# UNIT-1 INTRODUCTION

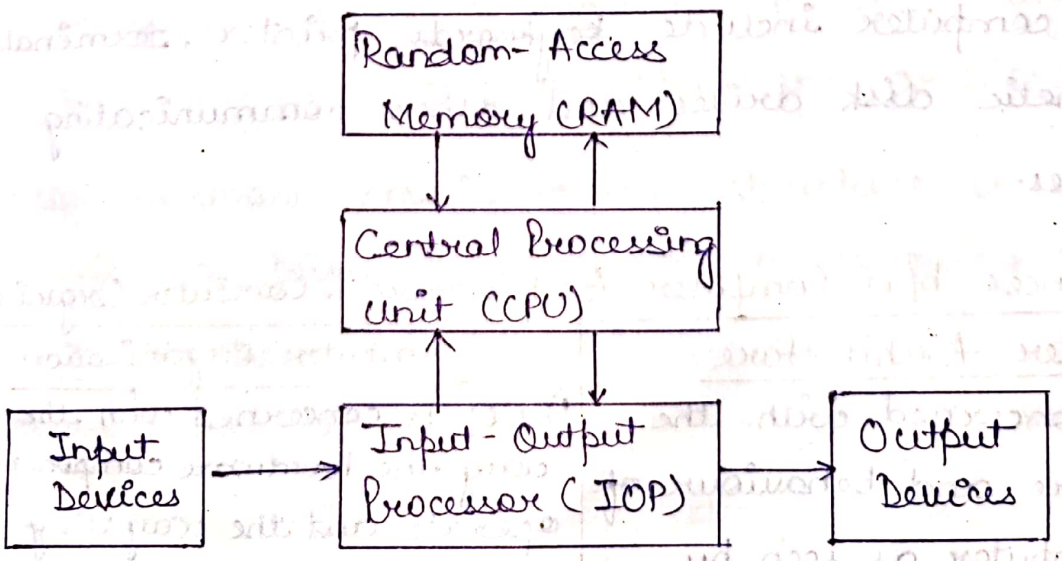
## FUNCTIONAL UNITS OF DIGITAL SYSTEM AND THEIR INTERCONNECTION :-

COMPUTER :- A computer is a combination of hardware and software resources which integrates together and provides various functionalities to the user.

- A computer system is sub-divided into two functional entities :- hardware and software.

Hardware :- The computer hardware are the physical components of a computer like processor, memory devices, monitor, keyboard etc.

Software :- Computer software consists of the instructions and data that the computer manipulates to perform various data processing tasks.




Block diagram of a digital computer

- The hardware of the computer is divided into three major parts -

1) The CPU contains an arithmetic and logic unit for manipulating data, a number of registers for storing data and control circuits for fetching and executing instructions.

2) The memory of a computer contains storage for instructions and data. It is called Random-Access Memory (RAM) because the CPU can access any location in memory at random and retrieve binary information within a fixed interval of time.

3) The IOP contains electronic circuits for communicating and controlling the transfer of information between the computer and the outside world.

- The input and output devices connected to the computer include ~~keyboards~~, printers, terminals, magnetic disk drives and other communicating devices.

## Differences b/w Computer Architecture & Computer Organization

| Computer Architecture | Computer Organization |
|---|---|
| It is concerned with the structure and behaviour of the computer as seen by user. | 1) CO is concerned with the way the hardware components operate and the way they are connected together to form the computer system. |

| | |
|---|---|
| 2) Low level concepts | 2) High level concepts |
| 3) Defines the system in an abstract manner. | 3) Realization of abstract manner. |
| 4) What does the system do/ what functionality is being provided. | 4) How to implement. |
| 5) Types of instructions, techniques for adressing memory. | 5) Physical components like circuits with adder or subtraction. |

- Two basic types of computer architecture :-
  1) von Neumann Architecture
  2) Harvard Architecture

**von-Neumann Architecture :-**

von-Neumann Architecture describes a general framework or structure that a computer's hardwa programming and data should follow.

- The vast majority of computers in use today operate according to the von-Neumann architecture.

Von-Neumann envisioned the structure of a compute system as being composed of the following component

1) **Arithmetic-logic unit (ALU)** - this unit performs th computer's computational and logical functions.
2) **Memory** - the computer's main, or fast memory suc as Random Access Memory (RAM).

3) A control unit that directs other components of the computer to perform certain actions, such as directing the fetching of data or instructions from memory to be processed by the ALU.

4) Man-machine interfaces i.e., input and output devices such as keyboard for input and display monitor for output.

- It is based on the stored-program computer concept, where instruction data and program data are stored in the same memory.

- The sequential execution of programming imposes a sort of 'speed limit' on program execution; since only one instruction at a time can be handled by the computer's processor.

- CPU can be either reading an instruction or reading/writing data from/to the memory. Both cannot occur at the same time since the instruction and data use the same signal pathways and memory.

## BUS :-

- A bus is a communication pathway connecting two or more devices.
- A bus is a shared transmission medium.

- Multiple devices connect to the bus, and a signal transmitted by any one device is available for reception by all other devices attached to the bus.
- If two devices transmit during the same time period, their signals will overlap and become garbled. Thus, only one device at a time can successfully transmit.
- A bus consists of multiple communication pathways or lines. Each line is capable of transmitting signals representing binary 1 and binary 0.
- A bus that connects major computer components (processor, memory, I/O) is called system bus.
- On any bus, the lines can be classified into three functional groups:- data, address and control lines.

Data lines :- It provide a path for moving data between system modules. These lines collectively are called data bus.

- The data bus may consist of from 32 to hundreds of separate lines.
- The number of lines are referred to as the width of the data bus.
- Each line can carry only 1 bit at a time. so the number of lines determines how many bits can be transferred at a time.

- The width of the data bus is a key factor in determining overall system performance.

Address lines :- The address lines are used to designate the source or destination of the data on the data bus.

- If processor wishes to read a word (8, 16 or 32 bits) of data from memory, it puts the address of the desired word on the address lines.
- The width of the address bus determines the maximum possible memory capacity of the system.

Control ~~Bus~~ lines :- The control lines are used to control the access to and the use of the data and address lines.

Because the data and address lines are shared by all components, there must be a means of controlling their use.

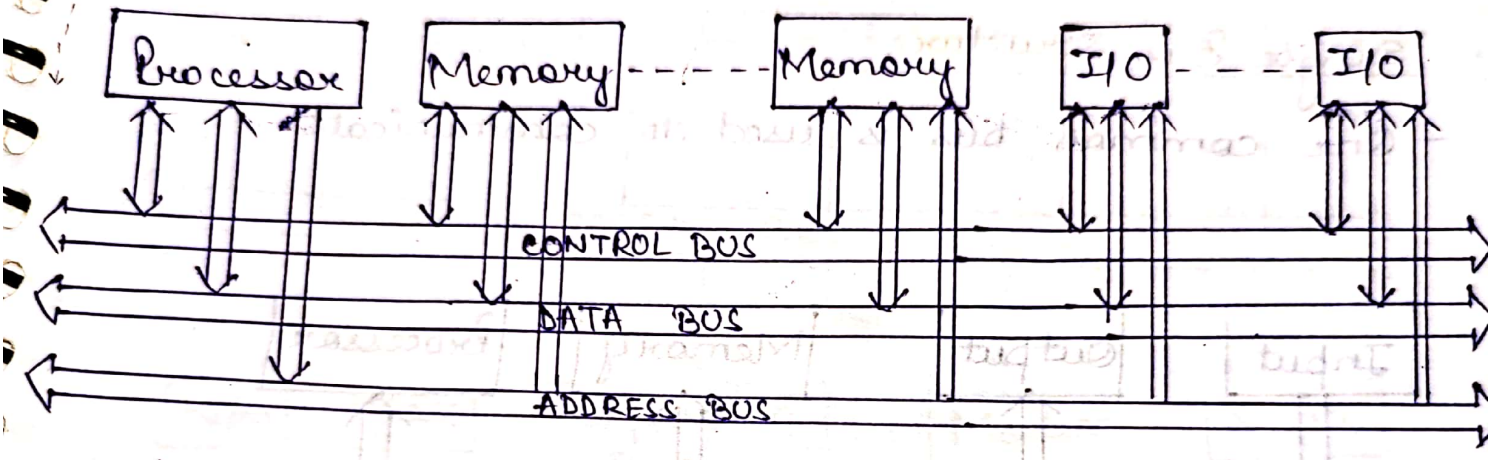
- Control signals transmit both command and timing information between system modules.
- Timing signals indicate the validity of data and address information.

Command signals specify operations to be performed.

Control lines include the following :-

Memory write :- Causes data on the bus to be written into the addressed location.

