Microprocessor:

- The microprocessor plays a significant role in the everyday functioning of industrialized societies.
- The microprocessor can be viewed as a data processing unit or a computing unit of a computer.
- The microprocessor is a programmable integrated device that has computing and decision making capability similar to that of the CPU (central Processing Unit) of a computer.

Micro-processor
- Small (in physical size)
- which can process the given commands and data

"A microprocessor is a multipurpose, programmable, clock driven, register based electronic device that reads binary instructions from the storage device called memory, accepts binary data as input, processes data according to those instructions, and provides results as output."

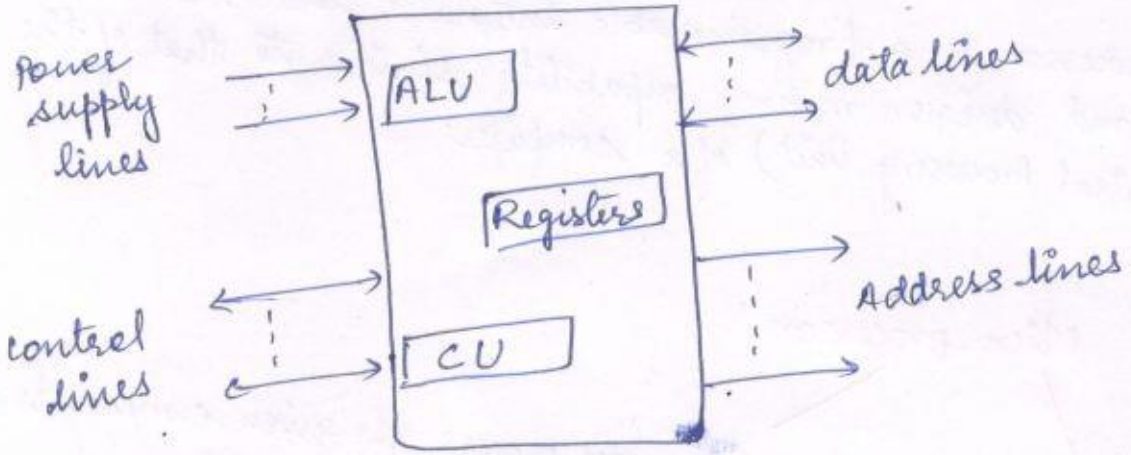
* microprocessor: small size more computing power

→ The microprocessor communicates and operates in the binary numbers 0 and 1, called bits. Each microprocessor has a fixed set of instructions in the form of binary patterns called a machine language.

-

However, it is difficult for human to communicate in the language of 0's and 1's. Therefore, the binary instructions are given abbreviated names called mnemonics, which form the assembly language for a given microprocessor.

The basic microprocessor chip contains the following components.



ALU: Arithmetic Logic Unit : performs the arithmetic and logic operations.

CU: control Unit: CPU communicates with devices such as memory, input and output. The timing of communication process is controlled by the group of circuits called CU.

Control unit provides timing and control signals to the components of computer to perform the specific tasks.

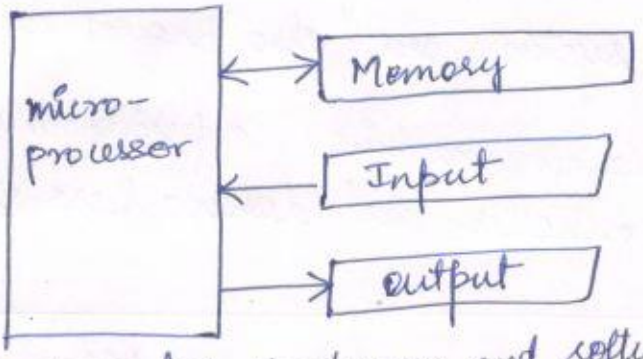
Registers: holds data temporarily during any operation.

Microcomputer: A computer with a microprocessor as its CPU is known as microcomputer.

Programmable Machine:

A typical programmable machine can be represented with four components

- Microprocessor
- Input
- output
- memory



This system comprises two components: Hardware and software.

Hardware: Physical components of the system.

Programs: A set of instructions written to perform a specific task.

Software: A group of programs.

This system can be programmed according to various purposes depending upon the applications.

Microprocessor Applications:

Microprocessor applications are classified into two categories.

- Reprogrammable Systems
- Embedded Systems

Reprogrammable Systems:

In reprogrammable systems, the microprocessor is used for computing and data processing. These systems includes general purpose microprocessors capable of handling large data, mass storage devices and peripherals.

Example: Microcomputer

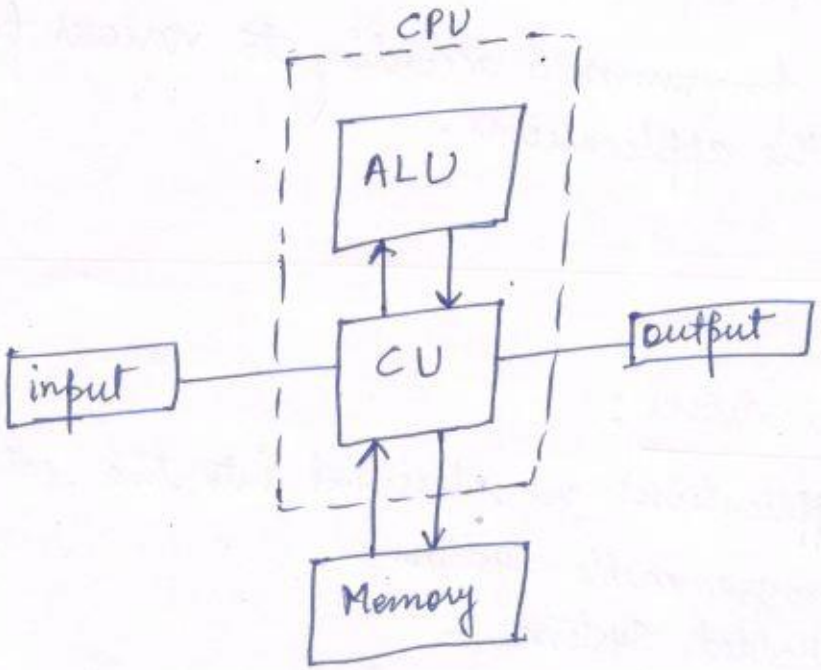
Embedded Systems:

- In embedded systems, the microprocessor is the part of final product and is not available for reprogramming.
- Embedded systems are also known as microprocessor based system.

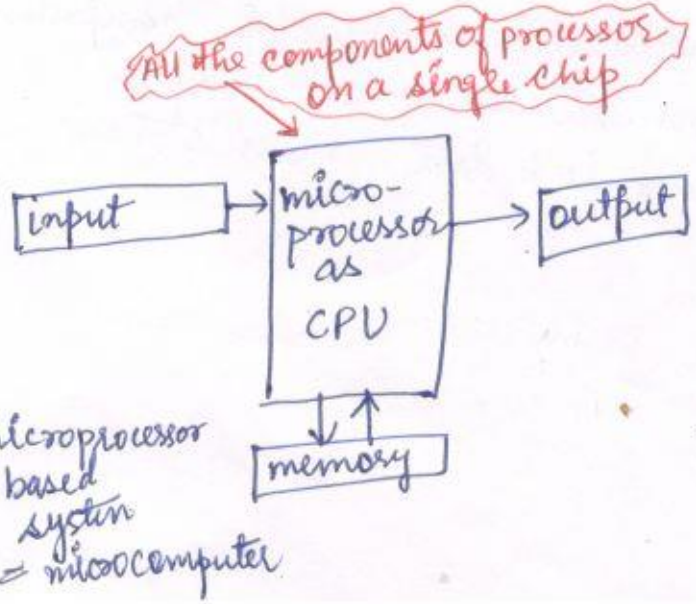
Example: Washing machine, copying machine, traffic light controller and dishwashers etc.

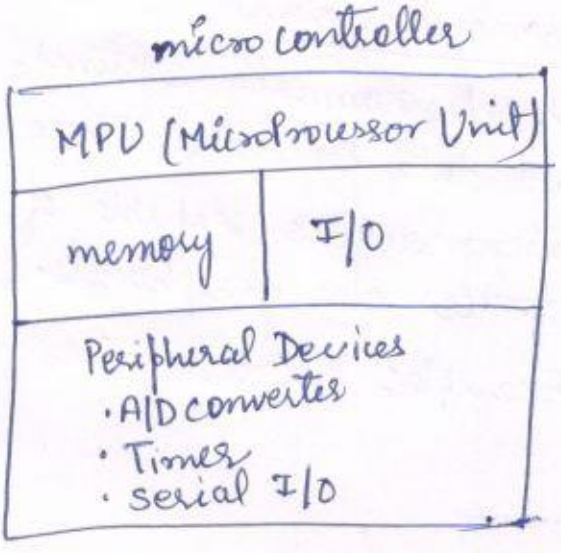
CPU vs. Microprocessor vs. Microcontroller

CPU : can have its components on separate boards



CPU based system





Microprocessor is similar to the CPU but the difference is in size.
In microprocessor, a complete processing unit with the necessary control signals must be on a single chip while previously the components of the CPU can be on separate boards.

Microcontroller:

→ a MPU (microprocessor), memory and I/O interfacing circuit on a single chip.

→ A microcontroller also includes additional devices such as A/D converter, serial I/O, and timer etc.

→ A microcontroller or microcontroller unit is essentially an entire computer on a single chip.

Scales of Integration

Integrated circuits (ICs) or chips contain a large number of logic gates.

- SSI (small scale Integration) < 12 gates/chip
- MSI (Medium scale Integration) < 100 gates/chip
- LSI (Large Scale Integration) < 10^4 gates/chip
- VLSI (Very Large scale Integration) > 10^5 gates/chip
- VLSI as of today

Multicore System:

→ A multicore processor is a computer processor IC (integrated circuit) with two or more separate processing units, called cores.

→ These cores can individually read and execute program instructions

→ These cores work in such a that it feels like the computer system has several processors but in reality, those are cores not processors.

→ It increases the overall speed of program execution.

