processor pentinum-III was introduced in the year of 1999. Its Address Bus width is 36 bit and data bus width is 64 bit.

(14) The processor pentinum-IV was introduced in the year of 2000. Its Address Bus width is 36 bit and data bus width is 64 bit.

DISTINGUISH BETWEEN MICROPROCESSOR & MICROCOMPUTER

MICROPROCESSOR	MICROCOMPUTER
→ It is a programmable electronic circuit and IC chip	→ It is a system computer is the combination of I/P unit and O/P unit, storage unit, CPU.
→ Microprocessor is also the components of the micro-computer.	→ Microcomputer is a complete computer similar to other computer.
→ It is also actually brain of the computer.	→ It is also included application software and system software.
→ It include ALU (Arithmetic Logic Unit), ID (Instruction Decoder) and register	→ It also included microprocessor and the peripheral devies (I/P, O/P devices, storage)
→ It is the fully based of the computer system.	→ Hardware is the base of microcomputer
→ Example:- 8085, 4004, 8086, 80286, etc.	→ Example:- Laptop, Desktop, etc.

Microcomputer :-

The function as the CPU of a computer is called microcomputer, e.g. Laptop, Notebook, etc.

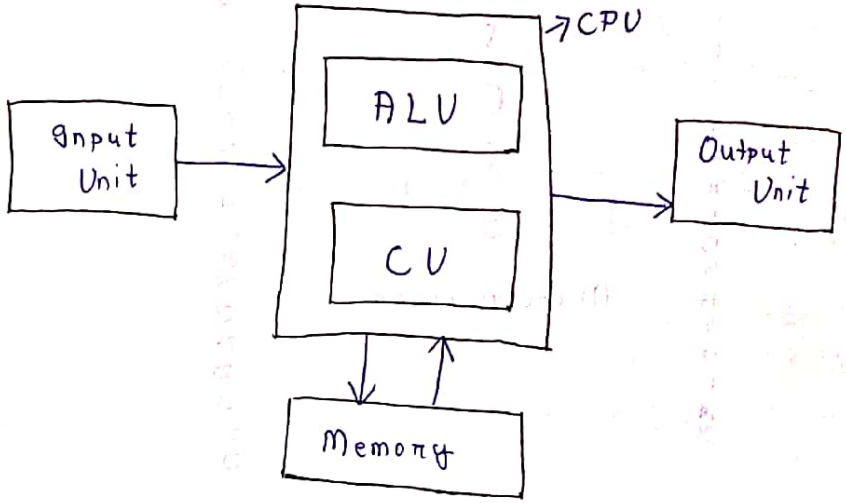


Fig:- Microcomputer

Addressing Range of 8085 Microprocessor :-

The address range of 8085 microprocessor is 64 Kilobyte.

* 8085 Address bus range = 16

$= 2^{16} = 2^{10} \times 2^6$

$= 1 \times 64$

$= 64$ KB

The address range of 8086 microprocessor is 1 MB (Megabyte)

* 8086 Address bus range = 20

$= 2^{20} = 2^{20}$

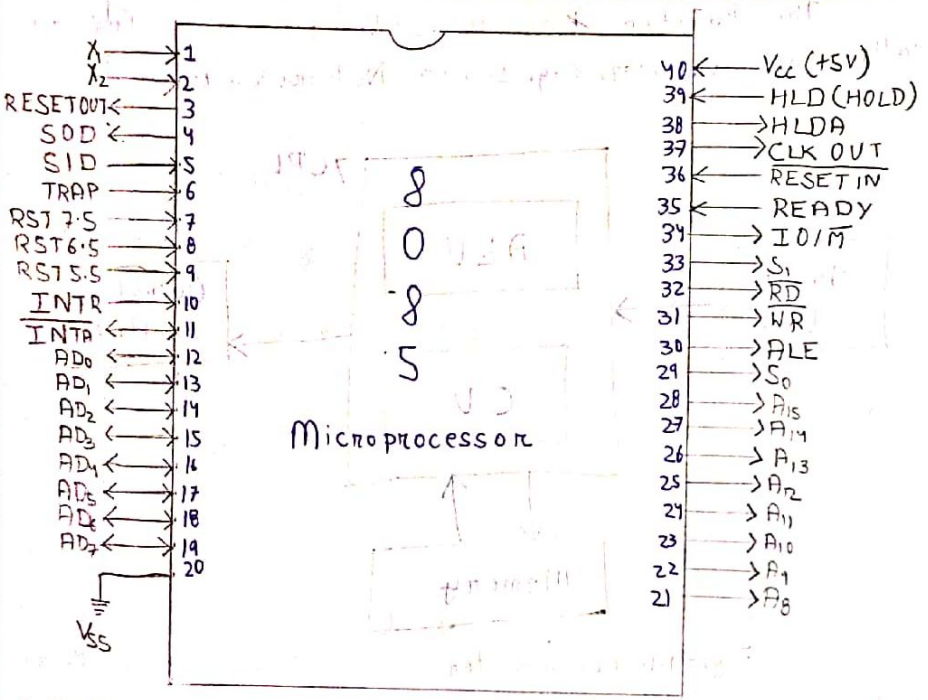
$= 1$ MB

1 KB = 1024 = 2^{10} byte

1 MB = 1024 = 2^{20} KB

1 GB = 1024 = 2^{30} MB

PIN DIAGRAM OF 8085 MICROPROCESSOR :



→ 8085 microprocessor is an 8-bit Microprocessor, it has 40 pin IC, DIP (Dual Inline Package).

→ The signal from this pin can be categorized in six groups such as:

I) Power Supply
II) Address Bus
III) Data Bus
IV) Timing & Control Signal
V) Interrupt Control
VI) Serial I/O port.

I) Power supply and Clock Signal

⇒ There are two power signal i.e. V_{cc} and V_{ss}.

⇒ Vcc indicates +5V power supply and it is connected pin number 40.

⇒ Vss in the ground signal and it is connected to the pin number 20.

Clock Signals

⇒ There are three clock signals (i) X_1
(ii) X_2
(iii) CLK OUT

⇒ X_1, X_2 are the terminal which are connected to the crystal oscillator.

⇒ It is use to set the frequency of Internal Clock generator.

⇒ Microprocessor require 3MHz frequency, these frequency is internally divided into two parts. So, that the frequency is 6 MHz.

Clock Out

⇒ This is the output pin. This signal is used as system clock per other device connected to the microprocessor.

2) Address Bus

⇒ It is from A_8 to A_{15} pin

⇒ It is the high order Address Bus.

⇒ It is use to address the memory location of microprocessor.

(3) Data Bus

⇒ It is from AD_0 to AD_7

⇒ It is the low order multiplex address and databus. So, it can use to carry 8-bit address bus

(4) TIMING AND CONTROL SIGNAL

→ **ALE (Address Latch Enable Signal)**

⇒ It is an output signal

⇒ It is used to give the information of $AD_0 - AD_7$ content

⇒ It indicate the bus function i.e Address Bus and Data Bus.

⇒ When ALE=1 (Set), the bus function is address bus

⇒ When ALE=0 (Reset), the bus function is data bus

→ **IO/$\overline{M}$**

⇒ It is an Output signal

⇒ This signal define input, output and memory location.

⇒ If IO/$\overline{M}$ goes high it indicate I/O operation

⇒ When the signal goes low it indicate memory location

→ **$\overline{RD}$ (Read)**

⇒ It is an Output signal

⇒ It is a active low signal

⇒ It control the signal used for read operation from memory or I/O device.

⇒ When the signal goes low, the microprocessor send the data from memory.

→ **$\overline{WR}$ (Write)**

⇒ It is an output signal.

⇒ It is a active low signal

⇒ It control the signal used for write operation. Either from the memory or I/O device.

⇒ When write signal goes low, the data is written into memory.

→ **S_0, S_1 (Status Signal)**

⇒ S_0, S_1 are the status signal. These status signal may be used to know the type of operation to the CPU perform.

S_1	S_0	Comments
0	0	HLT (STOP)
0	1	$\overline{WR}$
1	0	$\overline{RD}$
1	1	OPCODE FETCH

→ **Table for control signal**

	Machine Cycle	IO/$\overline{M}$	S_1	S_0	Control Signal
1	OPCODE FETCH	0	1	1	$\overline{RD}=0$
2	MEMORY READ	0	1	0	$\overline{RD}=0$
3	MEMORY WRITE	0	0	1	$\overline{RD}=0$
4	I/O READ	1	1	0	$\overline{WR}=0$
5	I/O WRITE	1	0	1	$\overline{INTA}=0$
6	INTERRUPT	1	1	0	$\overline{WR}=0$
7	HLT	Z	0	1	$\overline{RD}, \overline{WR}=Z$
8	HOLD	Z	X	X	$\overline{RD}, \overline{WR}=Z$
9	RESET	Z	X	X	—

'Z' = High Impedance

'X' = Unspecific

→ HLD (HOLD)

⇒ It is used to request the microprocessor for DMA Transfer. (Direct Memory Access)

⇒ It is an input signal.

⇒ This request is send by a DMA controller i.e Intel 8237, Intel 8257, etc.

→ HLDA (HOLD ACKNOWLEDGEMENT)

⇒ It is an output signal

⇒ On the receive of hold signal the microprocessor acknowledgement the request send by the HLDA signal.

⇒ After the HLDA signal the DMA controller transmit the different data between memory, input and output device.

→ READY

⇒ It is an input signal

⇒ The ready signal is used by the microprocessor to check wheather the peripherial device are ready to transfer data or not,

⇒ When the ready signal goes high the peripherial devices are ready to transfer the data

⇒ When the ready signal goes low the peripherial devices are not ready to transfer the data,

⇒ The microprocessor enter into a wait state when the ready pin is reset(0).

→ RESET IN

⇒ It is an input signal.

⇒ It is an active low signal

⇒ When the signal goes low the program counter is said to 0 and microprocessor is reset.

⇒ This signal is used to reset the microprocessor.

→ RESET OUT

⇒ It is an output signal.

⇒ It is used to reset the peripherial device and also the microprocessor.

⇒ The output on this pin goes high whenever the reset in pin goes low.

(5) INTERRUPT CONTROL SIGNAL

→ INTERRUPT

⇒ It means the interrupting the normal executive of microprocessor, when the microprocessor receive interrupt signal it discontinue whatever executive.

⇒ It start executing the new program indicated interrupt signal. Interrupt signal are generated by the extra peripherial device after executing the new program the microprocessor goes back to the previous program.

⇒ The pin number 6 to 11 are used as Interrupt signal. i.e. TRAP, RST7.5, RST6.5, RST5.5, INTR, $\overline{INTA}$.

→ TRAP

⇒ TRAP is the hardware interrupt

⇒ It has the highest priority among ALL the input signal.

→ RST7.5, RST6.5, RST5.5

⇒ This signal are the vector interrupt that transfer the program control to a specific memory location.

→ INTR (INTERRUPT REQUEST)

⇒ It is the lowest priority among the interrupt signal.

⇒ When INTR goes high the microprocessor complete connect instruction which are being executed.

→ INTA (INTERRUPT ACKNOWLEDGEMENT)

⇒ When the microprocessor recive ~~interrupt~~ interrupt signal it has to be acknowledgement.

⇒ This Acknowledgment is done by $\overline{INTA}$ signal.

⇒ When the $\overline{INTA}$ signal goes high the microprocessor receive interrupt signal.

(6) SERIAL INPUT AND SERIAL OUTPUT TRANSMISSION

⇒ There are two pins is used for transmission i.e. (i) SID
(ii) SOD

→ (i) Serial Input Data (SID)

⇒ This pin provides serial input data.

⇒ The serial data on this pin is loaded into the accumulator. When RIM (Read Interrupt ~~Mode~~ Mask) is executed.

→ (ii) Serial Output Data (SOD)

⇒ This pin provides serial output data

⇒ The serial data on this pin is loaded into the accumulator. When SIM (Same Interrupt Mask) instruction is executed.

Bus Architecture of 8085 Microprocessor/8085

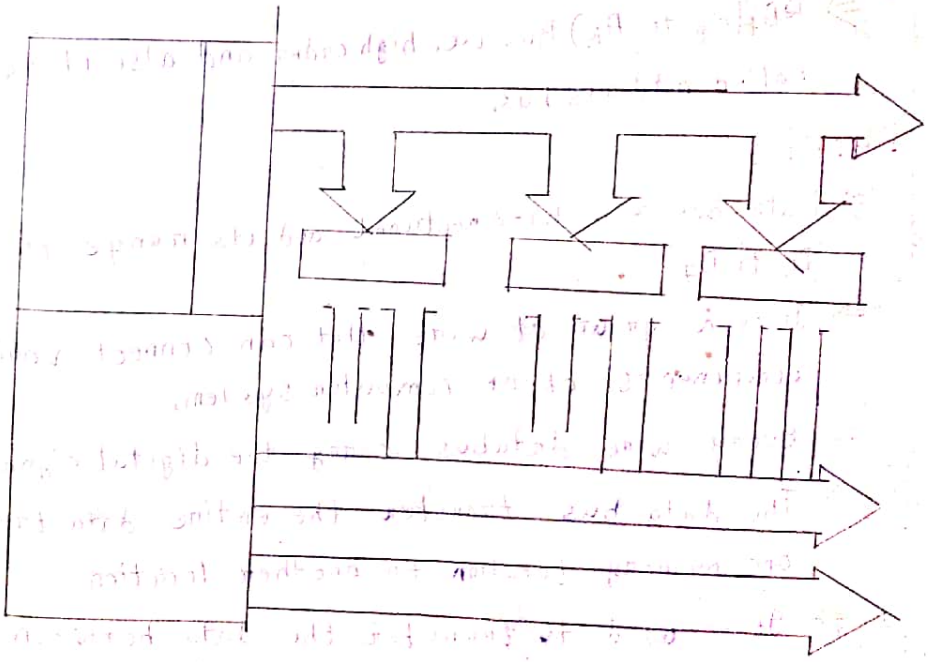
Bus Architecture :-

Bus

⇒ A group of line/wire is called Bus and Bus is also a medium through which data can be transfered from one place to another easily

⇒ In computer architecture, three types of Buses are necessary for performing all kinds of internal operation.

⇒ There are three types of Bus are (i) Address Bus
(ii) Control Bus
(iii) Data Bus



Address Bus

⇒ Address Bus is unidirectional and its range is 16-bit (A_0-A_{15}).

⇒ It is a group of line that are use to transfer the address of memory input or memory output.

⇒ The 8085 microprocessor can transfer maximum upto 16-bit address.

⇒ The address of each memory location input, output are different.

⇒ The bus is multiplex with 8-bit address.

⇒ Bus (AD_0 to AD_7) i.e. low order and also it is called data bus.

⇒ (A_8 to A_{15}) Bus i.e. high order and also it is called address bus.

Data Bus

⇒ Data bus is bi-directional and its range from D_0 to D_7

⇒ It is a group of wire that can connect various components of an computer system.

⇒ Every wire databus carry the digital signal.

⇒ The data bus transfer the entire data from one memory location to another location.

⇒ It is used to transfer the data between memory - input, output - memory.

⇒ After the address is given and the control signal is generated the data will be available on the data bus

⇒ The data bus also work as address bus when it multiplex with low order address bus.

Control Bus

⇒ The control bus is uni-directional.

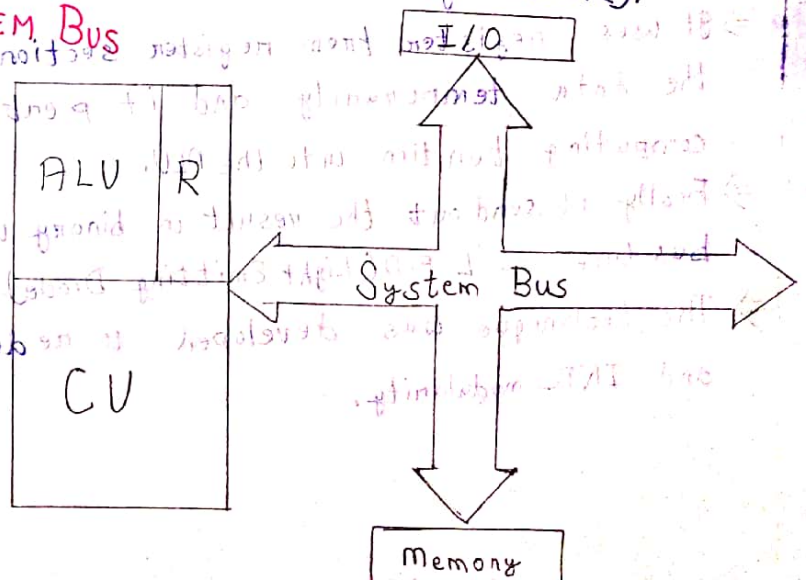
⇒ Microprocessor uses control bus to process the data

⇒ After giving address, the microprocessor will generate several control signal to tell the memory or i/p, o/p device (what kind of operation is perform and when to perform).

⇒ The various operation are perform by the microprocessor on with help of control bus

⇒ The control bus uses control signal such as memory read ($\overline{MRD}$), Memory write ($\overline{MWR}$), Input read ($\overline{IRD}$), Input write ($\overline{IWR}$).

System Bus



⇒ A system bus is a single computer bus that connects the major components of a computer system.

⇒ A system bus is also called a Bi-directional Bus.

⇒ It also combining the function of databus to carry information, an address bus to transfer the address of data and control bus to determine the operation of signal.

⇒ System bus is a communication path between microprocessor and peripherial devices (memory).

⇒ Here all the peripherial (memory) share the same Bus, the microprocessor communicate with only one peripherial device.

⇒ The timing is provided by control unit of the the microprocessor. Here in the microprocessor uses system bus to fetch binary instruction and data from the memory, transfer

⇒ It uses register from register section to store the data temporarily and it perform computing function into the ALU.

⇒ Finally it send out the result in binary using same bus line to L.E.D (Light Emitting Diode).

⇒ The technique was developed to reduce cost and INPC modularity.

DISTINGUISH BETWEEN ADDRESS BUS & DATA BUS

Data Bus	Address Bus
⇒ Databus is bidirectional.	⇒ Address Bus is uni-directional
⇒ Databus range upto D_0 - D_7	⇒ Address Bus range upto A_0 - A_{15}
⇒ 255 different memory location.	⇒ 65,536 different memory location
⇒ But data bus is not called address bus	⇒ Address Bus is also called data bus.
⇒ Complexity is less	⇒ Complexity is high
⇒ Data Bus require to send in receive the data	⇒ Address Bus require only to sent the data.

ACCUMULATOR:

⇒ It is a 8-bit general purpose register (GPR) and it is a part of ALU (Arithmetic Logic Unit).

⇒ The Accumulator is identified by 'A'.

⇒ The register is use to store on 8-bit data and to perform arithmetic and logic operations.

⇒ The result of an operation is store in the accumulator.

TEMPORARY REGISTER:

⇒ It is a 8bit register and also part of ALU.

⇒ It is otherwise called operand register because it provide operand to the ALU.

⇒ ALU can store the immediate result in temporary register and it is called accessible by the user.

ALU (Arithmetic Logic Unit):

⇒ It is a 8-bit register.

⇒ It perform arithmetic operation like that addition, subtraction, multiplication, division and the result of operation is stored in accumulator.

⇒ It also perform Logic operation like that roated, complement, compare, etc and also result of an operation is stored in accumulator.

⇒ ALU include accumulator, temporary register and flag.

⇒ It uses data from memory and accumulator to perform the operation.

Flag:

⇒ It is a 8-bit register and also part of the ALU.

⇒ Flag register is a group of flipflop.

⇒ It is used to give the status of the different operation result.

⇒ The flag register is connected to the ALU.

⇒ Once, one operation is performed by the ALU the result is transfered on internal data bus and status of result will be stored in flag.

⇒ Five flag register are as follows:

i) Sign Flag (S) ii) Zero Flag (Z)

iii) Auxilary Carry Flag (AC) iv) Parity Flag (P)

v) Carry Flag (C_y)

i) Sign Flag:-

⇒ After the execution of all operations, if D_7 bit is one then number will be considered as register.

⇒ If the D_7 bit is zero then number will be considered as positive

```
  1 0 0 1 1 1 0 1          1 0 0 0 1 1
  1 0 0 1 1 0 1 0            1 0 0 0 1 1
-------------------      -------------------
0 0 1 1 0 1 1 1 (Positive) 0 1 0 0 0 1 1 0 (Positive
```

ii) Zero Flag:-

⇒ It is represented by D_6 bit.

⇒ If the result of ALU operation is zero then zero flag is set otherwise reset.

iii) Auxilary Carry Flag:-

⇒ It is represented by D_4 bit.

⇒ If ALU operation, if carry is generated by the bit then D_3 and D_4 bit the auxilary flag is set otherwise reset.

iv) Parity Flag:-

⇒ If the result of ALU operation even number of one then parity flag will be set.

⇒ If the result of ALU operation provides odd number of one then parity flag will be reset.

v) Carry Flag:-

⇒ If the operation perform on ALU generate the carry from D_7 to next step the carry is set otherwise reset.

Register Section:-

⇒ In 8085 microprocessor, the section is divided into two group. ⇒ General purpose register (GPR)
⇒ Special purpose register (SPR)

General Purpose Register (GPR)

⇒ The general purpose register are W-Z, D-E, H-L, B-C.

Special Purpose Register (SPR)

⇒ The special purpose register are PC (Program Counter), stack pointer (SP),

Stack pointer:

⇒ Stack pointer is also 16-bit special purpose register.

⇒ It is known as memory pointer.

⇒ It point to the memory location that is called stack.

⇒ Stack is also LIFO (Last In First Out) concept when the ~~stack~~ data is loading into the stack, the stack point is decremented when the data is receive from the stack, the stack pointer is incremented.

Program Counter:

⇒ It is a 16 bit special purpose register.

⇒ It store the memory address of the next ~~the~~ instruction to be executed.

Increment / Decrement:

⇒ Increment concept is consider as data will be move by one by one and decrment concept is consider as remove data one by one.

Address Bus:

⇒ In 8085 microprocessor, the address bus range is from $A_0 - A_{15}$.

⇒ In that $(A_0 - A_7)$ is the low order address bus and $(A_8 - A_{15})$ is the high order address bus.

⇒ It is an uni-directional bus.

Data Bus:

⇒ In 8085 microprocessor, the data bus range is $D_0 - D_7$

⇒ It is also called as bidirectional bus. Low order address bus is also known as data bus.

Instruction register and decoder:

⇒ It is 8 bit register.

⇒ The instruction may be any-thing that is ADD, SUB, MUL, DIV, etc.

⇒ When the instruction is phase for the ~~memory~~ memory for internal databus, its store in the instruction register.

⇒ The machine encoder and decoder will encode and decode the data accrodingly.

Interrupt Control:

⇒ There are 6 interrupt pin used that is INTA, INTR, RST 5.5, RST 6.5, RST 7.5, TRAP.

⇒ These six pin provide interrupt signals send by external hardware to the microprocessor and microprocessor acknowledgement for receiving the interrupt signal where INTA is the interrupt acknowledgement.

Serial I/O Transmission:

⇒ There are two pin use for transmission that is SID (Serial input data), SOD (Serial Output data).

Serial Input Data (SID)

⇒ This pin provides serial input data

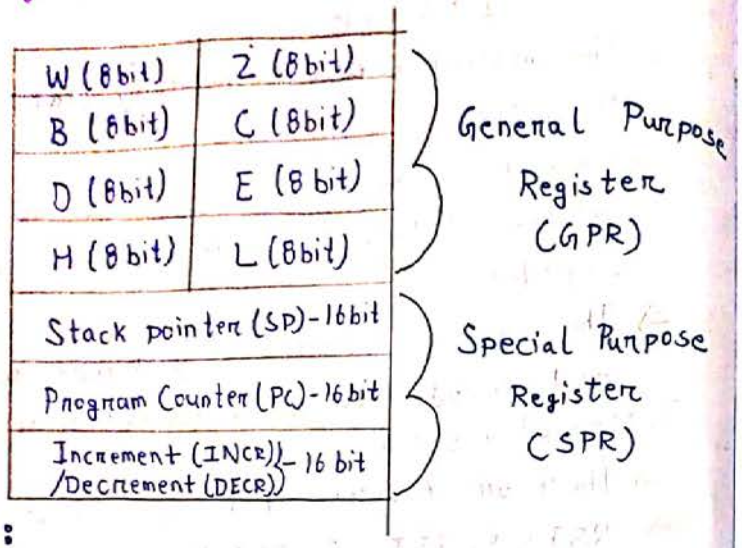
⇒ The serial data on this pin is loaded into the accumulator when RIM (Read Interrupt Mask) is executed.

Serial Output Data (SOD)

⇒ This pin provides serial output data.

⇒ The serial data on this pin is loaded into the accumulator when SIM (Serial Interrupt Mask) is executed.

Register Organization of 8085 Microprocessor:



Register:

⇒ Register is used by the microprocessor for manipulation of instruction

⇒ INTEL 8085 microprocessor use register array which contents 8-bit register.

⇒ The register W-Z pair are internal to the microprocessor.

⇒ The register B-C pair, D-E pair, H-L pair are available to the programmer.

⇒ The register H-L is used as memory in the microprocessor.

⇒ In 8085 microprocessor, there are 4 registers are available:

i) Temporary register (TR)
ii) Accumulator register (AR)
iii) Flag register (FR)
iv) Instruction register (IR)

Temporary Register (TR):

⇒ In 8085 microprocessor the register section is divided into GPR and SPR.

⇒ Example: GPR (B-C, D-E, H-L, W-Z)
SPR (PC, INCR, DECR)

⇒ Microprocessor is use W-Z pair as internal pair register and 3 pair in instruction are B-C pair, D-E pair, H-L pair.

⇒ GPR is also called as temporary register.

Accumulator Register (AR):

⇒ It is also GPR. It is 16 bit register and it is identity by "A".

⇒ The register is use to store on 8-bit data and to perform arithmetic and logic operation.

⇒ The result of an operation is store in the accumulator.

Flag Register (FR):

⇒ It is 8-bit register and it is the part of ALU.

⇒ Flag register is a group of flip-flop.

⇒ It is used to give the status of different operation result. The status of result will be stored in flag.

Instruction Register (IR):

⇒ It is a part of CPU's control unit that holds the instruction currently being executed or decoded.

⇒ A simple processor each instruction register executed is loaded into the instruction register which holds it while it is decoded.

⇒ When the instruction is fetch from the memory or internal databus, its stores in the instruction register.

⇒ The machine encoder and decoder will encode and decode the data accordingly.

Difference between GPR and SPR:

GPR	SPR
⇒ GPR stands for general purpose register.	⇒ SPR stands for special purpose register.
⇒ In 8085 microprocessor there are 8 GPR which contains 8-bit data	⇒ In 8085 microprocessor, there are 3 SPR which contains 16-bit data.
⇒ GPR is used by the programmer to stored the data	⇒ SPR is used by the CPU for temporary register storage of data for calculation or other purpose.
⇒ Temporary register is also called GPR (partially)	⇒ Instruction register is also called SPR.
⇒ Ex- W-Z, B-C, H-L, etc.	⇒ Ex- INCR/DECR, SP, PC.

Stack Pointer:

⇒ It is basically defined as the collection of memory location used for temporary storage of data.

⇒ The stack program ranges from 000H to FFFH

⇒ The stack in "LIFO" structure.

⇒ The stack is normally group backward into the memory.

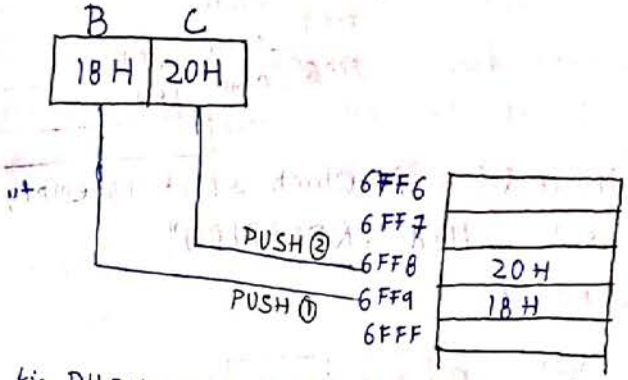
⇒ The 8085 microprocessor provides two instruction i.e. PUSH and POP instruction for storing and retrive information into the stack.

PUSH Operation OR Instruction:

⇒ PUSH in an operating within in used to insert an element into the stack.

⇒ PUSH operation decremented the stack pointer and then copy the data into the stack.

Example:



⇒ In this fig. PUSH operation first decrement the stack pointer then copy the content of register to the memory location period out by stack pointer.

⇒ Then again decrement the stack pointer and copy the data from the register C.

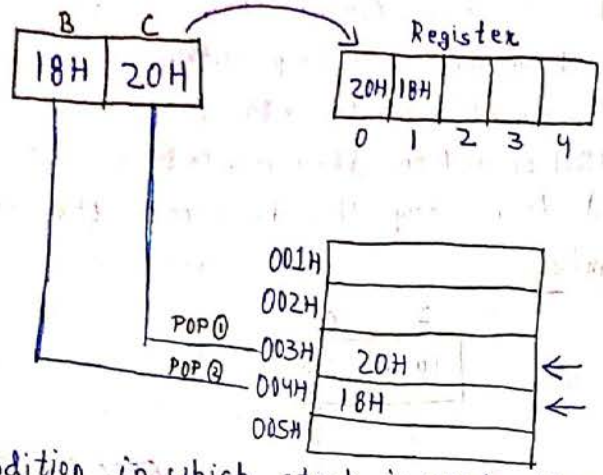
⇒ The condition in which the stack is full then it is called Stack overflow.

POP Operation OR Instruction:

⇒ POP is an operation which is used for delete an element from the stack.

⇒ POP operation first increment the stack pointer then delete from the top of the stack.

⇒ In the figure, first stack pointer increment then the data in delete from the stack and it store in register.



⇒ The condition in which stack is empty then it is used called "UNDERFLOW".

PROBLEM:

PUSH

A	32H	39H	F
B	22H	18H	E
D	19H	05H	C

LXI SP, 7006H
PUSH D
PUSH C

Using PUSH operation display this question in the stack memory location.

ANSWER:

D	C
19H	05H

7006H	39H
7005H	32H
7004H	18H
7003H	22H
7002H	05H
7001H	19H
7000H	

8085 INTERRUPTS

Interrupt

⇒ Interrupt is a mechanism by which an I/O or an instruction can suspend the normal execution of processor, and get itself serviced.

⇒ Generally, a particular task is assigned to that interrupt signal. In the microprocessor based system the interrupts are used for data transfer between the peripherial devices and the microprocessor.

Interrupt Service Routine (ISR)

⇒ A small program or a routine that when executed services the corresponding interrupting source is called as ISR.

⇒ When microprocessor receives interrupt signals, it sends an acknowledgement (INTA) to the peripherial which is requesting for its service.

Execution of Interrupts

When there is an interrupt requests to the Microprocessor then after accepting the interrupts Microprocessor send the INTA (active low) signal to the peripherial. The vectored address of particular interrupt is stored in program counter. The processor executes an interrupt service routine (ISR) addressed in program counter.

TYPES OF INTERRUPTS

Interrupt are classified into following groups based on their parameter:

⇒ **Hardware Interrupts:**

When the microprocessor receive interrupt signals through pins (hardware) of microprocessor they are known as microprocessor. They are- INTR, RST 7.5, RST 6.5, RST 5.5, TRAP.

⇒ **Software Interrupts:**

Those interrupts which are inserted in between the program which means these are mnemonics of microprocessor, i.e called Software Interrupt. They are- RST 0, RST 1, RST 2, RST 3, RST 4, RST 5, RST 6, RST 7.

⇒ **Vectored Interrupts:**

Those interrupt which have fixed vector address (starting address of ISR) and after executing these, program control is transferred to that address. For example: RST 7.5, RST 6.5, RST 5.5, TRAP.

⇒ **Non- Vectored Interrupts:**

Those interrupts in which vector address is not predefined. This is called non-vectored Interrupt. The interrupting device gives the address of ISR for these interrupts. INTR is the only non-vectored interrupt in 8085 microprocessor.

⇒ **Maskable Interrupts:**

Those interrupt which can be disabled or ignored by the microprocessor, i.e. called maskable interrupts, These interrupts either edge-triggered or level-triggered, so they can be disabled. INTR, RST 7.5, RST 6.5, RST 5.5 are maskable interrupt in 8085 microprocessor.

⇒ **Non- Maskable Interrupts:**

Those interrupt which cannot be disabled or ignored by microprocessor. TRAP is a non-maskable interrupt. It consist of both level as well as edge triggering and is used in critical power failure conditions

Software Interrupts

⇒ A software interrupts is a particular instruction that can be inserted into the desired location in the program

⇒ There are eight software interrupts in 8085 microprocessor and they are:

- → RST 0
- → RST 1
- → RST 2
- → RST 3
- → RST 4
- → RST 5
- → RST 6
- → RST 7

⇒ They allow the microprocessor to transfer program control from the main program to the ISR program. After completing the ISR program, the program control returns back to the main program.

⇒ We can calculate the vector address of these interrupts using the formula given below:

Vector Address = Interrupt Number * 8

⇒ Vector address table for the software interrupts:

Interrupt	Vector Address
RST 0	0000_H
RST 1	0008_H
RST 2	0010_H
RST 3	0018_H
RST 4	0020_H
RST 5	0028_H
RST 6	0030_H
RST 7	0038_H

Hardware Interrupt

⇒ There are six interrupt pins in 8085 microprocessor used as Hardware interrupts and they are:

- → TRAP
- → RST 7.5
- → RST 6.5
- → RST 5.5
- → INTR

Note: INTA is not an interrupt. INTA is used by the microprocessor for sending the acknowledgement

⇒ The vector address of these interrupts are given below:

Interrupt	Vector Address
RST 7.5	$003C_H$
RST 6.5	0034_H
RST 5.5	$002C_H$
TRAP	0024_H

⇒ **TRAP:**

→ It is non-maskable edge and level trigged interrupt.

→ TRAP has the highest priority and vectores interrupt

→ Edge and level triggered means that the TRAP must go high and remain high until it is acknowledged.

→ In case of sudden power failure, it executes a ISR and send the data from main memory to backup memory.

→ As we know the TRAP can not be masked but it can be delayed using HOLD signal. This interrupt transfers the microprocessor's control to Location 0024 H, It can only be masked by resetting the microprocessor or cannot be masked.

⇒ **RST 7.5:**

→ It has the second highest priority

→ It is maskable and edge level trigged interrupt

→ The vector address of this interrupt is 003CH,

→ Edge sensitive means input goes high and no need to maintain high state until it is recognized.

→ It can also be reset on masked by resetting microprocessor. It can also be resetted by DI instruction.

⇒ RST 6.5 and RST 5.5

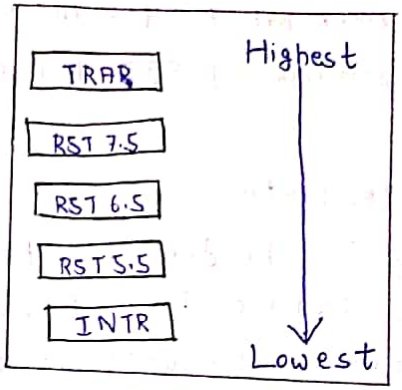
→ These are level triggered and maskable interrupts.
→ When RST 6.5 pin is at Logic 1, INTE flip-flop is set.
→ RST 6.5 has third highest priority and RST 5.5 has fourth highest priority.
→ It can be masked by giving DI and SIM instructions or by resetting microprocessor.

⇒ INTR

→ It is level triggered and maskable interrupt.
→ The following sequence of events occurs when INTR signal goes high.
→ The 8085 checks the status of INTR signal during execution of each instruction.
→ If INTR signal is high, then 8085 complete its current instruction and sends active Low interrupt acknowledge signal, if the interrupt is enabled.
→ On receiving the instruction, the 8085 save the address of next instruction on stack and execute received instruction.
→ It has Lowest priority.
→ It can be disabled by reseting the microprocessor or by DI and SIM Instruction.

Priority of Interrupts

When microprocessor receives multiple interrupt requests simultaneously, it will execute the interrupt service request (ISR) according to the priority of the interrupts.



Instruction for Interrupts

1. Enable Interrupt (EI)

The interrupt enable flipflop is set and all interrupts are enabled following the execution of next instruction followed by EI. No flag are affected. After a system reset, the interupt enable flip-flop is reset, thus disabling the interrupts. This instruction is necessary to enable the interrupts again.

2. Disable Interrupt (DI)

The instruction is used to reset the value of enable flipflop hence disabling all the interrupts. No flags are affected by this instruction.

3. Set Interrupt Mask (SIM)

It is used to implement the hardware interrupts by setting various bits to form masks or generate output data via SOD Line.

4. Read Interrupt Mask (RIM)

This instruction is used to read the status of the hardware interrupts by loading into the A register a byte which defines the condition of the mask bits for the interrupts. It also reads the condition of SID bit on the microprocessor.

Masking of Interrupts

Masking can be done for four hardware interrupts INTR, RST 5.5, RST 6.5, and RST 7.5. The masking of 8085 interrupts is done at different level is done at different levels.

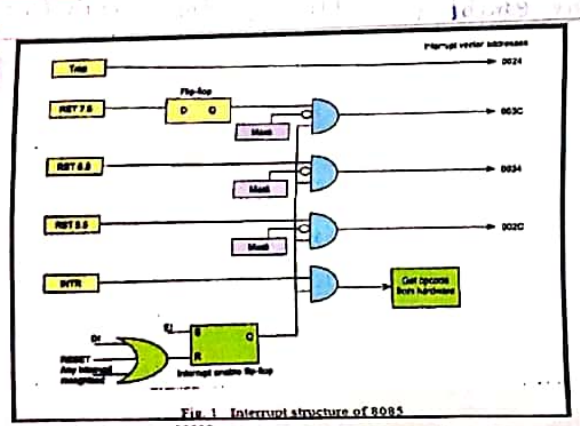


Fig. 1 Interrupt structure of 8085

Fig. 13 Interrupt structure of 8085

⇒ The above dig. can be explained by these five points and they are:

i) The maskable interrupts are by default masked by the Reset signal. So no interrupt is recognized by the hardware reset.

ii) The interrupts can be enabled by the EI instruction.

iii) The three RST interrupts can be selectively masked by loading the appropriate word in the accumulator and executing SIM instruction. This is called software masking.

iv) All maskable interrupts are disabled whenever an interrupt is recognized.

v) All maskable interrupt can be disabled by executing the DI instruction.

⇒ RST 7.5 alone has a flipflop to recognize edge transition. The DI instruction reset interrupt enable flipflop in the processor and the interrupts are disabled. To enable interrupts, EI instruction has to be executed.

SIM Instruction

⇒ The SIM instruction is used to mask or unmask RST hardware interrupts.

⇒ When executed, the SIM instruction reads the content of accumulator and accordingly mask or unmask the interrupts.

⇒ The format of control word to be stored in the accumulator before executing SIM instruction is as shown in the below figure

Bit position	D7	D6	D5	D4	D3	D2	D1	D0
Name	SOD	SDE	X	R7.5	MSE	M7.5	M6.5	M5.5
Explanation	Serial data to be sent	Serial data enable set to 1 for sending	Not used	Reset RST 7.5 flip flop	Mask Set enable — Set to 1 to mask interrupt	Set to 1 to mask RST 7.5	Set to 1 to mask RST 6.5	Set to 1 to mask RST 5.5

⇒ Accumulator bit pattern for SIM instruction. In addition to masking interrupts, SIM instruction can be used to send serial data on the SOD line of the processor.

⇒ The data to be send is placed in the MSB bit of the accumulator and the send serial data on the SOD output is enabled by making D6 bit to 1.

RIM Instruction

⇒ RIM instruction is used to read the status of the interrupt mask bits.

⇒ When RIM instruction is executed, the accumulator is loaded with the current status of the interrupt masks and the pending interrupts.

⇒ The format and the meaning of the data stored in the accumulator after execution of RIM instruction is shown in below figure.

⇒ In addition RIM instruction is also used to read the serial data on the SID pin of the processor.

⇒ The data on the SID pin is stored in the MSB of the accumulator after the execution of the RIM instruction.

Bit position	D7	D6	D5	D4	D3	D2	D1	D0
Name	SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5
Explanation	Serial input data in the SID pin	Set to 1 if RST 7.5 is pending	Set to 1 if RST 6.5 is pending	Set to 1 if RST 5.5 is pending	Set to 1 if interrupt are enabled	Set to 1 if RST 7.5 is masked	Set to 1 if RST 6.5 is masked	Set to 1 if RST 5.5 is masked

Accumulator bit pattern after execution of RIM instruction

Instruction Set and Assembly Language Programming

Instruction Word Size:-

⇒ The 8085 instruction set is divided into three groups according to the word size or byte size.

1) 1-byte instruction
2) 2-byte instruction
3) 3-byte instruction

1-Byte Instruction:

⇒ 1 byte instruction include opcode and operand in the same byte.

Opcode	Operand
ADD	B

Binary Code	Hexcode
10010010	00

⇒ Here accumulator and register B are the same byte i.e. 1 byte.

⇒ It is also known as 8-bit Instruction.

⇒ The value range upto (0 - 255).

⇒ Example

```
MOV A, B
```

2-Byte Instruction:

⇒ In 2 byte instruction the first byte specify the opcode and the 2nd byte specify the operand.

Opcode	Operand	Hexcode
MVI A	82H	MVI A ⟶ 3E → 1 byte 82H ⟶ 82 → 1 byte

⇒ It is also known as 16-bit instruction.

⇒ The value range upto $(2^{16}-1)$

⇒ Example

```
MVI  A, 15H
 |     |
1st byte  2nd byte
```

3-Byte Instruction:

⇒ The 3 byte instruction the 1st byte specify the opcode and the 2 byte specify the 16 bit address / data.

⇒ In 2-byte the 2nd

Opcode	Operand
LDA	6000 H

LDA → Hexcode → 3A → 1-byte

⇒ In 6000, 00 is the lower byte and 60 is the higher byte

⇒ It is also known as 24-bit instruction on 32bit,

⇒ The value range upto $(2^{24}-1)$ on $(2^{32}-1)$.

⇒ Example

LDA 6000H

1st byte

ADDRESSING MODE:-

⇒ The way in which operand are specified is known as addressing mode.

OR

⇒ The way of specifying data to be operated by an instruction is called addressing mode.

⇒ In 8085 microprocessor there are five types of addressing modes:

→ Immediate Addressing Mode

→ Register Addressing Mode

→ Direct Addressing Mode
→ Register Indirect Addressing Mode
→ Implied / Implicit Addressing Mode

Immediate Addressing Mode:

In this addressing mode, the source operand is always data. If the data is 8-bit, then the instruction will be of 2 bytes, if the data is of 16-bit then the instruction will be of 3 bytes

Examples:

MVI B 45 (move the data 45H immediately to register B)
LXI H 3050 (load the H-L pair with the operand 3050H immediately)
JMP address (jump to the operand address immediately)

Register Addressing Mode:

In register addressing mode, the data to be operated is available inside the register(s) and register(s) is (are) operands. Therefore the operation is performed within various registers of the microprocesson.

Examples:

MOV A, B (move the contents of register B to register A)
ADD B (add contents of registers A and B and store the result in register A)
INR A (increment the contents of register A by one)

Direct Addressing Mode:

In this addressing mode, that data to be operated is available inside a memory location is directly specified as an operand.

⇒ The operand is directly available in the instruction itself.

Examples:

LDA 2050 (load the contents of memory location into accumulator A)

LHLD address (load contents of 16-bit memory location into H-L register pair)

IN 35 (read the data from port whose address is 01)

Register Indirect Addressing Mode:

In this addressing mode, the data to be operated is available inside a memory location and that memory location is indirectly specified by a registered pair.

Examples:

MOV A, M (move the contents of the memory location pointed by the H-L pair to the accumulator)

LDAX B (move contains of B-C register to the accumulator)

LXIH 9570 (load immediate the H-L pair with the address of the location 9570)

Implied / Implicit Addressing Mode:

In this addressing mode, the operand is hidden and the data to be operated is available in the instruction itself.

Examples:

CMA (finds and store the 1's complent of the contains of accumulator A in A)

RRC (rotate accumulator A right by one bit)

RLC (rotate accumulator BA left by one bit)

INSTRUCTION SETS:-

⇒ An instruction is a command that given to the microprocessor to perform a specified operation on given data.

⇒ The instruction set of microprocessor is the collection of an instruction that microprocessor design to execute

⇒ The programmer can write the programming assembly level language using that instruction,

⇒ The instruction has been classified into six categories:- ① Data Transfer Group
② Arithmetic Group
③ Logical Group
④ Shift on Rotation Group
⑤ Branch Control Group
⑥ Input & Output and Machine Control Group

Data Transfer Group:

The instruction which are used to transfer the data from one register to another register (R-R), from memory to register (M-R), or register to memory (R-M) comes under this group

Examples: MOV (Move), LDA (Load Accumulator), STA (Store Accumulator) etc--

Arithmetic Group:

The instruction of this group perform arithmetic operation such as addition (+), subtraction (-), increment, decrement of the contained of memory or register.

