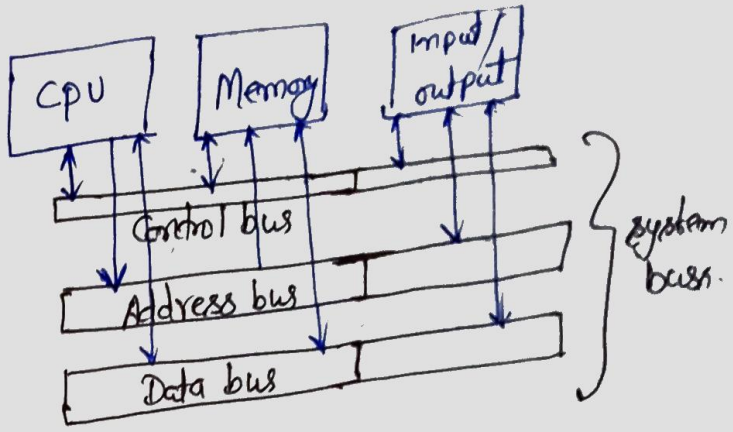
System Buses

Buses is the group of conducting wires which carries information. All the peripherals are connected to microprocessor (CPU) through Bus.
There are three different buses -

(i) Data Bus
(ii) Address Bus
(iii) Control Buss



(i) Data Bus :-

The most common bus is the data bus.
A data bus carries data.
* It is an electrical path that connects the CPU, Memory, Input/output devices and secondary storage devices.
* The number of lines in bus affects the speed at which data travels between different components.
* It is bidirectional.

(ii) Address Bus :-

An address bus carries address information. It is set of wires similar to the data bus but it only connect CPU and

Memory.

* Whenever the processer needs data from the memory, it places the address of data on the address bus.
* The address is carried to the memory where the data from the requested address is fetched and placed on the data bus. The data bus carries to CPU
* It is unidirectional because data flow in one direction from microprocesser (CPU) to memory or from memory to microprocesser. to Input/output devices.

③ Control bus:-

* Control bus carries Control Information the Control unit to the other unit
* The Control Information is used for directing the activities of all unit
* The Control unit Control the functioning of other unit eg. Input/output devices, secondary storage etc

USB (Universal serial bus)
IDE (Integrated Development Environment)

Bus Arbitration:-

Bus Arbitration scheme usually try to balance 'Bus priority' and 'fairness'.

* Bus Arbitration refers the process by which the current master accessed and then leaves the control of the bus and passed it to the another bus requesting processor unit. The controller that has access to a bus at an instance is known as Bus master.

Daisy chain:-

There are two type of approaches to bus arbitration

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

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

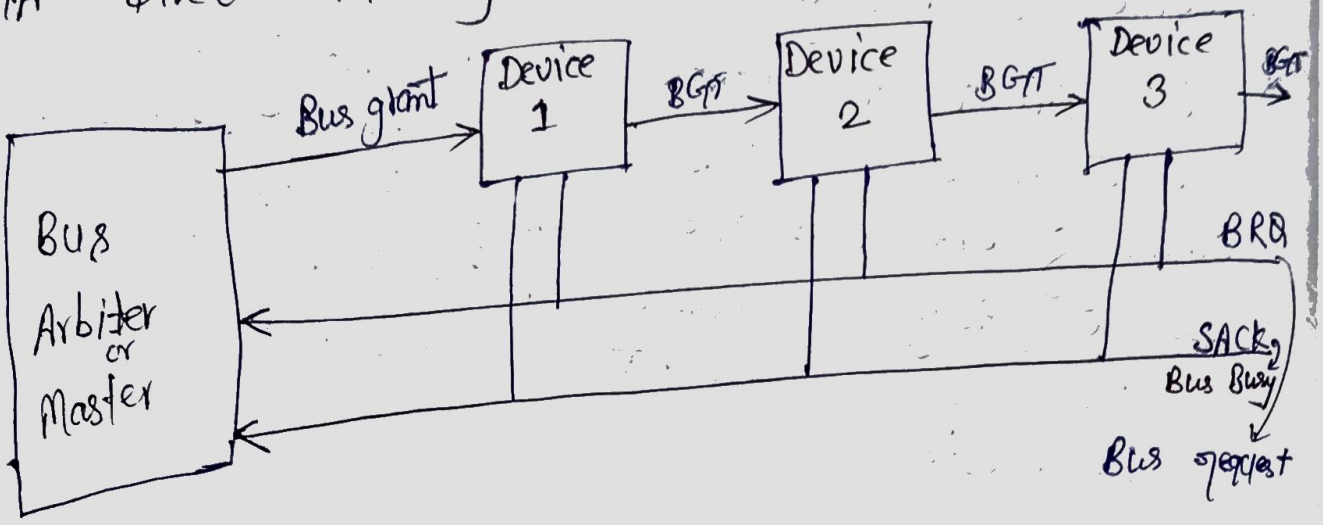
Methods of Bus Arbitration:-

There are three bus arbitration methods:-

① Daisy chaining method:- It is a centralized bus arbitration method. During any bus cycle, the bus master may be any device - the processor or

(IX) any other DMA Controller unit, Connected to the bus.