Multiprocessor System:

Two or more processors present in the same computer, sharing the bus, memory and other I/O is said to be multiprocessing system

Difference between multicore and multiprocessor system

Multicore	multiprocessor
① A single CPU with two or more independent processing units called cores that are capable of reading and executing a program instructions.	A system with two or more CPUs that allows simultaneous processing of programs.
② Executes a single program faster	Executes multiple program faster.
③ Not as reliable as multiprocessor.	More reliable since failure in one CPU with not affect the other
④ Cheaper	costly.
⑤ Have less traffic	Have more traffic.

Microprocessor classification: (or Microprocessor Types)

Based on their specification, application and architectures.

Based on size of data bus microprocessor are:

- → 4-bit microprocessor (Intel 4004)
- → 8-bit microprocessor (8008, 8080, 8085, Motorola 6800)
- → 16-bit microprocessor (8086, 8088, Zilog Z800, 80186, 8028
- → 32-bit microprocessor (Intel 80386, 80837, 80486, Pentiu

Based on application:

- → General Purpose
- → Microcontroller
- → special purpose.

- General purpose microprocessor - used in general computer system and can be used by the programmer for any application examples 8085 to pentium
- Microcontroller - microprocessor with built in memory and ports and can be programmed for any generic control applications. example 8051
- Special purpose processors - designed to handle special functions required for an application. Examples, digital signals processors, application specific circuit chips (ASIC) chips

Based on architecture:

- Reduced Instruction set computer (RISC) processors
- complex Instruction set Computer (CISC) processors.

Generation of Microprocessors (Evolution of Microprocessors).

First Generation

- 1971 to 1973
- Intel created its first microprocess 4004 in 1971.
- During this period, the other microprocesor in market: PPS-4, INTEL 8008 IMP-16 were in use.
- Instructions of these microprocessor were processed serially.
- slow speed.

Second Generation:

- 1973 to 1978
- 8-bit microprocessors.
- Motorola 6800 and 6801, INTEL 8085 and Zilog's Z80.
- based on NMOS Technology (N-type Metal Oxide Semiconductor)

Third Generation:

- 1979 to 1980
- 16-bit microprocessors.
- designed using HMOS technology.
- INTEL 8086/80186/80286 and Motorola 68000 and 68010
- speeds of these processors were four times better than 2nd generation processors.

Fourth Generation:

- 1981 - 1995
- 32-bit microprocessors.
- using HCMOS technology.

- INTEL 808036 and Motorola 68020 & 68030

Fifth Generation:

- from 1995 -
- 64 bit microprocessor.
- high performance and high speed.
- Such processors include Pentium, Celeron, Dual, Quad-core processors.

Applications of Microprocessor:

- Microprocessor is a multipurpose device which find applications in almost all the fields. Some sample applications given in variety of fields:

→ Electronics:
- Digital clocks and watches.
- Mobile phones.
- Measuring meters.

→ Mechanicals:
- Automobiles
- All remote machines

→ Electrical:
- Motors
- Lighting controls
- Power stations

→ Medical
- patient monitoring
- Medical Equipments.

→ Computer:
- Laptops
- Modems
- Scanners & printers

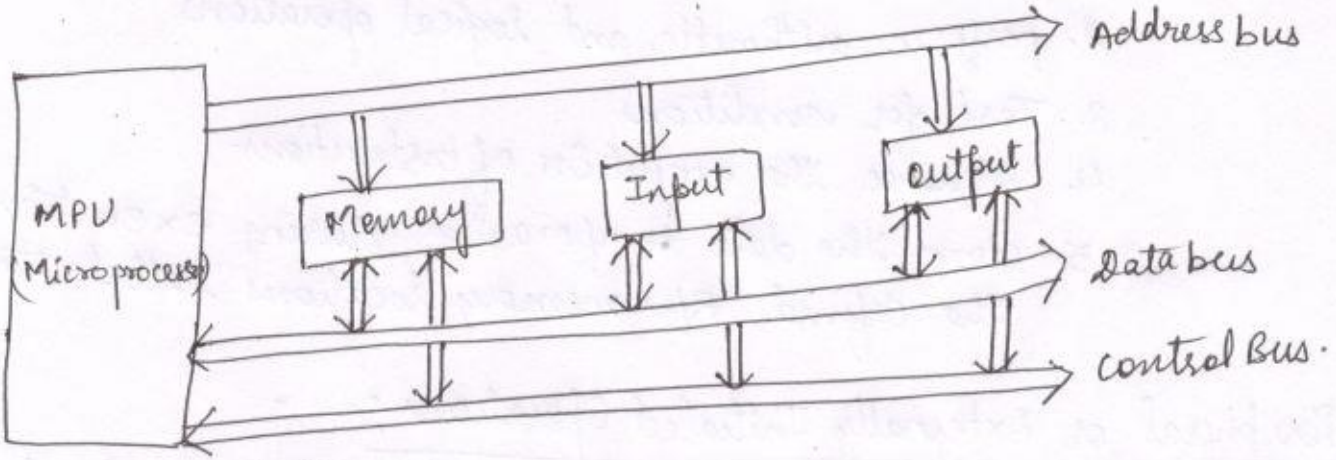
→ Domestics:
- → Microwave ovens
- Television/CD/DVD players
- Washing Machines
- Dishwashers.

Microprocessor Architecture and its operations

- The microprocessor is a programmable digital device, designed with registers, flipflops, and timing elements.
- The microprocessor has a set of instructions, designed internally, to manipulate data and communicate with peripherals.
- This process of data manipulation and communication is determined by the logic design of the microprocessor, called the architecture.

All the various functions performed by the microprocessor can be classified in three general categories.

- Microprocessor initiated operations
- Internal operations
- Peripheral (or externally) initiated operations



General Architecture

Microprocessor Initiated Operations:

The MPU performs primarily four operations

- Memory Read: Reads data (or instructions) from memory.
- Memory Write: writes data or instructions into memory.

- Input Read : accepts data from input devices.
- Output Write : Sends data to output devices.

To communicate with a peripheral (or a memory location), the MPU needs to perform the following steps.

Step1: identify the peripheral (or memory location (with its address.)

Step2: Transfer the binary information (data and instructions)

Step3: provide timing and synchronization signals.

Internal operations:

The microprocessor determines how and what operations can be performed with the data. These operations are:

1. Store 8-bit data
2. perform arithmetic and logical operations
3. Test for conditions
4. Sequence the execution of instructions.
5. Store the data temporarily during execution in the defined R/W memory locations called stack.

Peripheral or Externally Initiated Operations:

External devices (or signals) can initiate the following operations

- Reset
- Interrupt
- Ready
- Hold

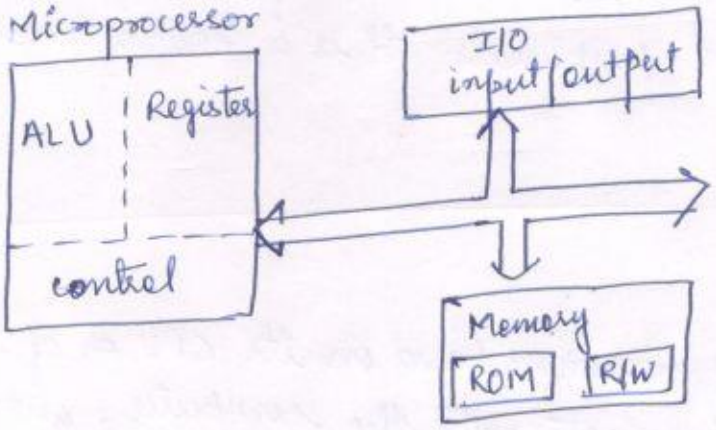
Reset: when the reset pin is activated by an external key (also called a reset key), all the internal operations are suspended and the program counter is cleared (it holds 0000H).

Interrupt: The microprocessor can be interrupted from the normal execution of instructions and asked to execute some other instructions called a service routine. The microprocessor resumes its operations after completing the service routine.

Ready: If the signal at the Ready pin is low, the microprocessor enters into wait state. This signal is used to synchronize slower peripherals with the microprocessor.

Hold: when the HOLD pin is activated by the external signal, the microprocessor relinquishes control of buses and allows the external peripheral to use them.

Organization of a microprocessor based system



→ Organization of microprocessor based system includes three components microprocessor, (Input/output) I/O and memory (read/write and Read only).

→ These components are organized around a common communication path called a bus.

→ This entire group of components is referred to as a system or a micro computer system.

→ A microprocessor is a clock-driven semiconductor device consisting of electronic logic circuits manufactured by using either a LSI or VLSI technique.

→ ALU performs arithmetic operations (addition and subtraction etc) and logical operations as AND, OR and X-OR.

→ Register Array :- various registers primarily used to store data temporarily during execution of a program and are accessible to the user.

→ The control unit provides the necessary timing and control signals to all the operation in microcomputer.

→ Memory stores instruction and data, and provides that information to the microprocessor whenever necessary. It includes Read only memory and Random Access Memory (R/w Read Write Memory)

→ I/O (Input/output): microprocessor communicates with outside outside world. These devices are also known as peripherals.

→ System Bus: The system bus is a communication pathway between microprocessor and peripherals. It is a group of wires carrying bits.

---

computer languages:

→ Each machine has its own set of instructions based on its CPU or of its microprocessor. To communicate with the computer, one must give the instructions in the binary language called machine language.

machine language ⇒ 0 and 1 language

→ Because it is difficult for the most people to write programs in set of 0's and 1's, compiler manufactures have devised English-like words to represent the binary instructions of a machine. This language is called assembly language. These english like word is called Mnemonics.

→ Because an assembly language is specific to a given machine, programs written in assembly language are not transferable from one machine to another.

→ To circumvent, this limitation, such general purpose language as C, C++, and Java etc, called high level language, have been devised.

machine language and Assembly language are microprocessor specific and considered both as low level language.

Writing and Executing an assembly Language Program:

→ steps:

1. write the instructions in mnemonics obtained from the instruction set supplied by the manufacturer.
2. Find the hexadecimal machine code for each instruction by searching through the set of instructions.
3. Enter (load) the program in the user memory in a sequential order by using the Hex keyboard as the input device.
4. Execute the program by pressing the Execute key. The answer will be displayed by the LED's.

