

Course Objectives:

1. The architecture and organization of a Microprocessor (8085/8086)
2. The basic operations, programming and application of Microprocessor
3. The interfacing I/O devices with the Microprocessor
4. Concept of the foundation for the microprocessor based system design

Chapter 1. Introduction to Microprocessors:

- Microprocessor is an electronic chip, that performs function as a central processing unit (CPU) of a computer.
- The word Microprocessor comes from the combination of micro and processor
- Where micro means small in size and processor is a device that performs process or manipulate numbers, specifically binary numbers (0's and 1's). According to the instruction (program) stored in the memory.
- Microprocessors contains both combinational logic and sequential digital logic

Std. Definition:

"Microprocessor is a multipurpose, programmable, clock driven, register based, digital integrated circuit that accepts binary data as input, processes it according to instructions stored in its memory and provides result as output."

Programmable :

Perform different set of operation on the data depending on the sequence of instructions supplied by the programmer.

Clock driven:

Microprocessor is a device which executes instructions in sequential order. In order to correctly execute the program instructions, microprocessor follows certain steps: Fetch, ~~code~~ decode, & execute. And in order to correctly follow those steps, microprocessor

needs to activate different components in different time. And how does microprocessor know when to do what? It uses clock.

Register Based : Storage based

Applications:

Mobile phone

TV remote control

Microwave oven

Washing Machine

Consumer electronic goods

Calculators

Industrial controllers

Computers

Traffic signal control systems

Speed controls of motors

Electronic & Electrical projects

: Its advance applications are Radar, satellites etc.

Advantage :

: processing speed is high

: Intelligence has been brought to systems

: Flexible

: Compact size

: Easy maintenance

: Complex mathematics

: Can be improved according to requirement

Disadvantage:

- overheating occurs due to oversize
- performance depends on size of data
- Large board size than microcontrollers
- Most microprocessor do not support floating point operations.

Computer's central processing unit (CPU) built on a single integrated circuit (IC) is called a microprocessor.

The microprocessor contains millions of tiny components like transistors, registers, and diodes that work together.

The MP is the "brain of the microcomputer".

Word Addressing:

Given M words (location to access), how many bits λ are required to address them?

$$\lambda = \log_2 M$$

Eg: To address 64MB, we used

$$\lambda = \log_2 (64 * 2^{10} * 2^{10}) = \log_2 (2^6 * 2^{10} * 2^{10}) = 26 \text{ bits}$$

Concept of Bin & Hex

Dec	Hex	Bin	8-bits
0000	0	0000	00H → 0000 0000
0001	1	0001	:
2	2H	0010	:
3	3H	0011	:
4	4H	0100	FFH → 1111 1111
5	5H	0101	Byte = 8-bits
6	6H	0110	
7	7H	0111	
8	8H	1000	16-bits
9	9H	1001	0000H → 0000 0000 0000
9Combination	AH	1010	
Trick 8421		1011	FFFF H → 1111 1111
---- 2 ³ 2 ² 2 ¹ 2 ⁰	FH	1111	Word = 16-bits
	16 Comb	4bits	Nibble = 4-bit

Power of 2

$$2^0 = 1$$

1KB = 1024 bytes

$$2^1 = 2$$

$$2^2 = 4$$

$$2^3 = 8$$

$$2^4 = 16$$

$$2^5 = 32$$

$$2^6 = 64$$

$$2^7 = 128$$

$$2^8 = 256$$

$$2^9 = 512$$

$$2^{10} = 1K$$

$$2^{11} = 2^1 \times 2^{10} = 2K$$

$$2^{12} = 2^2 \times 2^{10} = 4K$$

$$2^{13} =$$

:

:

$$2^{20} = 2^{10} \times 2^{10} = 1K \times 1K = 1M$$

$$2^{21} = 2 \times 2^{20} = 2M$$

$$2^{22} = 2^2 \times 2^{20} = 4M$$

:

$$2^{30} = 2^{10} \times 2^{20} = 1K \times 1M = 1G$$

$$2^{31} = 2^1 \times 2^{30} = 2G$$

:

:

$$2^{40} = 2^{10} \times 2^{30} = 1K \times 1G = 1T$$

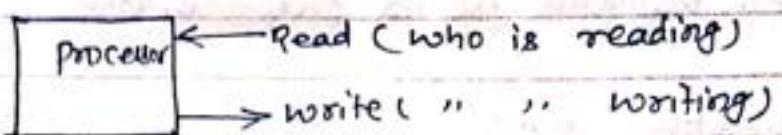
$$2^{41} = 2 \times 2^{40} = 2T$$

:

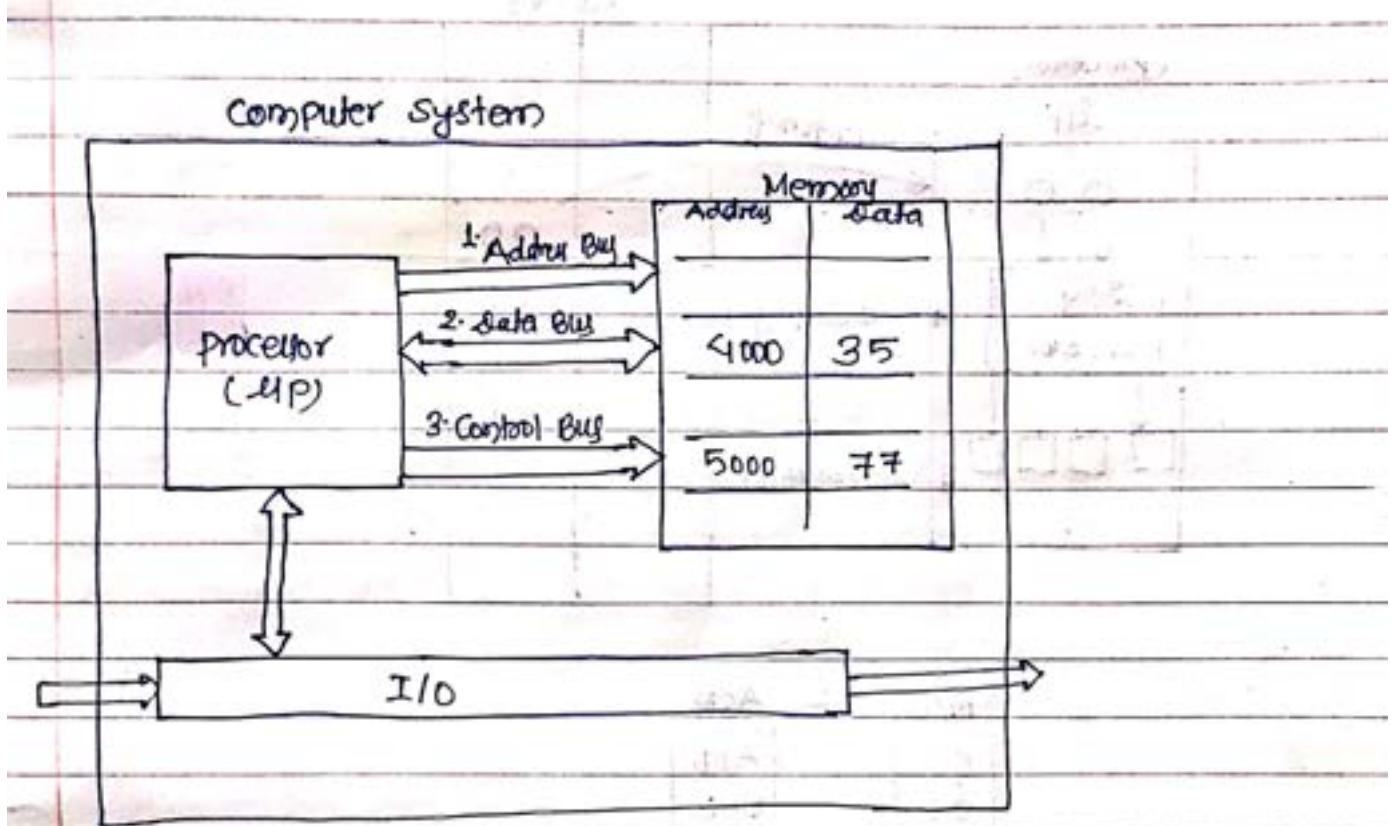
$$2^{50} = 2^{10} \times 2^{40} = 1K \times 1T = 1P$$

$$2^{51} = 2 \times 2^{50} = 2P$$

Read signal / write operation:



whenever processor gets the data read operation
whenever .. sends the .. write operation



$q = b + c;$ HLL
ADD B,C ; ASM
0110 1011 ; LLL/MC / Obj. / Bin
 ^(Machine code)
 "opcode"

Compiler

Assembler

Purpose of Compiler is converted to Machine language

Compiler program to converted to Machine language, if there is an error it can not be compile so it throws for error

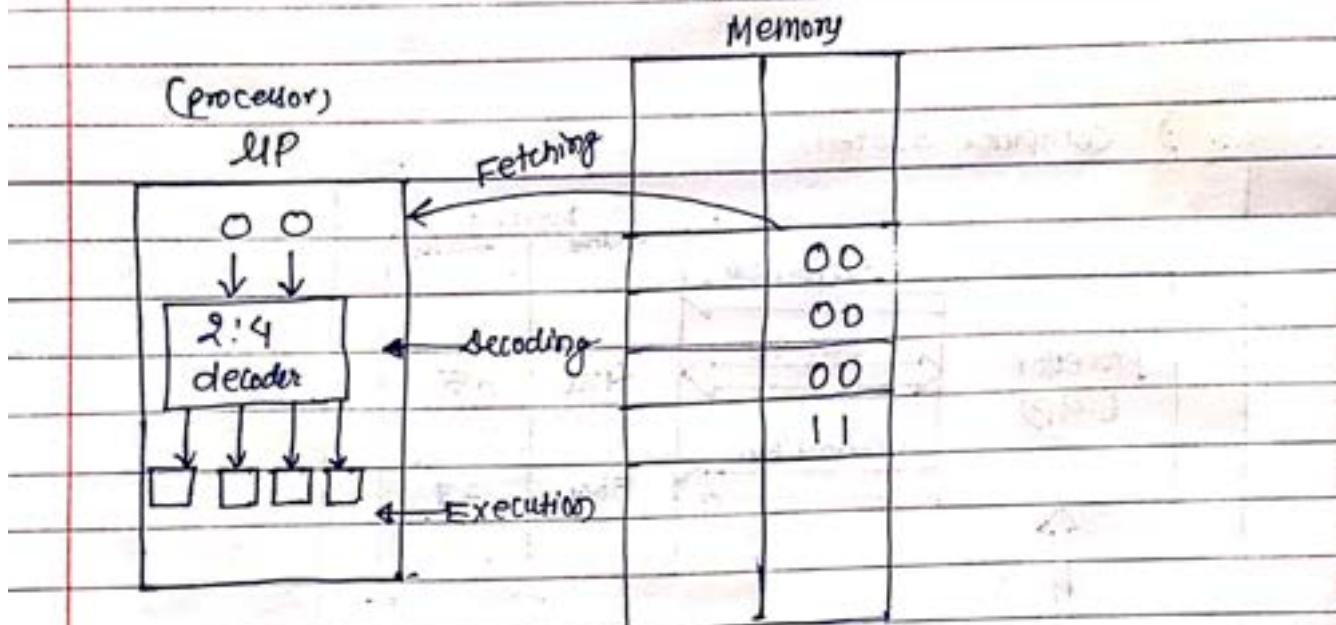
Every instruction has its own ^{unique} "opcode"

Instruction cycle:

Fetch

Decode: understanding of opcode

Execute



M/C	ASM
00	ADD
01	SUB
10	MUL
11	DIV

CH-1 Introduction to Microprocessor

1.1. Evolution of Microprocessor

- Transistor was invented in 1948 (23 dec, 1947 in Bell lab)
- IC was invented in 1958 (Fairchild semiconductors) by Texas Instruments J Kilby.

We can categorize the MP according to the generations or according to the size of the MP.

First Generation (4-bit)

- 4 bit MP (4004)
- The first microprocessor was introduced in 1971 by Intel corp.
- It was named Intel 4004 ^{Since} it was a 4 bit processor.
- It was a processor on a single chip.
- It could perform simple arithmetic and logical operations such as addition, subtraction, Boolean OR and Boolean AND.
- Clock speed = 740 KHz
- No. of transistors = 2.3K
- Instruction per second (IPS) = 60K
- It had a control unit capable of performing control functions like fetching an instruction from storage memory, decoding it, and then generating control pulses to execute it.
- Memory addressing capacity = 64KB 1 KB
- Pins = 16
- Address Bus = 10 bit, Data Bus = 4 bit

Second Generation (8-bit)

- The second generation microprocessor was introduced in 1972 again by Intel.
- It was a first 8-bit MP which could perform arithmetic and logic operations on 8-bit words.
- It was Intel 8008, and another improved version was Intel 8088/8085 in 1976

Some other 8-bit processors are Zilog 80 and Motorola M6800

Clock speed = 500KHz for 8008

2 MHz for 8080

3.5 MHz for 8085

No. of transistors = 6.5K for 8085

IPS = 769230 for 8085 (popular 8-bit MP)

Memory addressing capacity = 64KB for 8085

Pins = 40

for 8085, Address Bus = 16-bit, Data Bus = 8-bit

Third Generation (16-bit)

The third generation MP, introduced in 1978 were represented by Intel's 8086, Zilog Z800 and 80286, which were 16-bit processor with a performance like minicomputers.

Clock speed = 5-8 MHz (for 8086)

No. of transistors = 29K

IPS = 2.5 Million

Memory addressing capacity = 1 MB

Pins = 40

Widely used in PC/XT

Address Bus = 20-bit, Data Bus = 16-bit

Fourth Generation (32-bit)

Several different companies introduced the 32-bit MP, but the most popular one is the Intel 80386 in 1985

Clock speed = 16 MHz

No. of transistors = 275K

PGA = Pin grid Array
MMU = Memory Management Unit

Address Bus = 32-bit, Data Bus = 32-bit

- IPS = more than 5 Million
- Memory addressing capacity = 4GB real, 64TB virtual
- Pins = 132, or
14x14, PGA
- contains MMU on chip
- Pentium in 1993, CS = 66MHz, NOT = 3.1M, Pins = 237 PGA
for Pentium IPS = cache memory 8 bit for instructions 8-bit for data
- for pentium, Address Bus = 32-bit, Data Bus = 32/64-bit

Fifth Generation (64-bit)

- From 1995 till now we are in the fifth generation, 64-bit processors like PENTIUM, celeron, dual, quad and octa core processors came into existence.
- INTEL Core 2 in 2006, CS = 1.2-3GHz, NOT = 291 Millions
- i3, i5 in 2007, 2009, CS = 2.2GHz
- i7 in 2010 CS = 3.3GHz
- i9 in 2017 (Core i9 X-Series)

1.2 #

Von Neumann and Harvard architecture:

- : Von Neumann architecture was first published by John Von Neumann in 1945.
- : His computer architecture design consists of a Control unit, arithmetic and logic unit (ALU), Memory unit, Registers and Inputs/outputs.
- : Von Neumann architecture is based on the stored-program computer concept, where instruction data and program data are stored in the same memory.
- : This design is still used in most computers produced today.
- : Von Neumann architecture is required only one bus for instruction and data.
- : The processor need two clock cycle to complete an instruction. So pipelining is not possible with this architecture.
- : In the first clock cycle processor gets the instruction from memory and decodes it. In next clock cycle the required data is taken from memory. For each instruction this cycle repeat and hence need two cycle to complete an instruction.
- : Von Neumann architecture is relatively older and was replaced by Harvard Architecture.

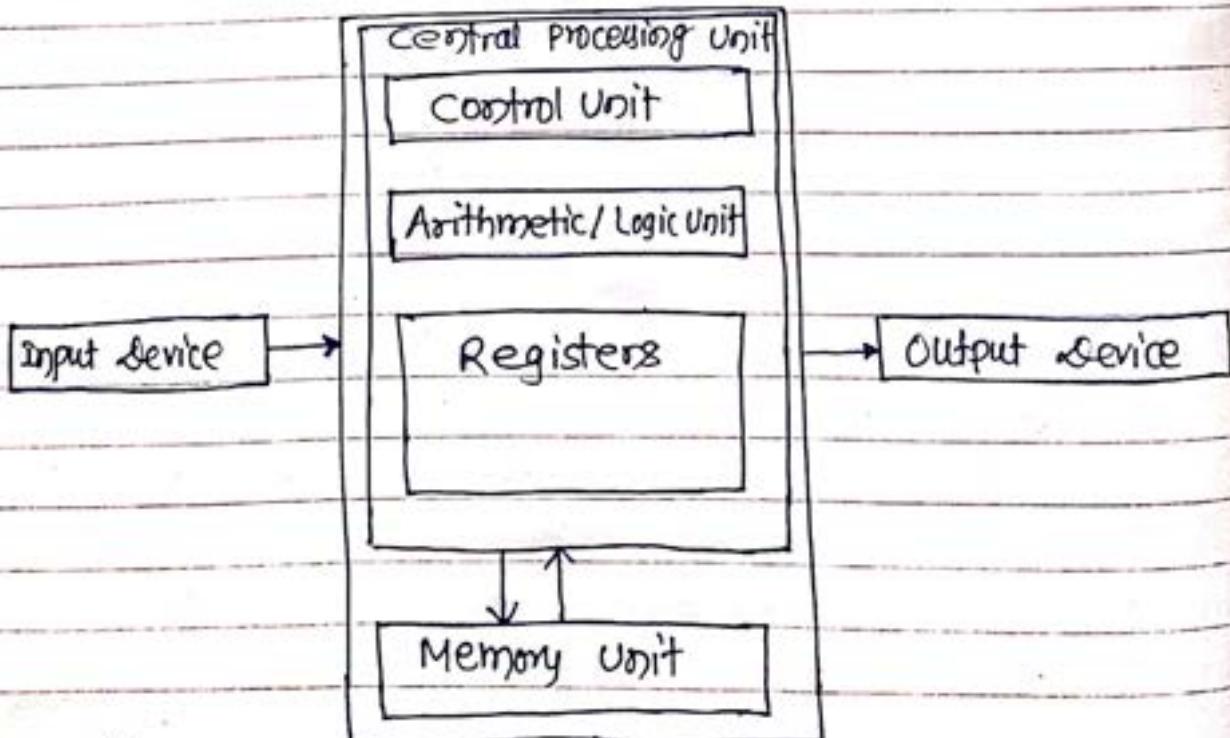


fig: Von Neumann Architecture.

central processing unit (CPU):

- The CPU is the electronic circuit responsible for executing the instructions of a computer program.
- It is sometimes referred to as the microprocessor or processor.
- The CPU contains the ALU, CU and a variety of registers.

control unit (CU):

- The control unit controls the operation of the computer's ALU, Memory and input/output devices, telling them how to respond to the program instructions it has just read and interpreted from the memory unit.
- The control unit also provides the timing and control signals required by the computer components.

Arithmetic and Logic Unit (ALU):

- The ALU allows arithmetic (add, subtract etc) and logic (AND, OR, NOT etc) operations to be carried out.

Registers:

- Registers are high speed storage areas in the CPU. All data must be stored in a register before it can be processed.
- The various models of registers are:

Memory Address Register (MAR): It stores the locations of instructions that need to be fetched from memory or stored into memory.

- Holds the memory location of data that needs to be accessed.

Memory Buffer Register (MBR): Holds data that is to be transferred to, and stored in, memory.

- Holds data that is being transformed to or from memory.

Instruction Register (IR): Contains the current instruction during processing.

program Counter (PC): contains the address of the next instruction to be executed.

Accumulator (AC): stores the results of calculations made by ALU.

Instruction Buffer Register (IBR):

: used to temporarily hold the right-hand instruction from the word in memory.

Memory Unit:

- : The memory unit consists of RAM, sometimes referred to as primary or main memory.
- : Unlike a hard drive (secondary memory), this memory is fast and also directly accessible by the CPU.
- : RAM is split into partitions. Each partition consists of an address and its contents (both in binary form).
- : The address will uniquely identify every location in the memory.
- : Loading data from permanent memory (hard drive), into the faster and directly accessible temporary memory (RAM) allows the CPU to operate much quicker.

Input/Output devices:

- : Program or data is read into main memory from the input device or secondary storage under the control of CPU input instruction. Output devices are used to output the information from a computer. If some results are evaluated by computer and it is stored in the computer, then with the help of output devices, we can present it to the user.

Harvard Architecture:

- The name is originated from 'Harvard Mark I' a relay based old computers.
- Harvard Architecture is the computer architecture that contains separate storage and separate buses for instruction and data.
- It was basically developed to overcome the bottleneck of Von Neumann Architecture.
- The main advantage of having separate buses for instruction and data is that CPU can access instructions and read/write data at the same time.

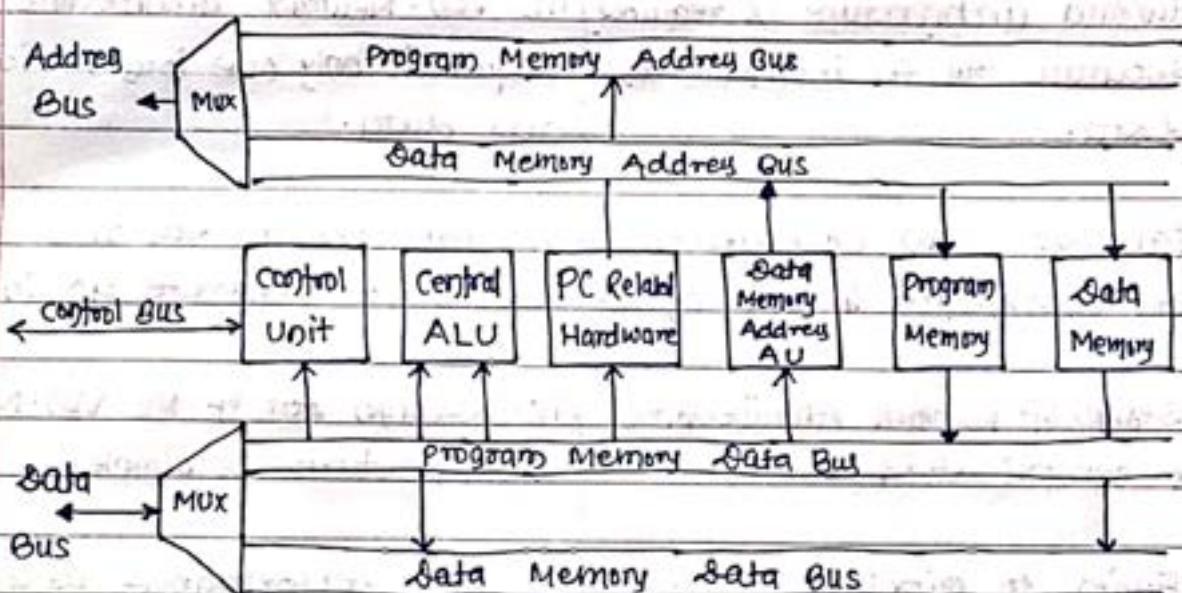


fig. Harvard Architecture

- There are two data and two address buses multiplexed for data bus and address bus. Hence, there are two blocks of RAM chips: one for program memory and another for data memory addresses.
- The Control unit controls the sequence of operations.
- central ALU consists of ALU, multiplier, accumulator, and scaling chief register.
- The PC is used to address program memory and always contains the address of next instruction to be executed.
- Here, data and control buses are bidirectional and address bus is unidirectional.

Difference between Harvard and von-Neumann Architecture:

Harvard Architecture

(i) Its name is originated from 'Harvard Mark I' a relay based old computer.

(ii) It required two memories for their instruction (program) and data.

(iii) Harvard architecture is required separate bus for instruction and data.

(iv) Processor can complete an instruction in one cycle.

(v) Design of Harvard Architecture is complicated.

(vi) Easier to pipeline, so high performance can be achieved.

(vii) Used in Laptop, Personal Computer and work station.

(viii) Two sets of memory and buses mean data can be handled more quickly which would result in decreasing execution time.

von Neumann Architecture

(i) It is named after the mathematician and early computer scientist John von Neumann based on store program computer concept.

(ii) It required only one memory for their instruction and data.

(iii) von-Neumann architecture is required only one bus for instruction and data.

(iv) processor needs two clock cycles to complete an instruction.

(v) Design for von-Neumann architecture is simple.

(vi) It is cheaper.

(vii) Used in microcontroller and signal processing.

(viii) Speed is limited when compared to Harvard due to only having one memory location and set of buses.

- (ix) USE RISC processor
 (Reduced Instruction set Computer)
 (x) More space is required in Harvard Architecture.

- (ix) USE CISC processor
 (Complex Instruction Set Computer)
 (x) Less space is required in Von Neumann architecture.

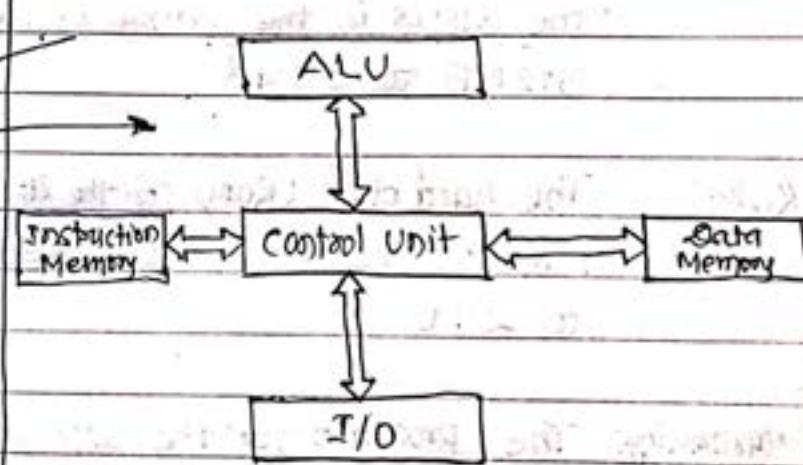
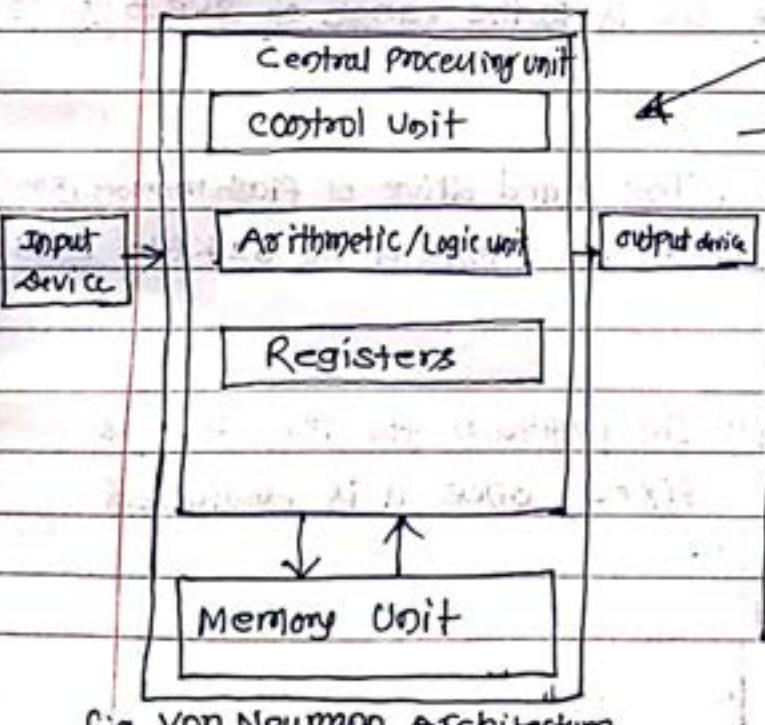
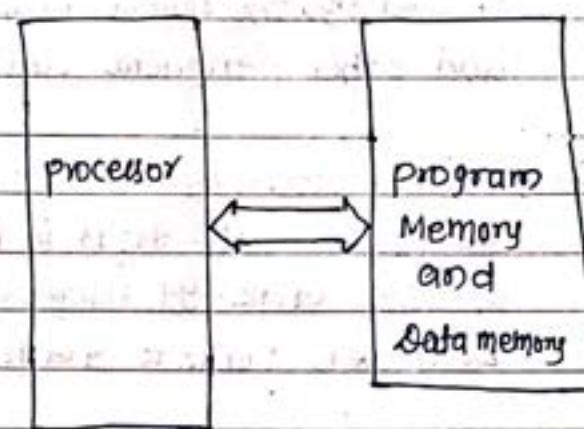
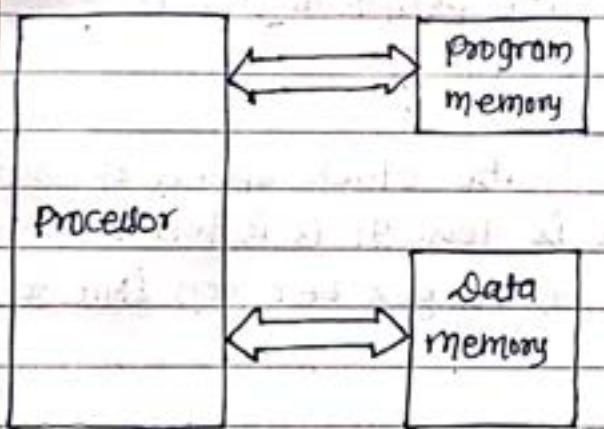


fig. Harvard Model

1.3 # Microprocessor and Microcontroller:

Microprocessor

Application: It is used where intensive processing is required. It is used in personal computers, Laptops, mobiles, video games etc.

