DATE : / /
PAGE NO. :

IIIrd Unit —

✱ CPU —

- CPU is partition into ALU (Birthmetic logical unit) and control unit.
- The function of control unit is to generate relevant timing and control signals to all operations in the comp.
- It controls the flow of data between the processor and memory and perifeller (Connection)

✱ Function of Control unit (CU) —

- The Control unit directs the entire computer system to carry out stored programme instructions.
- The CU must communicate with both the airthmetical logical unit and main memory.
- The CU instructs the ALU that ~~wat~~ which ~~or~~ logical or airthmetic operation to be performed.
- The CU Coordinates the activities of the other 2 units as well as all perifellers and auxilary storage devices link to the computer. All microprocessors are CU.

~~Design of control unit~~

✱ Design of control unit —

- Control unit generate control signals using one of the two organisation

Teacher's Signature

(1) Hardwire control unit —
(2) Microprogramme control unit —

* Microprocessor — (Microprocessing unit)
• Microprocessing unit is synonemus to CPU. Where CPU used in traditional computer and Microprocessor acts as a device or a group of devices which do the following task —
(1) Communicate with the perifeller devices.
(2) Provide timing signals.
(3) It have multiple of circuits.
(4) Recently used microprocessor is 8085.
(5) Direct data flow.
(6) Perform comp. as specified by the instructions in memory.

* 8085 Microprocessor —
• The 8085 Microprocessor is an eight bit general purpose microprocessor which is capable to address 64K of memory this processor has 40 pins.
Requires +5 V signal power supply and 3MHz single phase clock.

Teacher's Signature

1. ~~ALU~~
2. Multiplexer
3.
4. Accumulator
5. Temporary Register
6. ALU
7. Instruction Register
8. Instruction Decoder and machine cycle encoder
9. Timing & control
10. Address Buffer
11. Address & data
12. Flags

(1) ALU ⟶

- The ALU performs the computing function of microprocessor it includes the accumulator, tempory register, Arithmetic and logic circuit, & flag.
- Result is store in accumulator and flags.

(2) Accumulator —

It is an 8-bit register i.e. part of ALU this register is used to store 8-bit data & in performing Arithmetic & logic

Teacher's Signature

operation.
The result of operation is stored in accumulator.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z	X	AC	X	P	X	Cy
				undefined			

(3) Flags —

Flags are programmable they can be used to store and transfer the data from the registers by using instruction. The ALU include 5 flip-flop that are Set & Reset according to the data condition in accumulation and other register.

(a) S → "S (Sign) flag" → After the execution if bit D7 of the result is 1 the sign flag is Set.
On → -Ve Sign
Off → +Ve Sign
It is used to Sign number in a given byte.
If D7 is 1 means negative number if it is 0 then +Ve number

(b) Z (0) flag → The 0 flag is Set if ALU operation result is 0.

Teacher's Signature

(c) Auxilary carry (AC) —
In arithmetic operation when carry is generated by digit D_3 and passed on the digit D_4 the AC flag is Set.
This flag is used only internaly BCD operation.

(d) P (parity flag) →
After Airthmetic and logical operation if result has even no. of 1's the flag is Set.
If it has odd no. of 1's the flag is Reset

(e) Carry flag (cy) →
If airthmetic operation result is in a carry the carry flag is Set otherwise it is reset.

(4) Register Section —
It is basicaly a storage device and transfer data from registers by using instructions

(1) Stack register —
The stag pointer is also a 16-bit register which is used as a memory pointer.
It points to a memory allocation in read/write memory known as stag.

Teacher's Signature

- In between execution of program sometimes
- The beginning of the stag is define by loading a 16-bit address in the stag pointer

* Program counter (PC)

- This 16-bit registers deals with fourth operation to sequence. The execution of instruction.
- This register is also a memory pointer memory allocation have 16-bit address
- It is used to store the execution address.
- The function of the program counter is to point to memory address from which next bite is to be fetched.

Storage register →

- These register stores 8-bit data during a program execution.
- It is in a register pair for eg →
- If we have (B,C), (D,E), (H,L) they will count as 3 not as 6.
- If we have registers like B,C, D,E, H,L then they can be combined as register pair BC, DE, HL to perform some 16-bit operation

Teacher's Signature

SHREE
DATE : / /
PAGE NO. :

* Instructions — Code
- An instruction is binary that specify a sequence of micro operation.
- Instructions are code in memory each computer has its own different instruction set.
- It depends on a processor. The register and address bus which specify an instruction set design.
- An instruction of a computer is a group of operands and opcode

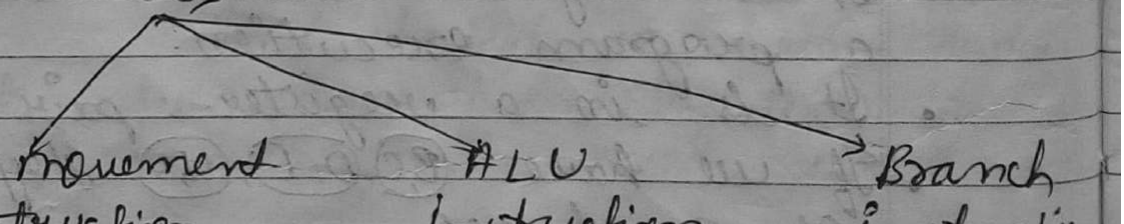
operands →

The operands specify the operation to be performed on which variable or the object on which the operation to be performed

Opcode →

It specifies the operation which is to be programmed

The instruction set can be divided into three classes



Teacher's Signature

(1) Data movement instruction —

- The primary storage for both data & program is the main memory.
- The instruction is to be moved into the CPU from where it is made available to the processor with the help of CPU register.
- In this instruction only the data on which operation is to be performed.
- These data transfer to the relevent register & after processing it is ~~stand~~ send back ~~tends~~ to the main memory.

(2) ALU Instructions →

- All the arithmetical operations and the logical operations which take place in ALU is instructed to ALU instruction
- The ALU operation can be perform on any data stored at any location in the registers and place back the value on to it orginal location with the ALU instruction.
- We always have to use a data movement instruction to take the result on a location.

(3) Branch Instruction

- Branch instruction, in a normal PC by default the instruction cycle increament by one when one instruction is completed this increment is performed to get the

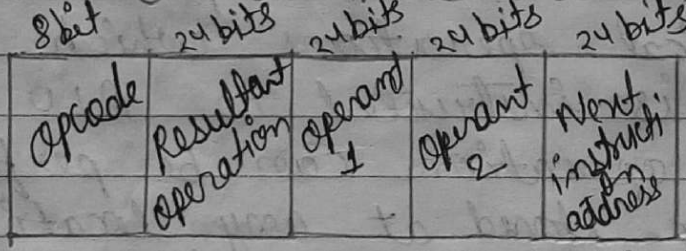
Teacher's Signature

next instruction set but in a normal programming with a conditional have to switch from one sequence to another sequence, to getting this another sequence we have to get the new address from fetching the inst. this overall process is done by the branch instruction.

★ D- Instruction format →

There are 6 different types of instruction format. Instruction format give us a detail of opcode and operand address of instruction.

(1) four address machine →



A four address machine specifies all the 4-items needed in the instruction sequence that are result storing address, 2 operant address and next instruction address with its opcode.

(2) 3- address machine

Stack work on last in first output

The CPU's where PC's registers is added with the CPU the next instruction address is not added in this instruction format.
It is workon on CPU.

(3) 2- address machine —
- In 2- address machine a resultant register is added to the CPU with the PC register so that's why we don't have to use resultant address in this instruction format.

(4) 1- address machine —
- In 1- address machine the instruction for performing one function if added with a operation of loading values from the register.
Here accumulator register is used for the storage of second operat.

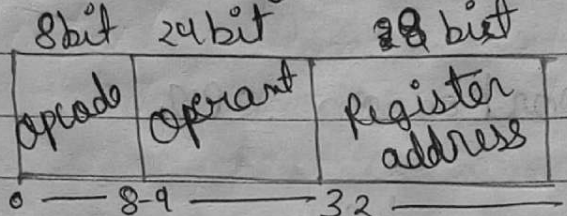
(5) 0- address machine — stack
- In this system a ~~stack~~ is attached to the CPU and using push & pop operation The push instruction is executed and only the opcode is given through instruction.

opcode	Push

(6) General address machine —
It is a register format which is given

Teacher's Sign

in general computer.
It is having an opcode, one operant & next register address.



Addressing Mode —

- When a microprocessor excess any memory unit for reading and writing it must specify the address of the location which it want to access.
- An assembly language instructions can use any type of addressing such as memory address, register address or direct value before decoding the instruction
- We have to encode the ~~instruction~~ address into a instruction format which can be an actual reference or indirect reference.

(1) Indirect addressing mode —

- A constant in the instruction specifies not the address of the value but the address of location where the value is stored.
- ~~Here the value is stored.~~ An eg — implementation of pointers. In this method the address is holding the value of new address where the actual value is store.
- Thus, in this format to access a value we have to access two address

Teacher's Signature

It consumes the time of instruction

(2) Direct Memory location —

- In this mode the memory location is directly representing the actual value. on with we have to operate.
- In this method the accessing time to the variable will we reedeice and we have to access only one address.