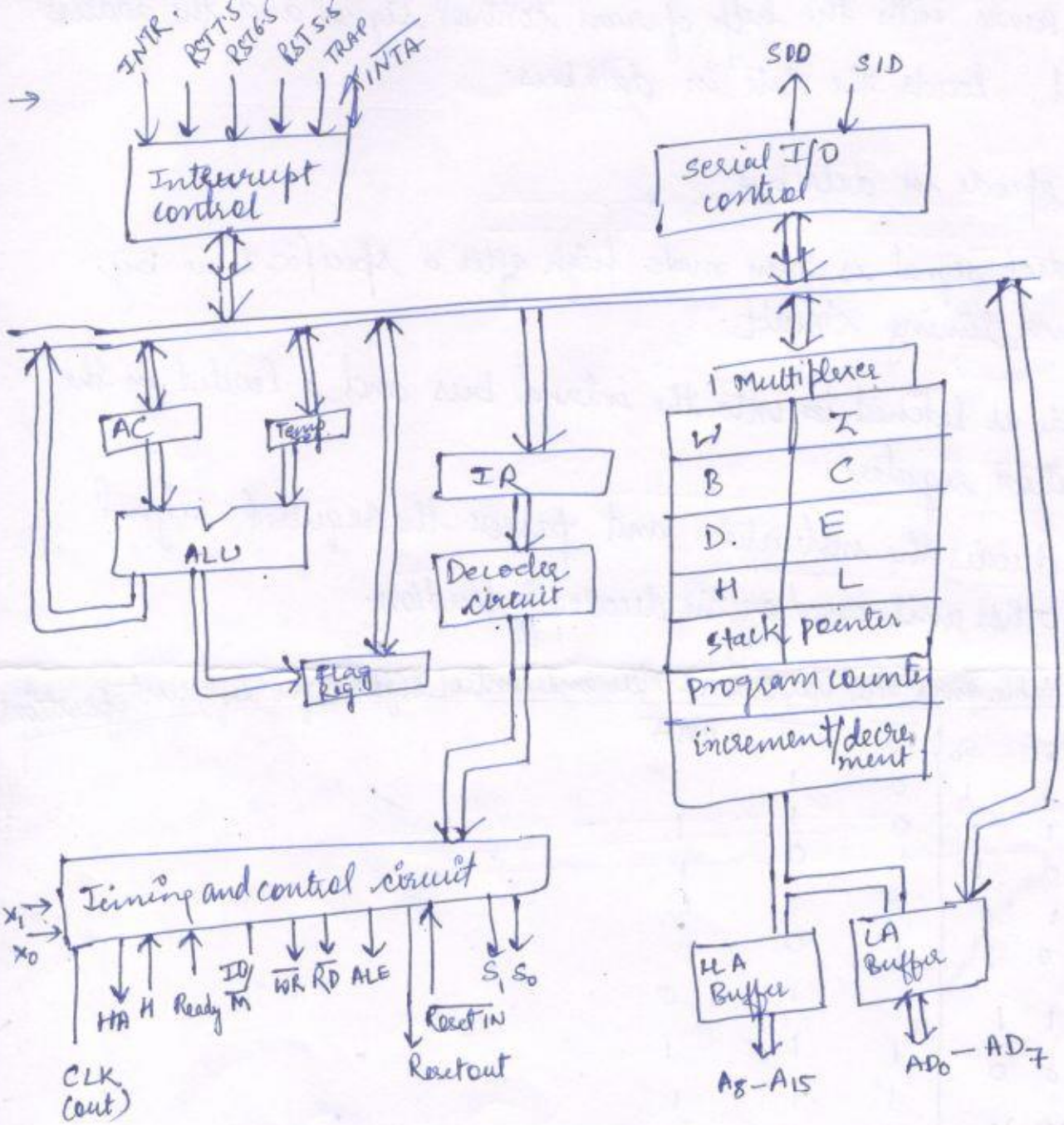
This procedure is called either Manual or hand assembly.

Assembler: The assembler is a program that translates the mnemonics entered by the ASCII keyboard into the corresponding binary machine codes of the microprocessor.

* Each microprocessor has its own assembler because the mnemonics and machine codes are specific to the microprocessor being used. Each assembler has rules that must be followed by the programmer.

INSTRUCTION and DATA FLOW

→ How the instruction and data flow for the normal operation of microprocessor.
→ instruction is a binary pattern that instructs a microprocessor to perform a specific task.



Architecture of 8085 microprocessor.

- PC is loaded with starting address
- 8085 loads the address in PC to the address bus and makes the read contr low.
- PC is then incremented to the next ~~add~~ instruction address.
- memory reads with the help of read control signal and the address provided, loads the data in data bus.
- This is opcode in data bus
- Read control signal is then made high after a specific time by control and timing circuit.
- The opcode is latched ~~in~~ onto the internal bus and loaded in the instruction register

→ Decoder decodes the instruction and passes the required signal to the other unit based on the decoded instruction.

→ Following table gives the various status and control signals for different operati

	IO/$\overline{M}$	S_1	S_0	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
opcode F	0	1	1	0	1	1
Mem R	0	1	0	0	1	1
Mem W	0	0	1	1	0	1
IO R	1	1	0	0	1	1
IO W	1	0	1	1	0	1
Interrupt Ack	1	1	1	1	1	0
HLT	Z	0	0	1	1	1
HOLD	Z	X	X	Z	1	1
Reset	Z	X	X	1	1	1

⇒ Z is a tristate pin neither connected to Vcc or ground (H, L, High impedance

⇒ X → don't care

Timing Diagram:

- Helps to understand the process of microprocessor step by step with respect to time.
- It is graphical representation of process in step with respect to time.
- The timing diagram represents clock cycle and duration, delay, content of address bus and data bus, type of operation i.e Read/write, status signals.

Important term related to timing diagrams:

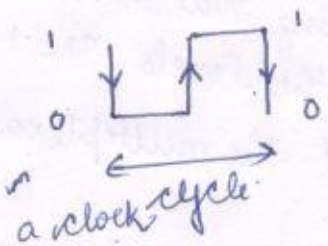
instruction cycle: This term is defined as the number of steps required by the CPU to complete the entire process i.e fetching and execution of one instruction.

machine cycle: It is the time required by the microprocess to complete One operation of accessing memory or I/O devices.

In machine cycle: various operation like opcode fetch, memory read, memory write, I/O Read and I/O write are performed.

T-state: Each clock cycle is a called a t-state. T-state is defined as one subdivision of the operation performed in one clock period.

clock cycle: is a single electric pulse of a CPU.



* For timing diagram, it is important to know how many bytes in an instruction
  - one byte
  - Two bytes instruction
  - Three byte instruction.

Each instruction in 8085 consists two parts
- opcode
- operand.

opcode: specifies which operation needs to be performed.
operand: specifies the data on which the operations to be performed.

To execute a program, 8085 performs various operations
- opcode fetch
- operand fetch
- memory read/write
- I/O read/write

Instruction cycle = Fetch cycle + Execute cycle

External communication functions are
- memory read/write
- I/O read/write
- Interrupt request acknowledge.

opcode fetch Machine cycle:

This is the first step in the execution of any instruction.

T1 clock cycle
- The content of PC is placed in the address bus; AD_0-AD_7 contains lower bit address and A_8-A_{15} contains higher bit address
- I/O $\overline{M}$ signal is low indicating that a memory location is accessed
- S1 and S0 also changed to the levels S1=1, S0=1
- ALE is high indicates that the multiplexed AD_0-AD_7 act as lower order address bus.

T2 clock cycle:
- Multiplexed bus now changed to data bus.
- $\overline{RD}$ signal is made low by the processor.
- ALE is made Low, indicates that multiplexed AD_0-AD_7 acts as data bus.

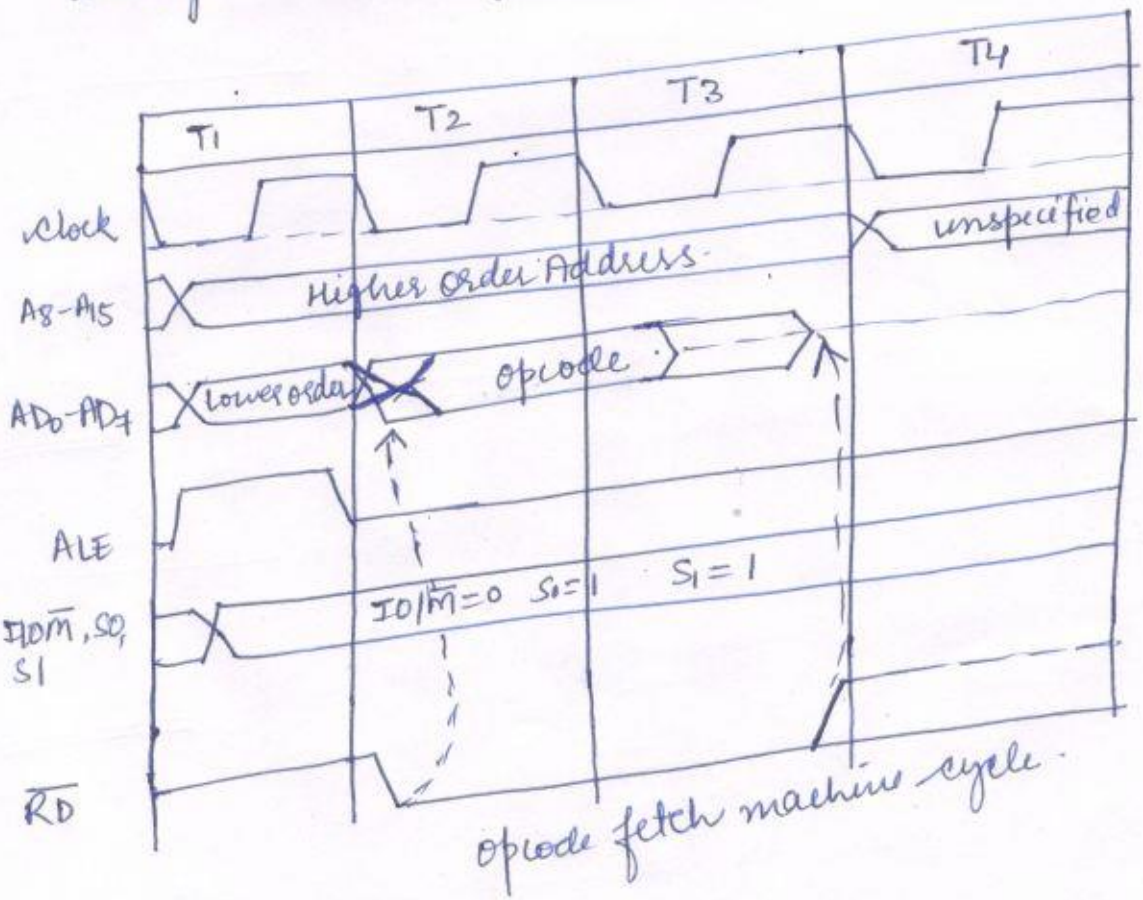
- $\overline{RD}$ signal makes the memory device load the data bus with the contents of location addressed by the processor.