- Memory read :- Causes data from the addressed location to be placed on the bus.
- I/O write :- Causes data on the bus to be output to the addressed I/O port.
- I/O read :- Causes data from the addressed I/O port to be placed on the bus.
- Transfer ACK :- Indicates that data have been accepted from or placed on the bus.
- Bus request :- Indicates that a module needs to gain control of the bus.




Bus Interconnection Scheme

- Bus grant :- Indicates that a requesting module has been granted control of the bus
- Interrupt request :- Indicates that an interrupt is pend
- Interrupt ACK :- Acknowledges that the pending interrup has been recognized.
- Clock :- Used to synchronize operations.
- Reset :- Initializes all modules.

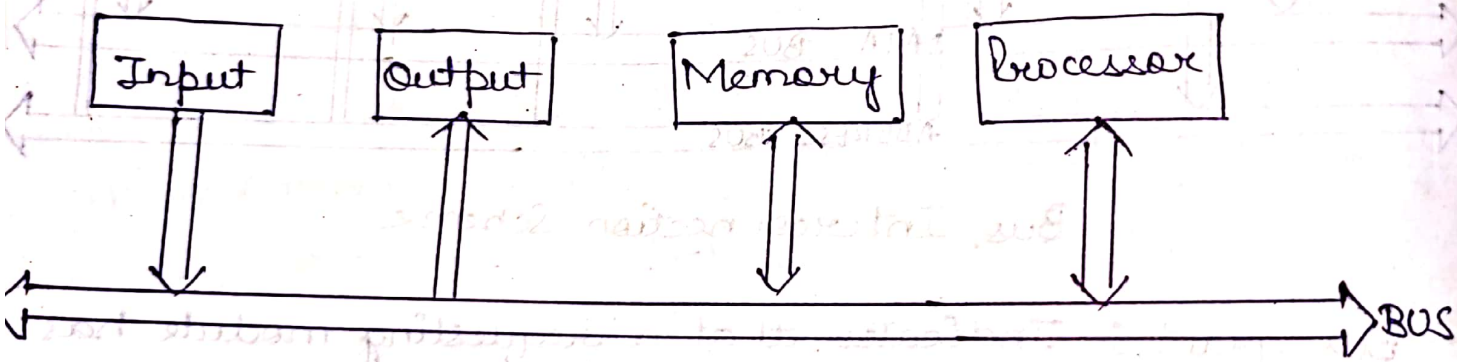
- The operation of the bus is as follows :-

- If one module wishes to send data to another, it must do two things :
  1) Obtain the use of the bus, and
  2) transfer data via the bus

- If one module wishes to request data from another module, it must
  1) Obtain the use of the bus, and
  2) transfer a request to the other module over the appropriate control and address lines.

## Single Bus Structure :-

- One common bus is used to communicate.

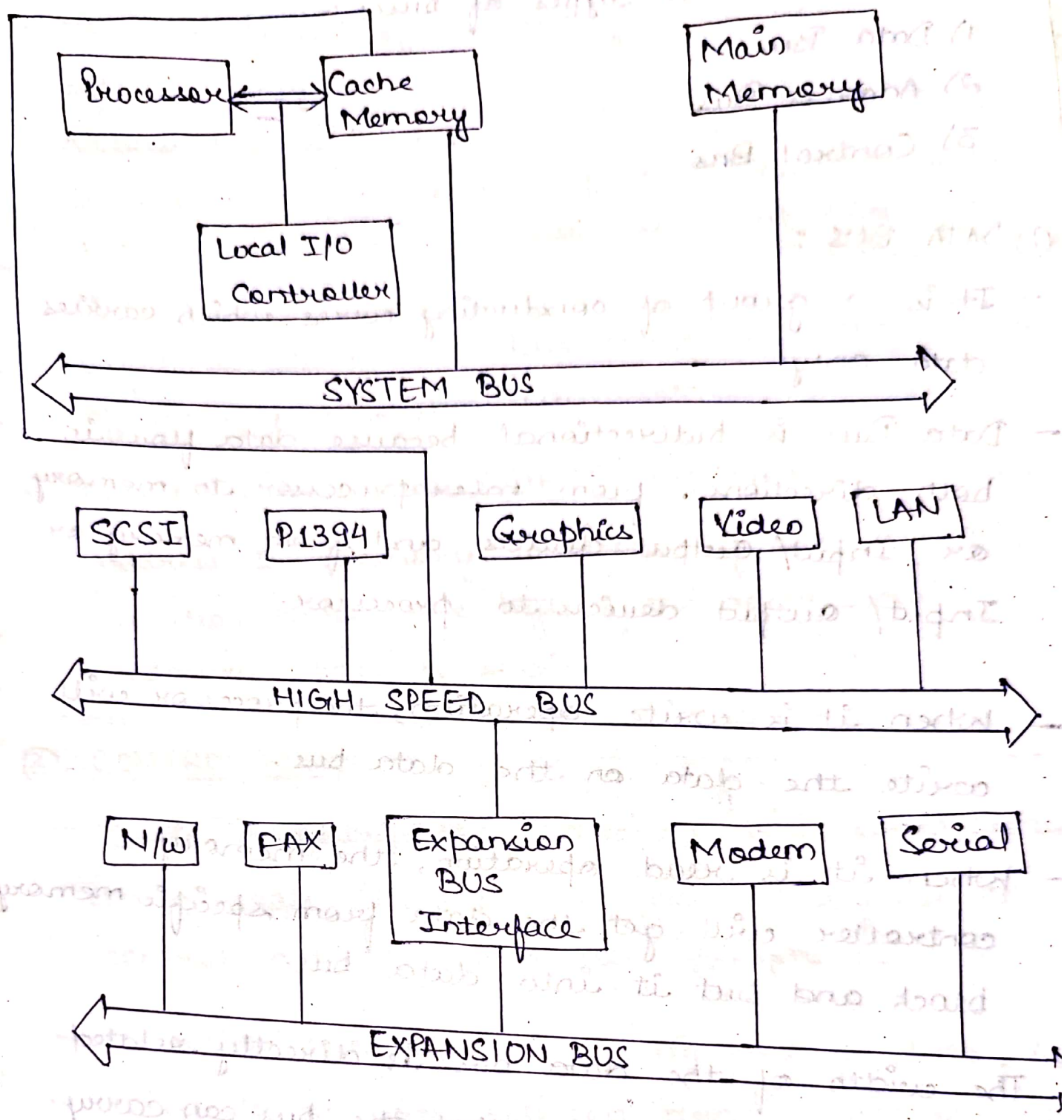



Single Bus Structure

## Drawbacks :-

- Propagation delay
- Limited Capacity
- Bottleneck Problem

## MULTIPLE BUS STRUCTURE :-




Advantage :-

- Fast accessing
- High Performance
- Scalable

SCSI :- Small com
System Interfac

## Types of Buses :-

There are three types of buses :-

1) Data Bus
2) Address Bus
3) Control Bus

### ① DATA BUS :-

- It is a group of conducting wires which carries data only.
- Data Bus is bidirectional because data flow in both directions, from microprocessor to memory or Input/Output devices and from memory or Input/Output devices to processor.

- When it is write operation, the processor will write the data on the data bus.
- When it is read operation, the memory controller will get the data from specific memo block and put it into data bus.
- The width of the data bus is directly related to the largest number that the bus can carry.

Example :- An 8 bit bus can represent 2 to the power 8 unique values, this equates to the number 0 to 255.

② ADDRESS BUS :-

- It is a group of conducting wires that carries address only.
- Address bus is unidirectional because data flow in one direction either from microprocessor to memory or from processor to Input / output devices.
- Length of the address bus determines the amount of memory a system can address.
- Example :- A system with 32-bit address bus can address $2^{32}$ memory locations. If each memory location holds one byte, the addressable memory space is 4 GB.

③ CONTROL BUS :-

- It is a group of conducting wires which is used to generate timing and control signals to control all the associated peripherals.
- Processor uses control bus to process data, i.e. what to do with selected memory location.
- Some control signals are -
  - Memory read
  - Memory write
  - I/O read
  - I/O write
  - Opcode fetch.

## BUS ARBITRATION :-

- Bus Arbitration refers to the process by which the current bus master accesses and then leaves the control of the bus and passes it to another bus requesting processor unit.
- The controller that has access to a bus at an instance is known as a BUS MASTER.
- A conflict may arise if the number of controllers or processors try to access the common bus at the same time, but access can be given to only one of these.

  Only one processor or controller can be Busmaster at the same point in time.

  To resolve these conflicts, the Bus Arbitration procedure is implemented to coordinate the activities of all devices requesting memory transfers.
- The selection of the bus master must take into account the needs of various devices by establishing a priority system for gaining access to the bus.

  The Bus Arbiter decides who would become the current bus master.

There are two approaches to bus arbitration:-

1) Centralized bus arbitration:-
A single bus arbiter performs the required arbitration.