Examples: ADD, SUB, INR (Increment Register) DCR (Decrement Register) etc--

Logical Group:

The instruction of this group perform logic operation such as AND, OR, ROTATE, etc.

Examples: ANA (AND Accumulator), RAR (Rotate Accumulator Right)

Branch Control Group:

This group include the instruction for conditional and unconditional jump, sub routine, call, return.

Examples: JMP (Jump Parity), CALL, etc.

I/O and machine Control Instruction:

This group include the instruction for input and output (I/O) port, stack and machine control.

Examples: IN, OUT, NOP (No operation) etc-

Data Transfer Group :-

1. MOV (Move) :- (I) MOV (R-R)

* Syntax :- MOV R_d, R_s

The contained of source register (R_s) and move into the destination register (R_d).

* Examples :- MOV B, C [B] ← [C]

⇒ This is indirect addressing mode

⇒ No flag are affected.

Byte	Machine Cycle	T-state
1	1	4

(II) MOV (R-M)

* Syntax :- MOV Rd, m

⇒ The contained of the memory location [HL] are transfer to the destination register (Rd).

* Examples :- MOV B, HL [B] ← [HL]

Byte	Machine Cycle	T-state
1	2	7

(III) MOV (M-R)

* Syntax :- MOV M, Rs

⇒ The contained of the source register (Rs) are transfer into the destination memory.

Examples :- MOV HL, B [HL] ← [B]

Byte	Machine Cycle	T-state
1	2	7

(iv) (iv) MOV (M-M)

* Syntax:- MOV M, M

⇒ The content of memory is move to the another memory

* Examples:- MOV H, L [H] ← [L]

Byte	Machine Cycle	T-state
1	2	7

2. MVI (MOVE IMMEDIATE):-

* Syntax:- MVI R/M, 8 bit data

⇒ This instruction move the contain of 8-bit data into the register or memory.

* Example:- MVI R, 8 bit data
  MVI B, 23H [B] ← [23]
  MVI M, 8 bit data
  HL, 32H [HL] ← [32]

⇒ This is indirect addressing mode.

Byte	Machine Cycle	T-state
2	2	7

3. LXI (Load Register Pair Immidiate):-

* Syntax:- LXI RP, 16 bit data

⇒ This instruction Load the contain of 16-bit data into the register pair (RP).

* Example:- LXI B, 2000H

B	C
20	00

⇒ This is immidiate addressing mode,

Byte	Machine Cycle	T-state
3	3	10

↓

(OF + MR + MR)

4. LDA (Load Accumulator):

* Syntax :- LDA 16 bit address

⇒ This instruction load the contain of 16 bit address into the accumulator,

* Examples: LDA 6000H

[A] ← [32H]

⇒ This is direct addressing mode,

Byte	Machine Cycle	T-state
3	4	13

5. STA (Stone Accumulator):-

* Syntax :- STA 16 bit address

⇒ This instruction stone the condition of the accumulator into the 16-bit address,

* Examples:- STA 6000H

[A] → 32H

[A] → 6000H

⇒ This is a direct addressing mode,

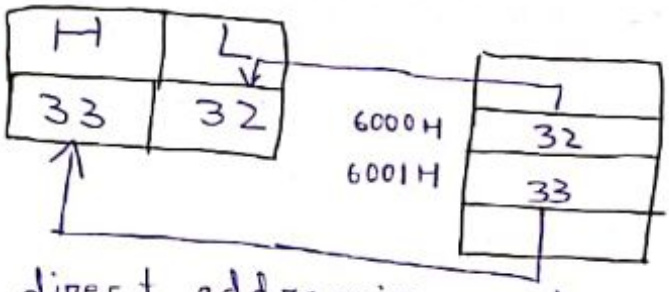
Byte	Machince Cycle	T-state
3	4	13

6. LHLD (Load H-L pair Direct):

* Syntax: LHLD 16 bit address

⇒ The content of memory location pointed out by 16-bit address are loaded into register L and the content of next memory location are loaded into register H.

* Examples:



⇒ This is direct addressing mode.

⇒ No flag are affected

Byte	Machine Cycle	T-state
3	5	16

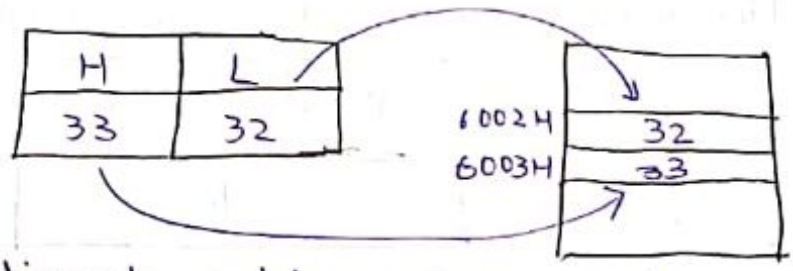
OF + 2MR + 2MR

7. SHLD (Store H-L pair Direct):-

* Syntax:- SHLD 16-bit address

⇒ The content of register L are stored at the memory location and the content of register H are stored at the next memory location by incrementing 1.

* Examples:-



⇒ This is direct addressing mode.

⇒ No flag are affected.

Byte	Machine Cycle	T-state
3	5	16

8. LDAX (Load Accumulator indirect):

* Syntax: LDAX B/D/ Register pair (R_P)

⇒ This instruction copies (load) an 8-bit data to the accumulator from the memory location pointed out by B-C register pair or D, E register pair.

* Examples: LDAX B

[A] ← [B, C]

LDAX D

[A] ← [D, E]

⇒ This is indirect addressing mode.

⇒ No flag are affected.

Byte	Machine Cycle	T-state
1	2	7

9. STAX (Store Accumulator Indirect):

* Syntax: STAX B/D R_P (Register pair)

⇒ This instruction will store the accumulator content in the memory location pointed out by register pair or D, E Register pair.

* Example: STAX B

[A] ⟶ [B, C]

STAX D

[A] → [D, E]

⇒ This is indirect addressing mode

⇒ No flag are affected.

Byte	Machine Cycle	T-state
1	2	7

10. XCHG (Exchange H-L pair with D-E Register pair)

* Syntax: XCHG

* Description

The content of the register H and register D are exchanged with content of register L and E is exchanged.

* Example

H = 50H	After XCHG operation →	H = 65 H
L = 20H		L = 26 H
D = 65H		D = 50 H
E = 26H		E = 20 H

⇒ No flag are affected.

Byte	Machine Cycle	T-state
1	1	4

ARITHMETIC GROUP:-

① ADD (ADD Register or Memory to the accumulator)

Syntax: ADD R/M

⇒ The content of register or memory are added to the content of accumulator and the result store in the accumulator.

⇒ If the operand is a memory location, that is indicated by the 16-bit address in the HL register.

⇒ All flag are modified to reflect the result of the addition.

Examples :-

ADD R

ADD B [A] ← [A+B]

ADD M

ADD HL [A] ← [HL+A]

⇒ This is a one byte instruction

	Byte	Machine Cycle	T-state
ADD R	1	1	4
ADD M	1	2	7

② ADI (ADD Immediate Data to Accumulator)

* Syntax: ADI 8-bit data

⇒ The 8 bit data are added to the content of accumulator and the result is stone in the accumulator

⇒ All flag are modified to reflect the operation.

* Examples: ADI 8 bit data

ADI 23H [A] ← [A+23H]

ADI 19H

19H + A ⟶ A

Byte	Machine Cycle	T-state
2	2	7

⇒ This is a 2-byte instruction.

③ ADC (Add register to accumulator with carry):-

* Syntax:- ADC R/m

⇒ The content of the operand (Register and memory) and carry flag are added to the content of the accumulator and the result is store in the accumulator.

* Example:-

ADC R [A] ← [Register + Carry + Accumulator]

ADC B [A] ← [B + Cy + A]

	Byte	Machine Cycle	T-state
ADC R	1	1	4
ADC M	1	2	7

④ ACI (Add Immediate Data to Accumulator with carry)

* Syntax:- ACI 8 bit data

⇒ The 8 bit data and the carry flag are added to the content of accumulator and store the result in the accumulator

* Example:- ACI 8 bit data

ACI 23H [A] ← [A + 23H + Cy]

⇒ All flag are modified to reflect the result of addition.

	Byte	Machine Cycle	T-state
ACI 23H	2	2	7

⇒ This is a 2-Byte instruction.

⑤ DAD (ADD register pair to H-L register):-

* Syntax:- DAD R_P $[HL] \leftarrow [HL + R_P]$

⇒ This instruction adds the register pair content with HL content & store the result in HL register pair.

* Example:-

DAD R_P

DAD BC $[HL] \leftarrow [[HL] + [BC]]$

DAD DE $[H\text{-}L] \leftarrow [DE] + [HL]$

Byte	Machine Cycle	T-state
1	3	10

⑥ SUB (Subtract Register on memory from accumulator)

* Syntax:- SUB R/M

⇒ The content of register in memory are subtracted from the context of accumulator and result store in the accumulator.

⇒ All flag are affected to reflect the result of the subtraction.

* Example:

SUB R

SUB B $[A] \leftarrow [A - B]$

SUB M $[A] \leftarrow [A - HL]$

	Byte	Machine Cycle	T-state
SUB R	1	1	4
SUB M	1	2	7

⑦ SBB (Subtract source and Borrow from ACC)

* Syntax: SBB R/m

⇒ The content of the operand (register or memory) and the borrow flag are subtracted from the content of the accumulator and the result are stored in the accumulator.

* Examples:

SBB R

SBB B [A] ← [A-B-Br]

SBB M [A] ← [A ~~←~~ A-HL-Br]

⇒ All flag are modified to reflect the result of operation

	Byte	Machine Cycle	T-state
SBB R	1	1	4
SBB M	1	2	7

⑧ SBI (Subtract Immidiate data with borrow from the accumulator)

* Syntax: SBI 8-bit data

⇒ The 8-bit (operand) and the borrow are subtracted from the content of the accumulator and the result are stone in the accumulator.

⇒ All flag are altered to reflect the result of the operation.

* Examples:

SBI 8 bit data

SBI 23H [A] ← [A -23H-Br]

Byte	Machine Cycle	T-state
2	2	7

(9) SUI (Subtract Immediate from Accumulator)

* Syntax: SUI 8 bit data

⇒ The 8 bit data (operand) are subtracted from the content of the accumulator and the result are store in the accumulator.

⇒ All flag are modified to reflect the result of subtraction

* Examples:

SUI 8 bit data

SUI 23H [A] ← [A-23H]

Byte	Machine Cycle	T-state
2	2	7

(10) INR (Increment content of R/M by 1)

* Syntax: INR R/M

⇒ The content of register/memory are increment by 1 and the result are stored in the same place. If the operand is a memory location, it is specified by the content of HL Register

* Examples:-

INR R

INR B

If B = 43H

INR B

⇒ B = 44H [B] ← 44H

If M = 42H

INR M

HL = 43H

[HL] ← 43H

*Flag :- Sign Flag, zero flag, auxillary carry flag, parity flag are modified to reflect of operation but carry flags are not modified.

	Byte	Machine Cycle	T-state
INR R	1	1	4
INR M	1	3	10

(11) INX (Increment Register Pair By 1)

* Syntax: INX Rp

⇒ The content of register pair are incremented by 1 and the result is stone in the same place.

* Examples: INX Rp

9h B.C = 2050

INX. B → BC ← 2051

B C

20 51

* Flag: No flag are effected.

	Byte	Machine Cycle	T-state
INX Rp	1	1	6

(12) DCR (Decrement Source by 1) or (Decrement Content register/memory by 1)

* Syntax :- DCR R/M

⇒ The content of register/memory is decremented by one and the result are stored in the same place.

⇒ If the operand is a memory location, it is specified by the content of HL Register pair.

✳ Examples:- DCR R

If B = 19H

DCR B

⇒ B = 18H

⇒ [B] ← [18]

DCR M

If HL = 18H

DCR HL

⇒ HL = 17H

⇒ [HL] ← [17]

✳ Flag: Sign Flag, Zero Flag, Parity Flag, auxilary carry flag are modified to ~~result~~ reflect the result of the operation, carry flag is not modified.

	Byte	Machine Cycle	T-state
DCR R	1	1	4
DCR M	1	3	10

(13) DCX (Decrement Register pair by 1)

✳ Syntax: DCX Rp

⇒ The content of register pair are decremented by one and the result are stone in the same place.

✳ Example: DCX Rp

If D = 2052

DCX - D · DE = 2051

D	E
20	51

✳ Flag: No flag are affected.

	Byte	Machine Cycle	T-state
DCX Rp	1	1	4

(4) DAA (Direct Adjust Accumulator)

* Syntax: DAA
* Descriptions

⇒ The content of the accumulator are change from a binary value to 4 bit BCD digit. This is the only instruction that uses the AC to perform BCD conversion.

⇒ Rules:

→ If the value of lower order 4-bit (D_3-D_0). In the accumulator is greater than 9 or if the auxilary carry flag is set then the instruction add 6 to the low order 4-bit.

→ If the value of higher order 4-bit i.e. D_7 to D_4 in the accumulator is greater than 9 or carry flag is set, the instruction add 6 to high-order 4-bit.

* Flag: Sign Flag, Zero Flag, Auxilary Carry Flag, Carry Flag, Parity Flag are alter to reflect the result of the operation.

Byte	Machine Cycle	T-state
1	1	4

*Example:-

(Lower Byte)

39 → 0011 1001
12 → 0001 1000
0100 1011
4 B(11)
0000 0110
0101 0001
5 1

(Higher Byte)

55 → 0101 0101
55 → 0101 0101
1010 1010
A A
0110 0110
Cy ← [1] 0001 0000
1 0

85 → 1000 1101
68 → 0110 1000
1110 1101
14(1) 13(D)
0110 0110
Cy ← [1] 0101 0011
5 3

LOGICAL GROUP:-

① ANA (Logical AND with accumulator)

* Syntax: ANA R/M
* Description:
⇒ The content of the accumulator are logically AND with the content of the operand (R/M) and the result is stone in the accumulator.
⇒ If the operand is a memory location, its address is specified by the content of 'HL' Register
* Example: ANA R
ANA B [A] ← [A] ∧ [B]
ANA M [A] ← [A] ∧ [HL]

*Flag: Sign Flag, Zero Flag, Parity Flag are modified to reflect the result of the operation, Carry Flag is reset and Auxilary Carry is set.

	Byte	Machine Cycle	T-state
ANA R	1	1	4
ANA M	1	2	7

② ANI (AND Immediate data with Accumulator)

*Syntax: ANI 8 bit data

*Description:

⇒ The content of the accumulator are logically AND with the 8 bit data and the result is store in the accumulator.

*Example: ANI 8 bit data

ANI 23H [A] ← [A] ∧ [23H]

*Flag: Sign Flag, Zero Flag, Parity Flag, are modified to reflect the result of the operation, Carry flag is reset, Auxilary Carry is set.

	Byte	Machine Cycle	T-state
ANI 23H	2	2	7

③ ORA (Logically OR with Accumulator)

*Syntax: ORA R/M

*Description: ~~The~~

⇒ The content of the accumulator are logically OR with the content of operand (R/m) and the result is store in accumulator

*Example: ORA R
ORA B
ORA M

[A] ← [A] V [B]
[A] ← [A] V [HL]

*Flag: Zero Flag, Sign Flag, Parity Flag are modified to reflect the result of the operation, Auxilary Carry, Carry Flag are reset.

	Byte	Machine Cycle	T-state
R	1	1	4
M	1	2	7

④ ORI (OR Immediate data with Accumulator)

*Syntax: ORI 8 bit data

*Description:

⇒ The content of the accumulator are Logically OR with the 8 bit ~~address~~ data in the operand and the result are placed in the accumulator.

*Example: ORI 8 bit data
ORI 24H [A] ← [A] V [24H]

*Flag: Sign Flag, Zero Flag, Parity Flag are modified to reflect the result of the operation. Carry Flag and Auxilary Carry Flag are reset.

	Byte	Machine Cycle	T-state
ORI 24H	2	2	7

⑤ XRA (Exclusive OR with Accumulator)

*Syntax: XRA R/M

*Description:

⇒ The content of the operand (R/M) are exclusive OR with the content of the accumulator and the result are stored in the accumulator.

⇒ The content of the operand are not allowed.

* Example: XRA R
  XRA B [A] ← [A] ∀ [B]
  XRA M
  XRA HL [A] ← [A] ∀ [HL]

* Flag: Zero Flag, Sign Flag, Parity Flag are altered to reflect the result of the operation, Canny flag and auxilary canny flag are reset

	Byte	Machine Cycle	T-state
XRA R	1	1	4
XRA M	1	2	7

⑥ XRI (Exclusive OR immediate data with accumulator):

* Syntax: XRI 8 bit data
* Description:

⇒ The 8 bit data are exclusive OR with the contents of the accumulator and the result are stored in the accumulator.

* Example: XRI 8 bit data
  XRI 40H [A] ← [A] ∀ [40H]
* Flag: Sign Flag, Zero Flag, Parity Flag are attended to reflected the result of the operation Canny Flag and Auxilary Canny Flag reset.

	Byte	Machine Cycle	T-state
XRI 8 bit data	2	2	7

⑦ CMP (Compare with Accumulator)

* Syntax: CMP R/M
* Description:

⇒ The content of the operand (R/M) are compared with the contents of the accumulator. Both contents are presented and the comparision shown by setting the flag.

* Example:

Cy	Z	Comment
0	0	A > R/M
0	1	A = R/M
1	0	A < R/M
1	X	A ≥ R/M

* Flag: Sign Flag, Parity Flag, Auxilary Carry Flag are also modified in addition to Zero flag and Carry flag to reflect of the operation.

	Byte	Machine Cycle	T-state
CMP R	1	1	4
CMP M	1	2	7

⑧ CMI (Compare with Accumulator Immediate)

* Syntax: CMI 8 bit data
* Description:

⇒ The content of 8-bit data in compare with the content of accumulator.

⇒ The resulting of comparision are indicated by setting the flag.

* Flag:

⇒ Sign flag, parity flag, auxilary carry flag are modified to reflect the result of operation,

⇒ But carry flag and zero flag are not ~~re~~modified to reflect the result of operation

* Example:

Carry Flag	Zero Flag	Comments
0	0	A > 8-bit data
0	1	A = 8-bit data
1	0	A < 8 bit data
1	1	A ≥ 8 bit data

	Byte	Machine Cycle	T-state
CMI 8-bit data	2	2	7

⑨ CMA (Complement with Accumulator)

* Syntax: CMA

* Description:

⇒ This instruction with complement the content of accumulator and result stored in the accumulator.

* Example: If ~~A~~ = CMA A = 64H = 0110 0100

CMA A ⇒ 1001 1011 → [A]

* Flag: No flag are affected.

	Byte	Machine Cycle	T-state
CMA	1	1	4

(10) CMC (Complement with carry flag)

* Syntax: CMC
* Description:

⇒ This instruction will complement with the content of carry flag and result store in the carry flag.

* Example:

If C = 24H ⇒ 0010 0100

⇓ Complement

1101 1011 → [CS]

* Flag: Carry flag are modified and no other flag are affected.

	Byte	Machine Cycle	T-state
CMC	1	1	4

SHIFT OR ROTATION GROUP :-

The instruction of this group perform rotation i.e. left of the content of accumulator.

Example: RAR (Rotate Accumulator Right)
RLC (Rotate Acccumulator left with carry)

(1) RAL (Rotation Accumulator Left)

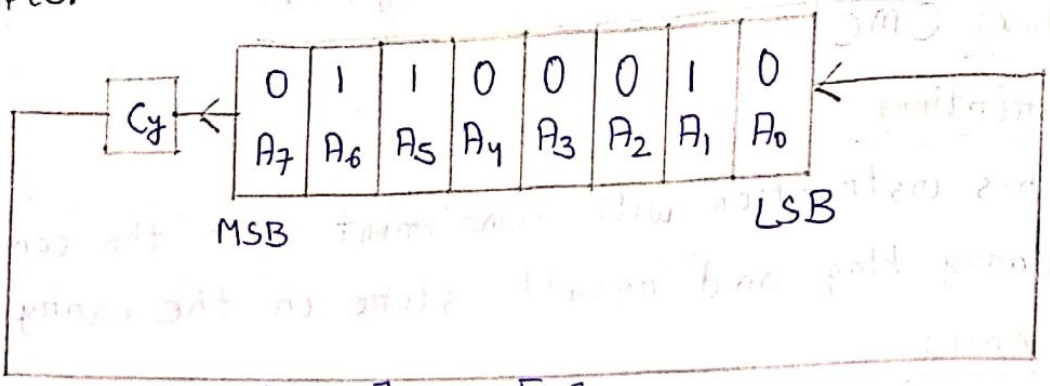
* Syntax: RAL
* Description:

⇒ Each binary of accumulator is rotate left by one position through the carry flag. Bit A_7 is placed in the bit carry flag and the carry flag is placed in the LSB.

* Flag:

⇒ The carry flag is modified according to the bit of A_7. Rest flags are not modified.

* Example:



[An+1] ← [An]

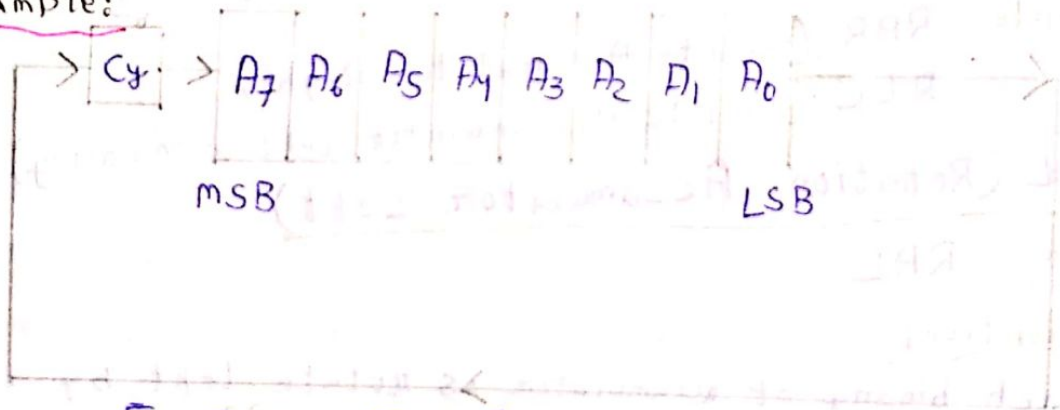
	Byte	Machine Cycle	T-state
RAL	1	1	4

(2) RAR (Rotate Accumulator Right)

* Syntax: RAR
* Description:

⇒ Each binary bit of accumulator is rotate right by one position through carry flag. "A0" is placed in the carry flag and the bit in the carry flag is placed in the MSB.

* Example:



[An] ← [An+1]

* Flag: Carry flag is modified according to the bit of A0, Sign flag, zero flag, parity flag, auxiliary carry flag are not affected.

	Byte	Machine Cycle	T-state
RAR	1	1	4

③ RLC (Rotate accumulator left but not through carry)

* Syntax: RLC
* Description:

⇒ Each binary bit of the accumulator is rotated left by one position but A_7 is placed in the position A_0.

* Flag: Carry flag is modified accordingly. Rest are not modified.

	Byte	Machine Cycle	T-state
RLC	1	1	4

④ RRC (Rotate Accumulator right but not through carry):

* Syntax: RRC
* Description:

⇒ Each binary bit of the accumulator is rotated right by 1 position. Bit A_0 is placed in the position A_7 as well as in the carry flag.

* Flag: Carry flag is modified according to the bit A_0. Rest are not modified.

	Byte	Machine Cycle	T-state
RRC	1	1	4

Branch Control Group:-

⇒ It is sequential machine, executing there code from one memory location to another location,

⇒ Branch instruction are more powerful because they allows the microprocessor to change the sequence of a program either conditionaly or unconditionaly

⇒ Branch instruction instruct the microprocessor to go a different memory location and microprocessor continue to execute a machine code from a new location.

⇒ The branch instruction ~~sets~~ are classified into three categories:

i) JUMP instruction
ii) CALL instruction
iii) RETURN instruction

i) JUMP instruction

⇒ This instructions are classified into two types

a) Conditional Jump
b) Unconditional Jump

a) Conditional Jump

⇒ This is 8 instruction allows the microprocessor to made the decision based on the certain condition i.e. intiated by flag.

⇒ After the logical or mathematical operation the flags are set or reset reflect the data condition.

⇒ The conditional jump instruction check the flag condition and make the decision to change or not to change the sequence of program.

⇒ In 8085 microprocessor, five flag are present out of five flag. Auxiliary Carry Flag is used as a internally to the microprocessor.

Syntax:

JMP 16-bit address

Description:

⇒ This instruction transfer the program sequence to the memory location specified under the flag condition.

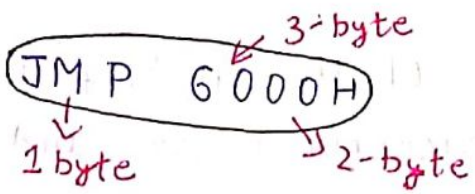
⇒ This is 3 byte instruction where

i) First byte is for opcode/operation code.

ii) Second byte is for specify low order memory address

iii) Third byte is for specify high order memory address

Example:



b) Unconditional Jump

⇒ The 8085 instruction set include one unconditional jump.

⇒ The unconditional jump instruction enables the programmer to set of continuous loop.

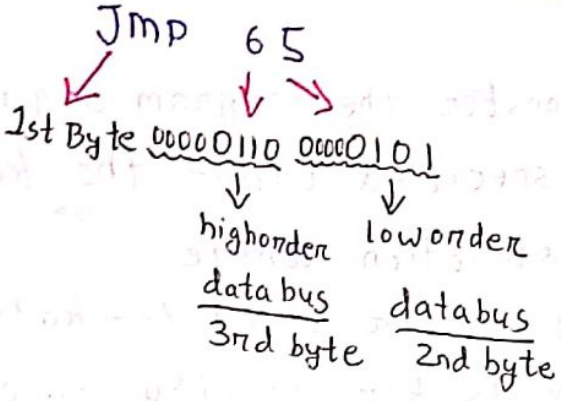
Syntax: JMP 16-bit address

Description:

⇒ The program sequence is transfer to the memory location specified by 16-bit address in the operand.

⇒ This is 3 byte instruction. First byte for op-code followed 16-bit memory address and the second and third byte specified to 16-bit memory address. The second byte specify low-order address and third-byte specify high-order memory address.

Example: JMP 16 bit address



⑩ JP (Jump on positive)

Syntax: JP 16-bit address

Description:

⇒ Jump is effective, if the sign flag is reset otherwise Continue in the microprocessor and data will be stored in the PC/main program.

Example:

JP 6000H

[PC] ← 6000H

⑪ JN (Jump on Negative)

Syntax: JN 16-bit address

Description:

⇒ Jump is effective if the zero flag is set otherwise continue in the microprocessor and stored in the main program/PC.

Example: JN 6000H

[PC] ← 6000H

(III) JZ (Jump on Zero)

Syntax: JZ 16-bit address

Description:

⇒ Jump is effective if the zero flag is set otherwise continue in the microprocessor and stored in PC.

Example: JZ 6000H

[PC] ← 6000H

(IV) JNZ (Jump on No Zero)

Syntax: JNZ 16-bit address

Description:

⇒ Jump is effective if the zero flag is reset otherwise continue in the microprocessor and data will be stored in the PC

Example: JNZ 6000H

[PC] ← 6000H

(V) JPE (Jump on Parity Even)

Syntax: JPE 16 bit addresss

Description:

⇒ Jump is effective if the parity flag is set otherwise Continue in the microprocessor and data will be stored in the PC.

Example: JPE 6000H

[PC] ← 6000H

(VI) JPO (Jump on Parity Odd)

Syntax: JPO 16 bit address

Description:

⇒ Jump is effective if the parity flag is set otherwise continue in the microprocessor and data stored in the PC.

Example: JPO 6000H

[PC] ← 6000H

(VII) JC (Jump on Canny)

Syntax: JC 16-bit address

Description:

⇒ Jump is effective if the canny flag is set, otherwise continue in the microprocessor and stored in the PC.

Example: JC 6000H

[PC] ← 6000H

(VIII) JNC (Jump on No canny)

Syntax: JNC 16-bit address

Description:

⇒ Jump is effective if the canny flag is reset otherwise continue in the microprocessor and stored in the PC.

Example: JNC 6000H

[PC] ← 6000H

ii) Call Instruction

⇒ It is used in main program to call a sub-routine when a sub-routine is call the content of program counter store on the stack and the program ~~counter~~ execution is transfer to the sub-routine address.

⇒ Call instruction is divided into two types:

a) Conditional Call

b) Unconditional Call

a) Conditional Call

⇒ The instruction is used conditionally to call a subroutine. This instruction transfer the program sequence to the subroutine address

⇒ The instruction set the content of program Counter on the stack and the stack pointer decremented by 2 and data will be stored in the program counter.

Syntax: CALL 16 bit address

Description:

⇒ This is 3byte instruction. 1st byte for opcode, 2nd byte for low order memory address, 3rd byte for high order memory address,

⇒ If the condition is true, then five machine cycle and eighteen T-state are used, if the condition is false then two machine cycle and nine T-state are used.

Example: CALL 6000H
[PC] ← 6000H

b) Unconditional Call

⇒ This instruction is used unconditionally to call a subroutine. This instruction transfer the program sequence to a subroutine address.

⇒ This instruction set the content of program counter on the stack and the stack pointer decremented by 2 and data will be stored in the program counter

Syntax: CALL 16-bit address

Description:

⇒ This is 3-byte instruction where 1st byte is for operation code, 2nd byte for low order memory address, 3rd byte for high order memory address.

⇒ No flag are affected.

Example:

CALL 6000H
[PC] ← 6000H

Byte	M/C	T-state
3	5	18

① CP (Call on Positive)

Syntax: CP 16-bit address

Description:

⇒ Call is effective if the sign flag is reset otherwise continue in the microprocessor and data will be stored in the program counter.

Example: CP 6000H

[PC] ← 6000H

(II) CN (Call on Negative)

Syntax: CN 16-bit address

Description:

⇒ Call is effective, if the sign flag is set otherwise continue in the microprocessor and data will be stored in the program counter.

Example: CN 6000H

[PC] ← 6000H

(III) CZ (Call on Zero)

Syntax: CZ 16 bit address

Description:

⇒ Call is effective, if the zero flag is set otherwise continue in the microprocessor and data will be stored in the program counter

Example: CZ 6000H

[PC] ← 6000H

(IV) CNZ (Call on No zero)

Syntax: CNZ 16-bit address

Description:

⇒ Call is effective, if the zero flag is reset otherwise continue in the microprocessor and data will be stored in the program counter (PC)

Example: CNZ 6000H

[PC] ← 6000H

(V) CPE (Call ~~Canny~~ on Parity Even)

Syntax: CPE 16-bit address

Description:

⇒ Call is effective if the parity flag is set otherwise continue in the microprocessor and data will be stored in the program counter.

Example: CPE 6000H

[PC] ← 6000H

(VI) CPO (Call on Parity Odd)

Syntax: CPO 16-bit address

Description:

⇒ Call is effective if the parity flag is ~~set~~ reset otherwise continue in the microprocessor and data will be stored in the program counter (PC).

Example: CPO 6000H

[PC] ← 6000H

(VII) CC (Call on Canny)

Syntax: CC 16 bit address

Description:

Call is effective if the canny flag is set otherwise continue in the microprocessor and data will be stored in the program counter (PC).

Example: CC 6000H

[PC] ← 6000H

(VIII) CNC (Call on No Carry)

Syntax: CNC 16-bit address

Description:

⇒ Call is effective if the carry flag is reset otherwise continue in the microprocessor and data will be stored in the program counter (PC).

Example: CNC 6000H

[PC] ← 6000H

iii) Return Instruction

⇒ The Return instruction is used at the end of the subroutine to return to the main program.

⇒ When the Return instruction is executed at the end of the subroutine, the memory address stored on the stack is retenive and ~~an~~ the Sequence of execution is stay in the main program.

⇒ Return instruction are two types:

~~No~~ Conditional Return

ii) Unconditional Return

a) Conditional Return

⇒ Before the execution of subroutine the adress of next instruction of main program is save on the stack through the conditionally and data will be stored in the program counter (PC).

Syntax: RET 16-bit address

Example: RET 6000H

[PC] ← 6000H

Description:

⇒ If the conditional is true and the programme return from the subroutine the execution of a conditional return take 1a ~~one~~ machine cycle 2 T-state

⇒ If the condition not true only one machine cycle 6 T-state (OF-4 + SP-2)

b) Un-conditional Return

Syntax: RET 16 bit address

Description:

⇒ Before the execution of the sub routine the address of the next instruction of the main program is saved in the stack.

⇒ Then the execution of return instruction bring back the saved address from the stack to the program counter (PC),

⇒ The content of the stack pointer incremented by two to indicate the new stack top,
Then the program JUMP to the instruction to the main program next to call instruction called subroutine

⇒ No flags are affected.

Byte	m/c Cycle	T-state
1	3	10

⇒ This is indirect addressing mode.

Example: RET 6000H

[PC] ⟵ 6000H

① RP (Return on Positive)

Syntax: RP 16-bit address

Description:

⇒ Return is effective if the sign flag is reset otherwise continue in the microprocessor and data will be stored in the program counter (PC).

Example:

RP 6000H

[PC] ⟵ 6000H

② RN (Return on Negative)

Syntax: RN 16 bit address

Description:

⇒ Return is effective if the sign flag is set otherwise continue in the microprocessor and data will be stored in the program counter (PC).

Example: RN 6000H

[PC] ⟵ 6000H

③ RZ (Return on Zero)

Syntax: RZ 16-bit address

Description:

⇒ Return is effective if the zero flag is set otherwise continue in the microprocessor and data will be stored in the program counter (PC).

Example: RZ 6000H

[PC] ⟵ 6000H

④ RNZ (Return on No Zero)

Syntax: RNZ 16-bit address

Descriptions

⇒ Return is effective if the zero flag is reset otherwise continue in the microprocessor and data will be stored in the program counter.

Example: RNZ 6000H

[PC] ← 6000H

⑤ RPE (Return on Parity Even)

Syntax: RPE 16-bit address

Description:

⇒ Return is effective if the parity flag is set otherwise continue in the microprocessor and data will be stored in program counter (PC)

Example: RPE 6000H

[PC] ← 6000H

⑥ RPO (Return on Parity Odd)

Syntax: RPO 16-bit address

Description:

⇒ Return will effective if the parity flag is reset otherwise continue in the microprocessor and data will be stored in the program counter (PC)

Example: RPO 6000H

[PC] ← 6000H

⑦ RC (Return on Carry)

Syntax: RC 16-bit address

Description:

⇒ Return is effective if the carry flag is reset otherwise continue in the microprocessor and data will be stored in the program counter.

Example: RC 6000H

[PC] ← 6000H

⑧ RNC (Return No Carry)

Syntax: RNC 16-bit address

Description:

⇒ Return is effective if the carry flag is reset otherwise continue in the microprocessor and data will be stored in the program counter (PC).

Example: RNC 6000H

[PC] ← 6000H

⑨ RST (Restart) Instruction

Syntax: RST n

Description:

⇒ Restart instruction are equivalent to 1 byte CALL instruction.

⇒ In this instruction, the content of program counter is a set on the stack.

⇒ The program JUMP to the instruction starting at Resart Location. The address of restart instruction is 8 times (0-7).

⇒ This is indirect addressing mode,

Byte	Machine Cycle	T-state
1	3	12

INSTRUCTION	RESTART LOCATION
RST 0	0000
RST 1	0008
RST 2	0010
RST 3	0018
RST 4	0020
RST 5	0028
RST 6	0030

MACHINE CONTROL INSTRUCTION:

NOP (No Operation)

Syntax: NOP

Description:

⇒ No operation is performed when the instruction executed

⇒ The register and flag are not affected,

Byte	Machine Cycle	T-state
1	1	4

HLT (Stop Instruction):-

Syntax: HLT

Description:

⇒ This instruction stop the execution of microprocessor.

⇒ No flags are affected.

Byte	Machine Cycle	T-state
1	2	5

XTHL (Exchange H-L with top of the stack)

(Exchange the exact register pair with the top of the stack)

Syntax: XTHL

[L] ↔ [SP]

[H] ↔ [SP+1]

Description:

⇒ The content of register L are exchange with the top of the stack,

⇒ The content of register L are exchange with below of the stack top,

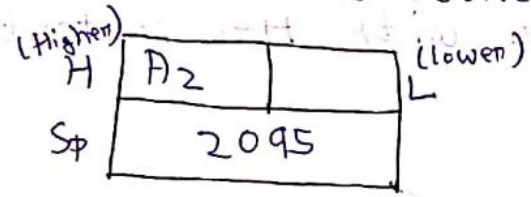
⇒ This is resister indirect addressing mode,

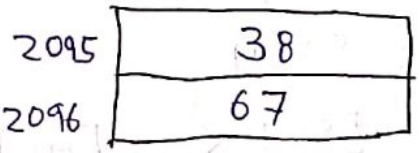
⇒ No flags are affected.

Byte	Machine Cycle	T-state
1	5	16

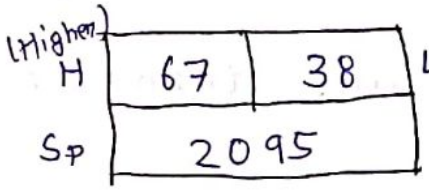
Example:

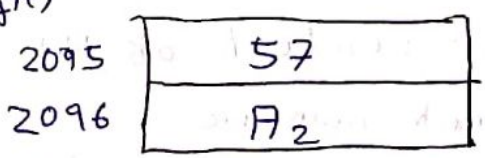
Before XTHL between





After XTHL instruction





PCHL (Program Counter H-L):-

(Load PC with H-L Content):-

Syntax:- PCHL

[PC] ← [HL]

[PCH] ← [H]

[PCL] ← [L]

Description:-

⇒ The content of H-L pair are transferred to the program counter

⇒ The content of H are move to the high order (H) 8 bit of the program counter and the content of L are move to the low order (L) 8 bit of the program counter.

⇒ No flags are affected.

Byte	Machine Cycle	T-state
1	1	6

SPHL (Load Stack pointer with H-L content):-

Syntax: SPHL

[SP] ← [HL]

Description:

⇒ The content of HL pair are transfer to the Stack pointer

⇒ No flags are affected,

Byte	Machine Cycle	T-state
1	1	6

INPUT OUTPUT INSTRUCTION:-

IN INSTRUCTION (Input to accumulator from 8-bit data)

Syntax: IN 8 bit port address

[A] ← [port]

Description:

⇒ The data are available on the port is move to the accumulator.

⇒ After the IN instruction the address of port is specified. The second byte of the instruction content address of the port,

⇒ No flags are affected.

⇒ This is a direct addressing mode.

Byte	Machine Cycle	T-state
2	3	10

OUT Instruction (Output from accumulator to 8-bit data):-

Syntax: OUT 8 bit port address

[A] ⟶ [port]

Description:

⇒ The content of accumulator is move to the 8 bit port address

⇒ No flags are affected.

Byte	Machine Cycle	T-state
1	1	4

STACK RELATED INSTRUCTION:

i) PUSH

Syntax: PUSH R_p

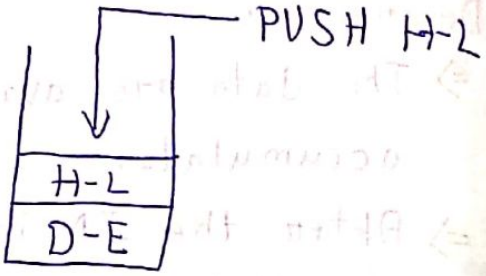
Description:

The PUSH the content of register pair into the stack.

Example:

PUSH H-L

W-Z D-E H-L



ii) POP

Syntax: POP R_p

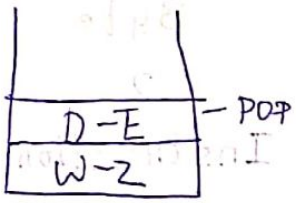
Description:

POP the content of register pair into the stack

Example:-

POP D-E

W-Z



iii) EI (Enable Interrupt)

Syntax: EI

Description:

⇒ When this instruction is executed the interrupt are active to the enable suggestion.

⇒ No flags are affected.

Byte	Machine Cycle	T-state
1	1	4

iv) DI (Disable Interrupt)

Syntax: DI

Description:

⇒ When this instruction is executed the interrupt are deactive to the enable suggestion

⇒ No flags are affected.

Byte	Machine Cycle	T-state
1	1	4

v) SIM (Set Interrupt Mask)

Syntax: SIM

Description:

⇒ When this instruction is executed the bit of the accumulator are used in the program.

⇒ No flags are affected,

Byte	Machine Cycle	T-state
1	1	4

vi) RIM (Read Interrupt Mask)

Syntax: RIM

Description:

⇒ .....

⇒ No flags are affected.

Byte	Machine Cycle	T-state
1	1	4

STC (Set Canny)

Syntax: STC

Description:

⇒The canny flag can be set using this instruction

⇒No flags are affected.

Byte	Machine Cycle	T-state
1	1	4

Assembler Directive

⇒ Assembler directive is a message that is tells assembler where the program is located in the memory.

⇒ It is also called as Pseudo.

ORG (Origin)

Syntax: ORG

Description:

⇒The next instruction or data is to be stored in the memory location storing at 6000H

Example: ORG 6000H

END (End of assembly)

Syntax: END

Description:

⇒It is the end of assembly program,

EQU (Equate)

⇒It is equate to variable name to a numeric value on a value name.

DS (Define Storage)

→ The directive define amount of free space.

TIMING DIAGRAM

What do you mean by Timing Diagram?

The necessary step, which are carried out in machine cycle can be represented graphically such graphically representation called timing diagram.

What do you mean by Machine Cycle? (MC)

It is defined as the time required to complete one operation of acessing memory, accessing input and output data etc.

T-State

⇒ T-state is defined as one sub-division of the operation perform in an one clock period.

⇒ $T\text{-state} = \frac{1}{f}$ [F = clock frequency]

$T\text{state} = 0.02040$

⇒ T-state is fully depends open the clock frequency

⇒ Each t-state = one o clock period.

Fetch Cycle (FC)

It is a standard process which is needed for processing of a data. It is called fetch cycle. And otherwise it is known as fetch decode execute cycle.

Instruction Cycle (IC)

⇒ It is the time taken to compute the execution of an instruction. Instruction cycle is the combination of fetch Cycle (FC) and execution cycle (EC)

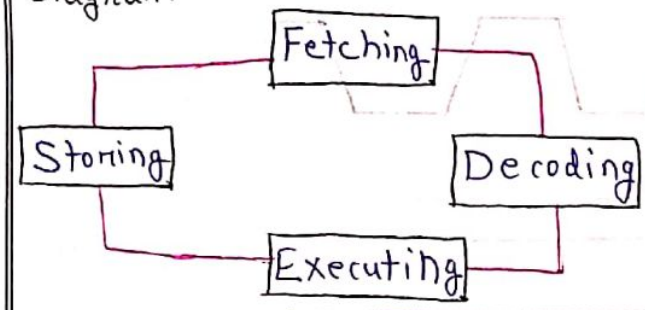
$$IC = FC + EC$$

⇒ The necessary steps which are carried out to get the data from the memory and execute it constitute to a fetch cycle.

Machine Cycle

⇒ It is defined as the time required to complete one operation to accessing memory, accessing input and output data is called machine cycle

Diagram



Process

Fetching, decoding, execution, storing

Memory

Machine Cycle does have memory capability

Components

Memory Unit (MU) (CPU) etc.

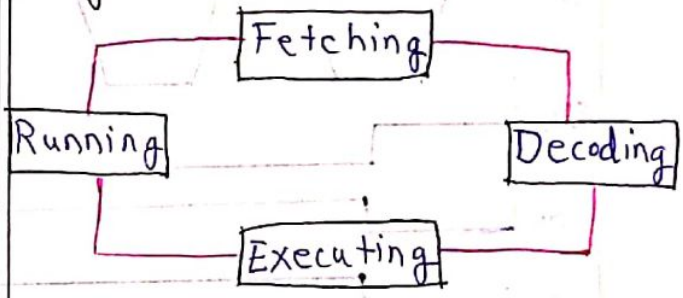
Memory Size:

Machine cycle considering more memory space than IC

Instruction Cycle

⇒ It is the time taken to compute the execution of an instruction. Instruction cycle is also fined as combination of FC & EC.

Diagram



Process

Fetching, decoding, execution, running.

Memory

Instruction Cycle does not have memory capability

Components

Register, ALU, etc.