(3) Index ~~Method~~ Mode — a[0], a[1] — a[n]

- The memory addresses form by adding a fix constant to create a new address of the memory location.
- In this system constant address is stored in the register and on incremental is used to create or decode the address.
- In a general format this is a method which provide the feel array facility. In this mode we have the facility for giving a common name to a number of location and can access any location with only one fetch value.

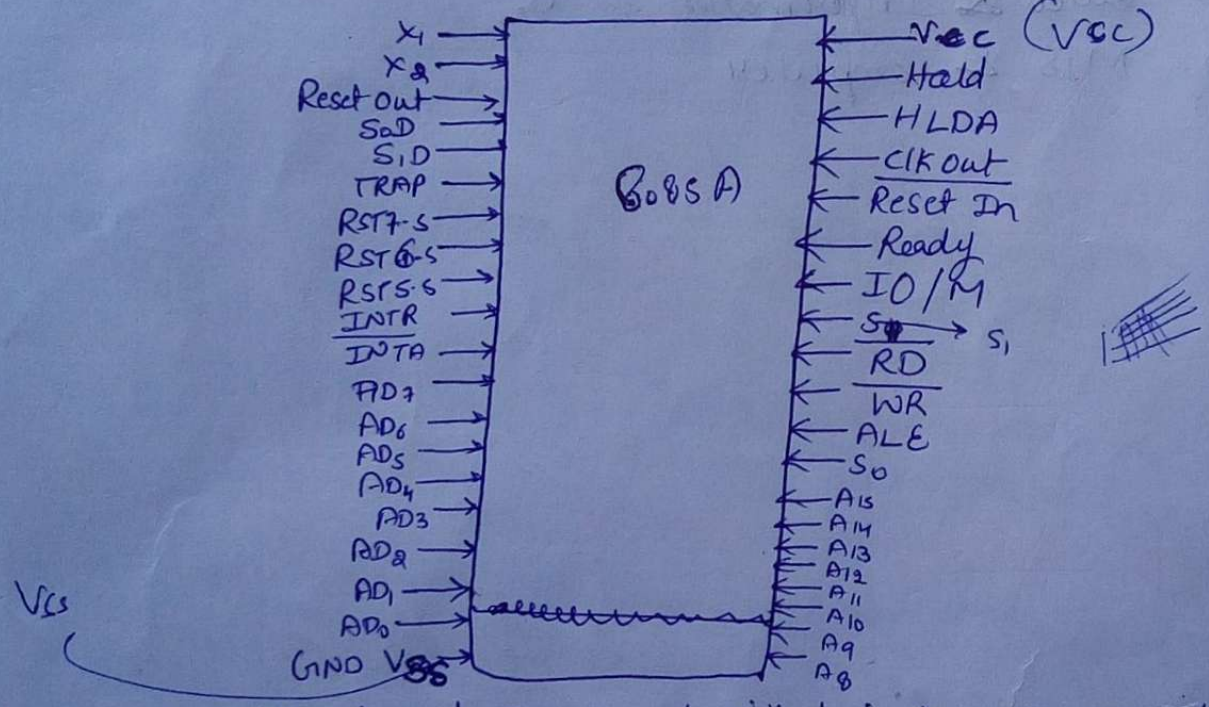
(4) Relative addressing mode —

RAM is similar to index mode but the base address or the constant address is stored in the PC register. This allow us to store memory operant and ~~operant~~ opcode with fix offset from the current expression.

Teacher's Signature

Time and Control Section :→

This unit is responsible to Synchronize Microprocessor operation as per the clock pulse and to generate the control signals which are necessary for smooth communication between Microprocessor and peripherals devices. The RD bar and WR bar signals are synchronous pulses which indicates whether data is available on the data bus or not. The control unit is responsible to control the flow of data between microprocessor, memory and peripheral devices.



* All the signal can be classified into six groups:

S.N.	Group	Description
1.	Address Bus	The 8085 microprocessor has 8 signal line, A_{15} - A_8 which are unidirectional and used as a high order Address bus.
2.	Data bus	The signal line AD7 - AD0 are bidirectional for dual purpose. They are used as low order address bus as well as data bus.

3.	Control Signal & Status Signal	Control Signal :→ RD bar :- It is a read Control Signal (active low). If it is active then memory read the data. WR bar :- It is write Control Signal (active low). It is active when written into selected memory. Status Signal :→ ALE (Address Latch Enable) :→ when ALE is high 8085 microprocessor use address bus. when ALE is low 8085 microprocessor is use data bus. IO/M bar :→ This is a Status Signal used to differentiate between I/O and memory operation when it is high, it indicates an I/O operation and when it is low, it indicate memory operation. S1 and So :- These Status Signals, similar to I/O and memory bar, can identify various operations but they are

		rarely used in Small Systems.
4.	Power Supply and frequency Signal	Vcc - +5v power Supply Vss - ground reference x1x2 - A crystal is connected at these two pins. The frequency is internally divided by two operate System at 3-MHz. the crystal should have a frequency of 6 - MHz CLKout - This Signal can be used as the System clock for other devices.
5.	Externally initiated Signal	INTR (i/p) - Interrupt request. INTA bar (o/p) - It is used as acknowledge interrupt. TRAP (i/p) :- This is non maskable interrupt and has highest priority. HOLD (i/p) :- It is used to hold the executing program. HLDA (o/p) :- Hold Acknowledge. READY (i/p) :- This Signal is used to delay the microprocessor read or write cycle until a slow responding peripheral is ready to accept or Send data.

Through Mathematical formula we will create Vector graphics so we can change it in any shape without breaking and all.

RESET IN pin - when the signal on this pin goes low, the program counter is set to zero, the buses are tri-stated, & MPU is reset.

RESET OUT :- This signal indicate that MPU is being reset. The signal can be used to reset other devices.

RST 7.5, RST 6.5, RST 5.5 (Reuest interrupt) - It is used to transfer the program control to specific memory location. They have higher priority than INTR interrupt.

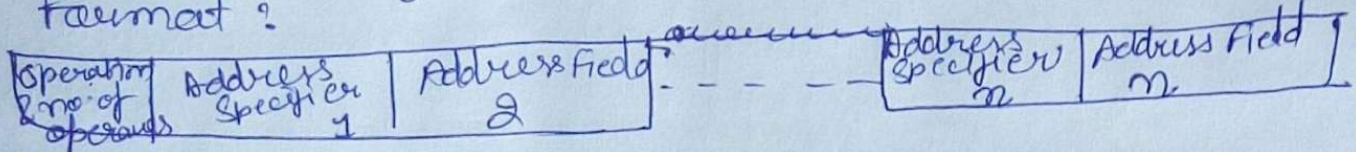
6.	Serial I/P Ports	The microprocessor has two signals to implement the Serial transmission Serial input data and Serial Output data.

Instruction Format :→ Each instruction is represented by a sequence of bits with in the computer. The instruction is divided into group of bits called field. The way instruction is expressed is known as instruction format. It is usually represented in the form of rectangular diax. The instruction format may be of the following types.

Variable Instruction formats :- These are the instruction formats in which the instruction length varies on the basis of opcode & address specifiers. For example, VAX instruction vary between 1 and 53 bytes while X86

instruction vary between 1 and 17 bytes.

Format :



Advantage :-

These formats have good code density.

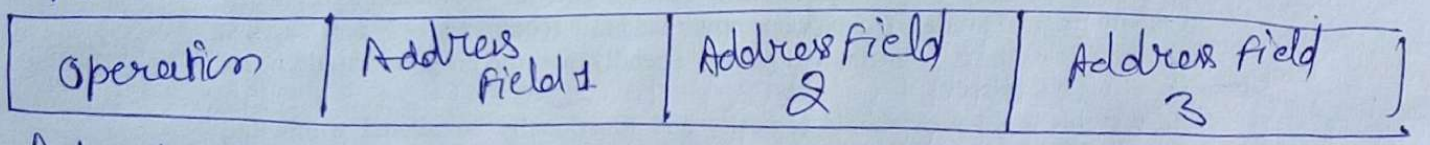
drawback :-

These instruction formats are very difficult to decode and pipeline.

Fixed Instruction formats :-

In this type of instruction format, all instructions are of same size. For example MIPS, PowerPC, Alpha and ARM.

Format



Advantage :-

They are easy to decode & pipeline.

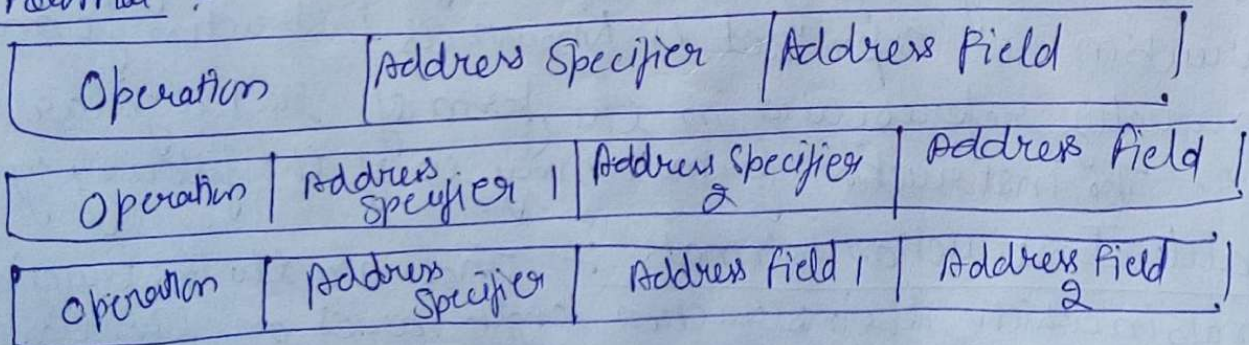
Drawback :-

They don't have good code density.