DMA - Direct memory access.



Advantage:-

(I) simplicity

(II) the user can add more devices anywhere along the chain, up to a certain maximum value.

Disadvantages:-

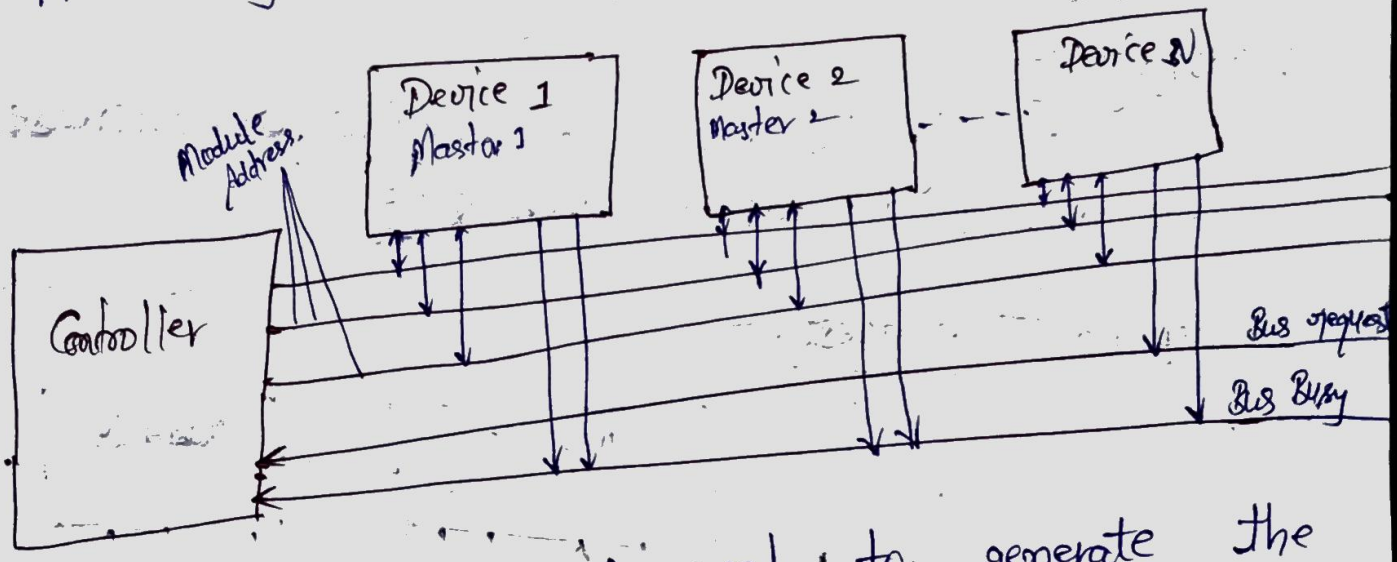
(I) The value of priority assigned to a device it's depends on the position of master bus.

(II) Propogation delay in this method.

(III) If one device fails then entire system will stop working.

⑤ Polling or Rotating Priority method :-

In this method the devices are assigned unique priority and Complete to access the bus, but the priorities are dynamically changed to give every device an opportunity to access the bus.



* In this the Controller is used to generate the address for the master. No of address line required depends on the number of master Connected in the system.

* For example, if there are 8 masters Controller generates Connected in the system, at least three address lines are required [$2^3 = 8$]

for 16 master = $2^4 = 16$ so 4 lines are required

32 " = $2^5 = 32$ so 5 " "