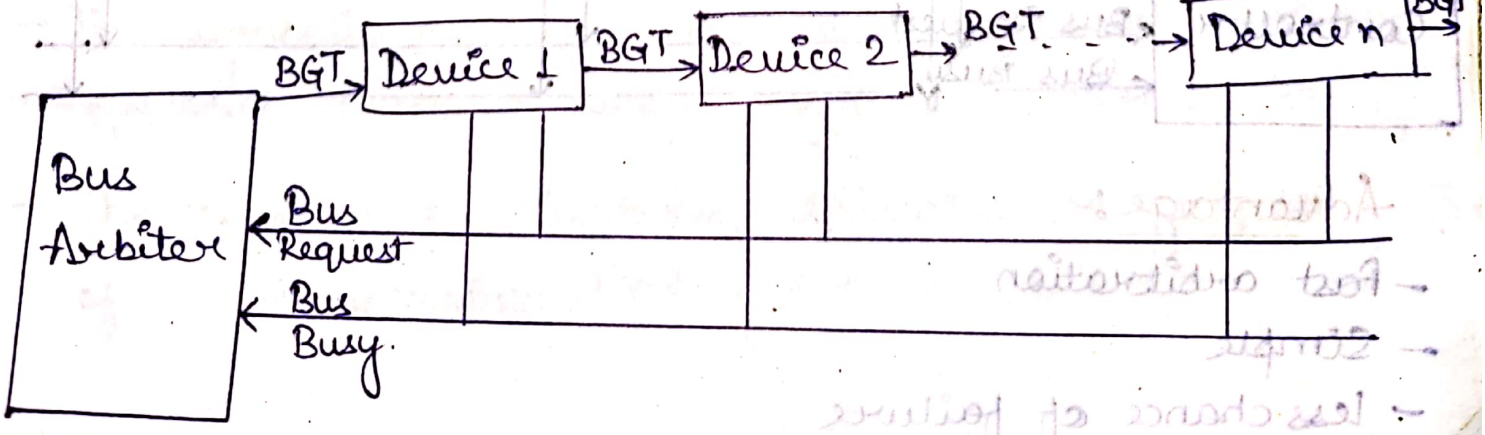
2) Distributed bus arbitration:-
All devices participating in the selection of the next bus master.

## Methods of Centralized Bus Arbitration:-

There are three bus arbitration methods:-

1) Daisy Chaining Method:-

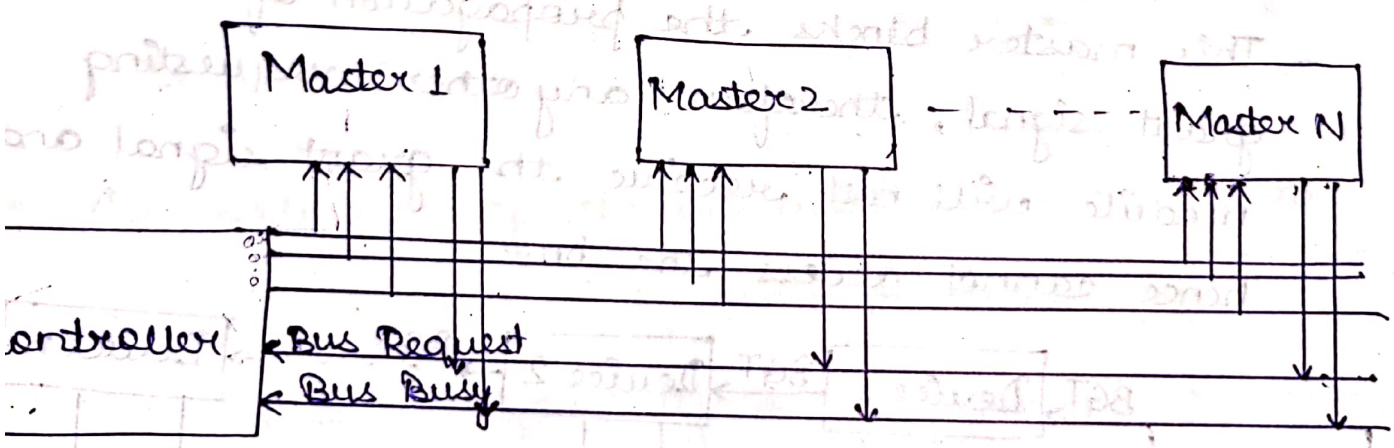
- It is simple and cheaper method where all the bus masters use the same line for making bus requests.
- The bus grant signal serially propagates through each master until it encounters the first one that is requesting access to the bus.
- This master blocks the propagation of the bus grant signal, therefore any other requesting module will not receive the grant signal and hence cannot access the bus.



Advantage :-
- Cheaper and simple
- least no. of lines

Disadvantage :-
- Slow arbitration time
- Priority depends upon physical location
- Chance of failure is more.

ii) Polling or Rotating Priority Method :-
- In this, the controller is used to generate the address for the master (unique priority), the number of address lines required depends on the number of masters connected in the system.
- The controller generates the sequence of master addresses.
- When the requesting master recognizes its address, it activates the busy line and begins to use the bus.



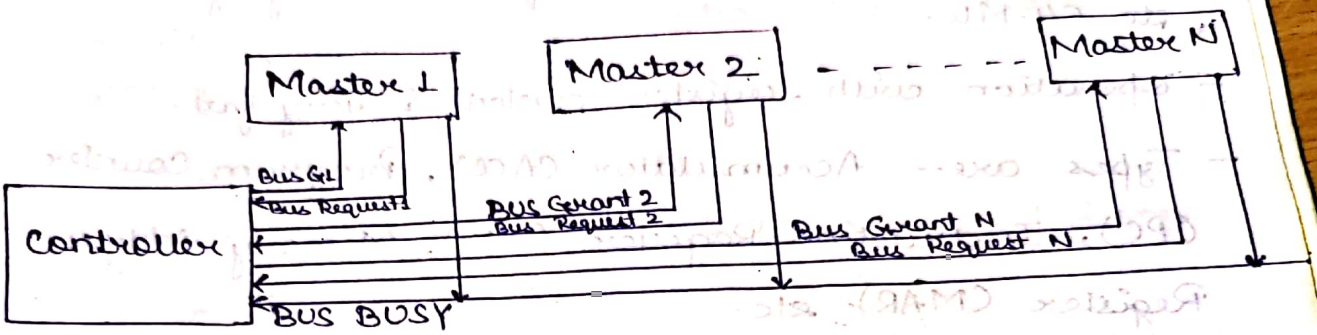

Advantage :-
- Fast arbitration
- Simple
- less chance of failure

Disadvantage :-

- Increasing the size will require more no: of address lines

iii) Fixed Priority or Independent Request Method :-

- In this, each master has a separate pair of bus request and bus grant lines and each pair has priority assigned to it.




Advantage :-

- Very fast arbitration
- Very less chance of failure

Disadvantage :-

- Cost is high
- Chance of starvation

## REGISTER TRANSFER LANGUAGE :-

- The symbolic notation used to describe the micro-operation transfer among registers is called RTL
- The control function that initiates the sequence of micro-operation.

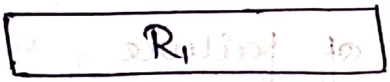
- It is a convenient tool for describing the internal organization of digital computer in concise and precise manner.

## Registers :-

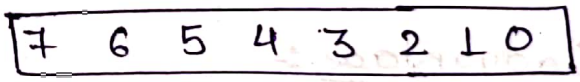
- Holds the operands or instructions that CPU is currently processing.
- Stores the small amount of data ranging 32-bits to 64-bits.
- Operation with register content is very fast.
- Types are- Accumulator (ACC), Program Counter (PC), Instruction Register (IR), Memory Address Register (MAR). etc.

Fig :- Representation of registers in block diagram form

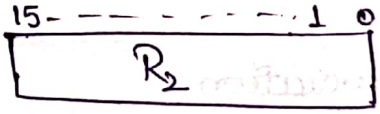
a) Register R



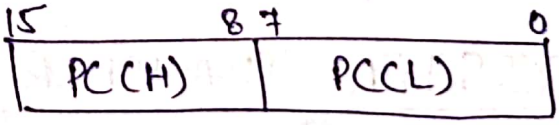

b) Showing individual bits




c) Numbering of bits




d) Divided into two parts




## REGISTER TRASFER :-

- Information transfer from one register to another is of symbolic form by means of a replacement operator.

destination register ↓ $R_2 \leftarrow R_1$ ↑ source register

- Transfer of content of register $R_1$ into register $R_2$.
- Replacement of content of $R_2$ by content of $R_1$.
- The content of source register $R_1$ does not change after the transfer.

** If we want the transfer to occur only under a predetermined control condition then it can be shown by if-then statement.