T3 clock cycle:

- The opcode available on the data bus is read by the processor and moved to the instruction register.
- $\overline{RD}$ signal is deactivated by making it at logic 1.

T4 clock cycle:

- The processor decodes the instruction in the instruction register and generate the necessary control signals to execute the instruction
- Based on the decoded instruction further operations such as fetching writing into memory etc take place.



opcode fetch machine cycle.

- Memory Read cycle
- Memory Write cycle
- I/O Read
- I/O Write.

Timing Diagram for MOV Instruction.

Instruction:

A000H MOV A,B

Corresponding coding:

A000H 78

A000H → represents Hexadecimal value.

78 → machine code.

address.

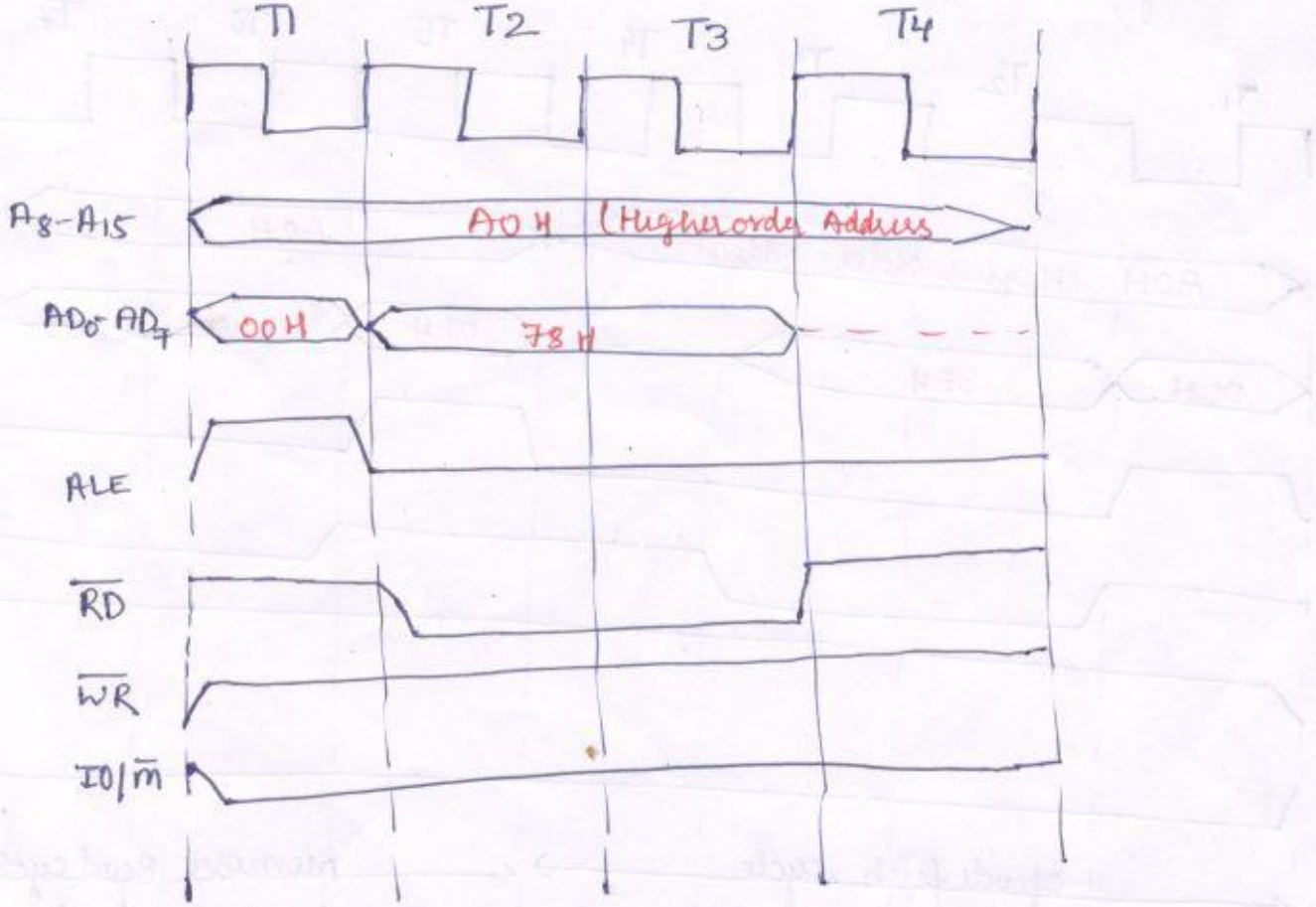
→ This instruction is 1-byte instruction has only opcode

→ For example this instruction is stored at address A000 H.

A0 00

Higher order address

Lower order address.



Timing Diagram for MVI instruction

A000H MVI A, 45H ⇒ Example instruction

→ Meaning of MVI ⇒ Move Immediate

→ with this instruction, we can ~~move~~ load a register with an 8-bit or 1byte value.

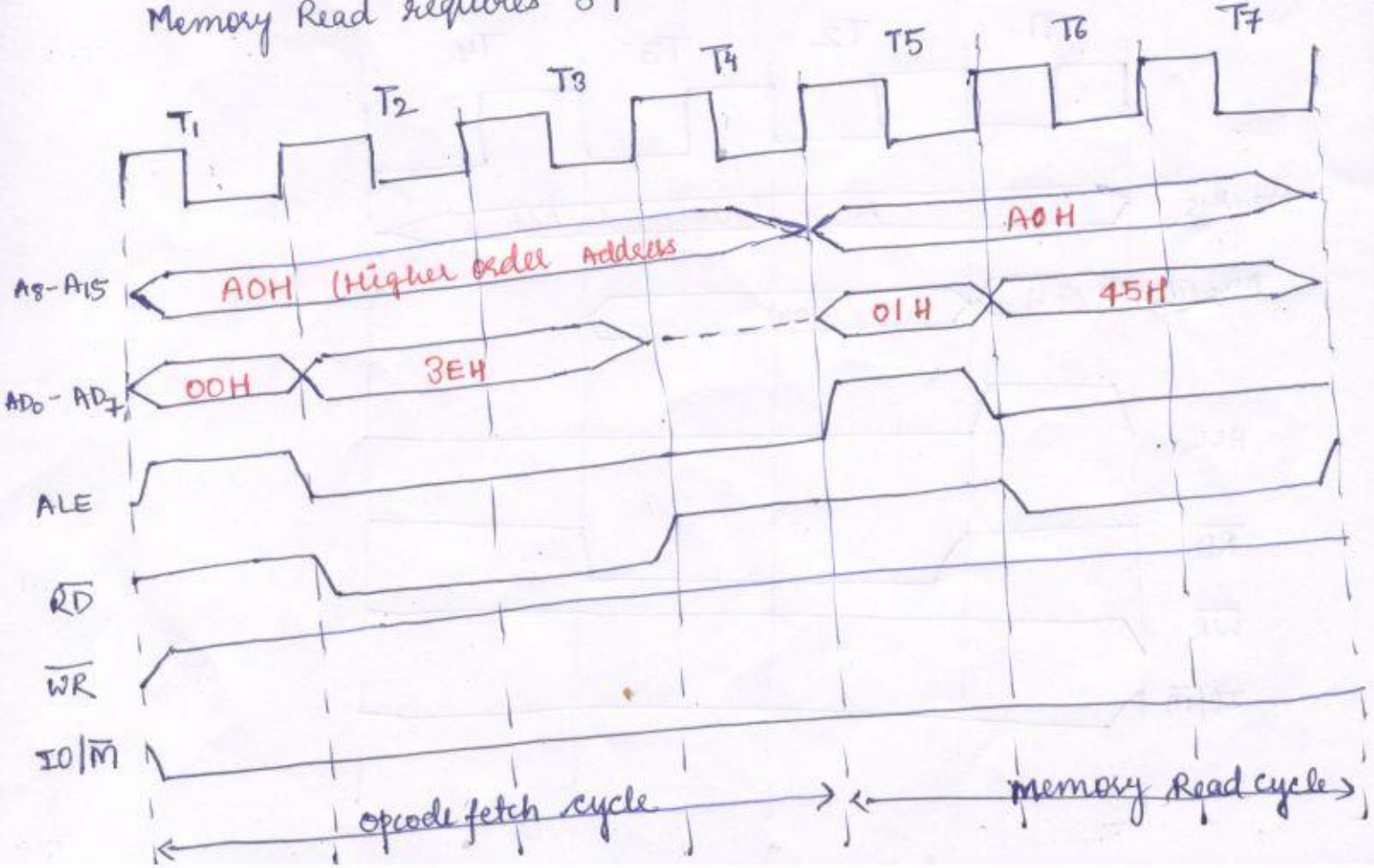
→ This instruction supports immediate addressing mode for specifying the data in the instruction

Instruction:
A000H MVI A, 45H
corresponding coding:

A000H	3E
A001H	45

→ • MVI is 2 byte instruction
• requires two access memory access.
- one for opcode fetch.
- one for data or value fetching.