Hybrid Instruction formats :-

In this type of instruction formats, we have multiple format length specified by opcode, for example, IBM 360/70, MIPS16, Thumb.

Format :-



performance has to be done upon the system's functionality.

Advantage :-

These Compromise between code density & instruction of these type are very easy to decode.

Addressing Modes :-

Addressing mode provides different ways for accessing an address of given data to a processor. Operated data is stored in the memory location, each instruction required certain data on which it has to operate. There are various techniques to specify address of data. These techniques are called Addressing Modes.

- **Direct Addressing Mode** :- In the direct addressing mode, address of the operand is given in the instruction and data is available in the memory location which is provided in instruction. We will move this data in desired location.
- **Indirect Addressing Mode** :- In the indirect addressing mode, the instruction specifies a register which contain the address of the operand. Both internal RAM and external RAM can be accessed via indirect addressing mode
- **Immediate Addressing Mode** :- In the immediate addressing mode, direct data is given ~~the~~ in the operand which move the data in accumulator. It is very fast.
- **Relative Addressing Mode** :- In the relative addressing mode, The effective address is determined by the index mode by using the program counter in stead of general purpose processor register. This mode is called relative address mode.
- **Index addressing Mode** :- In the index address mode, the effective address of the operand is generated by adding a content value to the contents of the register. This mode is called address mode.

Instruction cycle :-

- Instruction is command which is given by the user to Computer.

Instruction cycle:-

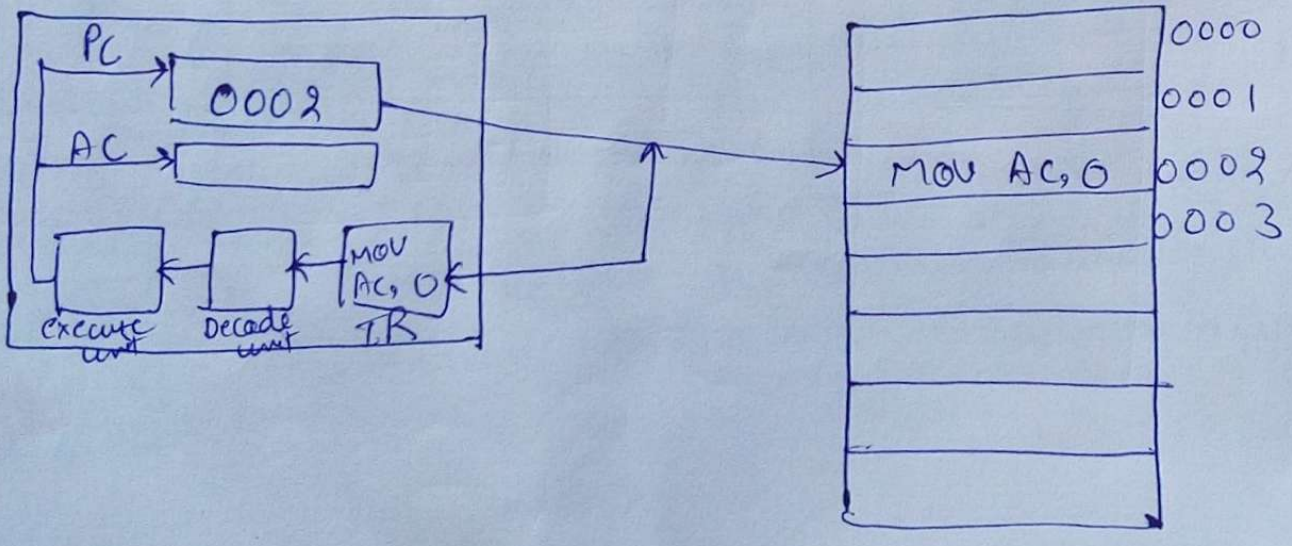
- The time period during which one instruction is fitched from memory and execute when a Computer given an instruction in machine language.
- Each instruction is further divided into seaunce of phases.
- After the execution the program Counter is increamented to point to the next instruction.

Phases :-

- Fetch an instruction from memory.
- Decode the instruction
- Execute the instruction.

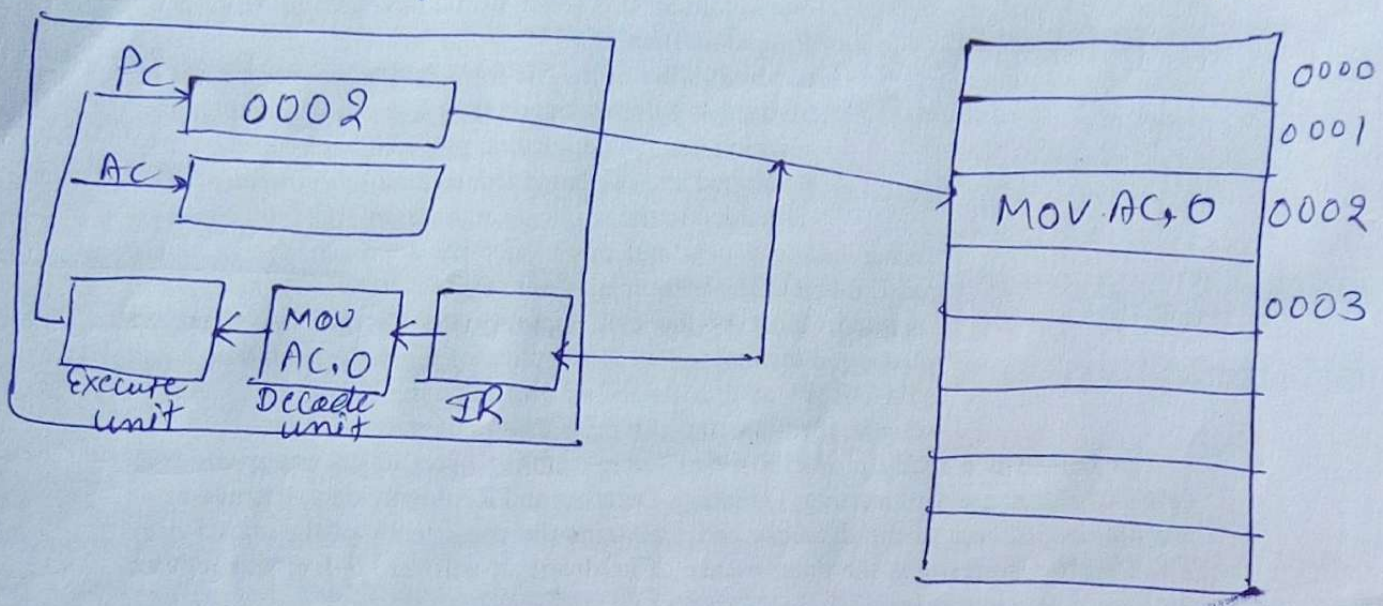
Fetch cycle:-

- In this phase the seauence Counter is initialized to 0.
- The address of first instruction from PC is loaded into address register during the first clock cycle.