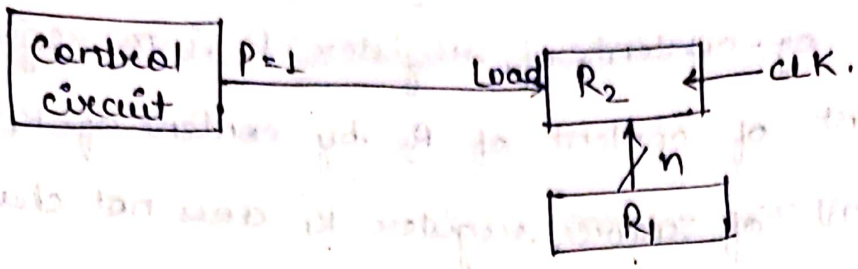
$$\text{if } (P=1) \text{ then } R_2 \leftarrow R_1$$

- P is control signal generated by control section.
- We can separate the control variables from the register transfer operation by specifying a CONTROL FUNCTION.
  - Control function is a Boolean variable that is equal to 0 or 1.

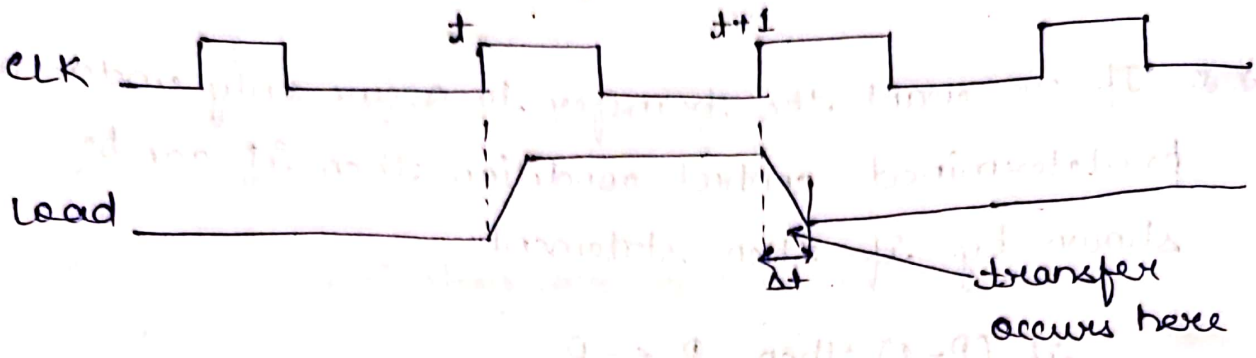
$$P: R_2 \leftarrow R_1$$

- Control condition is terminated by a colon implies transfer operation be executed by the hardware only if $P=1$.
- Every statement written in a register transfer notation implies a hardware construction for implementing the transfer.

Fig :- Transfer from $R_1$ to $R_2$ when $P = 1$.




a) Block Diagram




b) Timing Diagram

- It is assumed that the control variable is synchroni-zed with the same clock as the one applied to the register.

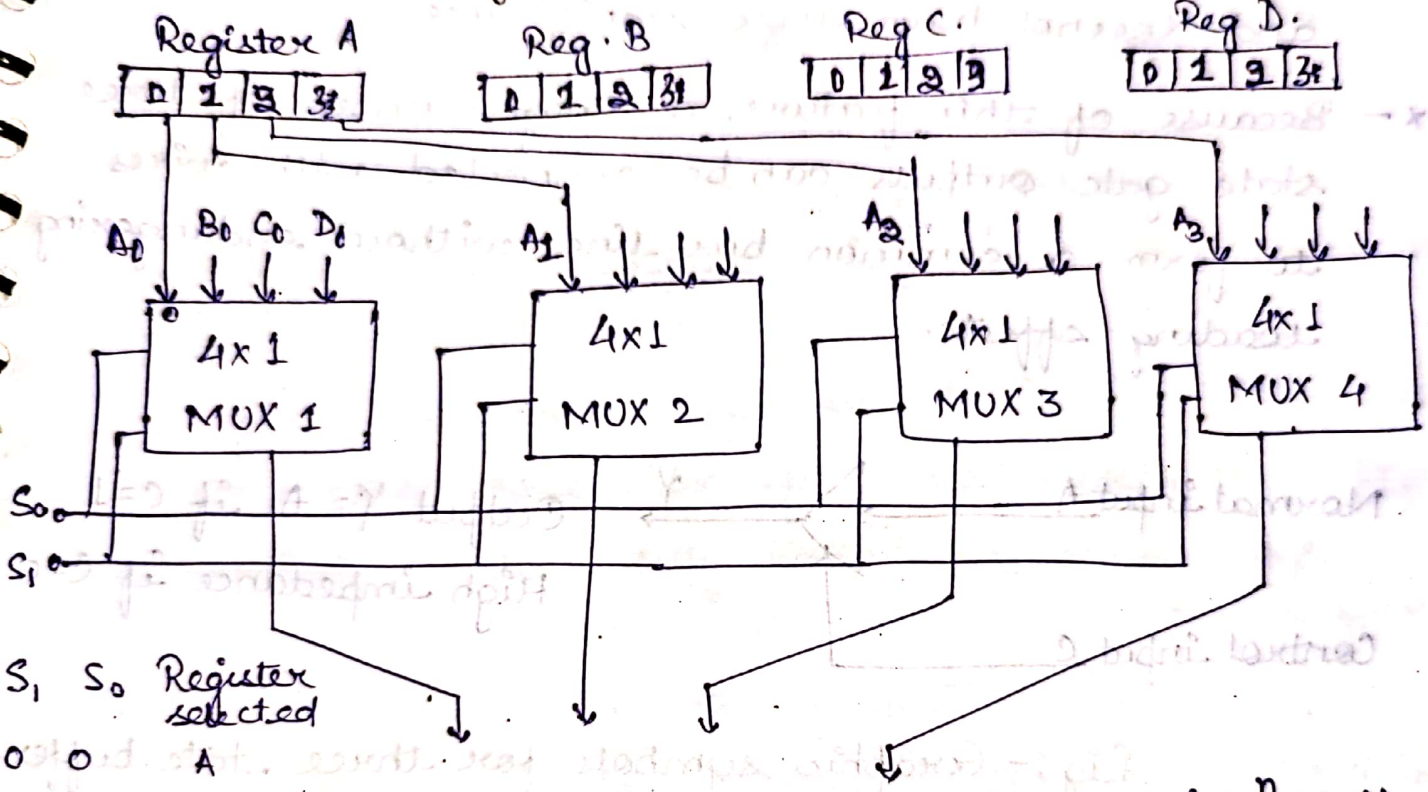
** The basic symbols of the register transfer notation are listed

| Symbol | Description | Examples |
|---|---|---|
| - letters (and numerals) | Denotes a register | MAR, $R_2$ |
| - Parentheses ( ) | Denotes a part of register | $R_2(0\text{-}7)$, $R_2(L)$ |
| - Arrow ← | Denotes transfer of information | $R_2 \leftarrow R_1$ |
| - Comma , | Separates two micro-operations | $R_2 \leftarrow R_1$, $R_1 \leftarrow R_2$ |

## BUS TRANSFER:-

- A more efficient scheme for transferring information between registers in a multiple-register configuration is a common Bus System.
- A common bus consists of a set of common lines, one for each bit of a register.
- Control signals determine which register is selected by the bus during each particular register transfer

Common bus system is with multiplexers :-




| $S_1$ | $S_0$ | Register selected |
|---|---|---|
| 0 | 0 | A |
| 0 | 1 | B |
| 1 | 0 | C |
| 1 | 1 | D |

4-line Bus.

If MUX is $2^n \times 1$ then selection lines = n

1) No. of bits in each register = n

MUX used = $n \times 1$

2) No. of registers = n

no. of MUX used = n

No. of MUX req. = No. of bits in each register.

Size of MUX = No. of req. used.

= 4x1

## Three-State Bus Buffers :-

- A bus system can be constructed with 3-state gates instead of multiplexers.
- A three state gate is a digital circuit that exhibits three states.
- Two of the states are signals equivalent to logic 1 and 0 as in conventional gate.
- The third state is a high-impedance state.
- The high-impedance state behaves like an open circuit which means that the output is disconnected and does not have logic significance.
- Because of this feature, a large number of three state gate outputs can be connected with wires to form a common bus line without endangering loading effects.

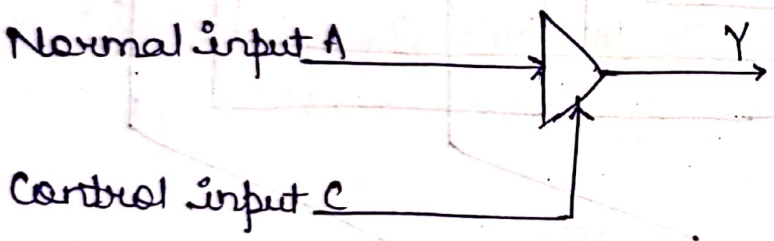



fig :- Graphic symbols for three state buffer.