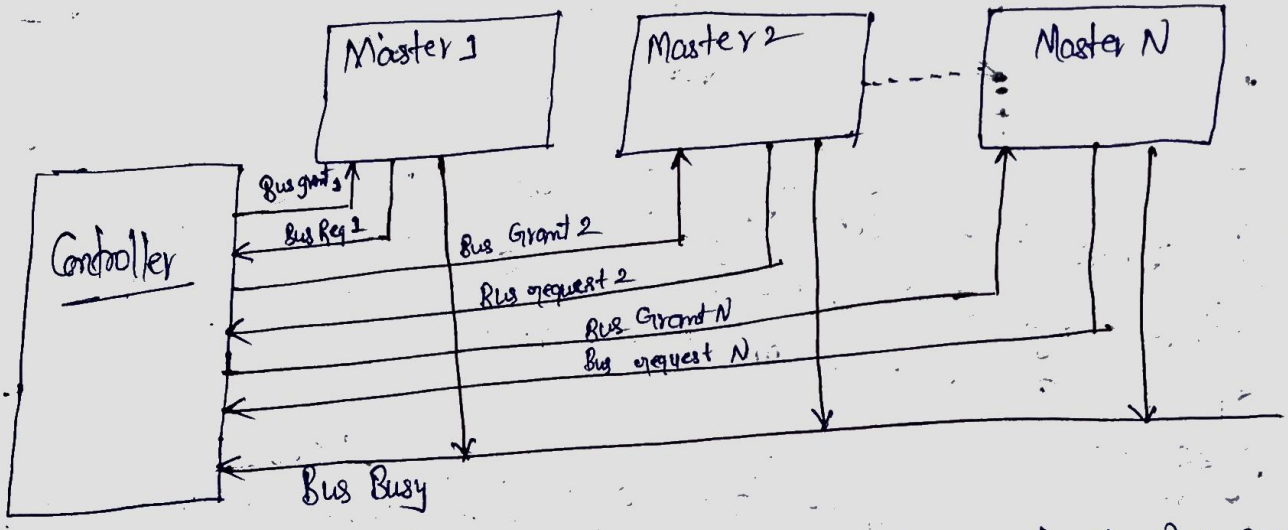
Advantage :-

* This method does not favor any particular device and processer
* The method is also quite simple.
* If one device fails the entire system will not stop working.

Disadvantage :-

* Adding bus master is difficult as increase the number of address lines of the circuit.

③ Independent request :-



In this method each master have seperate pair of Bus grant and bus request lines and each pair has a priority assigned to it.

Bus = bunch of wires

Bus system = bunch of wires + Bus Controller

Advantage :-

* This method generates fast response.

Disadvantage :-

* Hardware Cost is high as large no. of Control lines are required.

—: Digital circuit :—

* DS

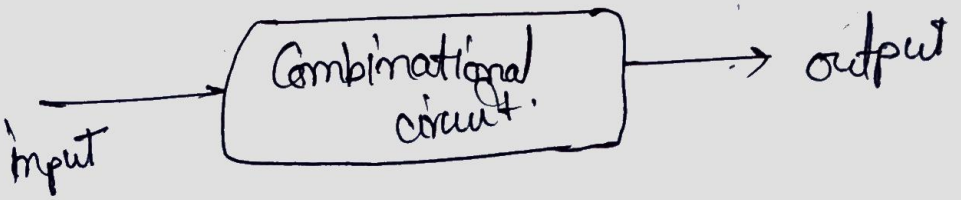
① Combinational circuits :-

Combinational circuits are defined as the time independent circuits which do not depends upon previous input to generate any output.

Advan

* speed is fast
* In this output is depends only upon present input
* It is designed easy.
* Time independent.
* logic gates : Elementry building blocks.
* These circuit do not have memory element.

Eg:- Encoder, Decoder, Multiplexer, DeMultiplexer, Adder



Micro operations are low level instructions. They generally perform operations on data stored in one or more registers. Also perform arithmatic and logic operations on register.

Ex: shift, Count, clear and load.

* The operation executed on data stored in register.

Instructions:- Instructions are a segment of code containing steps that need to be executed by the processor.

Operation:- An operation in mathematics or Computer science, is an action that is carried out to accomplish a given task. There are five basic types of Computer operations: Inputting, Processing, outputting, storing and Controlling.

Register transfer and Microoperations:-

Microoperations:-

* The operation on the data in registers are called microoperation.

* Example of microoperation:- shift, load, clear, Increm

* An elementry operation performed (during one clock pulse), on the information stored in one or more register.

Computer system microoperations are of 4 type.

(I) Register transfer M.O

(II)

(I) Register transfer Microoperation:-
transfer binary inform
from one register to another.

(II) Arithmatic transfer Microoperation:-
perform Arithmatic operations on numeric data stored in register.

(III) Logical Microoperations:-
Perform bit manipulation

(IV) shift operations:-
Perform shift operations on data stored in register.

Note:- higher order bit/byte and lower order bit/byte:-
int is 4 byte in length. The 1st byte is the lower-order byte whereas 4th byte is higher order byte. This is same for bits. 1st bit is lower order bit whereas the 32nd bit is higher order bit. One Nibble is 4 bits in length. 2 Nibble make a byte. The lower order nibble is the first 4 bits. The last 4 bits is the higher order nibbles.