Memory Size:

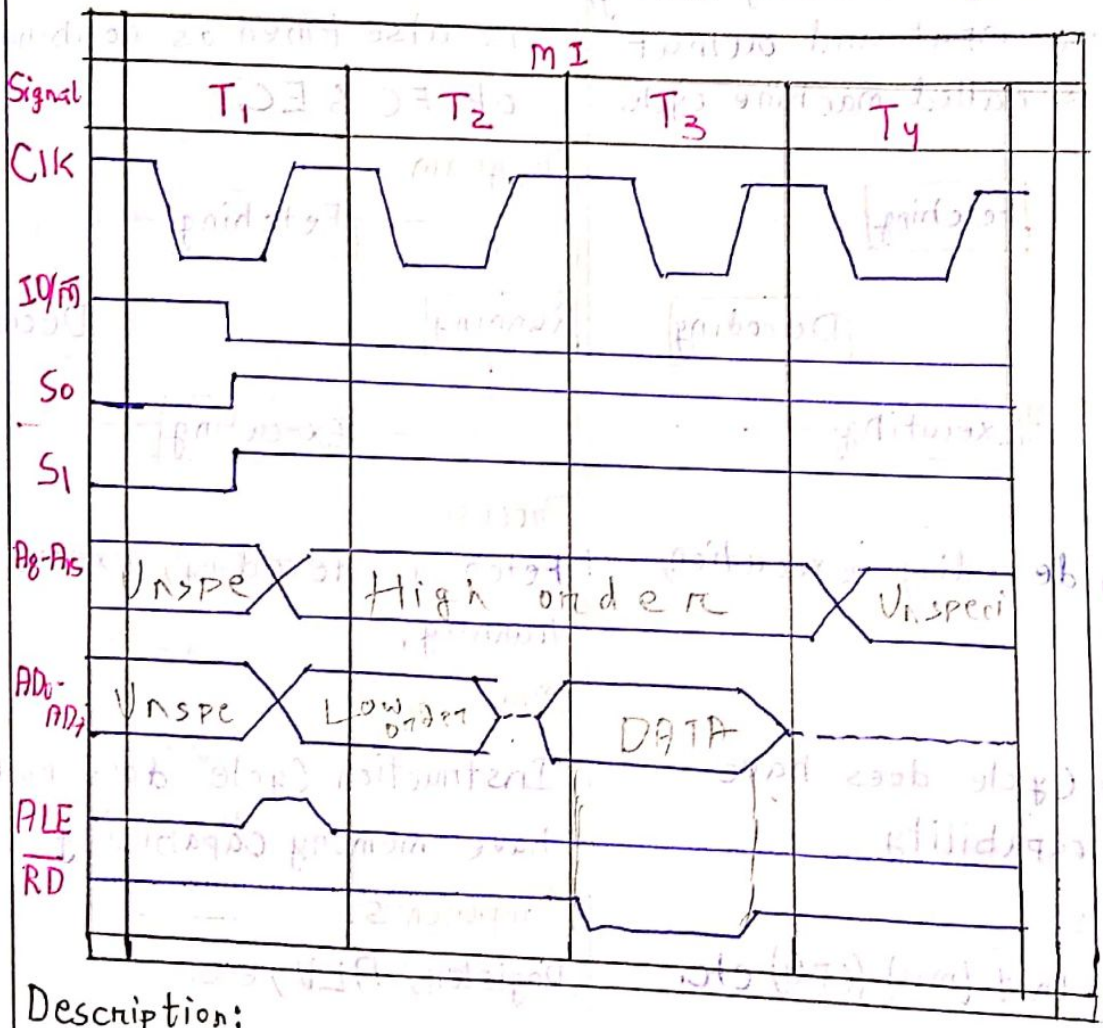
Instruction cycle consider less memory space than MC

Clock Cycle (cc)

⇒ The speed of a computer CPU is determined by clock cycle. The clock cycle is measured in Hertz(Hz).

⇒ Computer processor can execute one or more instruction per clock cycle depending upon the type of processor.

Timing Diagram for Opcode Fetch Cycle



Description:

⇒ In a fetch cycle, the microprocessor fetches the opcode of an instruction from the memory and above timing diagram for an opcode fetch cycle

⇒ T1, T2, T3, Ty timing diagram for an opcode fetch cycle. T1, T2, T3, Ty are consecutive four clock cycle.

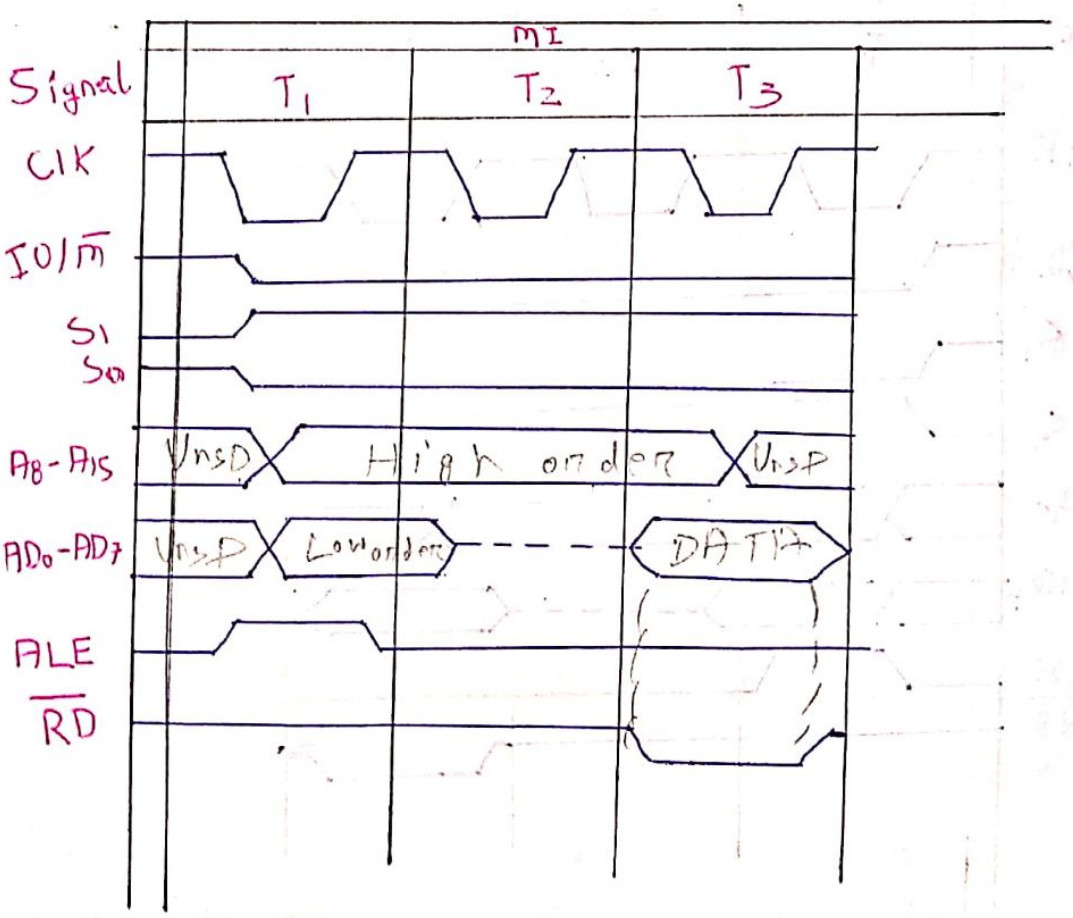
⇒ The microprocessor issues a low $IO/\overline{M}$ signal to indicate that its want to make communication with the memory.

⇒ Again the microprocessor, sends out high S_0 and S_1 signal to indicate that it is going to perform fetch operation.

⇒ $A_8 - A_{15}$, $A_0 - A_7$ depends upon the data which will be carried out on the opcode fetch cycle

⇒ ALE transfer address latch enable and it is mainly used for transfer the data from one location to another location. After that read operation ($\overline{RD}$) will be perform.

Timing Diagram for Memory Read Operation



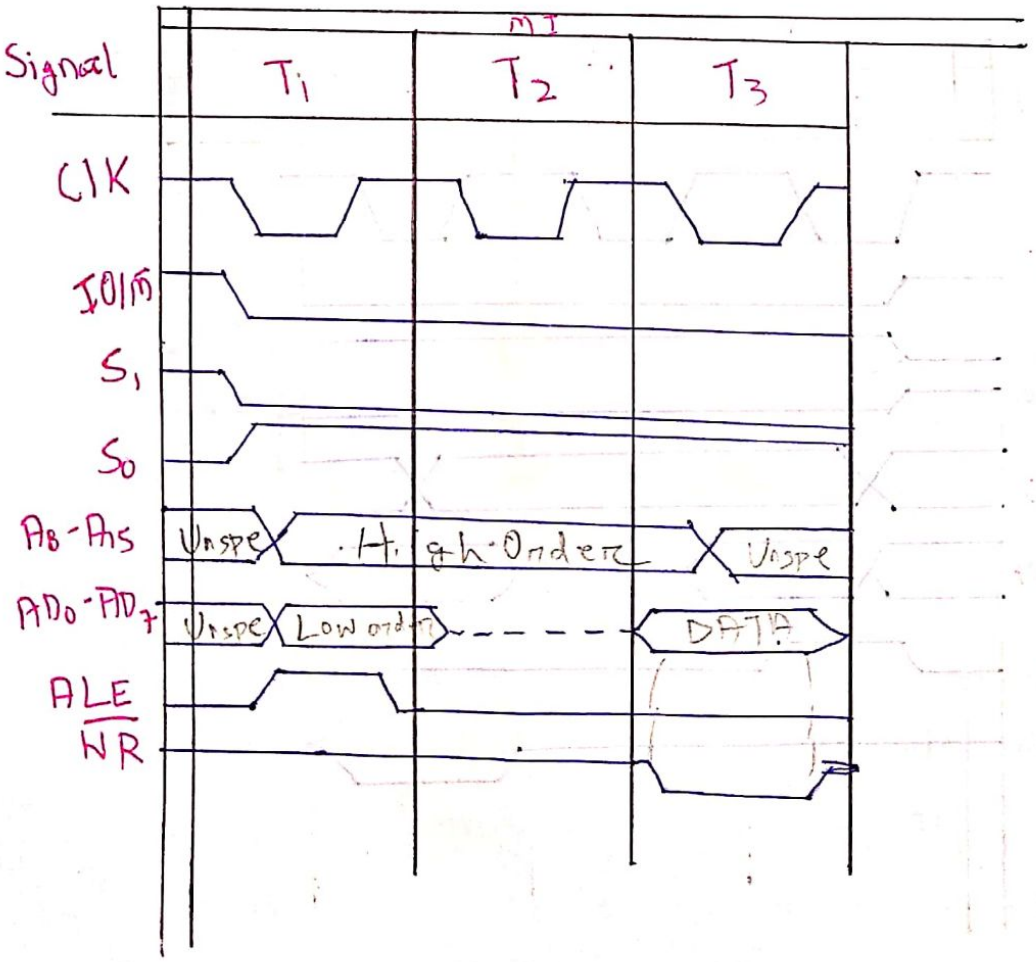
Description:

⇒ The processor takes three T-states to execute the cycle for memory read operation. And seven signals are used to read the memory operation and that is CLK, $IO/\overline{M}$, S_0, S_1, A_8-A_{15}, A_0-A_7, ALE, $\overline{RD}$

⇒ The memory read machine cycle is executed by the process or to read a data byte from memory.

⇒ The instruction which have more than 1 byte word size will use the machine cycle after the opcode the machine cycle.

Timing Diagram for Memory Write Operation



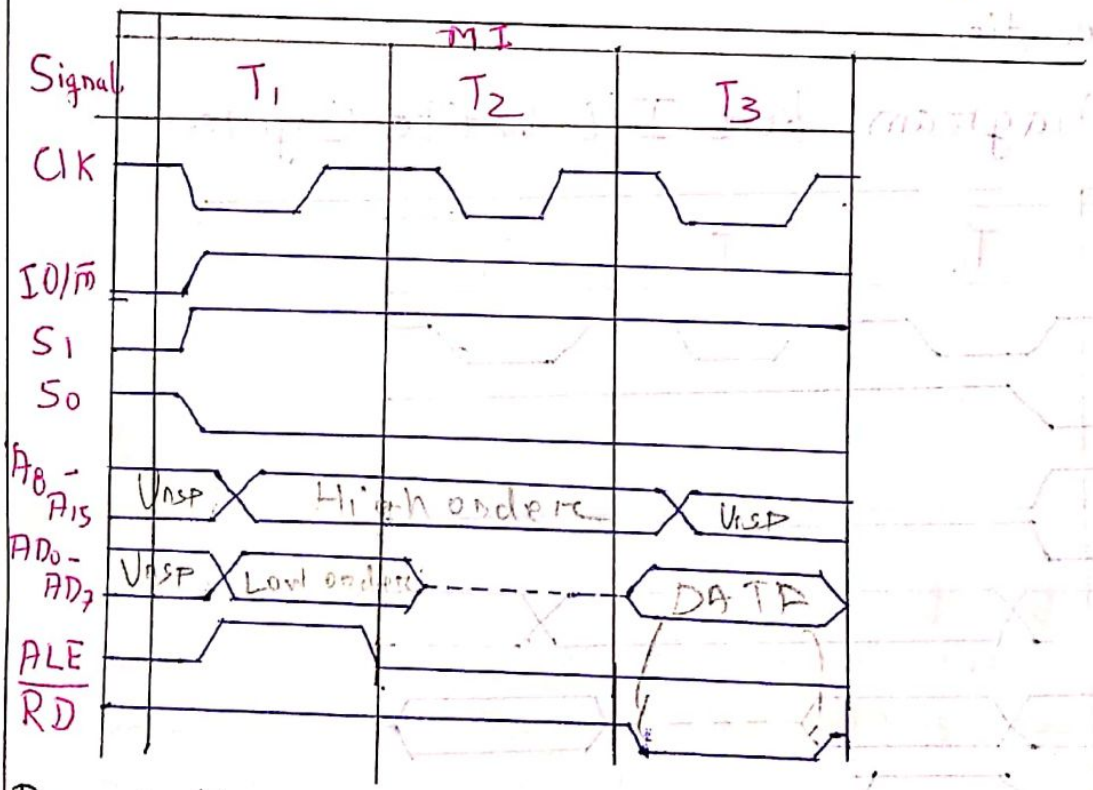
Description:

⇒ The process takes three T-states to execute the cycle for memory write operation. And seven signals are used to write the memory operation and that is CLK, IO/$\overline{M}$, S_0, S_1, A_8-A_{15}, A_0-A_7, ALE, $\overline{WR}$

⇒ The memory write machine cycle is executed by the process on to write a data byte from memory.

⇒ The instruction which have more than 1 byte word size will use the machine cycle after the opcode the machine cycle.

Timing Diagram for I/O Read Cycle



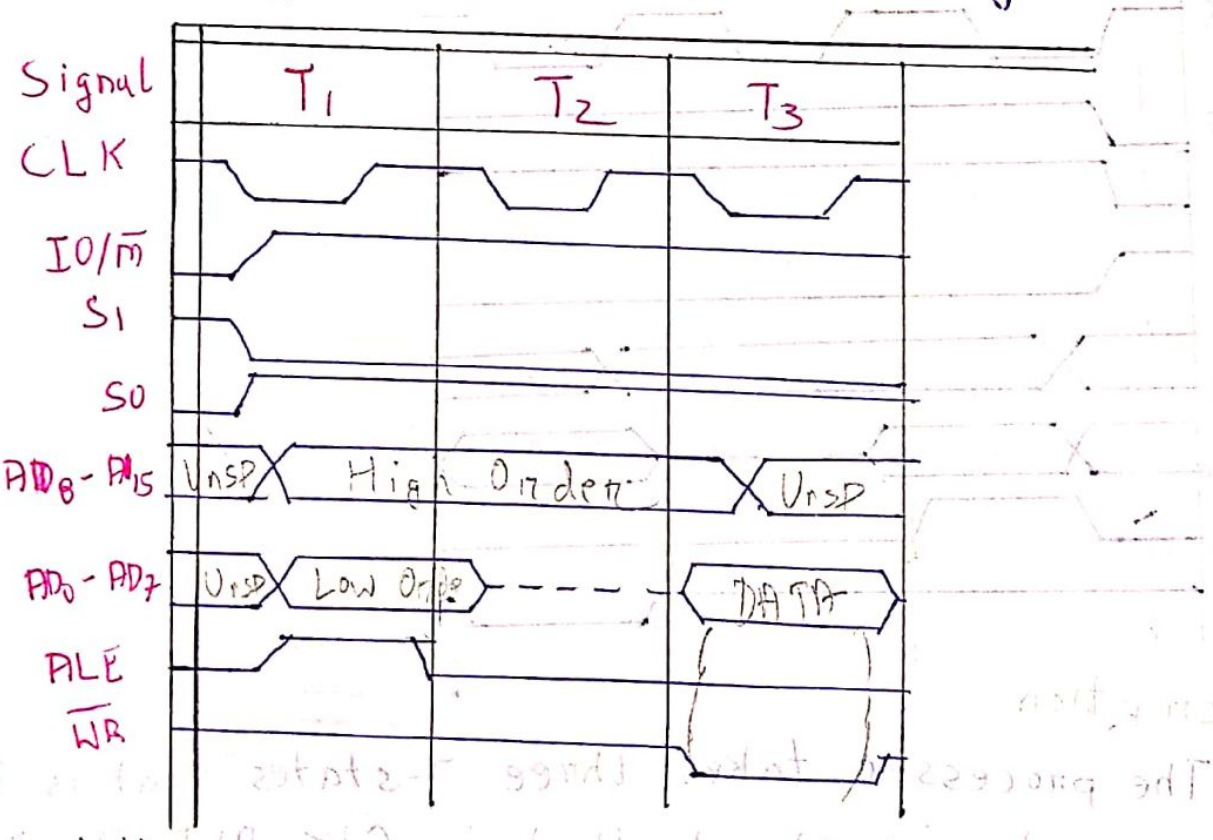
Description

⇒ The processor takes three T-states that is T_1, T_2, T_3 and six signal that is CLK, ALE, A_5-A_8, A_7-A_0, $\overline{RD}$, IO/$\overline{M}$, S_1, S_0 to execute the cycle for I/O read operation.

⇒ I/O read cycle is executed by the process of to read a data byte I/O port or from the peripheral or which is I/O, mapped in the system,

⇒ The input instruction uses the machine cycle during the execution

Timing Diagram for I/O Write Cycle



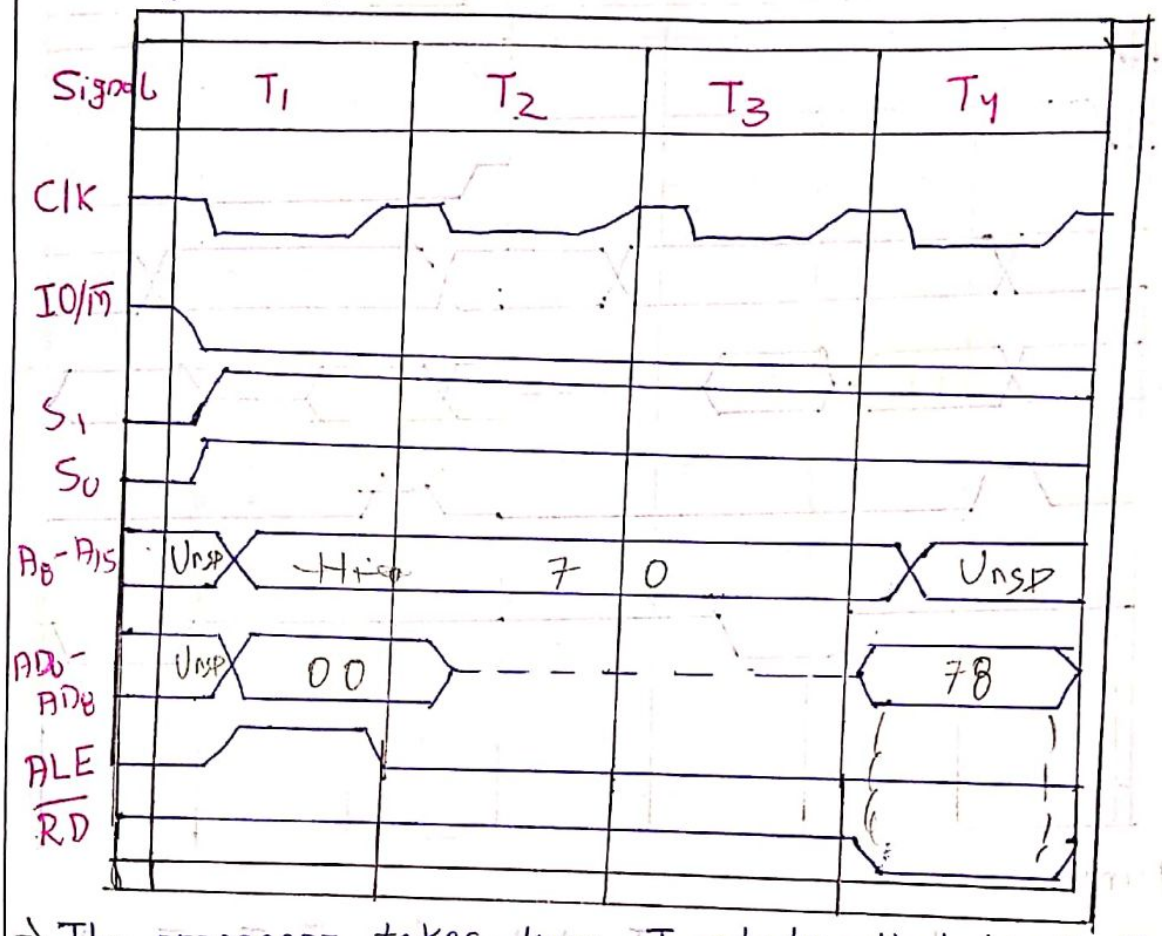
Description

⇒ The processor takes three T-states i.e. T_1, T_2, T_3 and seven signals that is CLK, ALE, $A_{15}-A_8$, AD_7-AD_0, $\overline{WR}$, S_1 & S_0.

⇒ I/O Write cycle is executed by the process to write a data byte I/O port or from the peripherial or which is I/O mapped in the system,

⇒ The input instruction uses the machine cycle during the execution.

Timing Diagram for MOV A,B



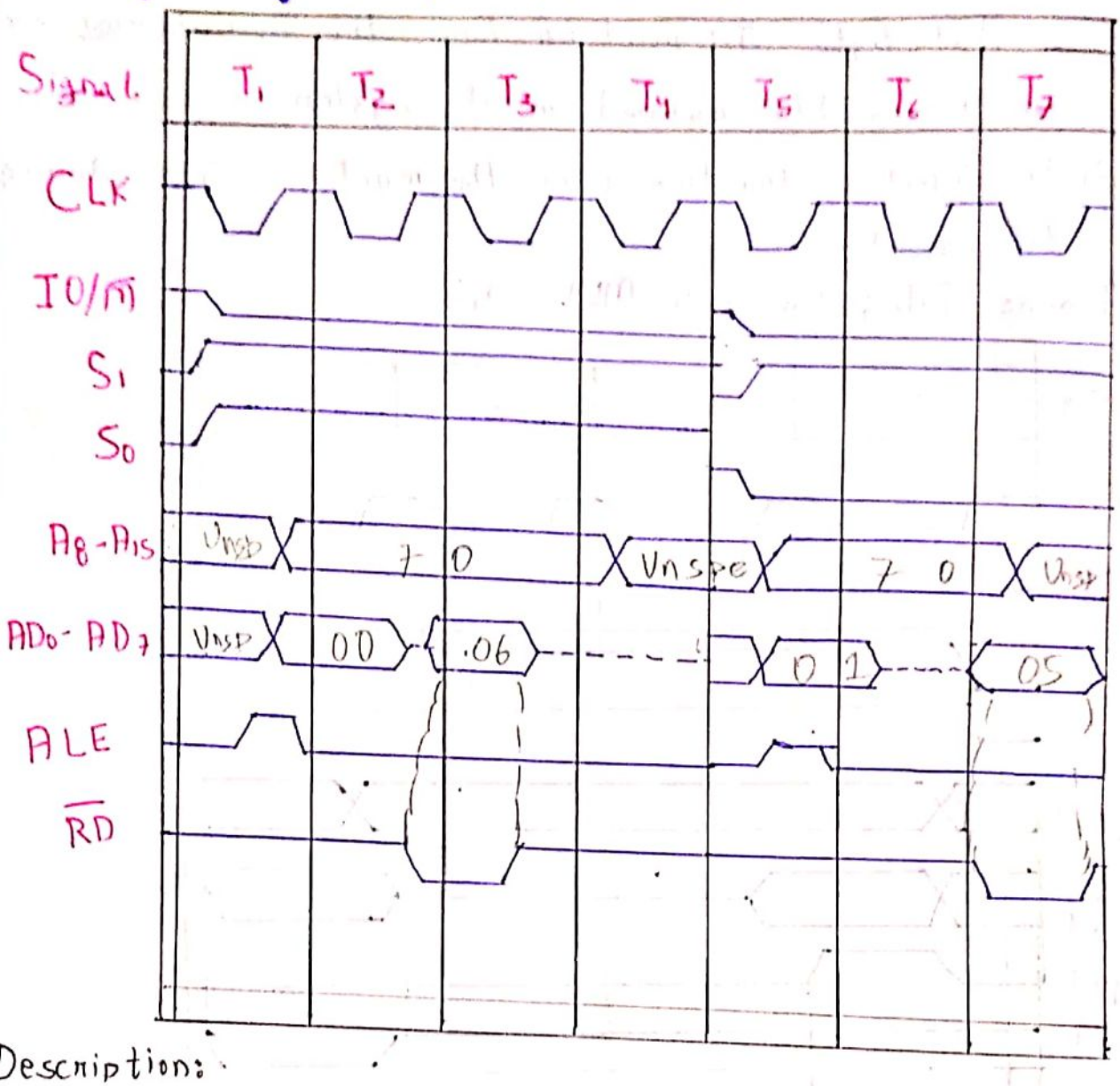
MOV A,B
7000

⇒ The processor takes four T-state that is T_1, T_2, T_3, T_4 and eight signals are used to execute the operation such that MOV A,B that means data will be move from B-C pair register to accumulator.

⇒ The instruction MOVE A,B is a 1 byte instruction microprocessor takes one machine cycle (opcode fetch) to complete instruction,

⇒ Hence, the code for MOV A,B is passed the microprocessor

Timing Diagram for MVI B, 05H

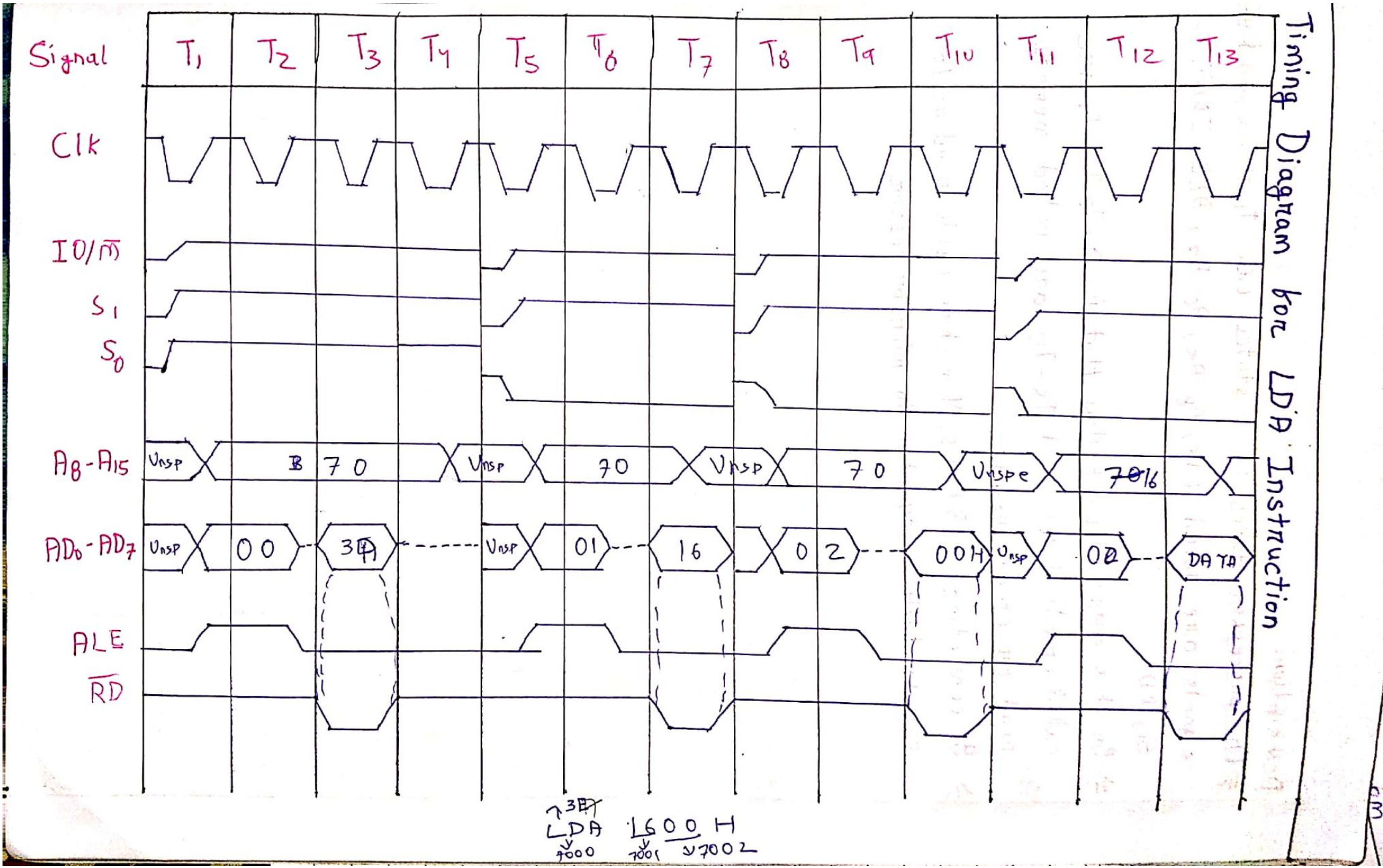


Description:

⇒ The processor takes seven T-states i.e. $T_1, T_2, T_3, T_4, T_5, T_6, T_7$ and eight signals are used i.e. CLK, A_8-A_{15}, AD_0-AD_7, ALE, IO/$\overline{M}$, S_1, S_0 to execute the MVI B, 05H operation.

⇒ The instruction MVI B, 05H is a 2 byte instruction and microprocessor takes two machine cycle (opcode fetch) to complete instruction.

⇒ Hence, the code for MVI, B, 05H is passed to the microprocessor



Description:

⇒ The processor takes 13 T-states i.e. T_1 to T_{13} and eight signals are used CLK, A_8-A_{15}, A_0-A_7, ALE, IO/$\overline{M}$, S_0, S_1, $\overline{RD}$

⇒ In this diagram, we follow that is T_1, T_2 T_3, T_4 are called opcode fetch and T_5-T_{13} are called memory read or opcode read.

⇒ It consisting of 3 bytes, 4 machine cycle used as to process LDA instruction in microprocessor

MICROPROCESSOR BASED SYSTEM DEVELOPMENT AIDS

Interfacing:

⇒ It is a microprocessor is to connect it with various peripherials to perform various operation to obtain a desired output.

⇒ Interfacing are of two types: ① Memory Interfacing
② I/O Interfacing

① Memory Interfacing

⇒ Memory interfacing is use, when the microprocessor needs to access memory frequently for reading and writing data stored in the memory.

⇒ It is also used when writing or reading to a specific register of a memory chip.

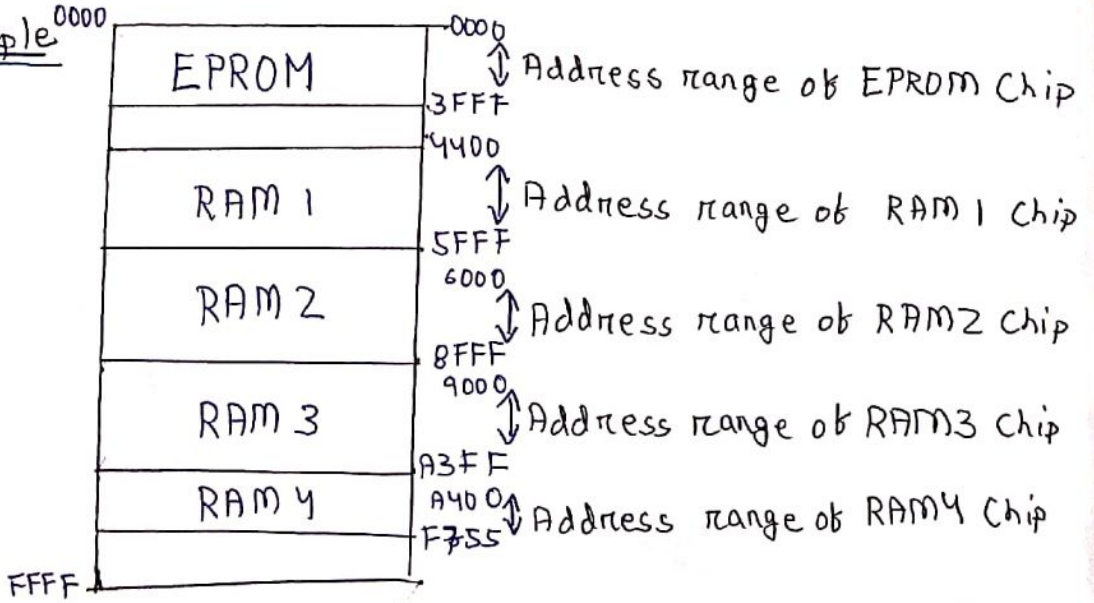
② I/O Interfacing

⇒ I/O interfacing is achieved by connecting keyboard (input) and display monitor (output) with the microprocessor.

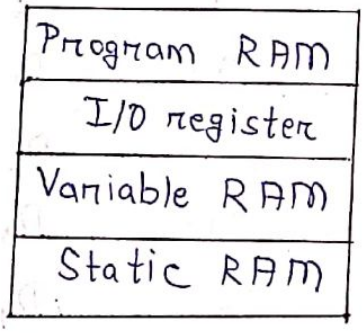
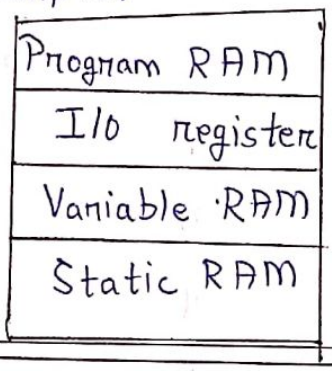
Mapping

⇒ A memory map is a pictorial representation of the address range and shows where the different memory chips are located within the address range.

Example



Difference between Memory Mapped I/O and I/O Mapped I/O

Memory Mapped I/O	I/O Mapped I/O
⇒ Common address space required	⇒ Different address space required.
⇒ A single set of read or write control lines	⇒ Separate read or write control signals.
⇒ More decoding is required	⇒ Less decoding is required.
⇒ Memory control signal is used to control read and write i/o operation	⇒ I/O control signal is also use to control read and write operation.
⇒ Memory and i/o share the entire address range of the processor.	⇒ Processor provides separate address range for memory and I/O
⇒ Example: 	Example: 

8255 PPI (Programmable Peripherial Interface)

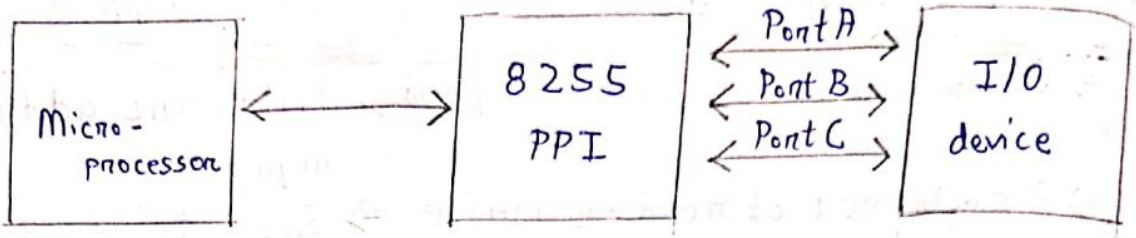
⇒ 8255 is a programmable peripherial interface.

⇒ It was developed and manufactured by intel corporation in 1970.

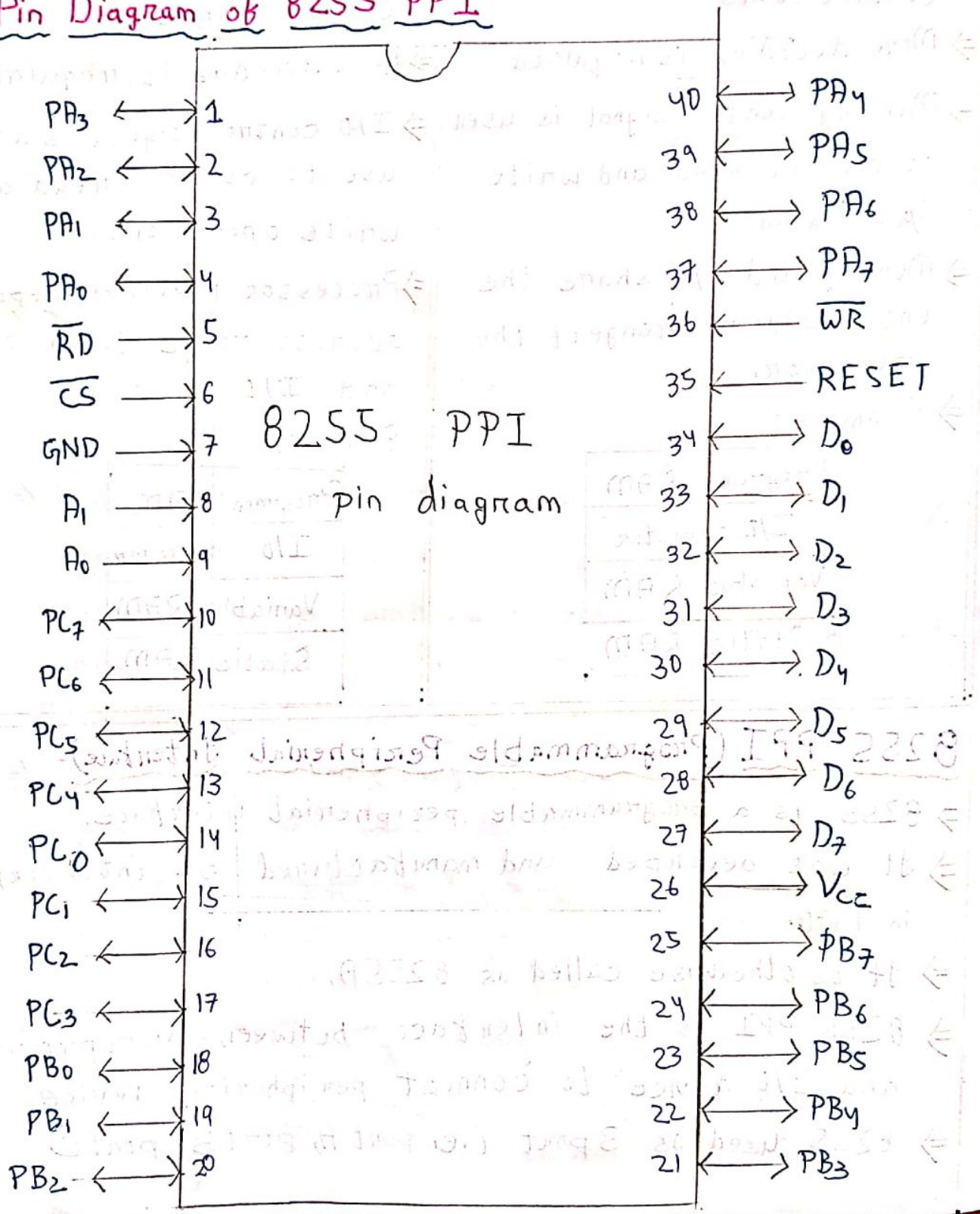
⇒ It is otherwise called as 8255A.

⇒ 8255 PPI is the interface between microprocessor and I/O device to connect peripherial device

⇒ 8255 used as 3 port i.e. port A, port B, port C.



Pin Diagram of 8255 PPI



⇒ 8255 is a 40 IC pin package and each pin divided into four categories: (i) Port
(ii) Data bus
(iii) Power Supply
(iv) Control Signal

Port

⇒ It is generally a specific place for being physically connected to the other devices using a socket and plug in computer generally two ports are used:

i) Serial Port
ii) Parallel Port

Serial Port

⇒ Transmit 16 bit data at a time in sequence order.

Ex: Scanner

Parallel Port

⇒ Transmit more than 16 bit data at a time parallely.

Ex: Printer

⇒ 8255 has 4 port: (i) Port A
(ii) Port B
(iii) Port C (lower)
(iv) Port C (upper)

→ $PA_0 - PA_7$ = Port A (8 bit)
→ $PB_0 - PB_7$ = Port B (8bit)
→ $PC_0 - PC_3$ = Port C (lower 4bit)
→ $PC_4 - PC_7$ = Port C (upper 4bit)

Data bus:

⇒ It is a 8 bit that is D_0-D_7 and it is also called bidirectional and pin occuring 27 to 34 in the 8255 diagram.

Power Supply:

⇒ 8255 PPI uses +5V power supply as V_{cc} and 26 number of pin use for the V_{cc} and 7 number pin as ground (GND) pin.

Control Signal:

$\overline{CS}$ (Chip Select)

⇒ A Low on this input signal enable the communication between 8255 and CPU.

$\overline{WR}$ (Write Signal)

⇒ It is active low signal, It performs write operation when the signal goes low the microprocessor write into a selected input or output port and a control word register.

$\overline{RD}$ (Read Signal)

⇒ It is active low signal. It performs read operation when the signal goes low the microprocessor read into a selected input or output port and a control word register.

RESET

⇒ It is active high signal. It clear the control word register and save all port in the input port.

$A_1 A_0$

⇒ These are normally connected to the lower bit of address bus that is A_0-A_1 and occupying the pin 8 & 9.

Operational Mode of 8255:-

⇒ There are two different operating mode for 8255 PPI : (i) Bit set or Reset Mode
(ii) I/O Mode

BSR Mode

→ It is mainly used to define handshek signal

I/O Mode

→ It is used for input or output data transfer.

Control Word Register

⇒ 8255 may be operated in one of the 2 mode (that is BSR or I/O mode) by initialising D_7 bit.

⇒ If D_7 bit is equal to 1 then it is operating in I/O mode.

⇒ If $D_7 = 0$, it is operating in BSR mode.

Input Or Output Port:

→ This mode is divided in to 3 types

i) Mode 0
ii) Mode 1
iii) Mode 2

Mode 0

→ It is a simple I/O mode,

⇒ In this mode port A, port B, port C are used as simple 8 bit I/O port.

⇒ This port couldnot require handshaking signal.

⇒ There is no need of BSR mode, that's why we are using I/O mode in this mode.

Mode 1

⇒ It is also known as storbed i/o mode.

⇒ When Group A and Group B are programmed to mode 1 then two port operated in storbed in mode, to this first of all control word register to BSR mode to define the store signal as handshaking.

⇒ After that the control word register is program to i/o mode.

Mode 2

⇒ It is also known as bi-direction handshaking i/o.