- The control input determines the output state. When control input is equal to 1, the output is enabled and the gate behaves like any conventional buffer, with output equal to the normal input.
- When the control input is 0, the output is disabled and the gate goes to high-impedance state,

regardless of the value in the normal input.

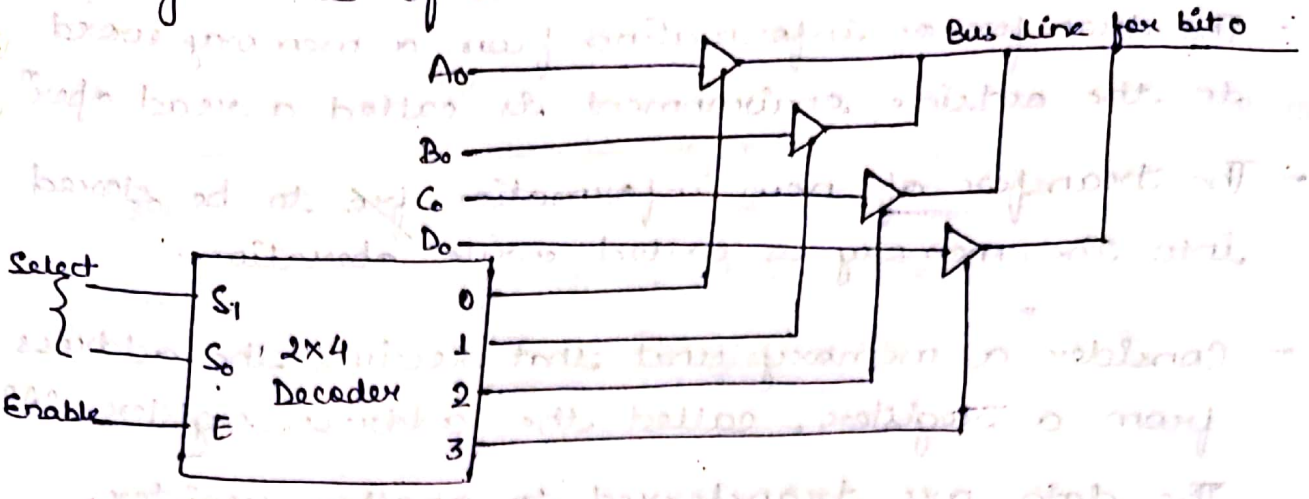



Fig :- Bus-line with three state buffers

- The output of four buffers are connected together to form a sigle bus line.
- The control inputs to the buffers determine which of the four normal inputs will communicate with the bus line.
- No more than one buffer may be in the active state at any given time. The connected buffers must be controlled so that only one three-state buffer has access to the bus line while all other buffers are maintained in a high-impedance state.
- One way to ensure that no more than one control input is active at any given time is to use a decoder.
- When the enable input of the decoder is 0, all of its four outputs are 0 and the bus line is in high impedance state because all four buffers are disabled.
- When the enable input is active, one of the three-state buffers will be active, depending on the binary value in the select inputs of the decoder.

## MEMORY TRANSFER :-

- The transfer of information from a memory word to the outside environment is called a read operation.
- The transfer of new information is to be stored into the memory is called write operation.
- Consider a memory unit that receives the address from a register, called the address register (AR).
- The data are transferred to another register, called the data register (DR).

Read : $DR \leftarrow M[AR]$

This causes a transfer of information into DR from the memory word M selected by the address in AR.

Write : $M[AR] \leftarrow R_1$

This causes a transfer of information from $R_1$ into memory word M selected by address in AR.

## Types of Micro-operations :-

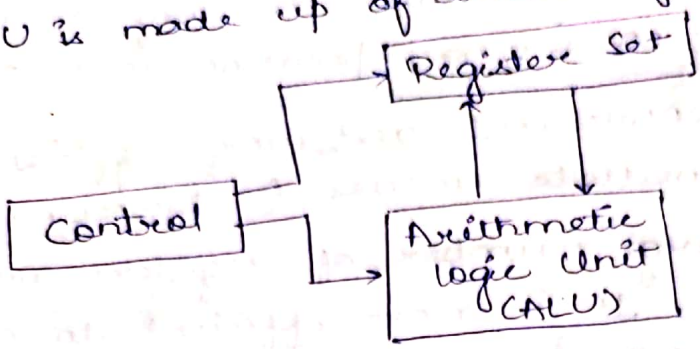
1) Register Transfer Micro-operation
2) Arithmetic Micro-operations
3) Logical Micro-operations
4) Shift Micro-operations

# PROCESSOR ORGANIZATION

## Central Processing Unit :-

- The part of the computer that performs the bulk of data processing operations is called the CPU.

The CPU is made up of three major parts.



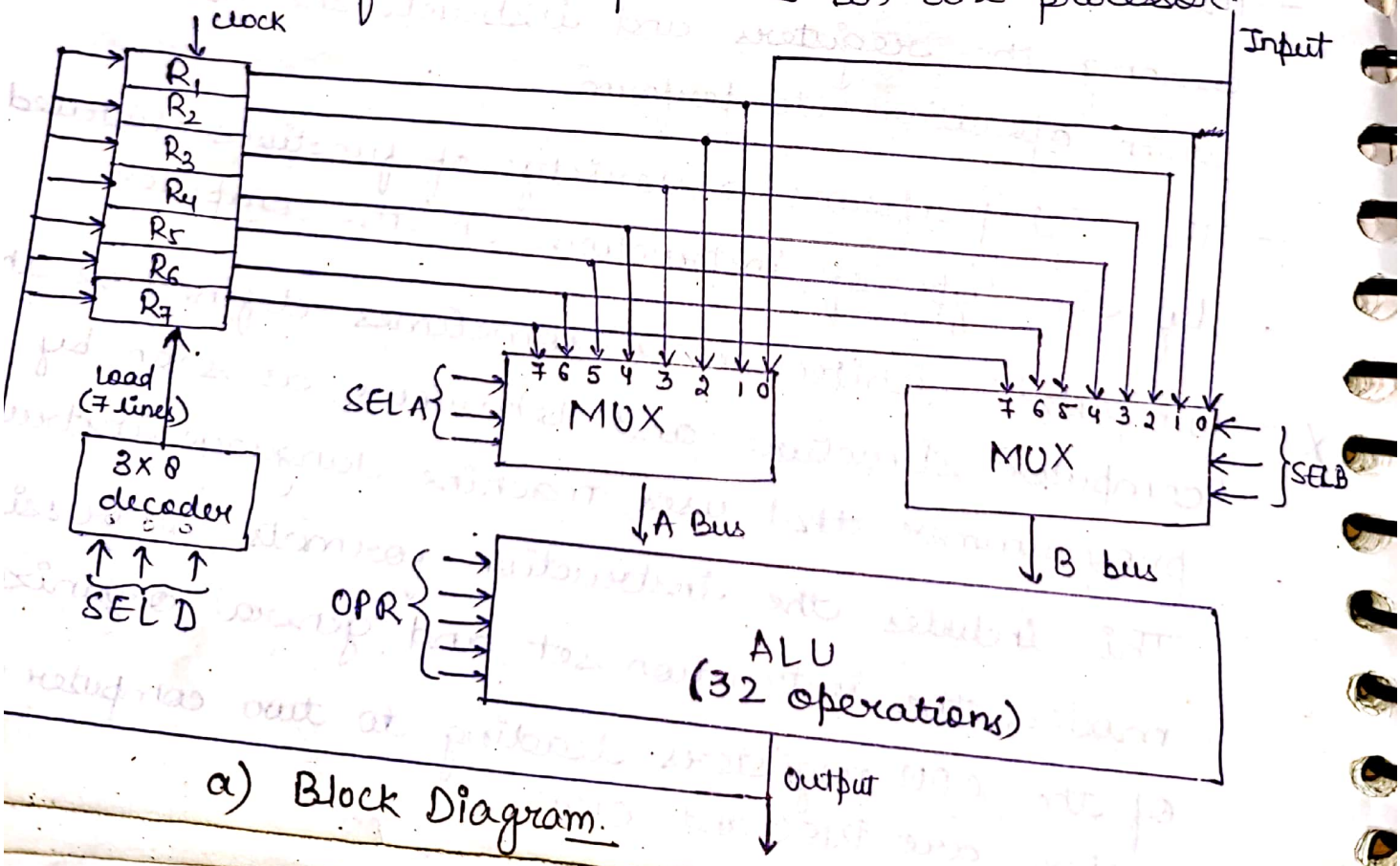

Major components of CPU.