Structure: It has only the CPU in the chip. Other devices like I/O port, memory, timer is connected externally.

The structure of EP is flexible. User can decide the amount of memory, the number of I/O port and other peripheral devices.

Clock speed: The clock speed of the EP is high. It is in terms of the GHz. It ranges between 1 GHz to 4 GHz.

RAM: The volatile memory (RAM) for the EP is in the range of the 512 MB to 32 GB.

ROM: The hard disk (ROM) for the EP is in the range of the 128 GB to 2 TB.

Programming: The program for the EP can be changed for different applications. The programming of the EP is difficult compared to the MC.

Microcontroller

It is used where the task is fixed and predefined. It is used in the washing machine, alarm, microwave, cameras etc.

: CPU, Memory, I/O port and all other devices are connected on the single chip.

The structure is fixed. Once it is designed the user cannot change the peripheral devices.

The clock speed of microcontroller is less. It is in terms of the MHz. It ranges between 1 MHz to 300 MHz.

The volatile memory (RAM) for the MC is in the range of 2 KB to 256 KB.

The hard drive or flash memory (ROM) for the MC is in the range of the 32 KB to 2 MB.

The program for the MC is fixed once it is designed.

Bit Size: 9t is available in 32-bit and 64-bit : 9t is available in 8-bit, 16-bit, and 32-bit.

Cost : The cost of MP is high compared to the MC : 9t is cheaper

Power consumption : The power consumption of the MP is high. : The power consumption for the MC is less.

Size: The overall size of the system is large. : The overall size of the system is small.

Architecture: Based on von Neumann architecture : Based on Harvard Architecture.

Power saving Features : Do not have power saving modes like idle mode and power saving mode. This helps to reduce power consumption even further.

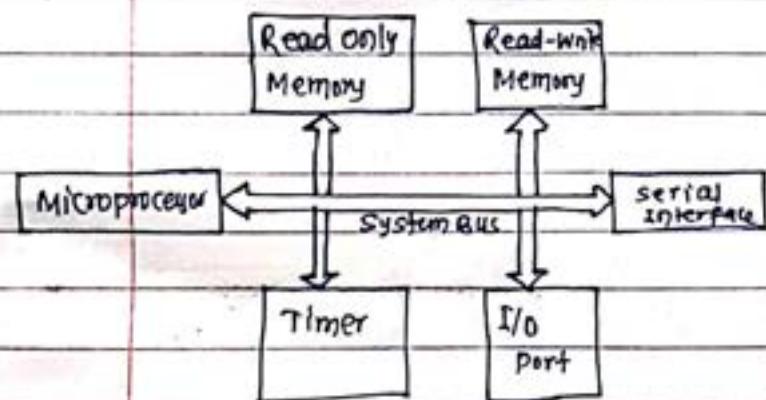


fig: Block diagram of MP

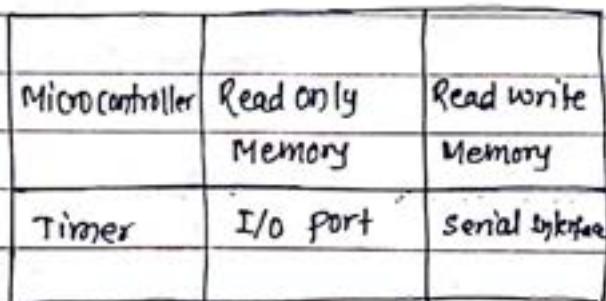


fig: Block diagram of MC

1.4 #

Internal architecture of 8 bit microprocessor 8085

The architecture of 8085 up is shown below:

Chapter-2

Programming with 8085 microprocessor

Internal Architecture of 8 bit microprocessor and its registers:

The Intel 8085 A is a complete 8 bit parallel central processing unit. The main components of 8085A are array of registers, the arithmetic logic unit, the encoder/decoder, and timing and control circuits linked by an internal data bus. The block diagram is shown below:

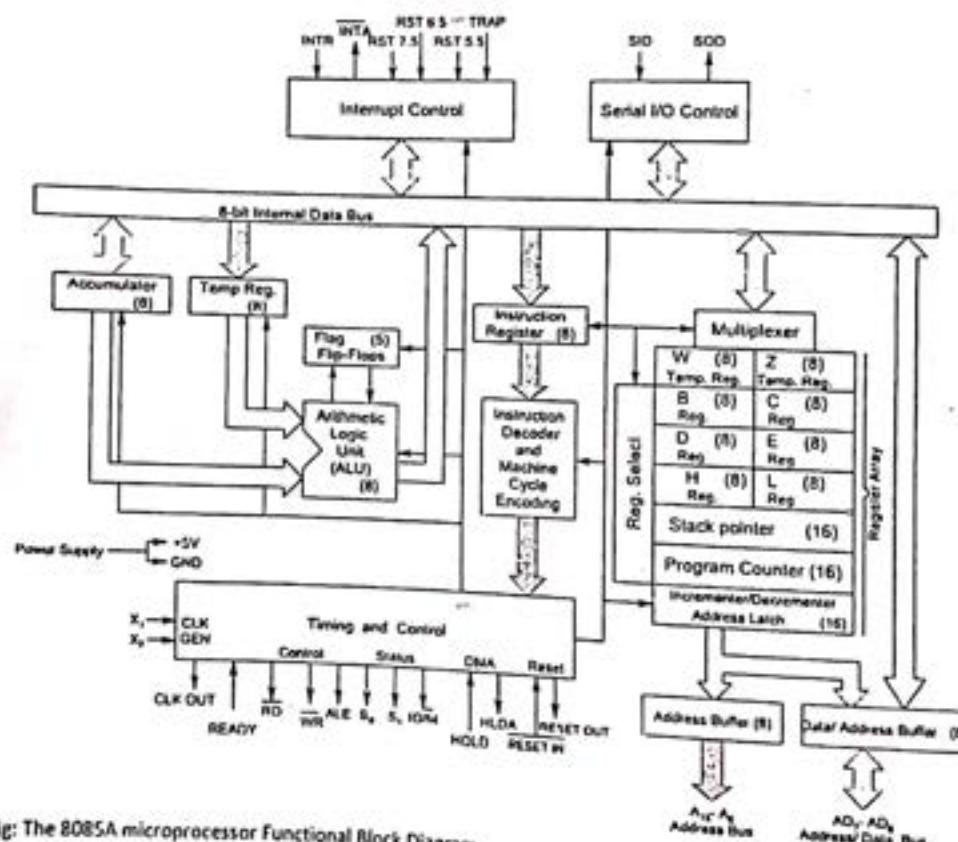


Fig: The 8085A microprocessor Functional Block Diagram

The various functional blocks of 8085 UP are as follows:

- (i) Registers
- (ii) Arithmetic & Logic unit (ALU)
- (iii) Timing & control circuitry
- (iv) Instruction decoder and machine cycle encoder
- (v) Interrupt control
- (vi) Serial I/O control
- (vii) Address Buffer and Address/Data buffer
- (viii) Internal data bus

(i) Registers:

(a) Accumulator or Register A:

- 8085 UP is accumulator based processor.
- It is an 8-bit programmable register which is used to accumulate the final result of any ALU operations.
- It is one of the two operands of ALU and is an implicit one.
- Any data input/output to/from the UP takes place via the accumulator.
- It is generally used for temporary storage of data and for the placement of final result of arithmetic/logic operations.

(b) Temporary Register (W and Z):

- There are 3 temporary register (one operand of ALU CT) and W & Z register.
- These registers are not available to the programmer, but uses them internally to hold temporary data during execution of some instructions.

(c) General purpose registers

- The general purpose registers are B, C, D, E, H and L.
- They are all 8-bit register but can also be used as 16-bit.

register pairs - BC, DE and HL

- These register are used in programming so called programmable registers.
-

(d) Stack pointer (SP):

- It is the 16-bit register that holds the address of the top location of the stack.
- The stack is an area of R/W memory in which temporary information is stored in First in last out (or Last in first out) basis.
- It is incremented / decremented during PUSH and POP operation.
- On every PUSH SP get decremented & on every POP SP get increment.

(e) Program Counter (PC):

- It is a 16-bit register that holds the address of the next instruction to be executed.
- As the processor executes instructions one after another, the PC is incremented - the number by which the PC increments depends on the nature of the instruction.
- Eg: for a 1-byte instruction PC is incremented by 1, while for a 3-byte instruction, the processor increments PC by 3 address locations.

(f) Flag Register

- It is an 8-bit register in which five bit positions contain the status of five condition flags which are zero (Z), sign (S), carry (CY), parity (P) and auxiliary carry (AC).
- Each of these five flags is a 1 bit F/F.
- The flag register format is shown in fig. below:

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
S	Z	X	AC	X	P	X	CY

Fig. Flag register format

- flags are set or reset according to the result of the operation.

(g) Instruction Register (IR):

- It is a 16-bit register that holds the address of the next instruction to be executed.

- It is not accessible to the programmer.

- It receives the operation codes of instruction from internal data bus and passes to the instruction decoder which decodes so that UP knows which type of operation is to be fetched.

-

(h) Increment/decrement address latch register:

- This 16-bit register increments/decrements the contents of PC or SP when instructions related to them are executed.

(ii) Arithmetic & Logic Unit (ALU):

- The ALU performs the actual numerical and logical operation such as 'add', 'subtract', 'AND', 'OR', etc

- ALU consists of accumulator, flag register and temporary register.

- ALU uses data from memory and from accumulator to perform the arithmetic & logic operations and always stores the result of the operation in accumulator.

(iii) Timing and control unit:

- The control unit generates signals within UP to carry out the instruction, which has been decoded.

- In reality, it causes certain connections between block of the UP to be opened or closed, so that the data goes where it is required and the ALU operations occur.

(iv) Address buffer and Data/address buffer:

- These registers hold the address/data, received from PC/internal data bus and then load the external address and data buses.
- These registers actually behave as the buffer stage between the UP and external system buses.

(v) Internal Data bus (8-bit)

- It is used for the interconnection of different operational units of UP. i.e. the processor uses this bus for transferring a data from one register to another register, ALU to register etc within the UP.

(vi) Serial I/O Control:

- Serial I/O control provides two lines, SOD and SID for serial communication.
- The serial output data (SOD) line is used to send data serially and serial input data line (SID) is used to receive data serially.

(vii) Interrupt control:

- It is an obstacle or disturbance to the processor
- It is sent by different devices to take CPU time for some time to perform a specific task to of that particular device.
- The interrupt signals are INTR, RST 5.5, RST 6.5, RST 7.5, TRAP and one acknowledge signal INTA.

(viii) Instruction decoder & Machine cycle Encoder:

- The IR sends the machine code to this unit.
- This unit, as its name suggests, decodes the opcode and finds out what is to be done in response of the coming opcode and how many machine cycles are required to execute this instruction.

1.5 # Concept of fetch, decode and execution: (Instruction cycle)

- : The sequence of steps in which instructions are loaded from memory & executed is called instruction cycle.
- : This cycle describes the process of execution of an instruction within the computer.
- : One round of steps from getting an instruction back to getting the next instruction is called instruction cycle.
:-
- * Fetch - Get an instruction from main memory
- * Decode - Translate it into computer commands
- * Execute - Actually process the command
- :- The computer takes a certain period to complete instruction fetching, decoding, and executing cycle on the basis of clock speed. Such time period is called 'instruction cycle'.
- :- Time required to complete execution of an instruction.

Step 2: decode instructions into computer's step 3: Execute Commands

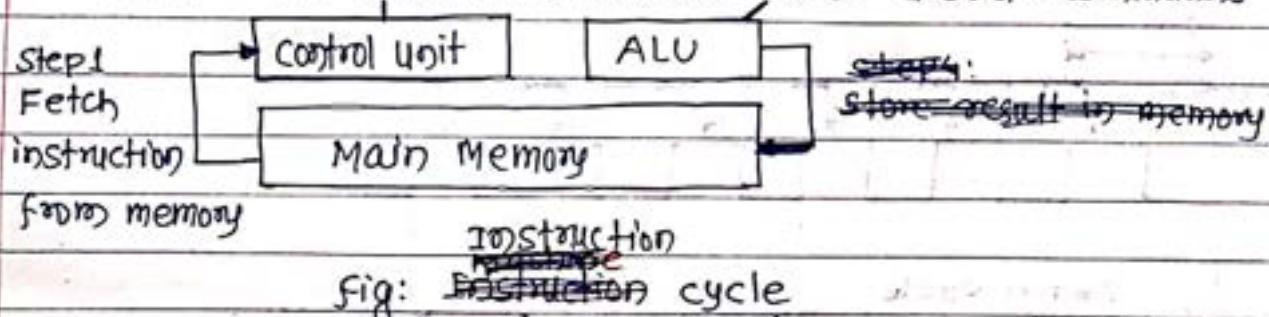


fig: ~~Instruction~~ cycle

Fetching:

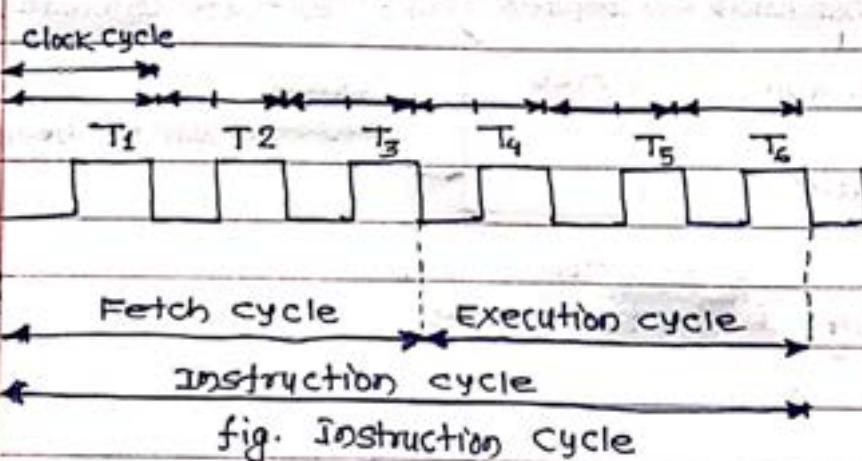
- : The CPU reads the value of PC and the instruction is copied into the instruction register (IR). This process involves following steps:
 1. Copy the contents of PC into MAR
 2. Copy the data from the memory to MBR
 3. Instruction is copied into IR.
- : After completing the above steps the value of PC is incremented and it points the address of next instruction.

Decoding:

- Decoding means to activate the appropriate circuit to execute the instructions.
- After completing the fetch process the CU decodes the instruction by analyzing the opcode of the instruction.
- CU also reads the value of operands specified in the **Exe** instruction.

Execution & storage:

- The process of performing an action on the decoded instruction is called execution.
- After decoding CPU executes the instruction by using the activated circuit.
- After execution the results are copied into the register and memory.



Flags in 8085

- A flag is a flip flop.
- It indicates some conditions produced by the execution of an instruction.
- The flag register of 8085 UP consists of 5 flags.
- The flag register is connected to ALU.
- When an operation is performed by ALU the result is transformed transferred on data bus and status of result will be stored in flip flops.
- They are called Zero(Z), carry(CY), sign(S), parity(P) and auxiliary carry(AC) flags.
- The UP uses these flags to set and test data conditions.
- The flag register format is shown in fig. below:

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
S	Z	X	AC	X	P	X	CY

fig. Flag register in 8085

- The flags are stored in the 8-bit register so that the programmer can examine these flags by accessing the register through an instruction.

Sign flag (S):

- Sign flag indicates whether the result of a mathematical operation is negative or positive.
- If the MSB or D₇ bit of the result of an operation is 1, this flag is set, otherwise it is reset. i.e If D₇=1, S=1 (Negative)
If D₇=0, S=0 (Positive)

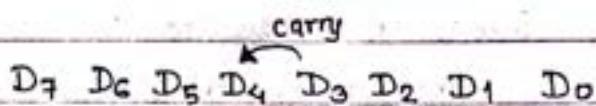
Zero flag (Z):

- This flag indicates whether the result of an mathematical or logical operation is zero or not.
- If the result of the current operation is zero, then this flag will be set, otherwise reset.

Auxiliary carry (AC) flag:

- If there is a carry out of bit 3 and into bit 4 resulting from the execution of an arithmetic operation, it is set otherwise reset.

:



- This flag is used for BCD operation and is not free to the programmer.

Parity (P) flag:

- This flag is set when the result of an operation contains an even number of 1's and reset otherwise.

Carry (CY) flag:

- If an instruction results in a carry (for addition operation) or borrow (for subtraction or comparison) out of bit D₇, then this flag is set, otherwise reset.

Eg: ADD B

If register B has 41H and accumulator has 57H. Then

$$B: 41H = 0100\ 0001$$

$$A(\text{Before Addition}) 57H = 0101\ 0111$$

$$A(\text{After Addition}) 98H = 1001\ 1000$$

Since D₇ is 1, sign flag is set i.e S=1

Result is non zero, zero flag is reset i.e Z=0

There is no carry from D₃ into D₄, Auxiliary flag is reset
i.e AC=0

There is odd number of 1's in result, parity flag is reset, i.e $P=0$
There is no carry or borrow out of bit D₇, carry flag is reset,
i.e $CY=0$.

CH-2 Assembly Language Programming

2.1 Instruction Formats (OpCodes, mnemonics and operands)

- : It is a symbolic or diagrammatic representation of any instructions.
- : An instruction consists of an operation code (called opcode) and the address of the data (called operand), on which the opcode operates.
- : This is the structure on which an instruction is based.
- : The opcode specifies the nature of the task to be performed by an instruction.
- : Symbolically, an instruction looks like

Operation code	Address of data
opcode	operand

- : 8085 can handle at the maximum of 256 ($= 2^8$) instructions; however, only 246 instructions are used in 8085.
 - : The sheet which contains all these instructions with their hex code, mnemonics, decryptions and functions is called an instruction sheet.
- :- Opcode:** opcode is a part of instruction that tells the processor what should be done.
- :- Operand:** operand is a part of the instruction that contains the data to be acted on, or the memory location of the data in a register.
- Eg: MVI A,B
here instruction MVI is an opcode. A & B are operands
- Eg: ADD R₁,R₂
here ADD is the opcode and R₁ and R₂ are the operand.
- :- Mnemonic :** Mnemonic is the acronym/abbreviation, for operation : it is used in instruction code to make easy and suitable coding.

Eg: The mnemonics are R used for the register,
A for the accumulator, Z for zero flags,
ADD for addition etc.

The instruction format can be classified into the 3 categories

(i) 1 Byte Instruction:

- In 1 Byte instruction, the opcode and the operand
are in the same byte i.e.

opcode / operand

Eg: ADD B, MOV A,B

(ii) 2-Byte Instruction:

- In 2-Byte instruction, the first byte will be opcode
and second byte will be the operand or data/ address

opcode	operand
--------	---------

Eg: MVI C, 07H

(iii) 3-Byte Instruction:

- In 3-Byte instruction, the first byte will be opcode,
second byte will be low order byte of address, and
third byte will be High order byte of address

opcode	Low order byte of address	High order byte of address
--------	---------------------------	----------------------------

Eg: LXI B, 2050H

* **opcode:** Specifies what operation to be performed.

* **Operand:** Specifies where to perform the operation.

2.8 8085 Instruction sets:

- An instruction is a command given to the microcomputer to perform a specific task or function on a given data.
- An instruction set is a collection of instructions that the microprocessor is designed to perform.
- Functionally, the instructions can be classified into five groups:

- I Data Transfer (copy) Group
- II Arithmetic Group
- III Logical Group
- IV Branch Group
- V Stack, I/O and machine control Group

I Data Transfer (copy) Group:

- This instruction copies data from one location called source to another location called destination without modifying the content of the source.
- The transfer of data may be between the registers or between register and memory or between an I/O device and accumulator.
- None of these instructions changes the flag.
- The instruction of this group are:-

1. MOV Rd, Rs (Move Register to Register)

- 1-byte instruction
- copies the contents of the source register to destination register.
- Rd & Rs may be A, B, C, D, E, H and L
- Eg. MOV A, B ; A \leftarrow B

2. MOV M, R (Move to memory from Register)

- 1-byte instruction
- Copies the contents of the specified register to memory.
- Here memory is the location specified by the content of HL Register pair
- Eg. MOV M, B

(3) MOV R, M (Move to Register from Memory)

: 1-byte instruction

: copies the contents of memory location as specified by HL register pair to a Register

: Eg., MOV B, M

(4) MVI R, 8-bit data (Move Immediate Data to Register)

: 2-byte instruction

: Loads the 8-bit data into the specified register

: R may be A, B, C, D, E, H and L

: Eg. MVI B, 57H ; B \leftarrow 57H

(5) MVI M, 8-bit data (Load Memory with Immediate Data)

: 2-byte instruction

: Loads the 8-bit data to the memory location whose address is specified by the contents of HL register pair.

: Eg.: MVI M, 35H ; [HL] \leftarrow 35H

(6) LXI Rp, 16-bit data (Load Register pair with Immediate Data)

: 3-byte instruction

: Load immediate 16-bit data to register pair.

: Register pair may be BC, DE, HL and SP

: First it loads lower 8-bit data into lower order register and then loads higher 8-bit data into higher order register.

: Eg. LXI B, 4532H ; B \leftarrow 45, C \leftarrow 32H

(7) LDA 16-bit address (Load Accumulator direct)

: 3-byte instruction

: Loads the contents of memory location whose address is specified by 16-bit address.

: Eg. LDA 4035H ; A \leftarrow [4035H]

(8) LDAX Rp (Load Accumulator Indirect)

: 1-byte instruction

: Loads the contents of memory location pointed by the contents of register pair to accumulator.

: Eg. LDAX B; [A] \leftarrow [[BC]]

(9) STA 16-bit address (Store Accumulator content Direct)

: 3-byte instruction

: Stores the contents of accumulator to specified address

: Eg. STA FA00H; [FA00] \leftarrow [A]

(10) STAX Rp (Store Accumulator content Indirect)

: 1-byte instruction

: Stores the contents of accumulator to memory location specified by the contents of register pair.

: Eg. STAX B; [BC] \leftarrow A

(11) LHLD 16-bit address (Load HL pair Direct)

: 3-byte instruction

: Loads the contents of specified memory location to L-register and contents of next higher location to H-register.

: Eg. LHLD 5000H ; H = 03, L = 02

Lower Byte	5000	02	↑
Higher Byte	5001	03	↑

(12) SHLD 16-bit address (Store HL pair Direct)

: 1-byte instruction

: It is opposite to LHLD instruction

: Stores the contents of L-Register to specified memory location and contents of H-Register to next higher memory location.

: Eg. SHLD 2000H ; [2000] = 05, [2001] = 04

2000	05	←
2001	04	←

(13) XCHG (EXchange HL pair with DE pair)

: 1 - byte instruction

: EXchange DE pair with HL Pair

: Eg. XCHG ; H = 95, L = 32 ; D = 75, E = 00 $\begin{bmatrix} H = 75, L = 00 \\ D = 95, E = 32 \end{bmatrix}$

LXI H ; 9500H

LXI D ; 9500H

(14) XCHG ; H = 95, L = 32 ; D = 75, E = 00

(14) IN ^{port} 8-bit address (Input data from Input port)

: 2 - byte instruction

: Reads data from the input port address specified in the second byte and loads data into the accumulator i.e., input port to accumulator.

: Eg. IN 40H ; A $\leftarrow [40H]$

(15) OUT = 8-bit port address (Output data to output port)

: 2 - byte instruction

: copies the contents of the accumulator to the output port address specified in the second byte i.e., accumulator to output port.

: Eg. OUT 40H ; [40] $\leftarrow A$

(Move HL to PC)

(16) PCHL (Load the ^{or} program counter with HL contents)

: 1 - byte instruction

: The contents of HL Register pair are copied with into the program counter

: The contents of H are placed as the higher order byte and the contents of L as the loworder byte.

: Eg. PCHL ; PC $\leftarrow HL$

(17) SPHL (Move HL to SP)

: 1-byte instruction

: It is an instruction with the help of which stack pointer will get initialized with the contents of register pair HL

: Eg. SPHL ; $SP \leftarrow HL$

(18) XTHL (exchange Top of Stack with HL pair)

: 1-byte instruction

: This instruction exchanges the contents of the top two location of the stack with the contents of register pair HL.

: Here it is not an exchange between SP with HL

: At location $(SP) \leftarrow$ contents of L-Register

: At location $(SP+1) \leftarrow$ contents of H-Register

(19) PUSH Rp (Store Register pair on stack)

: 1-byte instruction

II Arithmetic Group Instructions

- : These instructions perform arithmetic operations such as addition, subtraction, increment and decrement.
- 1. ADD R/M (Addition)
 - 1-byte instruction
 - Adds the contents of register/memory to the contents of the accumulator and stores the result in accumulator.
 - All flags are affected
 - Eg. ~~ADD~~ B; $A \leftarrow A + B$
~~ADD~~ M; $A \leftarrow A + [HL]$
- 2. ADI 8-bit data (Addition Immediate)
 - 2-byte instruction
 - Adds the 8-bit data with the contents of accumulator and stores result in accumulator.
 - All flags are affected
 - Eg: ADI 25H ; $A \leftarrow A + 25H$
- 3. ADC R/M (Addition with carry)
 - 1-byte instruction
 - It adds the contents of register/memory with the content of accumulator using previous carry and carry flag and stores the result in accumulator.
 - All flags are affected
 - Eg. ADC B; $A \leftarrow A + B + CY$
ADC M; $A \leftarrow A + [HL] + CY$
- 4. ~~ACI~~ ACI 8-bit data (Addition with Carry Immediate)
 - 2-byte instruction
 - It adds the 8-bit data and the carry flag with the contents of accumulator and stores the result in accumulator.
 - All Flags are affected
 - Eg . ACI 45H; $A \leftarrow A + CY + 45H$

5. **DAD RP** (Double Addition)
- : 1-byte instruction
 - : Adds the contents of Register pair with the content of HL pair and stores the 16-bit result in HL pair.
 - : only carry flag is affected if the result is greater than 16-bit.
 - : Eg: DAD B; $HL \leftarrow [HL] + [BC]$
6. **SUB R/M** (Subtraction)
- : 1-byte instruction
 - : Subtracts the contents of specified register/memory with the contents of accumulator and stores the result in accumulator.
 - : All flags are affected
 - : Eg. SUB D; $A \leftarrow A - D$
SUB M; $A \leftarrow A - [HL]$
7. **SBB R/M** (Subtraction with Borrow)
- : 1-byte instruction
 - : It subtracts the contents of Register/Memory and Borrow flag from the content of accumulator.
 - : All flags are affected
 - : Eg: SBB B; $A \leftarrow A - B - \text{Borrow}$
8. **SBI** 8-bit data (Subtraction with Borrow Immediate)
- : 2-byte instruction
 - : It subtracts the 8-bit data and the Borrow flag from the contents of the accumulator and the result is stored in the accumulator.
 - : All flags are affected.
 - : Eg. SBI 45H; $A \leftarrow A - 45H - \text{Borrow}$

9. ~~INR~~ INR R/M (increment)
DCR R/M (decrement)

- : 1-byte instruction
- : INR R/M and DCR R/M increases and decreases the contents of R(Register) or M(Memory) by 1 respectively.
- : All flags are affected except carry.
- : Eg: INR B; $B = B + 1$
INR M; $[HL] = [HL] + 1$
DCR B; $B = B - 1$
DCR M; $[HL] = [HL] - 1$
- : The result is stored in the same place.

10. INX Rp (Increment Register pair)
DCX Rp (Decrement Register pair)

- : 1-byte instruction
- : Increases and decreases the content of register pair by 1.
- : And the result is stored in the same place.
- : No flags are affected
- : Eg: INX BC; $BC = BC + 1$
DCX DE; $DE = DE - 1$

11. DAA

- : 1-byte instruction
- : Used only after addition
- : The content of accumulator is changed from binary to two 4-bit BCD digits.
- : This is the only instruction that uses the auxiliary flag to perform the binary to BCD conversion. and the conversion procedure is:-
 - (a) If the value of the low order 4-bits in the accumulator is greater than 9 or if AC flag is set, the instruction adds 6 to the low order four bits.

(b) if the value of the high order 4-bits in the accumulator is greater than 9 or if the carry flag is set, the instruction adds 6 to the high-order four bits.

III Logical Instruction:

The instruction of this group perform logical operation like AND, OR, compare, rotate etc.

The logical operations have the following instructions.