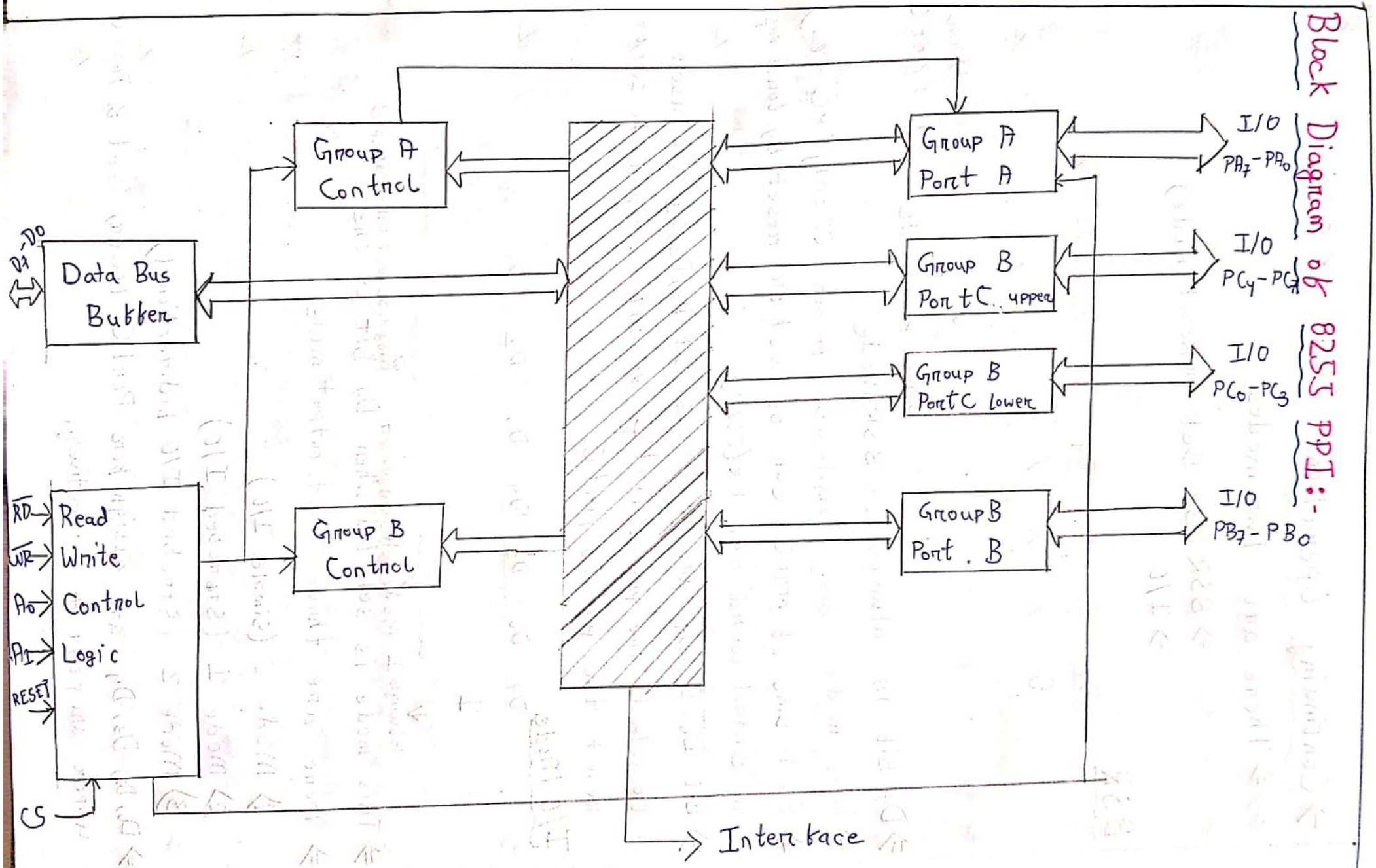
⇒ In this mode port A is used as bi-direction mode and port B is either on mode 0 on mode 1.

Continuing of A_0 A_1

⇒ The selection of i/o port and control word register is done by using A_0A_1.

A_0	A_1	Selection
0	0	Port A
0	1	Port B
1	0	Port C
1	1	Control Word

Block Diagram of 8255 PPI:-

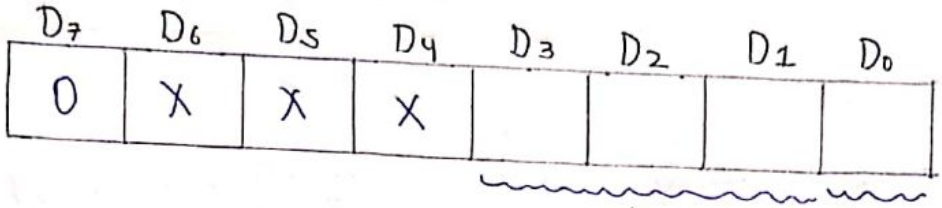


→ Continuing Operational Mode of 8255

⇒ There are two modes:

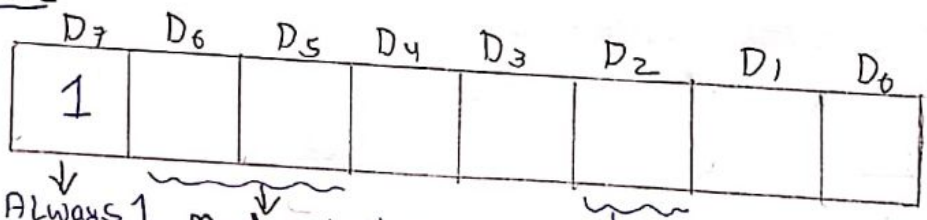
→ BSR (Bit Set or Reset Mode)
→ I/O

BSR



⇒ D_7 bit is always for BSR Mode
⇒ BSR mode is always applicable to port C only (PC_0-PC_7)
⇒ Each Line of port C can be set or reset by loading in control word register (CWR).
⇒ Bit D_6, D_5, D_4 are unspecified. Bit D_3, D_2, D_1 are used to select the pin of port C. D_0 bit is used to set or reset the pin of port C.

I/O Mode



⇒ This mode is selected when D_7 bit is one
⇒ There are three input, output mode
1) Mode 0 (Simple I/O)
2) Mode 1 (Strobed I/O)
3) Mode 2 (Strobed I/O bidirectional)
⇒ D_0, D_1, D_3, D_4 are assign for Port C lower, Port B, Port C upper and port A respectively.

⇒ If $D_0 = D_4 = 0$, the port C lower and port A is act as output port. If $D_0 = D_4 = 1$, the port C upper & port A act as input port.

⇒ If $D_1 = D_3 = 0$, then port B & port C upper act as input port. If $D_1 = D_3 = 1$, then port B & port C upper act as output port.

⇒ D_2 is used for the mode selection of group B (port B and port C lower)

⇒ When $D_2 = 0$, mode 0 is selected and when $D_2 = 1$, mode = 1 is selected.

⇒ D_5 and D_6 are used for the mode selection of group A (Port A, Port C, upper)

⇒ Mode selected in bit D_2, D_5, D_6 are all 0 for mode 0 operation

Control Word

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Bit no.
Control Word Register				Port A	Port C upper	Mode for Port B	Port B	Port C lower	

Bit no (D_0) <If is for Port C lower>

⇒ To make port C lower an i/p port, the bit set to 1.

⇒ To make port C lower an o/p port, the bit set to 0.

Bit no (D_1) <If is for Port B>

⇒ To make port B an i/p port the bit is set to 1.

⇒ To make port B an o/p port the bit is set to 0.

Bit no. (D_2) <If is for selection of the mode for the port B>

⇒ It is for the selection of the mode for the port B.

If the port B operated as mode 0 the bit is set to 0.

for mode 1 operation of the port B, it is set to 1.

Bit no. (D_3) (It is for the port C upper)

⇒ To make port C upper are i/p port the bit is set to 1.

⇒ To make port C upper are o/p port the bit is set to 0.

Bit no. (D_4) (It is for Port A)

⇒ To make port A an i/p port, if it is the bit is set to 1.

⇒ To make port A an o/p port, if it is the bit is set to 0.

Bit no. D_5, D_6

⇒ This bit is to define the operating mode of Port A.

MODE OF PORT A	Bit No. 6	Bit No. 5
MODE 0	0	0
MODE 1	0	1
MODE 2	1	0 or 1

Control Word

⇒ Control Word is used for the programming port of 8255. The control word is written into the control word register.

⇒ No read operation of the control word register is allowed. If a particular port is to be made an i/p port the corresponding bit for the point is said to 1.

⇒ If a particular port is made an output port the corresponding bit for the port is set to 0.

Control Group

⇒ The 24 line of i/o port are divided into two groups i.e. group A, group B,

⇒ The group A contains port A & Port C upper

⇒ The group B contains port B & Port C lower.

Interfacing Seven Segment Displays

⇒ Interface the 8085 microprocessor system with seven segment display through the programmed I/O port 8255.

⇒ Seven segment displays is often used in the digital electronic equipment to display information regarding certain process.

⇒ I/O devices such as LEDs and keyboards are essential components of the microprocessor based or microcontroller based system.

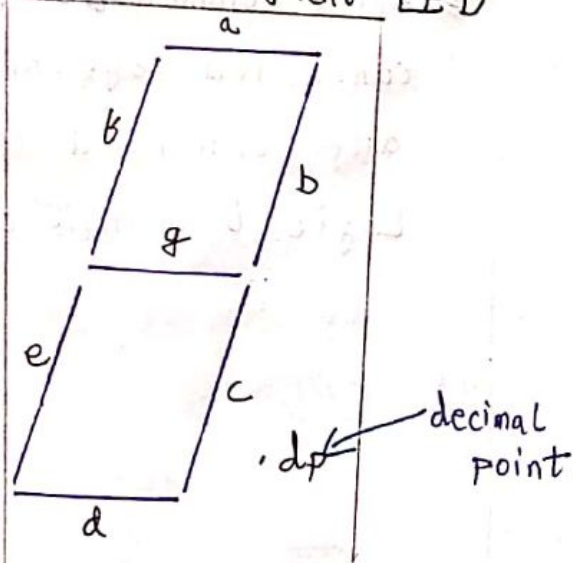
⇒ Seven-segment LEDs often used to display BCD numbers (1 through 9) and a few alphabets,

⇒ A group of eight LEDs physically mounted in the shape of the number eight plus a decimal point,

⇒ Each LED is called a segment and labeled as 'a' through 'g'.

⇒ Commonly used output peripherials in embedded systems are LEDs, seven segment LEDs and LCDs, the simplest is LED

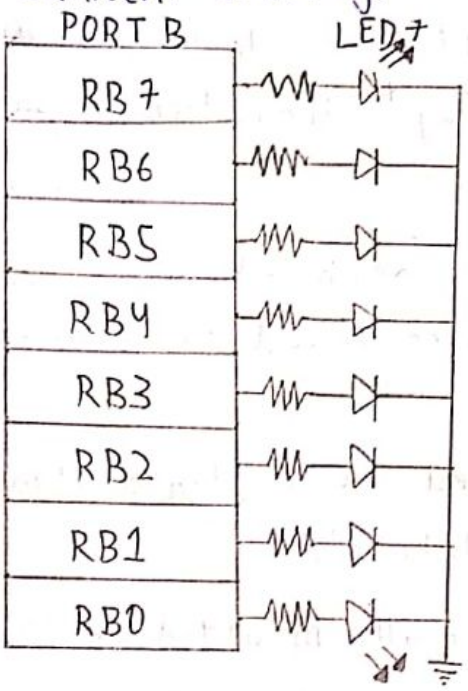
Fig. Seven Segment LED



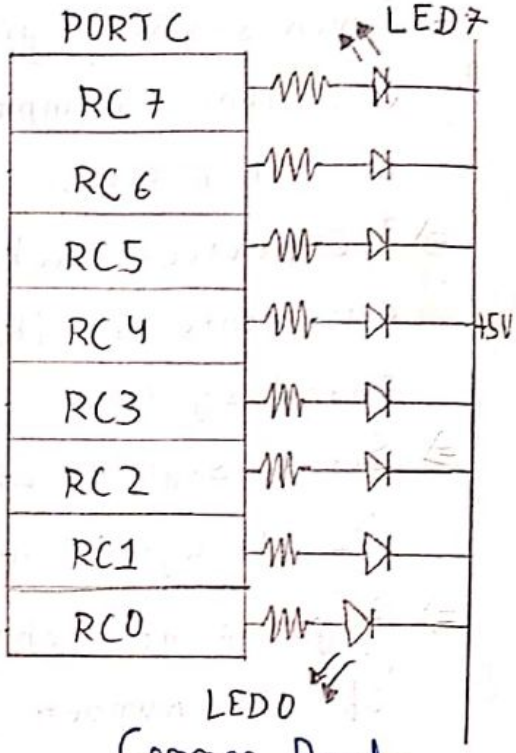
Two ways of connecting LEDs to I/O ports

i) LED cathodes are grounded and logic 1 from the I/O port turns on the LEDs. The current is supplied by the I/O port called current sourcing.

ii) LED anodes are connected to the power supply and logic 0 from the I/O port turns on the LEDs – The current is received by the chip called current sinking.



Common Cathode
Active high



Common Anode
Active Low

⇒ In a common anode seven segment LED. All anodes are connected together to a power supply and cathodes are connected to data lines,
Logic 0 turns on a segment

Memory Interfacing in 8085

⇒ Memory is an integral part of a microprocessor system,

⇒ Memory Interfacing in 8085 is used to access memory quite frequently to read instruction codes and data stored in the memory.

⇒ This read/write operations are monitored by control signals.

⇒ The microprocessor activates these signals when it wants to read from and write into memory.

Memory Structure and its Requirements

⇒ Read/Write memories consist of an array of registers, in which each register has unique address.

⇒ The size of the memory is N x M as shown in below figure where N is the number of registers and M is the word length, in number of bits.

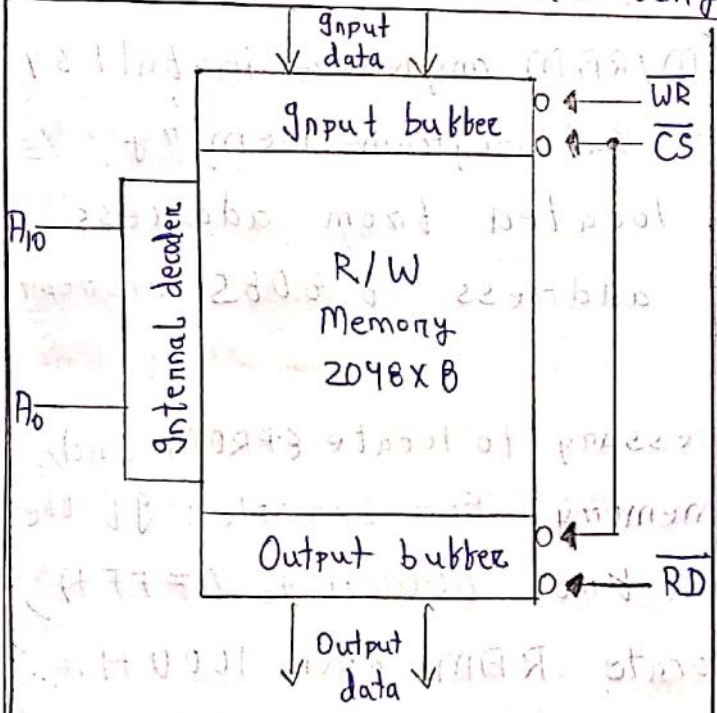


Fig (a) Logic diagram for RAM

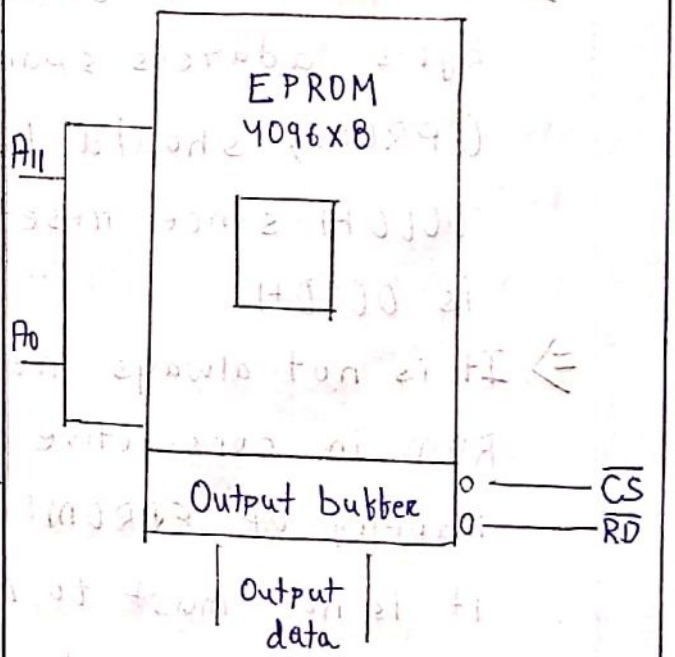


Fig (b) Logic Diagram for EPROM

Basic Concepts in Memory Interfacing

⇒ Microprocessor 8085 can access 64 Kbytes memory since address bus is 16-bit. But it is not always necessary to use full 64 Kbytes address space. The total memory size depends upon the application.

⇒ Generally EPROM (or EPROMs) is used as a program memory and RAM (or RAMs) as a data memory. When both, EPROM and ROM are used, the total address space 64 Kbytes is shared by them.

⇒ The capacity of program memory and data memory depends on the application.

⇒ It is not always necessary to select 1 EPROM and 1 RAM. We can have multiple EPROMs and multiple RAMs as per the requirement of application.

⇒ We can place EPROM/RAM anywhere in full 64 bytes address space. But program memory (EPROM) should be located from address 0000H since reset address of 8085 microprocessor is 0000H.

⇒ It is not always necessary to locate EPROM and RAM in consecutive memory. For example: If the mapping of EPROM is from 0000H to 0FFFH, it is not must to locate RAM from 1000H. We can locate it anywhere between 1000H and FFFH. Where to locate memory component totally depends on the application.

The memory interfacing requires to:

→ Select the chip
→ Identify the register
→ Enable the appropriate buffer

Microprocessor system includes memory devices and I/O devices, it is important to note that microprocessor can communicate with only one device at a time, since the data, address and control buses are common for all the devices. In order to communicate with memory or I/O devices, it is necessary to decode the address from the microprocessor. Due to this each device can be accessed independently.

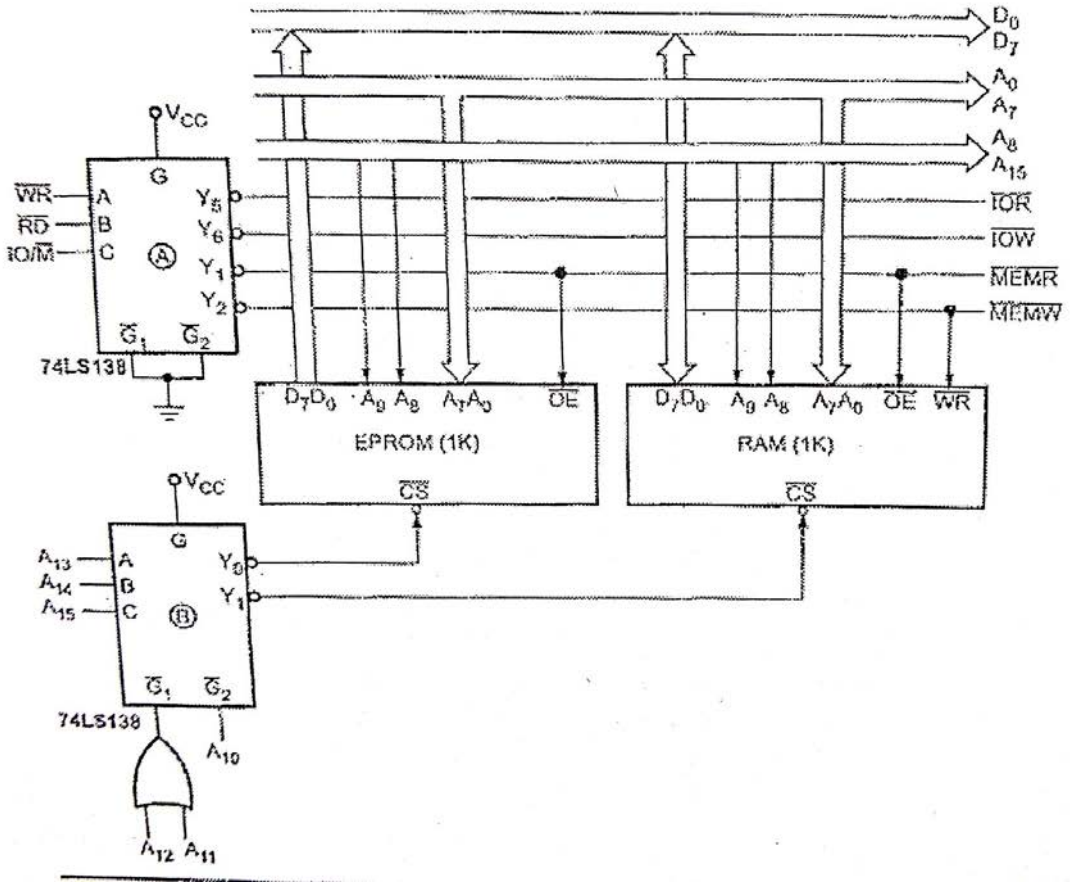
Address Decoding Techniques

⇒ There are two techiques of address decoding:

→ Absolute Decoding / Full Decoding
→ Linear Decoding / Partial Decoding

Absolute Decoding / Full Decoding

In absolute decoding technique, all the higher address lines are decoded to select the memory chip and the memory chip is selected only for the specified logic levels on these high order address lines; no other logic levels can select the chip. Below Figure shows the memory interfacing in 8085 with absolute decoding. The address technique is normally used in large memory systems.

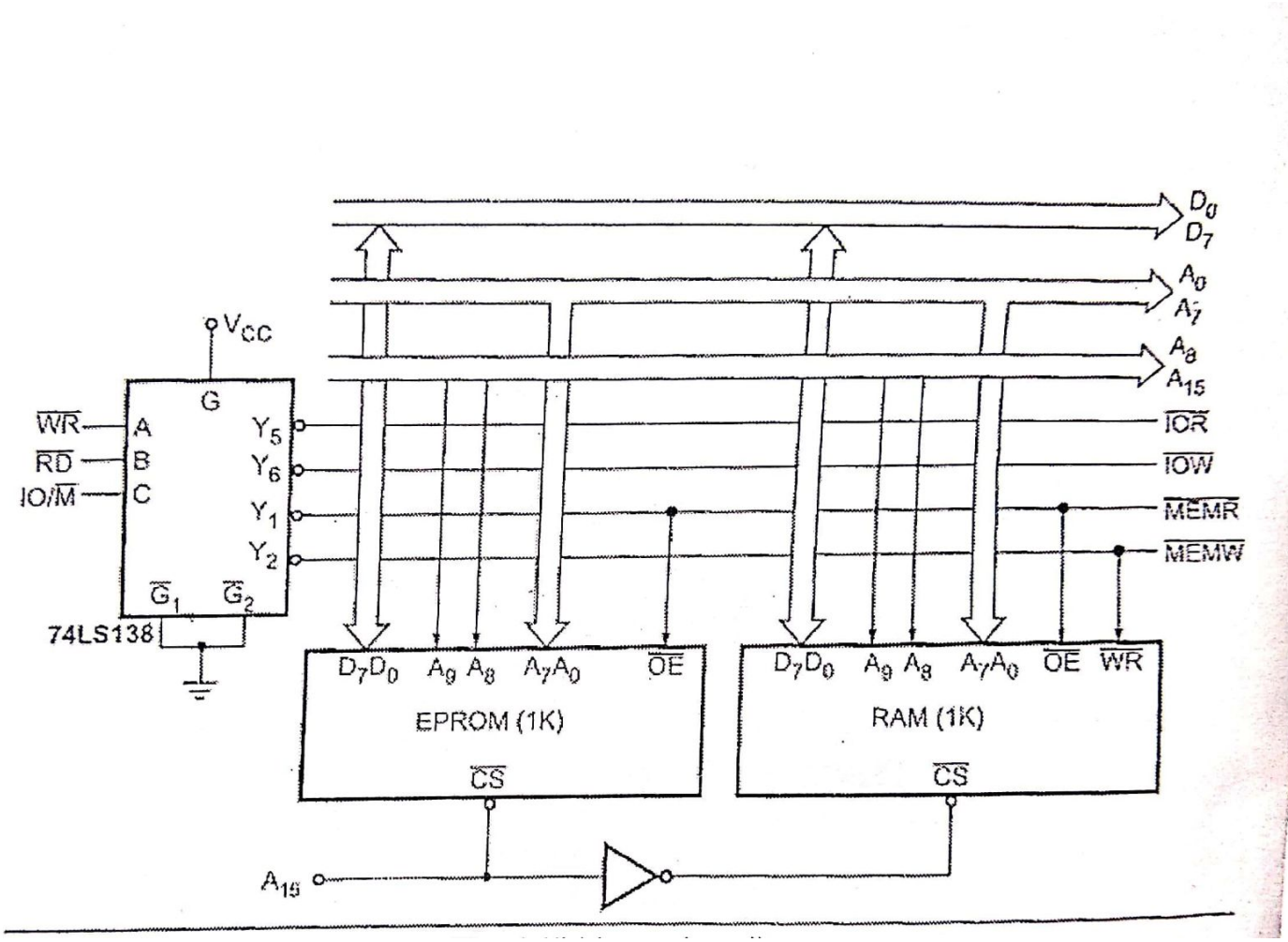


Linear decoding

In small systems, hardware for the decoding logic can be eliminated by using individual high-order address lines to select memory chips. This is referred to as linear decoding. Below Figure shows the addressing of RAM with linear decoding technique. This technique is also called partial decoding. It reduces the cost of decoding circuit, but it has a drawback of multiple addresses (shadow addresses).

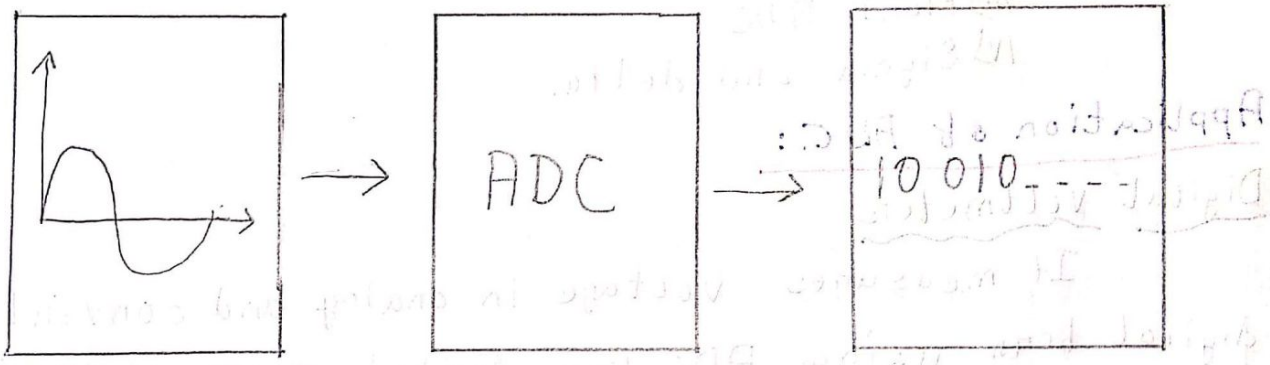
Below Figure shows the addressing of RAM with linear decoding technique. A_{15} address line, is directly connected to the chip select signal of EPROM and after inversion it is connected to the chip select signal of the RAM. Therefore, when the status of A_{15} lines

is 'zero', EPROM gets selected and when the status of A_{15} line is 'one' RAM gets selected, The status of the other address lines is not considered, since those address lines are not used for generation of chip select signals,



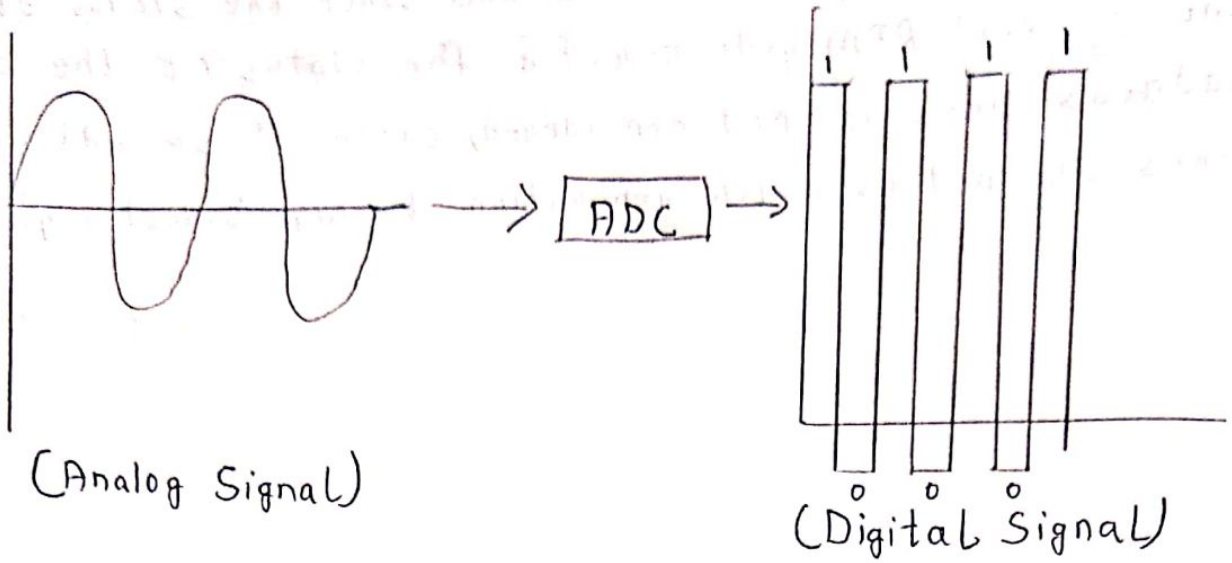
ADC & DAC with Interfacing

ADC (Analog to Digital Converter)



Analog Signal

Digital Signal



(Analog Signal) (Digital Signal)

⇒ It is an electronic integrated circuit which transforms analog signal (continuous) to digital signal (discrete).

⇒ Analog signals are directly measurable quantities while digital signals only have two state that is 0 and 1.

Purpose of ADC

⇒ Processor perform arithmetic operations on digital signals.

⇒ Signals in digital form are less suspectable to effect of noise.

⇒ ADC provides link between analog signal & digital signal.

⇒ Following are the types of analog and digital converter (ADC):-
- I) Counter type
- II) Successive approximation
- III) Flash ADC
- IV) Sigma and delta.

Application of ADC:

Digital voltmeter

It measures voltage in analog and convert to digital form using ADC for digital representation form

Mobile Phone:

Analog voice is converted to digital form for

further processing (speech operation encoding etc.).
before it is converted back to analog form for transmission.

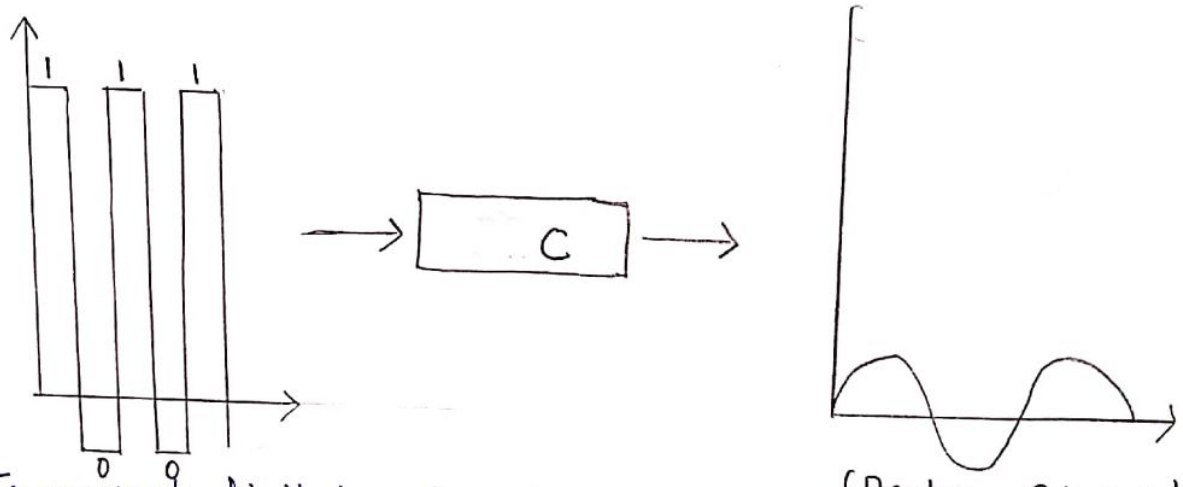
Scanner:

When we take photo or document the scanner uses ADC internally to convert analog information provided by picture or document into digital information.

Voice recorder:

It uses ADC to convert analog voice information to digital information.

DAC (Digital to analog converter)



⇒ To convert digital values to analog voltage DAC is used.
⇒ It performs inverse operation of ADC.
⇒ There are two types of DAC (i) Weighted Resistor
(ii) Resistive Divider
⇒ Following specifications are considered from the design, development as well as selection of DAC. (digital to analog converter)

- → Resolution
- → Reference Voltages
- → Setting time
- → Linearity
- → Speed
- → Errors

Application of DAC:

⇒ Modem: Modem needs DAC to convert data to analog form. So that it can be carried over telephone wires

⇒ Pointers

⇒ Digital audio

⇒ Digital motor control

⇒ Sound equipments.

Data Transfer Schemes

In a microprocessor-based system or in a computer data transfer takes place between two devices such as microprocessor and memory, microprocessor and I/O devices, and memory and I/O devices. Usually, memories are compatible with microprocessors because the same technology is employed in the manufacturing of both memories and microprocessors. Hence, there is less problem associated with the interfacing of memory.

A microprocessor-based system or a computer may have several I/O devices of different speed. A slow I/O device can not transfer data when microprocessor issues instruction for the same because it takes some time to get ready. To solve the problem of speed mismatch between a microprocessor and I/O devices a number of data transfer techniques have been developed. The data transfer schemes are classified into the following two broad categories:

→ Programmed data transfer schemes

→ DMA data transfer schemes.

Programmed Data Transfer Scheme

Programmed Data transfer schemes are controlled by the CPU. Data are transferred from an I/O device to the CPU or vice versa under the control of programs which reside in the memory.

These programs are executed by the CPU when an I/O device is ready to transfer data. The microprocessor executes the program to transfer data. Programmed data transfer schemes are employed when small amount of data are to be transferred. The programmed data transfer schemes are classified into the following three categories,

i) Synchronous data transfer scheme
ii) Asynchronous data transfer scheme
iii) Interrupt driven data transfer scheme.

Synchronous Data Transfer Scheme

⇒ Synchronous means "at the same time"
⇒ The device which receives data are synchronized with same clock.
⇒ When the CPU I/O devices match in speed, the synchronous data transfer is employed.
⇒ The data transfer with I/O devices a performed executing IN ~~and~~ or OUT instruction for I/O mapped I/O devices or memory read/write instruction for memory.
⇒ This technique of data transfer is rarely used because I/O device compatible with microprocessor in speed are usually not available

Asynchronous Data Transfer

⇒ Asynchronous means "at regular intervals"

⇒ In this method, data transfer is not based on predetermined timing pattern.

⇒ This technique of data transfer is used when the speed of an I/O device does not match the speed of the microprocessor.

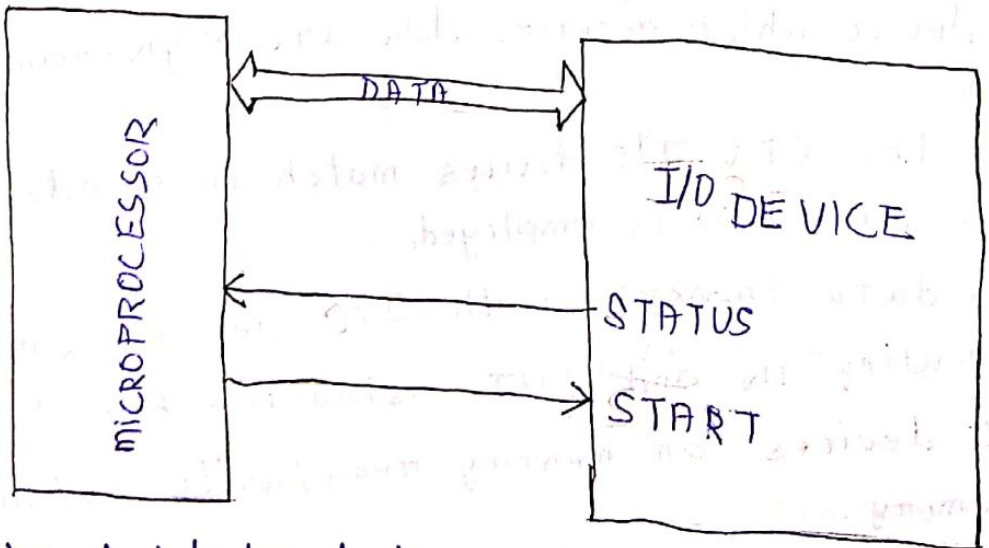
⇒ In this technique, the status of the I/O device i.e. wheather the device is ready or not checked by the microprocessor before the data transfer.

Purpose of Asynchronous Data Transfer

⇒ Microprocessor check the status ~~of~~ signal before the data transfer.

⇒ Then microprocessor indicates the I/O device to get ready.

⇒ When I/O device get ready the microprocessor sends instructions to transfer the data



⇒ A keyboard interfaced to a microprocessor is an example of this type of data transfer scheme.

Asynchronous Data Transfer Scheme of an A/D converter

⇒ Asynchronous data transfer is used for slow I/O device.

⇒ An A/D converter has been transferred to the microprocessor to transfer data in asynchronous mode.

Process:

⇒ The microprocessor sends a start of conversion signal S/C to the A/D converter.

⇒ The A/D is so slow therefore it take more time to convert analog signal to digital signal.

⇒ When conversion is over A/D converter make end of conversion signal. It means E/C signal is high.

⇒ When E/C signal becomes high the microprocessor issues instruction for data transfer.

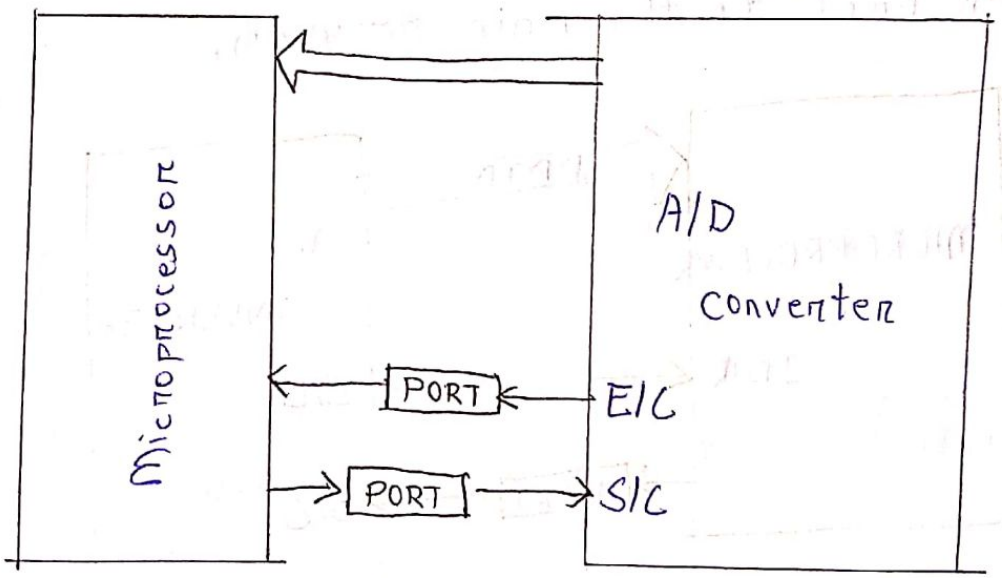


Fig. Asynchronous Data transfer for an A/D converter

Interrupt Driven Data Transfer

⇒ In this scheme the microprocessor initiates an I/O device to get ready, and then it executes its main program instead of remaining in a program

Loop to check the status of the I/O device.

⇒ It just interrupt the normal processing sequence of the microprocessor.

⇒ On receiving an interrupt the microprocessor complete the current instruction at hand and then attends the I/O device.

⇒ It saves the contents of the program counter on stack then take a subroutine called ISS.

⇒ It executes ISS to transfer data from or to the I/O device.

⇒ Different ISS are to be ~~pur~~provided for different I/O device

⇒ After completing the data transfer the microprocessor return back to the main program.

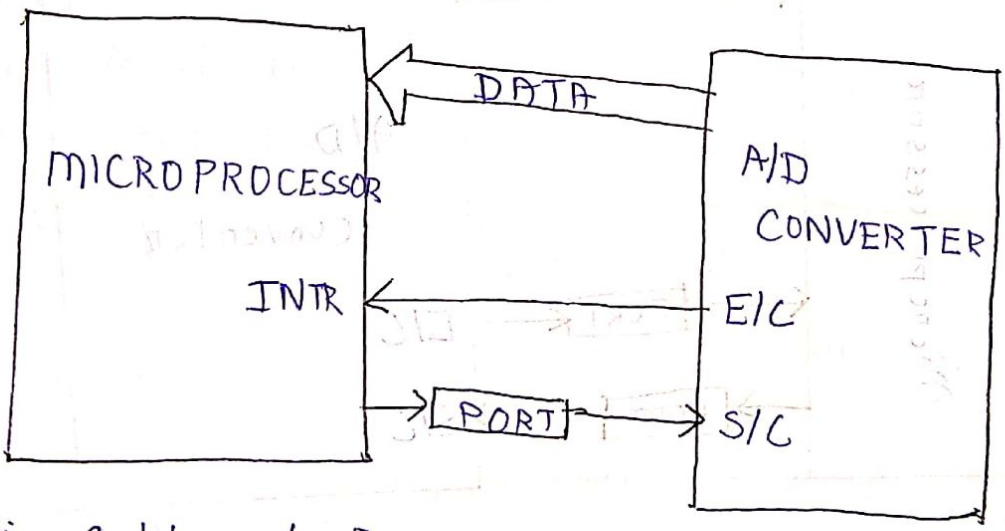
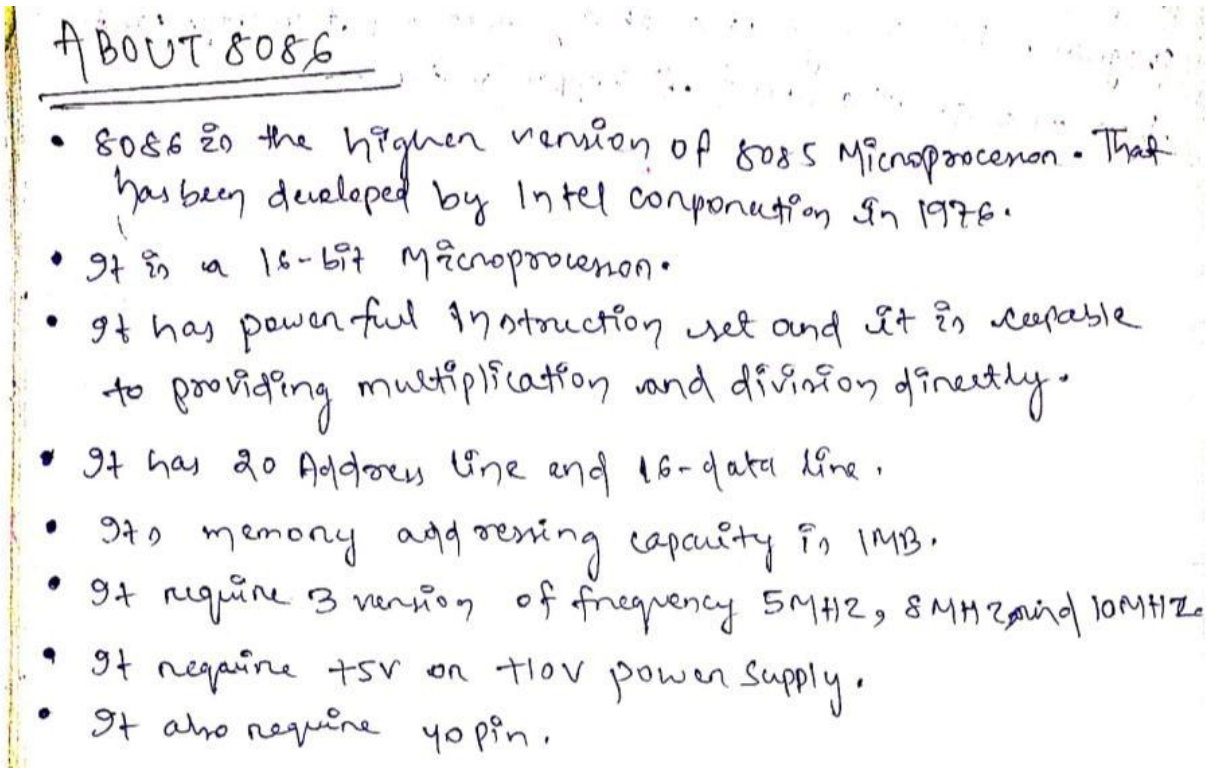


Fig. Interrupt Driven Data Transfer Scheme for an A/D converter.

Chapter 5 : Microprocessor(Architecture and Programming-16 bit-8086).