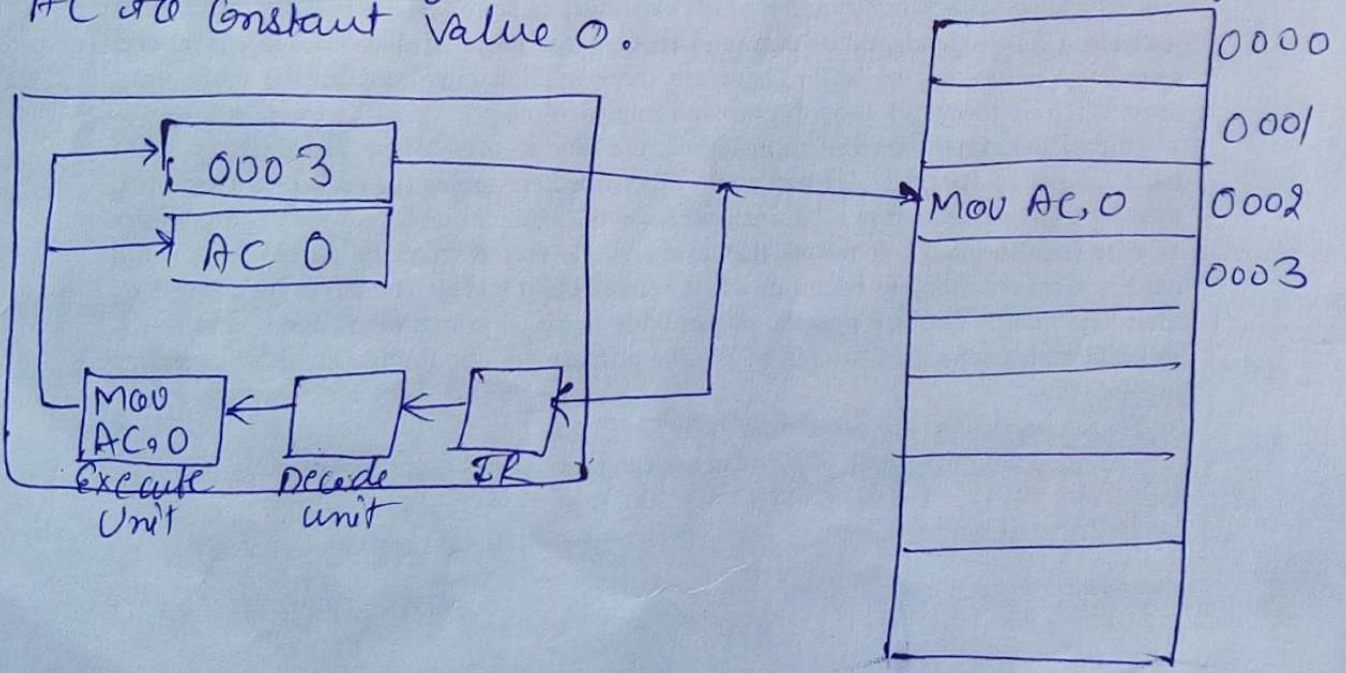
Decode cycle :-

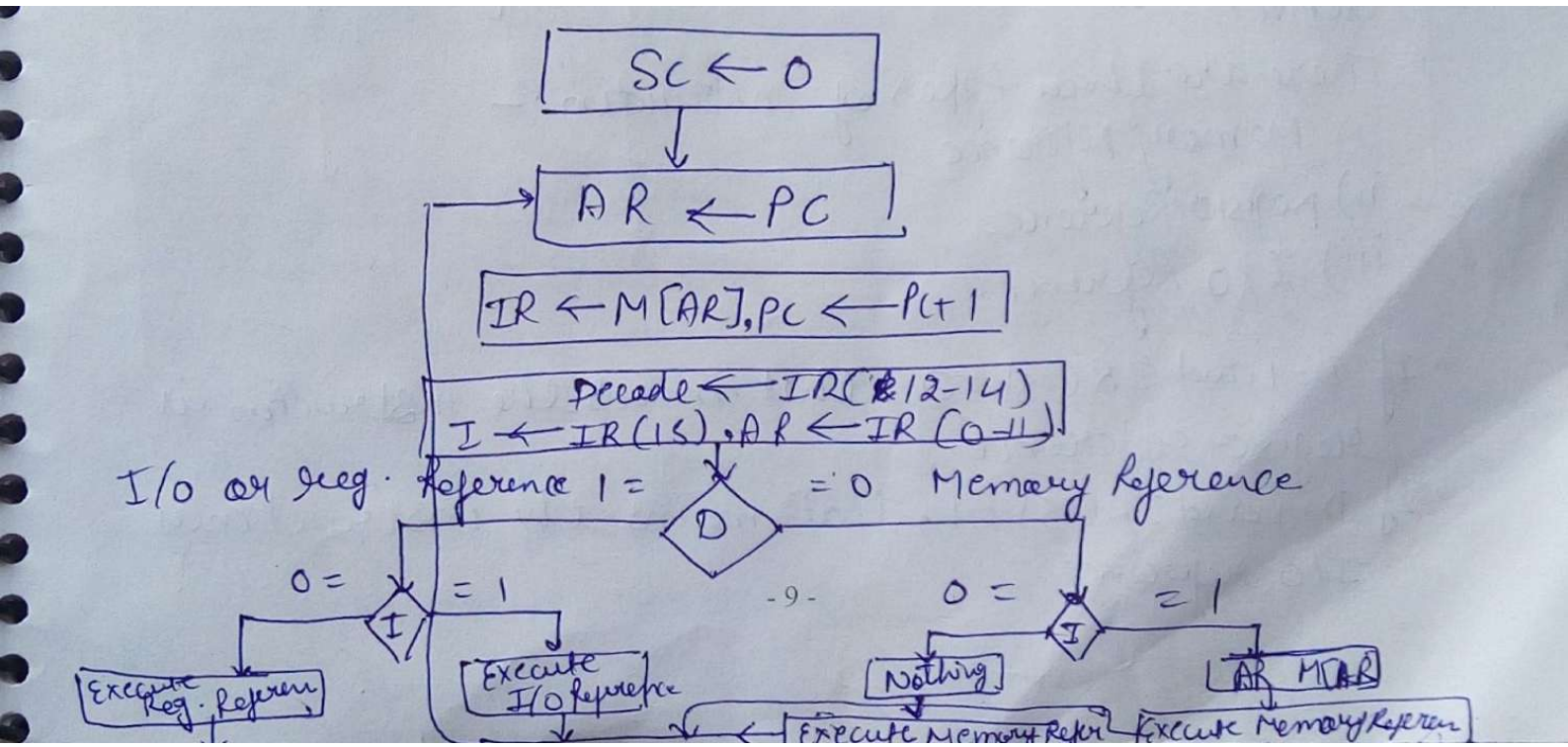
- The instruction is decoded by the instruction decoder of a processor.
- All the bits of the instruction under execution stored in IR are analyzed and decode in third clock cycle.



Execute cycle:-

- In the last phase, the processor execute the instruction.
- This involves setting the contents of the internal register AC to constant value 0.





Fetch operation :-

- The circuit comprises of various registers, memory unit, I/O devices control and timing unit ALU and a common bus.
- $AR \leftarrow PC$
- $IR \leftarrow M[AR]$, $PC \leftarrow PC+1$
- The instruction read from memory is then placed in the instruction register.

Decode Operation :-

- In decode operation processor decode the instruction which is fetched from memory.
- $I \leftarrow IR(15)$. Decode $\leftarrow IR(12-14)$ $AR \leftarrow IR(0-11)$.
- 0-11 bits in the instruction format are store in the address register.
- 12-14 bits are decoded.
- 15 bit is is I means direct or indirect address.

Execution Operation :-

- After decoding the instruction the timing signal that is active.
- There are three types of instruction :-
  i) Memory Reference.
  ii) Register Reference.
  iii) I/O Reference.
- If D=1 and IR(15) =0, that means the instruction is register reference.
- If D=1 and IR(15) =1, that means the instruction is I/O reference.

- If D=0 and IR(15) =0, that means memory reference instruction is direct address instruction.
- If D=0 and IR(15) =1. that means memory reference instruction is indirect address instruction.

2.3 Database Analyzing, design and implementation

The database for the system should include information of company's staff, respectively of its employees. The data is subdivided into the following groups:

Employees' Basic Details	Working History	Time_Information
Employee_ID_Number	Employee_ID_Number	Employee_ID_Number
Personal_ID_Number	Company_Name	Wroked_Hours
First_Name	Employer_Name	Off_Hours
Middle_Name	Company_Employer_Address	Days_off
Last_Name	Company_Employer_Cellular_Phone	Over_Time
Day_of_Birth	Company_Employer_Office_Phone	Extra_Days
Month_of_Birth	Previous_Qualification	w_From_Date_Day
Year_of_Birth	Previous_Experience	w_From_Date_Month
Cellular_Phone	p_Start_Date_Day	w_From_Date_Year
Home_Phone	p_Start_Date_Month	w_To_Date_Day
City	p_Start_Date_Year	w_To_Date_Month
Address	p_End_Date_Day	w_To_Date_Year
Postal_Code	p_End_Date_Month	
Qualification	p_End_Date_Year	
Current_Experience		
Start_Date_Day		
Start_Date_Month		
Start_Date_Year		
End_Date_Day		
End_Date_Month		
End_Date_Year		
Type_of_Employee		
Gender		
Marital_Status		