Register Transfer Language :-

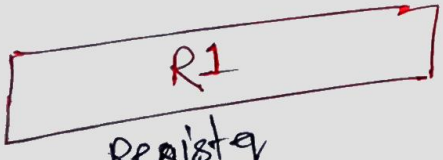
Registers :- Computer Registers are designated by Capital letters sometimes followed by numbers. (A, R1, IR)

For example,

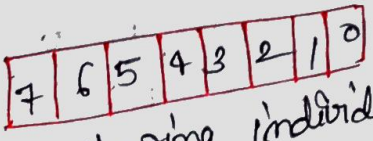
* MAR – Memory Address Register
* PC – Program Counter
* IR – Instruction Register
* R1 – Processer Register.

* A register can be viewed as a single entity.

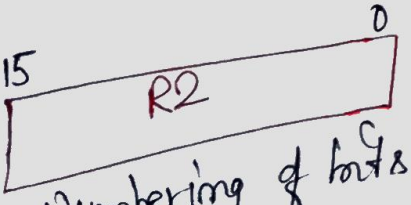
* Registers may also be represented showing the bits of data they contain.



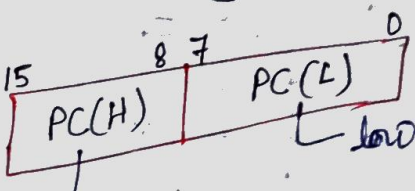
Register



showing individual bits.



Numbering of bits



lower order bits

higher order bits.

* ~~A register~~

Register transfer language :-

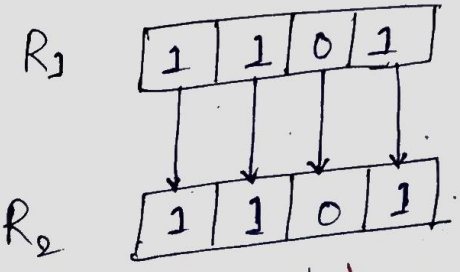
The symbolic notation used to describe the microoperation transfers among registers is called a register transfer language.

* The term "register transfer" implies the availability of hardware logic circuits that can perform a stated microoperation and transfer the result to the same or another register.

* Information transfer from one register to another is designated in symbolic form by means of a replacement operater is known as 'Register transfer'.

$$R2 \leftarrow R1$$

denote a transfer of the content of register R1 into register R2



Register transfer with Control function :-

Normally, we want to transfer to occur only under a predetermined control condition using 'if-then' statement.

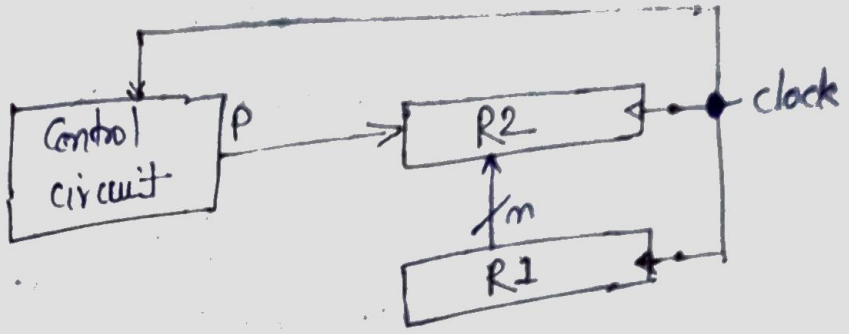
if (P=1) then (R2 ← R1)

where P is Control signal generated in Control section.

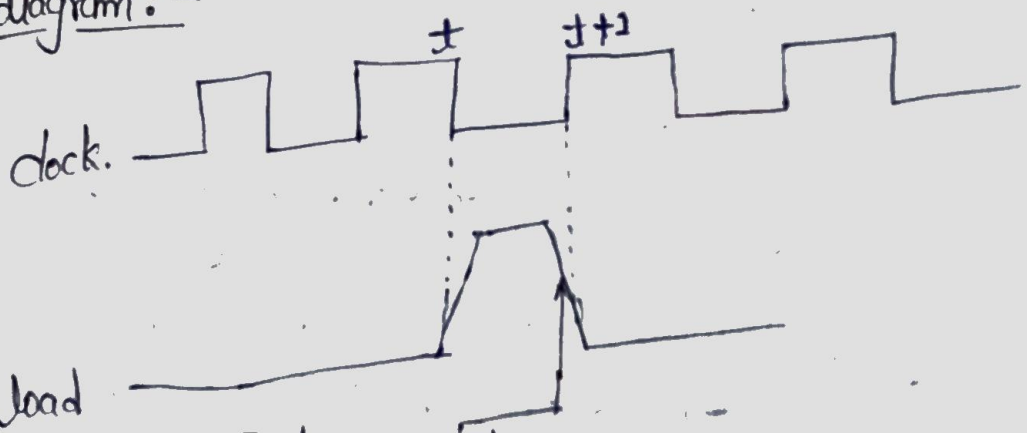
* A Control function is a boolean variable that is equal to 1 or 0.