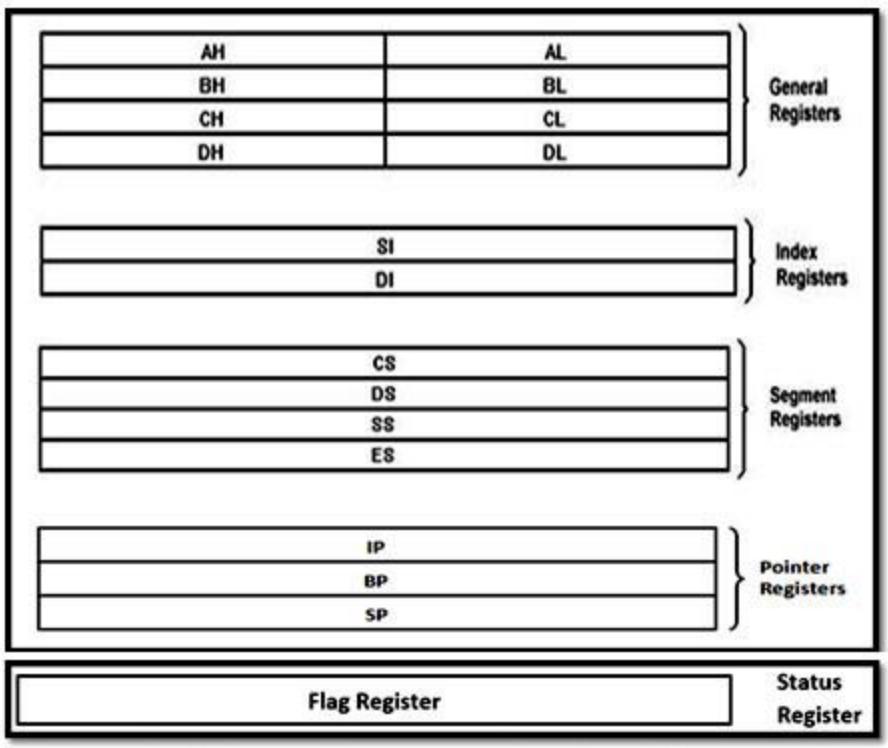


ABOUT 8086

- 8086 is the higher version of 8085 Microprocessor. That has been developed by Intel corporation in 1976.
- It is a 16-bit Microprocessor.
- It has powerful Instruction set and it is capable to providing multiplication and division directly.
- It has 20 Address line and 16-data line.
- Its memory addressing capacity is 1MB.
- It require 3 version of frequency 5MHz, 8MHz and 10MHz.
- It require +5V or +10V power supply.
- It also require 40 pin.

Register organization

A register is a very small amount of fast memory that is built in the CPU (or Processor) in order to speed up the operation. Register is very fast and efficient than the other memories like RAM, ROM, external memory etc,. That's why the registers occupied the top position in memory hierarchy model.



The 8086 microprocessor has a total of fourteen registers that are accessible to the programmer. All these registers are 16-bit in size. The registers of 8086 are categorized into 5 different groups.

a) General registers
b) Index registers
c) Segment registers
d) Pointer registers
e) Status Register

d) General Registers

All general registers of the 8086 microprocessor can be used for arithmetic and logic operations. These all general registers can be used as either 8-bit or 16-bit registers. The general registers are:

i. **AX (Accumulator):**
AX is used as 16-bit accumulator. The lower 8-bits of AX are designated to use as AL and higher 8-bits as AH. AL can be used as an 8-bit accumulator for 8-bit operation.

This Accumulator used in arithmetic, logic and data transfer operations. For manipulation and division operations, one of the numbers must be placed in AX or AL.

ii. **BX (Base Register):**
BX is a 16 bit register, but BL indicates the lower 8-bits of BX and BH indicates the higher 8-bits of BX. The register BX is used as address register to form physical address in case of certain addressing modes (ex: indexed and register indirect).

iii. **CX (Count Register):**
The register CX is used default counter in case of string and loop instructions. Count register can also be used as a counter in string manipulation and shift/rotate instruction.

iv. **DX (Data Register):**
DX register is a general purpose register which may be used as an implicit operand or destination in case of a few instructions. Data register can also be used as a port number in I/O operations.

e) Segment Register:

The 8086 architecture uses the concept of segmented memory. 8086 can able to access a memory capacity of up to 1 megabyte. This 1 megabyte of memory is divided into 16 logical segments. Each segment contains 64 Kbytes of memory. This memory segmentation concept will discuss later in this document.

There are four segment registers to access this 1 megabyte of memory.

The segment registers of 8086 are:

- **CS (Code Segment):**
  Code segment (CS) is a 16-bit register that is used for addressing memory location in the code segment of the memory (64Kb), where the executable program is stored. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions.
- **Stack segment (SS)**
  Stack Segment (SS) is a 16-bit register that used for addressing stack segment of the memory (64kb) where stack data is stored. SS register can be changed directly using POP instruction.
- **Data segment (DS)**
  Data Segment (DS) is a 16-bit register that points the data segment of the memory (64kb) where the program data is stored. DS register can be changed directly using POP and LDS instructions.
- **Extra segment (ES):**
  Extra Segment (ES) is a 16-bit register that also points the data segment of the memory (64kb) where the program data is stored. ES register can be changed directly using POP and LES instructions.

f) Index Registers

The index registers can be used for arithmetic operations but their use is usually concerned with the memory addressing modes of the 8086 microprocessor (indexed, base indexed and relative base indexed addressing modes).

The index registers are particularly useful for string manipulation.

- **SI (Source Index):**
  SI is a 16-bit register. This register is used to store the offset of source data in data segment. In other words the Source Index Register is used to point the memory locations in the data segment.
- **DI (Destination Index):**
  DI is a 16-bit register. This is destination index register performs the same function as SI. There is a class of instructions called string operations that use DI to access the memory locations in Data or Extra Segment.

g) Pointer Registers:

Pointer Registers contains the offset of data(variables, labels) and

instructions from their base segments (default segments).8086 microprocessor contains three pointer registers.

i. **SP (Stack Pointer):**
Stack Pointer register points the program stack that means SP stores the base address of the Stack Segment.

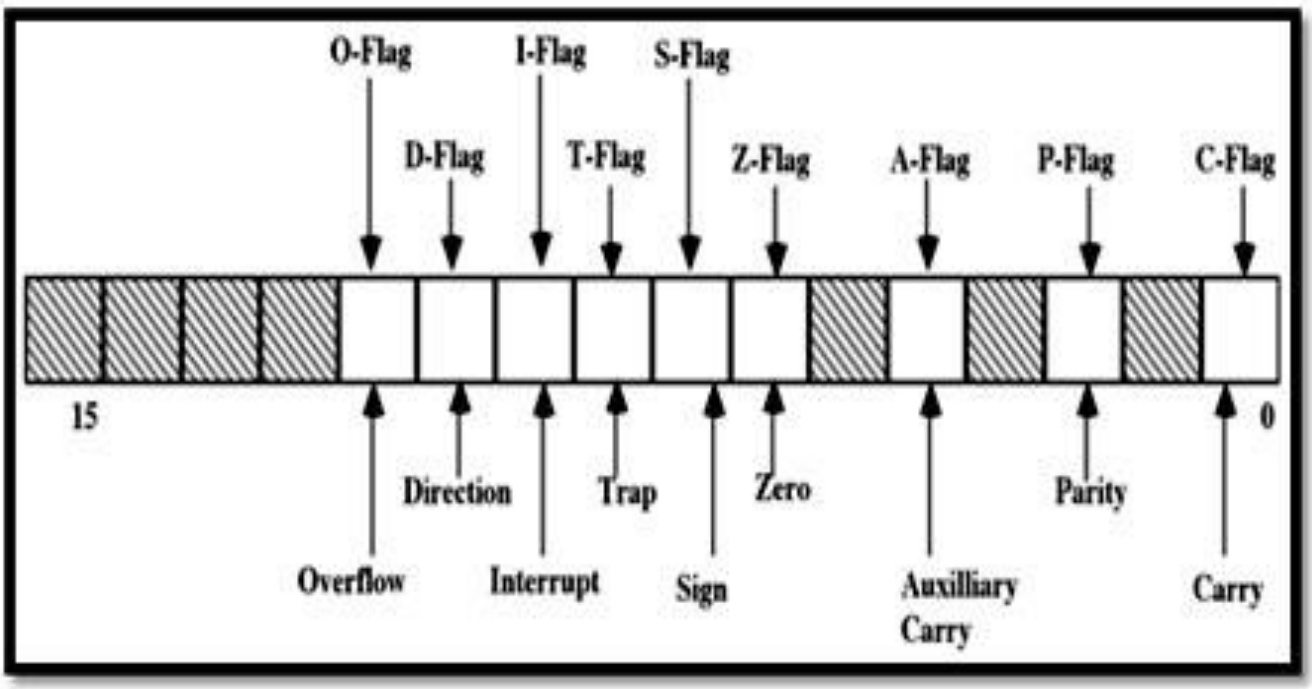
ii. **BP (Base Pointer):**
Base Pointer register also points the same stack segment. Unlike SP, we can use BP to access data in the other segments also.

iii. **IP (Instruction Pointer):**
The Instruction Pointer is a register that holds the address of the next instruction to be fetched from memory.It contains the offset of the next word of instruction code instead of its actual address

h) Status Register:

The status register also called as flag register. The 8086 flag register contents indicate the results of computation in the ALU. It also contains some flag bits to control the CPU operations.

Flag register is 16-bit register with only nine bits that are implemented. Six of these are status flags. The complete bit configuration of 8086 is shown in the figure.



SF (Sign Flag): This flag represents sign of the result.
0-Result is Positive

1-Result is Negative

ZF (Zero Flag): ZF is set if the result produced by an instruction is zero. Otherwise, ZF is reset.

PF (Parity Flag): This flag is set to 1, if the lower byte of the result contains even number of 1's.

0- Odd parity

1- Even parity

CF (Carry Flag)

This flag is set, when there is a carry out of MSB in case of addition or borrow in case of subtraction.

0- No Carry/ Barrow

1- Carry/ Barrow

TF (Trap Flag):

If this flag is set, the processor enters the single step execution mode. When in the single-step mode, it executes an instruction and then jumps to a special service routine that may determine the effect of executing the instruction. This type of operation is very useful for debugging programs.

IF (Interrupt Flag):

If this flag is set, the maskable interrupts are recognized by the CPU, otherwise they are ignored.

DF (Direction Flag):

This is used by string manipulation instructions.

0- The string is processed beginning from the lowest address to the highest address, i.e., auto incrementing mode.

1- The string is processed from the highest address towards the lowest address, i.e., auto incrementing mode.

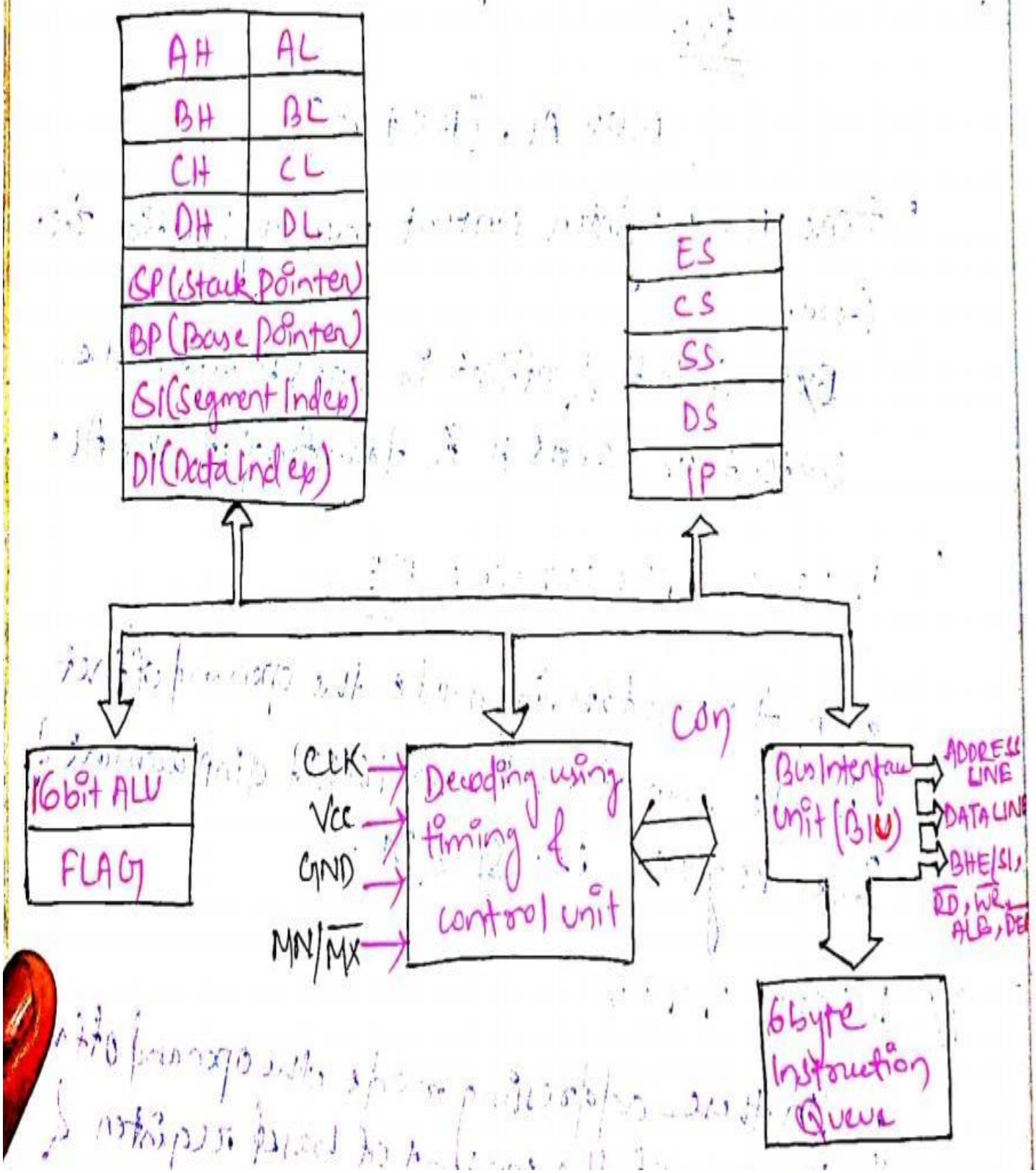
AC (Auxiliary Carry Flag):

This is set when there is a carry from the lowest nibble (i.e, bit three during addition), or borrow for the lowest nibble (i.e, bit three, during subtraction).

OF(Over flow Flag):

This flag is set, if an overflow occurs, i.e, if the result of a signed operation is large enough to accommodate in a destination register.

ARCHITECTURE OF 8086 - MICROPROCESSOR



- 8086 Architecture is divided in two ways :-
  (I) Bus Interface Unit (BIU)
  (II) Execution Unit (EU)

(I) BUS INTERFACE UNIT

- This unit handle all the transfer of data and memory address on the bus for execution unit.
- It fetch instruction from memory.
- It read data from the port and memory and also write data to the memory and port.
- It generate 20 bit physical address for memory-acces.
- It support pipe lining by using 6 byte instruction queue.

DIFFERENT PARTS OF BIU

1- Segment Register

It is 16-bit register. In Bus Interface Unit (BIU) segment register divided into 4 parts :-
(I) Extra Segment Register (ES)
(II) Code Segment Register (CS)
(III) Stack Segment Register (SS)
(IV) Data Segment Register (DS).

(I) Extra Segment Register (ES)

- It is also a 16-bit register and containing 64KB Segment.

- It hold the base address for extra segment.
- It is multiplied by 10H to give 20 bit physical address of extra segment.

(II) Code Segment Register (CS)

- It is also 16-bit register and containing 64KB segment.
- It hold the base address for code segment.
- It is multiplied by 10H to give 20 bit physical address of code segment.

(III) Stack Segment Register (SS)

- It is also 16-bit register and containing 64KB segment.
- It hold the base address for stack segment.
- It is multiplied by 10H to give 20 bit physical address of stack segment.

(IV) Data Segment Register (DS)

- It is also 16-bit register and containing 64KB Segment.
- It hold the base address for Data segment.
- It is multiplied by 10H to give 20 bit physical address of ~~stack~~ data segment.

IP (INSTRUCTION POINTER)

- It is a 16-bit register.
- It hold offset of next instruction in a code segment register.
- Address of next instruction is calculated by (CS x 10H + IP)
- It is implemented by 1 after every instruction.

INSTRUCTION QUEUE

6byte Instruction Queue

- It is 6byte FIFO, use to implement pipeline.
- Fetching the next instruction while executing the current instruction is called as pipelining.
- BIU fetch next 6byte instruction from the code-segment and it store into the queue.

(II) EXECUTION UNIT (EU)

- It fetch instruction from the queue, decode and executing the Microprocessor.
- It perform Arithmatic, logical and internal data transfer operation within in the microprocessor.

DIFFERENT PARTS OF EXECUTION UNIT

GPR (General Purpose Register)

- 8086 Microprocessor has 4 16-bit GPR i.e. AX, BX, CX, DX.

- There are available to the programmer for storing the value during execution.
- Each 16-bit register is divided into 2 8bit registers. i.e. AH, AL, BH, BL, CH, CL, DH, DL.
- AX register hold operand and result during the multiplication and division.
- BX register hold the memory address in Indirect addressing mode or register Indirect Addressing mode.
- CX register hold the Instruction like rotate and loop operation.
- DX register is used with AX etc hold 32 bit data.

SPR (Special purpose register)

SP (Stack pointer)

- It is a 16-bit register.
- It hold offset address if the top of the stack.

BP (Base pointer)

- It is 16-bit register.
- It hold address of any location in the Stack Segment.

SEGMENT INDEX (SI)

- It is 16-bit register.

i.e. It hold the offset address for data segment.

DATA INDEX (DI)

- It is 16-bit register.
- It hold offset address of extra segment.

16 BIT ALU

- It is 16-bit register.
- It perform 8-bit or 16-bit Arithmetic & logic operation

Operand Register

- It is 16-bit register.
- It hold the operand temporarily.

INSTRUCTION REGISTER & DECODER

- The execution unit fetch opcode from the queue into the Instruction register and the Instruction decoder decode it and send out the instruction to the control circuit for execution.

FLAG REGISTER

D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
X	X	X	X	OF	DF	IF	TF	S	Z	X	Ac	X	P	X	CY

- It is 16-bit register and it is divided into "9 types of flags and that flags are "6" are conditional flag and "3" are control flag.

TRAP FLAG (TF)
INTERRUPT FLAG (IF)
DIRECTION FLAG (DF)
OVERFLOW FLAG (OF)

TRAP FLAG (TF)

- If TF = 1 the CPU automatically general Internal Interrupt after any Instruction.
- If TF = 0 then there is no Interrupt save there will be occur.

INTERRUPT FLAG (IF)

- If IF = 1 the CPU recognize the external Interrupt.
- If IF = 0 then there is no Interrupt will be occur

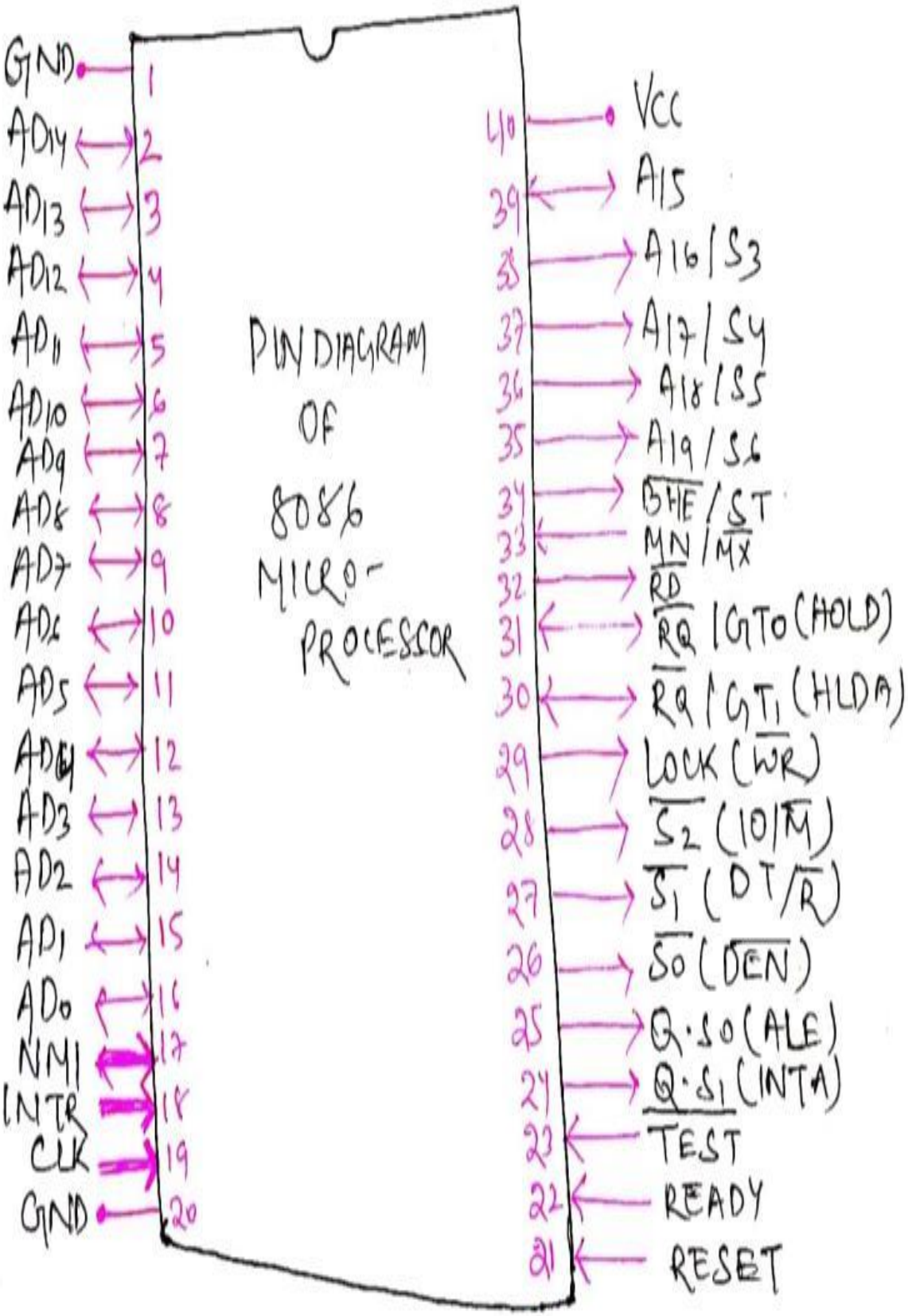
DIRECTION FLAG (DF)

- If DF = 1 the String Instruction will automatically decrement.
- If DF = 0 the string instruction will automatically increment the pointer.

OVERFLOW FLAG

- If OF = 1 the result is too large positive (+ve) number.
- If OF = 0 the result is too small Negative (-ve) number.

PIN STRUCTURE OF 8086 MICROPROCESSOR



DESCRIPTION OF EACH PIN

The total pin is divided in seven parts :-

(I) Data Bus
(II) Address Bus
(III) power supply & frequency signal
(IV) CLK signal
(V) Control & Status signal
(VI) Interrupt signal
(VII) Mode selection

(I) DATA BUS

- It is 16-bit and also it is Bi-directional Bus.
- To transfer the data between CPU and memory and I/O device ~~is divided into two parts~~ and CPU, data-Bus in use (AD_0 - AD_{15}).

(II) ADDRESS BUS

- It is a Uni-directional Bus.
- The Address Bus range is (AD_0 - AD_{19}).
- The Address Bus is divided into two parts :-
  (I) low Address Bus (AD_0 - AD_{15})
  (II) High Address Bus (AD_{16} - AD_{19})

(III) POWER SUPPLY & FREQUENCE SIGNAL

- Vcc is the power supply pin and it is present in the 40 no pin.
- It require +5 V and ±10 V power supply.
- The version of frequence signal is 5MHz, 8MHz, 10MHz.
- The pin no. 1 and 20 is used as Ground pin.

(IV) CLOCK SIGNAL

- The Pin no. 19 in the clock signal.
- 8086 Microprocessor require clk signal from the internal device synchronize the internal ~~org~~ option in the proces with the maximum frequency range from 5-10 MHz.

(V) CONTROL AND STATUS SIGNAL

RD

- It is active low signal.
- The signal is used for read the data
- When it is on how the MP need the data from the memory I/O device

READY :-

- The I/P, O/p and memory send an acknowledgement through this pin.
- When this pin goes high the penophenal device is ready the trasfer the data.

RESET

- It is active high signal.
- It is use to reset the systemb reg and other device

(VI) INTERRUPT SIGNAL

INTR (Interrupt Request)

- It is a Maskable interrupt signal.
- It is I/p signal which INTR goes high.
- The Micro proceson complete the current intruction which cere being executed.

NMI

- It is a Non-movable interrupt signal.

(vii) MODE SELECTION

8086 can be operate in two mode:-
(i) Minimum mode
(ii) Maximum Mode

(ii) MAXIMUM MODE

~~INTA, ALE, DEN, DT/R, IO/M, WR, RQ~~
~~HOLD, HLDA, QS0 - QS1, LOCK, RQ~~

- When more than one processor is used in a Microcomputer system then 8086 in an Maximum mode of operation.

→ When MN/$\overline{MX}$ in low the CPU operate in maximum mode.

→ The pin no. 24 to 31 and 34 to 38 in used as maximum mode.

Q.S0 - Q.S1

The pin no. 24 & 25 in the Instruction queue Status.

QS0	QS1	Commands
0	0	Nop
0	1	opcode fetch (1 byte opcode from the queue)
1	0	Empty the queue.
1	1	Next byte opcode from the queue.

S0, S1, S2

The pin no. 26, 24, 28 in the Status-Signal.

S_2	S_1	S_0	Comment
0	0	0	INTA
0	0	1	$\overline{RD}$ Data from I/O port
0	1	0	$\overline{WR}$ Data from I/O port
0	1	1	HLT
1	0	0	Opcode fetch
1	0	1	MR
1	1	0	MW
1	1	1	Positive state

LOCK

- The pin no. 29 is the lock signal.
- It is active low signal.
- When it is low all the interrupt are masked and no hold request is granted.

$\overline{RQ}$ / $\overline{GT_1}$ - $\overline{GT_0}$

- The pin no: 30 & 31 is the request and grant signal.
- It is used by the other processor requesting the CPU to release the systembus.
- When the signal is receive by the CPU, it sent acknowledgement signal.
- The $\overline{RQ}$ / $\overline{GT_0}$ is the highest priority than $\overline{RQ}$ / $\overline{GT_1}$.

ADDRESS BUS / STATUS SIGNAL

- This pin no. 35 to 38 is the Address Bus/ Status Signal.
- It is high order address Bus. The address bus A_{16}, A_{17}, A_{18}, A_{19} are multiplex with status signal i.e. S_3, S_4, S_5, S_6 respectively.

BHE/S_7

- It is 34 in the bus high enable signal.
- BHE signal is multiplex with status signal S_7
- It is active low signal.
- It is use to enable the data on the MSB of data bus i.e. D_8-D_{15}.

(1) MINIMUM MODE

- When only one processor is used in a micro computer system then 8086 is in minimum mode of operation.
- MN/$\overline{MX}$ is high the CPU operate in minimum mode. In minimum mode the pin no. 24-31 is used.

INTA

- The pin no. 24 is the interrupt acknowledgement signal.
- Once receiving the interrupt signal, the microprocessor issue an acknowledgement signal through this pin.
- It is Active low signal.

ALE

- This pin no. 25 is the ALE signal.
- It goes high during the T_1 state.

DEN (Data Enable signal)

- The pin no. 26 is the data enable signal. It is active low signal.
- It is use to enable the trans receiver. The trans receiver is a device use to separate the data from address and databus.

DT/$\overline{R}$ (Data Transmitter & Receiver)

- The pin no. 27 is the data transmitter and receiver.
- It decide the direction of data flow through the trans receiver. When it is high the data is transmitted out and vice-versa.

IO/$\overline{M}$

- The pin no. 28 is the input output and memory signal.
- When it is high the micro processor want to access I/O operation when it is low the microprocessor want to acces memory.

$\overline{WR}$

- The pin no. 29 is the write signal.
- When this signal goes low CPU perform write operation and write the data into the memory.

HLDA

- The pin no. 30 is the hold acknowledgement signal.
- It is issued by the Microprocessor when it receive HOLD signal.
- It is active high signal, when hold request is remove then it goes low.

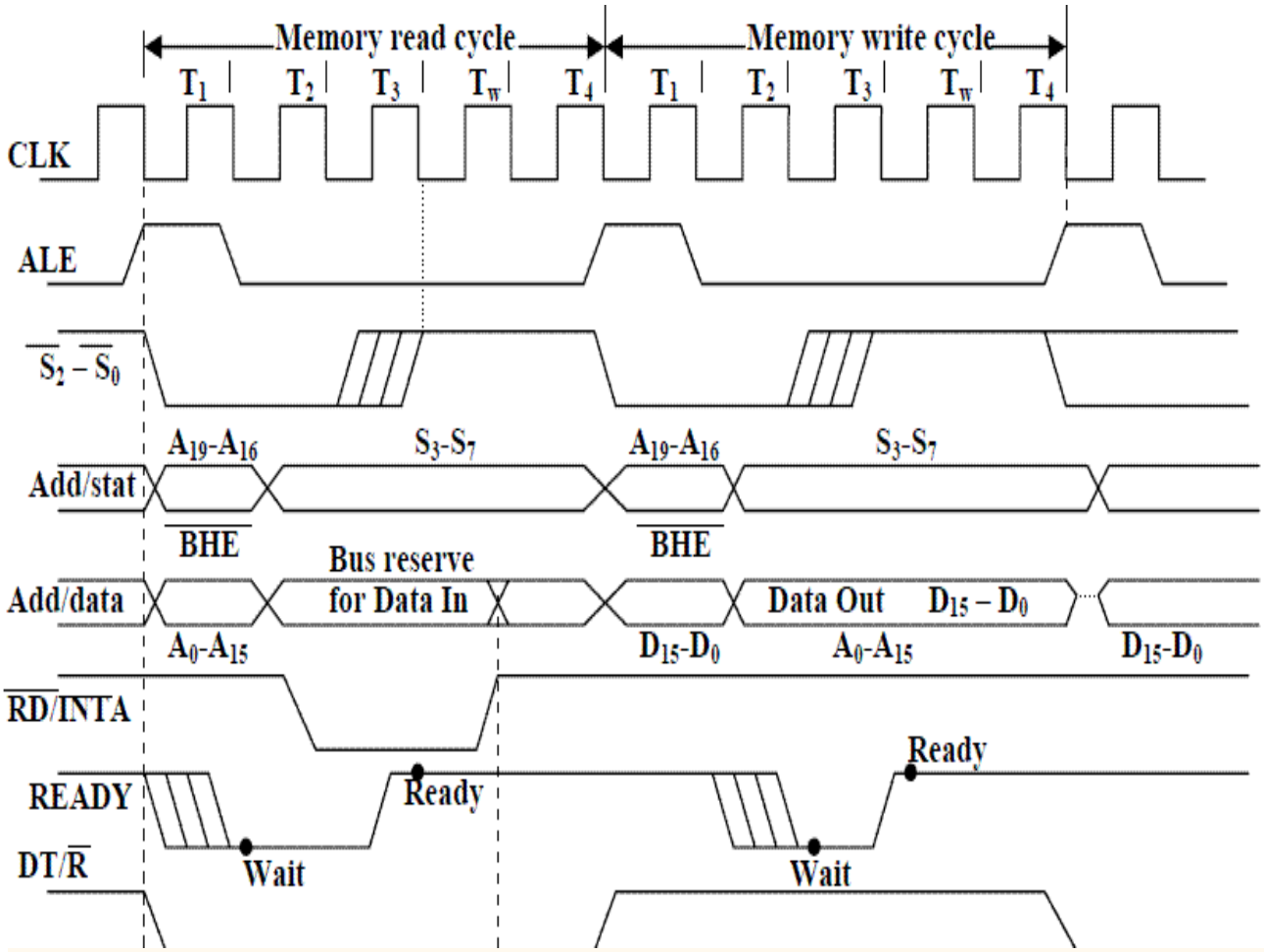
HOLD

- The pin no. 31 is the HOLD signal.
- When another device in the microcomputer system are used to data bus and address bus it send hold request to the CPU through this pin.
- It is active low signal.

General bus operation of 8086

- The 8086 has a combined address and data bus commonly referred as a time multiplexed address and data bus.
- The main reason behind multiplexing address and data over the same pins is the maximum utilisation of processor pins and it facilitates the use of 40 pin standard DIP package.
- The bus can be demultiplexed using a few latches and transreceivers, when ever required.
- Basically, all the processor bus cycles consist of at least four clock cycles. These are referred to as T1, T2, T3, T4. The address is transmitted by the processor during T1. It is present on the bus only for one cycle.
- The negative edge of this ALE pulse is used to separate the address and the data or status information. In maximum mode, the status lines S0, S1 and S2 are used to indicate the type of operation.

- Status bits S3 to S7 are multiplexed with higher order address bits and the BHE signal. Address is valid during T1 while status bits S3 to S7 are valid during T2 through T4.



Maximum mode

- In the maximum mode, the 8086 is operated by strapping the MN/MX pin to ground.
- In this mode, the processor derives the status signal S2, S1, S0. Another chip called bus controller derives the control signal using this status information .
- In the maximum mode, there may be more than one microprocessor in the system configuration.

Minimum mode

- In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/MX pin to logic 1.
- In this mode, all the control signals are given out by the microprocessor chip itself.
- There is a single microprocessor in the minimum mode system.

Memory Organization

As far as we know 8086 is 16-bit processor that can supports 1Mbyte (i.e. 20-bit address bus: 220) of external memory over the address range 0000016 to FFFFF16. The 8086 organizes memory as individual bytes of data. The 8086 can access any two consecutive bytes as a word of data. The lower-addressed byte is the least significant byte of the word, and the higher- addressed byte is its most significant byte.

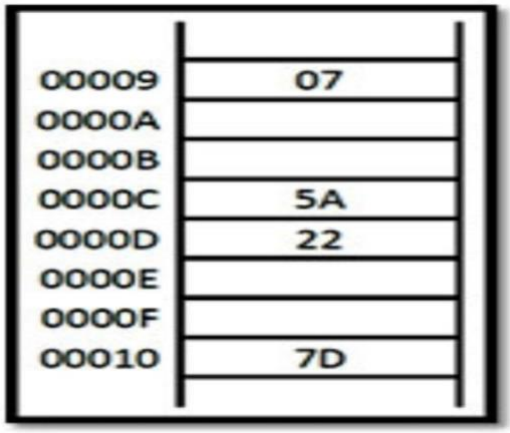


Figure: Part of 1 Mbyte Memory

The above figure represents: storage location of address 0000916 contains the value 716, while the location of address 0001016 contains the value 7D16. The 16-bit word 225A16is stored in the locations 0000C16 to 0000D16

The word of data is at an even-address boundary (i.e. address of least significant byte is even) is called aligned word. The word of data is at an odd-address boundary is called misaligned word, as shown in Figure below.

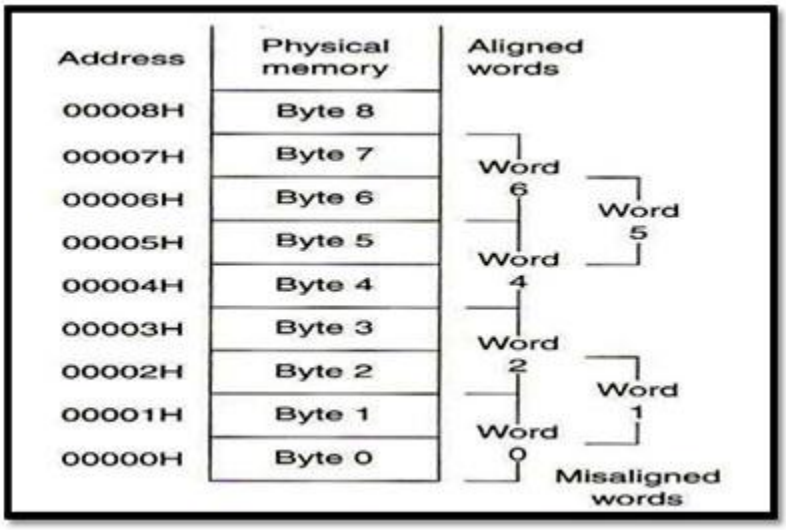


Figure: Aligned and misaligned word

To store double word four locations are needed. The double word that it's least significant byte address is a multiple of 4 (e.g. 0 16, 416, 816 ...) is called aligned double word. The double word at address of non-multiples of 4 is called misaligned double word shown in Figure below.

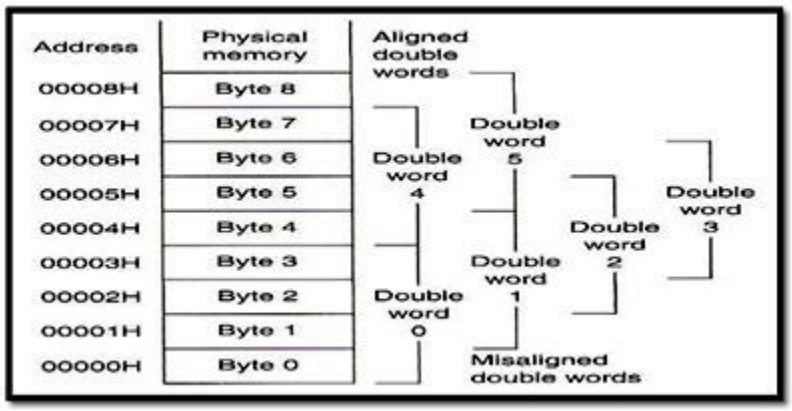


Figure: Aligned and misaligned double word

a) Memory segmentation

The size of address bus of 8086 is 20 and is able to address 1 Mbytes ( ) of physical memory, but all this memory is not active at one time. Actually, this 1Mbytes of memory are partitioned into 16 parts named as segments. Size of the each segment is 64Kbytes (65,536).
Only four of these segments are active at a time:

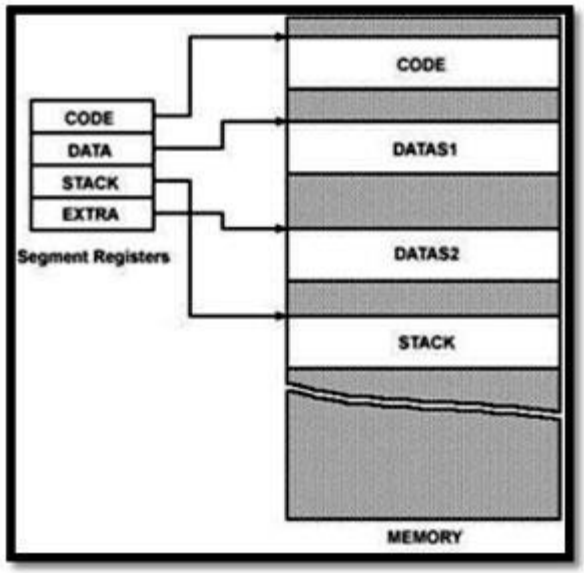
v. Code segment holds the program instruction codes

vi. Stack segment is used to store interrupt and subroutine return addresses

vii. Data segment stores data for the program

viii. Extra segment is an extra data segment (often used for shared data)

ix. Each of these segments are addressed by an address stored in corresponding segment registers: CS(code segment), SS(stack segment), DS(data segment), and ES(extra segment). These registers contain a 16-bit base address that points to the lowest addressed byte of the segment. Because the segment registers cannot store 20 bits, they only store the upper 16 bits. The BIU takes care of this problem by appending four 0's to the low-order bits of the segment register. In effect, this multiplies the segment register contents by 16.



The segment registers are user accessible, which means that the programmer can change the content of segment registers through software.

b) Programming model:

How can a 20-bit address be obtained, if there are only 16-bit registers? However, the largest register is only 16 bits (64k); so physical addresses have to be calculated. These calculations are done in hardware within the microprocessor.

The 16-bit contents of segment register gives the starting/ base address of particular segment. To address a specific memory location within a segment we need an offset address. The offset address is also 16-bit wide and it is provided by one of the associated pointer or index register.

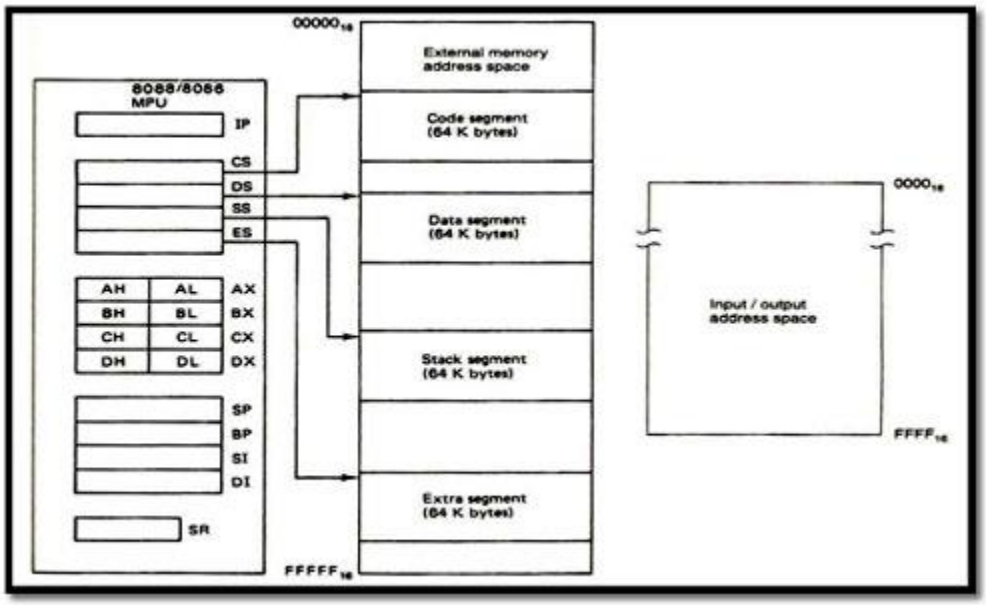


Figure: Software model of 8086 microprocessor

To be able to program a microprocessor, one does not need to know all of its hardware architectural features. What is important to the programmer is being aware of the various registers within the device and to understand their purpose, functions, operating capabilities, and limitations.

The above figure illustrates the software architecture of the 8086 microprocessor. From this diagram, we see that it includes fourteenl6-bit internal registers: the instruction pointer (IP), four data registers (AX, BX, CX, and DX), two pointer registers (BP and SP), two index registers (SI and DI), four segment registers (CS, DS, SS, and ES) and status register (SR), with nine of its bits implemented as status and control flags.

The point to note is that the beginning segment address must begin at an address divisible by 16.Also note that the four segments need not be defined separately. It is allowable for all four segments to completely overlap (CS = DS = ES = SS).

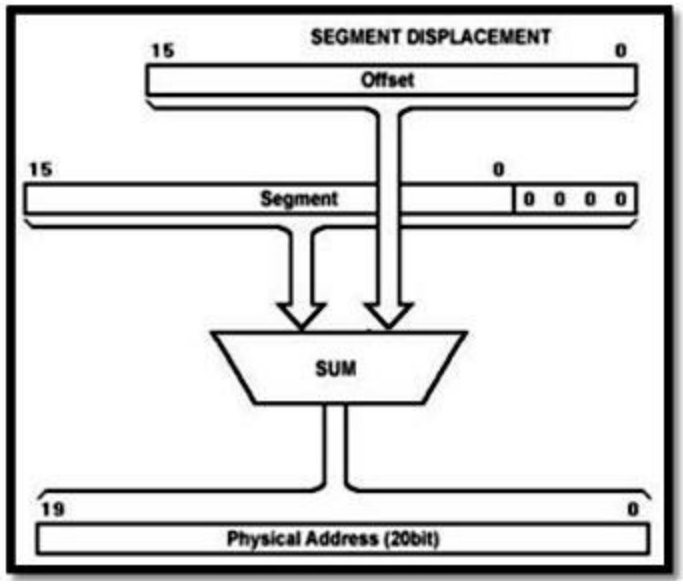
c) Logical and Physical Address

Addresses within a segment can range from address 00000h to address 0FFFFh. This corresponds to the 64K-bytelength of the segment. An address within a segment is called an offset or logical address.

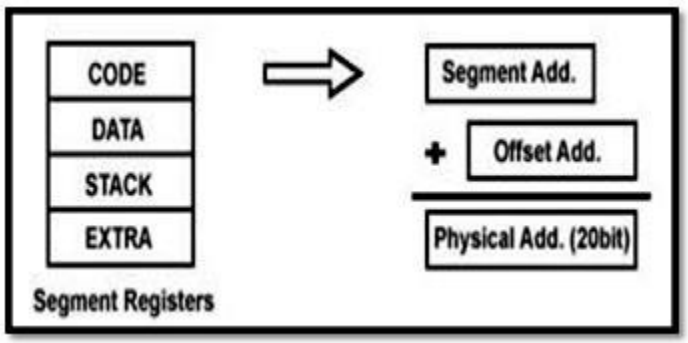
A logical address gives the displacement from the base address of the segment to the desired location within it, as opposed to its "real" address,

which maps directly anywhere into the 1 MByte memory space. This "real" address is called the physical address.