- The register set stores intermediate data during the execution of the instructions.
- The ALU performs the required micro-operations for executing the instructions.
- The control unit supervises the transfer of info. among the registers and instructs the ALU as to which operation to perform.
- The CPU performs a variety of functions dictated by the type of instructions in the computer.
- Computer architecture is sometimes defined as the computer structure and behaviour as seen by programmer that uses machine language instructions. This includes the instruction formats, addressing modes, the instruction set and general organization of the CPU registers leading to two computer architecture are RISC and CISC.

## General Register Organization :-

- Memory locations are needed for storing pointers, counters, return addresses, temporary results, and partial products during multiplication. Having to refer to memory locations for such applications is time consuming because memory access is the most time consuming operation in a computer.
- It is more convenient and more efficient to store these intermediate values in processor registers. When a large number of registers are included in the CPU, it is more efficient to connect them through a common bus system.
- The registers communicate with each other not only for data transfer but also while performing various micro-operations.

Hence it is necessary to provide a common unit that can perform all the arithmetic, logic and shift micro-operations in the processor.




a) Block Diagram.

| 3bit | 3bit | 3bit | 5bit |
|---|---|---|---|
| SELA | SEL B | SEL D | OPR |

b) Control Word

Fig:- Register set with common ALU.

- When SEL A or SEL B is 000, the corresponding multiplexer selects the external input data.
- When SEL D = 000, no destination register is selected but the contents of the output bus are available in the external output.

ex:- $R_1 \leftarrow R_2 - R_3$

| Field :- | SEL A | SEL B | SEL D | OPR |
|---|---|---|---|---|
| Symbol:- | $R_2$ | $R_3$ | $R_1$ | SUB |
| Control Word:- | 010 | 011 | 001 | 00101 |

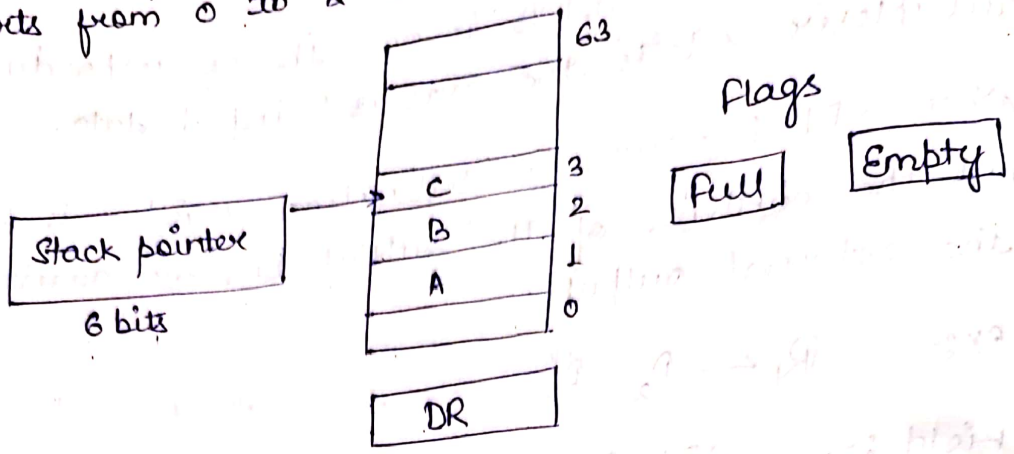
## STACK ORGANIZATION :-

- A stack is a storage device that stores information in such a manner that the item stored last is the first item retrieved. (LIFO)
- The register that holds the address for the stack is called a stack pointer (SP) because its value always points at the top item in the stack.
- The two operations of a stack are the insertion and deletion of items i.e. push and pop.

## Register Stack :-

- A stack can be placed in a portion of a large memory or it can be organized as a collection of a finite number of memory words or registers.

- The size of stack depends on the no. of bits in the stack pointer for an n-bit stack pointer.
- The size of stack is $2^n$ records where number in stack starts from 0 to $2^n - 1$.




Block-diagram of a 64-word stack.

- These are 2 flags associated with stack organised register. The size of each flag is 1-bit

FULL and EMTY

2 flags describe the current situation of stack.

- The bit corresponding to FULL is high (1). When all the memory locations in the stack are occupied.
- EMPTY flag is enabled when there is no element in the stack.

PUSH :-

SP ← SP + 1

M[SP] ← DR

if (SP = 0) then FULL ← 1

EMPTY ← 0

POP :-

DR ← M[SP]

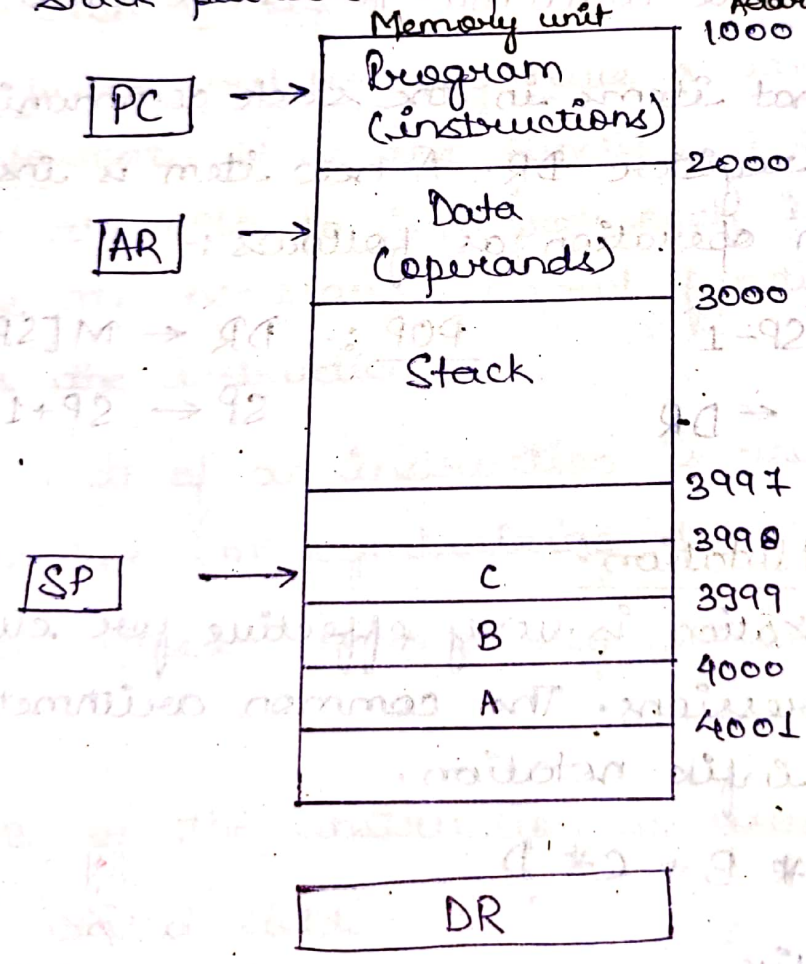
SP ← SP - 1

if (SP = 0) then EMPTY ← 1

FULL ← 0

## Memory Stack :-

- A stack can exist as a stand-alone unit or can be implemented in a random-access memory attached to CPU.
- The implementation of a stack in the CPU is done by assigning a portion of memory to a stack operation and using a processor register as a stack pointer.
- The portion of computer memory partitioned into three segments :- program, data and stack.
- The program counter PC points at the address of the next instruction in the program.
- The address register AR points at an array of data
- The stack pointer SP points at the top of stack.




computer memory with program, data and stack segments.

- The three registers are connected to a common address bus and either one can provide an address for memory.
- PC is used during the fetch phase to read an instruction.
- AR is used during the execution phase to read an operand.
- SP is used to push or pop items into or from the stack.
- Initial value of SP is 4001. and the stack grows with decreasing addresses. Thus the, first item stored in the stack is at address 4000, second is at 3999 and last address that can be used for stack is 3000.
- No provisions are available for stack limit check.
- We assume that items in the stack communicate with a data register DR. A new item is inserted with the push operation as follows :-

PUSH :- $SP \leftarrow SP - 1$
$M[SP] \leftarrow DR$

POP :- $DR \leftarrow M[SP]$
$SP \leftarrow SP + 1$

## Reverse Polish Notation :-

A stack organization is very effective for evaluating arithmetic expressions. The common arithmetic expressions are written in infix notation.