P: R2 ← R1

Block diagram:-



Timing diagram:-



* If two or more operation occur simultaneously, they are seprated with Commas.

P: R3 ← R5, MAR ← IR

here if the Control function P=1, load the Contents of R5 into R3 and at the same time (clock), load the Content of register IR into MAR.

* Basic symbols for register transfer.

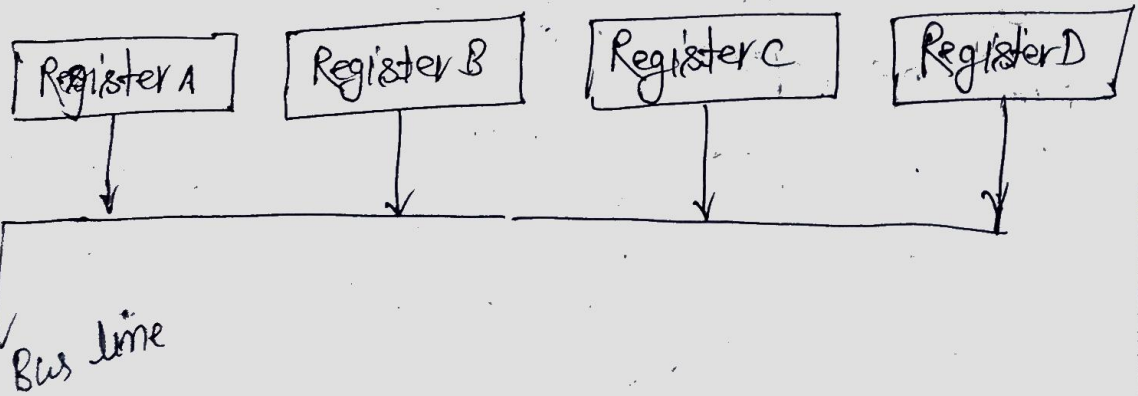
Symbols	Description	Example
Capital letters & numerals.	Denotes a register	MAR, R2
Parentheses ( )	Denotes a part of register	R2(0-7), R2(L)
Arrow ←	Denotes transfer of Information.	R2 ← R1
Colon :	Denote termination of Control function	P:
Comma ,	Separate two microoper	A ← B, B ← A

Bus and bus transfer:-

Common bus system:- for 4 register :-

Bus is a path (of a group of wire) over which information is transferred from any of several sources to any of several destination.