What is the difference between the physical and the logical address?
The physical address is 20 bits long and corresponds to the actual binary code output by the BIU on the address bus lines. The logical address is an offset from location 0 of a given segment.



You should also be careful when writing addresses on paper to do so clearly. To specify the logical address XXXX in the stack segment, use the convention SS:XXXX, which is equal to [SS] * 16 + XXXX.



Logical address is in the form of: Base Address: Offset
Offset is the displacement of the memory location from the starting location of the segment.
To calculate the physical address of the memory, BIU uses the following formula:

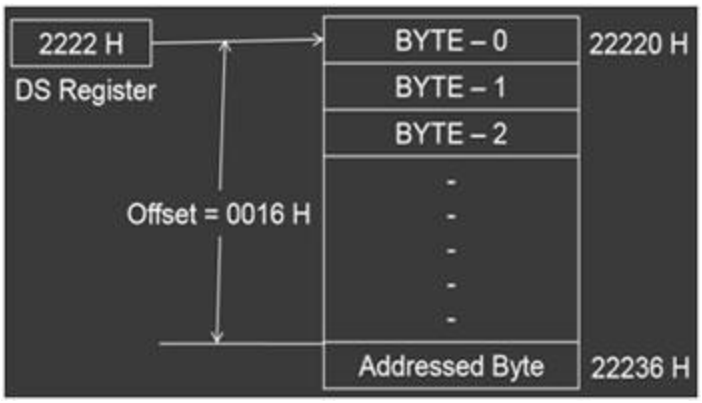
Physical Address = Base Address of Segment * 16 + Offset

Example:
The value of Data Segment Register (DS) is 2222H.

To convert this 16-bit address into 20-bit, the BIU appends 0H to the LSB (by multiplying with 16) of the address. After appending, the starting address of the Data Segment becomes 22220H.

Data at any location has a logical address specified as:2222H: 0016H

Where 0016H is the offset, 2222 H is the value of DS
Therefore the physical address:22220H + 0016H
: 22236 H



The following tables describes the default offset values to the corresponding memory segments.

Segment	Offset Registers	Function
CS	IP	Address of the next instruction
DS	BX, DI, SI	Address of data
SS	SP, BP	Address in the stack
ES	BX, DI, SI	Address of destination data (for string operations)

Some of the advantages of memory segmentation in the 8086 are as follows:

- With the help of memory segmentation a user is able to work with registers having only 16-bits.
- The data and the user's 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.

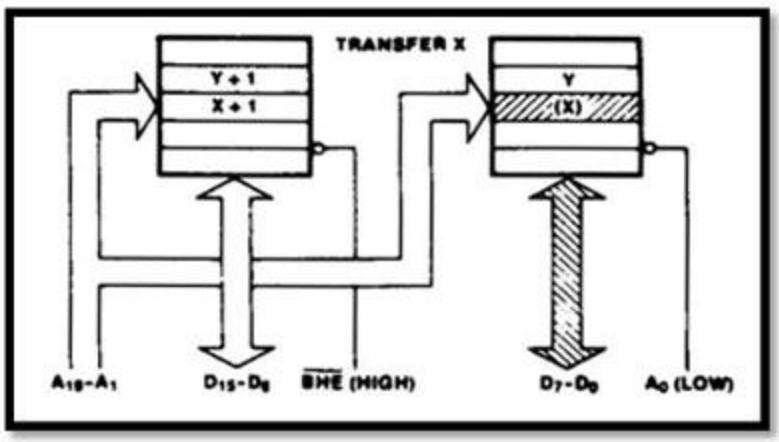
d) Physical memory organization:

The 8086's 1Mbyte memory address space is divided in to two independent 512Kbyte banks: the low (even) bank and the high (odd) bank. Data bytes associated with an even address (0000016, 0000216, etc.) reside in the low bank, and those with odd addresses (0000116, 0000316, etc.) reside in the high bank.

Address bits A1 through A19 select the storage location that is to be accessed. They are applied to both banks in parallel. A0and bank high enable (BHE) are used as bank-select signals.

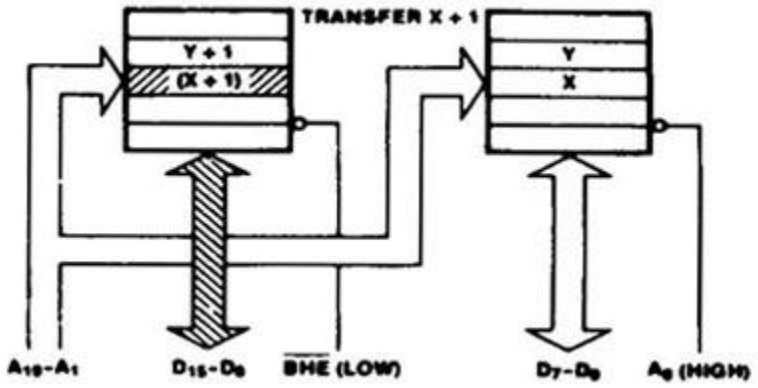
The four different cases that happen during accessing data:

Case 1: When a byte of data at an even address (such as X) is to be accessed:



- A0 is set to logic 0 to enable the low bank of memory.
- BHE is set to logic 1 to disable the high bank.

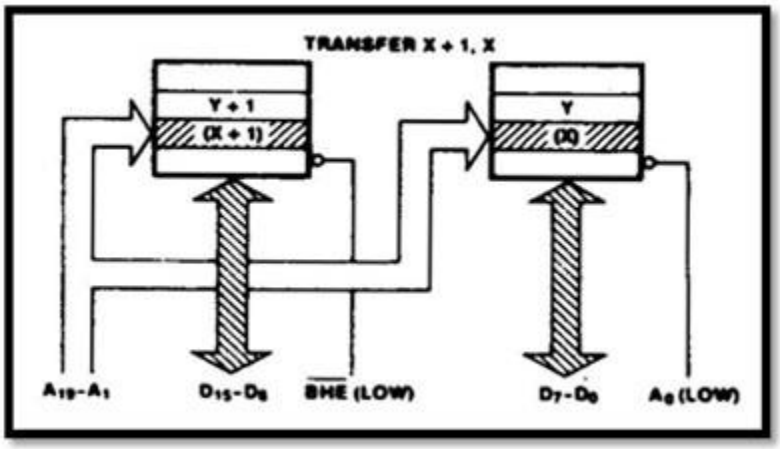
Case 2: When a byte of data at an odd address (such as X+1) is to be accessed:



iv. A0is set to logic 1 to disable the low bank of memory.

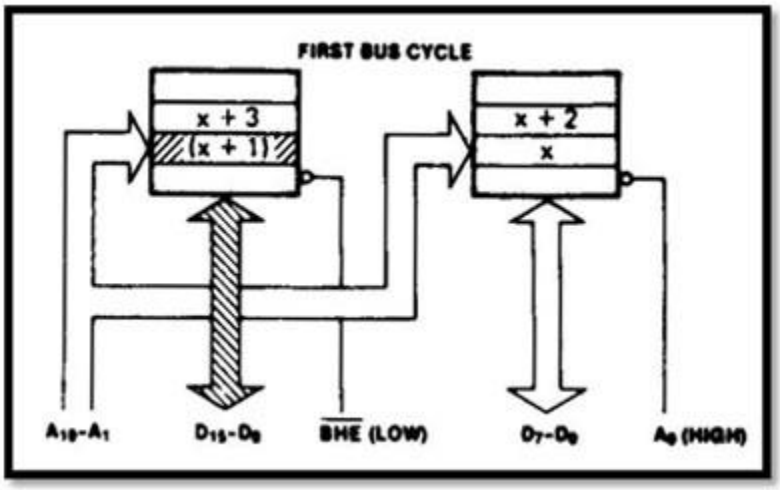
v. BHE is set to logic 0 to enable the high bank.

Case 3: When a word of data at an even address (aligned word) is to be accessed:



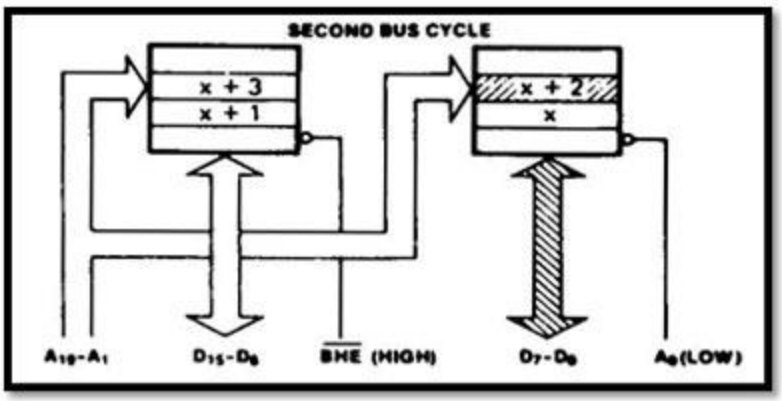
- A0 is set to logic 0 to enable the low bank of memory.
- BHE is set to logic 0 to enable the high bank.

Case 4: When a word of data at an odd address (misaligned word) is to be accessed, then the 8086 need two bus cycles to access it:
a) During the first bus cycle, the odd byte of the word (in the high bank) is addressed



- A0 is set to logic 1 to disable the low bank of memory
- BHE is set to logic 0 to enable the high bank.

b) During the second bus cycle, the odd byte of the word (in the low bank) is addressed

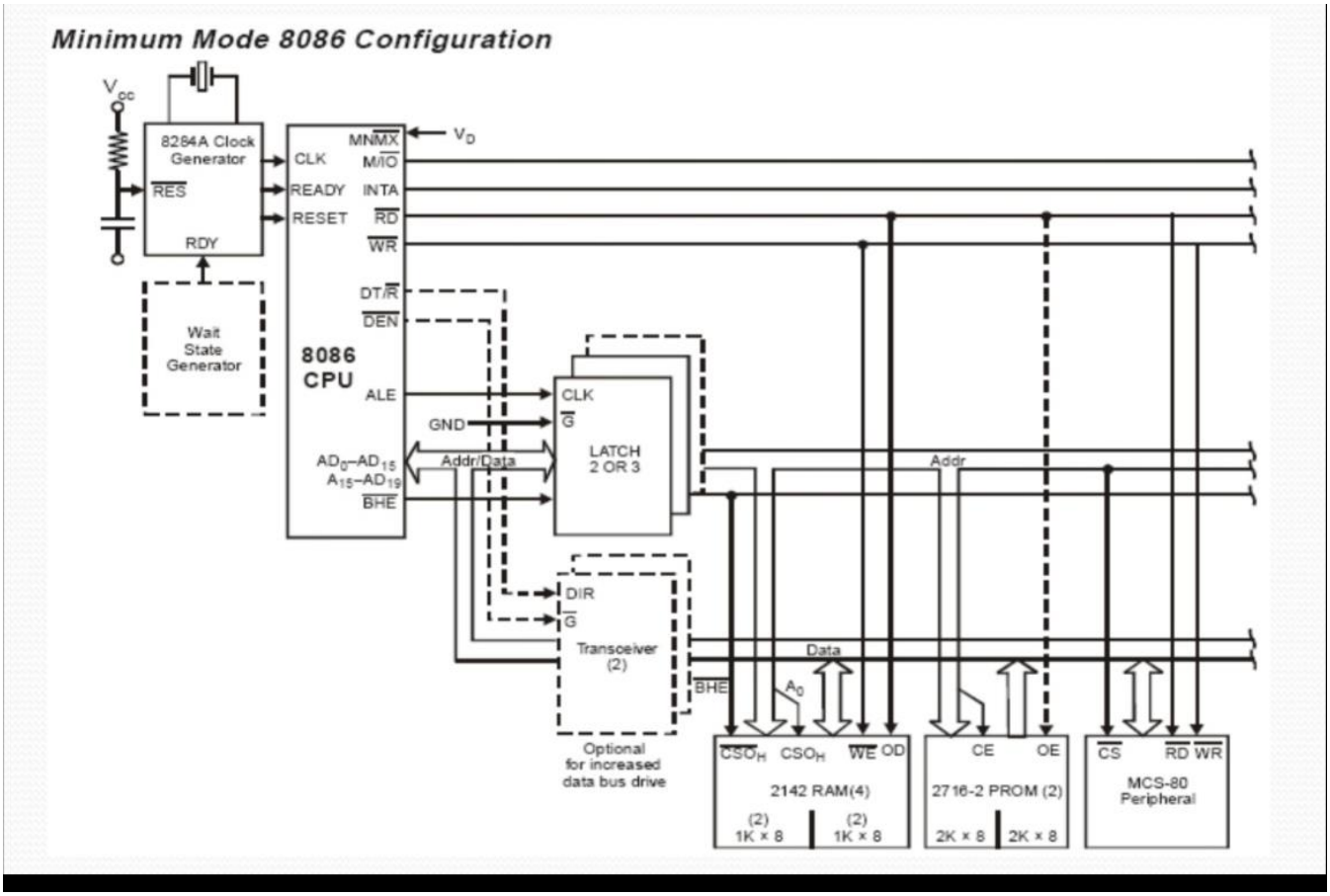


1. A0is set to logic 0 to enable the low bank of memory.
2. BHE is set to logic 1 to disable the high bank.

Minimum Mode 8086 System

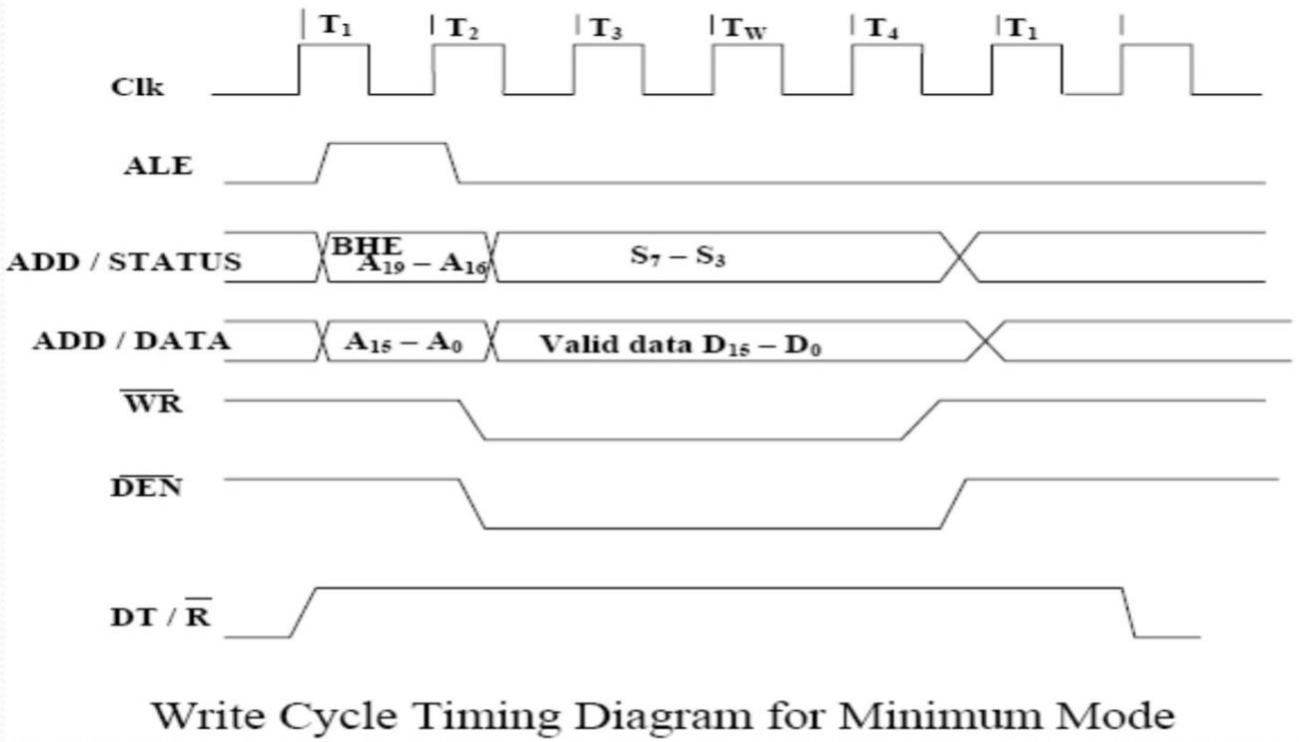
- The microprocessor 8086 is operated in minimum mode by strapping its MN/MX pin to logic 1.
- In this mode, all the control signals are given out by the microprocessor chip itself. There is a single microprocessor in the minimum mode system.
- The remaining components in the system are latches, transreceivers, clock generator, memory and I/O devices.
- Latches are generally buffered output D-type flip-flops like 74LS373 or 8282. They are used for separating the valid address from the multiplexed

address/data signals and are controlled by the ALE signal generated by 8086.



- Trans receivers are the bidirectional buffers and some times they are called as data amplifiers. They are required to separate the valid data from the time multiplexed address/data signals. They are controlled by two signals namely, DEN and DT/R.
- The DEN signal indicates the direction of data, i.e. from or to the processor.
- The system contains memory for the monitor and users program storage. Usually, EPROM are used for monitor storage, while RAM for users program storage. A system may contain I/O devices.
- The opcode fetch and read cycles are similar. Hence the timing diagram can be categorized in two parts, the first is the timing diagram for read cycle and the second is the timing diagram for write cycle.
- The read cycle begins in T1 with the assertion of address latch enable (ALE) signal and also M / IO signal. During the negative going edge of this signal, the valid address is latched on the local bus
- The BHE and A0 signals address low, high or both bytes. From T1 to T4 , the M/IO signal indicates a memory or I/O operation.

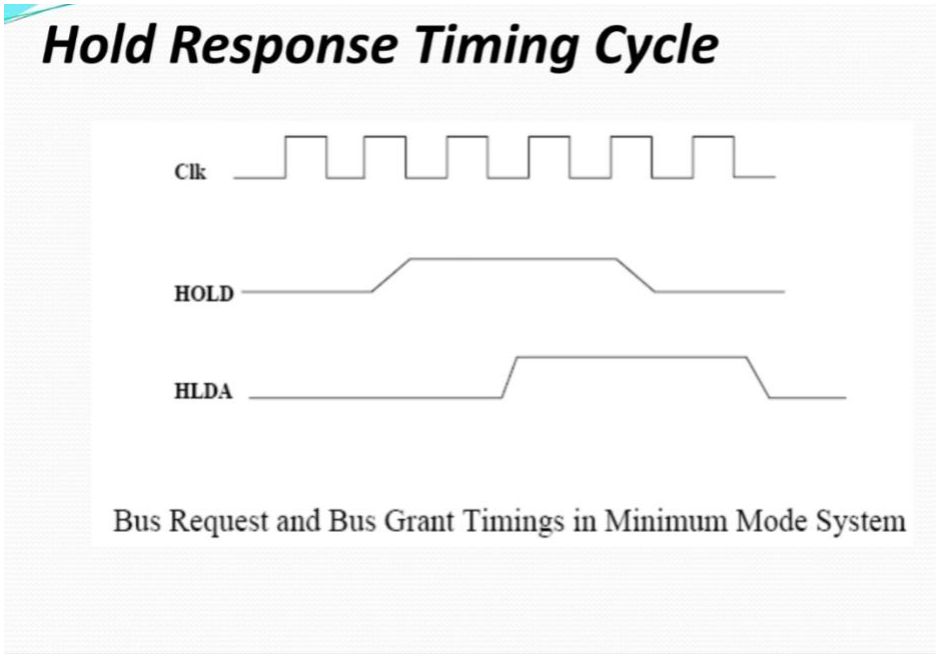
- At T2, the address is removed from the local bus and is sent to the output. The bus is then tristated. The read (RD) control signal is also activated in T2.
- The read (RD) signal causes the address device to enable its data bus drivers. After RD goes low, the valid data is available on the data bus.
- The addressed device will drive the READY line high. When the processor returns the read signal to high level, the addressed device will again tristate its bus drivers.
- A write cycle also begins with the assertion of ALE and the emission of the address.
- The M/IO signal is again asserted to indicate a memory or I/O operation. In T2, after sending the address in T1, the processor sends the data to be written to the addressed location.
- The data remains on the bus until middle of T4 state. The WR becomes active at the beginning of T2 (unlike RD is somewhat delayed in T2 to provide time for floating).
- The BHE and A0 signals are used to select the proper byte or bytes of memory or I/O word to be read or write.
- The M/IO, RD and WR signals indicate the type of data transfer as specified in table below.



Write Cycle Timing Diagram for Minimum Mode

Hold Response sequence:

- The HOLD pin is checked at leading edge of each clock pulse. IfIf it is received active by the processor before T4 of the previous cycle or during T1 state of the current cycle, the CPU activates HLDA in the next clock cycle and for succeeding bus cycles, the bus will be given to another requesting master.
- The control of the bus is not regained by the processor until the requesting master does not drop the HOLD pin low.
- When the request is dropped by the requesting master, the HLDA is dropped by the processor at the trailing edge of the next clock.

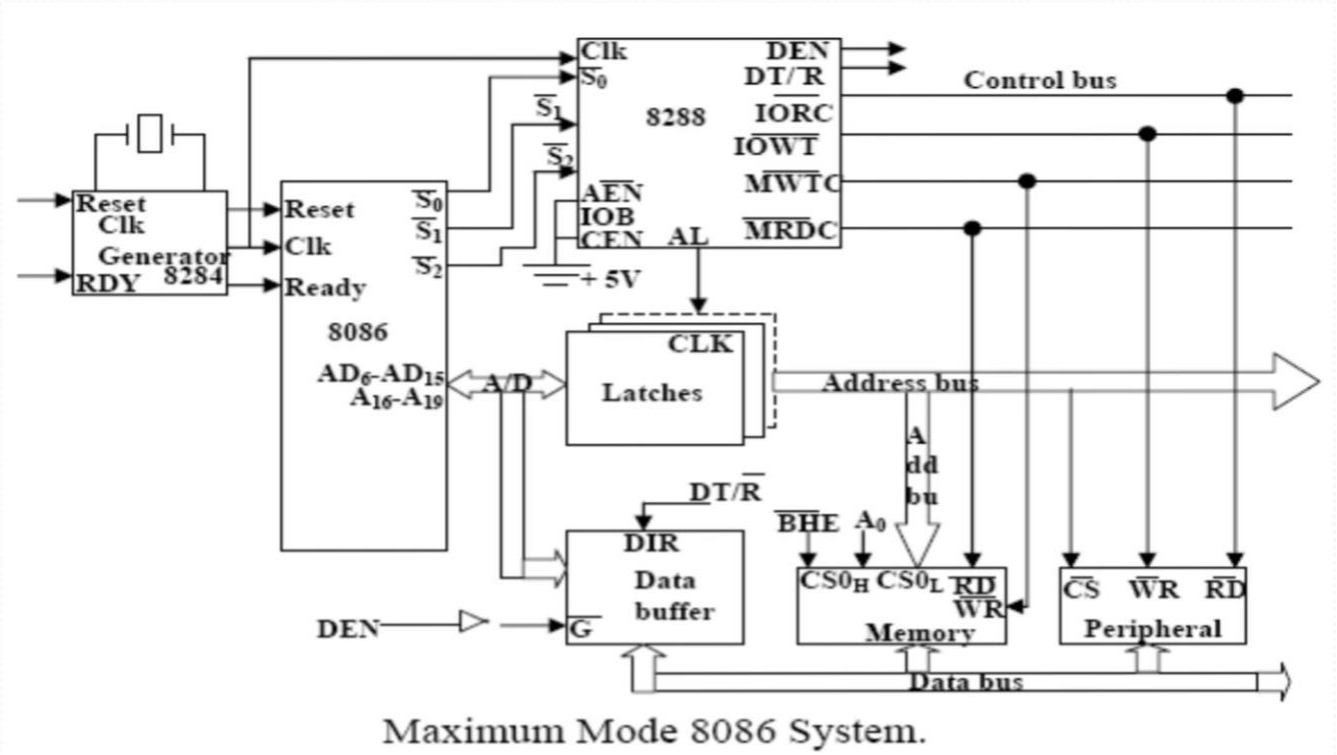


Maximum Mode 8086 System

- In the maximum mode, the 8086 is operated by strapping the MN/MX pin to ground.
- In this mode, the processor derives the status signal S2, S1, S0. Another chip called bus controller derives the control signal using this status information .
- In the maximum mode, there may be more than one microprocessor in the system configuration. The components in the system are same as in the minimum mode system.

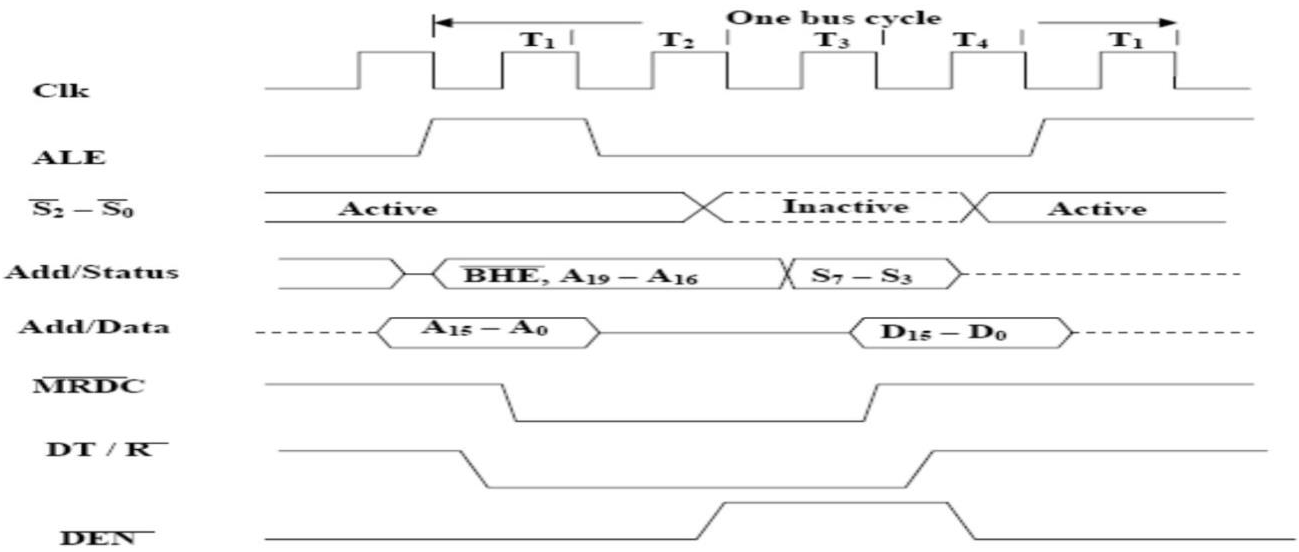
- The basic function of the bus controller chip IC8288, is to derive control signals like RD and WR (for memory and I/O devices), DEN, DT/R, ALE etc. using the information by the processor on the status linline.
- The bus controller chip has input lines S2, S1, S0 and CLK. TheseThese inputs to 8288 are driven by CPU.
- It derives the outputs ALE, DEN, DT/R, MRDC, MWTC, AMWC, IORC, IOWC and AIOWC. The AEN, IOB and CEN pins are specially useful for multiprocessor systems.
- AEN and IOB are generally grounded. CEN pin is usually tied to +5V. The significance of the MCE/PDEN output depends upon the status of the IOB pin.
- INTA pin used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device.
- IORC, IOWC are I/O read command and I/O write command signals respectively . These signals enable an IO interface to read or write the data from or to the address port.
- The MRDC, MWTC are memory read command and memory write command signals respectively and may be used as memory read or write signals.
- All these command signals instructs the memory to accept or send data from or to the bus.
- Here the only difference between in timing diagram between minimum mode and maximum mode is the status signals used and the available control and advanced command signals.

Maximum Mode Configuration For 8086

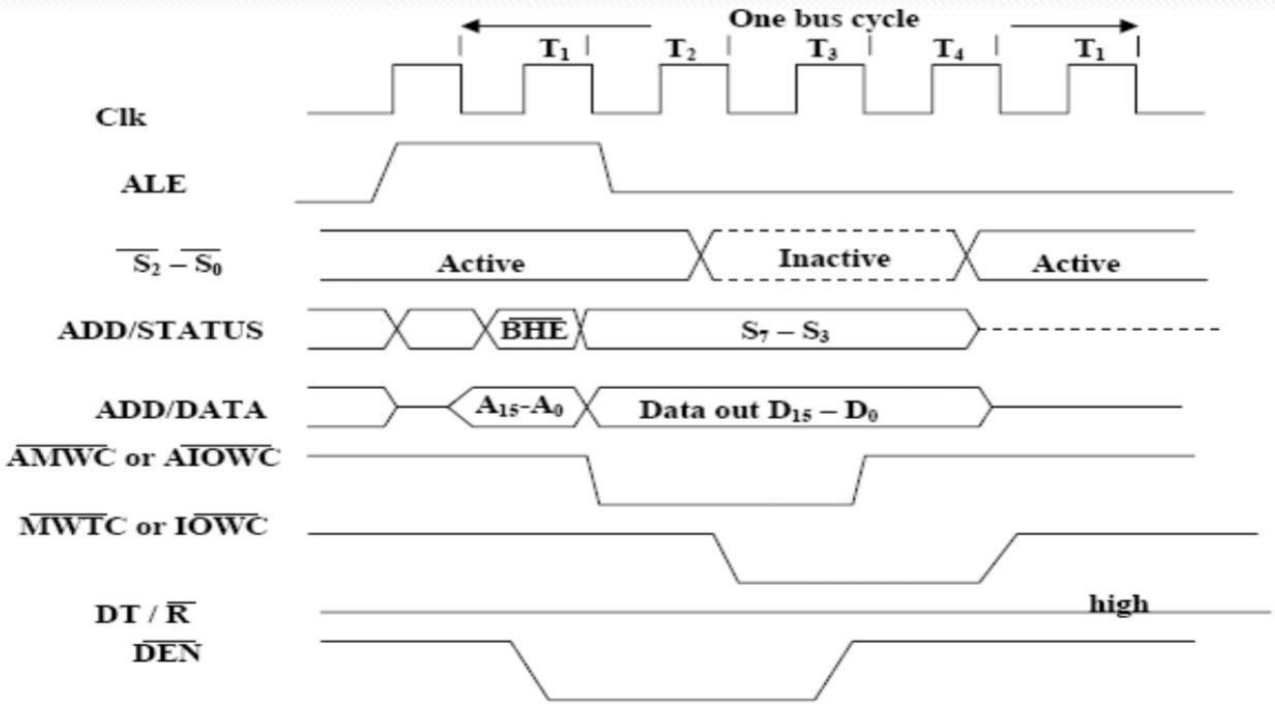


Maximum Mode 8086 System.

- R0, S1, S2 are set at the beginning of bus cycle.8288 bus controller will output a pulse as on the ALE and apply a required signal to its DT / R pin during T1.
- In T2, 8288 will set DEN=1 thus enabling transceivers, and for an input it will activate MRDC or IORC. These signals are activated until T4.
- For an output, the AMWC or AIOWC is activated from T2 to T4 and MWTC or IOWC is activated from T3 to T4.
- The status bit S0 to S2 remains active until T3 and become passive during T3 and T4.
- If reader input is not activated before T3, wait state will be inserted between T3 and T4.



Memory Read Timing in Maximum Mode

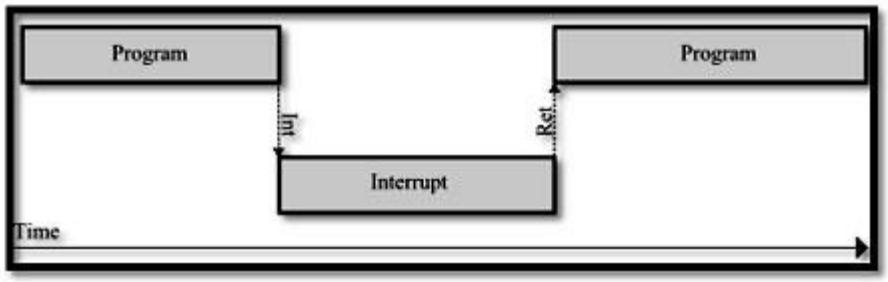


Memory Write Timing in Maximum mode.

Interrupt

An INTERRUPT is a condition that causes the microprocessor to temporarily work on a different task and then return to its previous task. Interrupt is an event or signal that request to attention of CPU.

Whenever an interrupt occurs the processor completes the execution of the current instruction and starts the execution of an Interrupt Service Routine (ISR) or Interrupt Handler. ISR is a program that tells the processor what to do when the interrupt occurs. After the execution of ISR, control returns back to the main routine where it was interrupted.



Whenever an interrupt is occurred, it will be acknowledged by the processor at the end of the current memory cycle. The processor then services the interrupt by branching to a special service routine written to handle that particular interrupt. Upon servicing the device, the processor is then instructed to continue with what is was doing previously by use of the "return from interrupt" instruction.

The status of the program being executed must be saved first. The processors registers will be saved on the stack, or at very least, the program counter will be saved. Preserving those registers which are not saved will be the responsibility of the interrupt service routine. Once the program counter has been saved, the processor will branch to the address of the service routine.

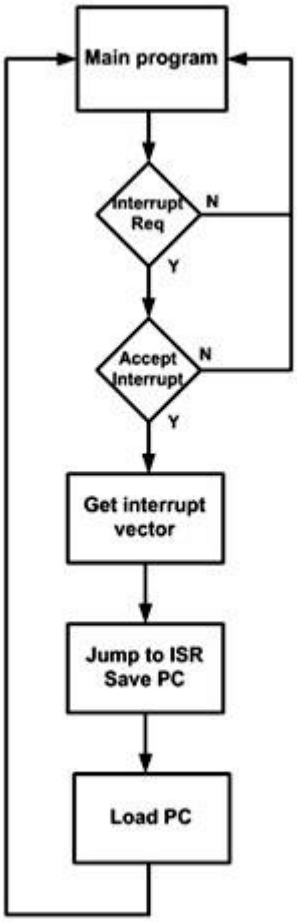


Figure: Interrupt processing flow

Purpose of Interrupts

As we studied, the Microprocessor can serve several devices. There are two ways to offer service: Interrupts and Polling.

x. The advantage of interrupts is that the microprocessor can serve many devices (not all at the same time, of course); each device can get the attention of the microprocessor based on the priority assigned to it.

xi. The polling method cannot assign priority because it checks all devices in a round-robin fashion.

xii. More importantly, in the interrupt method the microprocessor can also ignore (mask) a device request for service.

xiii. This is not possible with the polling method.

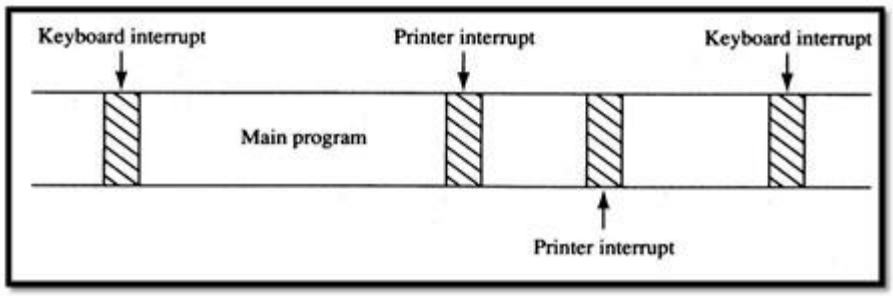
xiv. The most important reason that the interrupt method is preferable is that the polling method wastes much of the microprocessor's time by polling devices that do not need service.

xv. So interrupts are used to avoid tying down the microprocessor.

To understand the difference better, consider this example. The polling method is very much similar to a salesperson. The salesman goes door-

to-door requesting to buy his product. Like processor keeps monitoring the flags or signals one by one for all devices. Interrupt is very similar to a shopkeeper. Whosever needs a service or product goes to him and approaches him. Like, when the flags or signals are received, they notify the processor that they need its service.

Interrupts are useful when interfacing I/O devices with low data-transfer rates, like a keyboard or a mouse, in which case polling the device wastes valuable processing time

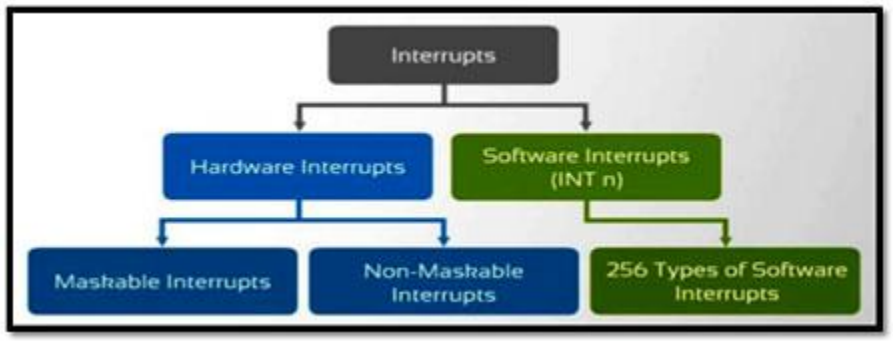


Above time line shows typing on a keyboard, a printer removing data from memory, and a program executing. The keyboard interrupt service procedure, called by the keyboard interrupt, and the printer interrupt service procedure each take little time to execute

Types of Interrupts

In general there are two types of Interrupts:

- Internal (or) Software Interrupts are triggered by a software instruction and operate similarly to a jump or branch instruction.
- External (or) Hardware Interrupts are caused by an external hardware module.



SOFTWARE INTERRUPTS–

INT nn is invoked software (sequence ofcode)

Examples:

- DOS INT 21H, BIOS INT 10H.
- INT 00 (divide error)
- INT 01 (single step)
- INT 03 (breakpoint)
- INT 04 (signed number overflow)

HARDWARE INTERRUPTS

Hardware interrupts are generated by hardware devices when something unusual happens; this could be a key-press or a mouse move or any other action.

Maskable Interrupts:

The processor can inhibit certain types of interrupts by use of a special interrupt mask bit. This mask bit is part of theflags/condition code register, or a special interrupt register. In the 8086 microprocessor if this bit is clear, and aninterrupt request occurs on the Interrupt Request input, it is ignored.

Non-Maskable Interrupts:

There are some interrupts which cannot be masked out or ignored by the processor. These are associated with highpriority tasks which cannot be ignored (like memory parity or bus faults). In general, most processors support the Non-Maskable Interrupt (NMI). This interrupt has absolute priority, and when it occurs, the processor will finish thecurrent memory cycle, then branch to a special routine written to handle the interrupt request.

Interrupt Service Routine

For every interrupt, there must be an interrupt service routine (ISR), or interrupt handler. When an interrupt is invoked, the microprocessor runs the interrupt service routine. For every interrupt, there is a fixed location in memory that holds the address of its ISR. The group of memory locations set aside to hold the addresses of ISRs is called the interrupt vector table.

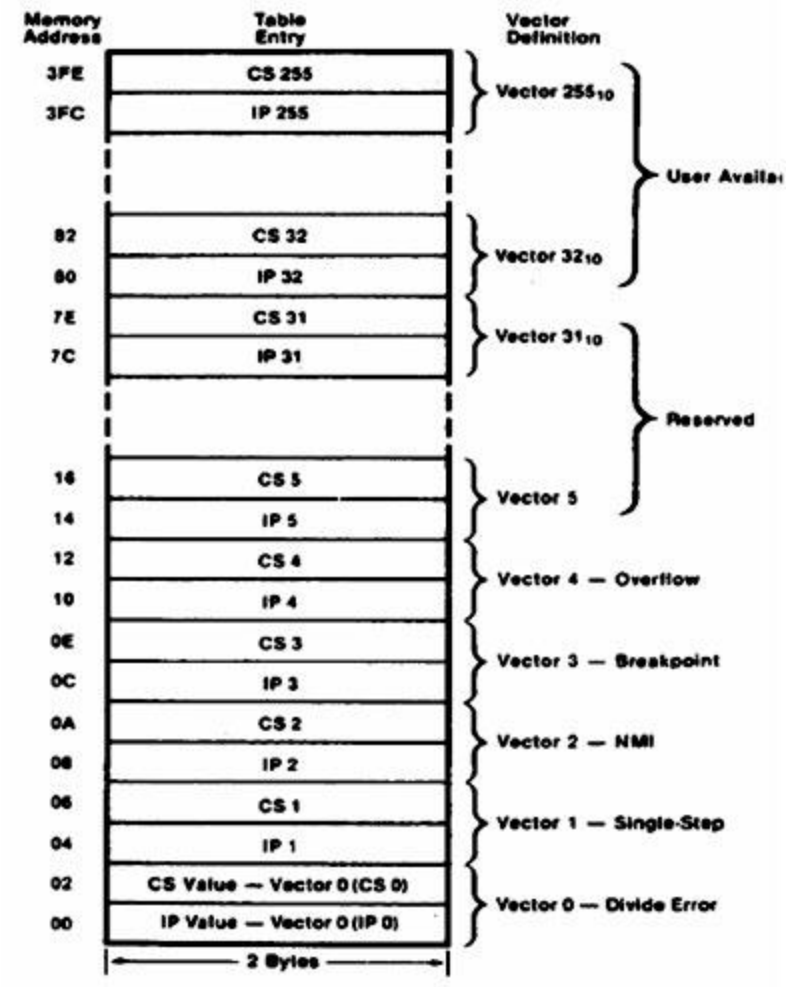
When an interrupt is occurred, the microprocessor stops execution of current instruction. It transfers the content of program counter into stack. It also stores the current status of the interrupts internally but not on stack. After this, it jumps to the memory location specified by Interrupt Vector Table (IVT). After that the code written on that memory area will

execute.

Interrupt Vector Table

The first 1Kbyte of memory of 8086 (00000 to003FF) is set aside as a table for storing thestarting addresses of Interrupt Service Procedures(ISP).Since 4-bytes are required for storing starting addresses of ISPs, the table can hold 256 Interrupt procedures.

The starting address of an ISP is often called theInterrupt Vector or Interrupt Pointer. Therefore the table is referred as Interrupt Vector Table. In this table, IP value is put in as low word of thevector & CS is put in high vector.



8086 Interrupts

We are aware of the fact that the interrupt can be either hardware or software. If the interrupts are generated by the inbuilt devices, like timers or by the interfaced devices, they are called as hardware interrupts. If the interrupts are generated by the software code, they are called as software interrupts.

In other words an 8086 interrupt can come from any one of three sources.

- An external signal applied to the non-maskable interrupt (NMI) input pin or to the interrupt input pin (HARDWARE INTERRUPT).
- Execution of the interrupt instruction (SOFTWARE INTERRUPT)
- Some error condition produced in the 8086 by the execution of an instruction.

Example:
If you attempt to divide an operand by zero, the 8086 will automatically interrupt the currently executing program. At the end of each instruction cycle, the 8086 checks to see if any interrupts have been requested. If an interrupt has been requested, the 8086 responds to the interrupt by stepping through the following series of major actions:

- It decrements the stack pointer by 2 and pushes the flag register on the stack.
- It disables the 8086 INTR interrupt input by clearing the interrupt flag in the flag register.
- It resets the trap flag in the flag register.
- It decrements the stack pointer by 2 and pushes the current code segment register contents on the stack.
- It decrements the stack pointer again by 2 and pushes the current instruction pointer contents on the stack.

Divide-By-Zero Interrupt-Type 0:

The 8086 will automatically do a type 0 interrupt if the result of a DIV operation or an IDIV operation is too large to fit in the destination register. For a type 0 interrupt, the 8086 pushes the flag register on the stack, resets IF and TF and pushes the return addresses on the stack.

Single Step Interrupt-Type 1:

The use of single step execution feature is found in some of the monitor & debugger programs. When we tell a system to single step, it will execute one instruction and stop. We can then examine the contents of registers and memory locations.

In other words, when in single step mode a system will stop after it executes each instruction and wait for further direction from user. The

8086 trap flag and type 1 interrupt response make it quite easy to implement a single step feature direction.

Non-maskable Interrupt-Type 2:

The 8086 will automatically do a type 2 interrupt response when it receives a low to high transition on its NMI pin. When it does a type 2 interrupt, the 8086 will push the flags on the stack, reset TF and IF, and push the CS value and the IP value for the next instruction on the stack. It will then get the CS value for the start of the type 2 interrupt service procedure from address 0000AH and the IP value for the start of the procedure from address 00008H.

Breakpoint Interrupt-Type 3:

The type 3 interrupt is produced by execution of the INT3 instruction. The main use of the type 3 interrupt is to implement a breakpoint function in a system. Whenever we insert a breakpoint, the system executes the instructions up to the breakpoint and then goes to the breakpoint procedure.

Overflow Interrupt-Type4:

The 8086 overflow flag will be set if the signed result of an arithmetic operation on two signed numbers is too large to be represented in the destination register or memory location.

Example: If we add the 8 bit signed number 01101100 and the 8 bit signed number 010111101, the result will be 10111101. This would be the correct result if we were adding unsigned binary numbers, but it is not the correct signed result.