1. ANA R/M (Logical AND)



- 1-byte instruction

- The contents of the accumulator are logically ANDed with the contents of the operand (register or memory) and the result is placed in the accumulator.

- CY is reset, AC is set and others as per result.

- Eg. Let A contains 54H and D contains 82H. Then ANAD is executed as

$$54H = 0101 \ 0100$$

$$82H = \underline{1000} \ 0010$$

$$ \underline{\underline{0000}} \ 0000$$

Flags: S=0, Z=1, P=1, AC=1, CY=0

2. ANI 8-bit data (AND immediate with accumulator)

- The contents of the accumulator are logically ANDed with the contents 8-bit data and the result is placed in the accumulator.

- 2-byte instruction

- CY is reset, AC is set and others as per result.

- Eg. ANI 25H

3. ORA R/M (Logically OR with accumulator)

- The contents of the accumulator are logically ORed with the contents of the operand (register or memory), and the result is placed in the accumulator.

- 1-byte instruction

- Eg. CY and AC reset and others as per result.

- Eg. ORA C;

Let A has 03H and register C has 81H. Then ORA C is

$03H = 0000\ 0011$

$81H = 1000\ 0001$

$\underline{83H} = 1000\ 0011$

$S=1, Z=0, P=0, CY=0, AC=0$

4. ORI 8-bit data (Logically OR with Immediate data)

: 2-byte instruction

: The contents of the accumulator is logically ORed with the 8-bit data (operand) and the result is placed in the accumulator.

: CY and AC are reset and others as per result.

: Eg. ORI 54H

5. XRA R/M (Exclusive OR with accumulator)

: 1-byte instruction

: The contents of the accumulator are exclusive ORed with the contents of the operand (Register or memory), and the result is placed in the accumulator.

: CY and AC are reset and others as per result.

: Eg. A has 77H and register D has 56H. Then XRA D is

A : $77H = 0111\ 0111$

D : $56H = 0101\ 0110$

XOR : $\underline{0010\ 0001}$

$\Rightarrow D$

$S=0, Z=0, P=1, CY=0, AC=0$

6 XRI 8-bit data (Exclusive OR immediate with accumulator)

: The contents of the accumulator are exclusive ORed with the 8-bit data (operand) and the result is placed in the accumulator.

: 2-byte instruction

: CY and AC are reset and others as per result.

: Eg. XRI 25H

7. CMA (Complement Accumulator)

: 1-byte instruction

: The contents of accumulator are complemented

: No flags are affected.

: Eg. CMA ; $A \leftarrow \bar{A}$

8. CMP R/M (compare with accumulator)

: 1-byte instruction

: The contents of the operand (register/Memory) are compared with the contents of the accumulator.

: All flags are modified

CPI 8-bit data (compare Immediate with accumulator)

: 2-byte instruction

: The 8-bit data is compare with the contents of accumulator.

: The content of operands are not modified.

: All flags are modified

:- The result of the comparison is shown by setting the flags:

case	CY	Z
[A] < [R/M] or 8-bit data	1	0
[A] = [R/M] or 8-bit data	0	1
[A] > [R/M] or 8-bit data	0	0

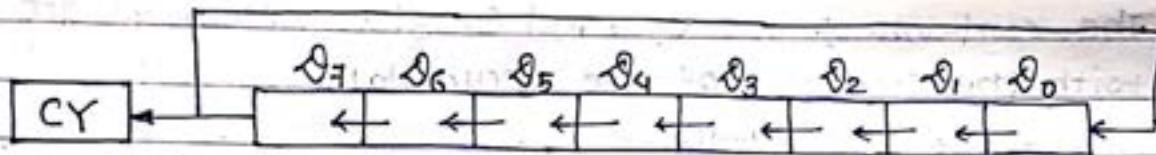
Eg ~~CMP~~ CMP B ; compares register B with accumulator.

CPI 76H ; compares 76H with accumulator.

9. Logically Rotate Instruction:

(i) RLC (Rotate Accumulator Left)

- 1-Byte instruction
- Each bit is shifted to the adjacent left positions.
- Bit θ_7 is placed in the position of θ_0 as well as carry in the carry flag.
- Carry flag (CY) is modified according to bit θ_7 .
- S,Z,P, AC are not affected



$$CY = \theta_7, \theta_7 = \theta_6, \theta_6 = \theta_5, \theta_5 = \theta_4, \theta_4 = \theta_3, \theta_3 = \theta_2, \theta_2 = \theta_1, \theta_1 = \theta_0, \theta_0 = \theta_7$$

If Accumulator content = 95H and CY=0 Then execution of RLC will change the contents of CY flag and Accumulator as follows:

CY	θ_7	Acc	θ_0
0	1 0 0 1 0 1 0 1		

Accumulator content before instruction

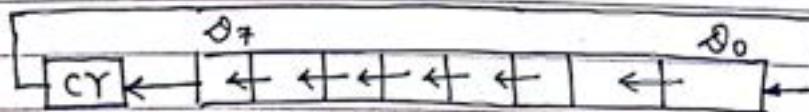
CY	θ_7	Acc	θ_0
1	0 0 1 0 1 0 1 1		

Accumulator content after instruction

Here, θ_7 is placed in θ_0 as well as in CY flag.

(ii) RAL (Rotate Accumulator Left through carry)

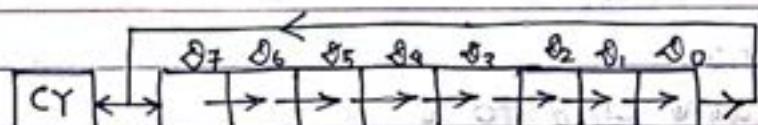
- 1- byte instruction
- Each bit is shifted to the adjacent left position.
- Bit θ_7 becomes the carry bit and the carry bit is shifted into θ_0 .
- S,Z,P, AC are not affected.



$$CY = \theta_7, \theta_7 = \theta_6, \theta_6 = \theta_5, \theta_5 = \theta_4, \theta_4 = \theta_3, \theta_3 = \theta_2, \theta_2 = \theta_1, \theta_1 = \theta_0, \theta_0 = CY$$

(iii) RRC (Rotate Accumulator Right)

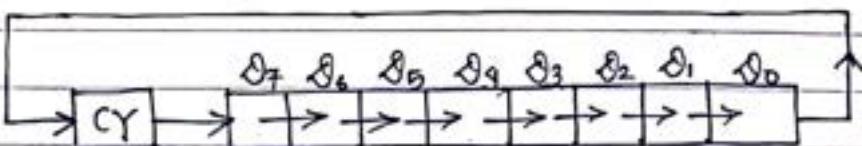
- 1-byte instruction
- Each bit is rotated right by one position.
- * Bit D_0 is placed in the position of D_7 as well as in the carry flag. CY is modified according to D_0 .
- S, Z, P, AC are not affected.



$$CY = D_0, D_7 = D_0, D_6 = D_7, D_5 = D_6, D_4 = D_5, D_3 = D_4, D_2 = D_3, D_1 = D_2, D_0 = D_1$$

(iv) RAR (Rotate accumulator Right through carry)

- 1-byte instruction
- Each bit is shifted rotated right by one position through the carry flag.
- Bit D_0 is placed in the carry flag and the carry flag is placed in D_7 .
- The carry flag is modified according to D_0 .



$$CY = D_0, D_0 = D_1, D_1 = D_2, D_2 = D_3, D_3 = D_4, D_4 = D_5, D_5 = D_6, D_6 = D_7, D_7 = CY$$

(10) CMC (complement carry)

- 1-byte instruction
- It complements the carry flag
- No other flags are affected

(11) STC (Set Carry flag)

- 1-byte instruction
- It sets the carry flag to 1
- No other flags are affected

(12) CMA (Complement Accumulator)

- 1-byte instruction
- It complements the content of accumulator

IV Branching Instructions

These instructions alter the sequence of program execution either conditionally or unconditionally.

(1) # Jump instruction:

(a) Unconditional jump:

Unconditional jump instructions to enable the programmer to set up continuous loop without depending on any type of conditions.

(1) JMP 16-bit address

3-byte instruction

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag.

Eg. JMP 2034H

The jump location can also be specified using a label (or name). However we, should not specify both a label and its 16-bit address in a jump instruction.

(b) Conditional jump

The conditional jump instructions allow the jump to

The program sequence is transferred to the memory location specified by the 16-bit address given in the operand based on the specified flag of the PSW as described below.

(2) JC 16-bit address : Jump on carry, flag status: CY=1 Eg. JC 2050H

(3) JNC 16-bit address : Jump on no carry, Flag status: CY=0 Eg. JNC 2050H

(4) JP 16-bit address : Jump on positive, flag status: S=0 Eg. JP 2050H

(5) JM 16-bit address : Jump on Minus , flag status : S=1 Eg. JM 2050H

(6) JZ 16-bit address : Jump on zero, flag status : Z=1 Eg. JZ 2050H

(7) JNZ 16-bit address : Jump on no zero, flag status : Z=0 Eg. JNZ 2050H

(8) JPE 16-bit address : Jump on Parity even, flag status: P=1 Eg. JPE 2050H

(9) JPO 16-bit address : Jump on parity odd, flag status : P=0 Eg. JPO 2050H

B) Call and Return Instructions (Subroutines)

- : call and return instructions are associated with subroutine technique.
- : Subroutines:
- : A subroutine is a group of instructions that perform a subtask.
- : A subroutine is written as a separate unit apart from the main program and the microprocessor transfers the program execution sequence from main program to subroutine whenever it is called to perform a task.
- : After the completion of subroutine task, CPU returns to main program.
- : The subroutine technique eliminates the need to write a subtask repeatedly, thus it uses memory efficiently.
- : Before implementing the subroutine, the stack must be defined; the stack is used to store the program that follows the subroutine call.

Unconditional Call:

1. CALL 16-bit address
2. CC 16-bit address call on carry, flag status: CY=1 Eg CC 2050H
3. CNC 16-bit address call on No carry, flag status: CY=0 Eg CNC 2050H
4. CP " call on Positive, flag status: S=0 Eg CP 2050H
5. CM " call on Minus, flag status: S=1 Eg CM 2050H
6. CZ " call on zero, flag status: Z=1 Eg CZ 2050H
7. CNZ " call on no zero, flag status: Z=0 Eg CNZ 2050H
8. CPE " call on parity Even, flag status: P=1 Eg CPE 2050H
9. CPO " call on odd Parity, flag status: P=0 Eg CPO 2050H

Unconditional Return (RET)

1. RET
- : 1-byte instruction
- : Returns from
- : The program sequence is transferred from the subroutine back to the calling program.

Conditional Return

2.	RC	Return on Carry, Flag status: CY=1	Eg RC
3.	RNC	Return on No carry, Flag status: CY=0	Eg RNC
4.	RP	Return on positive, Flag status: S=0	Eg RP
5.	RM	Return on Minus, Flag status: S=1	Eg RM
6.	RZ	Return on Zero, Flag status: Z=1	Eg RZ
7.	RNZ	Return on Non zero, Flag status: Z=0	Eg RNZ
8.	RPE	Return on Parity Even, Flag status: P=1	Eg RPE
9.	RPO	Return on parity odd, Flag status: P=0	Eg RPO
10.			

(C) Restart Instructions

- : These are 1-byte instructions and transfer the program execution to a specific location.

1.	RST 0	0000H	(Restart address)
2.	RST 1	0008H	"
3.	RST 2	0010H	"
4.	RST 3	0018H	"
5.	RST 4	0020H	"
6.	RST 5	0028H	"
7.	RST 6	0030H	"
8.	RST 7	0038H	"

- : These instructions are generally used in conjunction with interrupts and inserted using external hardware.
- : When RST instruction is executed, the ~~contents~~ of PC on SP and the 8085 stores the contents of PC on SP and transfer the program to the restart location.

(7) Stack, I/O, & Machine Control Group

Stack

- (a) PUSH RP (Store Register pair on stack)
- (b) POP RP (Retrive Register pair from stack)

I/O

- (a) IN 8-bit port address (Input data from Input Port)
- (b) OUT 8-bit port address (Output data to output port)

Machine control

- (a) NOP: (No operation)

- No operation is performed

- The instruction is fetched & decoded. However no operation is executed.

- (b) HLT (Halt)

: The CPU finishes executing the current instruction and halts any further execution.

2.4 Addressing Modes of 8085

- In any microprocessor instructions, the source and destination operands can be a register, memory location, 8-bit number. The method by which the address of source of data and the address of the destination of the result given in the instruction are called the addressing modes.
 - The various ways of specifying operands for an instruction are called the addressing modes.
 - Addressing mode specifies where the operands are located rather than their nature.
 - The 8085 UP has five addressing modes
- (1) Immediate Addressing Mode:
- In this instruction mode 8/16 bit data is specified in instruction itself as one of its operand.
 - If immediate data is of 8-bits the whole instruction will be of 2-bytes. If the immediate data is of 16-bit the whole instruction will be a 3-byte instruction.
 - Example

MVI B, 20H ; 20H is copied into register B.

LXI B, 1000H ; Load the BC pair with immediate data 1000H

ADI 06H ; Add 06H to the content of the accumulator

Whenever the symbol (I) is present in the instruction, then it is a immediate addressing mode.

(2) Direct Addressing Mode: (Absolute Addressing Mode)

- In this mode 8/16 bit address is directly specified in instruction itself as one of its operand.
- The instruction size is either 2-bytes or 3-bytes
- Example

LDA 3000H; Load the contents of memory location 3000H in accumulator

STA 2400H; Store the content of accumulator to memory location 2400H.

INT 02H; Input data from the input port 02H in accumulator.

- In this mode for 3-byte instruction, 2nd & 3rd byte is the memory address for 2-byte instruction, 2nd byte is the port address

(3) Register Direct Addressing Mode :- (Register Addressing Mode)

- In this mode specified the register or register pair that contains the data rather than address.

- Example:

MOV A,B ; Move the Content of Register B to the accumulator.

ADD B ; Add the Content of Register B to the content of accumulator.

DRA C ;

- ~~Size~~ The instruction size is 1-byte

- In Register addressing mode the operand (i.e source or destination) is a general purpose register.

(4) Register Indirect Addressing Mode :-

- In this instruction mode 16-bit memory address is indirectly provided with the instruction using a register pair.

- Example Here the data will be in memory and its 16-bit

- ~~Address~~ address will be seen in the 16-bit register pair.

- Example:

MOV A,M ; Move the contents of the memory location pointed by the HL Register pair to the accumulator.

LDA X B ; Load the accumulator with the content of memory location pointed by the BC Register pair.

ADD M ; Add the content of memory location pointed by HL register to the Content of accumulator.

MVI M, 5DH ; Move the immediate data to the memory location pointed by the HL pair

(5) Implied Addressing Mode:-

- In this mode doesn't require any operand, data is Specified by the opcode itself.

- Example

CMA ; Complement the Content of accumulator

NOP ; No operation

HLT ; Halt

RAL REA ; Rotate Accumulator Left

- The instruction size is 1-byte

2.6 Timing Diagram:

Instruction cycle:

- Instruction cycle is defined as time required to complete execution of an instruction.
- 8085 instruction cycle consists of 1 to 6 Machine cycles. or

Machine cycle:

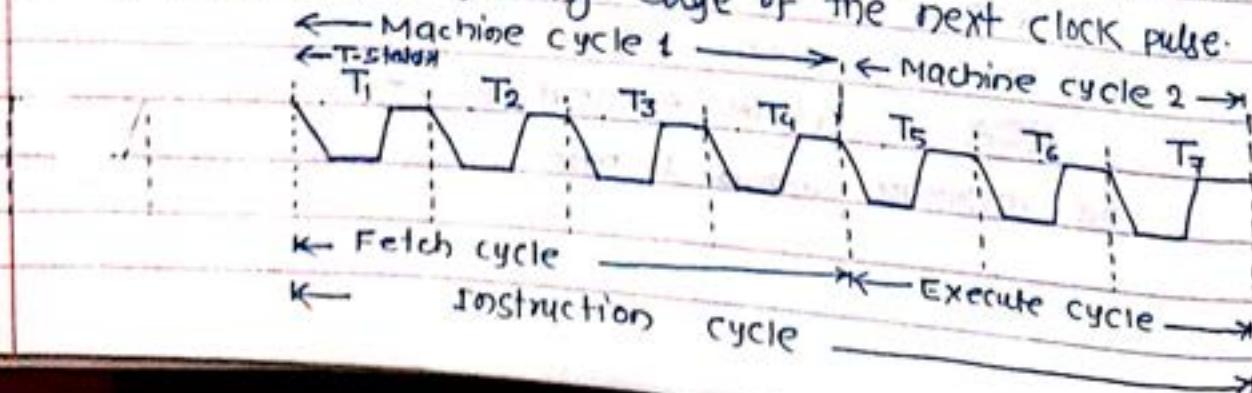
- Machine cycle is defined as the time required to complete one operation of accessing memory, I/O or acknowledging an external request.
- This cycle may consist 3 to 6 T-states.
- The basic op such as reading a byte from I/O port or writing a byte to memory is called machine cycle.
- The following are the various Machine cycle of 8085 CPU:-

Machine cycle	I _{O/M}	S ₁	S ₀	RD	WR	INTA	T states
Opcode Fetch	0	1	1	0	1	1	4-6
Memory Read	0	1	0	0	1	1	3
Memory Write	0	0	1	1	0	1	3
I/O Read	1	1	0	0	1	1	3
I/O Write	1	0	1	1	0	1	3
Interrupt Acknowledgement	1	1	1	1	1	0	6-12

T-States:

- T-States is defined as one subdivision performed in one clock period.

A T-state is measured from the falling edge of one clock pulse to the falling edge of the next clock pulse.



Timing Diagram:

: The necessary steps which are carried in a machine cycle can be represented graphically. Such graphical representation is called timing diagram.

Significance of Timing Diagram:-

- :- Timing diagram give us a perspective and help us understand the process of execution of a particular instruction in detail.
- :- We get to see what's going on inside the CPU in every clock cycle.
- :- It is an extremely minute level of examining the working of the CPU.
- :- It helps us visualize the whole process with respect to time.

Information for Timing Diagram:-

- :- Timing diagram is drawn for specific instruction, written at a specific location
eg: 2000H MOV A,B 78H
- :- We need to know following values to draw timing diagram
 - Address, where the instruction is written
Eg: 2000H : $A_{15}-A_8 = 20H$, $A_{7}-A_0 = 00H$
 - Opcode of the instruction
Eg: MOV A,B opcode is 78H, $D_7-D_0 = 78H$
 - No. of machine cycles required (in this case : 1 Machine cycle)
 - Total no. of T-states T-states (Adding T-state of all machine cycle : 4 T-states)

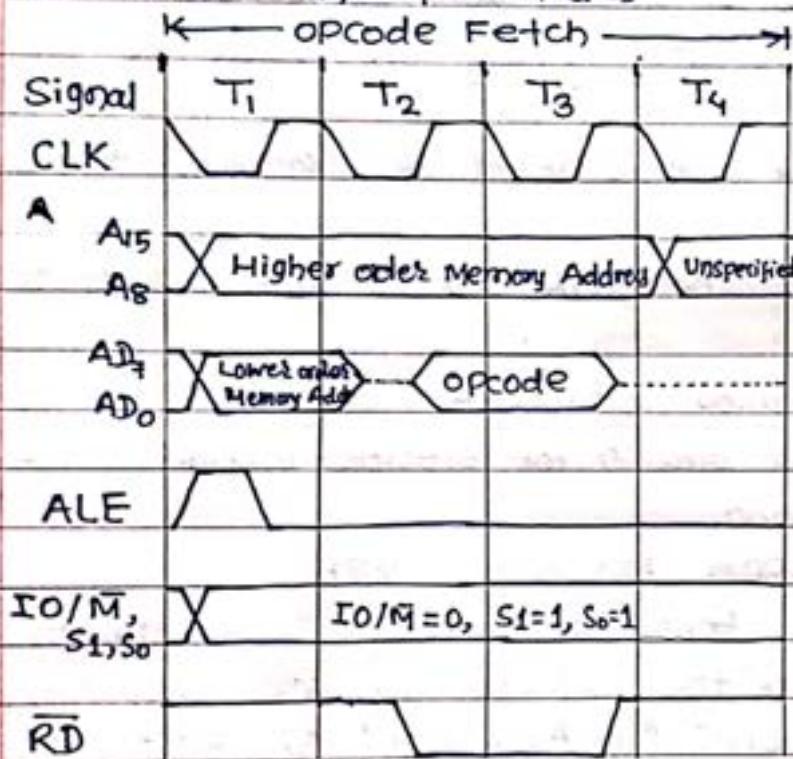
Steps to draw Timing Diagram:-

- :- Mark the T-states column wise
- :- Draw clock cycles equal to T-states & mark Machine cycle
- :- Draw higher order address bus ($A_{15}-A_8$) and mark its contents (20H)
- :- Draw ALE: It goes high for first clock cycle of every machine cycle
- :- Draw lower multiplexed Address/data bus ($A_{7}-A_0$). Mark lower order address (00H) whenever ALE is high and mark the data (78H) for next two clock cycles.

- $\overline{IO/M}$, \overline{RD} , \overline{WR} , S_0 & S_1 values are defined as per the machine cycle.

Opcode Fetch Machine cycle

- The CPU uses this cycle to take the opcode of an instruction from the memory location to processor.
- It consists of 4-T-states

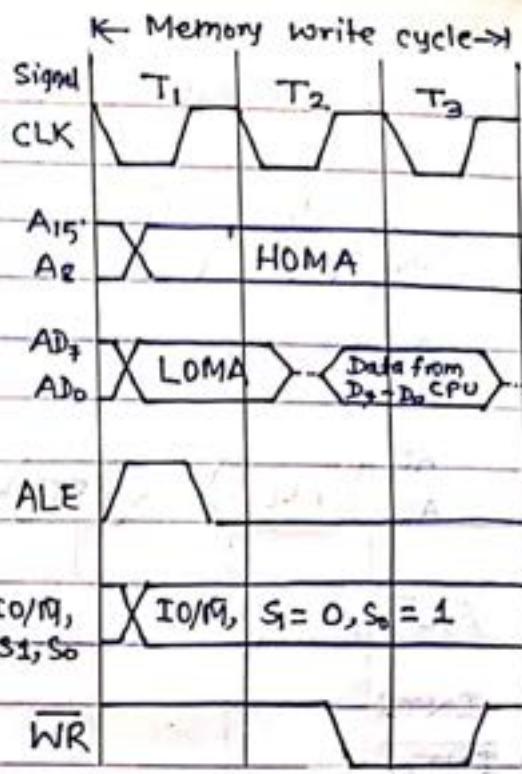
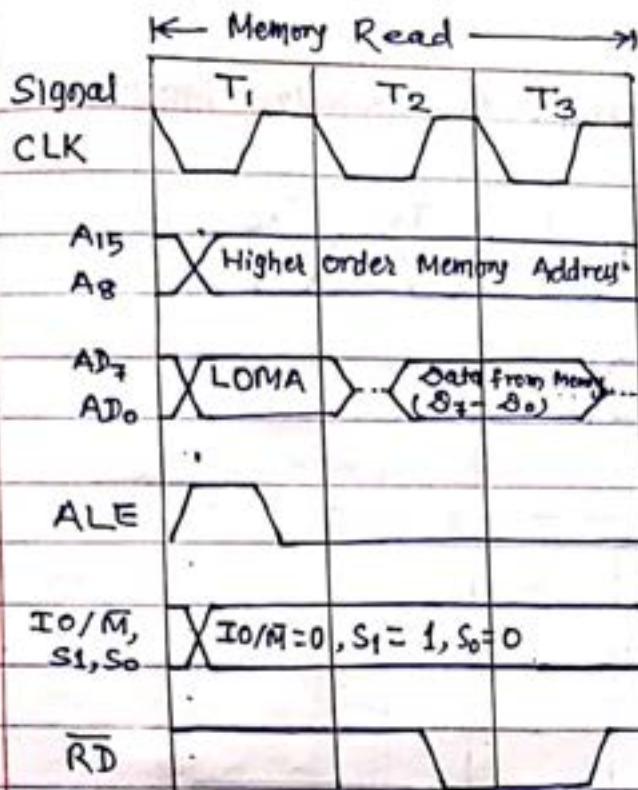


Memory Read cycle:

- The CPU executes these cycles to read data from memory
- It consists of 3-T-states

Memory Write cycle:

- The CPU executes these cycles to write data to memory
- It consists of 3-T-states



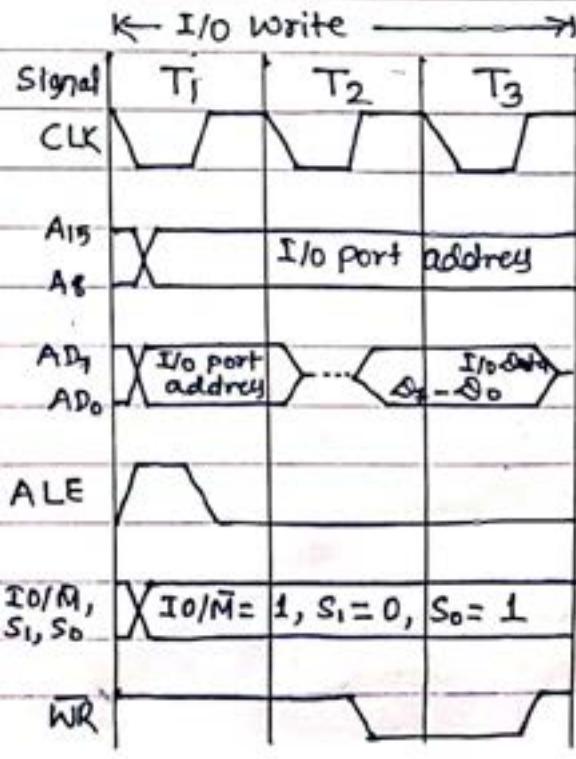
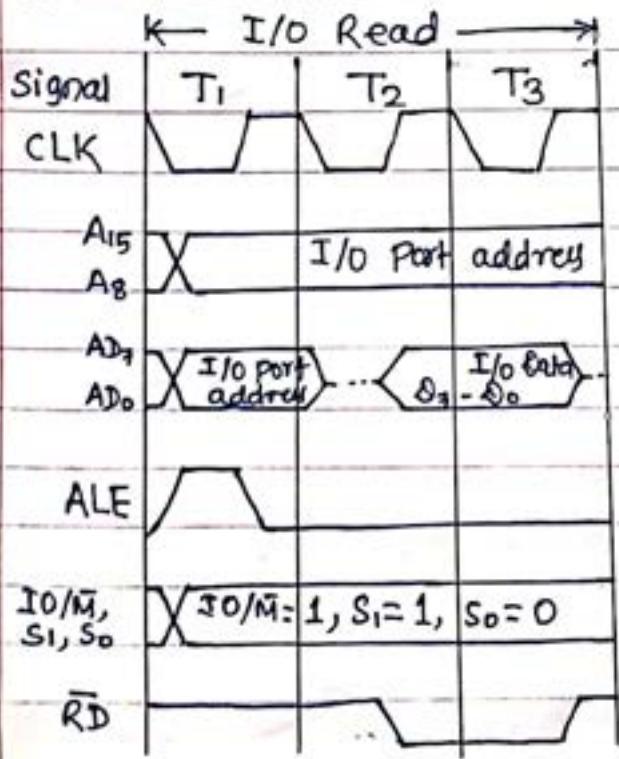
I/O Read and write cycle:

I/O Read Cycle:-

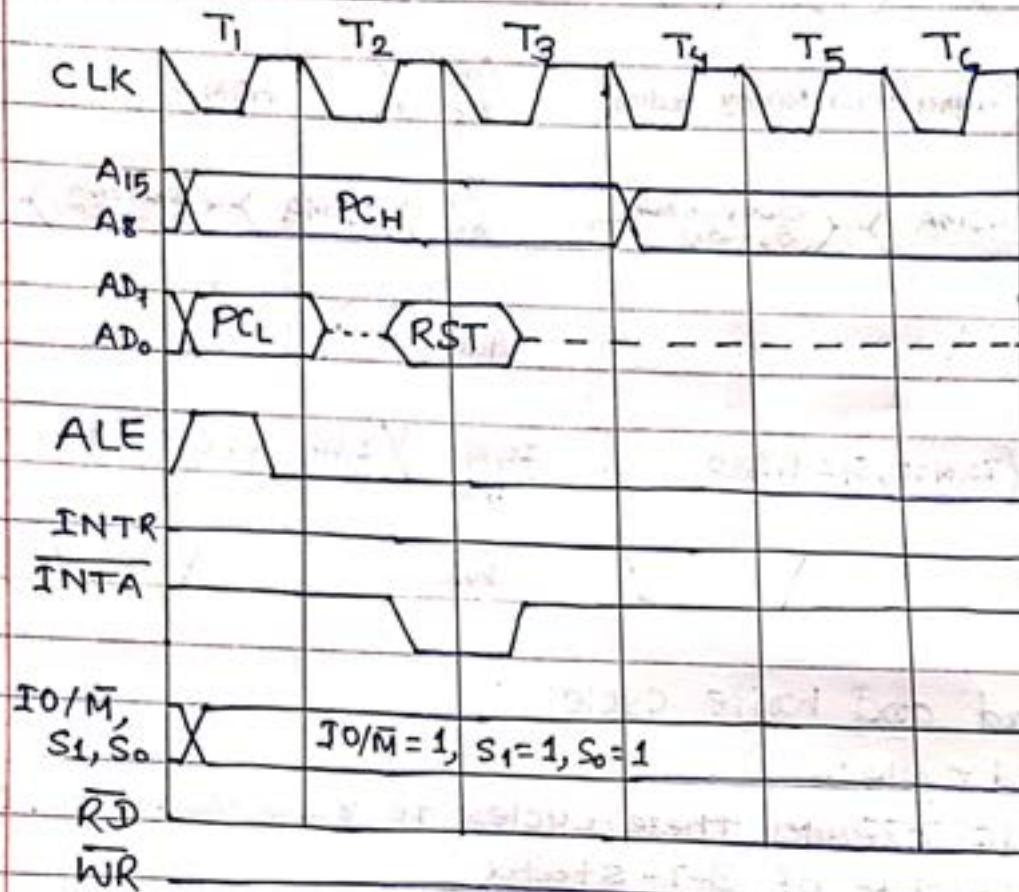
- The MP executes these cycles to read data from I/O devices
- gt consists of 3-T states