From register to bus: Bus ← R



MUX (4x1) :- takes 4 input and give one output

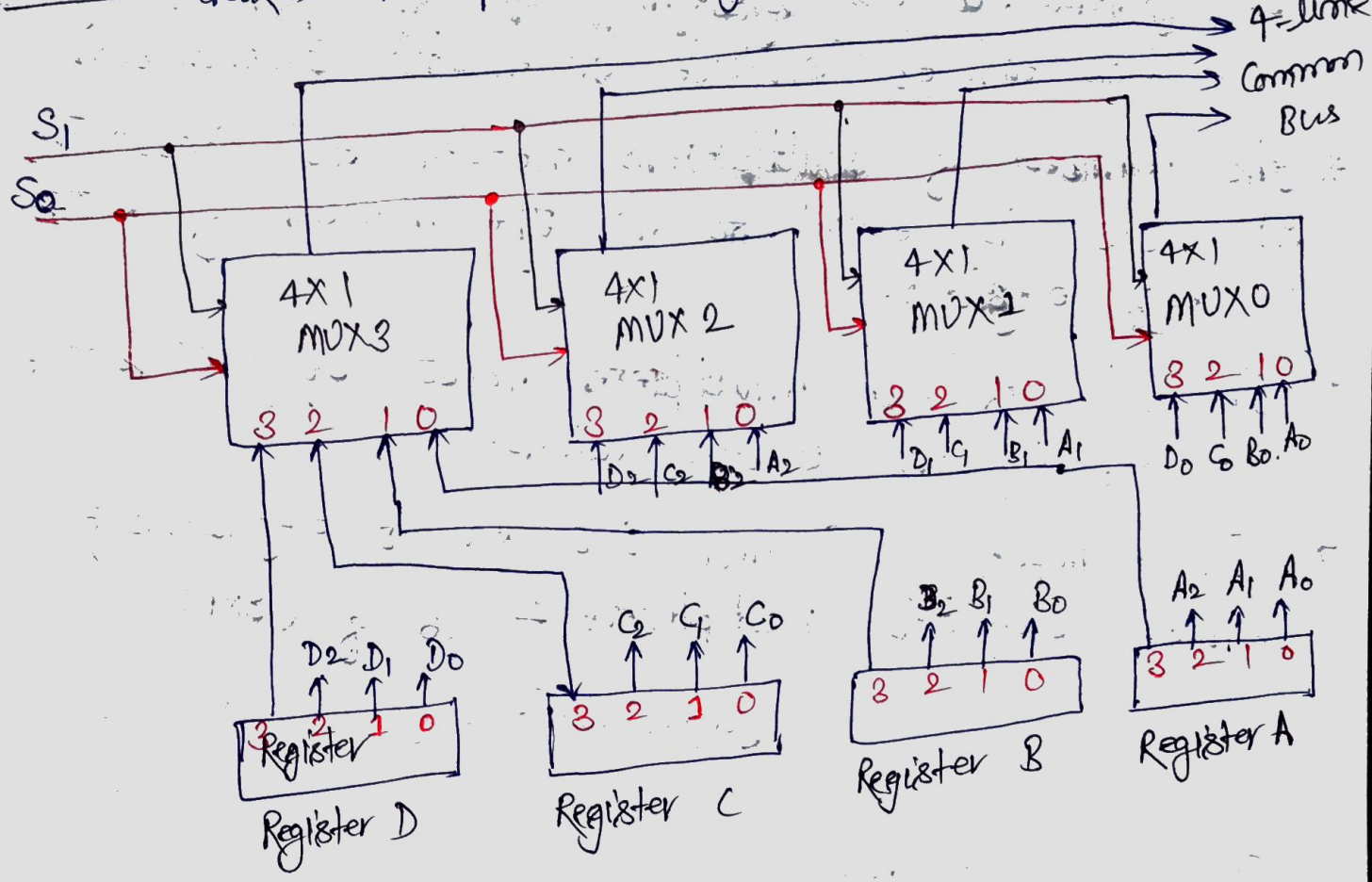


Table shows the register that is selected by the Bus for each of the four possible binary values of the selection lines.

S_1	S_0	Register selected
0	0	A
0	1	B
1	0	C
1	1	D

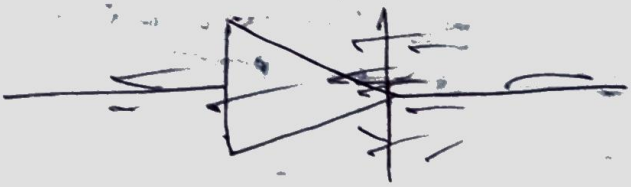
binary
0 0 for 1
0 1 for 1
1 0 for 2
1 1 for 3

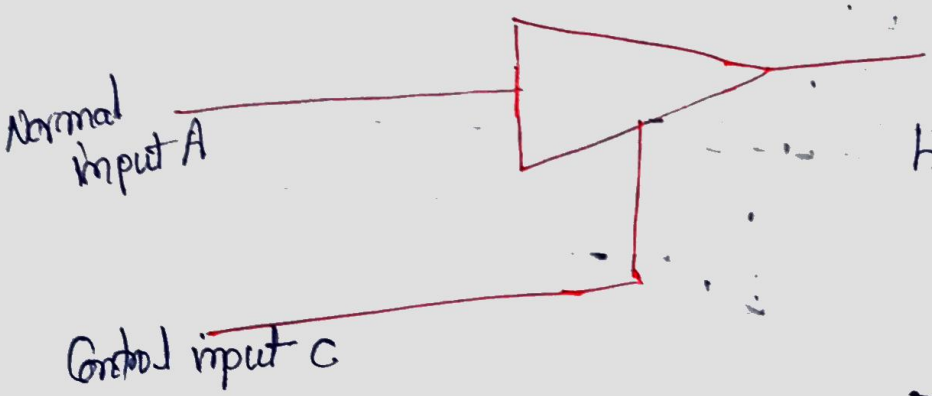
* In general, a bus system will multiplex k registers of n bits each to produce an n-line common bus.
* The number of multiplexers needed to construct the bus is equal to n, the number of bits in each register.
* The size of each multiplexer must be kX1 since it multiplexes k data lines.
* For example, a common bus for 8 registers of 16 bits requires Multiplexers - 16 of (8x1)