Software Interrupts-Type O through 255:

The 8086 INT instruction can be used to trigger the 8086 to do any one of the 256 possible interrupt types. The desired interrupt type is specified as part of the instruction.

The instruction INT32, for example will cause the 8086 to do a type 32 interrupt response. The 8086 will push the flag register on the stack, reset TF and IF, and push the CS and IP values of the next instruction on the stack.

INTR Interrupts-Types 0 through 255:

The 8086 INTR input allows some external signal to interrupt execution of

a program. Unlike the NMI input, however, INTR can be masked so that it cannot cause an interrupt. If the interrupt flag is cleared, then the INTR input is disabled. IF can be cleared at any time with CLEAR instruction.

Mnemonic	Meaning	Format	Operation	Flags Affected
CLI	Clear interrupt flag	CLI	0 → (IF)	IF
STI	Set interrupt flag	STI	1 → (IF)	IF
INT n	Type n software interrupt	INT n	(Flags) → ((SP) − 2) 0 → TF,IF (CS) → ((SP) − 4) (2 + 4 · n) → (CS) (IP) → ((SP) − 6) (4 · n) → (IP)	TF, IF
IRET	Interrupt return	IRET	((SP)) → (IP) ((SP) + 2) → (CS) ((SP) + 4) → (Flags) (SP) + 6 → (SP)	All
INTO	Interrupt on overflow	INTO	INT 4 steps	TF, IF
HLT	Halt	HLT	Wait for an external interrupt or reset to occur	None
WAIT	Wait	WAIT	Wait for $\overline{\text{TEST}}$ input to go active	None

Figure: 8086 Interrupt Instructions.

Interrupt Priority

If two or more interrupts occur at the same time then the highest priority interrupt will be serviced first, and then the next highest priority interrupt will be serviced.

Example: If suppose that the INTR input is enabled, the 8086 receives an INTR signal during the execution of a divide instruction, and the divide operation produces a divide by zero interrupt. Since the internal interrupts-such as divide error, INT, and INTO have higher priority than INTR the 8086 will do a divide error interrupt response first.

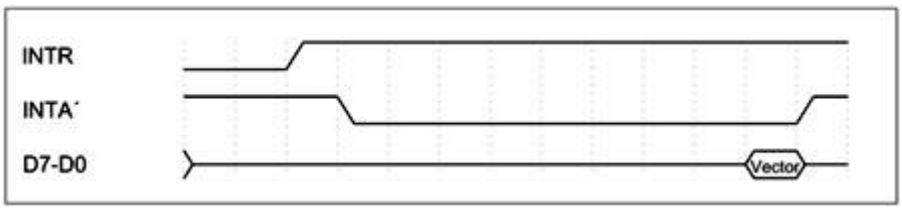
The interrupt that has a lower address, has a higher priority.

For example, the address of external interrupt 0 is 2, while the address of external interrupt 2 is 6; thus, external interrupt 0 has a higher priority, and if both of these interrupts are activated at the same time, extern al interrupt 0 is served first.

8086 Interrupt Pins and Timing

3. INTR: Interrupt Request. Activated by a peripheral device to interrupt the processor.
    1. Level triggered. Activated with a logic 1.

4. INTA: Interrupt Acknowledge. Activated by the processor to inform the interrupting device the interrupt request (INTR) is accepted.
    1. Level triggered. Activated with a logic 0.
5. NMI: Non-Maskable Interrupt. Used for major system faults such as parity errors and power failures.
    1. Edge triggered. Activated with a positive edge (0 to 1) transition.
    2. Must remain at logic 1, until it is accepted by the processor.
    3. Before the 0 to 1 transition, NMI must be at logic 0 for at least 2 clock cycles.
    4. No need for interrupt acknowledgement.



ADDRESSING MODE

- The way in which the operand is addressed then it is called addressing mode.
- The required operand may be placed in a register & accumulator.
- 8086 Microprocessor has 8 addressing mode:-
    a) register
    b) Immidiate
    c) Direct
    d) Indirect register
    e) Based
    f) Indexed
    g) Based indexed
    h) Based indexed with displacement.

a) REGISTER

- In these addressing mode the operand are available in the form of general purpose register.

Ex:-

(1) MOV AX, BX

(2) MOV AL, BL

b) IMMIDIATE

- In immidiate addressing mode the operand is represented in the form of 8bit data or 16-bit data

Ex:-

MVI AL, 20H

LXI BX, 7000H

c) DIRECT ADDRESSING MODE

- In these addressing mode the operand offset given in the instruction of 8bit & 16-bit displacement.

Ex:-

MOV AX, [6000H]

ADD AL, [7000H]

- The Instruction add the content of offset i.e. 7000H to the AL content & result stored in the AL.

d) REGISTER INDIRECT

- In these addressing mode the operand is placed in one of the register i.e. BX / BP / SI / DI as specified in the Instruction.

Ex:- MOV AX, [BX]

- The Instruction move the content of Bx to the Ax

Ex:- ADD AL, [SI]

- The Instruction ADD SI content with AL content & result stored in the AL.

e) BASED

- In these addressing mode the operand offset in the sum of 8-bit or 16bit displacement & the content of based register · BX/BP

Ex:-

MOV AL, [BX + 05]

- there Bx register content memory location i.e. 6000.

Bx:- 6000H & offset is 6005H after the content of 6005H is transfered to the AL.

f) INDEX ADDRESSING MODE

- In these addressing mode the operand offset in the sum of 8-bit or 16-bit displacement & index register i.e. SI & DI.

g) BASED INDEX

- In these addressing mode the operand offset in the sum of the content of based register & index register.

Ex:- MOV AX [BX + SI]
MOV AX [BP + DI]

h) BASED INDEXED WITH DISPLACEMENT

- In these addressing mode the operand offset is the sum of the index register & based register & 8bit data or 16 bit data.

Ex:- MOV AX, [Bx + SI + 05H] 8bit
MOV AX, [BP + DI + 1150H] 16bit

INSTRUCTION SETS OF 8086 MICROPROCESSOR

In 8086 microprocessor instruction sets are divided into 8 Groups.

(1) Data Transfer Group
(2) Arithmatic Group
(3) Logical Group & Bit Manipulation Group
(4) String Group
(5) Branch & Loop Group
(6) Process Control Group
(7) Iteration Control Group
(8) Interrupt Group.

(1) DATA TRANSFER GROUP

- These Instructions are use to transfer the data from source operand to destination operand.

Example:-

MOV, MVI etc.

(2) ARITHMATIC GROUP

- These Instructions core use to perform Arithmatic-Operation such as:- ~~ADD, SUB, DIV, MULT etc.~~

Example:- Addition, Subtraction, Multiplication, Division.

ADD, SUB, MUL, DIV etc.

(3) LOGICAL GROUP

- These Instructions are use to perform logical operation and shift operation.

Example:- AND, OR, EX-OR, ANI, ORI, Etc. on ROL, ROR

(4) STRING GROUP

- String is a group of byte or word and in string data are allocated in sequential order.

Example :-

REP, IN, OUT etc.

(5) BRANCH & LOOP GROUP

- These instruction are use to transfer the instruction during an execution without any condition.

Example :-

CALL, JMP, RET etc.

(6) PROCESS CONTROL GROUP

- These instruction are use to microprocessor action by set or reset flag.

Example :-

STC, CLC etc.

(7) ITERATION CONTROL GROUP

- These instructions are use to execute the given instruction for number of time.

Example :-

LOOP

(8) INTERRUPT GROUP

- These instruction are use to call the interrupt during an execution.

Example :- INT. etc.

TO FIND LARGER OF TWO NUMBER

Memory Address	Machine code	Labels	Mnemonics	Operands	Comments
2000	21, 01, 25		LXI	H, 2501H	Address of 1st no in H-L pair
2003	7E		MOV	A, M	1st no in Accumulator
2004	23		INX	H	Address of 2nd number in H-L pair
2005	BE		CMP	M	Compare 2nd number with 1st no is the 2nd no > 1st?
2006	D2, 0A, 20		JNC	AHEAD	No, larger no is in accumulator go to Ahead
2009	7E		MOV	A, M	Yes, get 2nd number in accumulator
200A	32, 03, 25, AHEAD		STA	2503H	Store large no in 2503H
200D	76		HLT		STOP

FIND OUT SMALLEST NUMBER IN A DATA ARRAY

Memory Address	Machine codes	Labels	Mnemonics	Operands	Comments
2000	21,00,25		LXI	H,2500H	Get address for count in H-L pair
2003	4E		MOV	C,M	Count in register
2004	23		INX	H	Get address of 1st number in H-L pair
2005	7E		MOV	A,M	1st number in Accumulator
2006	0D		DCR	C	Decrement count
2007	23	Loop	INX	H	Address of next number in H-L pair
2008	BE		CMP	M	Compare next number with previous smaller
2009	DA,0D,20		JC	AHEAD	Previous smallest less than next number? Yes, smallest number in accumulator goto ahead.
200C	7E		MOV	A,M	No, get number in accumulator
200D	0D	AHEAD	DCR	C	decrement count
200E	C2,07,20		JNZ	Loop	
2011	32,50,24		STA	2400H	Store smallest number in 2450H
2014	76		HLT		STOP

Difference Between Minimum Mode & Maximum Mode:-

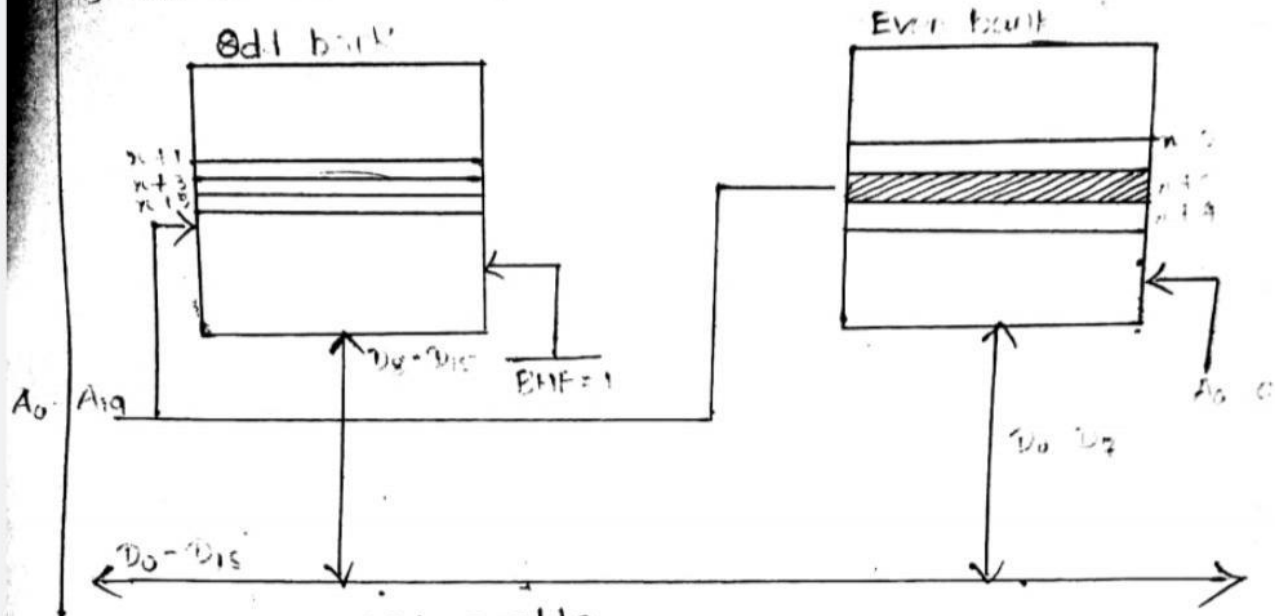
Minimum Mode	Maximum Mode
→ The minimum mode ckt is simple.	→ The maximum mode ckt is complex.
→ If MN/$\overline{MX}$ = 1, then ckt consider to as a minimum mode.	→ If MN/$\overline{MX}$ = 0, then ckt consider to as a maximum mode.
→ In this mode there can be only one processer.	→ In this mode there can be multiple processor.
→ Multiprocessing can't be perform hence performance is lower.	→ Multiprocessing can be perform hence performance is higher.
→ Control signal $\overline{M}$/IO are given by 8086.	→ Instead of control signal status signals are used like that S_0, S_1 & S_2.
→ INTA is given by 8086 in responds to an interrupt all interrupt line.	→ Instead of INTA 'a' status signals are used.

Instruction format - Memory Addressing 8086:-
Data can be access from the memory in 4 different ways.

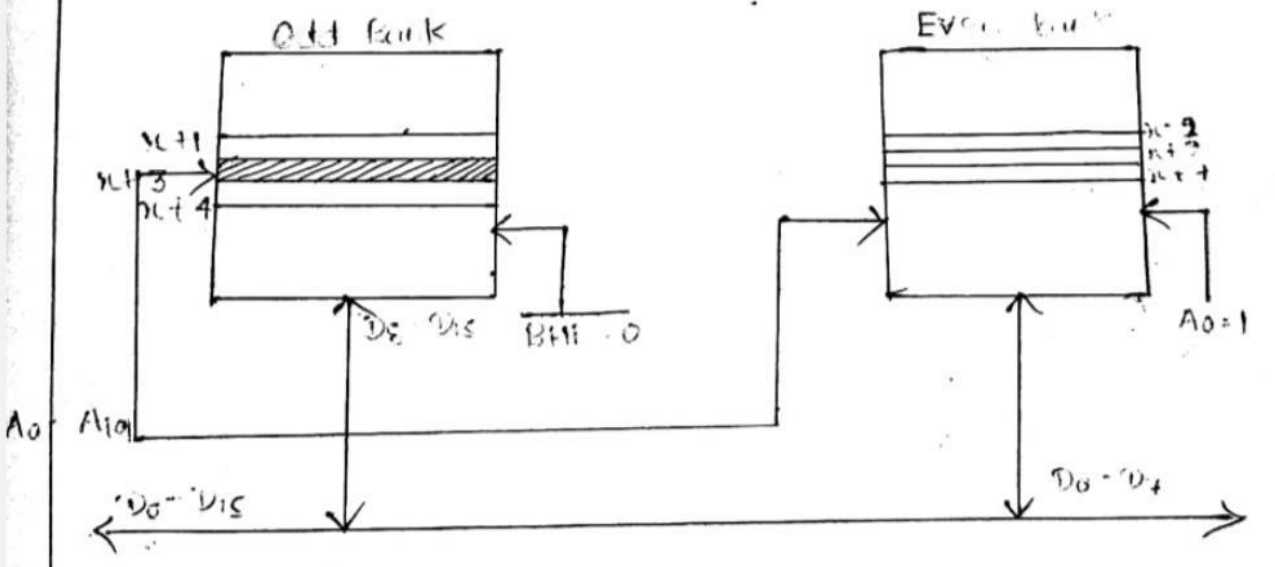
(i) 8 bit data from lower {Even} address Bank
(ii) 8 bit data from higher {ODD} address Bank
(iii) 16 bit data starting from {Even} address Bank
(iv) 16 bit data starting from {ODD} address Bank

Instruction format 8086:-

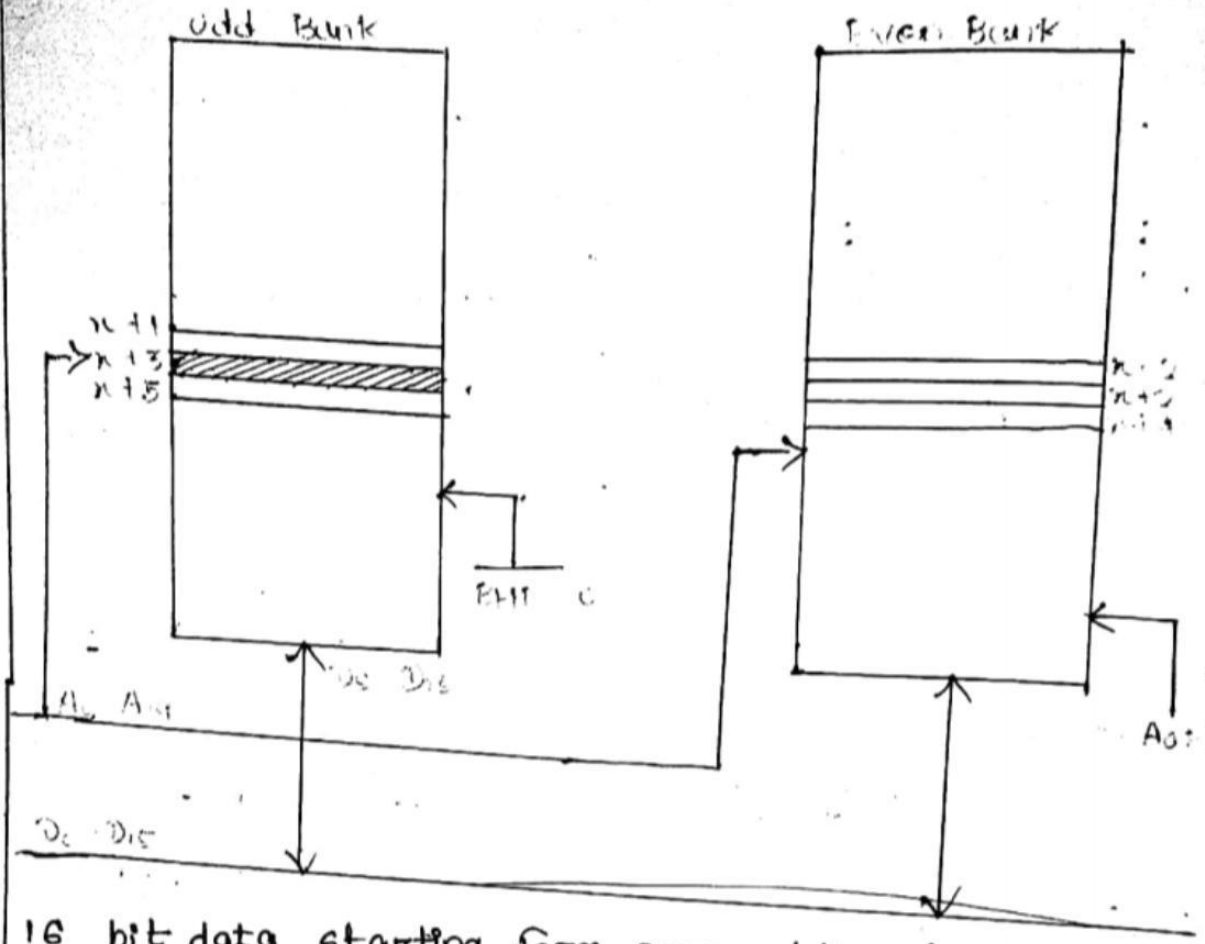
8 bit data from lower {even} address Bank:-



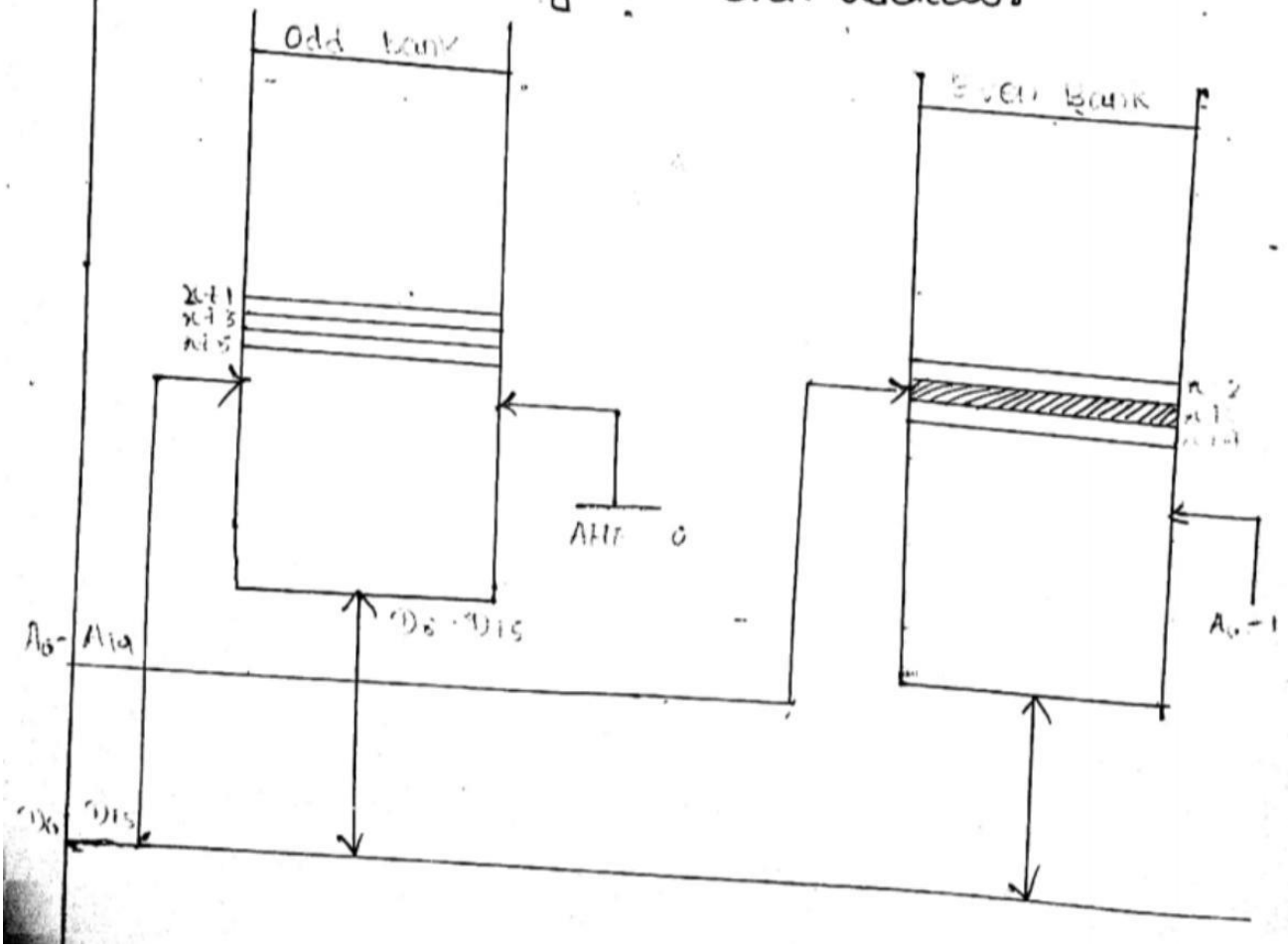
BHE:- Bus high enable



16 bit data starting form odd address:-



16 bit data starting form even address:-



UNIT-6 MICROCONTROLLER

(Architecture and Programming - 8 bit)

6.1 Distinguish between Microprocessor & Microcontroller

Microprocessor	Microcontroller
⇒ A microprocessor is a general purpose device which is called a CPU.	⇒ A microcontroller is a dedicated chip which is also called single chip computer.
⇒ A microprocessor do not contain on chip I/O ports, Timers, Memories etc.	⇒ A microcontroller includes RAM, ROM serial and parallel interface, timers, interrupt circuitary in a single chip.
⇒ Microprocessor are most commonly used as the CPU in microcomputer systems	⇒ Microcontroller are used in small, minimum component designs performing control oriented application.
⇒ Microprocessor instructions are mainly nibble or byte addressable	⇒ Microcontroller instructions are both bit addressable as well as byte addressable
⇒ Microprocessor instruction sets are mainly intended for catering to large volumes of data	⇒ Microcontroller have instructions sets catering to the control of inputs and outputs.
⇒ Microprocessor based system design is complex and expensive	⇒ Microcontroller based system design is rather simple & cost effective.
⇒ The instruction set of microprocessor is complex with large number of instruction.	⇒ The instruction set of n. Microcontroller is very simple with less number of instructions.

⇒ A microprocesson has zero status flag.	⇒ A microcontroller has no zero flag.

6.2 8-bit and 16-bit Microcontroller

8-bit Microcontroller

⇒ These are the most popular and ~~which~~ widely used microcontrollers.

⇒ About 55% of all CPUs sold in the world are 8-bit microcontrollers only.

⇒ The 8-bit microcontroller has 8-bit internal bus and the ALU performs all the arithmetic and logical operation on a byte instruction.

⇒ The well known 8-bit microcontroller is 8051 which was designed by Intel in the year 1980 for the use in Embedded systems.

⇒ Other 8-bit microcontroller are Intel 8031/8052 and Motorola MC68HC11 and AVR Microcontroller Microchip's PIC Microcontrollers 12C5XX, 16C5X and 16C505 etc.

16-bit Microcontroller

⇒ When the microcontroller performs 16-bit arithmetic and logical operations at an instruction, the microcontroller is said to be a 16-bit microcontroller.

⇒ The internal bus width of 16-bit microcontroller is of 16-bit.

⇒ These ~~are most~~ microcontroller are having increased memory size and speed of operation when compared to 8-bit microcontrollers.

⇒ These are most suitable for High Level Language like

C or C++.

⇒ They find application in disk drivers, modems, printers, scanners and servomotor control.

⇒ Examples of 16-bit microcontroller are Intel 8096 family and Motorola MC68HC12

6.3 CISC & RISC processor

RISC Processor

⇒ It is known as Reduc Instruction Set Computer.

⇒ It is a type of microprocessor that has a limited number of instructions.

⇒ They can execute their instructions very fast because instructions are very small and simple.

⇒ RISC chips require fewer transistors which make them cheaper to design and produce.

⇒ In RISC, the instruction set contains simple and basic instruction from which more complex instruction can be produced.

⇒ Most instructions complete in one cycle, which allows the processor to handle many instructions at same time.

⇒ In the instructions are register based and data transfer takes place from register to register.

CISC Processor

⇒ It is known as Complex Instruction Set Compiler.

⇒ It was developed by Intel

⇒ It contains large number of complex instructions

⇒ In this instructions are not registered based

⇒ Instructions cannot be completed in one machine cycle.

⇒ Data transfer is from memory to memory.

⇒ Micro programmed control unit is found in CISC.

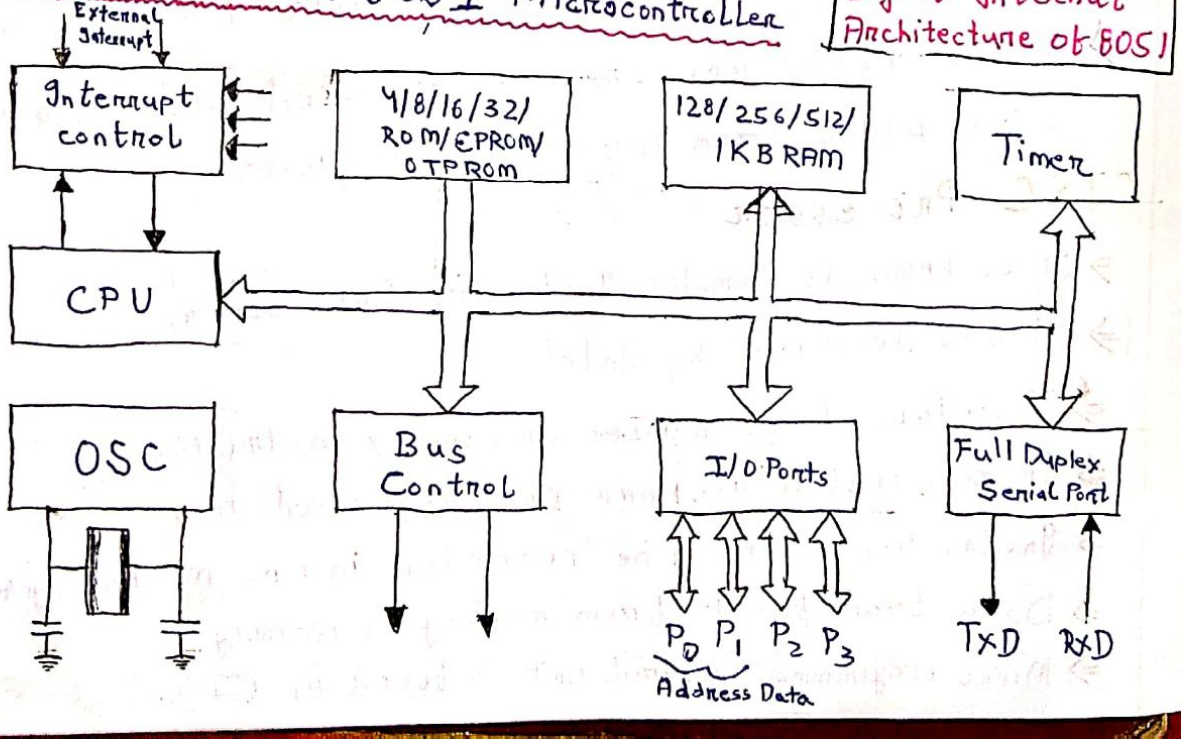
⇒ Also they have variable instruction formats.

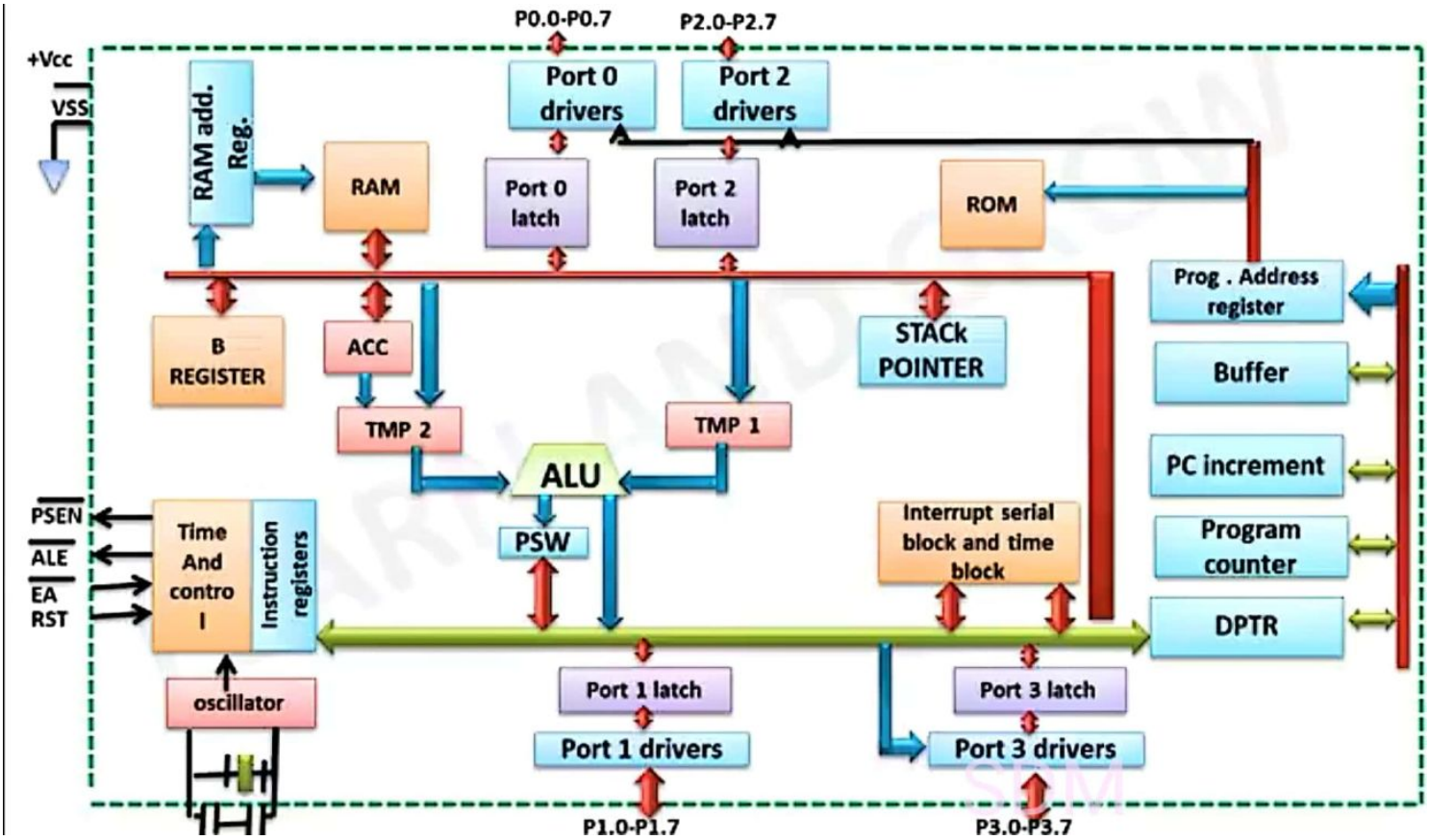
Difference Between:

	CISC	RISC
Inst'n size and format	⇒ Large set of instruction with variable formats. (16-64 bit)	⇒ Small set of instructions with fixed format (32 bit)
Data transfer	⇒ Data is transfer from memory to memory	⇒ Data is transfer from register to register.
CPU control	⇒ Most micro coded using control memory but modern CISC use hardware control	⇒ Most hardware without control memory.
Memory & Instruction type access	⇒ Not register based instructions More memory access	⇒ Register based instructions ⇒ Less memory access
Instruction type Clocks	⇒ Includes multi-clock	⇒ Includes single-clock
Addressing Mode	⇒ Instructions are complex ⇒ Many addressing mode	⇒ Instructions are reduced and simple ⇒ Few addressing mode.

6.4 Architecture of 8051 Microcontroller

Dig. of Internal Architecture of 8051





① Arithmetic and Logic Unit (ALU)

⇒ It performs arthimetic operations like add, sub, multi, divide.

⇒ It also performs logical operations like Logical OR, X-OR, AND, NOT, etc.

⇒ ALU can also manipulate one bit as well as 8 bit data types.

⇒ Individual bits can be set, cleared, complimented, tested and used in logical computation.

② Accumulator (A register)

⇒ It is a 8-bit register

⇒ It holds data and receives the result of arthimetic instruction

③ B register

⇒ It is a 8-bit general purpose register.

④ PSW (Program Status Word)

⇒ Many instruction implicity or explicity affect or are affected by several status flags which are together as PSW

⑤ Flag

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
CY	AC	F0	RS1	RS0	OV	-	P

Bit-7 : Carry flag

Bit-6 : Auxilary Carry flag

Bit-5 : User defined flag

Bit-4 & 3 : Select the working register ports

Bit-2 : Overflow Flag

Bit-1 : Reserved

Bit-0 : Parity flag

RS1	RS0	Port Selection
0	0	Port 0
0	1	Port 1
1	0	Port 2
1	1	Port 3

⑥ Stack Pointer

⇒ It is incremented before data is stored using PUSH and CALL instructions.

⇒ After reset the value of stack pointer is 07_H.

⑦ Data Pointer

⇒ It has two parts : → DPH
→ DPL

⇒ It is a 16-bit register used to hold the 16-bit address of data memory.

⇒ 8-bit pointers are used for accessing internal RAM and SFR

⇒ 16-bit data pointer is used for accessing external memory.

⇒ The contents of data pointer are programmable using instructions

⑧ Program Counter (PC)

⇒ It is a 16-bit register

⇒ PC is used as address pointer to access program instructions and it is automatically incremented after every byte of instruction fetch.

⑨ I/O Ports

⇒ There are four 8-bit I/O ports.

⇒ Each port has a Latch and a driver (buffer)

⇒ Port 0 will act as multiplexed low-order address bus and data bus (AD_0-AD_7)

⇒ Port 2 will act as high order address bus (A_8-A_{15})

⇒ Port 1 is dedicated I/O port does not have any alternate function

⇒ Port 3 dedicated I/O port and also it have some alternative functions

(10) Instruction Register (IR) and Timing and Control Unit

⇒ Microcontroller fetches the instruction one by one, starting from the address stored in PC and store it in IR which decodes the instruction and give information to timing and control unit.

⇒ Using this information, timing and control unit generate control signals necessary for internal and external operations.

(11) Interrupts Control

⇒ Interrupt is a subroutine call that interrupts of the microcontrollers main operations or work and causes it to execute any other program, which is more important at the time of operation.

⇒ The feature of interrupt is very useful as it helps in case of emergency operations.

⇒ Generally five interrupt source are there in 8051 microcontroller are:

- → INT0
- → TF0
- → INT1
- → TF1
- → R1/T1

(12) Oscillator

⇒ The microcontroller is a device, therefore it requires clock pulses for its operation of microcontroller applications.

⇒ For this purpose, microcontroller 8051 has an on chip oscillator which works as a clock source for CPU of the microcontroller.

⇒ The output pulses of oscillator are stable

⇒ Therefore, it enables synchronized work of all parts of the 8051 microcontroller.

(13) Timer/Counter and Serial Port

⇒ 8051 has two 16-bit programmable timers/counters

Timer-1 Timer-0

⇒ In the counter mode - they can count the number of high to low transitions of the signal applied to the timer pins.

⇒ In the timer mode - they can be independently programmed to work in any one of four operation mode.

→ Mode 0 → Serial port can either receive or transmit at fixed baud rate.

→ Mode 1 and Mode 3 → Timers can work as full duplex serial port with variable baud rate which is programmed using timer-1

→ Mode 2 → Simultaneously transmit or receive at any one of the two selectable baud rate.

(14) Memory

→ Mainly 8051 microcontroller has two memory

EPROM: It is used for permanent storage of data and program

RAM: Temporary storage of data and stack.

→ Microcontroller can only read data from program memory and the signal PSEN is used as read control for reading program memory

→ Microcontroller can both read and write with data memory RAM.

⇒ Separate $\overline{RD}$ and $\overline{WR}$ control signals are present.

⇒ Only memory mapped I/O interfacing is possible.

⇒ Program Memory → 64 KB

16-bit

0000H to FFFFH

64 KB — 4KB memory internal — 0FFF 0000 — PSEN is high

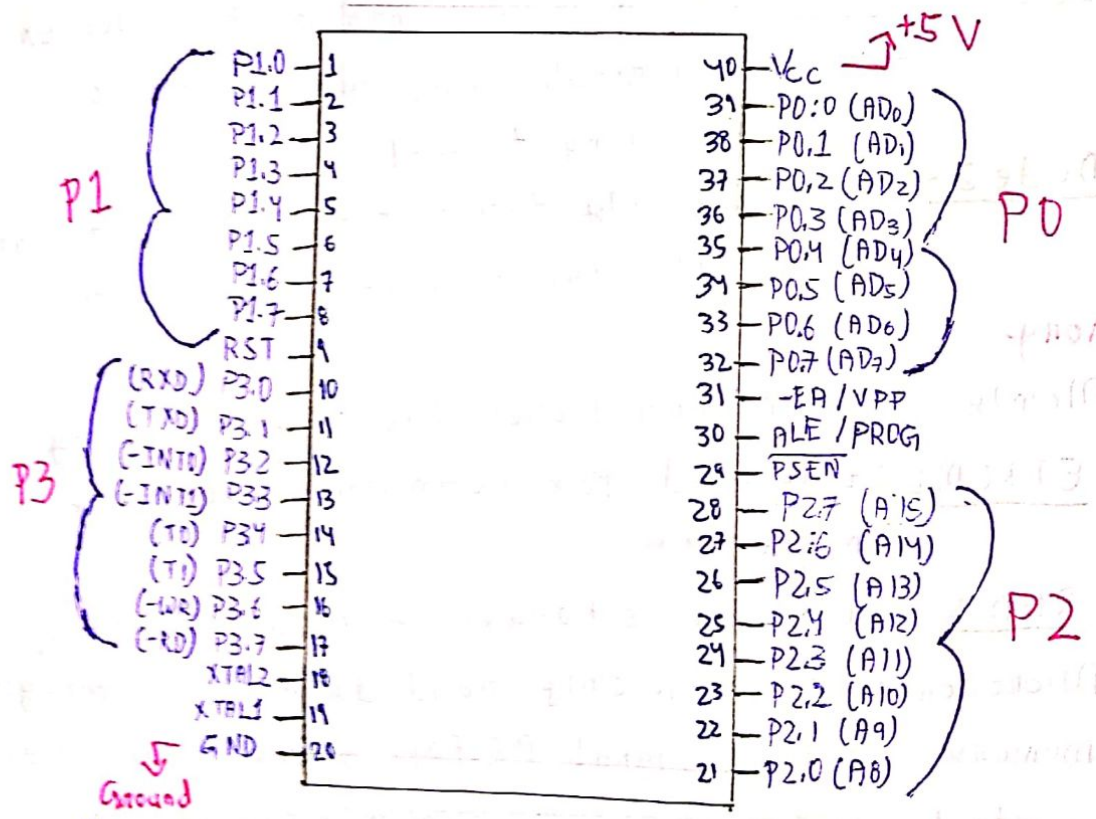
60KB memory external — 1000 FFFF

PSEN is Low — 64 KB Memory External

0000 to FFFF

⇒ Lower part of program memory stores the vector address for various interrupt service routines.

6.5 Signal Description of 8051 microcontroller



In 8051 microcontroller, a total of 32 pins are set aside for the four ports P0, P1, P2, P3, where each port takes 8 pins.

⇒ 8051 is a 8-bit microcontroller which has 40 pin IC, DIP (Dual Inline Package)

⇒ The signal from this pin can be categorized in six groups such as:

i) Port 0
ii) Port 1
iii) Port 2
iv) Port 3
v) Power supply & Clock signal
vi) Timing and control signal

Port 0

⇒ It is connected from 32 to 39 pins

⇒ If external memory is not used, these pins can be used as general inputs/outputs,

⇒ Otherwise, P_0 is configured as address output (A_0-A_7) when the ALE pin is driven high (1) or as data output (Databus). When the ALE pin is driven Low (0),

Port 1

⇒ It is connected from 1 to 8 pins

⇒ Each of these pin can be configured as an input or an output

Port 2

⇒ It is connected from 21 to 28 pins

⇒ If there is no intention to use external memory then these port pins are configured as general input/outputs.

⇒ In case external memory is used, the higher address byte, i.e. addresses A_8-A_{15} will appear on this port.

⇒ Even though memory with capacity of 64 KB is not used, which means that not all eight port bits are used for its addressing the rest of them are not available as input/outputs.

Port 3

⇒ It is connected from 10 to 17 pins.

⇒ Each of these pins can serve as general input or output. Besides, all of them have alternate functions.

⇒ The alternate function are as follows:

→ RXD
- It is the serial asynchronous communication input or serial synchronous communication output
- It is connected to pin 10 as P3.0

→ TXD
- It is the serial asynchronous communication output or serial synchronous communication input
- It is connected to pin 11 as P3.1

→ $\overline{INT0}$
- It is external interrupt 0 input
- It is connected to pin 12 as P3.2

→ $\overline{INT1}$
- It is the external interrupt 1 input
- It is connected to pin 13 as P3.3

→ T_0, T_1
- It is the timer 0 and timer 1 respectively
- It is at pin no. 14 as P3.4 and pin no. 15

as P3.5 respectively.

→ $\overline{WR}$
- It is used to write to external RAM
- It is at pin no. 16 as P3.6

→ $\overline{RD}$
- It is used to read from external RAM
- It is at pin no. 17 as P3.7

Power Supply & Clock Signals

Power Supply

→ There are two power supply i.e. V_{cc} and V_{ss} (GND)

→ V_{cc} indicates +5V power supply and its connected to pin no. 40

→ GND is the ground signal and its connected to pin no. 20

Clock Signals

→ There are two clock signals :
- XTAL2
- XTAL1

→ These are internal oscillator input and output

→ A quartz crystal which specifies operating frequency is usually connected to these pins.

Timing and Control Signals

RESET

→ A Logic one on this pin disables the microcontroller and clears the contents of most registers.

→ In other words, the positive voltage on this pin resets the microcontroller.

→ By applying logic zero to this pin, the program starts execution from the beginning.

PSEN

→ If external ROM is used for storing program than a logic zero (0) appears on it every time the microcontroller reads a byte from memory.

ALE

→ Prior to reading from external memory, the microcontroller puts the lower address byte ($A_0 - A_7$) on P_0 and activates the ALE output.

→ After receving signal from the ALE pin, the external latch lathes the state of P_0 and uses it as a memory chip address

→ Immediately after that, the ALE pin is returned its previous logic state and P_0 is now used as a Data Bus.

EA

→ By applying logic zero to this pin, P_2 and P_3 are used for data and address transmission with no regard to whether there is internal memory or not

→ It means that even there is a program written to the microcontroller, it will ~~be~~ not be executed.

→ Instead, the program written to external ROM will be executed.