→ Opcode Fetch requires 4T
Memory Read requires 3T



Timing Diagram for LXI instruction:

instruction LXI A, F045H stored at A000H address.

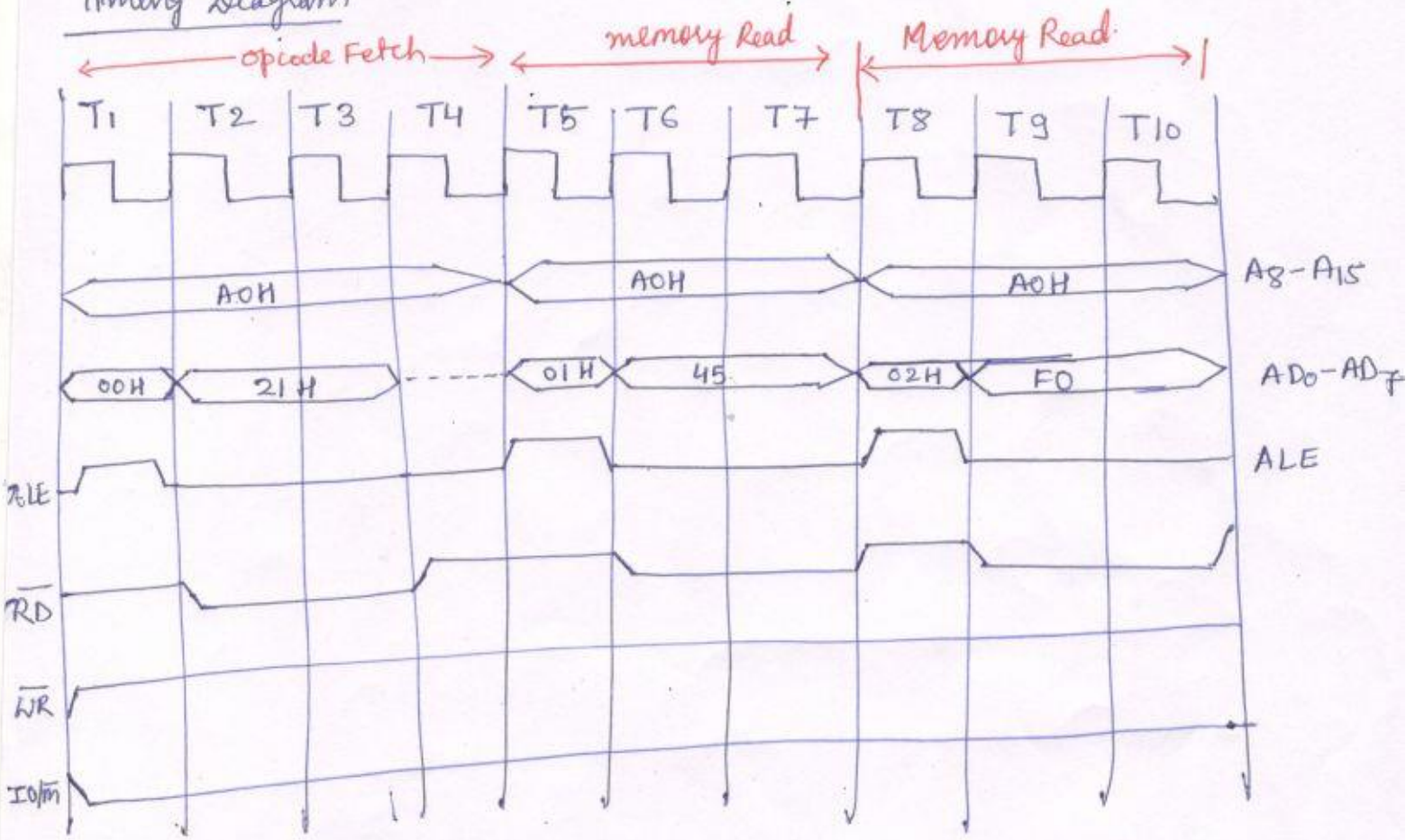
Corresponding Coding

A000H	21 ⟵ 21 is machine code for LXI A,
A001H	45
A002H	F0

* LXI A, means load extended Immediate
F045 is the address from where the data will be loaded into register A.

* LXI A, F045H is 3-byte instruction.

Timing Diagram



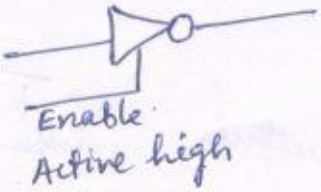
Interfacing Devices:

- Interfacing devices are the semiconductor chips that are needed to connect peripherals to the bus system.
- Interfacing devices are necessary to interconnect the components of a bus oriented system.
- commonly used interfacing devices are:
  - Tristate Devices
  - Buffer
  - Decoder
  - Encoder
  - D Flip flop (latched and clocked)

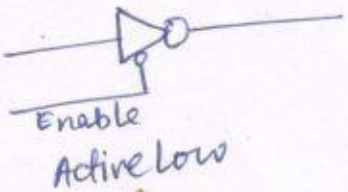
Tristate Devices:

- Tristate logic devices have three states; logic 1, logic 0 and high impedance.
- A tristate logic device has a third line called 'Enable'. When this line is activated, the tristate device functions the same way as ordinary logic devices. When this third line is disabled, the logic device goes into high impedance state - as if it were disconnected from the system.

Example



Tristate inverter



Tristate inverter

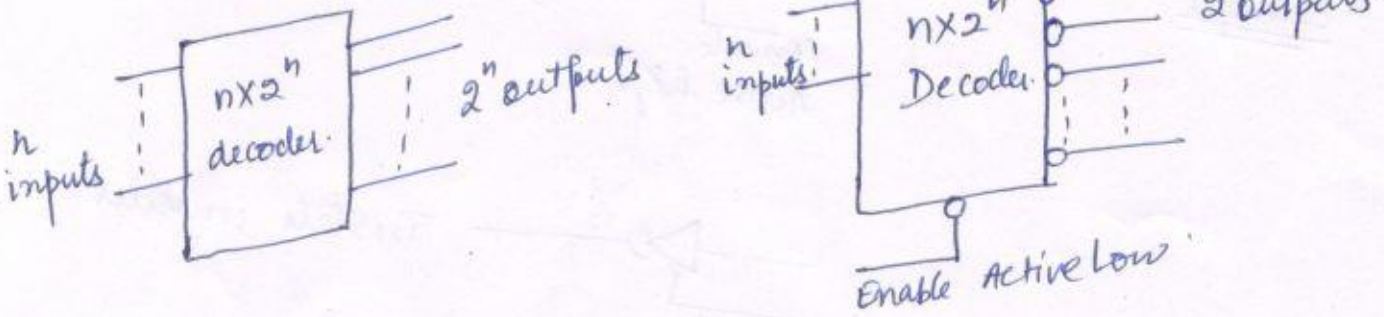
Buffer: The buffer is a logic circuit that amplifies the current or power.

- The logic level of the output is the same as that of the input.
- The buffer is primarily used to increase the driving capability of a logic circuit. (It is also known as driver)

- Tristate buffer, when the enable line is activated, the circuit functions as a buffer, otherwise it stays in high impedance state.

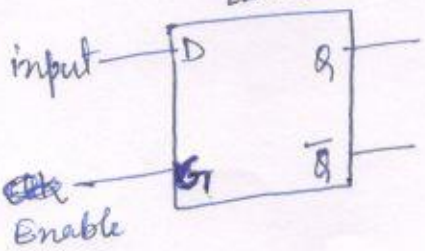
Decoder:

- The decoder is a logic circuit that identifies each combination of the signals present at its input.
- A decoder is a combinational circuit that has n inputs and 2^n outputs.
- In general decoders have active low output lines as well as Enable lines.

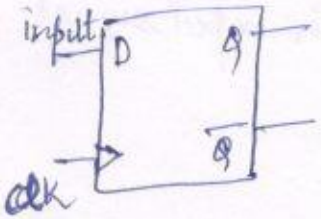


D Flip flops: Latches and clocked

Latch.



Positive edge triggered Flip Flop.

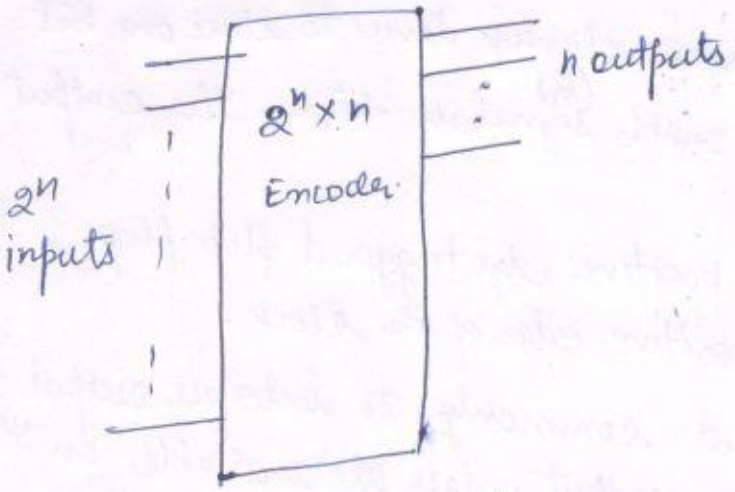


- A latch is a storage device to store one bit.
- In this latch, when enable (G) signal is high, the output changes according to the input D.
- On the other hand, in positive edge triggered flip flop, the output changes with the positive edge of the clock.

A latch is used commonly to interface output devices. When the MPU sends an output, data are available on the data bus only for a few microseconds, therefore, a latch is used to hold data for display.

Encoder:

- The encoder is a logic circuit that provides the appropriate code (binary or BCD) as output for each input signal.
- Reverse of a decoder.
-



- This encoder is unable to provide an appropriate output code if two or more input lines are activated simultaneously.
- Priority Encoders can resolve the problem of simultaneous inputs.

ADDRESSING MODES:

Two issues: ① How is the address of an operand specified?
② How the bits of an instruction organized to define the operand addresses and operation of that instruction.