I/O write cycle:

- The MP executes these cycle to write data into I/O devices.
- gt consists of 3-T states



Timing Diagram of the Interrupt Acknowledge Machine cycle:



- RST instruction has only one interrupt acknowledge cycle of 6-T states.
- CALL instruction has three interrupt acknowledge cycles (First \rightarrow 6T states; Second and Third \rightarrow 3-T states)

Instruction	No. of Machine cycle	No. of T-states	Fetch, RD & WR
1. MOV A,B	1	4	1-F
2. ADD B	1	4	1F
3. SUB B	1	4	1F
4. INR B	1	4	F
5. ADC B	1	4	F
6. ANA B	1	4	F
7. CMA	1	4	F
8. CMC	1	4	F
9. CMP B	1	4	F
10. DAA	1	4	F
11. DCR B	1	4	F
12. RAL/RAR/RLC/RRC	1	4	F
13. SBB B	1	4	F
14. STC	1	4	F
15. XCHG	1	4	F
16. XRA B	1	4	F
17. ACI NOP	1	4	F
17. ACI 8-bit	2	7	FR
18. ADC M	2	7	FR
19. ADD M	2	7	FR
20. ADI 8-bit	2	7	FR
21. ANA M	2	7	FR
22. ANI 8-bit	2	7	FR
23. CMP M	2	7	FR
24. CPI 8-bit	2	7	FR
25. LDAX B	2	7	F R
26. MOV M,B	2	7	F W
27. MOV B,M	2	7	F R
28. MVI B, 8-bit	2	7	F R
29. ORA M	2	7	F R
30. ORI 8-bit	2	7	F R

31	SBB M	2	7	F R
32	SBI 8-bit	2	7	F R
33	STAX B	2	7	F W
34	SUB M	2	7	F R
35	SUI 8-bit	2	7	F R
36	XRA M	2	7	F R
37	XRI 8-bit	2	7	F R
38	DCR M	3	10	F R W
39	INR M	3	10	F R W
40	IN 8-bit port	3	10	F R R (I/O)
41	OUT 8-bit port	3	10	F R W (I/O)
42	JMP 16-bit	3	10	F R R
43	LXI B, 16-bit	3	10	F R R
44	MVI M, 8-bit	3	10	F R W
45	POP Rp	3	10	F R R
46	LDA 16-bit	4M	13	F R R R
47	STA 16-bit	4	13	F R R W
48				
49	LHLD	5	16	F R R R R
50	SHLD	5	16	F R R W W
51	XTHL	5	16	F R R W W
52	CALL	5	$6+4 \times 3 = 18$	INTA INTA INTA WW
53	RET	3	$6+3+3 = 12$	INTA R R
54	PUSH Rp	3	$6+3+3 = 12$	F W W
55	INX Rp	1	6	F
56	DCX Rp	1	6	F

57	SPHL	1	6	F
58	PCHL	1	6	F
59	DAD	3	10	F. Bus Idle Bus Idle
60	HLT	2	5	F. Bus Idle
61	RIM	1	4	F
62	SIM	1	4	F
63	EE	1	9	F
64	DL	1	4	F

S1 S0

- 0 0 (Halt)
- 0 1 (write operation)
- 1 0 (Read operation)
- 1 1 (either opcode or interrupt acknowledge)

I0/M

- 0 opcode Fetch / memory read / Memory write / Bus idle Machine cycle
- 1 Ioread / I/O write / Interrupt acknowledge machine cycle

Eg Draw the neat and labeled timing diagram for MVI A, 43 H.

Soln:-

Address Mnemonics Hexcode

2000H MVI A, 43H .06H

2001H 43H

- Fetching the opcode 06H from the memory 2000H (opcode Fetch Machine cycle)
- Read (Move) the data 43H from memory 2001H (Memory Read Machine cycle)

← Opcode Fetch Machine cycle → Memory Read Machine cycle

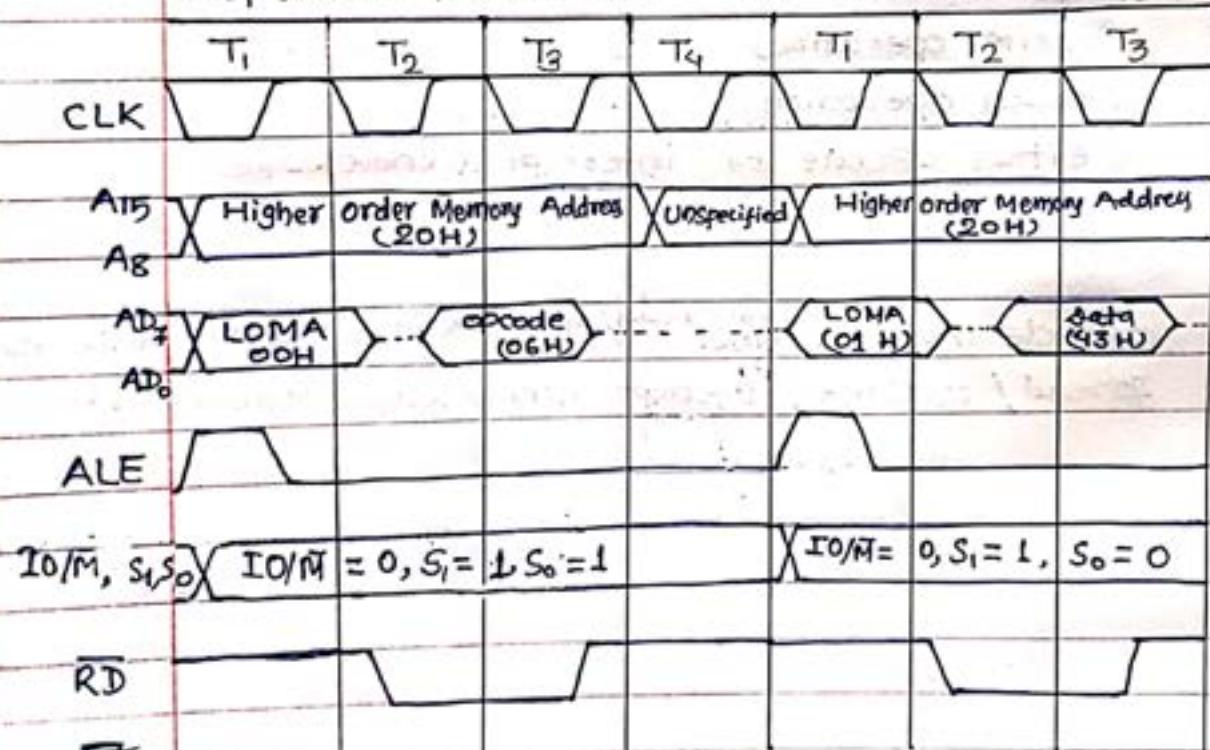


fig: Timing Diagram of MVI A, 43 H

LOMA = Low order Memory Address

Eg: Draw the neat and labeled timing diagram for MOV M, B
Soln:

Address	Mnemonics	opcode
2000H	MOV M, B	70H

Let

HL Register is pointing to memory location 4050H, and the memory location 4050H contains after execution will be data of Register B (assume AB 50H → AB H)

- Fetching the opcode 70H from the memory 2000H. (Opcode Fetch Machine cycle)
- Write (Move) the data contents of Register B (ABH) to the memory location pointed by HL register (4050H). (Memory Write Machine cycle)

← Opcode Fetch Machine cycle → Memory Write Machine cycle

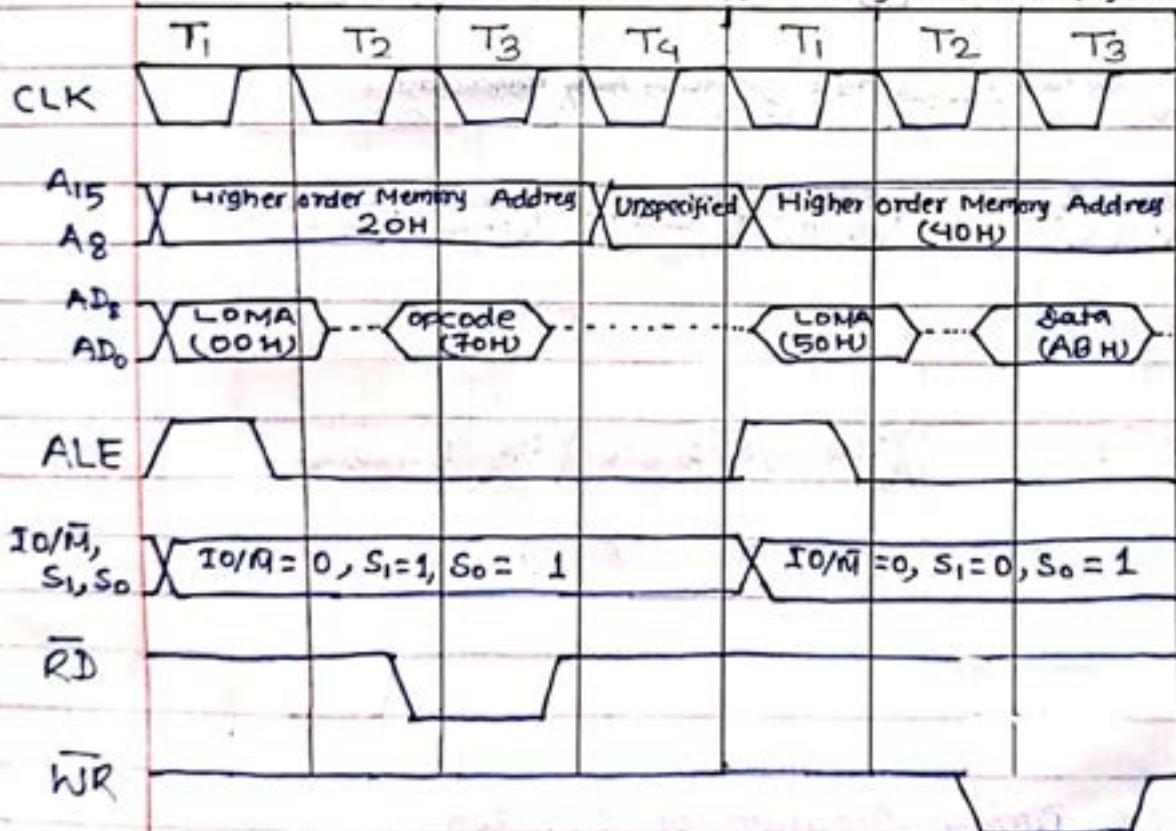


fig. Timing Diagram of MOV M, B

Eg: Draw the neat and labeled timing diagram for OUT 20H.

- 2050H: opcode of OUT (D3 H)
- 2051H: 20H

(a) opcode Fetch Machine cycle:

- Fetching the opcode D3 H from the memory 2050H

(b) Memory Read Machine cycle:

- Read the port address 20H from the memory 2051H.

(c) I/O Write Machine cycle:

- Transfer the content of accumulator to the output port 20H.

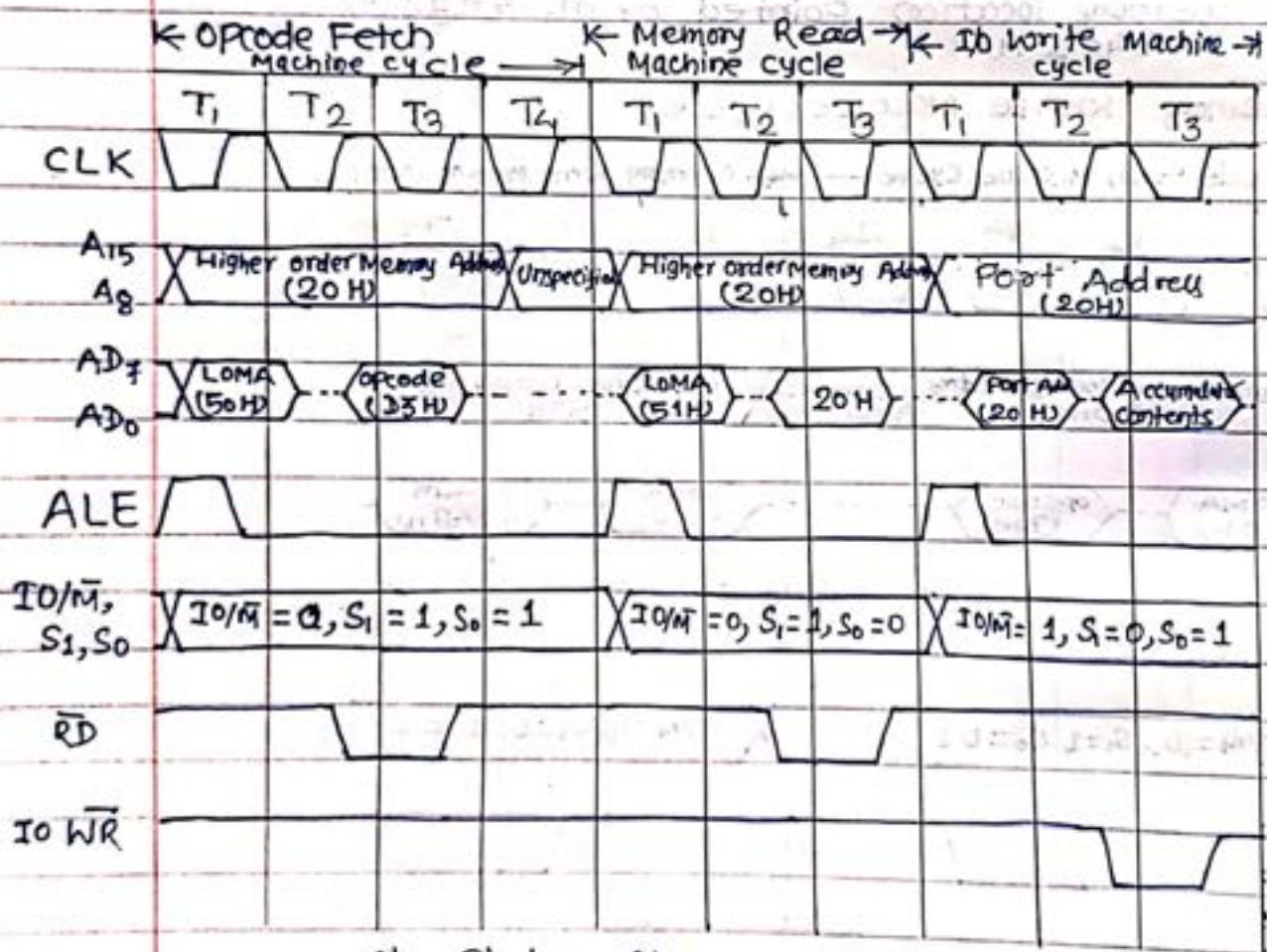


Fig. Timing Diagram of OUT 20H

Eg: Draw the neat and labeled timing diagram for STA 2050H

SOL:- The instruction copies the contents of the accumulator into the memory location 2050H. It is a 3-byte instruction and the CPU needs four machine cycles (13 T-states) to complete its execution.

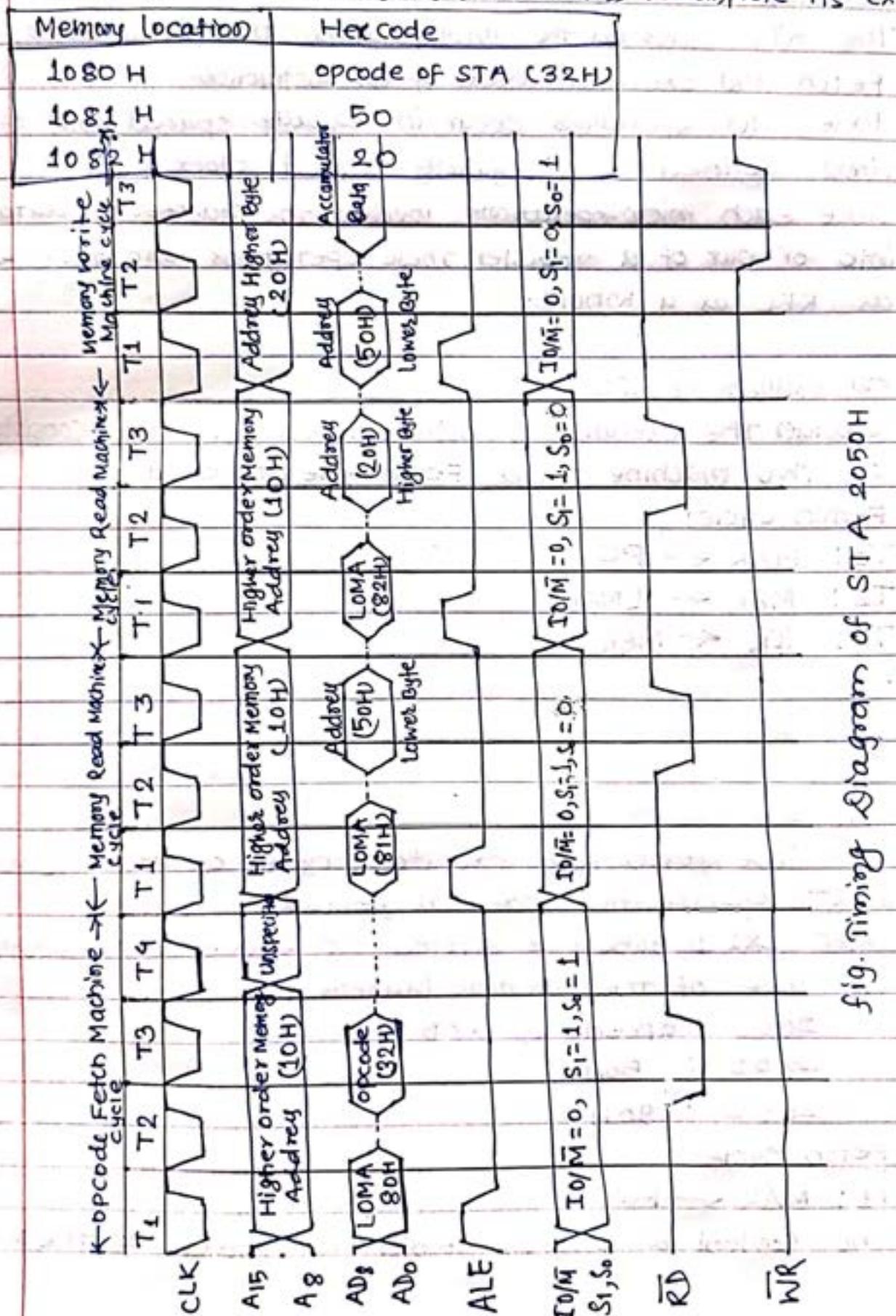


Fig. Timing Diagram of STA 2050H

2.7 RTL Instruction Descriptions:

- Register Transfer Language (RTL) is the symbolic representation of the steps that express the transfer of data among the registers.
- The RTL shows all the microoperations that occur during fetch and execution cycles of an instruction.
- These micro-operations occur in equally spaced units of time defined by a regularly spaced clock pulse.
- Since each micro-operation involves the transfer of data into or out of a register, these operations are termed as RTL as a whole.
- Microprocessor need to transfer data between register to perform specific task. The only way to understand about the transfer of data between register is RTL.
- This helps programmer to understand and have clear view about transfer of data between register in executing some instructions.
- For example: LXI D, 9050H

This instruction loads 16-bit data 9050H in register pair DE such that low order byte 50H is loaded into register E and the high order byte 90H is loaded into the register D.

The fetch and execution cycle of this instruction in RTL specification are as follows:

- Let LXI D, 9050H is stored at 2000H. So the corresponding hex code of the memory locations are:

2000 : opcode of LXID

2001 : 50H

2002 : 90H

Fetch cycle

T1: MAR \leftarrow PC

The content of program counter is moved to the MAR.

The PC contains the address of next instruction to be executed. MAR is 16 bit register that holds the address of memory temporarily during read / write operation. So now MAR has 2000H.

T2: MBR \leftarrow [MAR]

The content of memory pointed by MAR is moved to MBR. MBR is memory buffer register that holds data temporarily before writing in memory or after reading from the memory. So MBR will now have opcode of LXI D.

T3: IR \leftarrow MBR, PC \leftarrow PC + 1

The content of MBR i.e. opcode of LXI D is now sent to IR. The content of IR is decoded and PC incremented by 1.

T4: Unspecified

Execute cycle

M1: T1: MAR \leftarrow PC

The new content of PC i.e. 2001H is sent to MAR.

T2: MBR \leftarrow [MAR]

The content of location 2001H i.e. 50H is sent to MBR.

T3: E \leftarrow MBR, PC \leftarrow PC + 1

This lower order byte 50H is sent to register E and PC is incremented by 1.

M2: T1: MAR \leftarrow PC

The new content of PC i.e. 2002H is sent to MAR.

T2: MBR \leftarrow [MAR]

The content of location 2002H i.e. 90H is sent to MBR.

T3: D \leftarrow MBR

The higher order byte 90H is sent to register D.

Eg: STA 2000H using RTL Specification

This instruction copies the contents of the accumulator to the memory location 2000H. It is 3-byte instruction and requires 4 machine cycles (13T-states) for its execution.

The RTL for this instruction are shown below:

Fetch cycle

- T₁: MAR \leftarrow PC
- T₂: MBR \leftarrow [MAR]
- T₃: IR \leftarrow MBR, PC \leftarrow PC + 1
- T₄: Unspecified

Execute cycle

- M₁: T₁: MAR \leftarrow PC
 - T₂: MBR \leftarrow [MAR]
 - T₃: Z \leftarrow MBR, PC \leftarrow PC + 1
-
- M₂: T₁: MAR \leftarrow PC
 - T₂: MBR \leftarrow [MAR]
 - T₃: W \leftarrow MBR, PC \leftarrow PC + 1
-
- M₃: T₁: MAR \leftarrow WZ
 - T₂: MBR \leftarrow A
 - T₃: [MAR] \leftarrow MBR, PC \leftarrow PC + 1

Chapter-3 Bus Structure, Memory and I/O Interfacing

3.1 Bus structure of microprocessor:

- Bus is a group of conducting wires which carries information, all the peripherals or memory are connected to microprocessor through bus.
- The common bus used to carry data and address between microprocessor, i/o, memory is known as system bus.
- All the peripherals and memory share the same bus, however the microprocessor communicates with only peripherals at a time.

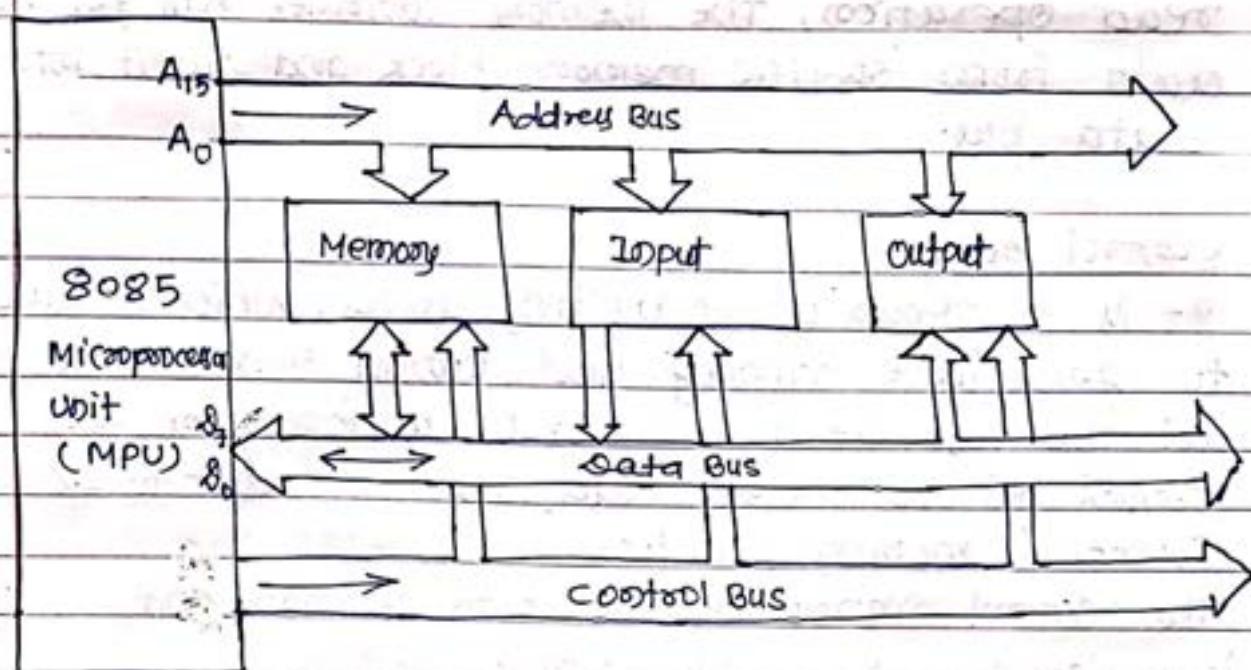


fig. Bus structure of 8085

There are three types of buses

(i) Address Bus:

- It is a group of conducting wires which carries address only.
- The address bus is a group of 16 lines generally identified as A₀ to A₁₅.
- It is unidirectional because data flow in one direction, from microprocessor to memory/peripherals.
- The 8085 MPU with its 16 address lines is capable of addressing $2^{16} = 65536$ or 64K memory locations.

Data Bus:

- : It is a group of wires which carries data only.
- : It is a group of 8 lines identified as S_0 to S_7 .
- : They are bidirectional i.e. data flow in both directions between microprocessor, memory and peripherals.
- : 8 data lines enable microprocessor to manipulate data ranging from $00H$ to FFH ($2^8 = 256$ numbers).
- : When it is write operation, the processor will put the data (to be written) on the data bus, when it is read operation, the memory controller will get the data from specific memory block and put it into the data bus.

Control Bus:

- : It is a group of conducting wires, which is used to generate timing and control signals to control all the associated peripherals, microprocessor uses control bus to process data, that is what to do with Selected memory location.
- : The control signals transmit both command and timing information between the system modules.
- : The timing signals indicate the validity of data and address information while the command signals specify the operation to be performed.
- : Some of the control signals are

ALE (Address Latch Enable)

IO/M

\overline{RD}

\overline{WR}

S_1, S_0 etc.

Synchronous and Asynchronous bus:

Synchronous Bus:

- In Synchronous buses, the steps of data transfer take place at fixed clock cycles.
- Synchronous buses are typically faster than asynchronous buses.
- Used in high speed transmission.

Asynchronous Bus:

- There is no fixed clock cycles in asynchronous buses.
- Asynchronous buses are slower than synchronous buses.
- Used in low speed transmission.

Advantages

- Involves very little logic and can run very fast
-

Advantages

- It is appropriate for different speed devices.
- It is reliable for long transmission and higher number of devices.

Disadvantage

- It is not reliable for long transmission
- Every device on the bus run at the same clock rate.