Select line = 3. $= \begin{cases} 0:8 = 2^3 \\ 4 = 2^2 \end{cases}$

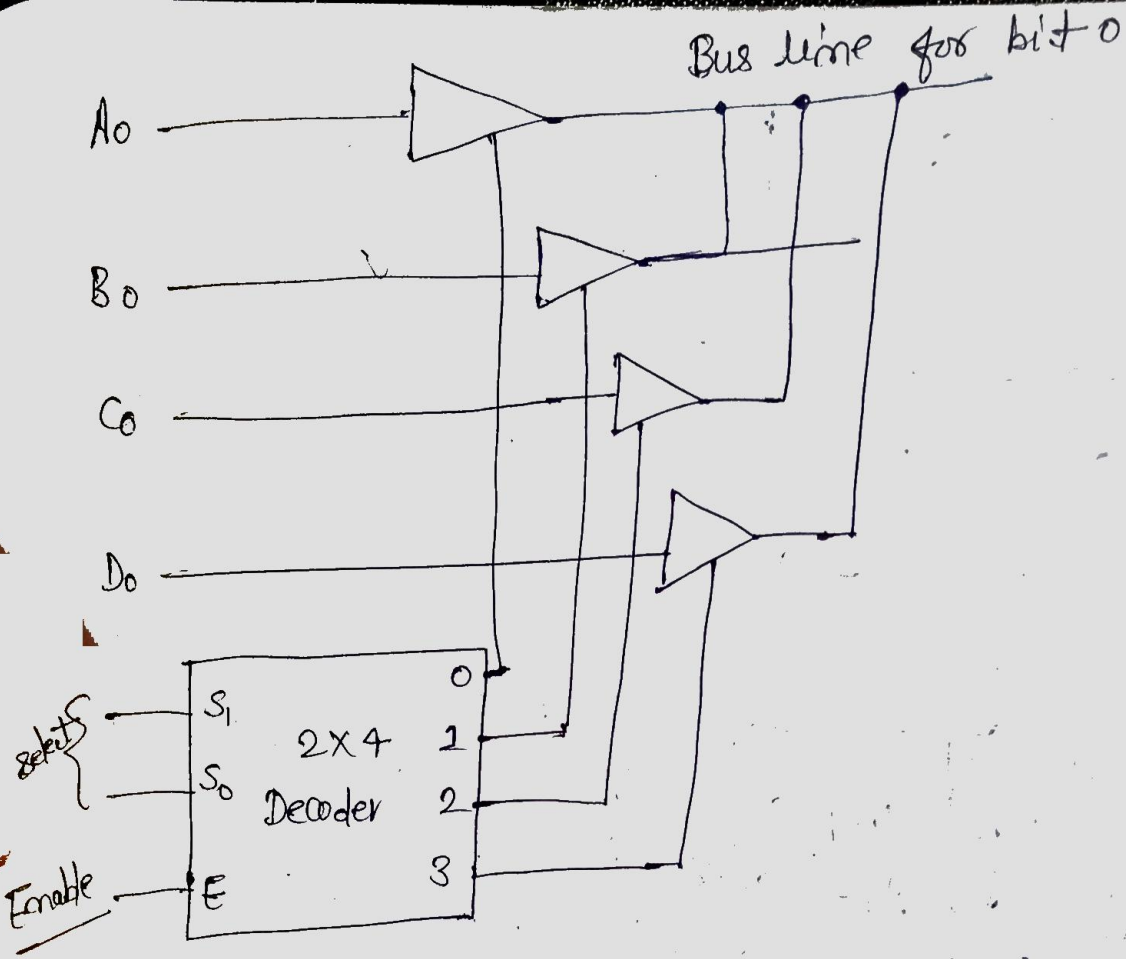
Tri-state buffer :-

* A bus system can be constructed with three state gates instead of multiplexes.
* 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 a conventional gate.
* A third state is high impedance state which behaves like an open circuit, which means that the output is disconnected and does not have logic significance.



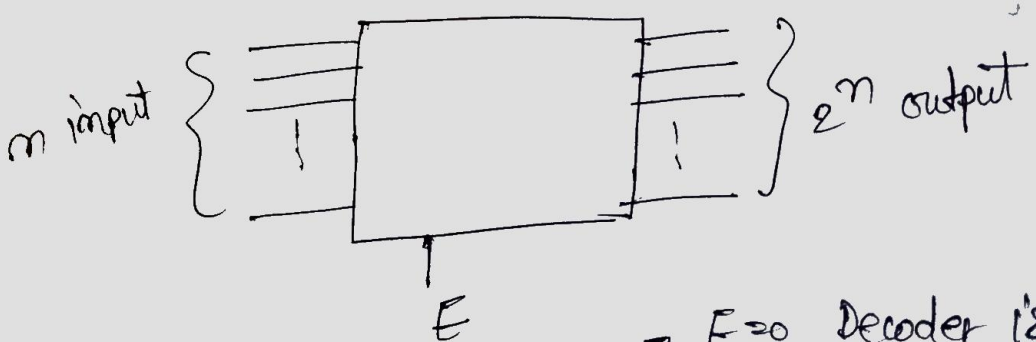


* The Control input determined the output state. When the Control input C is equal to 1, the output is enabled and the gates behaves like any conventional buffer, with the output equal to the normal input.
* When Control input C is 0, the output is disabled and the gate goes to a high impedance state regardless of the value in the normal input.