The operation field of an instruction specifies the operation to be performed. This operation must be executed on some data stored in the computer registers or memory words.
The way the operands are choosen during the program execution depends on the addressing mode of the instructio

The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

Most Common addressing techniques:

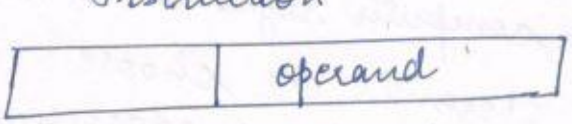
- Immediate Addressing
- Direct Addressing
- Indirect addressing
- Register Addressing
- Register Indirect Addressing
- Displacement → Relative Addressing
  → Base Register Addressing
  → Indexing
- Stack addressing

① Immediate Addressing:

→ In this addressing mode, the operand value is present in the instruction.

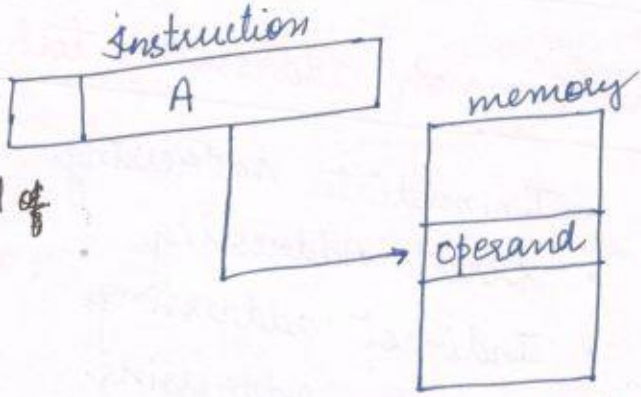
→ Advantage: no memory reference other than the instruction fetch is required to obtain the operand.

→ Disadvantage: Limited operand magnitude.



② Direct Addressing:

In this addressing mode, the address field contains the actual/effective address of the operand.



EA = A

A ⇒ content of an address field of in the instruction

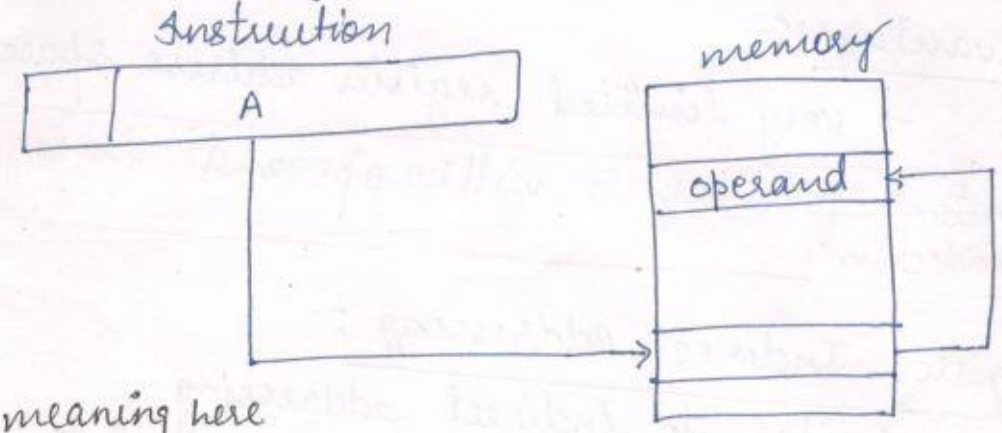
Advantage:

- It requires only one memory reference and no special calculation.
- Simple

Disadvantage: Limited address space.

(3) Indirect Addressing:

In this addressing mode, address field refers to the address of a word in the memory.



EA = (A)

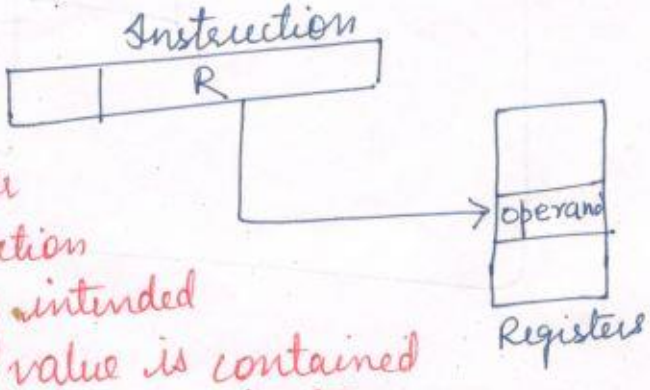
* parentheses meaning here
  ⇒ "content of"
* EA = Actual / effective address of the location containing the referenced operand.

Advantage: Large address space:
→ for a word length of N, an address space of 2^N is now available.

Disadvantage: requires two memory reference
→ one for address of operand.
→ one for to get its value (operand's)

(4) Register Addressing:

Register addressing is ~~similar~~ similar to direct addressing. The only difference is that the address field refers to a register rather than a main memory address.



EA = R

if the content of a register address field in an instruction is 5, then register R5 is intended address, and the operand value is contained in R5.

26/ Advantages:

→ only a small address field in instruction is needed
→ no time consuming memory references.

Disadvantages:

→ very limited register address space.

"Content of the register will be operand" in Register Addressing

⑤ Register Indirect Addressing:

- similar to indirect addressing
- But here, the address field of instruction specifies a register whose contents give the address of the operand in the memory.

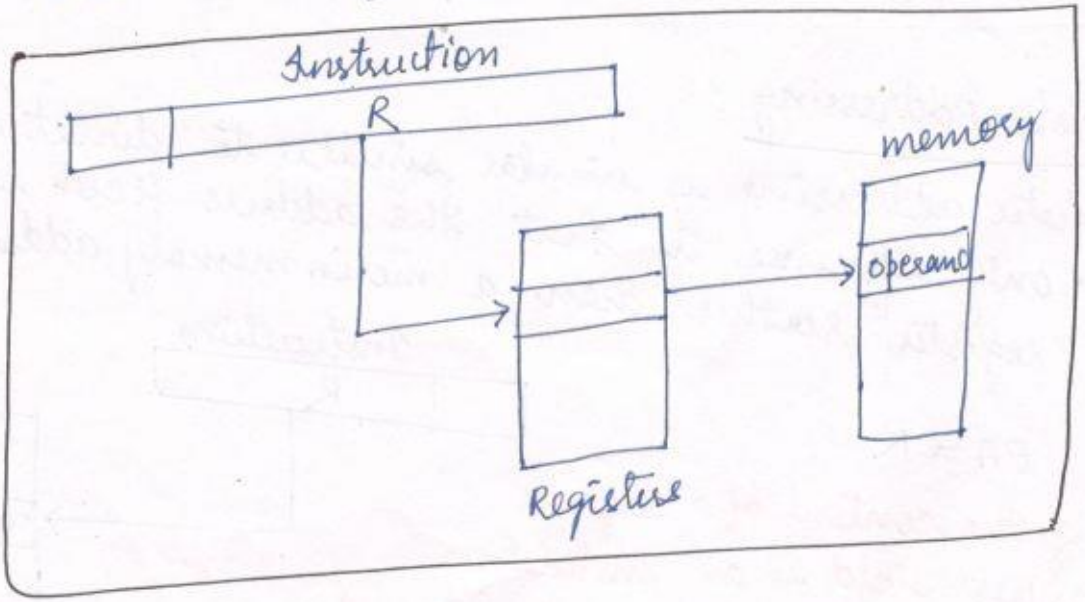
$$EA = (R)$$

Advantage:

→ fewer bits to select a register than a memory addr
→ Large address space.

Disadvantage:

- Extra memory Reference.



(6) Displacement Addressing :

This addressing mode combines the capabilities of direct addressing and register indirect addressing.

$$EA = A + (R)$$

- Three most common uses of displacement addressing
  - → Relative addressing
  - → Base-register addressing
  - → Indexing

Relative Addressing :

- Also called PC-relative addressing
- In this mode the content of the program counter is added to the address part of the instruction in order to obtain the effective address.

Example: assuming PC = 825

address part in instruction = 24

The instruction at location 825 is read from memory during the fetch phase and the program counter is then incremented by one to 826.

The effective address computation for relative address mode is 826 + 24 = 850.

This is 24 memory locations forward from the address of the next instruction.

- often used with branch-type instructions
- saves address bits in the instruction

Effective address = content of Program counter + address in instruction

28/

Base Register Addressing:

- In this mode the content of a base register is added to the address part of instruction to get the effective address.
- A base register is assumed to hold the base address and the address part of the instruction gives a displacement relative to this base address.

Effective Address = content of Base Register + address in instruction

Indexing or Indexed Addressing mode:

- In this mode, the content of an index register is added to the address part of an instruction to obtain the effective address.
- The index register is a special CPU register that contains an index value. The address field of the instruction defines the beginning address of a data array in the memory
- Each operand in the array is stored in the memory relative to the beginning address.

Effective address = content of index Register + address in instructio

(7) Stack Addressing :

: A stack is a linear array of locations.
. A stack is also referred to as a pushdown list or last-in-first-out queue.

- LIFO concept.
- The stack mode of addressing is a form of implied addressing The machine instruction need not include memory reference but implicitly operate on the top of stack.

EA = top of the stack

Advantages: No memory reference
Disadvantage: Limited applicability.

(8) Implied Mode or Implicit mode :

- operands are specified implicitly
- Example Instruction such as CMA. (complement Accumulator)
- In fact all register reference instructions that ~~are~~ use an accumulator are implied mode instructions.
- Zero address instructions in stack organized computer are implied mode instructions

(9) Auto increment or Autodecrement Mode:

30/

The auto increment mode is similar to register indirect mode except that the register (content) is incremented after the execution of the instruction.