Disadvantage

- Slower

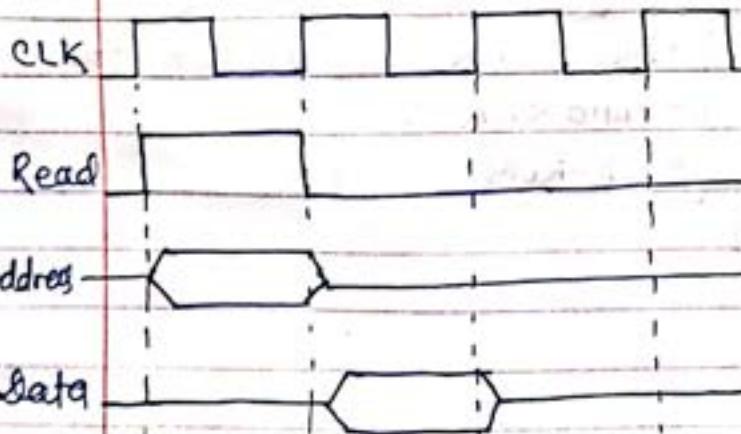


fig: Synchronous Read operation

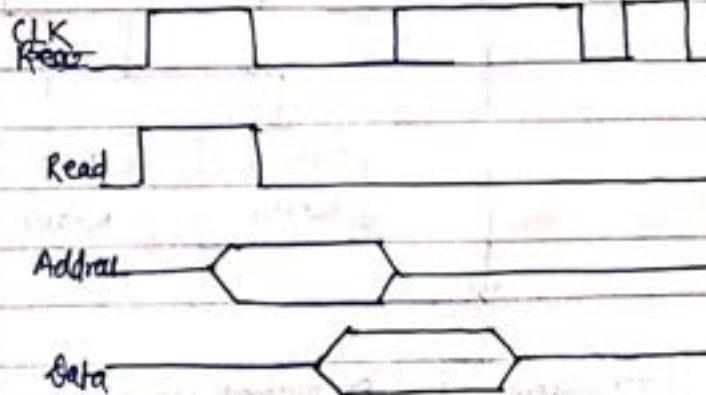


fig: Asynchronous Read operation

3.2 Memory Interfacing:

- Interface: It is a path of communication between two components.
- Interfacing is a technique of connecting microprocessor with memory.
- When MP is executing an instruction, it needs to access the memory for reading instruction codes and the data.
- for interfacing, both the memory and the microprocessor require some signals to read from and write to registers.
- Interfacing process needs to match the memory requirements and MP signals.

Types of Memory:

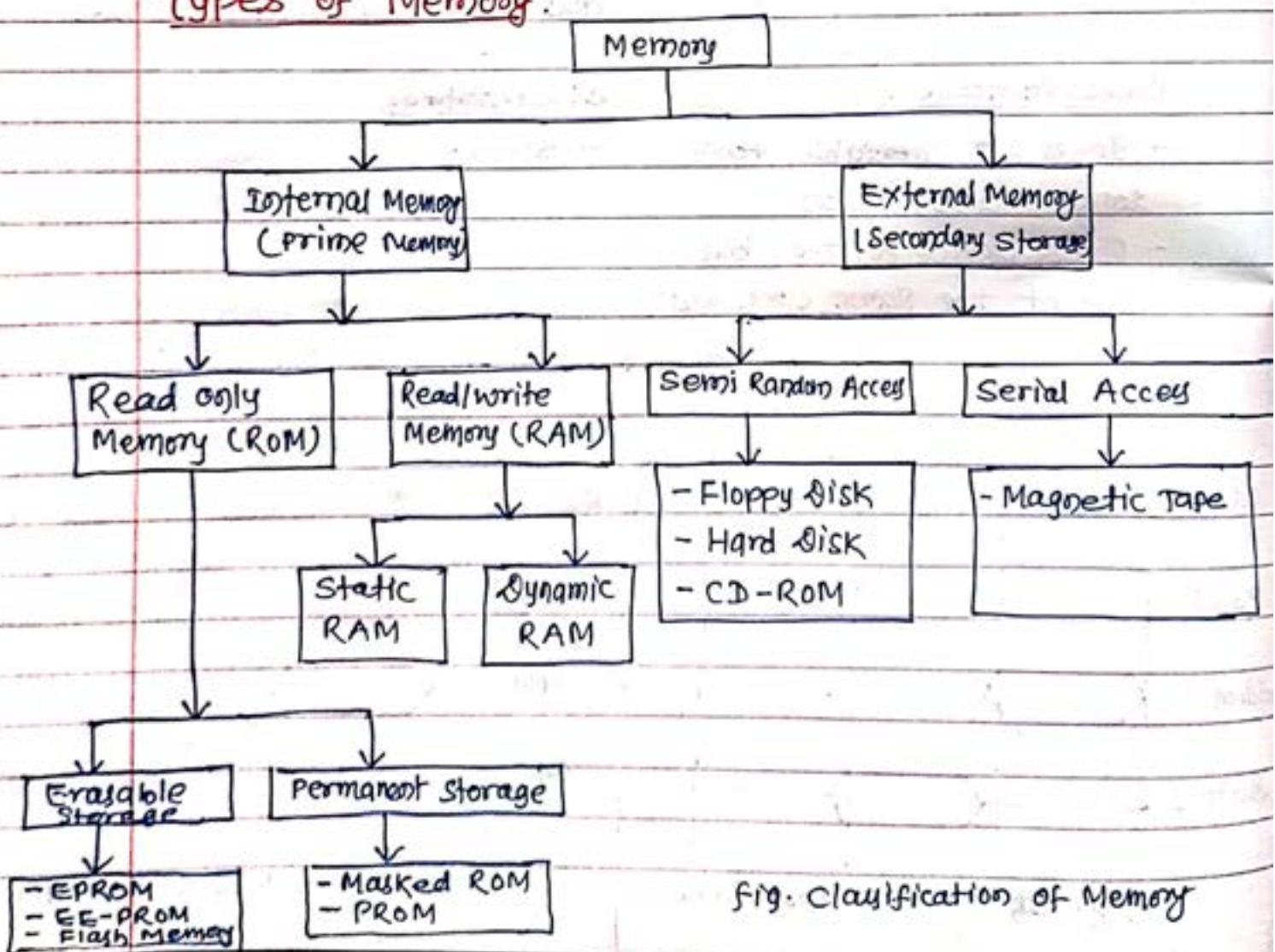


fig. Classification of Memory

1. Internal Memory (Prime Memory)

- A microprocessor needs internal memory ~~and~~ for executing and storing programs.
- It can be directly accessed by up with a random access. Due to this such memories are able to respond fast enough to synchronize with the execution speed of the up.
- ~~It is volatile, i.e., it loses its information when power is off.~~
- The primary memory is further divided into two parts:
 - (a) Read Only Memory (ROM)
 - (b) Random Access Memory (RAM)

(a) ROM

- It is a read only memory. We can't write data in this memory.
- It is a memory device or storage medium that is used to permanently store information inside a chip.
- It is non-volatile memory i.e. the data remains intact even if power is turned off.
- There are five types of ROM
 - (i) Masked ROM
- The program or data are permanently installed at the time of manufacturing as per requirement. The data can not be altered.
- The process of permanent recording is expensive but economic for large quantities.
-

(ii) PROM (Programmable ROM)

- It is a type of digital ROM, in which we can write any type of information or program only once.
- PROMs can be programmed by the user with a special PROM programmer.

(iii) EPROM (Erasable programmable ROM)

- EPROM can be programmed (i.e. information stored) by the user with a special EPROM programmer.
- The stored information can be erased by exposing the chip to ultra violet rays through its quartz window for 15 to 20 minutes. However it is not possible to erase selective information; when erased the entire information is lost. Then the chip can be reprogrammed.
- EPROMs are suitable for product development, experimental projects, college laboratories since they can be used several times.

(iv) EEPROM (Electrically Erasable programmable ROM)

- This is similar to EPROM except that the erasing is done by electrical signals instead of ultraviolet light.
- The main advantage is the memory location can be selectively erased and reprogrammed. But the manufacturing process is complex and expensive so do not commonly used.

(v) Flash Memory

- Flash memory is an non volatile memory storage medium that can be electrically erased and reprogrammed.
- The memory chips can be erased and programmed at least a million times.
- It is also used to transfer data between the computer and digital devices.

(b) RAM

- Microprocessor can read from and write into this memory.
- It is used primarily for information that are frequently altered, such as writing programs or receiving data.
- It is volatile memory. It can not hold data when the power is turned off.

There are two types of RAM:

(i) SRAM (Static RAM)

- This memory is made up of flip flops and it stores bit as voltage. A single flip flop stores binary data either 1 or 0.
- Each flip flop is called storage cell.
- It is a type of RAM used to store static data in the memory. It means to store data in SRAM remains active as long as the Computer System has a power supply. However, data is lost ~~data~~ in SRAM when power failure have occurred.

(ii) DRAM (Dynamic RAM)

- It is type of RAM that is used for the dynamic storage of data in RAM. ~~It~~
- In DRAM each cell carries one bit information.
- The cell is made up of two parts: a capacitor and a transistor.
- It has high density and lower power consumption and is cheaper than SRAM.
- When data is 1, the capacitor will be charged and if data is 0, the capacitor will not be charged.
- Because of capacitor leakage currents, the data will not be held by these cells. So the DRAMs require refreshing of memory cells.

Q2) Secondary Memory (secondary storage)

- :- The devices that provides backup storage are called Secondary memory.
- :- Secondary storage include the slow storage devices like CD-ROM, magnetic tape, floppy etc.
- :- Such device are used to store large size files and programmes such as compilers and database management systems that are not needed by the processor frequently.
- :- Secondary storage devices are high capacity, slow access and low cost.
- :- Types of Secondary Memory
 - (a) Hard disk
 - (b) Floppy disk
 - (c) CD (compact disk)
 - (d) pen drive
 - (e) Magnetic tape

Eg:- Design the address decoding circuit for two RAM blocks of 256 bytes each consequently at address 5600H.

SOL:

Address Decoding circuit

For RAM 1

$$\text{Base Address} = 5600H$$

$$\text{End Address} = \text{Base Address} + (\text{Number of locations in 256B RAM} - 1)$$

$$= 5600H + (256 - 1)D$$

$$= 5600H + 255D$$

$$= 5600H + 00FF H$$

$$= 56FF H$$

$$\text{No. of address line (CN)} = \frac{\log(256)}{\log(2)} = 8$$

i.e. A₇ ... A₀ are connected to internal decoder of RAM 1.

For RAM 2

$$\text{Base Address} = \text{End Address of RAM 1} + 0001H$$

$$= 56FF H + 0001H$$

$$= 5700H$$

$$\text{End Address} = 5700H + (256 - 1)D$$

$$= 5700H + 255D$$

$$= 5700H + 00FF H$$

$$= 57FF H$$

$$\text{No. of address line} = 8$$

i.e. A₇ ... A₀ are connected to internal decoder of RAM 2.

Hence,

RAM 1: A₁₅ A₁₄ A₁₃ A₁₂ A₁₁ A₁₀ A₉ A₈ A₇ A₆ A₅ A₄ A₃ A₂ A₁

Base Address = 5600H 0 1 0 1 0 1 1 0 0 0 0 0 0 0 0 0

End Address = 56FFH 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1

RAM 2:

Base Address = 5700H 0 1 0 1 0 1 1 1 0 0 0 0 0 0 0 0

End Address = 57FFH 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1

↓
To enable the external
decoder

↓
External
decoder

↓
TO internal decoders
of RAM

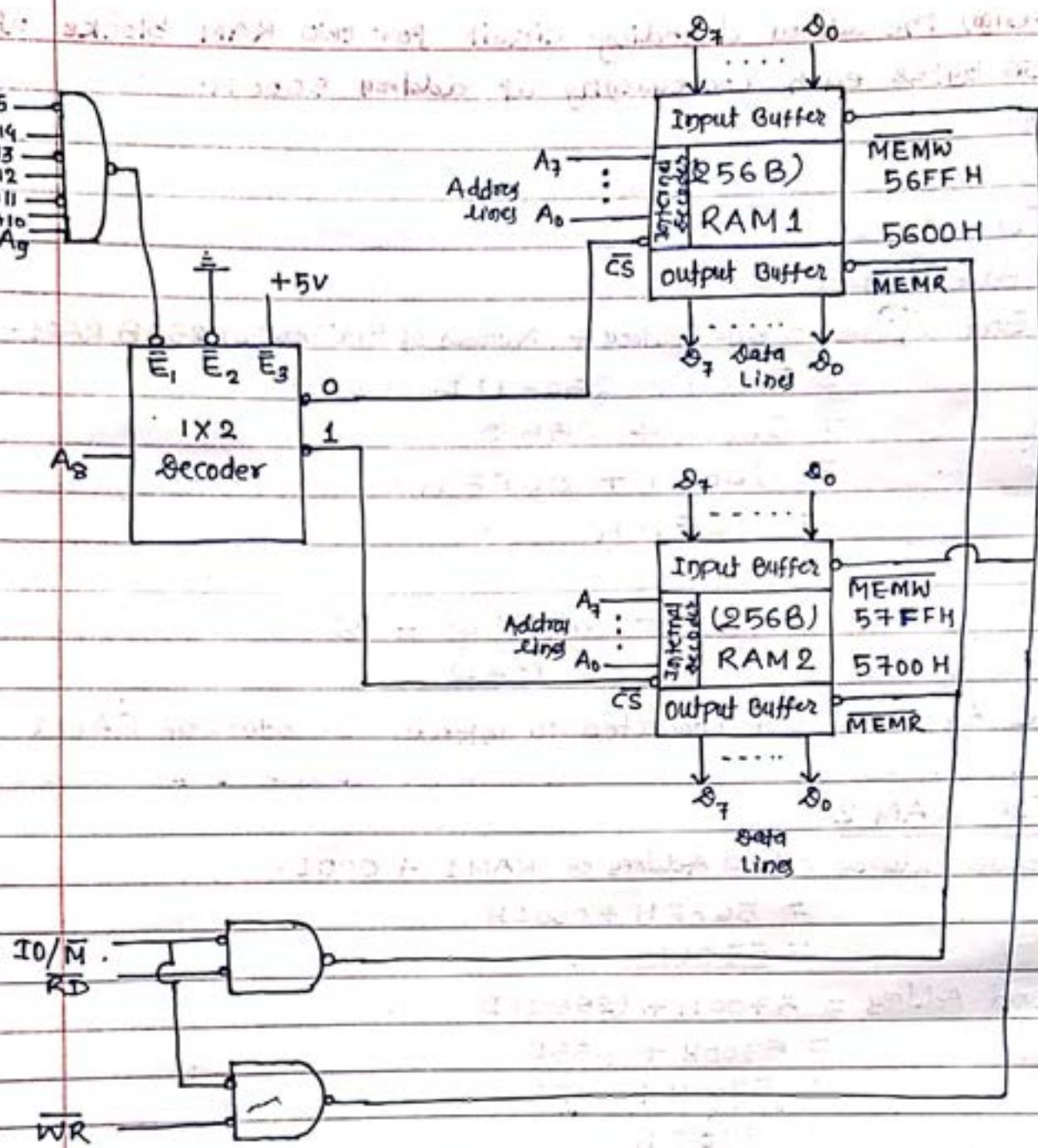


fig. Address decoding circuit.

Eg: design an interfacing circuit to interface one 4KB EPROM and two 2KB R/W memory for 8085 microprocessor. Using 3-to-8 decoder
Soln:

since no specific starting address is given, let it be 8000H

FOR EPROM

$$\text{Base Address} = 8000H$$

$$\text{End Address} = \text{Base Address} + (\text{Number of locations in 4KB EPROM} - 1)$$

$$= 8000H + (4 \times 1024 - 1)D = 8000H + 4095D$$

$$= 8000H + 0FFFH = 8FFFFH$$

Number of Address line = $\frac{\log(4 \times 1024)}{\log(2)} = 12$

i.e. A₁₁...A₀ are connected to internal decoder of EPROM

For RAM1

$$\text{Base Address} = \text{End Address of EPROM} + 0001H$$

$$= 8FFFFH + 0001H = 9000H$$

$$\text{End Address} = 9000H + (2 \times 1024 - 1)D = 9000H + 2047D$$

$$= 9000H + 07FFH = 97FFFH$$

No. of Address line = $\frac{\log(2 \times 1024)}{\log(2)} = 11$

i.e. A₁₀...A₀ are connected to internal decoder of RAM1

For RAM2

$$\text{Base Address} = \text{End Address of RAM1} + 0001H$$

$$= 97FFFH + 0001H = 9800H$$

$$\text{End Address} = 9800H + (2 \times 1024 - 1)D = 9800H + 2047D$$

$$= 9800H + 07FFH = 9FFFH$$

No. of Address line = 11

i.e. A₁₀...A₀ are connected to internal decoder of RAM2

Memory Mapping Table

EPROM	A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	A ₁₀	A ₉	A ₈	A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀
-------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

Base Address = 8000H 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

End Address = 8FFFFH 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1

RAM1

Base Address = 9000H	1 0 0 1	0 0 0 0	0 0 0 0	0 0 0 0
----------------------	---------	---------	---------	---------

End Address = 97FFFH	1 0 0 1	0 1 1 1	1 1 1 1	1 1 1 1
----------------------	---------	---------	---------	---------

RAM2

Base Address = 9800H	1 0 0 1	1 0 0 0	0 0 0 0	0 0 0 0
----------------------	---------	---------	---------	---------

End Address = 9FFFH	1 0 0 1	1 1 1 1	1 1 1 1	1 1 1 1
---------------------	---------	---------	---------	---------

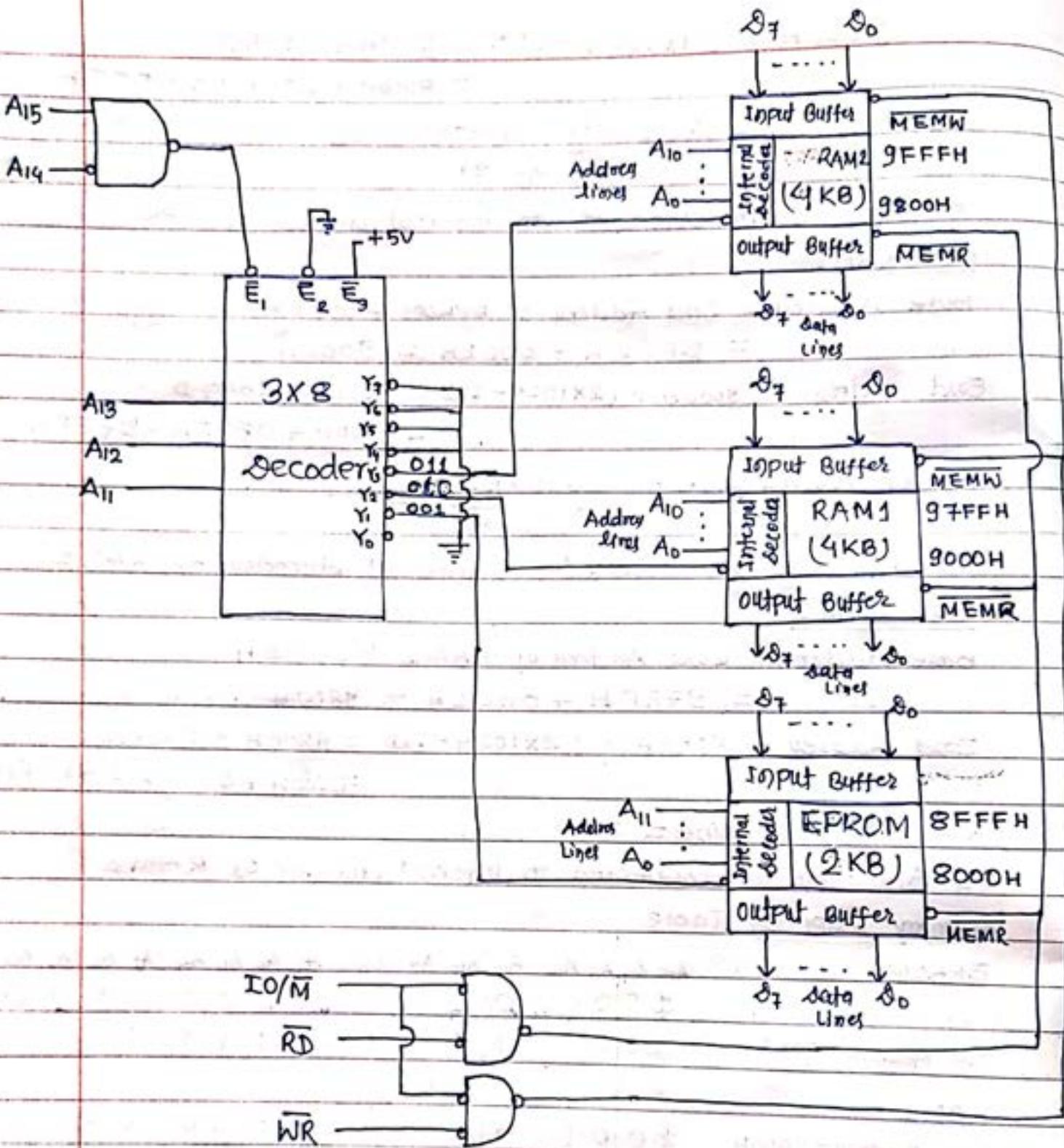


fig. Address Decoding circuit

eg. Design address decoding circuit for interfacing two ROM chips each of 256×8 at $5300H$
So?

Step 1 calculate address range and number of address lines

For ROM 1

$$\text{Base address} = 5300H$$

$$\begin{aligned}\text{End address} &= \text{Base Address} + (\text{Number of locations in } 256B \text{ ROM}-1) \\ &= 5300H + (256-1)D \\ &= 5300H + 255D = 5300H + 00FFH \\ &= 53FFH\end{aligned}$$

$$\begin{aligned}\text{For } 256 \text{ Number of Address lines} &= \frac{\log(\text{No. of locations in B})}{\log(2)} \\ &= \frac{\log(256)}{\log(2)} \\ &= 8\end{aligned}$$

i.e. $A_7 \dots A_0$ are connected to the internal decoder of ROM 1

For ROM 2

$$\begin{aligned}\text{Base Address} &= \text{End Address of ROM 1} + 0001H \\ &= 53FFH + 0001H = 5400H\end{aligned}$$

$$\begin{aligned}\text{End Address} &= 5400H + (256-1)D \\ &= 5400H + 255D = 5400H + 00FFH \\ &= 54FFH\end{aligned}$$

Also, $N=8$

i.e. $A_7 \dots A_0$ are connected to the internal decoder of ROM 2

Step 2 Memory Mapping Table

ROM 1

	$A_{15} A_4 A_3 A_2 A_1 A_0$	$A_{11} A_{10} A_9 A_8$	$A_7 A_6 A_5 A_4$	$A_3 A_2 A_1 A_0$
Base Address	$5300H$	0101	0011	10000000000000
End Address	$53FFH$	0101	010111111111	1111

ROM 2

Base Address	$5400H$	0101	0100	10000000000000	0000
End Address	$54FFH$	0101	0100	111111111111	1111

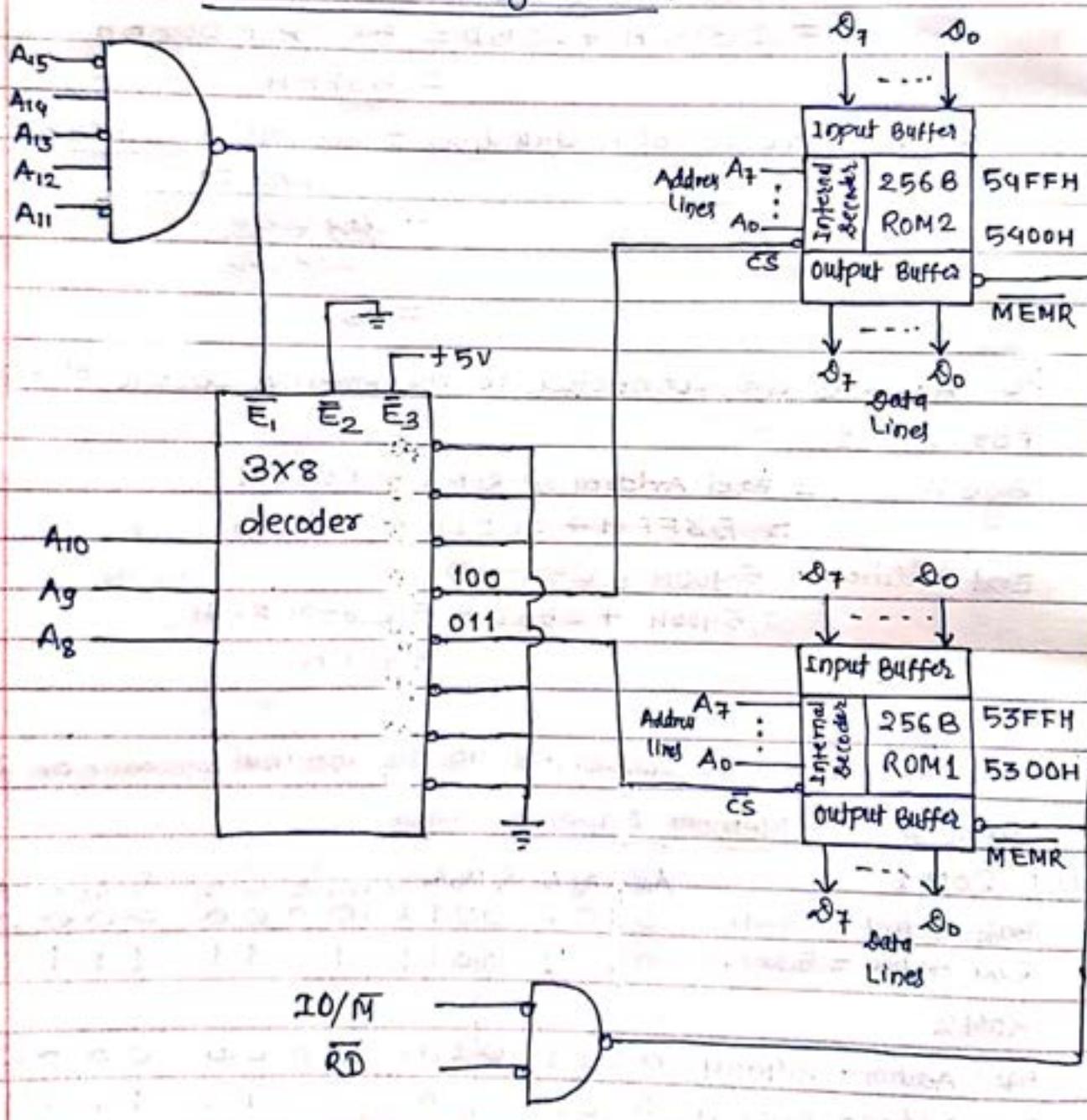
\downarrow To enable the External decoder \downarrow To External decoder \downarrow To Internal Decoder of ROM

Step 3 Decide decoder pins

Here, Bits A₈, A₉ and A₁₀ address lines are different for two chips referring to start or end address lines. We need to use a 3x8 decoder.

We can use address lines A₁₁, A₁₂, A₁₃, A₁₄, A₁₅ to generate chip enable signals for 3x8 decoder.

Step 4 Draw a decoding circuit



Address Decoding circuit

$$\# 1KB = 1024 B = 1024 \text{ bytes} = 1024 \times 8$$

eg 4096×8
 $= 4 \times 1024 \times 8$
 $\therefore = 4KB$

size of decoder = $2^n \times 2^n$

Address Decoding:

- The processor communicates with all the parts interconnected in the system through a common address and data bus. As a result of this, only one device can transmit data at a time and others can only receive that data.
- If more than one device attempt to send data through the bus at the same time, the proper communication among the devices does not become possible because the data sent by them gets garbled. To avoid this situation, insuring that the proper device gets addressed at proper time, the technique called address decoding is used.
- In address decoding method, all devices like memory blocks, I/O units, etc are assigned with a specific address. The address of this device is determined from the way in which the address lines are used to derive a special device selection signal known as chip select (CS). If the microprocessor has to write to or read from a device, the CS signal to that block should be enabled and the address decoding circuit must ensure that CS signal to other devices are not activated.
- Depending upon the no. of address lines used to generate chip select signal for the device, there are two methods of address decoding. They are
 - (i) I/O Mapped I/O
 - (ii) Memory Mapped I/O

Memory Mapped I/O

- I/O devices are identified and addressed with 16-bit address i.e. A₀ to A₁₅ lines are used to generate device address.
- The devices are enabled by memory related control signals.
- Data bytes are transferred using the instructions related to memory access like STA address, LDA address, LDAX R#, ADD M, CMP M, MOV R, M etc.
- The total addressing capability of the 8085 microprocessor in this method is $2^{16} = 65536$ bytes = 64 KB
- Decoding 16-bit address may require more hardware.
- Memories like RAM, ROM, EPROM etc use memory mapped I/O.
- MEMR and MEMW control signals are used to control read and write I/O operations.
- Data transfer between any register and I/O devices.

I/O Mapped I/O

- I/O devices are identified and addressed with 8-bit address. So, A₀ to A₇ or A₈ to A₁₅ lines are used to generate device address.
- The devices are enabled by I/O related control signals.
- Data bytes are transferred using IN/OUT instructions.
- The total addressing capability of the 8085 microprocessor in this method is $2^8 = 256$ bytes.
- Decoding 8-bit address will require less hardware.
- Programmable peripherals device like 8255 A, I/O latches etc use I/O mapped I/O.
- IOR and IOW control signals are used to control read and write I/O operations.
- Data transfer between I/O devices and accumulator only.