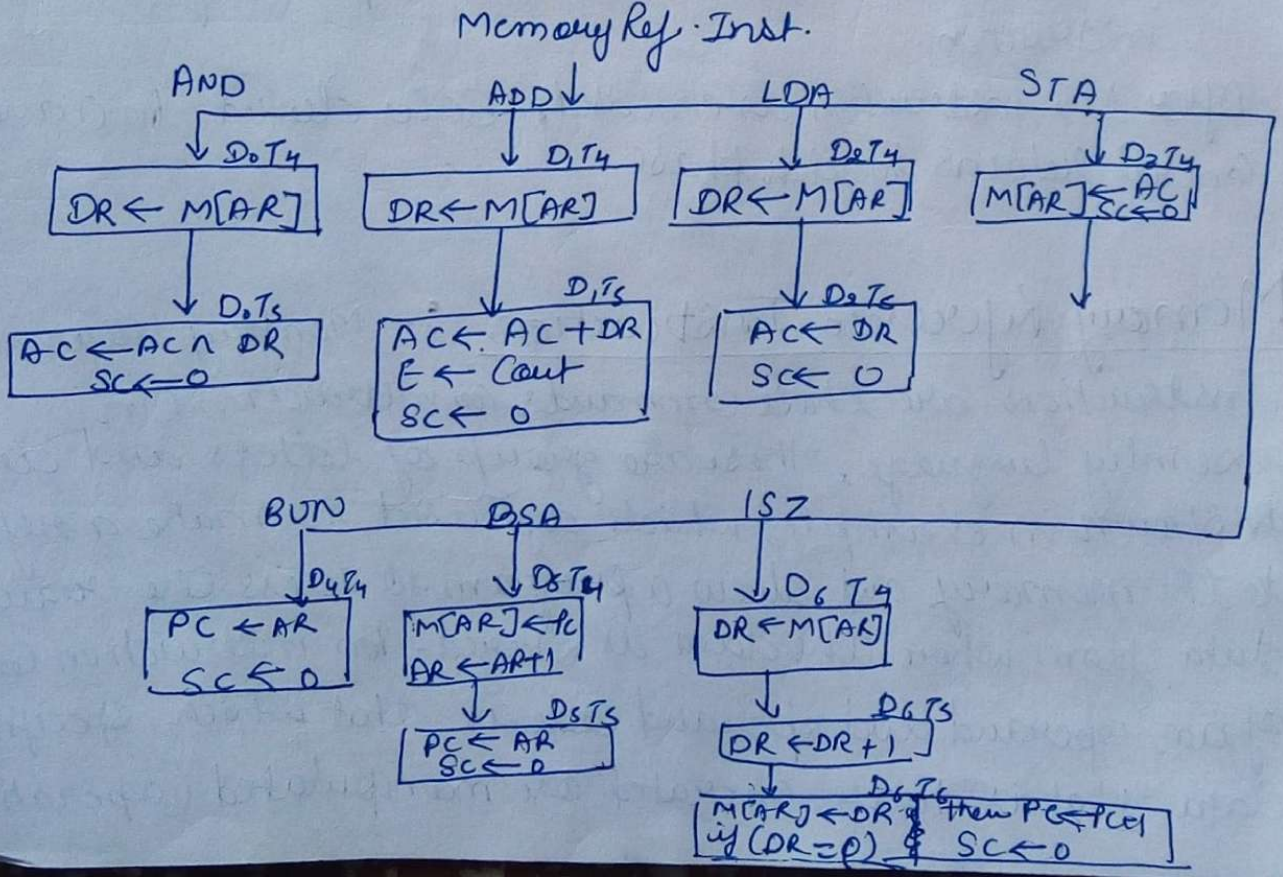
- AR ← M[AR], AR holds the address part of the instruction.
- After the instruction is executed, SC is cleared to 0 and control returns to fetch phase.

Memory Reference Instruction :- Memory reference instructions are those commands or instruction (in assemly language, these are group of letters and are shown in example) which are used to make a reference to the memory and allow a program to access the required data from where the data is stored. An instruction consist of an operand and operand code is that which specifies the data that is to be operated or manipulated (operations

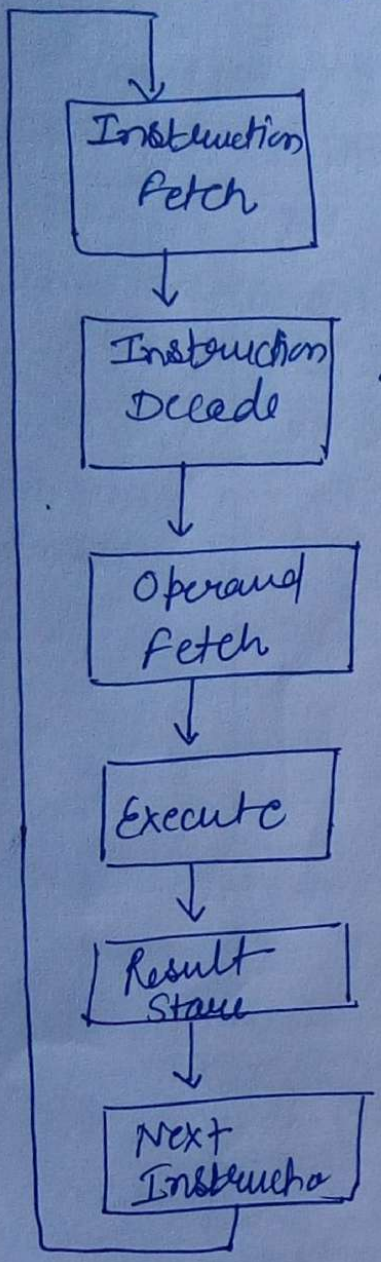
like addition, subtraction etc. are performed on this data specified by operand). An address is also a part of the instruction if it is used in direct addressing mode (in which memory address of the value is declared in the instruction itself) and this address represents the location of the value in the memory. There are 7 memory reference instructions as below.

- LDA
- STA
- AND
- ADD
- BUN
- BSA
- ISZ

For the memory reference instructions, we can compare the below flow chart ~~with~~ that represents the step by step operations performed to execute each of these instructions.



Flow chart for Memory Reference Instruction :-



Obtain instruction from program Storage

Determine required actions and instruction size.

Locate and ~~oper~~ obtain operand data

Compute result value or Status

Deposit results in storage for later use.

Determines successor instruction ; Can generally be Combined W/ Decode.

LDA - This instruction is used to load the Accumulator with a value that is located at a specific memory location. The correct usage of this Command when we write in an assembly language is as follows:

Example of LDA 2500 H

This Command says that load the value that is Stored at the memory location 2500 into the accumulator (a temporary Storage area). H represents that the entered address is a hexadecimal memory address.

So, when a value is stored in the accumulator, further operations are performed as required by the user on that value. The changes made to the value stored in the accumulator are updated in regular time intervals. In excess, this instruction comes under direct addressing mode that is the address or memory location of the operand is written in the instruction itself.

Opcode	Operand	Description
LDA	16 bit Address	Load the Accumulator

STA :→ This instruction is exactly opposite to the above one as this memory reference instruction stores the value that is given to the instruction.

Example of STA 2600H :- This Command Says that Store the value that is Stored in the accumulator to the memory location 2600. When the accumulator is to be assigned with a new value and the present value is to be Sent back to Some memory location, this instruction is used. In excess, this instruction also comes under direct addressing mode.

OpCode	Operand	Description
STA	16 bit Address	Store accumulator direct

AND :- This instruction performs AND operation on the values Stored in accumulator and memory location and the resultant is Stored in the accumulator that is the previous value in the accumulator is updated with this new result. In assembly language, this instruction is not written as AND but is used as written in the example below:

Example of ANA M :- Don't Confuse, AND is written as ANA but not AND.

This Command Says that perform logical AND operation on the value Stored in accumulator and at memory location Specified by M as an operant. Here in the above example, the right most A in ANA Stands for accumulator and M stands for memory location represented with H-L pair. Observe the below table.

Opcode	operand	Description
ANA	R M	Logical AND with register or Memory with Accumulator

ADD :→ This instruction performs arithmetic operation (addition) on the values stored in the accumulator and memory location. The resultant after addition is stored in the accumulator by replacing the older value. This instruction is written as below in order to perform addition on two values.

Example of Add B :-

This command performs adds the value stored in the accumulator with value at memory location specifies by B as an operand. The operand can be a letter that represents the memory location of the value. Here it is taken as B. There are various types of additions which are discussed below.

Opcode	Operand	Description
ADD	B	Add register or memory to accumulator

Other additions and their examples that can be performed using this ~~of~~ instruction are as follows:

ADC M :- This instruction performs addition with carry (carry is a value obtained after addition of the left most digits of both the numbers) by adding the values in accumulator and memory location specified by M as an operand.

ADI 45H :- This instruction says add immediate the value stored in accumulator with the 8 bit data.

ACI 45H :- This instruction says that add immediate along with carry.

DAD B :- If we want to perform a hexadecimal addition, we use this instruction. This instruction performs addition of the 16 bit value in the register pair with the value in H-L pair.

BUN :→ Bun Stands for branch unconditionally and this instruction allows one to select an instruction from a program (which is a group of instructions) and gives him access to modify the programas he wants to. Example of how to write this instruction in programs is as follows:

Example of D_4T_4 : PC ← AR, SC ← 0 // observe the below table to understand the functionality of BUN instruction.

Opcode	Operand	Description
BUN	16 bit Address	Branch Unconditionally

BSA :→ BSA instruction in Computer architecture Stands for branch and Save return address which means it performs two functions. Branching means the instruction is currently being executed may have Sub routines or procedures within it and returning the address in the sense it stores the address of the instruction which should be executed immediately after it and is stored as program Counter.

Example of D_5T_4 : M[AR] ← PC, AR ← AR+1 // increment the value of address register (AR) and returns the value of program counter.

D_5T_5 : PC ← AR, SC ← 0

Opcode	Operand	Description
BSA	16 bit Address	Branch and Save return address

Difference between branch unconditionally (bun) and branch and save return address (bsa)?

Difference between bun and bsa is that in BUN. Unconditional branching and no saving of address but allows program modification whereas in BSA, Branching and saving the returned address but no program modification is allowed.

ISZ :- ISZ stands for increment and skip if zero that is this instruction is used for incrementing the value at the specific address and if the value at that address is found to be zero, then this instruction makes the program counter get incremented by 1.

Example of D_6T_4 : $DR \leftarrow M[AR]$ // assigning the memory location held by address register to data register.

D_6T_5 : $DR \leftarrow DR + 1$ // increments the value of data register.

D_6T_4 : $M[AR] \leftarrow DR$ (if $DR = 0$, then $PC = PC + 1$ and $SC = 0$) // incremented address in data register is now assigned back to address register.

Opcode	Operand	Description
ISZ	16 bit address	Increment and skip if zero

Take a look at the following table to know the instructions and their Symbolic descriptions.