A * B + C * D

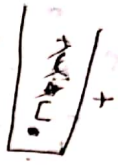
A + B infix
+ AB Prefix or Polish
AB + Postfix or Reverse Polish Notation

AB * CD * +

- The reverse polish notation is in a form suitable for stack manipulation.
- A * B + C * D in reverse polish AB * CD * +

(A+B) * [C * (D+E)+F]  AB+ CDE+ * F+ *

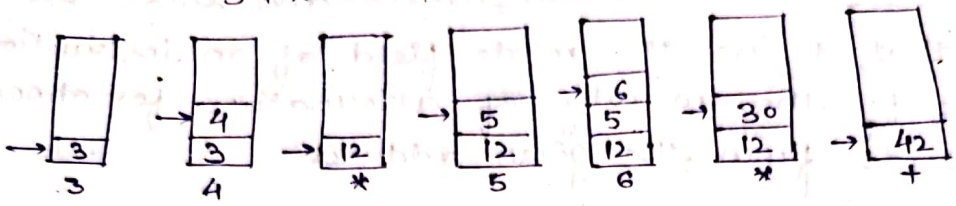
AB+ DE+ C* F+ *



## Evaluation of Arithmetic Expressions :-

(3 * 4) + (5 * 6)

3 4 * 5 6 * +




## Instruction formats :-

- A computer will usually have a variety of instruction code formats. It is the function of the control unit within the CPU to interpret each instruction code and provide the necessary control functions needed to process the instruction.
- The format of an instruction is usually depicted in a rectangular box symbolizing the bits of the instruction as they appear in memory words or in a control register.
- The bits of the instruction is usually divided into groups called fields.

The most common fields found in instruction format are-

1) An operation code field that specifies the operation to be performed.
2) An address field that designates a memory address or a processor register.
3) A mode field that specifies the way the operand or the effective address is determined.

* The operation code field of an instruction is a group of bits that define various processor operations such as add, subtract, complement and shift. The

x- The bits that define the mode field of an instruction code specify the variety of alternatives for choosing the operands from the given address.

x- In this section we are concerned with the address field of an instruction format and consider the effect of including multiple address fields in an instruction.

- Operations specified by computer instructions are executed on some data stored in memory or processor registers.
- Operands residing in memory are specified by their memory address.
- Operands residing in processor registers are specified with register address.

A register address is a binary number of k-bits that defines one of $2^k$ registers in the CPU.

Computer may have instructions of several different length containing varying number of addresses.

- The no. of address fields in the instruction format of a computer depends on the internal organization of its registers.
- Most computers fall into one of the three types of CPU organization :-

1) Single accumulator organization
2) General register organization
3) Stack organization

① All operations are performed with an implied accumulator register. The instruction format in this type of computer uses one address field.

ADD X - Where X is the address of the operand.

AC ← AC + M[X] - AC is accumulator reg.
M[X] - memory word located at add. X.

② Three address fields :-

ADD $R_1, R_2, R_3$ ⇐ $R_1 \leftarrow R_2 + R_3$

③ Two address fields :-

ADD $R_1, R_2$ ⇐ $R_1 \leftarrow R_1 + R_2$

→ Computer with multiple processor register use the move instruction with mnemonic MOV to symbolize a transfer instruction.

MOV $R_1, R_2$ ⇐ $R_1 \leftarrow R_2$ (or $R_2 \leftarrow R_1$, depending on the particular computer).

Thus transfer type instruction need two address fields to specify the source and the destination.

1) - General register type computer employ two or three address fields in their instruction format. Each address field may specify a processor register or a memory word.

ADD $R_1, X$ ⇐ $R_1 \leftarrow R_1 + M[X]$

- Computer with stack organization would have PUSH and POP instruction which require an address field.

PUSH X - will push the word at address X to the top of the stack. The stack pointer is updated automatically.

- Operation - type instructions donot need an address field in stack-organized computers. This is because the operation is performed on the two items that are on the top of the stack. The instruction

ADD - in a stack computer consists of an operation code only with no address field.

eg:- $X = (A+B) * (C+D)$ we will assume that the operands are in memory address A, B, C and D and the result must be stored in memory at address X.

Three - address instructions :-

| | | |
|---|---|---|
| ADD | $R_1, A, B$ | $R_1 \leftarrow M[A] + M[B]$ |
| ADD | $R_2, C, D$ | $R_2 \leftarrow M[C] + M[D]$ |
| MUL | $X, R_1, R_2$ | $M[X] \leftarrow R_1 * R_2$ |

it is assumed that computer has two processor registers, $R_1$ and $R_2$. M[A] denotes the operand at memory address symbolized by A.

## Two-address Instructions :-

| | |
|---|---|
| MOV $R_1$, A | $R_1 \leftarrow M[A]$ |
| ADD $R_1$, B | $R_1 \leftarrow R_1 + M[B]$ |
| MOV $R_2$, C | $R_2 \leftarrow M[C]$ |
| ADD $R_2$, D | $R_2 \leftarrow R_2 + M[D]$ |
| MUL $R_1$, $R_2$ | $R_1 \leftarrow R_1 * R_2$ |
| MOV X, $R_1$ | $M[X] \leftarrow R_1$ |

## One-address Instructions :-

| | |
|---|---|
| LOAD A | $AC \leftarrow M[A]$ |
| ADD B | $AC \leftarrow AC + M[B]$ |
| STORE T | $M[T] \leftarrow AC$ |
| LOAD C | $AC \leftarrow M[C]$ |
| ADD D | $AC \leftarrow AC + M[D]$ |
| MUL T | $AC \leftarrow AC * M[T]$ |
| STORE X | $M[X] \leftarrow AC$ |

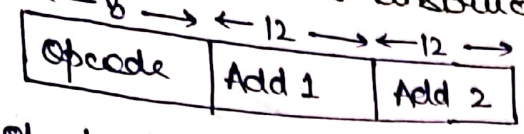
T - address of temp. memory required for storing intermediate result

## Zero-Address Instructions :-

Absence of an address field in the computational instructions

| | |
|---|---|
| PUSH A | $TOS \leftarrow A$ |
| PUSH B | $TOS \leftarrow B$ |
| ADD | $TOS \leftarrow (A + B)$ |
| PUSH C | $TOS \leftarrow C$ |
| PUSH D | $TOS \leftarrow D$ |
| ADD | $TOS \leftarrow (C + D)$ |
| MUL | $TOS \leftarrow (C + D) * (A + B)$ |
| POP X | $M[X] \leftarrow TOS$ |

Ques. A computer has 32-bit instructions and 12-bit addresses. If there are 250 two-address instructions. how many one-address instructions can be formulated?



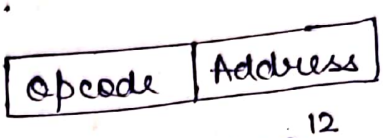

Two address instruction

No. of possible opcode of 8 bits = $2^8 = 256$

Among 256 opcodes, no. of two-address instructions = 250

∴ Remaining opcodes = 256 - 250 = 6 combinations are used for one-address instructions.

One-address Instructions



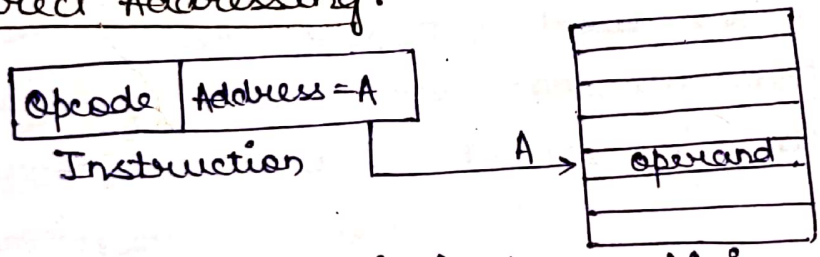

∴ Max. no. of one-address instruction = $6 \times 2^{12}$.

## ADDRESSING MODE :-

- Specify different ways of determining (calculating) effective address of an operand.

### Types of addressing Modes :-

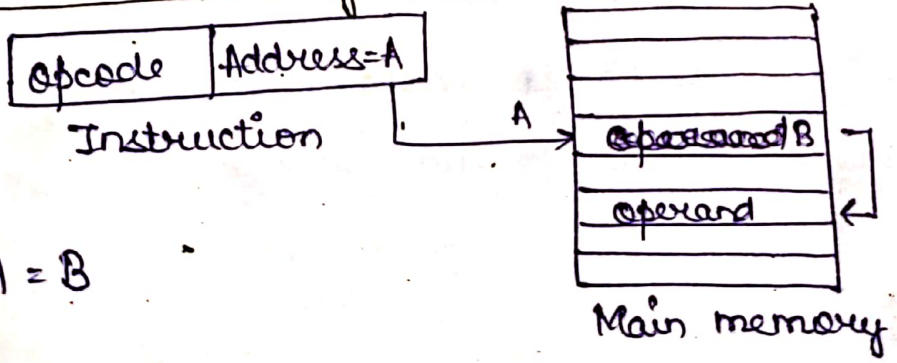
① Direct Addressing :-




Effective Address (EA) = A

- EA is equal to the address field part of the instructions. The operand resides in memory and its address is given directly by the address field of the instruction.