Address Decoding are categorized in the following two groups:-

(Non unique)
Partial Address Decoding

- If all of the address lines available on that mode are not used in address decoding, then that decoding is called partial (nonunique) address decoding.

- This technique of address decoding is simple but there may be a chance of address conflict.

- Non-unique decoding technique is used in small memory systems and it decreases the cost of decoding circuit as it simply eliminates the unnecessary hardware by using only the required number of address lines (not all).

- Difficult for future expansion

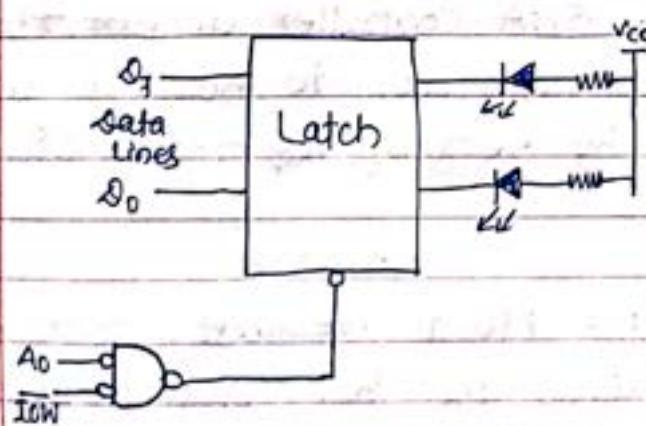


fig. Non Unique Address decoding

(Unique)
Complete Address Decoding

- If all the address lines are available on that mapping mode are used for address decoding, then that decoding is called unique address decoding (complete).

- This address decoding technique is complicated but is the best for all cases.

- Unique address decoding is used in large memory system. However, it increases the cost of decoding circuit.

- Easy for future Expansion

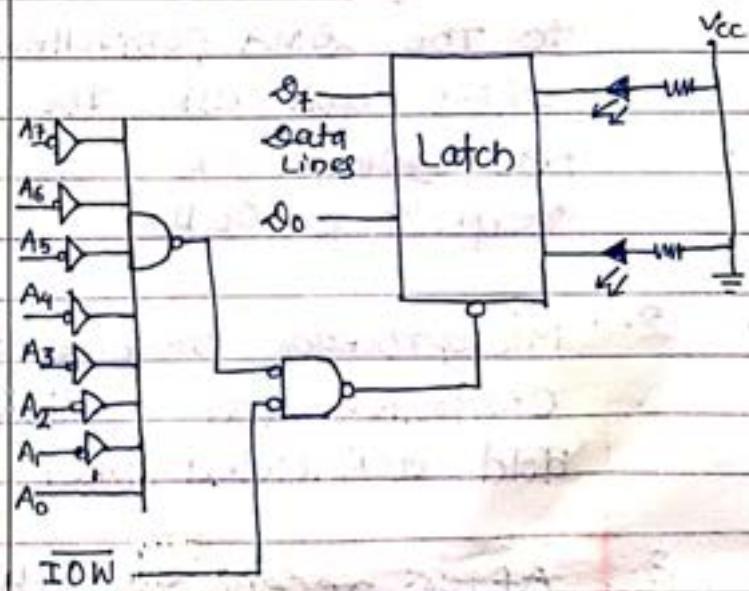


fig. Unique Address decoding

3.3. I/O Interfacing:

DMA (Direct Memory Access)

- DMA stands for direct memory access.
- It is designed by Intel to transfer data at fastest rate.
- This is a process where data is transferred between two peripherals directly without the involvement of the microprocessor.
- Basic DMA operation:

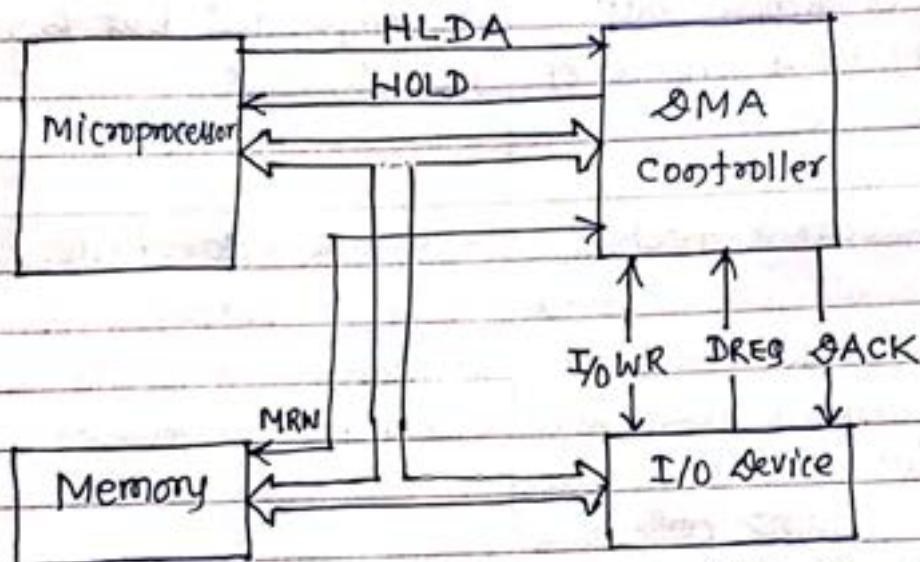


fig. DMA controller data Transfer

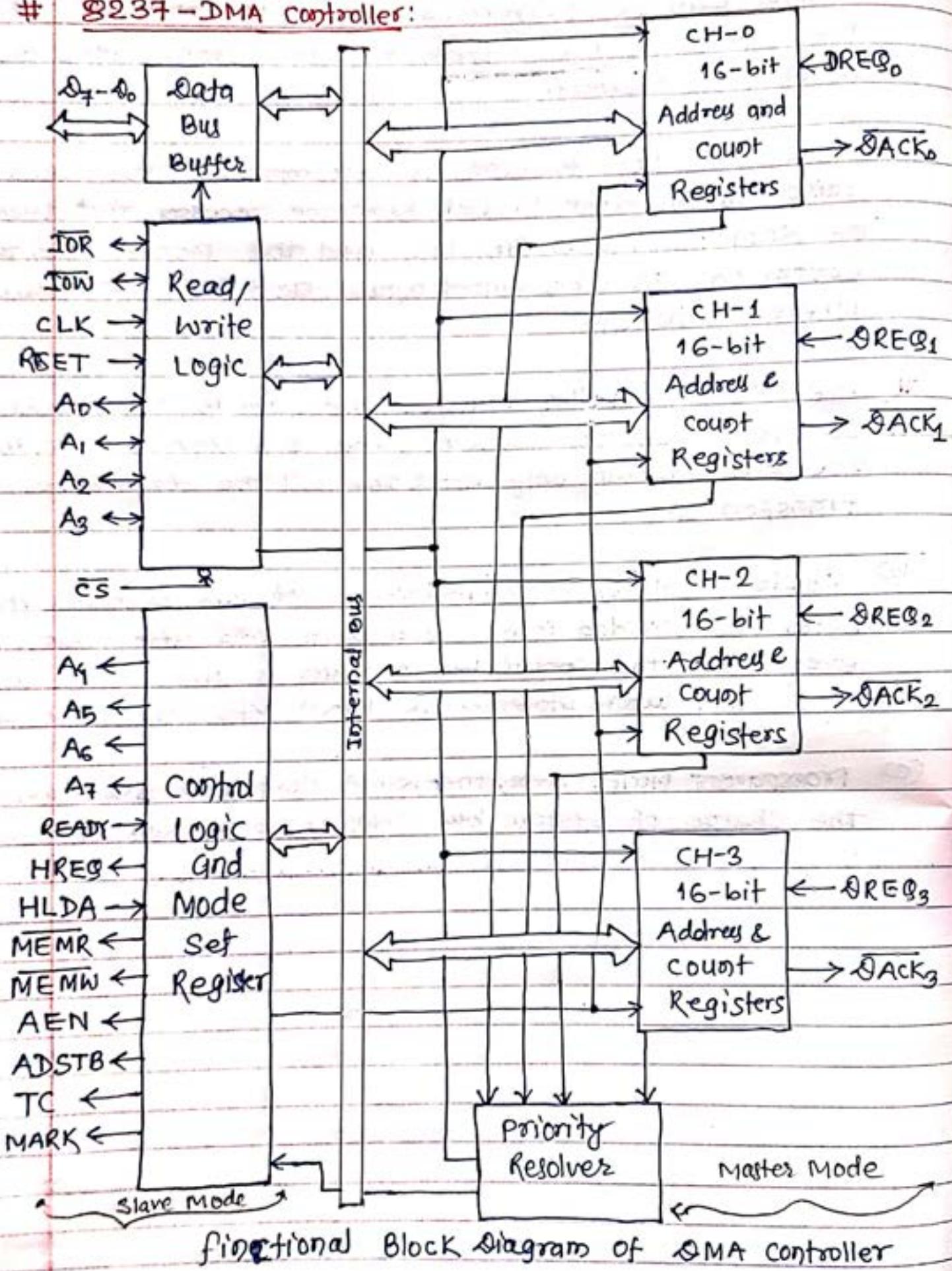
1. whenever an I/O device wants to transfer the data to or from memory, it sends DMA request (DREQ) to the DMA controller. DMA controller accept this DREQ and asks the microprocessor to hold for a few cycle clock cycles by sending it the Hold request (HOLD).
2. Microprocessor receives the HOLD request from DMA controller and relinquishes the bus and sends the Hold acknowledgement (HLDA) to DMA controller.
3. After receiving the HLDA, DMA controller acknowledges I/O device (DACK) that the data

transfer can be performed and DMA controller takes the charge of the system bus and transfer the data to or from memory.

4. When the data transfer is accomplished, the DMA raise an interrupt to let know the processor that task of data transfer is finished and the processor can take control over the bus signal again and start processing where it has left.

- # The DMA controller transfers the data in three modes:-
 - (a) Burst Mode:- In this scheme, the I/O device withdraws the DMA request only after all the data bytes transferred.
 - (b) Cycle Stealing technique (Mode):- In this scheme, the bytes are divided into several parts and after transferring every part the control bus of buses is given back to MPU and later stolen back when MPU does not need it.
 - (c) Transparent Mode:- Here, the DMA controller ~~asks~~ takes the charge of system bus only if the processor does not require the system bus.

8237-DMA controller:



gt is a device to transfer the data directly between I/O device and memory without through the CPU. So it performs a high speed data transfer between memory and I/O device.

The features of 8237 are:-

- The 8237 has four channels and so it can be used to provide DMA to four I/O devices.
- Each channel can be independently programmable to transfer upto 64 K bytes of data by DMA.
- Each channel can independently perform read transfer, write transfer and verify register.

The functional blocks of 8237 as shown in above figure are

- (i) Data bus buffer :- gt contains tristate, 8-bit bidirectional buffers.
 - Slave Mode: it transfers data between microprocessor and internal data bus
 - Master Mode: the o/p A8-A15 bits of memory address on data lines (unidirectional)
- (ii) Read/write logic :- gt controls all internal Read/Write operation.
 - Slave mode: gt accepts address bits and control signal from microprocessor.
 - Master Mode: gt generates address bits and control signals.
- (iii) Control logic & Mode set Registers:-
 - gt generates address and control signals
 - Master Mode: gt controls the sequence of DMA operation during all DMA cycles.
 - Slave Mode: it is disabled
- (iv) Priority Resolver: priority of each of the 4 channels can be set in fixed priority and rotating priority.

Operation of 8237 DMA controller:-

- Each channel of 8237 has two programmable 16-bit registers named as address register and count register.
- Address register is used to store the starting address of memory location for DMA data transfer.
- The address in the address register is automatically incremented by after every read/write/verify transfer.
- The count register is count the number of byte or word transferred by DMA.
- In read transfer the data is transferred from the number of bytes memory to I/O device.
- In write transfer the data is transferred from I/O device to memory.
- Verification operations generates the DMA addresses without generating the DMA memory and I/O control signals.
- The 8237 has two eight bit registers called mode set registers and status registers.
- It works in two modes (i) Master Mode (ii) Slave mode
- It is an I/O to the microprocessor (slave mode) and it is a data transfer processor to peripherals such as floppy disk (master mode).

8237 - DMA controller with internal Registers

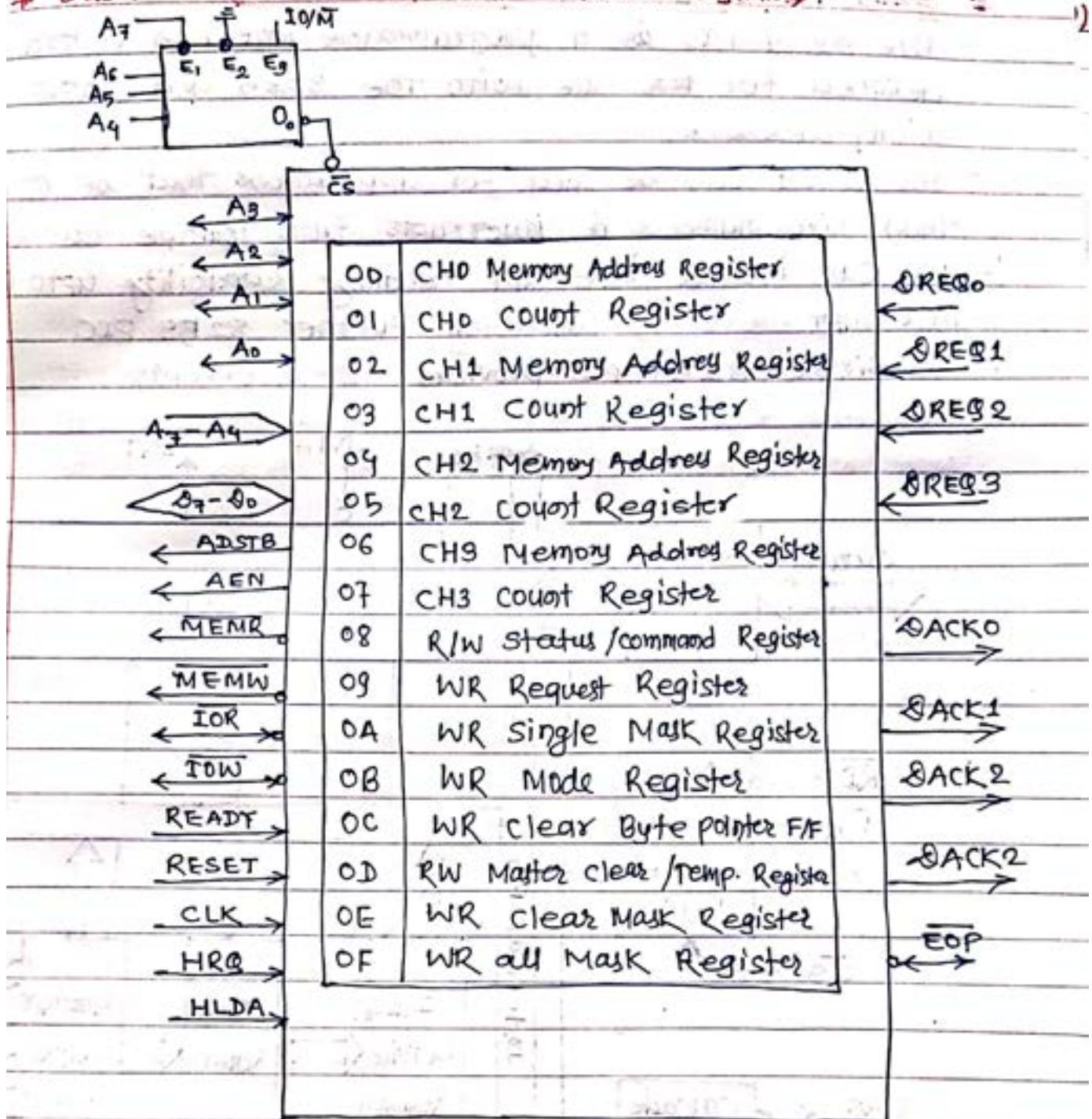


fig. 8237 DMA with internal registers

8259-PIC (Programmable Interrupt controller):

- The 8259-PIC is a programmable interrupt controller (PIC) designed for the use with the 8085 and 8086 microprocessors.
- The 8259 can be used for applications that use more than five numbers of interrupts from multiple sources.
- We can increase interrupt handling capability upto 64-bit interrupt level by cascading further 8259 PIC.
- A single 8259 PIC provides 8-interrupts.

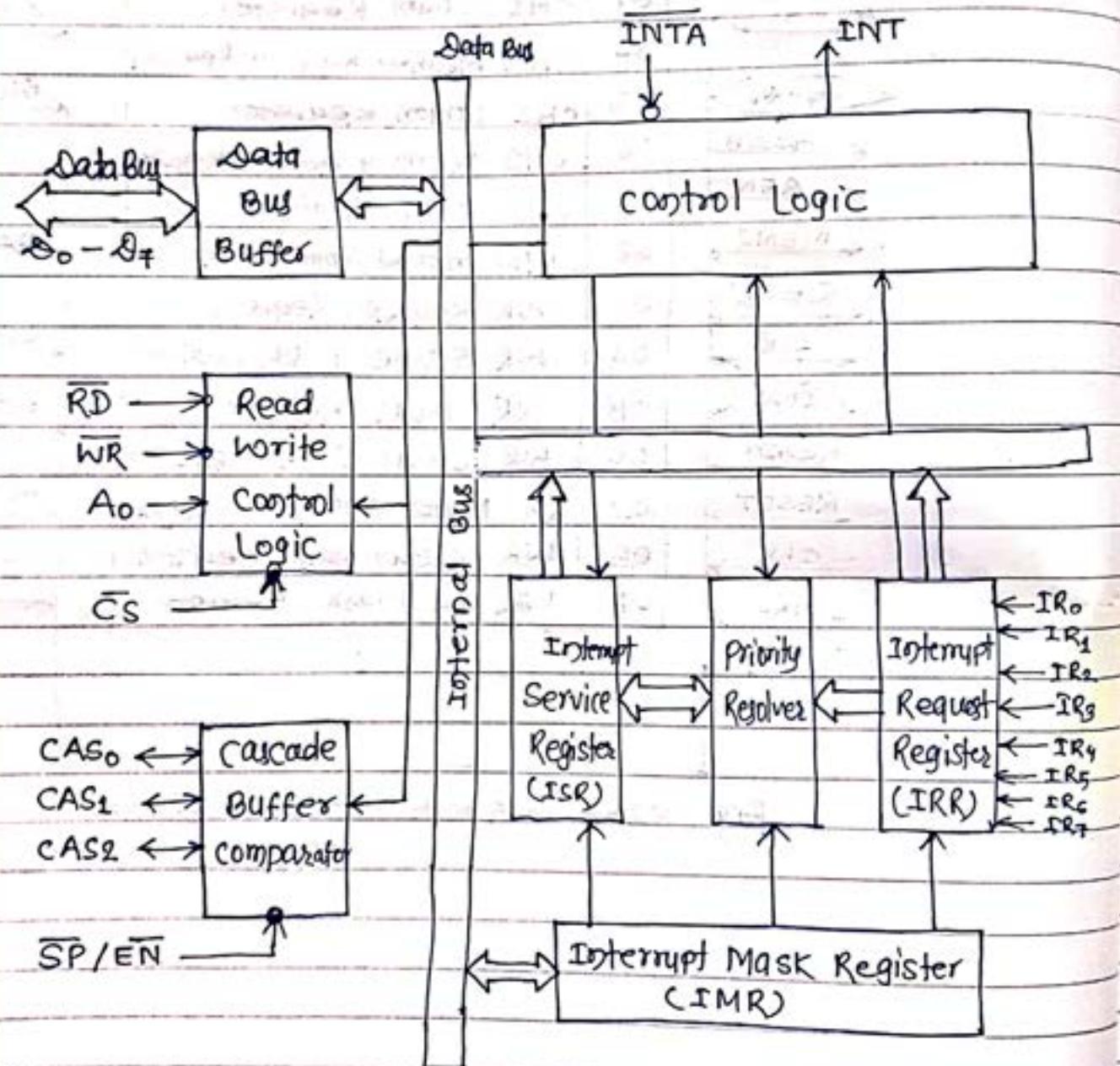


fig. Block Diagram of 8259-PIC

- gt includes eight blocks: Data Bus Buffer, Read/write logic, control logic, three registers (IRR, ISR and IMR), priority resolver, and cascade buffer.

(i) Data Bus Buffer:

- This block is used to communicate between 8259 and 8085/8086 microprocessor by acting as buffer.
- gt takes the control word from 8085/8086 and send it to the 8259.
- gt transfers the opcode of the selected interrupts and address of ISR to the other connected microprocessor.
- gt can send maximum 8-bit at a time.

(ii) Read/write control logic:

- This block works when the value of pin \overline{RD} is low (0).
- This block is used to flow the data depending upon the inputs of RD and WR. These are active low pins for read & write.

(iii) Cascade Buffer Comparator:

- To increase number of interrupt pins, we can cascade more number of pins, by using cascade buffer. When we are going to increase the interrupt capability, CAS lines are used to control multiple interrupt.

(iv) Control Logic:

- gt controls the functionality of each block.
- gt has two pins: INT as output and INTA as input
- The INT is connected to the INTR of the microprocessor, to send the interrupt to MP.
- The INTA is connected to MP, to receive the interrupt acknowledge.

(v) Interrupt Request Register (IRR):

- The IRR is used to store all the interrupt levels which are requesting the service.
- It has 8 interrupt input lines $IR_7, IR_6, IR_5, IR_4, IR_3, IR_2, IR_1, IR_0$.
- The IRR is an 8-bit register having one bit for each of the interrupt lines.

(vi) Interrupt Service Register (ISR):

- It is an 8-bit register, which stores the level of the interrupt request, which is currently being serviced.

(vii) Interrupt Mask Register (IMR):

- It is an 8-bit register, which stores the masking pattern for the interrupts of 8259.
- It stores the one bit per interrupt level.

(viii) Priority Resolver:

- It examines the IRR, ISR and IMR and determine which interrupt of highest priority and should be sent to the CPU.

- # 8255 - PPI (Programmable Peripheral Interface) : (Parallel I/O)
- 8255 is a general purpose programmable I/O device designed to interface the CPU with its outside world such as ADC, DAC, keyboard etc.
 - It is designed to increase I/O interfacing capability of 8085.
 - We can program it according to the given condition.
 - It can be used with almost any 8085.
 - It has 24 I/O pins that can be grouped in two 8-bits Parallel Ports : A and B, with the remaining 8-bits into Port C.
 - Port C is further divided into two 4-bit ports i.e. Port C lower and Port C upper.
 - 8255 functions in following modes
 - I/O Modes [Mode 0, Mode 1, Mode 2]
 - BSR Mode

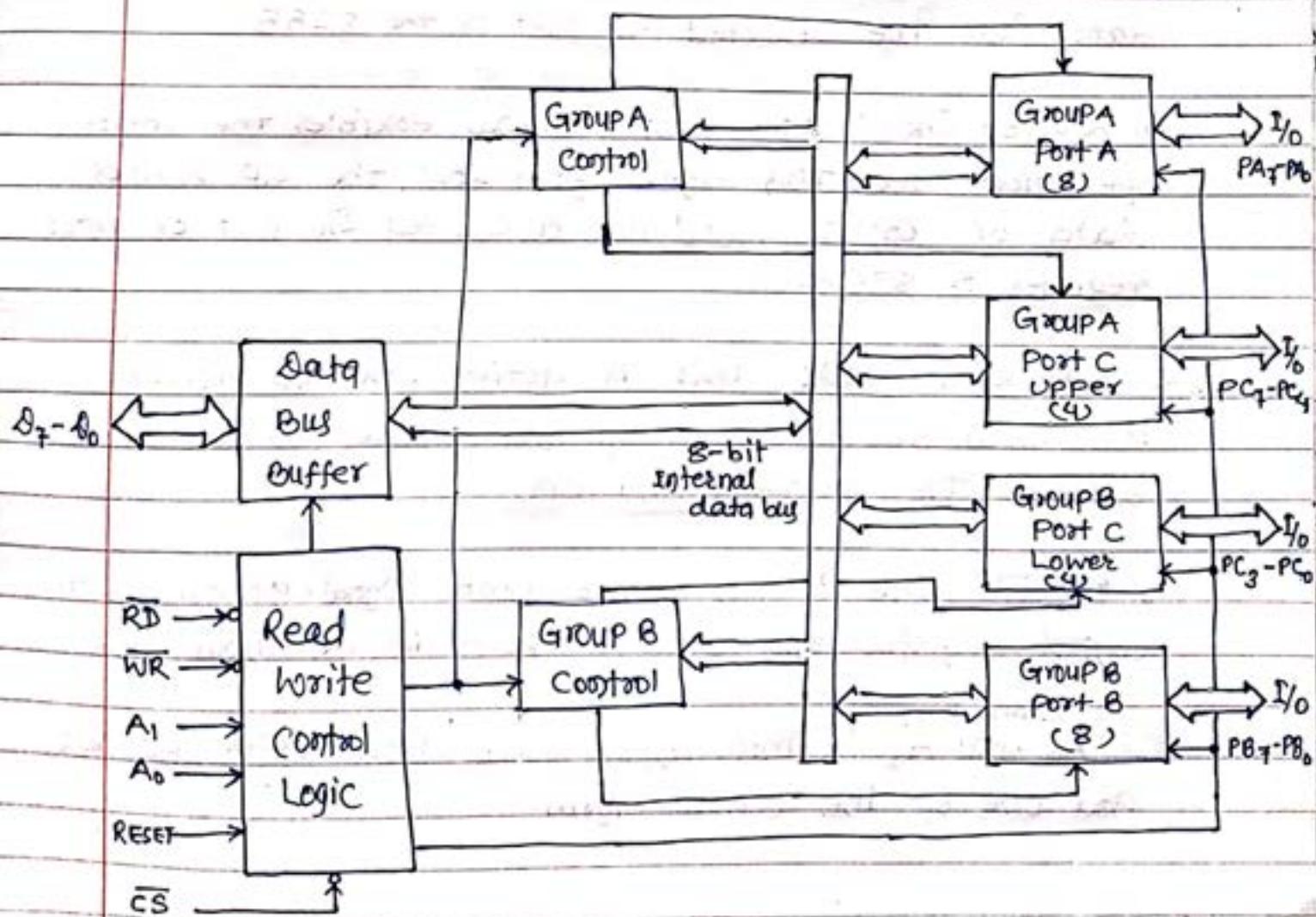


fig. Internal Block diagram of 8255A

The 8255A has the following main blocks:

(i) Data bus buffer:

- The 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus.
- Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU.
- Control words and status information are also transferred through the data bus buffer.

(ii) Read/write control logic:

- The function of this block is to manage all of the internal and external transfers of both data and control or status words.
- Control signals of 8255A are

(a) Read (RD): This control signal enables the read operation. When the signal is low, the microprocessor reads the data from the selected I/O port of the 8255.

(b) Write (WR): This control signal enables the write operation. When this signal goes low, the CPU writes data or control word into a selected I/O port or control register of 8255.

(c) Chip select (\bar{CS}): This is active low i/p signal. This input selects the chip and enables the communication between the 8255A and CPU.

(d) RESET: This is an active high signal. It clears the control register and sets all ports in the input mode.