Symbol	operation Decoder	Symbolic Description
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR] \rightarrow, E \leftarrow Cout$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, PC \leftarrow AR+1$
ISZ	D_6	$M[AR] \leftarrow M[AR]+1$ if $M[AR]+1=0$ then $PC \leftarrow PC+1$

Why we need Memory Reference instructions :→

In order to do a specific task, we need the instruction to be executed and for a temporary or permanent purpose, they need to be stored and retrieved from a memory location either registers or accumulators. So, to perform these memory related tasks that is storing, retrieval of values from accumulator and register is handled by these memory reference instructions.

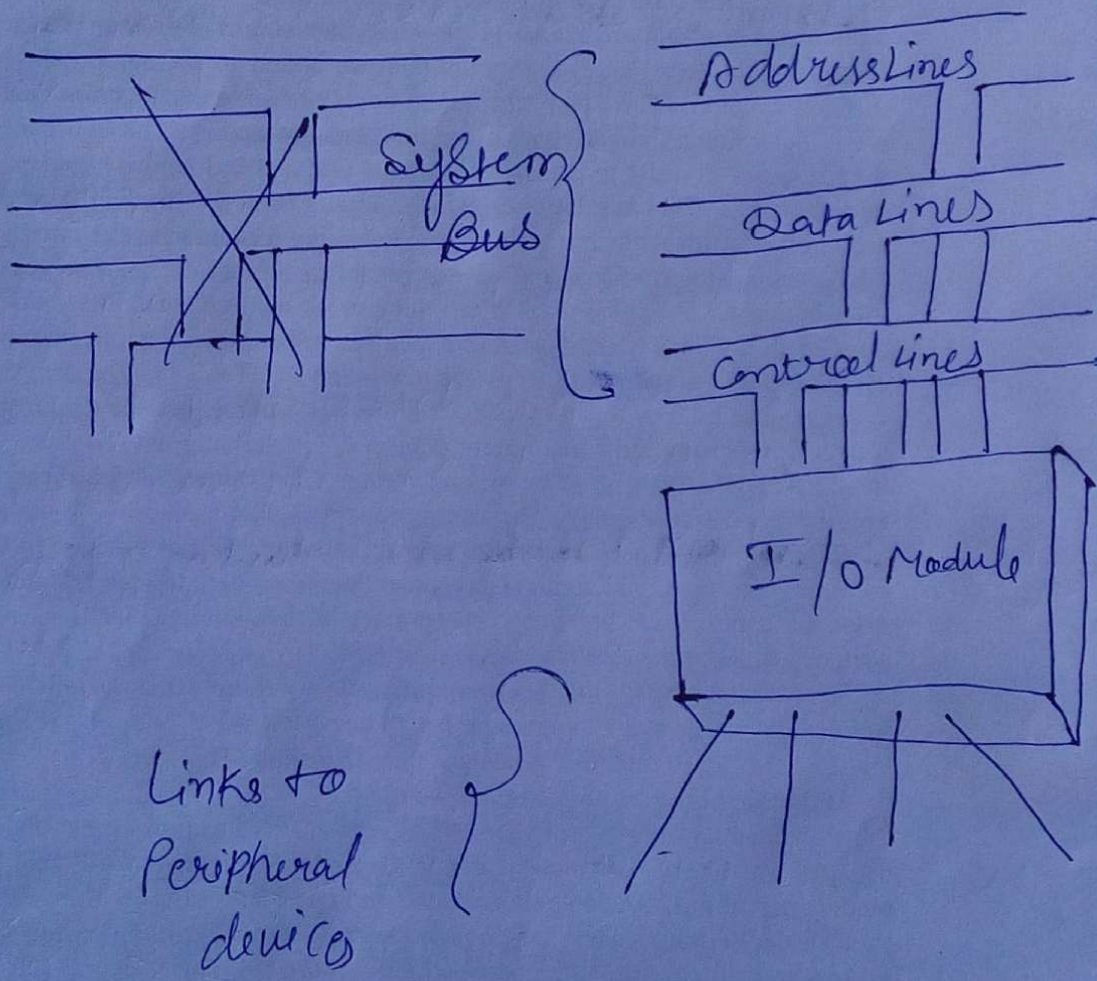
Yogesh Sharma Bhullanwad.

Input/output and Interrupts :→

Input/output Module :→ • External devices are not generally connected directly into the bus structure of the computer.

• I/O module is an interface for the external devices (peripherals) to CPU and Memory.

General Structure of I/O Module (Internal) :-



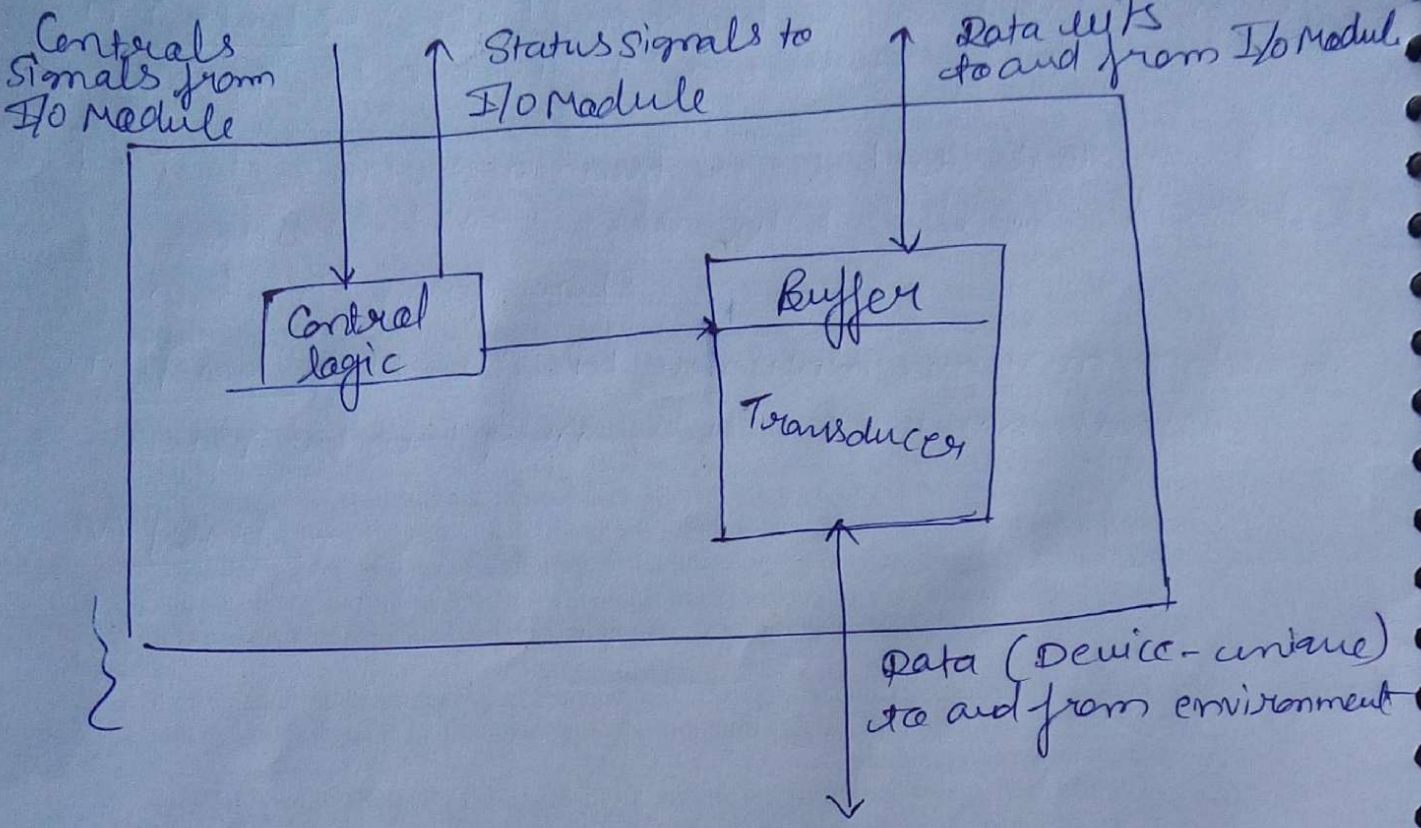
System Bus :- A system bus is a single computer bus that connects the major components of a computer system. It combines the functions of a data bus to carry information, an address bus to determine where it should be sent, and a control bus to determine its operation.

Peripheral devices :- A computer peripheral is a device that is connected to a computer but is not part of the core computer architecture. Peripheral devices can be external or internal. For example : Mouse, Keyboard, Monitor, Printer, Hard-Disk etc.

External Device Interface :-

- It is divided into control module, status signal & data module.
- control signal decide which function will execute by device. for example : Read, write operations.
- Status signals indicates the state (Ready, Not-Ready) of the device.
- Data module controls set of data bits need sending & receiving.
- control logic perform device operation control by getting command from I/O.
- The transducer converts data from electrical to other form of energy during output and from other forms to electrical during input.

External I/O Module Diagram :-



Input/output Techniques :- There are three principle I/O techniques. They are ....

i) Programmed I/O
ii) Interrupt Driven I/O
iii) DMA (Direct Memory Access) Technique.

i) Programmed I/O :-

a) CPU controls I/O directly by doing following three things .....
   i) Sending Status
   ii) I/O Commands
   iii) Transferring data
b) CPU waits for I/O module to complete Operation.
c) CPU time wastes.