→ By applying logic one to the EA pin, the microcontroller will use both memories, first internal then external (if exist)

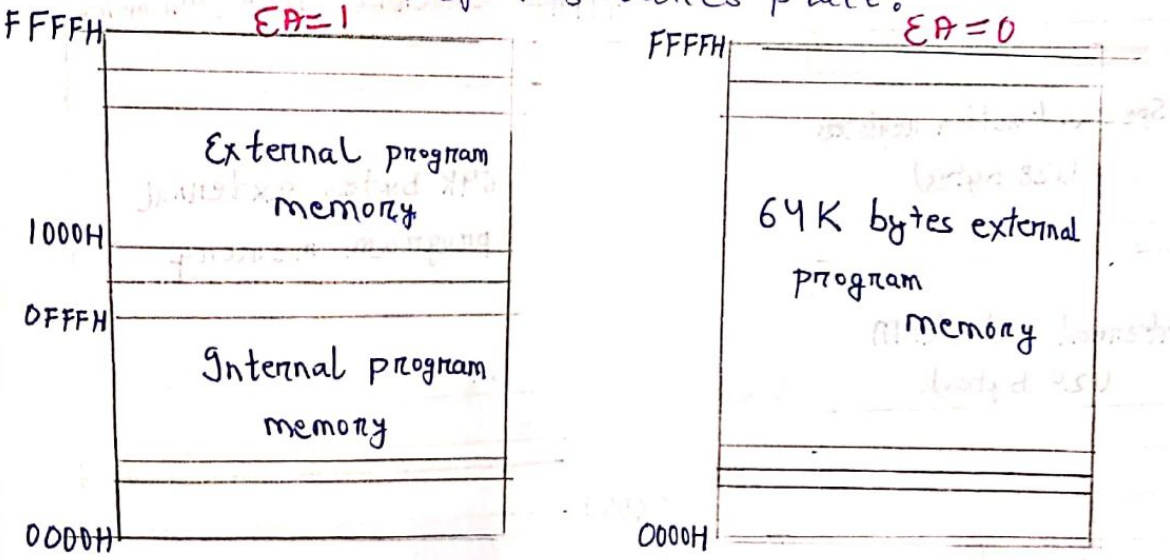
6.6 Memory Organisation - RAM structure, SFR

Memory Organisation of RAM

The applications of microcontroller are more. Hence knowledge about interfacing devices to the microcontroller and expanding of microcontroller memory is very important. Memories like SRAM, EPROM, EEPROM are interfaced to microcontroller to enchance capabilities of 8051.

1. Program Memory

Program memory accessed through EA pin. In program memory two categories takes place:



a) If EA is high, internal program memory is accessed to 0FFFH memory location and external program memory accessed from 1000H to FFFFH memories locations

b) If EA is low, only external program memory accessed from 0000H to FFFFH memory locations.

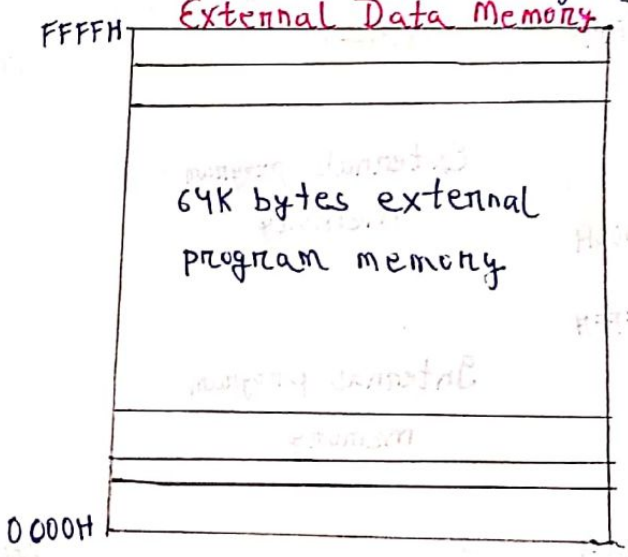
2. Data Memory

Data memory is used to store the memory in the registers each of 64K bytes size, to access the data memory instruction MOV X is used. Data Memory is of two types Internal and External.

i) Internal data memory:

The internal data memory consists of 256 bytes, these are divided in to two parts:

- 00H-FFH for internal data RAM (128 bytes)
- 80H-FFH for special function registers (128 bytes)



ii) External data memory

The 8051 gives the facility to interface external RAM and ROM, External RAM is accessed by DPTR and up to 64KB of RAM can be interfaced. External data memory interfacing is of two types i.e. RAM and ROM interfacing

1) RAM interfacing

The interfacing of memory chip with microcontroller has some regulations to follow;

a) The memory data bus is directly connected to memory chip data pins

b) Control signal connection

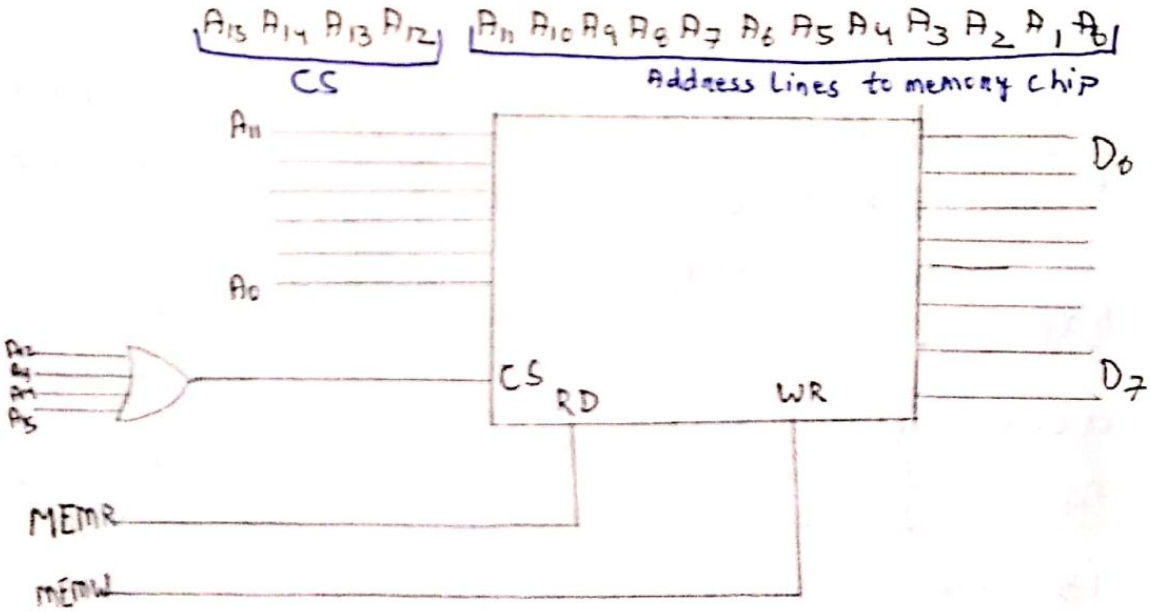
- RD (Read Memory) connected to OE (Output Enable)
- WR (Write Memory) connected to WE (Write Enable)

c) The CPU address lines are directly connected to memory chip addressing lines.

☆ The memory chip consist of Chipset (CS) and Chip enable (CE) address lines varies based on memory capacity chip should inbuilt with control signals, data lines. The accessing of memory is done when chip is activated.

Example: Interfacing 4KB RAM

Let address chip is 3000H to 3FFF



ii) ROM interfacing

In many systems the on chip ROM of 8051 is not sufficient, so 8031 chip is used, it is a ROM less version of 8051 which allows program size to be large as 64K bytes

EA Pin: To indicate the program code stored in microcontroller on chip ROM, EA pin is connected to Vcc, to indicate program code is stored in ROM EA pin is connected to ground.

SFR (Special Function Register)

The 8051 microcontroller special function registers act as a control table that monitor and control the operation of the 8051 microcontroller. ~~get~~ In, Internal RAM structure, the address space from 80H to FFH is allocated to SFRs.

Out of these 128 memory locations (80H to FFH), there are only 21 locations that are actually assigned to SFRs. Each SFR has one byte address and also a unique ~~no~~ name which specifies its purpose.

Since the SFRs are a part of the Internal RAM structure, we can access SFRs as if you access the Internal RAM. The main difference is the address space: first 128 bytes (00H to 7FH) is for regular Internal RAM and next 128 Bytes (80H to FFH) is for SFRs

Elements of Special Function Register

→ All the 21 8051 microcontroller special function registers (SFRs) along with their functions and internal RAM address is given in the following table.

Name of the Register	Function	Internal RAM Address (HEX)
ACC	Accumulator	E0H
B	B register (for Arithmetic)	F0H
DPH	Addressing External Memory	83H
DPL	Addressing External Memory	82H
IE	Interrupt Enable Control	A8H
IP	Interrupt priority	B8H
P0	Port 0 Latch	80H
P1	Port 1 Latch	90H
P2	Port 2 latch	A0H
P3	Port 3 Latch	B0H
PCON	Power Control	87H
PSW	Program Status Word	D0H
SCON	Serial Port Control	98H
SBUF	Serial Port Data Bubber	99H
SP	Stack Pointer	81H
TMOD	Timer / Counter Mode Control	89H
TCON	Timer / Counter Control	88H
TL0	Timer 0 LOW Byte	8AH
TH0	Timer 0 HIGH Byte	8CH
TL1	Timer 1 LOW Byte	8BH
TH1	Timer 1 HIGH Byte	8DH

There are many ways to categories these 21 SFR but the easiest way is in which they are categorized in seven groups are:

- → Math or CPU registers (A and B)
- → Status Register (PSW)
- → Pointer Registers (DPTR) - DPL, DPH and SP)
- → I/O Port Latches (P_0, P_1, P_2, P_3)
- → Peripherial Control Registers (PCON, SCON, TCON, TMOD, IE, & IP)
- → Peripherial Data Registers (TL0, TH0, TL1, TH1 and SBUF)

CPU or Math Registers

A or Accumulator (ACC)

⇒ The accumulator or register A is the most important and most used 8051 microcontroller SFRs.

⇒ The Register A is located at the address E0H in the SFR memory space

⇒ The accumulator is used to hold the data for almost all the ALU operations.

⇒ Some of the operations where the Accumulator is used are:

- o Arithmetic Operation like Addition, Subtraction, Multiplication etc.
- o Logical Operations like AND, OR, NOT etc.
- o Data transfer Operations (between 8051 and External memory)

⇒ The name "Accumulator" came from the fact this register is used to accumulate the result of all arithmetic and most of Logical operations

B (Register B)

⇒ The B register is used along with the ACC in Multiplication and Division operations.

⇒ These two operations are performed on data that are stored only in registers A and B.

⇒ It can be used as a General purpose register for normal operations and is often used as Auxilary register by programmers to store temporary results

PSW

⇒ The PSW is also called as Flag Register and is one of the important SFRs.

⇒ The PSW register consist of flag bits, which help the programmer in checking the condition of the result and also make decision.

⇒ Flag are 1-bit storage elements that storae and indicate the nature of the result that is generated by execution of certain instructions.

⇒ The following table describes the function of each flag.

BIT	SYMBOL	FLAG NAME	DESCRIPTION
7	Con CY	Canny	Used in anithmetic, logic and boolean operations
6	AC	Auxiliary Canny	Used in BCD Anithmetic
5	F0	Flag 0	General purpose User flag.
4 3	RS1 RS0	Register Bank Selection Bit 1 Register Bank Selection Bit 1 RS1 RS0 Bank 0 0 Bank0 0 1 Bank1 1 0 Bank2 1 1 Bank3	
2	OV	Overflow	Used in anithmetic operation
1	-	Reserved	May be used as General purpose flag
0	P	Parity	Set to 1 A has odd # of 1's, otherwise Reset.

Pointer Registers

Data Pointer (DPTR - DPL and DPH)

→ The Data Pointer is a 16 bit Register & is physically the combination of DPL (Data Pointer Low) & DPH (Data Pointer High) SFRs.

→ The Data Pointer can be used as a single 16 bit register (as DPTR) or two 8-bit registers (as DPL & DPH)

⇒ DPTR doesn't have a physical Memory Address but the DPL (Lower Byte of DPTR) and DPH (Higher Byte of DPTR) have separate addresses in the SFR Memory Space.

⇒ DPL = 82H and DPH = 83H

⇒ The DPTR Register is used by the programmer addressing external memory (Program - ROM or Data - RAM)

Stack Pointer (SP)

⇒ SP points out to the top of the stack and it indicates the next data to be accessed.

⇒ Stack Pointer can be accesses using PUSH, POP, CALL and RET instructions.

⇒ The Stack Pointer is an 8-bit register and upon reset, the Stack Pointer is initialized with 07H.

⇒ When writing a new data is written at an address SP + 1.

⇒ When reading data from stack, the data is retrieved from the Address in SP and after that the SP. is decremented by 1 (SP-1)

I/O Port Registers (P0, P1, P2 and P3)

⇒ The 8051 Microcontroller four Ports which can be used as Input and/or Output.

⇒ These four ports are P0, P1, P2 and P3.

⇒ Each Port has a corresponding register with same names (the Port Registers are also P0, P1, P2 and P3).

⇒ The addresses of the Port Registers are as follows:
P0 - 80H, P1 - 90H, P2 - A0H and P2 - B0H

⇒ Each bit in these SFRs corresponds to one physical Pin in the 8051 Microcontroller.

⇒ All these Port Registers are both Bit Addressable and Byte Addressable and

⇒ Writing 1 or 0 on a Port Register Bit will reflect as an appropriate voltage (5V and 0V) on the corresponding Pin.

⇒ If a Port Bit is SET (declared as 1), the corresponding Port Pin will be configured as Output.

⇒ Upon reset, all the ~~Port Pin is configured as Output~~. Port Bits are SET (1) and hence, all the Port Pins are configured as Inputs.

Peripheral Control Registers

PCON (Power Control)

⇒ The PCON, as the name suggests is used to control the 8051 Microcontroller's Power Modes and is located at 87H of the SFR Memory Space.

⇒ Using two bits in the PCON Register, the microcontroller can be set to Idle Mode and Power Down Mode.

⇒ During Idle Mode, the Microcontroller will stop the Clock Signal to the ALU (CPU) but it is given to other peripherals like Timer, Serial, Interrupts, etc.

⇒ In order to terminate the Idle Mode, you have to use an Interrupt or Hardware Reset.

⇒ In the Power Down Mode, the oscillator will be stopped and the power will be reduced to 2V.

⇒ To terminate the Power Down Mode, you have to use the Hardware Reset.

⇒ Apart from these two, the PCON Register can also be used for Few additional purposes.

⇒ The SMOD Bit in the PCON Register is used to control the Baud Rate of the Serial Port.

⇒ There are two General purpose Flag Bits in the PCON Register, which can be used by the programmer during execution.

SCON (Serial Control)

⇒ The SCON SFR is used to control the 8051 Microcontroller's Serial Port. It is

⇒ It is located as an address of 98H.

⇒ Using SCON, you can control the Operation Modes of the Serial Port, Baud Rate of the Serial Port and Send or Receive Data using Serial Port.

⇒ SCON Register also consists of bits that are automatically SET when a byte of data is transmitted or received.

Serial Port Mode Control Bits

SM0	SM1	Mode	Description	Baud Rate
0	0	0	8-Bit Synchronous Shift Register Mode	Fixed Baud Rate (Frequency of oscillator /12)
0	1	1	8-Bit Standard UART Mode	Variable Baud Rate (Can be set by Timer 1)
1	0	2	9-bit Multiprocessor Comm. mode	Fixed Baud Rate (Frequency of oscillator/ 32 or Frequency of oscillator/ 64)
1	1	3	9-bit Multiprocessor Comm. mode	Variable Baud Rate (Can be set by Timer 1)

TCON (Timer Control)

→ Timer Control is used to start or stop the Timers of 8051 Microcontroller.

⇒ It also contains bits to indicate if the Timers has overflowed.

→ The TCON SFR also consists of interrupt related bits.

TMOD (Timer Mode)

⇒ The TMOD is used to set the Operating Modes of the Timers T0 and T1.

⇒ The lower four bits are used to configure Timer 0 and the higher four bits are used to configure Timer 1.

IE (Interrupt Enable)

⇒ The IE Register is used to enable or disable individual interrupts.

⇒ If a bit is SET, the corresponding interrupt is enabled and if the bit is cleared, the interrupt is disabled.

→ The Bit7 of the IE register i.e. EA bit is used to enable or disable all the interrupts.

IP (Interrupt - Priority)

⇒ The IP Register is used to set the priority of the interrupt as High or Low.

⇒ IF a bit is CLEARED, the corresponding interrupt is assigned low priority and if the bit is SET, the interrupt is assigned high priority.

Peripheral Data Registers

SBUF (Serial Data Buffer)

⇒ The Serial Buffer register is used to hold the serial data while transmission or reception.

TL0/TH0 (Timer 0 Low/High)

⇒ The Timer 0 consists of two SFRs: TL0 and TH0.

⇒ The TL0 is the lower byte and the TH0 is the higher byte and together they form a 16-bit Timer 0 Register.

TL1/TH1 (Timer 1 Low/High)

⇒ The TL1 and TH1 are the lowerc and higher bytes of the Timer 0.

6.8 Addressing Modes of 8051

8051 addressing modes are classified as follows

1. Immediate addressing
2. Register addressing
3. Direct Addressing
4. Indirect Addressing
5. Relative Addressing
6. Absolute Addressing
7. Long Addressing
8. Indexed Addressing
9. Bit Inherent Addressing
10. Bit Direct Addressing.

Immediate Addressing

⇒ In this addressing mode the data is provided as a part of instruction itself. In other words data immediately follows the instruction

⇒ E.g. MOV, A #30H

ADD, A #83. #Symbol indicates the data is immediate

Register Addressing

⇒ In this addressing mode the register will hold the data. One of the eight general registers (R0 to R7) can be used and specified as the operand.

⇒ E.g. MOV A, R0

ADD A, R6

Direct Addressing

⇒ There are two ways to access the internal memory, Using direct address and indirect address. Using direct addressing mode we can not only address the internal memory but SFRs also.

In direct addressing, an 8 bit internal data memory address is specified as part of the instruction and hence, it can specify the address only in the range of 00H and FF H. In this addressing mode, data is obtained directly from memory.

Eg: MOV A, 60 H

ADD A, 30H

Indirect Addressing

The indirect addressing mode uses a register to hold the actual address that will be used in data movements

Registers R0 and R1 and DPTR are the only registers that can be used as data pointer. Indirect addressing cannot be used to refer to SFR registers. Both R0 and R1 can hold 8 bit address and DPTR can hold 16-bit address.

E.g. MOV A, @R0
ADD A, @R1
MOV X A, @DPTR

Indexed Addressing

In indexed addressing, either PC or DPTR is used to hold the base address, and the A is used to hold the offset address. Adding the value of the base address to the value of the offset address forms the effective address. Indexed addressing is used with JMP or MOVC instructions

E.g. MOVC A, @A+DPTR
MOVC A, @A+PC

Relative Addressing

Relative addressing is used only with conditional jump instructions. The relative address, is an 8-bit signed number, which is automatically added to the PC to make the address of the next instruction,

E.g. SJMP LOOP1
JC BACK

Absolute Addressing

Absolute addressing is used only by the absolute jump and absolute call instructions.

These are 2 bytes instructions.

→The absolute addressing mode specifies the lowest 11 bit of the memory address as part of instruction.

→The upper 5 bit of the destination address are the upper 5 bit of the current program counter.

E.g. AJMP LOOP1
ACALL LOOP2

Long Addressing

→The long addressing mode is used with the instruction LJMP and LCALL.#

→ These are 3 byte instructions.

E.g. LJMP FINISH
LCALL DELAY

Bit Inherent Addressing

In this addressing, the address of the flag which contains the operand, is implied in the opcode of the instruction.

E.g. CLR C;

Bit Direct Addressing

In this addressing mode the direct address of the bit is specified in the instruction. The RAM space 20H to 2FH and most of the special function registers are bit addressable.

E.g. CLR 07h ;
SETB 07h;

6.10 Interrupts, Timer & Counters

Interrupts in 8051 Microcontroller

During program execution if peripherial devices needs service from microcontroller, device will generate interrupt and gets the service from microcontroller.

When peripherial device activate the interrupt signal, the processor branches to a program called interrupt service routine. After executing the interrupt service routine the processor returns to the main program.

Steps taken by processor while processing an interrupt

1. It completes the execution of the current instruction
2. PSW is pushed to stack
3. PC content is pushed to stack
4. Interrupt flag is reset
5. PC is loaded with ISR address.

ISR will always ends with RETI instruction. The execution of RETI instruction results in the following.

1. POP the current stack top to the PC
2. POP the current stack top to the PSW.

Classification of interrupt

1. External and internal interrupts

⇒ External interrupt are those intiated by peripherial devices through the external pins of the micro controller

⇒ Internal interrupts are those activated by the internal peripherials of the microcontroller.

2. Maskable and non-masakable interrupts

⇒ The category of interrupts which can be disabled by the processor using program is called masakable interrupts.

⇒ Non-masakable interrupts are those category by which the programmer cannot disable it using program.

3. Vectored and non-vectored interrupts

⇒ Starting address of the ISR is called interrupt vector. In vectored interrupts the starting address is predicated predefined.

⇒ In non-vectored interrupts, the starting address is provided by the peripherial as follows.

→ Microcontroller receives an interrupt request from external device

→ Controller sends an acknowledgement (INTA) after completing the execution of current instruction

→ The peripherial device sends the interrupt vector to the microcontroller.

Interrupt Structure

8051 have five interrupts. They are masakable and vectored interrupts that of these five, two are external interrupt and three are internal interrupts.

Interrupt Source	Type	Vector address	Priority
External interrupt 0	External	0003	Highest
Timer 0 interrupt	Internal	0008	
External interrupt 1	External	0013	
Timer 1 interrupt	Internal	001B	
Serial interrupt	Internal	0023	Lowest

8051 makes use of two registers to deal with interrupts

1. IE Register

This is an 8-bit register used for enabling or disabling the interrupts. The structure of IE register is shown below.

IE: Interrupt Enable Register (Bit Addressable)

If the bit is 0, the corresponding interrupt is disabled, if the bit is 1, the corresponding interrupt is enabled.

EA			ES	ET1	EX1	ET0	EX0	

EA	IE.7	Disables all interrupts if EA=0, no interrupt will be acknowledged. If EA=1, interrupt source
-	IE.6	Not implemented, reserved for future use
-	IE.5	Not implemented, reserved for future use
ES	IE.4	Enable or disable the serial port interrupt
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt
EX1	IE.2	Enable or disable External Interrupt 1,
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt
EX0	IE.0	Enable or disable External Interrupt0,

2. IP Register

This is an 8-bit register used for ~~s~~setting the priority of the interrupts.

IP: Interrupt Priority Register (Bit Addressable)

If the bit is 0, the corresponding interrupt has a lower priority and if the bit is the corresponding interrupt has a higher priority.

-	-	-	PS	PT1	PX1	PT0	PX0

-	IP.7	Not implemented, reserved for future
-	IP.6	Not implemented, reserved for future
-	IP.5	Not implemented, reserved for future use.
PS	IP.4	Defines the serial port interrupt priority level
PT1	IP.3	Defines the Timer 1 interrupt priority level
PX1	IP.2	Defines External 1 interrupt priority level
PT0	IP.1	Defines the timer 0 interrupt priority level
PX0	IP.0	Defines the external interrupt 0 priority level

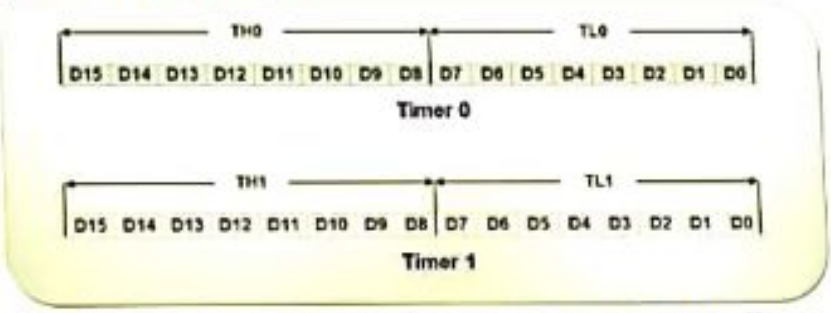
Timers and Counters

⇒ Timers / Counters are used generally for

- → Time reference
- → Creating delay
- → Wave form properties measurement
- → Periodic interrupt generations
- → Wave form generation

- Waveform generation

8051 has two timers, Timer 0 and Timer 1.



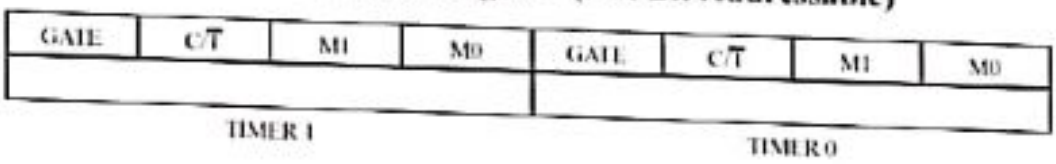
Timer in 8051 is used as timer, counter and baud rate generator. Timer always counts up irrespective of whether it is used as timer, counter, or baud rate generator: Timer is always incremented by the microcontroller. The time taken to count one digit up is based on master clock frequency.

If Master CLK=12 MHz,
Timer Clock frequency = Master CLK/12 = 1 MHz
Timer Clock Period = 1micro second
This indicates that one increment in count will take 1 micro second.

The two timers in 8051 share two SFRs (TMOD and TCON) which control the timers, and each timer also has two SFRs dedicated solely to itself (TH0/TL0 and TH1/TL1).

TMOD Register

TMOD : Timer/Counter Mode Control Register (Not Bit Addressable)



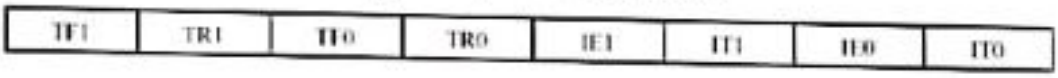
GATE	When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).
C/T	Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).
M1	Mode selector bit (NOTE 1).
M0	Mode selector bit (NOTE 1).

Note 1 :

M1	M0		OPERATING MODE
0	0	0	13-bit Timer
0	1	1	16-bit Timer/Counter
1	0	2	8-bit Auto-Reload Timer/Counter
1	1	3	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit Timer and is controlled by Timer 1 control bits.
1	1	3	(Timer 1) Timer/Counter 1 stopped

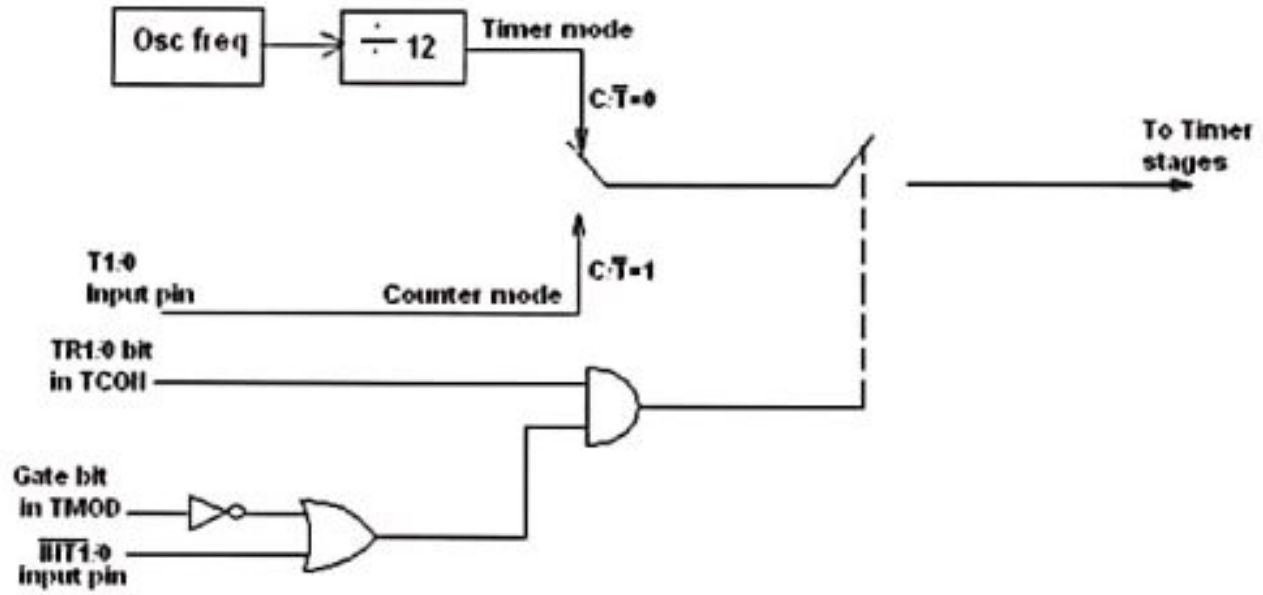
TCON Register

TCON : Timer/Counter Control Register (Bit Addressable)



TF1	TCON.7	Timer 1 overflow flag. Set by hardware when the Timer/Counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR1	TCON.6	Timer 1 run control bit. Set/cleared by software to turn Timer/Counter ON/OFF.
TF0	TCON.5	Timer 0 overflow flag. Set by hardware when the Timer/Counter 0 overflows. Cleared by hardware as processor vectors to the service routine.
TR0	TCON.4	Timer 0 run control bit. Set/cleared by software to turn Timer/Counter 0 ON/OFF.
IE1	TCON.3	External Interrupt 1 edge flag. Set by hardware when External interrupt edge is detected. Cleared by hardware when interrupt is processed.
IT1	TCON.2	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/flow level triggered External Interrupt.
IE0	TCON.1	External Interrupt 0 edge flag. Set by hardware when External Interrupt edge detected. Cleared by hardware when interrupt is processed.
IT0	TCON.0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

Timer/ Counter Control Logic.



TIMER MODES

Timers can operate in four different modes. They are as follows

Timer Mode-0: In this mode, the timer is used as a 13-bit UP counter as follows.

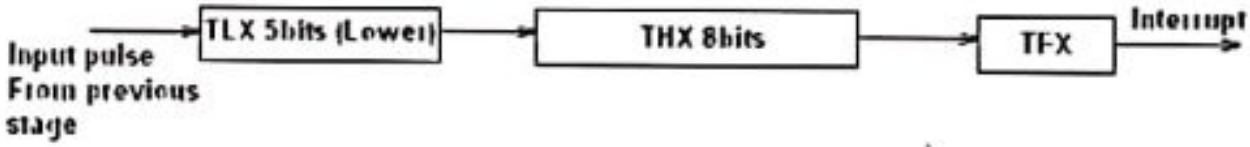


Fig. Operation of Timer on Mode-0

The lower 5 bits of TLX and 8 bits of THX are used for the 13 bit count.Upper 3 bits of TLX are ignored. When the counter rolls over from all 0's to all 1's, TFX flag is set and an interrupt is generated. The input pulse is obtained from the previous stage. If TR1/0 bit is 1 and Gate bit is 0, the counter continues counting up. If TR1/0 bit is 1 and Gate bit is 1, then the operation of the counter is controlled by input. This mode is useful to measure the width of a given pulse fed to input.

Timer Mode-1: This mode is similar to mode-0 except for the fact that the Timer operates in 16-bit mode.

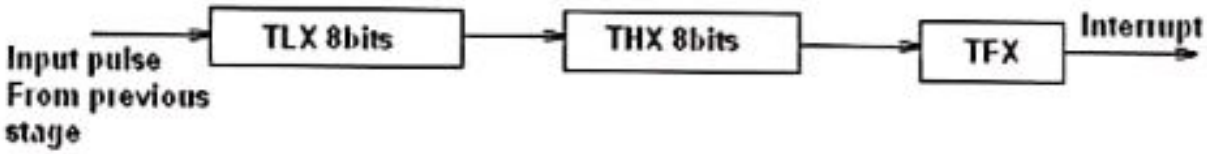


Fig: Operation of Timer in Mode 1

Timer Mode-2: (Auto-Reload Mode): This is a 8 bit counter/timer operation. Counting is performed in TLX while THX stores a constant value. In this mode when the timer overflows i.e. TLX becomes FFH, it is fed with the value stored in THX. For example if we load THX with 50H then the

timer in mode 2 will count from 50H to FFH. After that 50H is again reloaded. This mode is useful in applications like fixed time sampling.

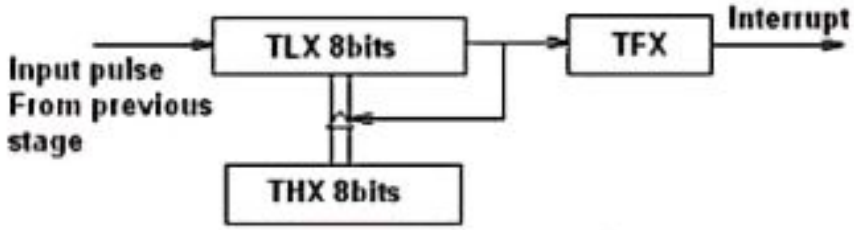


Fig: Operation of Timer in Mode 2

Timer Mode-3: Timer 1 in mode-3 simply holds its count. The effect is same as setting TR1=0. Timer0 in mode-3 establishes TL0 and TH0 as two separate counters.

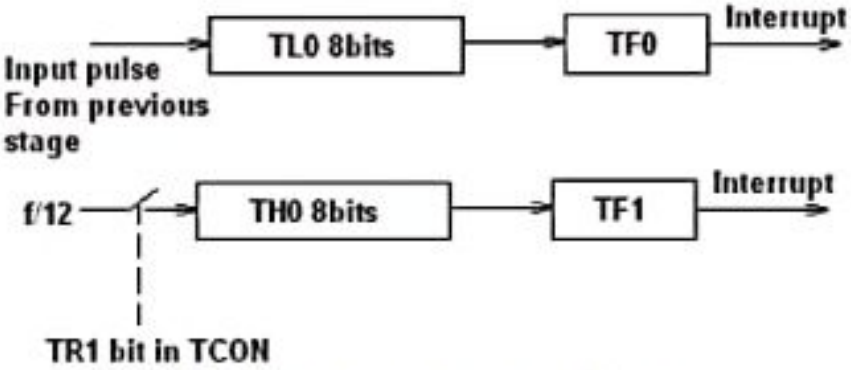


Fig: Operation of Timer in Mode 3

Control bits TR1 and TF1 are used by Timer-0 (higher 8 bits) (TH0) in Mode-3 while TR0 and TF0 are available to Timer-0 lower 8 bits(TL0).

6.1 SERIAL COMMUNICATION.

6.1.1. DATA COMMUNICATION

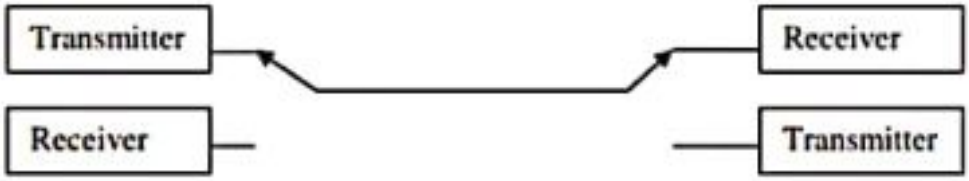
The 8051 microcontroller is parallel device that transfers eight bits of data simultaneously over eight data lines to parallel I/O devices. Parallel data transfer over a long is very expensive. Hence, a serial communication is widely used in long distance communication. In serial data communication, 8-bit data is converted to serial bits using a parallel in serial out shift register and then it is transmitted over a single data line. The data byte is always transmitted with least significant bit first.

6.1.2. BASICS OF SERIAL DATA COMMUNICATION,

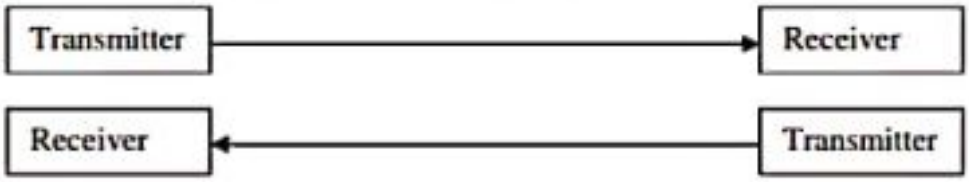
Communication Links

1. Simplex communication link: In simplex transmission, the line is dedicated for transmission. The transmitter sends and the receiver receives the data.

2. Half duplex communication link: In half duplex, the communication link can be used for either transmission or
reception. Data is transmitted in only one direction at a time.



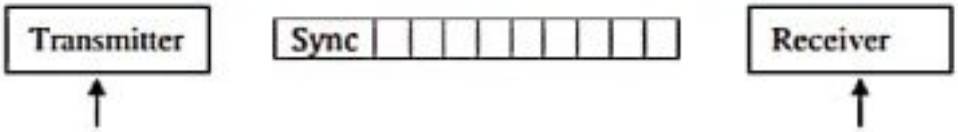
3. Full duplex communication link: If the data is transmitted in both ways at the same time, it is a full duplex i.e. transmission and reception can proceed simultaneously. This communication link requires two wires for data, one for transmission and one for reception.

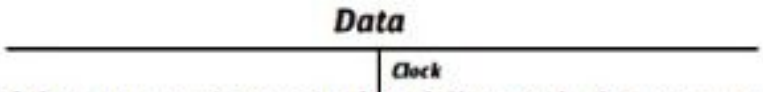


Types of Serial communication:

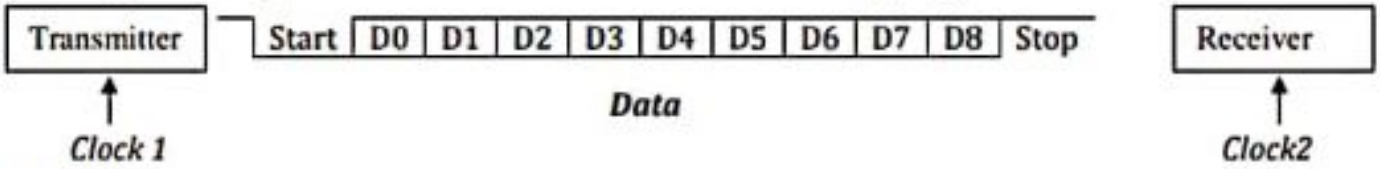
Serial data communication uses two types of communication.

1. Synchronous serial data communication: In this transmitter and receiver are synchronized. It uses a common clock to synchronize the receiver and the transmitter. First the synch character is sent and then the data is transmitted. This format is generally used for high speed transmission. In Synchronous serial data communication a block of data is transmitted at a time.





2. Asynchronous Serial data transmission: In this, different clock sources are used for transmitter and receiver. In this mode, data is transmitted with start and stop bits. A transmission begins with start bit, followed by data and then stop bit. For error checking purpose parity bit is included just prior to stop bit. In Asynchronous serial data communication a single byte is transmitted at a time.



Baud rate:

The rate at which the data is transmitted is called baud or transfer rate. The baud rate is the reciprocal of the time to send one bit. In asynchronous transmission, baud rate is not equal to number of bits per second. This is because; each byte is preceded by a start bit and followed by parity and stop bit. For example, in synchronous transmission, if data is transmitted with 9600 baud, it means that 9600 bits are transmitted in one second. For bit transmission time = 1 second/ 9600 = 0.104 ms.

6.1.3. 8051 SERIAL COMMUNICATION

The 8051 supports a full duplex serial port.

Three special function registers support serial communication.

1. SBUF Register: Serial Buffer (SBUF) register is an 8-bit register. It has separate SBUF registers for data transmission and for data reception. For a byte of data to be transferred via the TXD line, it must be placed in SBUF register. Similarly, SBUF holds the 8-bit data received by the RXD pin and read to accept the received data.
2. SCON register: The contents of the Serial Control (SCON) register are shown below. This register contains mode selection bits, serial port interrupt bit (TI and RI) and also the ninth data bit for transmission and reception (TB8 and RB8).

Serial Port Control (SCON) Register							
D7	D6	D5	D4	D3	D2	D1	D0
SM0	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**

- SM0 (SCON.7) : Serial communication mode selection bit
- SM1 (SCON.6) : Serial communication mode selection bit

SM0	SM1	Mode	Description	Baud rate
0	0	Mode 0	8-bit shift register mode	Fosc / 12
0	1	Mode 1	8-bit UART	Variable (set by timer 1)
1	0	Mode 2	9-bit UART	Fosc/ 32 or Fosc/64
1	1	Mode 3	9-bit UART	Variable (set by timer 1)

- SM2 (SCON.5): Multiprocessor communication bit. In modes 2 and 3, if set this will enable multiprocessor communication.
- REN (SCON.4) : Enable serial reception
- TB8 (SCON.3) : This is 9th bit that is transmitted in mode 2 & 3.
- RB8 (SCON.2) : 9th data bit is received in modes 2 & 3.
- TI (SCON.1) : Transmit interrupt flag, set by hardware must be cleared by software.
- RI (SCON.0) : Receive interrupt flag, set by hardware must be cleared by software.

3. PCON register: The SMOD bit (bit 7) of PCON register controls the baud rate in asynchronous mode transmission.

Power mode Control (PCON) Register							
D7	D6	D5	D4	D3	D2	D1	D0
SMOD	**—**	**—**	**—**	**GF1**	**GF0**	**PD**	**IDL**

- SMD (PCON.7): Serial rate modify bit. Set to 1 by program to double baud rate using timer 1 for modes 1, 2, and 3. cleared by program to use timer 1 baud rate.
- GF1 (PCON.3) : General Purpose user flag bit.
- GF0 (PCON.2) : General Purpose user flag bit.
- PD (PCON.1) : Power down bit. Set to 1 by program to enter power down configuration for CHMOS processors.
- IDL (PCON.0) : Idle mode bit. Set to 1 by program to enter idle mode configuration for CHMOS processors.

6.1.4. SERIAL COMMUNICATION MODES

1. ***Mode 0***

   In this mode serial port runs in synchronous mode. The data is transmitted and received through RXD pin and TXD is used for clock output. In this mode the baud rate is 1/12 of clock frequency.

2. ***Mode 1***

   In this mode SBUF becomes a 10 bit full duplex transceiver. The ten bits are 1 start bit, 8 data bit and 1 stop bit. The interrupt flag TI/RI will be set once transmission or reception is over. In this mode the baud rate is variable and is determined by the timer 1 overflow rate.

   Baud rate = $[2^{smod}/32]$ x Timer 1 overflow Rate

   = $[2^{smod}/32]$ x [Oscillator Clock Frequency] / [12 x [256 – [TH1]]]

3. ***Mode 2***

   This is similar to mode 1 except 11 bits are transmitted or received. The 11 bits are, 1 start bit, 8 data bit, a programmable 9th data bit, 1 stop bit.

   Baud rate = $[2^{smod}/64]$ x Oscillator Clock Frequency

4. ***Mode 3***

This is similar to mode 2 except baud rate is calculated as in mode 1

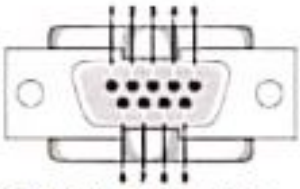
6.1.5. CONNECTIONS TO RS-232

RS-232 standards:

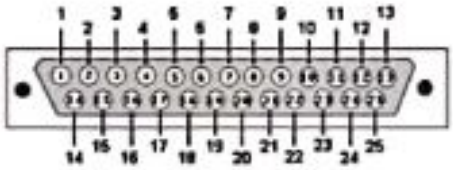
To allow compatibility among data communication equipment made by various manufactures, an interfacing standard called RS232 was set by the Electronics Industries Association (EIA) in 1960. Since the standard was set long before the advent of logic family, its input and output voltage levels are not TTL compatible.

In RS232, a logic one (1) is represented by -3 to -25V and referred as MARK while logic zero (0) is represented by +3 to +25V and referred as SPACE. For this reason to connect any RS232 to a microcontroller system we must use voltage converters such as MAX232 to convert the TTL logic level to RS232 voltage levels and vice-versa. MAX232 IC chips are commonly referred as line drivers.

In RS232 standard we use two types of connectors. DB9 connector or DB25 connector.



DB9 Male Connector



DB25 Male Connector

The pin description of DB9 and DB25 Connectors are as follows

DB-25 Pin No.	DB-9 Pin No.	Abbreviation	Full Name
Pin 2	Pin 3	TD	Transmit Data
Pin 3	Pin 2	RD	Receive Data
Pin 4	Pin 7	RTS	Request To Send
Pin 5	Pin 8	CTS	Clear To Send
Pin 6	Pin 6	DSR	Data Set Ready
Pin 7	Pin 5	SG	Signal Ground
Pin 8	Pin 1	CD	Carrier Detect
Pin 20	Pin 4	DTR	Data Terminal Ready
Pin 22	Pin 9	RI	Ring Indicator

The 8051 connection to MAX232 is as follows.

The 8051 has two pins that are used specifically for transferring and receiving data serially. These two pins are called TXD, RXD. Pin 11 of the 8051 (P3.1) assigned to TXD and pin 10 (P3.0) is designated as RXD. These pins TTL compatible; therefore they require line driver (MAX 232) to make them RS232 compatible. MAX 232 converts RS232 voltage levels to TTL voltage levels and vice versa. One advantage of the MAX232 is that it uses a +5V power source which is the same as the source voltage for the 8051. The typical connection diagram between MAX 232 and 8051 is shown below.