(Address pins)

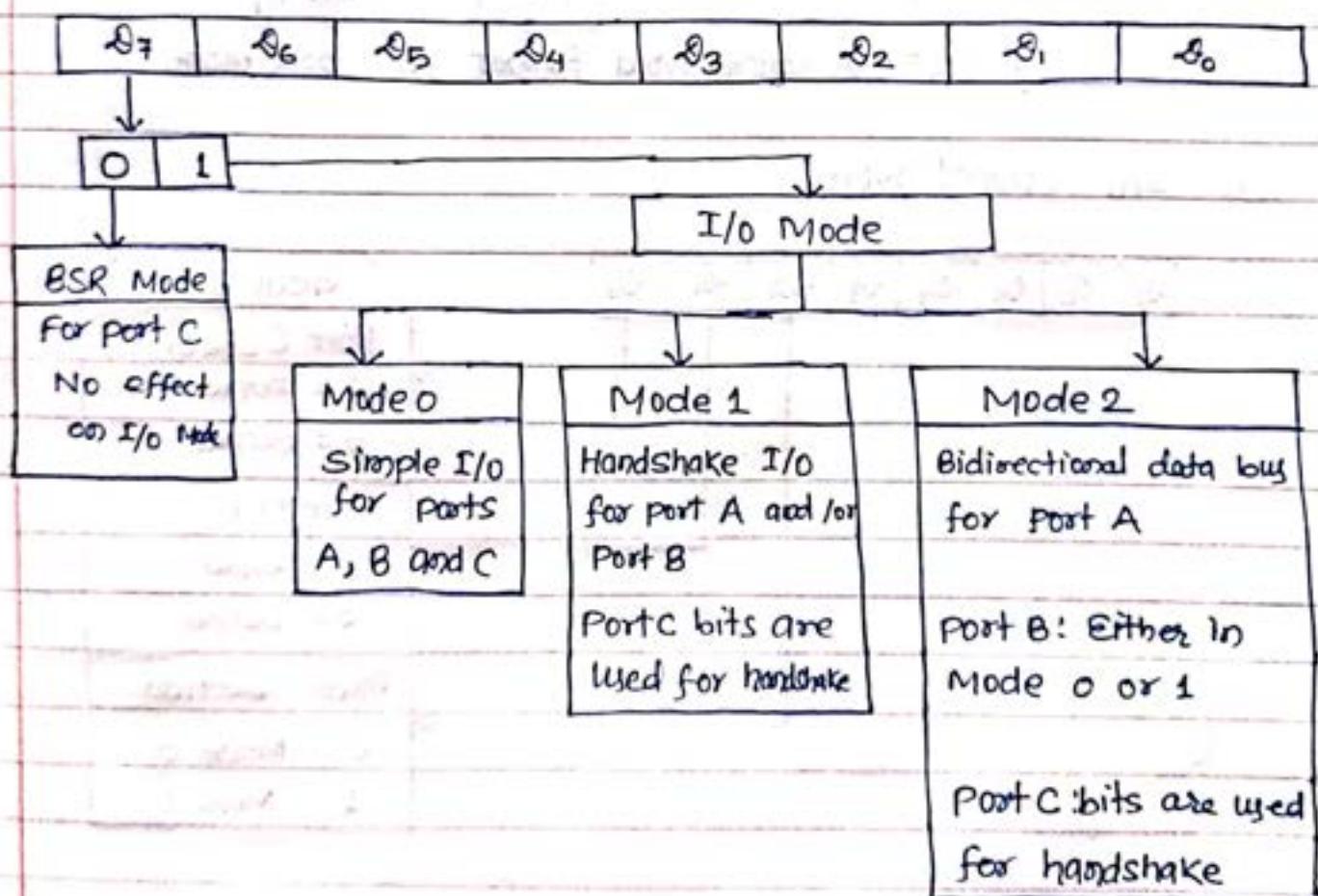
(e) A_0 and A_1 : - These input signals work with RD, WR, and one of the control signals.

CS	A ₁	A ₀	Result (Selection)
0	0	0	Port A
0	0	1	Port B
0	1	0	Port C
0	1	1	Control Register
1	X	X	No Selection

(iii) Group A and Group B Controls:

- These two groups accept 'commands' from the read/write control logic, receives control word from the internal data bus, and issues the proper commands to its associated pins-ports.
- Control Group A → Port A and Port C upper (C₇ - C₄)
- Control Group B → Port B and Port C lower (C₃ - C₀)

Control Word for operating mode of 8255



(i) BSR Control word:

If MSB of control word (δ_7) is 0, PPI works in BSR mode.

In this mode only port C bits are used for set or reset.

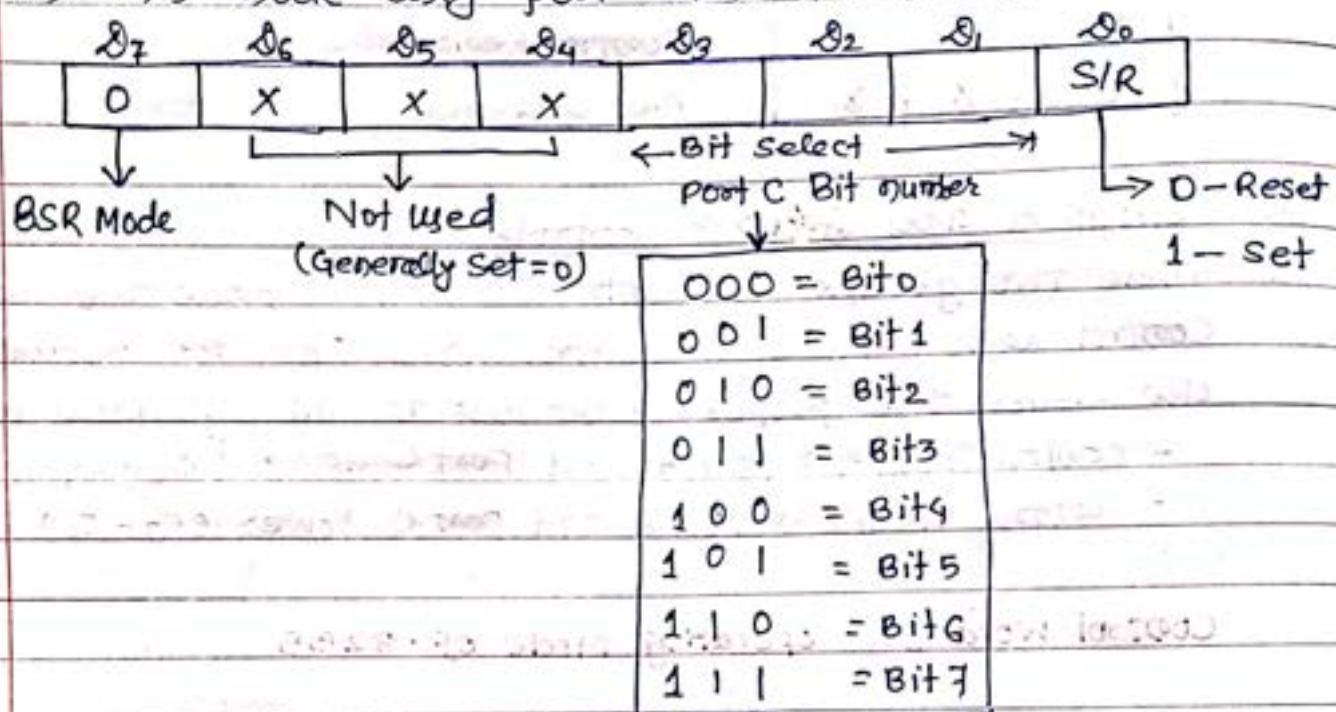


fig: 8255 A control word format for BSR mode

(ii)

(ii) I/O control word

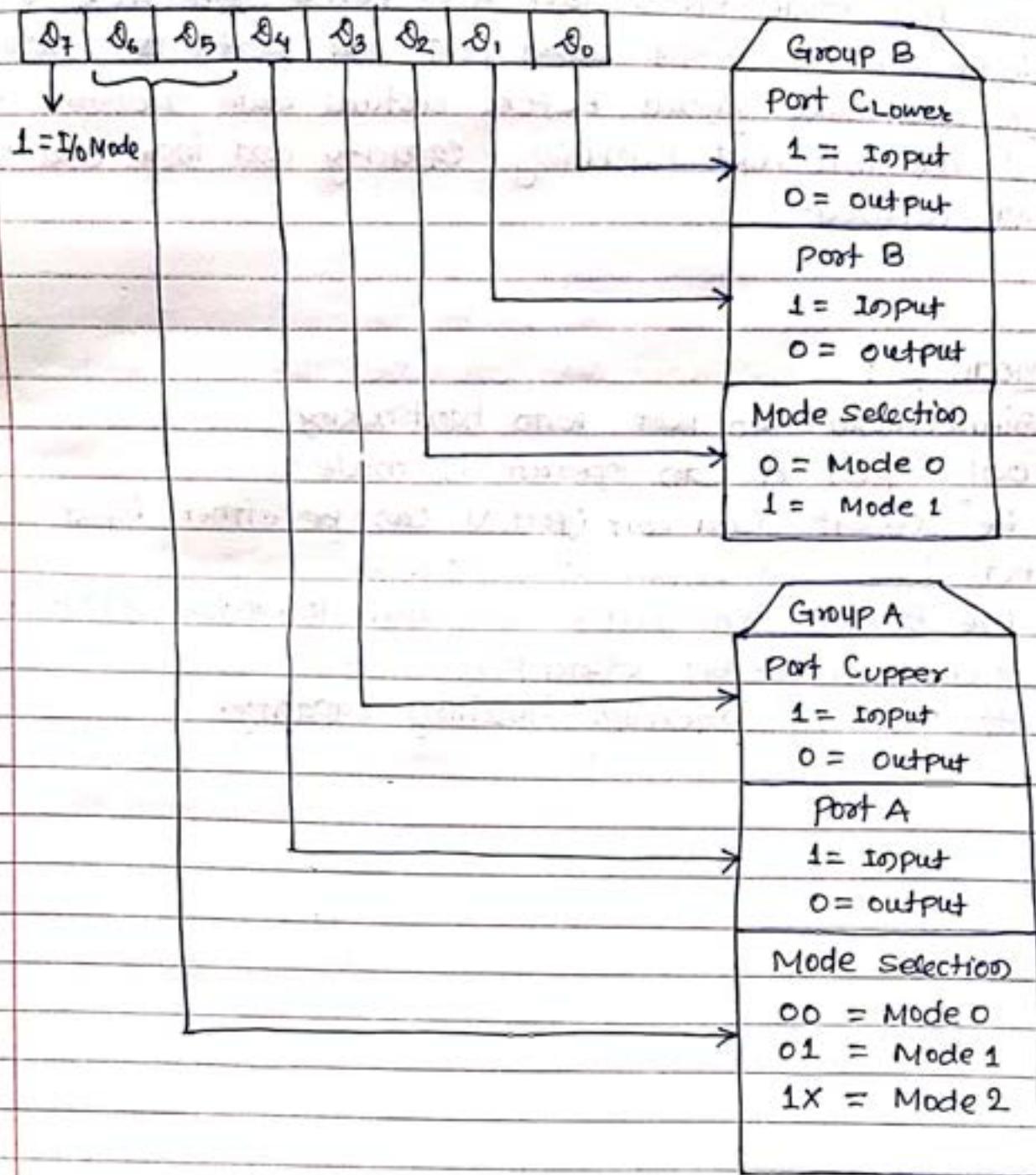


fig: 8255A control word format for I/O Mode

If MSB of control word (D_7) is 1, PPI works in input-output mode. This is further divided into three modes:

(a) Mode 0:

- In this mode all the three ports (Port A, B, C) can work as simple input function or simple output function.
- In this mode there is no interrupt or handshaking capability.

Mode 1 :

- In this mode either port A or port B can work as simple input port or simple output port, and port C bits are used for handshake signals before actual data transmission.
- It has interrupt handling capacity and input and output are latched.

Mode 2 :

- Bidirectional I/O port with handshaking
- Only port A can operate in mode 2.
- The 8-bit data port (port A) can be either input or output. Both inputs and outputs are latched.
- The 5 bits from port C are used to provide control and status for 8-bit data port.
- It also has interrupt handling capacity.

- (Serial I/O)
- # 8251 - USART (Universal synchronous Asynchronous Parallel transmitter)
- The 8251 is a programmable serial communications chip designed for synchronous and asynchronous serial data communication.
 - As a peripheral device of a microcomputer system, it receives parallel data from the CPU and transmits serial data after conversion.
 - This device also receives serial data from the outside and transmits parallel data to the CPU after conversion.

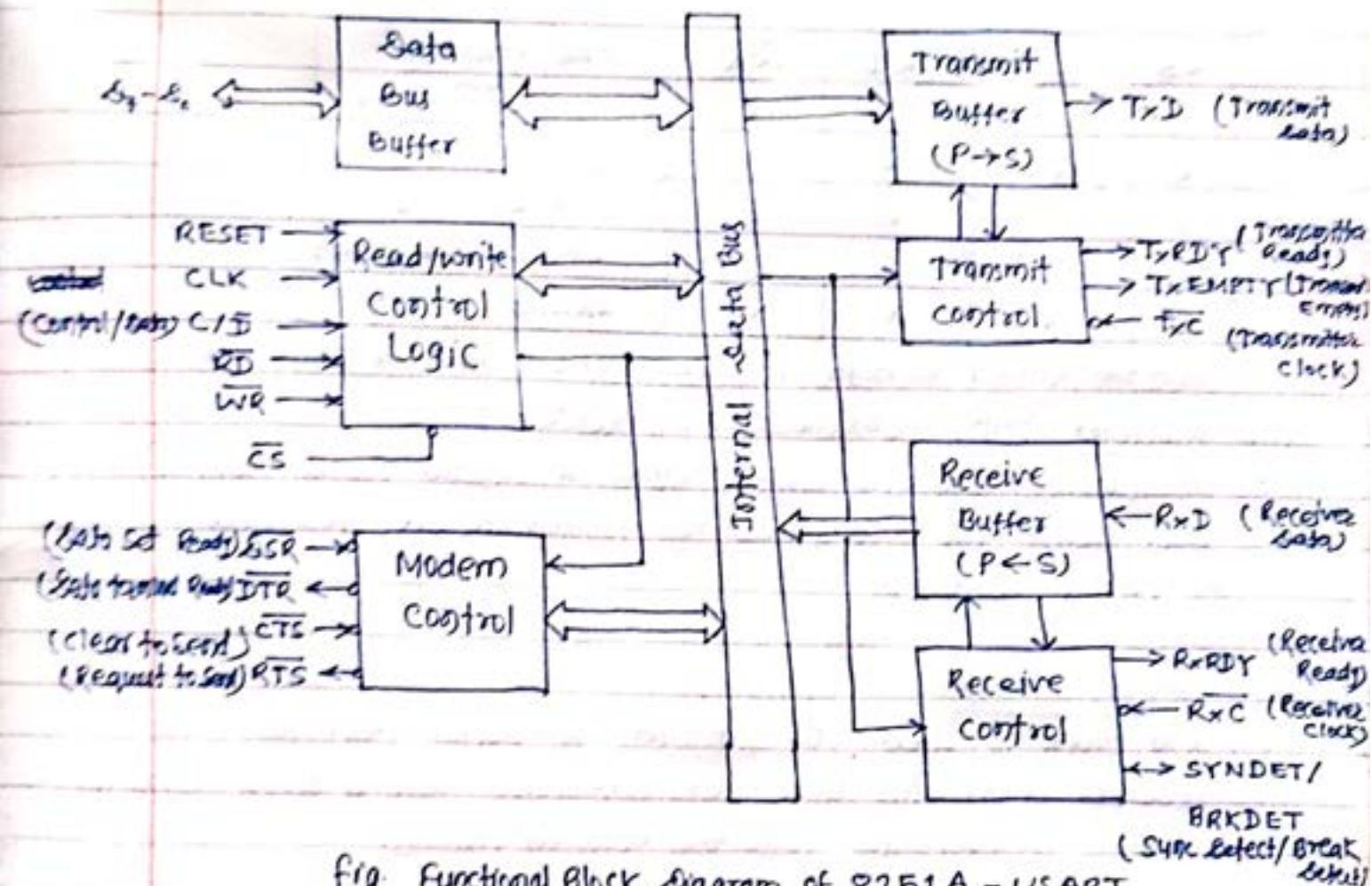


fig Functional Block Diagram of 8251A - USART

The functional block diagram of 8251A consists of five sections. They are:

- Data Bus Buffer
- Read/write Control Logic
- Transmitter
- Receiver
- Modem Control

(i) Data Bus Buffer:

- This block helps in interfacing the internal data bus of 8251 to the system data bus.
- The data bus transmission is possible between 8251 and CPU by the data bus buffer block.

(ii) Read/write control logic:

- It is a control block for overall device. It controls overall working by selecting the operation to be done.
- The operation selection depends upon input signal as:

\bar{CS}	C/D	\bar{RD}	\bar{WR}	Operation
0	0	0	1	CPU $\xleftarrow{\text{data}}$ 8251
0	0	1	0	CPU $\xrightarrow{\text{data}}$ 8251
0	1	0	1	CPU $\xleftarrow{\text{Status word}}$ 8251
0	1	1	0	CPU $\xleftarrow{\text{Control word}}$ 8251
1	X	X	X	Invalid

: When the RESET is high, it forces 8251A into the Idle mode.

(iii) Modem Control (Modulator/demodulator):

- A device converts analog signal to digital signals and vice-versa and helps the computers to communicate over telephone lines or cable wires.

(iv) Transmit Buffer:

- This block is used for parallel to serial converter that receives a parallel byte for conversion into a serial signal and further transmission onto the common channel.

(v) Receiver section:

- Receives Serial data on RxD
- Converts Serial into parallel
- Transfer data to Microprocessor unit
- Those data transmission can be done for Synchronous and Asynchronous data transfer.

SRAM (Static RAM)

DRAM (Dynamic RAM)

usage cache Memory

Main Memory

speed very fast

Fast

cost costly

cheaper than SRAM

Density less packing density

Higher packing density

Power consume more power

consume less power than SRAM

Maintenance no maintenance

Maintenance needed

construction not easy to construct

Easy simpler in construction

: stored data do not change with time : stored data changes with time

Material : it is made up of with transistors and flip flop.

: it is made up with capacitors and few transistors

Capacity : it is available in smaller

: it is usually available in large

storage capacity of few MB

storage capacity of few GB

: it stores data in the form of voltage.

: it stores data in the form of charge

: Refreshing circuit is not implemented

: Refreshing circuit is implemented

Example: cache memory

: DDR, DDR2, DDR3 etc.

CH-5 Data Communication Basics:

5.1. Serial and parallel data communication

Serial data communication

(1) Definition: The data bits are transmitted serially over a common communication link one after the other.

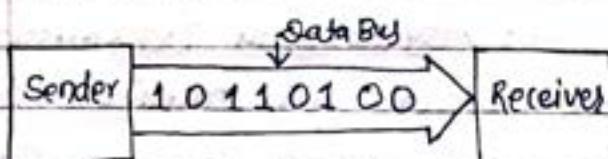


fig. Serial Communication

Parallel data communication

: The various data bits are simultaneously transmitted using multiple communication links between sender and receiver.

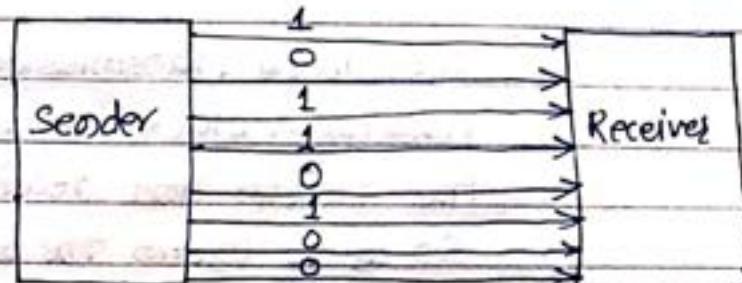


fig. Parallel Communication

(2) Data transmission speed : Slow

: comparatively fast

(3) No. of comⁿ link used : Single

: Multiple

(4) No. of transmitted bit/clock cycle : only one bit

: number of link will carry n bits

(5) Cost : Low

: High

(6) Crosstalk : Not present

: Present

(7) Mode of transmission : Full duplex

: Half duplex

(8) Suitable for : Long distance

: Short distance

(9) Circuit used : Simple

: relatively complex

(10) System up-gradation : Easy

: quite difficult

(11) High frequency operation : More efficient

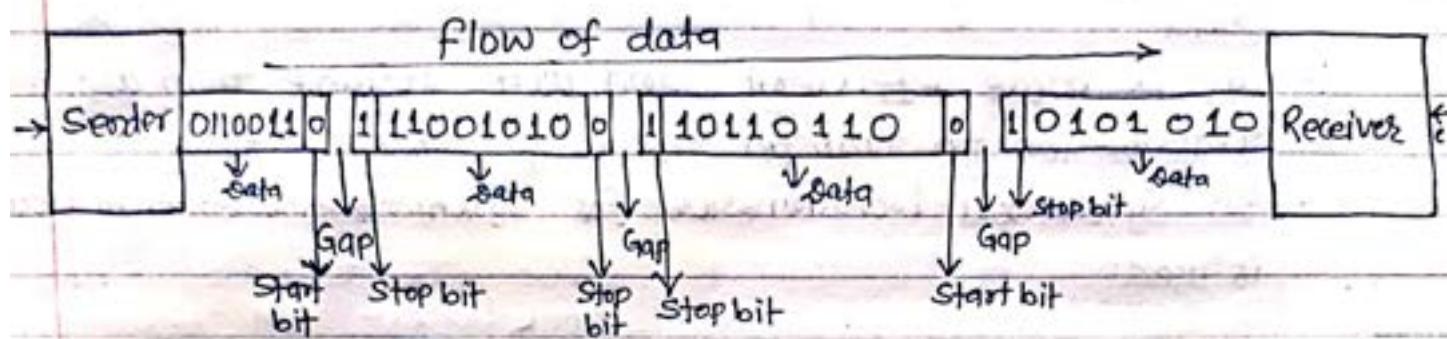
: Less efficient

Synchronous versus, Asynchronous, serial data transmission

Synchronous Data Transmission	Asynchronous Data Transmission
1. In Synchronous transmission data is sent in form of blocks or frames.	: In Asynchronous transmission data is sent in form of byte or character.
2. Synchronous transmission requires a clock signal between sender and receiver so as to inform the receiver about new byte.	: Asynchronous transmission sender and receiver does not require a clock signal as the data sent here has a parity bit attached to it with which indicates the start of the new byte.
3. External clock is used	: Internal clock is used
4. Synchronous transmission is fast	: Asynchronous transmission is slow.
5. Transmitter and receiver having common clock.	: Separate clock
6. Synchronous transmission is costly	: Asynchronous transmission is economical
7. In Synchronous transmission, time interval of transmission is constant.	: In asynchronous transmission, time interval of transmission is not constant, it is random.
8. In synchronous transmission, there is no gap present between data.	: In asynchronous transmission, there is present gap between data.
9. It is used for high speed of transmission.	: It is used for low speed of transmission.

2. Asynchronous Serial Data Communications

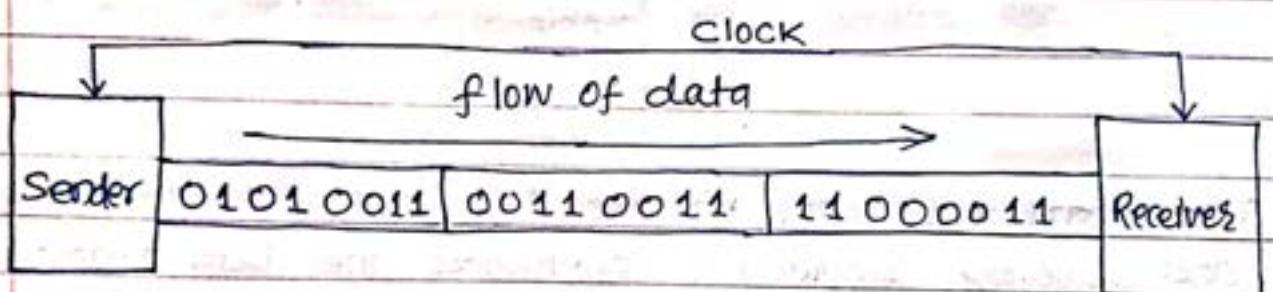
- In Asynchronous transmission, data is sent in form of byte or character.
- This transmission is the half duplex transmission.
- In this transmission start bits and stop bits are added with data.
- It does not require synchronization.



- The start and stop bit including gaps allow the receiving and sending computers to synchronize the data transmission.
- Asynchronous communication is used when slow speed peripherals communicate with the computer.
- The main disadvantage of asynchronous communication is slow speed.
- Asynchronous communication does not require complex and costly hardware equipment.
- It helps to transmit signals from the sources which have different bit rates.

5.4 Synchronous Serial Data Communication

- In Synchronous transmission, data is sent in form of blocks or frames.
- This transmission is full duplex type.
- Between sender and receiver the synchronization is compulsory.
- In synchronous transmission, there is no gap present before data.
- It is more efficient and more reliable than asynchronous transmission to transfer the large amount of data.
- In Synchronous transmission for synchronization external clock is used.



- There is no gap between characters being transmitted.
- It helps to transfer a large amount of data.
- Synchronous data communication is the high speed.
- The Synchronous communication require high-speed peripherals/devices and a good quality, high bandwidth communication channel.
-

Protocol of Synchronous Serial data communication (BISYNC protocol)

BISYNC (Binary Synchronous communication) protocol:

- It is a character or byte oriented communication protocol developed by IBM in 1960's.
- BISYNC is a half duplex protocol, it will synchronize in both directions on a full duplex channel.
- BISYNC supports both point to point and multipoint transmission.
-

Frame format of BISYNC:

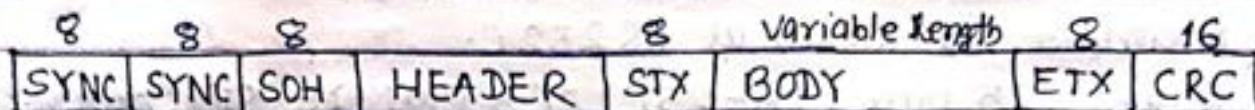


fig. Frame format of BISYNC

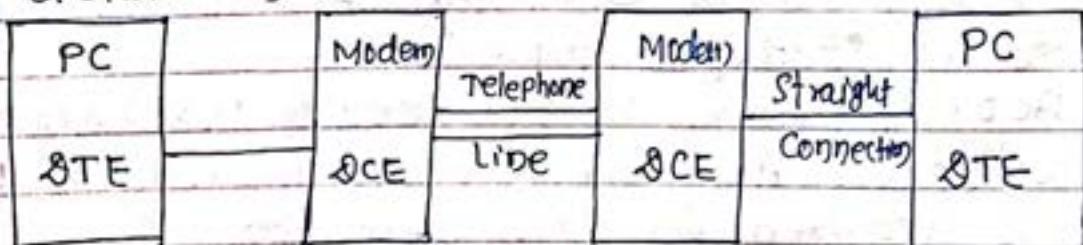
- BISYNC has two (2) Synchronous fields (SYNC) in the starting of the frame.
- The frame includes two synchronous fields of 8-bit each. These two fields are essential to spot the beginning of the frame.
- The third field is SOH (Start of header) which of 8-bits, followed by fourth field - HEADER
- The fifth field is STX (start of text) which of 8-bits, followed by a body of variable length.
- The BODY is followed by ETX (End of text) which of 8-bits.
- CRC (cyclic redundancy check) is the last field of 16-bits, which is required for error detection.
- Frames are transmitted, beginning with leftmost field.
- The start of the frame is denoted by sending a special SYNC (Synchronize) character.
- The data portion of the frame, is contained between special Character STX and ETX.
- BISYNC has largely been replaced by SDLC (Synchronous Data Link Control).

5.3. Serial Data Transmission Methods and standards (RS 232/RS 232C/RS 422/RS 423 A)

- RS 232 standard

- : Developed by Electronic Industry Alliance (EIA) for connection serial devices.
- : Used to connect DTE (Data terminal equipment) and DCE (Data communication equipment).
- : This defines DTE as computer and DCE as modem.
- : When data are transmitted as voltage, the commonly used standard is known as RS 232 C.
- : It uses 25 pins (DB-25P) or 9 pins (DE-9P) standard where 9 pins standard does not use all signals i.e. data, control, timing and ground.
- : It specifies that DTE connector should be male and DCE connector should be female.
- : The voltage level for RS 232 are
 - : Logic level 0 \Rightarrow +3V to +15V
 - : Logic level 1 \Rightarrow -3V to -15V
- : Normally, $\pm 12V$ level are used.

- The rate of data transmission is limited to 20 K bits/sec and a distance of 50 ft.



- : The PC or microcomputer acts like DTE (Data Terminal Equipment) and Modem acts like a DCE (Data Communication Equipment). The RS 232 standard is defined with respect to DTE and DCE devices. DTE transmits data through pin number 2 while DCE receives data via pin number 2 in RS 232 standard.

Hence the connection between a PC (STE) and Modem (SCE) is a straight through cable.

While the direct connection between a PC and PC without modem is crossover cable. The Pin number 2 is connected to a pin number 3 because STE sends data through pin 2 and STE receives data through pin number 3. This connection is known as Null modem connection. These two connection between STE and SCE and STE and STE is shown below.

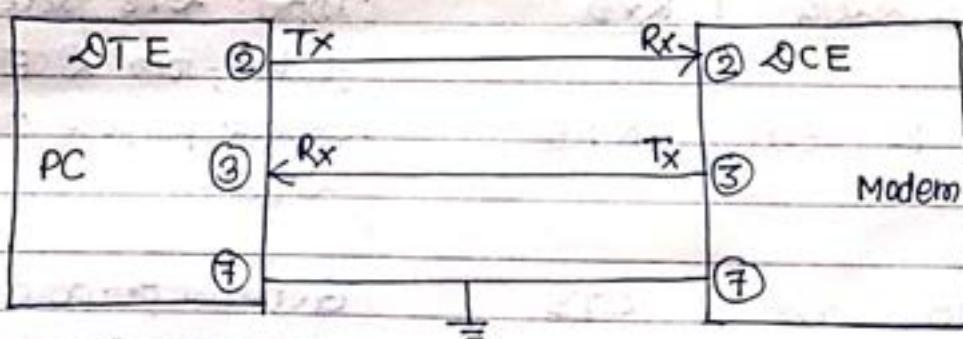


fig. RS 232 connection: STE to SCE

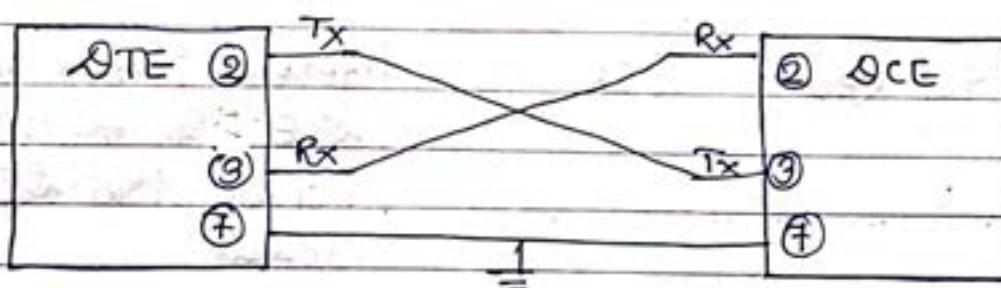


fig. RS 232 connection: STE to STE

Advantages

- It is widely used for point to point connection between SCE and STE devices.
- Low cost
- Due to simplicity, the interface RS 232 is supported in many devices
- It is free from noise.