I/O Commands :-

i) Control :- Control Command is used to active a peripheral and tell it what to do.

ii) Test :- This command is used to test various status conditions associated with an I/O module and its peripherals.

iii) Read :- It is used to obtain an item of data to peripheral and place it in an internal buffer.

iv) Write :- To take an item of data (byte or word) from the data bus and subsequently transmit that data item to the peripheral

Interrupt Driven I/O :-

- A better protocol is to have the Computer and IO device work independently.
- I/O module interrupts when ready.
- when the current instruction completes, the Computer interrupts the current program, saves the current state and goes to an interrupt Service routine.

How interrupt Driven I/O works :-

- CPU issues read Command.
- I/O module gets data from peripheral whilst CPU does other works.
- I/O module interrupts CPU

- CPU requests for data.
- I/O module transfer data.

DMA (Direct Memory Access) :-

Direct memory Access (DMA) is a feature of Computer Systems that allows certain hardware subsystems to access main system (RAM) memory independently of the central processing unit (CPU).

Without DMA when the CPU is using programmed I/O, it is fully busy for Read or write operation and unavailable to perform other work. With DMA, the CPU first begins the transfer, then it does other operations while the transfer is in progress.

I/O Problems :-

- Slower than CPU & RAM.
- Need I/O Modules.
- For variety of peripherals it may show different amount of data, speeds & formats.
- whole system must have to be efficient to receive input & show output.

Complete Computer description:-

Computer Architecture:-

- It is those attributes of a System that have a direct impact on the logical execution of a program.
- It is Concerned with the structure and behavior of the Computer as Seen by the user.
- It includes:
  - ⇒ The instruction set.
  - ⇒ The number of bits used to represent various data types.
  - ⇒ I/O mechanisms.
  - ⇒ Memory addressing techniques.

Computer Organization:-

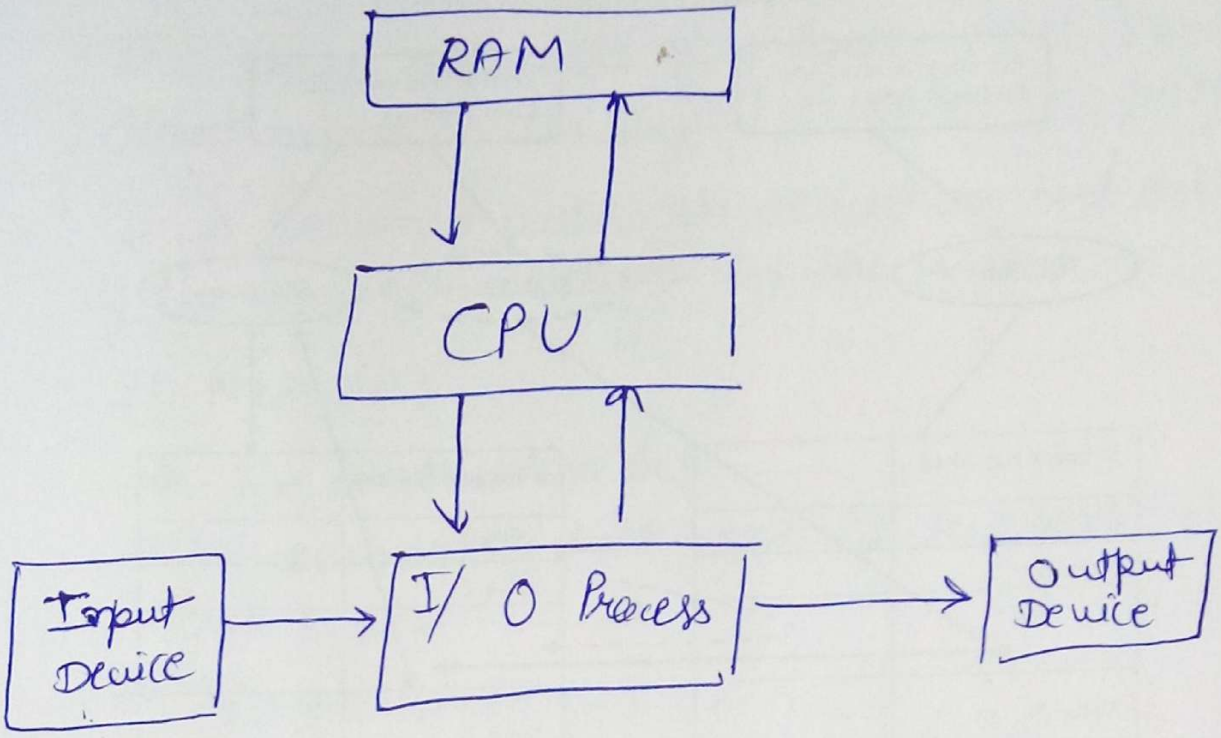
⇒ It refers to the operational units and their interConnections that realize the architectural Specifications.

⇒ It is Concerned with the way the hardware Components operate and the way they are Connected together to form the Computer System.

⇒ Examples are things that are transparent to the programmer:

- Control Signals.
- Interfaces between Computer and peripherals.
- The memory technology being used.

Description of Basic Computer :-



⇒ The Central Processing unit (CPU) for manipulating the data, registers for Storing data and instructions, and Control circuits for fetching and executing instructions.

⇒ The memory of a Computer Contains Storage for instructions and data.

⇒ The input-output processor Contains electronic circuits for Communicating and Controlling the transfer of information between the Computer and user.

~~Memory System Design :→~~

Input/output and Interfacing :→

An input-output interface is the piece of equipment or location at which information can be input and output from a device such as a computer. Example of an input interfaces are : Command line such as a Dos prompt. We manually enter in commands to achieve our result that I want. The Keyboard is also part of this interface.

A GUI (Graphical User Interface) is an icon based interface and is much more user friendly and easier to use. An output interface could be a printer a screen or a speaker.

Input/output or I/O is the communication between an information processing system, such as a computer, and the outside world, possibly a human or another information processing system. Inputs are the signals or data received by the system and outputs are the signals or data sent from it. The term can also be used as part of an action; to "perform I/O" is to perform an input or output operation. I/O devices are used by a human (or other system) to communicate with a computer. For instance, a keyboard or mouse is an input device for a computer, while monitors and printers are output devices. Devices for communication between computers such as modems and network cards. Typically perform both input and o/p operations.

An I/O interface is required whenever the I/O device is driven by the processor. The interface must have necessary logic to interpret the device address generated by the processor. Handshaking should be implemented by the interface using appropriate commands (like Busy, Ready and Wait) and the processor can communicate with an I/O device through the interface. If different data formats are being exchanged, the interface must be able to convert serial data to parallel form and vice versa. There must be provision for generating interrupts and the corresponding type numbers for further processing by the processor if required.

RISC Vs CISC :-

What is CISC ? A Complex instruction Set Computer (CISC) is a Computer where single instructions can execute several low-level operations (such as a load from memory, an arithmatic operation, and a memory store) or are capable of multi-step operations or addressing modes within single instructions, as its name suggest "COMPLEX INSTRUCTION SET".

What is RISC ? A reduced instruction Set Computer (RISC) is a Computer which only use simple instructions that can be divide into multiple instructions which perform low-level operation within single clock cycle, as its name suggest "Reduced Instruction Set".

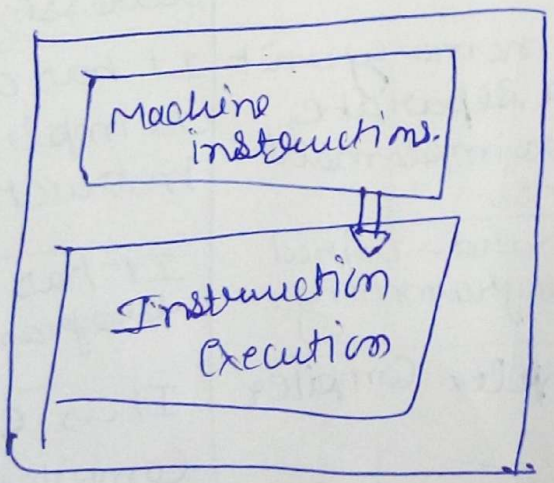
Requirements for I/O interface :-

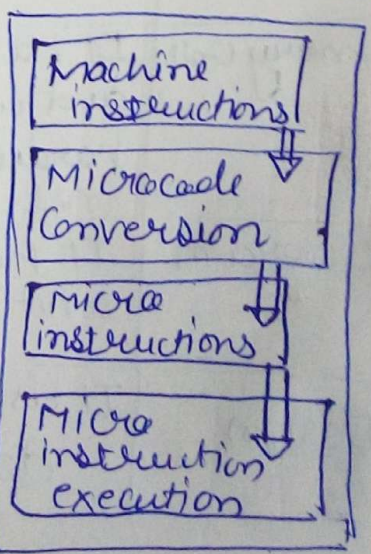
- CPU Communication
- Device "
- Data buffering
- Control and timing
- Error detection

CISC	RISC
Emphasis on hardware	Emphasis on software
Includes multi-clock	Single-clock
Complex instructions	Reduced instruction only
Memory-to-memory: "LOAD" and "STORE" incorporated in instructions	Register to register: "LOAD" and "STORE" are independent instructions
high cycles ~~per~~ per second, Small code sizes	low cycles per second, large code sizes
Transistors used for storing complex instruction	Spends more transistors on memory registers