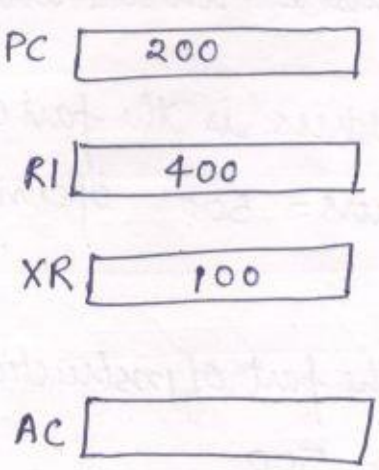
Effective address = content of register
Now increment the content of register $R = R+1$

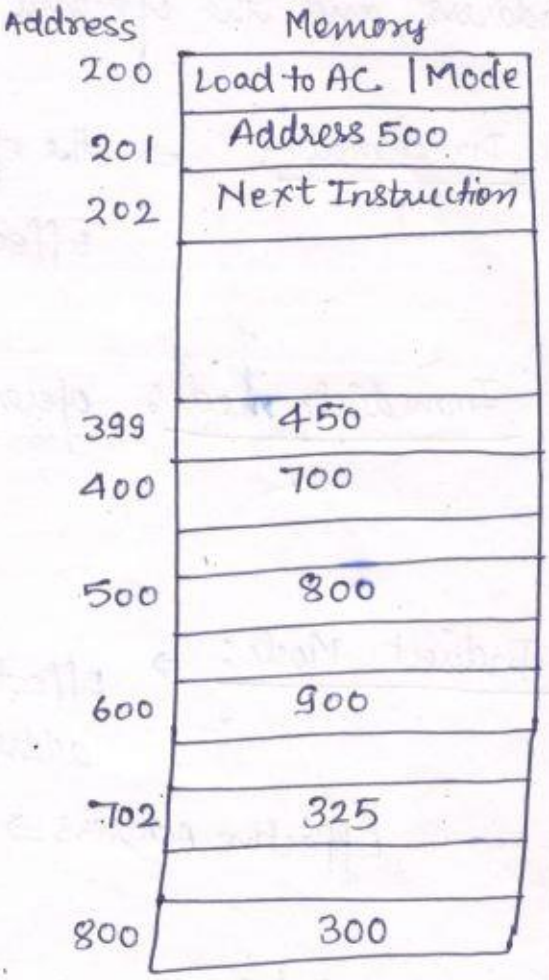
In the auto decrement mode, the content of register is decremented before the execution of instruction.

$R = R-1$
Effective address = content of register

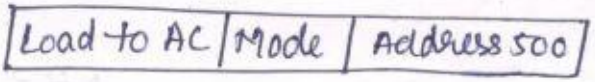
Effective address means ⇒ address of operand

31/ Numerical Example for Addressing Modes:





⇒ The two word instruction at address 200 and 201 is a



⇒ The first word of the instruction specifies the operation code and mode, and the second word specifies the address part.

→ PC (Program Counter) has the value 200 for fetching this instruction

→ The content of process register R1 is 400 and the content of index register XR is 400.

→ AC (Accumulator) receives the operand after the instruction is executed.

32/

⇒ For each possible mode, we calculate the effective address and the operand that must be loaded into AC.

① Direct mode: ⇒ the effective address is the part of instructio

Effective Address = 500 Operand ⇒ 800

② Immediate mode: operand is the part of instruction.

operand = 500

EA ⇒ ~~address~~ 201

③ Indirect Mode: ⇒ Effective address is stored at the address part of instruction '500'

Effective Address ⇒ 800 Operand = 300

④ Relative Mode: Effective address ⇒ content of PC + instructio address part.

EA = 202 + 500

= 702

Operand = 325

⑤ Index Mode: EA = content of index Register + instruction address part.

= 100 + 500

= 600

operand = 900

⑥ **Register Mode:**

⇒ There is no effective address. content of the register R1 will be loaded to AC.

operand = 400

⑦ **Register Indirect mode:**

Register R1 contains the effective address.

EA = 400 . operand = 700

⑧ **Auto-increment Mode:**

same as Register indirect mode, but the content of Register is incremented after the execution of the instruction

EA = 400, operand = 700 , R1 = 401

⑨ **Auto-Decremented Mode:**

Same as Register indirect mode, but the content of Register is decremented before the execution of the instruction

Effective Address = 309 operand = 450

Table

	Addressing Mode	Effective address	operand
①	Direct	500	800
②	Immediate	201	500
③	Indirect	800	300
④	Relative	702	325
⑤	Indexed	600	900

	Addressing mode	effective address	operand
(6)	Register	—	400
(7)	Register indirect	400	700
(8)	Auto increment	400	700
(9)	Auto-decrement	399	450

Modes of Data Transfer (I/O Techniques) (Data transfer Schemes)

- Programmed I/O
- Interrupt Driven I/O
- Direct Memory Access

Programmed I/O

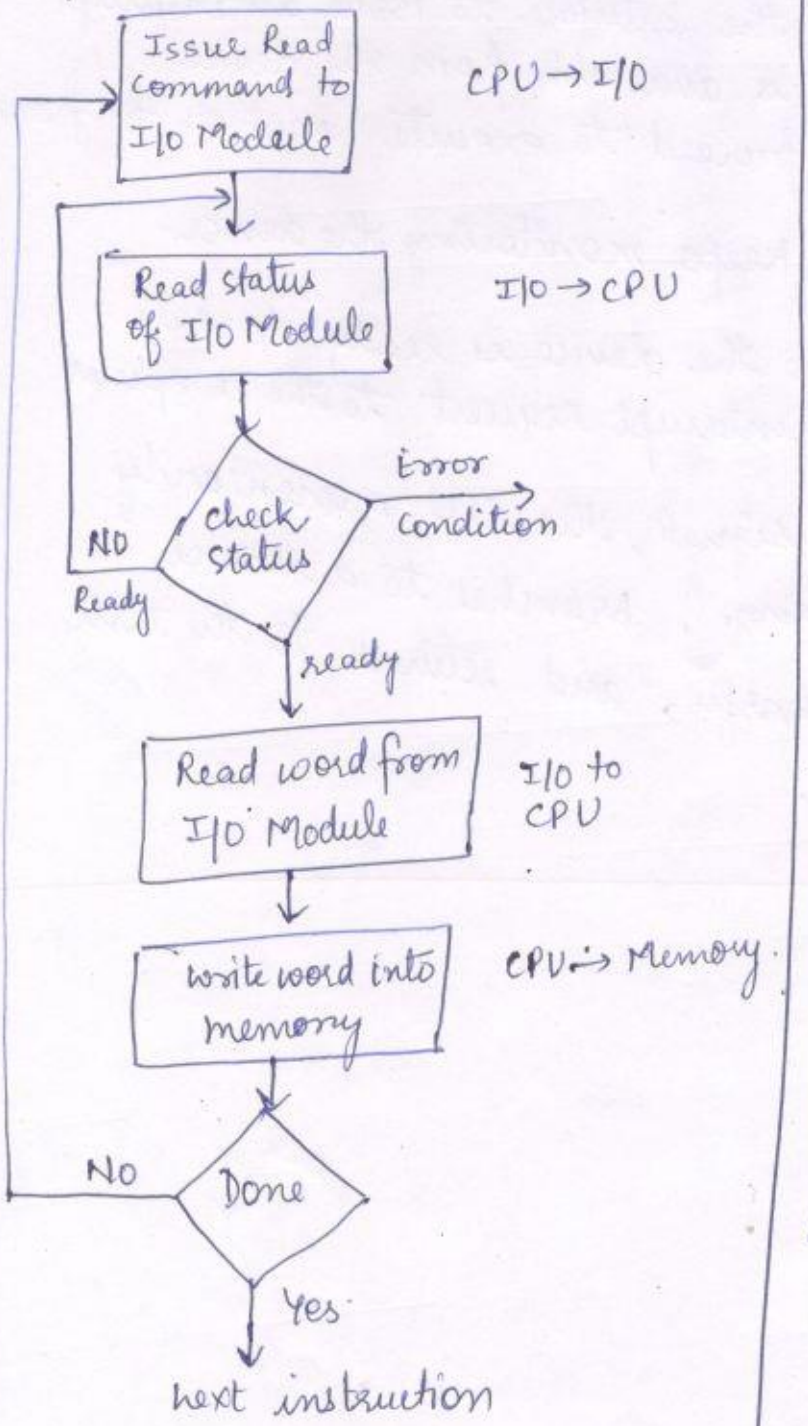
- Input-output instructions written in computer program.
- Each data item transfer is initiated by an instruction in the program
- Transferring data under program control requires a constant monitoring of peripherals by CPU. (once initiated when to transfer

⇒ ~~Once initiated~~

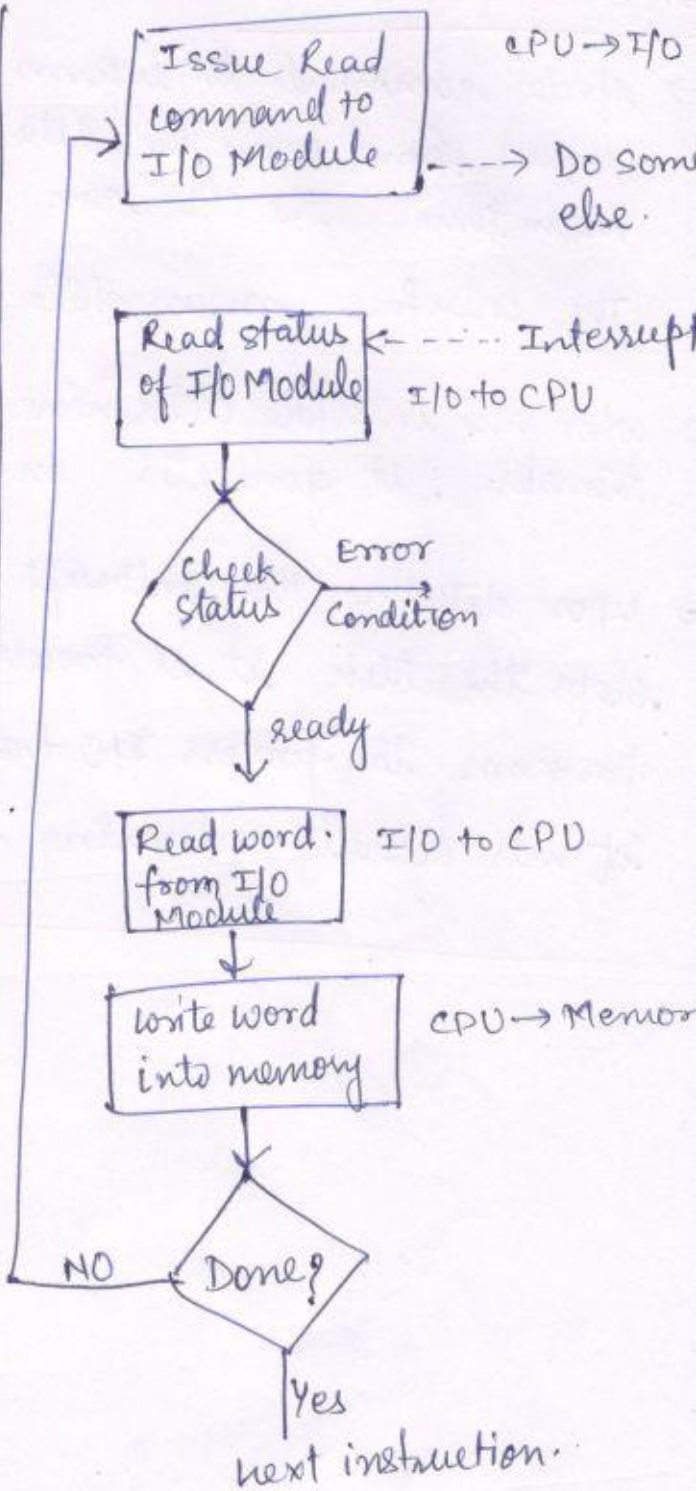
⇒ In this method, the CPU stays in a program loop until I/O unit indicates that it is ready for the data transfer.