Disadvantages

- : Bandwidth is low
- : Cable length is limited
- : Power consumption increases.

RS 232 Signals:

The various signals which are used with handshake data communication are as:

Pin No.	Signals	Acronym	Functions
2	Transmitted Data	TxD	: Output: transmits data from DTE to DCE
3	Received Data	RxD	: Input: DTE receives data from DCE
4	Request to send	RTS	: General purpose output from DTE
5.	Clear to send	CTS	General purpose input to DTE; can be used as a handshake signal.
6.	Data Set Ready	DSR	General purpose input to DTE; can be used to indicate that DCE is ready
7.	Signal Ground	GND	Common reference between DTE and DCE
8.	Data Carrier Detect	DCD	Generally used to DTE disable data reception
9.	Data terminal ready	DTR	Output: Generally used to indicate that DTE is ready

RS 422 Standard

- Developed by Electronic Industry Alliance (EIA) for connection serial devices.
- RS 422 is able to provide data rates upto 10Mbps at distance upto 50 feet.
- It is a newer standard for serial data transfer.
- Uses differential signalling.
- Full duplex communication.
- Used as an improvement over the RS 232.
- No connector defined.
- Signal level - 6V to +6V.
- Using Reduced data rates, RS 422 is able to transmit data over distances of 4000 feet.
- Number of devices = 10 Receiver, 5 transmitters.
- Cabling → single ended, multi drop.
- Signals = Tx+, Tx-, Rx-, Rx+.
- Physical media - Twisted pair of wires.
- One device is known as the Data terminal equipment (DTE) and the other device is known as data communication equipment (DCE).
- RS 422 is a balanced four wire system.
- Two wire is for DTE transmit signal to DCE, and other two wire is for DCE transmit signal to DTE.

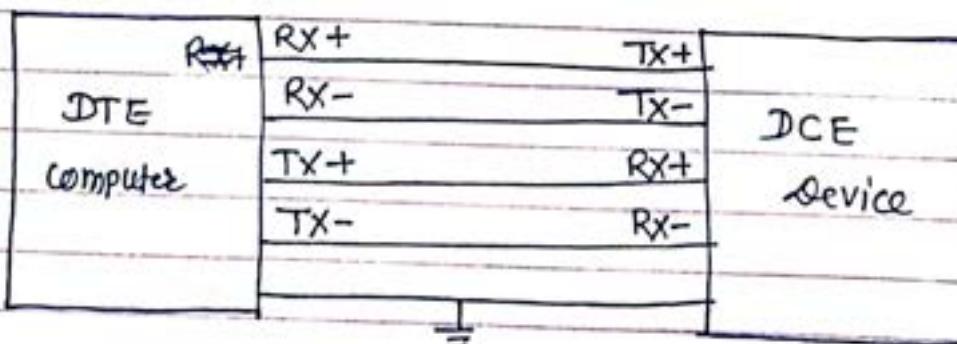


fig: Typical RS 422 wiring

Advantages

- Higher data rates
- Longer cable length
- Noise rejection

Disadvantages

- Only one transmitter for most applications
- Unidirectional
- Not as common

parameters	RISC	CISC
1. Stands for	: Reduced instruction set Computer	: Complex instruction set computers
2. No. of instructions	: Less	: large
3. Instruction format	: Fixed length	: variable length
4. Cost	: less	: High
5. Data transfer	: Register to Register	: Memory to Memory
6. Pipelining	: Not or less pipelined	: Highly pipelined
7. Instruction Set clock	: Execute in single clock	: More than one clock or Multiclock
8. Instruction type	: No. of register is large	: No. of register is very small
9. Bit format	: Fixed (32-bit) format	: varying format (16-64 bit for each instruction)
10. Memory Access	: Only for load/store	: Many instructions prefer memory
11. Execute time	: very less	: very high
12. Decoding of instructions	: Simple	: Complex
13. Examples	: SPARC, ARM	: VAX, AMD, 8085, 8086, 80386 etc
14. Focus on	: Software	: Hardware
15. code size	: large	: small

CH-4 16-Bit Microprocessor and programming

4.1 Internal organization of 8086

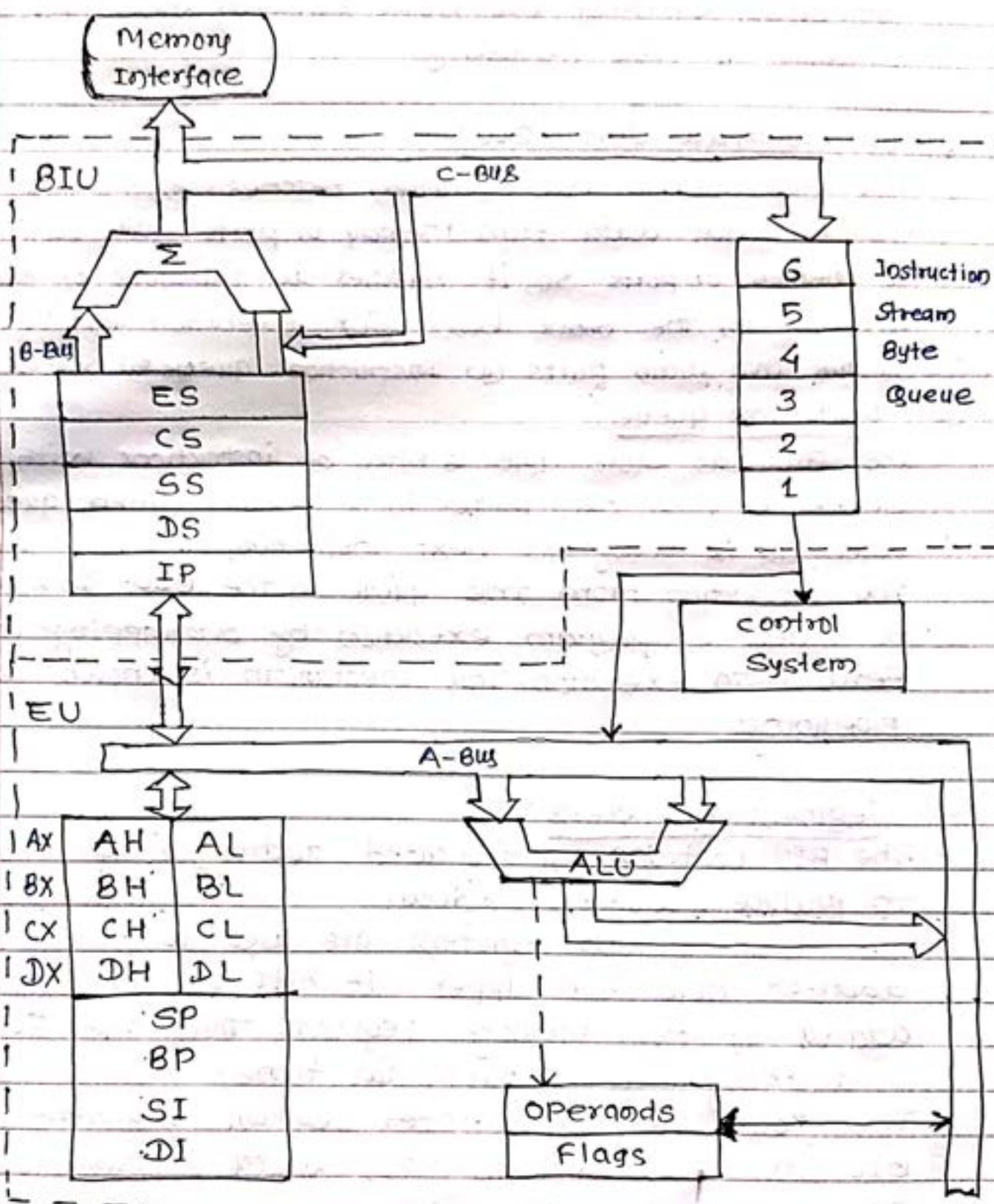


Fig. Internal organization of 8086

The 8086 microprocessor is divided into two independent functional parts, the bus interface unit (BIU) and the execution unit (EU). Dividing the work between these two units speeds up the processing.

1. Bus Interface Unit (BIU)

- The BIU sends out addresses, fetches instructions from memory, reads data from memory or ports and writes data to memory or ports. So, it handles all transfers of data and addresses on the basic buses for execution unit (EU).
- It has two main parts (a) instruction queue (b) segment registers

(a) Instruction queue:

- The BIU can store upto 6 bytes of instructions with FIFO (First in First out) manner in a register called queue.
- When EU is ready for next instruction, it simply reads the instruction from the queue in the BIU. This is done to speed up program execution by overlapping instruction fetch with execution. This mechanism is known as pipelining.

(b) Segment Registers:

- The BIU contains a dedicated address, which is used to produce 20-bit address.
- The four segment registers are used to produce 20-bit address hold the upper 16 bits of the starting address of four memory segments that the 8086 is working at a particular time.
- The four 16-bit register segment registers in BIU are code segments (CS), data segments (DS), stack segments (SS), Extra segments (ES)...

Code Segment (CS):

- The CS register is used for addressing a memory location in the code segment of the memory, where the

executable program is stored.

Data Segments (DS):

- The DS contains most data used by program. Data are accessed in the data segment by an offset address or the content of other register that holds the offset address.

Stack Segments (SS):

- SS defines the area of memory used for the stack.

Extra Segment (ES):

- ES is additional data segment that is used by some of the string to hold the destination data.

(c) Instruction pointer (IP):

- IP is a 16-bit register pointer to next instruction to be executed.

1.2 Execution Unit (EU):

- The execution unit decodes and executes the instructions.
- The EU contains the ALU, flags, and the general purpose registers and is responsible for carrying out all arithmetic and logical operations.
- The EU tells the BIU where to fetch the instructions or data from, decodes the instructions and executes the instructions.

The various parts of EU are:

(a) control circuitry, instruction decoder and ALU:

- The EU contains control circuitry which directs internal operations and control the flow of data inside the microprocessor.
- The instruction decoder translates or decodes the instructions ~~and executes~~ which are fetched from the memory.

- : After decoding the it places the instructions in a series to perform the required task by the EU.
- : The ALU is a 16 bit unit which performs the AND, OR, XOR, Addition, Subtraction, Increment, Decrement, Complement and Shift functions.

(b) General purpose Registers

- : The EU has 8 general purpose registers labeled as AH, AL, BH, BL, CH, CL, DH and DL.
 - : These registers can be used individually for temporary storage of 8-bit data.
 - : certain pairs of general purpose registers can be used to store 16-bit data words.
 - : The possible pairs and corresponding names are as
- | | |
|----------|-------------------------|
| AH, AL = | AX (Accumulator) |
| BH, BL = | BX (Base register) |
| CH, CL = | CX (Counter) |
| DH, DL = | DX (Data) |

AX: It is used as 16-bit accumulator in 16-bit operations and AL is used as an accumulator in 8-bit operations

BX: It is also called base register can be used as memory pointer in data segment.

CX: It is also called counter register can be used as a counter in string manipulation and shift/rotate instructions

DX: It is also called data register can be used in MUL and DIV instructions

(c) Pointers and base registers

- : These are 16-bit registers used to hold the offset or logical address within a segment.

Stack pointer (SP): Points to the program stack in stack segment.

Base pointer (BP): Points to data in stack segment.
 source index (SI) and destination index (DI): are used in indexed addressing. SI & DI are also used as source, and destination address in string manipulating instructions respectively.

(d) Flag registers:

- The 8086 has a 16-bit flag register.
- In this register 9-bits are active flags (6 conditional & 3 control flags), other 7 bits are undefined.

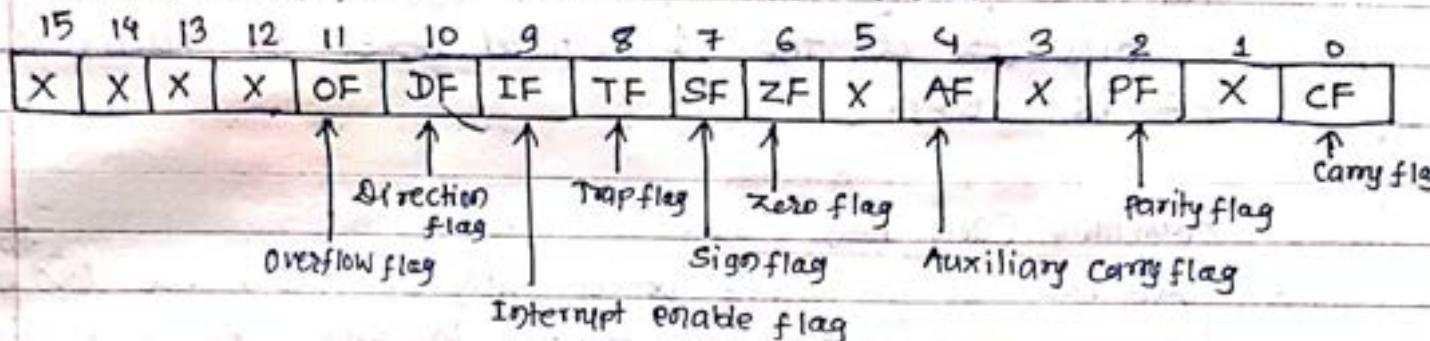
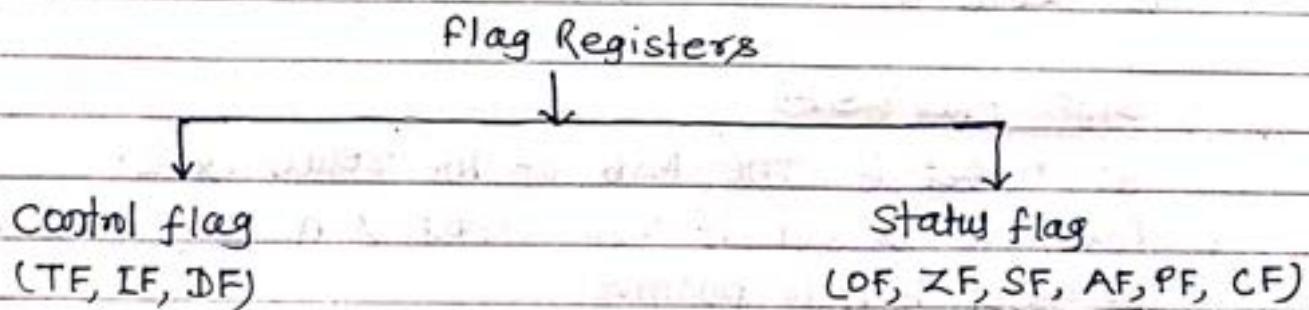


fig: Flag Register format of 8086

Flags of 8086

- The 16-bit flag register of 8086 contains 9 active flags (6 conditional and 3 control flags), other flags are undefined.



Status flag :- It indicates certain condition that arises during the execution. They are controlled by the processor.

Control flag :- It controls certain operations of the processor. They are deliberately set/reset by the user.

Status flag:

Carry flag (CF):

- This CF is set if there is a carry out or a borrow in for the most significance bit of the result during the execution of an instruction otherwise CF is reset.

Parity flag (PF):

- PF is set if the result produced by the instruction has even parity that is, if it contains an even number of bits ~~are~~ at the 1 logical level. If parity is odd, PF is reset.

Auxiliary Carry flag (AF):

- AF is set if there is a carry out from the low nibble ^(D₀-D₃) into the high nibble or borrow in from the high nibble ^(D₄-D₇) into the low nibble of the lower byte in a 16-bit word. otherwise AF is reset.

Zero flag (ZF):

- ZF is set if the result produced by an instruction is zero. Otherwise ZF is reset.

Sign flag (SF):

- SF is set if the MSB of the result is 1.
- Thus, SF is set if the result is a negative number or reset if it is positive.

Overflow flag (OF):

- This flag is used in signed arithmetic operation (addition or subtraction). If the signed result is of more bits than the destination operand, then this flag will be set otherwise it will be cleared.

control flag

Three out of the nine flags are used as controlling flags. By storing 0/1 in controlling flags we can control corresponding operations of microprocessor. The controlling flags are

Trap flag (TF):

- It is used for single step control.
- It allows user to execute ~~the~~ one instruction of a program at a time for debugging.
- When trap flag is set, program can be run in single step mode.

Interrupt flag (IF)

- It is an interrupt enable/disable flag.
- If it is set maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled.

Direction flag (DF)

- It is used in string operation.
- If it is set, string ~~one~~ bytes are accessed from higher memory address to lower memory address.
- When it is reset, the string bytes are accessed from lower memory address to higher memory address.

4.3 Pin Diagram of 8086

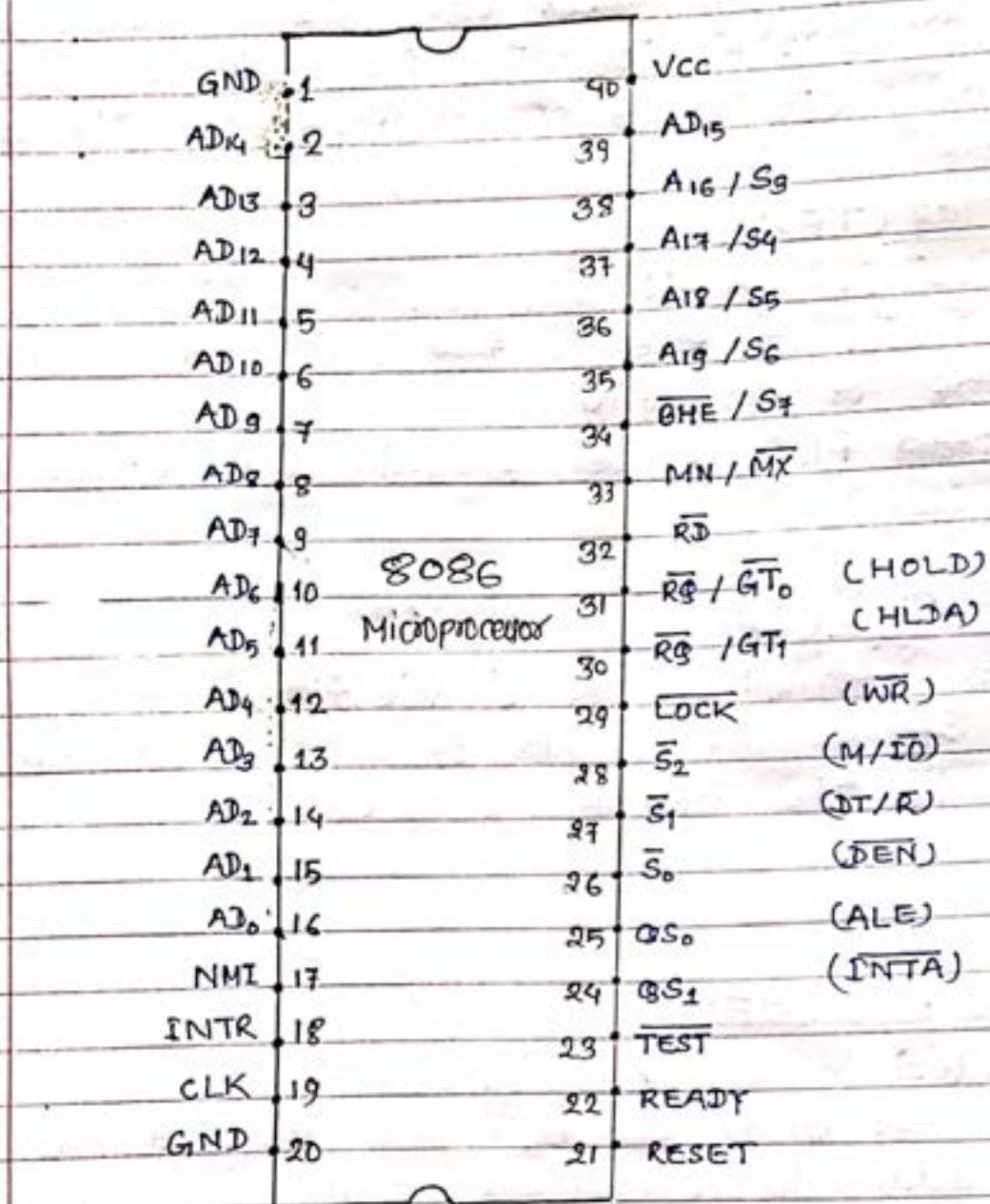


fig: Pin Configuration of 8086 Microprocessor

Address and Data Pins

AD₀ - AD₁₅ (Bidirectional)

- These lines are multiplexed bidirectional address/data bus.
- When ALE = 1, then address bus gets enabled, else data bus will get enabled.

A₁₆/S₃ - A₁₉/S₆ (Unidirectional)

- These lines are multiplexed and unidirectional address

and status bus.

- The address and bits are separated from the status bit using latches controlled by the ALE signal.
- S₅ gives the status of interrupt flag (IF)
- S₆ goes low, when 8086 controls the shared system bus.
- S₃ and S₄ indicates the segment register.

S ₄	S ₃	Register
0	0	ES
0	1	SS
1	0	CS
1	1	DS

BHE / S₇:

- The bus high enable (BHE) signal is used to indicate the transfer of data over the higher order data bus (D₈ - D₁₅)
- 8-bit I/O devices use this signal
- S₇ is reserved for future development

Interrupt Related pins

NMI :- It is a non-maskable interrupt signal

INTR :- It is an interrupt request signal

INTA :- This is an interrupt acknowledge signal

Clock Related Pins:

CLK : Generates clock signals that synchronize the operation of processor.

RESET :- When high, microprocessor enters into reset state and terminates activity of processor.

READY :- When high, it indicates the device is ready to transfer data; when low; microprocessor is in wait state.

Control pins:

TEST: This input is examined by a "WAIT" instruction. If the TEST pin goes low, execution will continue, else the processor remains in an idle state.

MN/MX

: If MN/MX is high, it works in minimum mode.

: If MN/MX is low, it works in maximum mode.

Mode Multiplexed pins

S ₂	S ₁	S ₀	Machine Language
0	0	0	Interrupt Acknowledgment
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	HALT
1	0	0	Code Access
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive

DEN:

- : This signal indicates the availability of valid data over the address / data lines.
- : It is used to enable the trans-receivers to separate the data from the multiplexed address / data signal.

DT/R (Data Transmit/Receive):

- When high: The processor sends out data (transmitted out)
- When Low: The processor receiving data

M/I/O

- : When it is low, it indicates the CPU is having an I/O operation, and when it is high, it indicates that the CPU is having a memory operation.

QS_1	QS_0	characteristics
0	0	No operation
0	1	First byte of opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

LOCK:

- : This signal indicates that other processors should not ask CPU (8086) to hand over the system bus.

WR:

- : It is used to write data in memory or output signal, depending on status of M/IO signal.

HOLD:

- : When DMA controller needs to use address/data bus, it sends a request to the CPU through this pin.

HLDA:

- : It is a Hold acknowledge signal.
- : It is issued after receiving the HOLD signal.

RQ/GT₀ and RQ/GT₁:

- : These are Request/Grant bidirectional pins
- : Other processors request the CPU through these lines to release the System bus
- : After receiving the request, CPU sends acknowledge signal on the same lines. RQ/GT₀ has higher priority than RQ/GT₁.

4.4. Instruction set of 8086

The 8086 instructions can be categorized as

1. Data transfer instruction
 2. Arithmetic instruction
 3. Logical instruction
 4. Shift and Rotate instruction
 5. Program Execution Transfer instructions (branch and loop instructions)
 6. String instructions
 7. Processor control instructions

1. Data Transfer instruction:

- The data transfer instruction provide the ability to move data either between its internal registers or between an internal register, I/O ports and a storage location in memory
 - The data transfer instruction include -

- (i) Instruction to transfer bytes or words

- (1) Instruction to transfer bytes or words
for May 2013 ECE 303 MATH 111 2013

e.g.: MOV, PUSH, POP, XCHG, XLAT

- (ii) Instruction for I/O port transfer

Eg: IN, OUT

- (ii) Instruction to transfer the address

Eg: LEA, LDS, LES

- (iv) instruction to transfer flag register

Eg: LAHF, SAHF, PUSHF, POPF

- (i) Instruction to transfer word

- ## (a) Mov instruction

- : Moves a byte from source to destination specified in our
(copies) instruction

Eg: MOV AX, 1200H ← source
 ↑ destination

Source : Register / Memory location / Immediate value

destination: Register / Memory location

Eg MOV AX, BX

MOV AX, EBXT

Eg MOV AL, [4000H]
Eg MOV [2310H], 37B2H

MOV memory, Memory is not possible

(b) XCHG Instruction

- Exchange a byte/word between the source and the destination specified in our instruction.
- Eg XCHG AX, BX

(c) XLAT instruction

- Move into AL, the contents of the memory location in the data segment (DS) whose offset address is formed by the sum of BX and AL.

Eg XLAT

$$AL \leftarrow DS : [BX + AL]$$

Initially

$$AL = 05H$$

$$\text{Therefore, } 1000 \times 10H + 0400H + 05H$$

$$BX = 0400H$$

$$= 10405H$$

$$DS = 1000H$$

(d) PUSH instruction

- The PUSH instruction pushes the data in the stack.
- Format : PUSH source
- The PUSH instruction decrements stack pointer by two and copies a word from some source to the location in the stack where the stack pointer points.
- Here the source must be a word (16 bit).
- The source of the word can be a general purpose register, a segment register or memory.

Eg: PUSH CX

Working $SP \leftarrow SP - 1$
 $[SP] \leftarrow CH$
 $SP \leftarrow SP - 1$
 $[SP] \leftarrow CL$

(e) POP instruction

- : It is opposite to the POP instruction.
- : The POP instruction copies a word from the stack location pointed by the stack pointer to the destination.
- : The destination can be a general purpose register, a segment register, or a memory location.
- : After the word is copied to the specified destination, the stack pointer is automatically incremented by 2.
- : Format : POP destination
- : Note: POP instruction does not support CS as a destination operation.
- : Eg POP CX

Working $CH \leftarrow [SP]$
 $SP \leftarrow SP + 1$
 $CL \leftarrow [SP]$
 $SP \leftarrow SP + 1$

(ii) Instruction for I/O port transfer

~~IN~~ IN instruction

- : The IN instruction will copy data from a port to the accumulator. If an 8-bit port is read, the data will go to AL and if an 16-bit port is read the data will goto AX.

OUT instruction

- : The OUT instruction copies a byte from AL or a word from AX to the specified port.

Eg: IN AX, 29H

Eg OUT 26H, AL

If we want to access a port number over 255 then first load the port address into DX and then use IN instruction
Eg:

MOV DX, FA32H

IN AX, DX

MOV DX, 456DH

OUT DX, AX

(iii) instruction to transfer the address

LEA instruction: Load Effective address

It takes the data from the source and copies it to the destination operand

The destination operand is always a register whereas the source can be an offset address of a variable or a memory location.

Eg: LEA AX, [BX]

LEA CX, Var_1

LEA BX, [BP][SI]; Loads effective address = BP+SI into BX Register

LDS instruction: Load data segment register

The word from first two memory locations is loaded into a register and the word from the next two memory locations gets stored to DS register.

Eg LDS BX, [SI]

Working: BL \leftarrow [SI]

BH \leftarrow [SI + 1]

- LES instruction: Load extra segment register
- This instruction is almost similar to the LDS instruction.
 - It loads data from first two memory locations to a specified register.
 - The data of the next two memory location goes to ES register.

(iv) Instruction to transfer flag register

LAHF instruction:

- The LAHF instruction loads the lower 8-bits of the flag register into AH register.
- The lower eight bits of the flag register into AH register.

SAHF instruction:

- The SAHF instruction stores the 8-bit data of AH register into the lower bits of the flag register.

PUSHF instruction:

- It pushes the contents of flag register into the top of stack.
- The PUSHF instruction decrements the stack pointer by two and then store the data of flag register at location pointed by stack pointer (SP).

POPF instruction:

- It pops the data from the first two memory locations pointed by stack pointer into the flag register and then increments SP by 2.