② Indirect Addressing Mode :-




EA = B

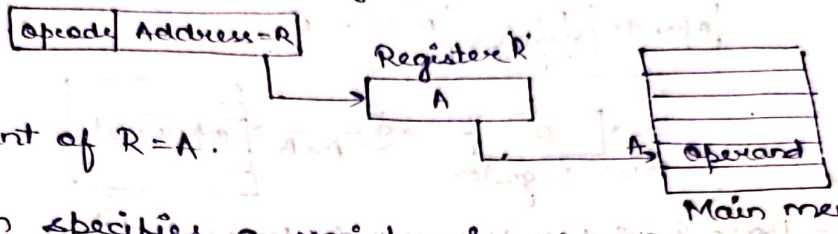
- Address field of the instruction gives the address where the effective address is stored in memory.

③ Register Addressing :-


EA = R.

- The operands are in registers that reside within the CPU. A particular register is selected from a register field in the instruction.

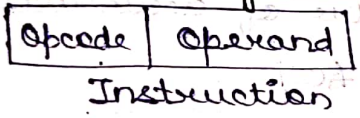
④ Register Indirect Addressing :-




EA = content of R = A.

- Instruction specifies a register in the CPU whose contents give the address of the operand in the memory. In other words, the selected register contains the address of the operand rather than the operand itself.

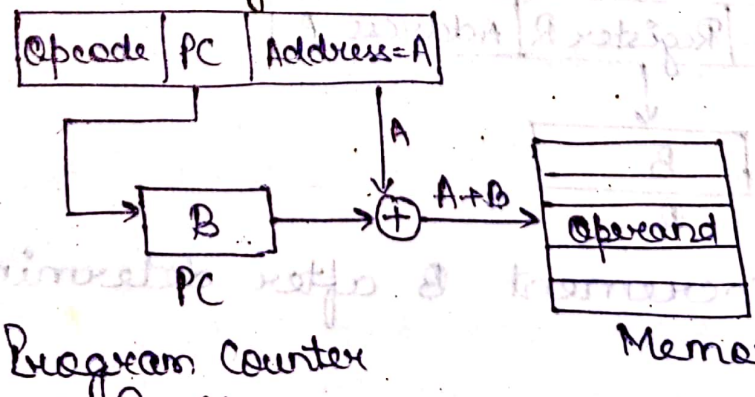
⑤ Immediate Addressing Mode :-




EA = address of instruction = 'A'

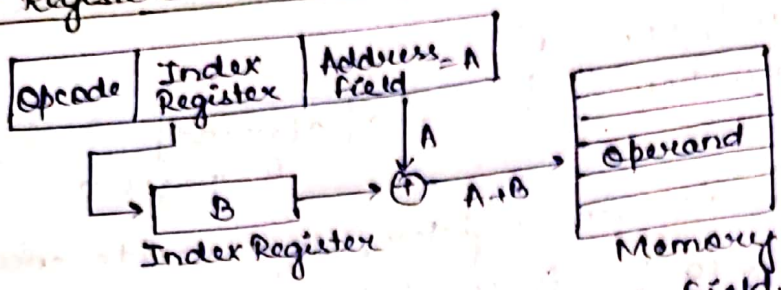
- Operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than an addr field.

⑥ Relative Addressing Mode :-



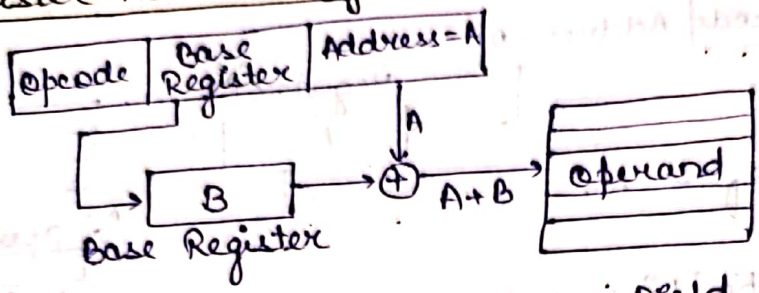
EA = value of PC + address field.

④ Index Register Addressing Mode :-




EA = Index Register Value + Address Field.

⑤ Base Register Addressing Mode :-




EA = Base Register Value + Address Field.

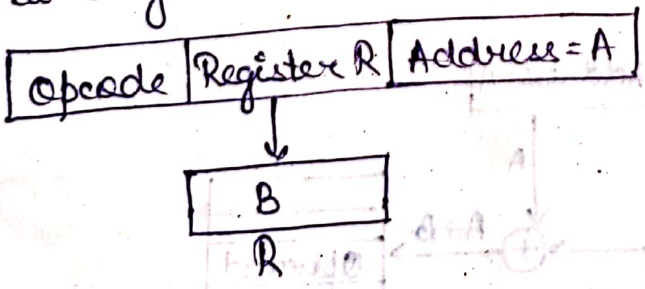
⑥ Implied Addressing Mode :- Operands are specified implicitly in the definition of the instruction.

eg :- CMA
↓
complement the value of accumulator register.

So, EA = Accumulator = AC.

-All register reference instructions that use an accumulator, zero-address instructions in a stack-organized computer are implied-mode instruction.

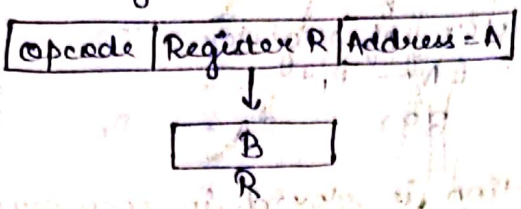
Autoincrement Addressing Mode :-
Similar to register-indirect addressing mode




EA = B  Increment B after determining EA.

(11) Autodecrement Addressing Mode :-

- Similar to register indirect addressing mode.




EA = B-1 ⇒ Before determining effective address, decrement value of R.

example:- A 2-word instruction stored in the memory at an address designated by 200. Find out value of effective address and operand for :-

a) Direct A.M (b) Indirect A.M (c) Immediate Operand Mode.
(d) Relative mode (e) Index Mode (f) Register Direct mode
(g) Register Indirect Mode (h) auto-increment (i) auto-decrement

PC = 200

$R_1$ = 400 → Processor Register

XR = 100

AC

| | | |
|---|---|---|
| 200 | Load to AC | mode |
| 201 | Address = 500 | |
| 202 | Next instruction | |
| 399 | 450 | |
| 400 | 700 | |
| 500 | 800 | |
| 600 | 900 | |
| 702 | 325 | |
| 800 | 300 | |

} 2-word instruction

Sol:-

| Mode | EA | Content of ACC. |
|---|---|---|
| a) Direct A.M | Value of Add. field = 500 | 800 |
| b) Indirect A.M | Memory content of Add. field = 800 | 300 |
| c) Immediate Operand Mode | Address of address field = 201 | 500 |
| d) Relative Mode | Value of PC + Value of Add. field 202 + 500 = 702 | 325 |
| e) Index Mode | Value of XR + Value of Add. field 100 + 500 = 600 | 900 |

f) Register Direct
g) Register Indirect
h) Auto increment
i) Auto-decrement

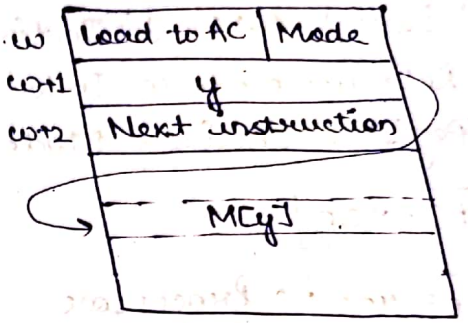
Name of register = $R_1$
Value of $R_1$ = 400
Value of register = 400
EA ← $R_1$+
399 EA ← −$R_1$

400
399
399
450

**Ques.** A- 2-word instruction is stored in memory at address 'w'. The address-field of instruction is stored at (w+1) and is designated by 'y' operands used in during execution of instruction is stored at an address represented by 'z'. An index register (XR) contains value 'x'. State how 'z' is calculated from other addresses if address mode is :-

a) Direct Addressing mode :-
EA = value of add. field = y

b) Indirect Addressing mode :-
EA = memory content of add. field
= M[y]




c) Relative Addressing mode :-
EA = value of PC + value of add. field
= (w+2) + y

PC = w | XR = x

d) Indexed Addressing mode :-
EA = value of XR + value of add. field
= x + y

**Ques.** An instruction is stored at location 300 with its address field at loc. 301. Address field has value 400. $R_1$ = 200 (processor register) (XR = $R_1$).
Calculate effective address if address mode is :-

a) Direct – value of add. field = 400
b) Immediate – add. of add. field = 301
c) Indexed – value of XR + value of add. field
= 200 + 400 = 600
d) Register Indirect – Value of register
= 200
e) Relative – Value of PC + value of add. field
= 302 + 400 = 702