⇒ Time consuming

⇒ ~~Less efficient method~~ Less efficiency.



Programmed I/O



Interrupt Driven I/O

Interrupt Driven I/O

→ special commands to inform the interface to issue an interrupt request signal when the data is available from the device meantime the CPU can proceed to execute another progra

The interface meanwhile keeps monitoring the device.

⇒ when the interface determines the device is ready to data transfer, it generates an interrupt request to the computer.

⇒ upon detecting the interrupt signal, the CPU momentarily stops the task it is processing, branches to a service program to process I/O transfer, and returns to the task it was actually performing.

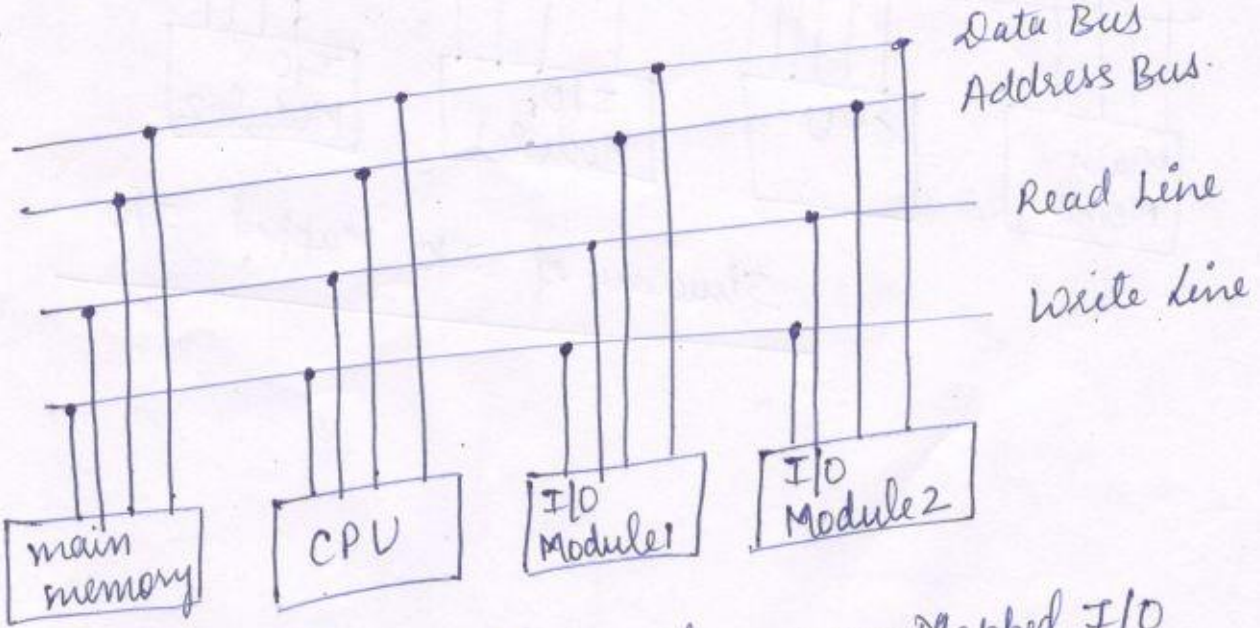
Isolated I/O vs. Memory Mapped I/O

When the processor, main memory and I/O share a common bus, two modes of addressing are possible.

- Memory Mapped I/O
- Isolated I/O (Input-output mapped I/O).

Memory Mapped I/O :-

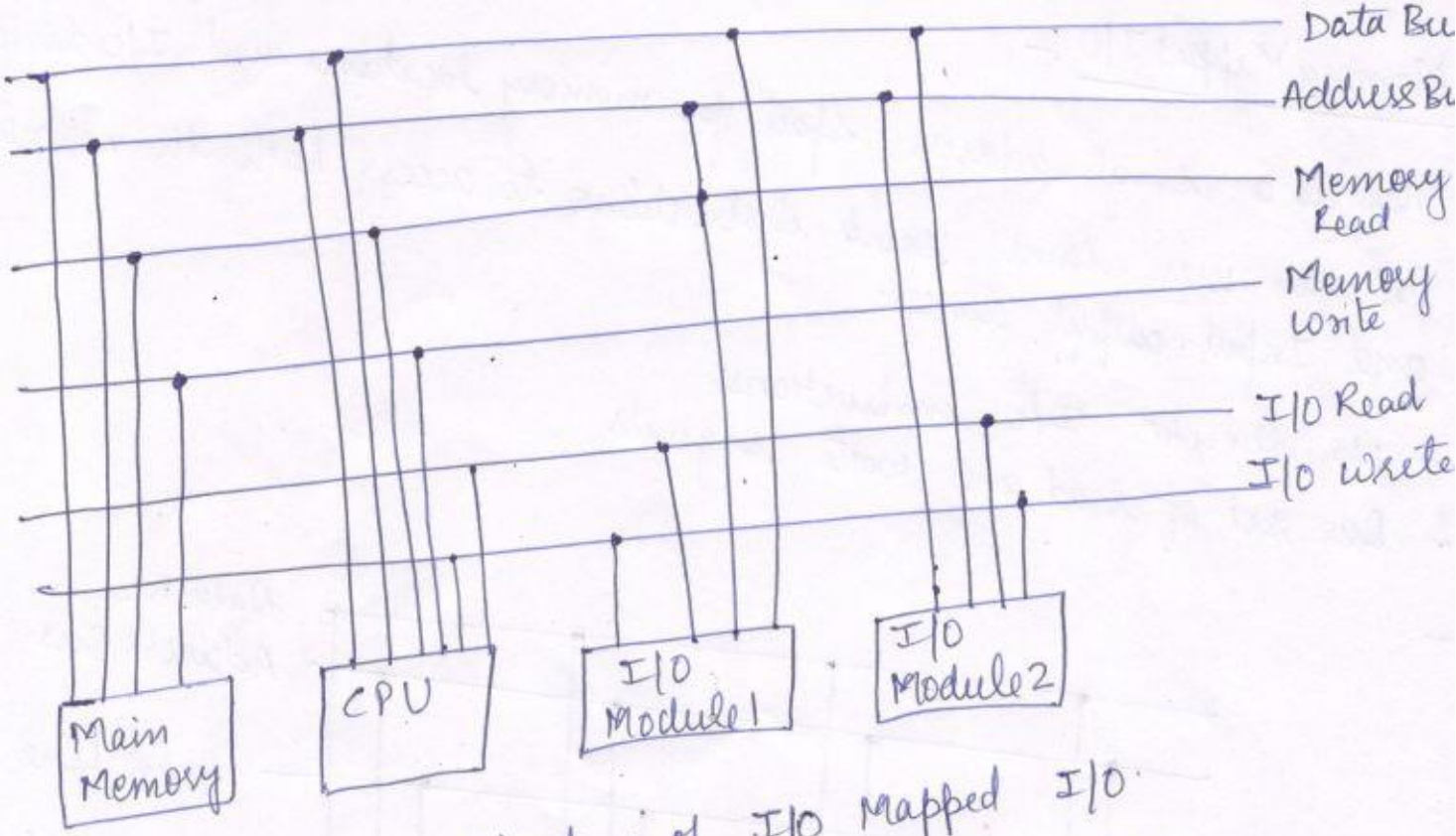
- There is a single address space for memory locations and I/O devices
- Processor uses same read instructions to access both the memory and Input-output devices.
- No specific I/O-instructions.
- One set of read and write signals



General structure of memory Mapped I/O

Isolated I/O or Input-output ~~I/O~~ Mapped I/O

- There are separate address spaces for memory or I/O device.
- CPU specifies that the address on lines is for memory or I/O
- Distinct input output commands.
- ~~separate~~ separate read and write lines.



Structure of I/O Mapped I/O

Comparison between programmed and Interrupt driven I/O

Programmed I/O	Interrupt Driven I/O
→ can be implemented without any additional hardware	→ Additional hardware required for interrupt handling
→ based on busy waiting CPU keeps checking the status of I/O device	→ CPU, switches to another task without waiting
→ Low efficiency	→ higher efficiency than programmed I/O
→ Only one activity can be handled using programmed I/O	→ Multiple I/O activity can be handled in overlapped fashion here

Comparison between Input-output Mapped (Isolated) and Memory Mapped I/O

Memory Mapped I/O	Peripheral Mapped I/O or Input-output Mapped I/O
- same address space for memory and I/O devices.	→ two (separate) address spaces for memory and I/O devices.
- uses same instructions for both I/O and memory operations	→ separate commands for memory operations and different command for I/O.
→ one set of read and write signals for memory and I/O	→ separate read and write signals for memory and I/O
- Memory Mapped I/O is less efficient	→ More efficient