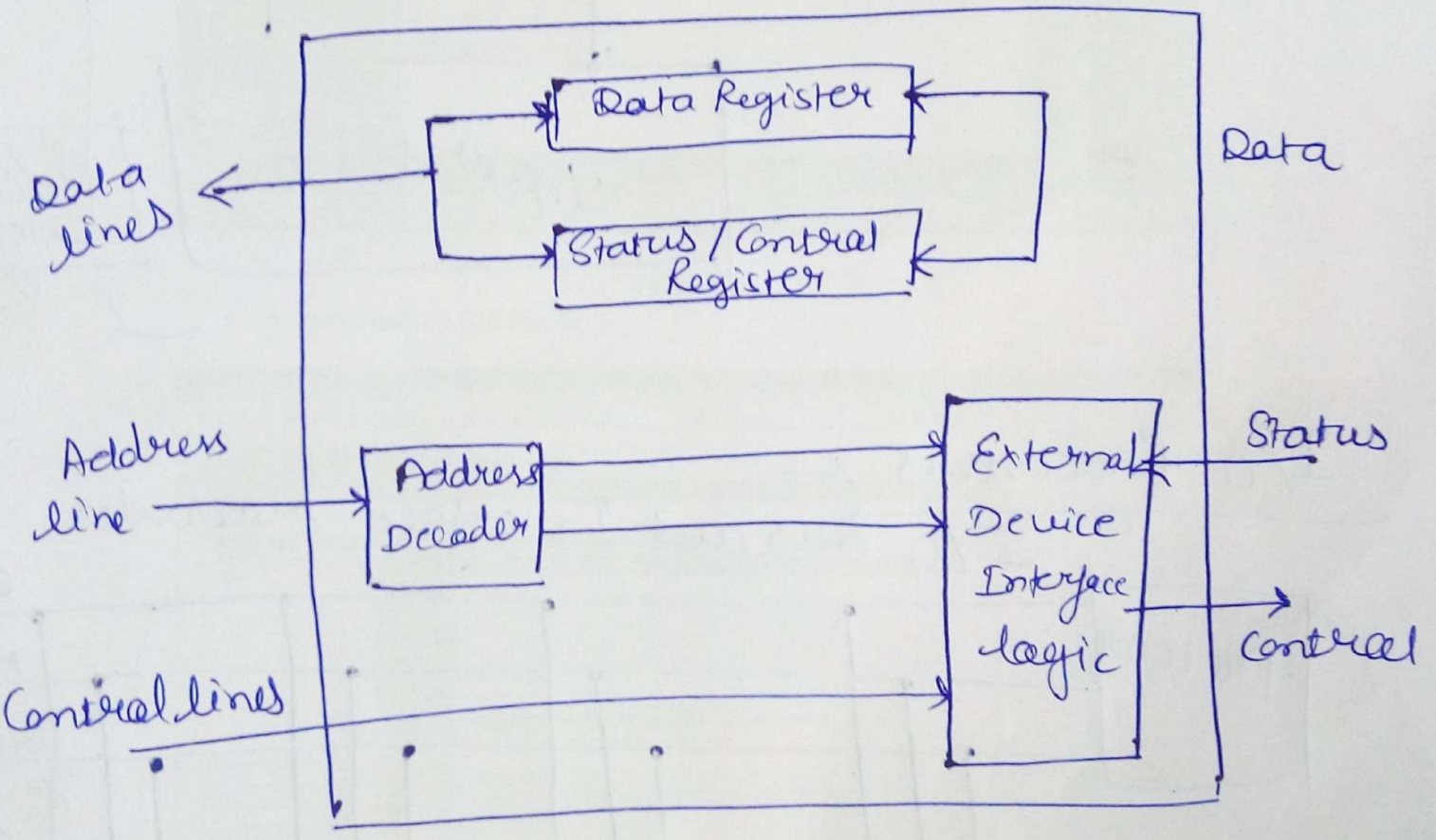
	RISC	CISC
Acronym	It Stands for 'Reduced Instruction Set Computer'	It Stands for 'Complex Instruction Set Computer'
Definition	The RISC processors have a Smaller Set of instructions with few addressing modes.	The CISC processors have a larger Set of instructions with many addressing modes.
Memory Unit	It has no memory unit and uses a Separate hardware to implement instructions.	It has a memory unit to implement Complex instructions.
Program	It has a hard-wired unit of programming.	It has a micro-programming unit.
Design	It is a Complex Compiler design..	It is an easy Compiler design.

Calculations	The Calculations are faster and precise	The calculations are slow and precise.
Decoding	Decoding of instructions is simple	Decoding of instructions is complex
Time	Execution time is very less.	Execution time is very high.
External Memory	It does not require external memory for calculations.	It requires external memory for calculations
Pipelining	Pipelining does function correctly.	Pipelining does not function correctly
Stalling	Stalling is mostly reduced in processors.	The processors often stall.
Code Expansion	Code expansion can be a problem.	Code expansion is not a problem.
Disc Space	The space is saved.	The space is wasted.
Applications	Used in high end applications such as video processing, telecommunications and image processing.	Used in low end applications such as security systems, home automations etc.

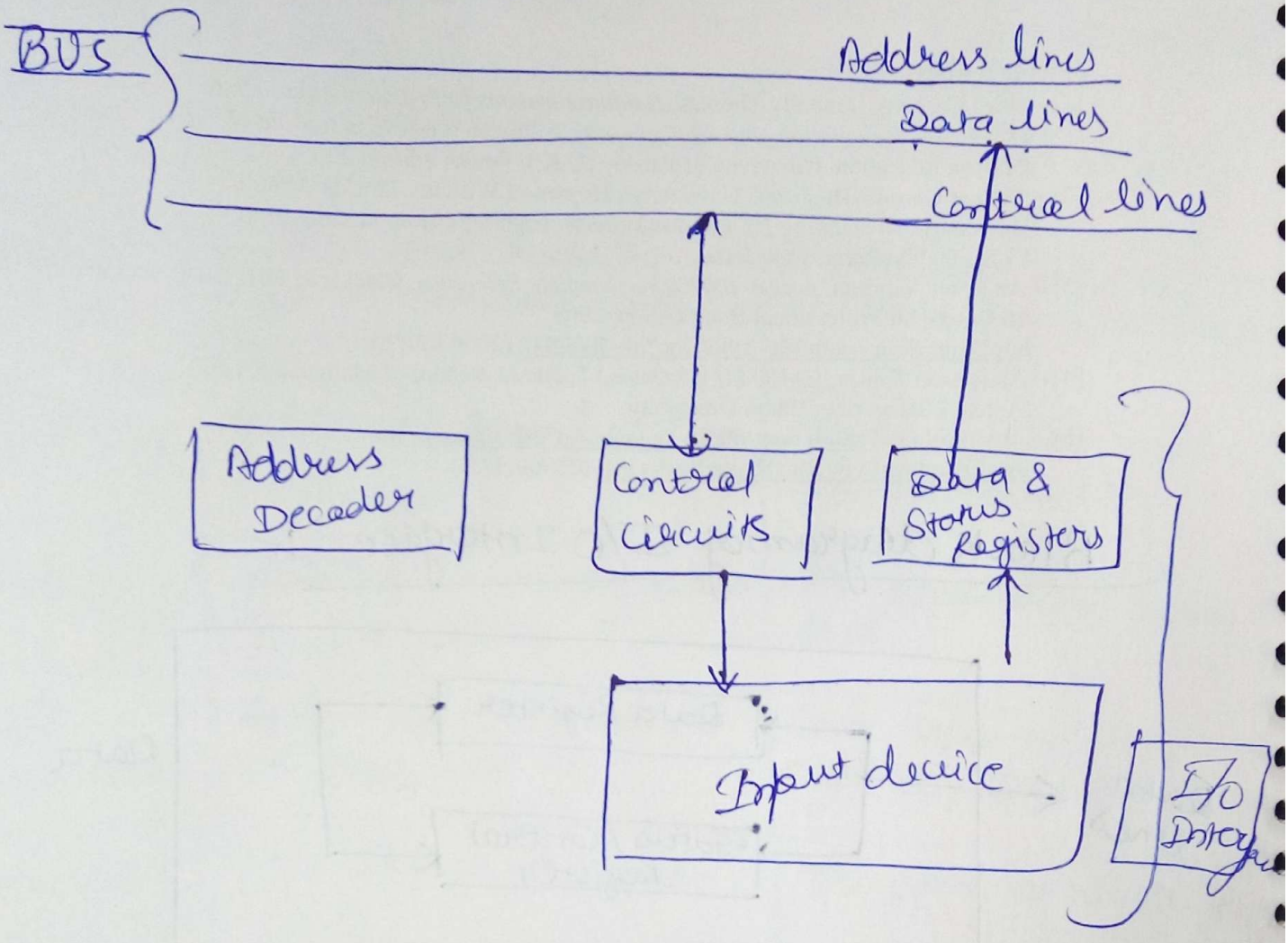




Block diagram of I/O Interface :-



I/O Interface for Input Device



I/O Processors :—

I/O Bus and Interface Modules