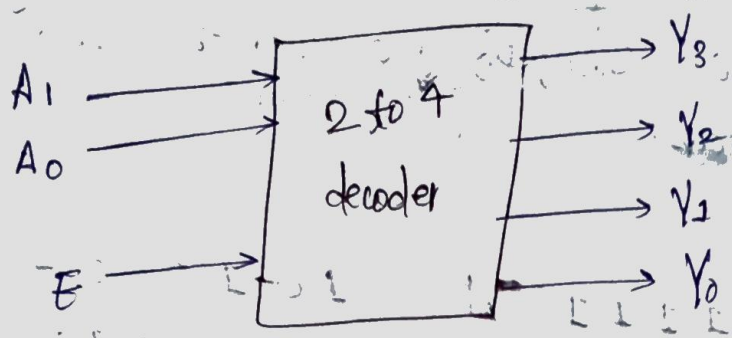
Decoder:-

Decoder :- it's a Combinational Circuit that has n input lines and maximum of 2^n output lines one of these outputs will be active High based on Combination of inputs present, when the decoder it's enabled.



where E = Enable
- E=0 Decoder is disable
- E=1 Decoder is enable.

2 to 4 decoder:-



			2^3	2^2	2^1	2^0
E	A	B	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
	0	1	0	0	1	0
	1	0	0	1	0	0
	1	1	1	0	0	0

Binary.
- 00 for 0
- 01 for 1
- 10 for 2
- 11 for 3

Memory transfer:-

* The transfer of information from a memory word to the outside environment is called a read operation.
* The transfer of new information to be stored into the memory is called write operation.
* A memory word is symbolized by the letter M.
* The read operation can be stated as follows.

Read: $DR \leftarrow M[AR]$

* The write operation can be stated symbolically as follows.

Write: $M[AR] \leftarrow R1$

- DR - data Register
- AR - address "
- AR - Memory Address Register
- R1 - Register name.
- M → memory Ram

Complements:-

Complements are used in the digital Computers in order to simplify operation and for the logical manipulations.

(i) 1's Complements:-

01111111
1's Comp. 10000000

10·10001
Compl. 01011110

(ii) 2's Complements:- (1's Complement + 1)

101
1's Cmpt 010
2's Complement = 010 + 1 = 011

Subtraction (Binary).

1111
1010
0101 - Direct Method

1's Complement method

~~1111~~ number
~~0000~~ 1's Complement
+1
~~10~~

1's Complement method

1111
1010

First we have to Convert number that have to subtract in 1's Complement

1010

1's Complement - 0101

then 1111
0101
+
1 0100
+1
0101

0-0 = 0
0-1 = 1 (Borrow 1)
1-1 = 0
1-0 = 0

1010
- 101
0101

(4) A = 1100
B = 0101

1's Complement of B = 1010

then 1100
+ 1010
1 0110
+1
0111

we have to add this because addition of 4 bit number will have not be 5 bits number.

Binary of negative number :-

(-8)

Binary of 8 is 1000

1's Complement = 0111

2's Complement = 0111 +1 (1000)

(-5) 101

1's = 010

2's = 010 +1 (011)

Addressing Modes:-

* The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.
* Computers used addressing mode technique for the purpose of accommodating one or both of the following provisions.
  (i) To give programming versatility to the user by providing such facilities as pointer to memory, counter for loop control, indexing of data, and program reallocation.
  (ii) To reduce the no. of bits in the address field of the instruction.

There are 10 basic addressing mode given below-

1. Implied mode
2. Immediate mode
3. Register mode
4. Register Indirect mode.
5. Autoincrement or Autodecrement mode
6. Direct Address mode
7. Indirect Address mode
8. Relative addressing mode
9. Indexed Addressing mode
10. Base Register addressing mode.

Immediate Addressing :-

(specified)

* operand is given explicitly in the instruction. (i.e operand = value)

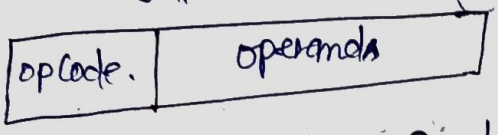
eg. ADD 5 i.e Add 5 to contents of accumulator where 5 is operand.

* No memory reference to fetch data
* Fast but limited range

Eg. MOV R1, 05H,
(Immediate number)

i.e MOV 05H to R1.

* It is useful for initializing register to Constant value.



* opCode - It is a part of instruction also known as Instruction Code.

(2) Direct addressing :-

* In this mode the effective address is equal to the address part of instruction.

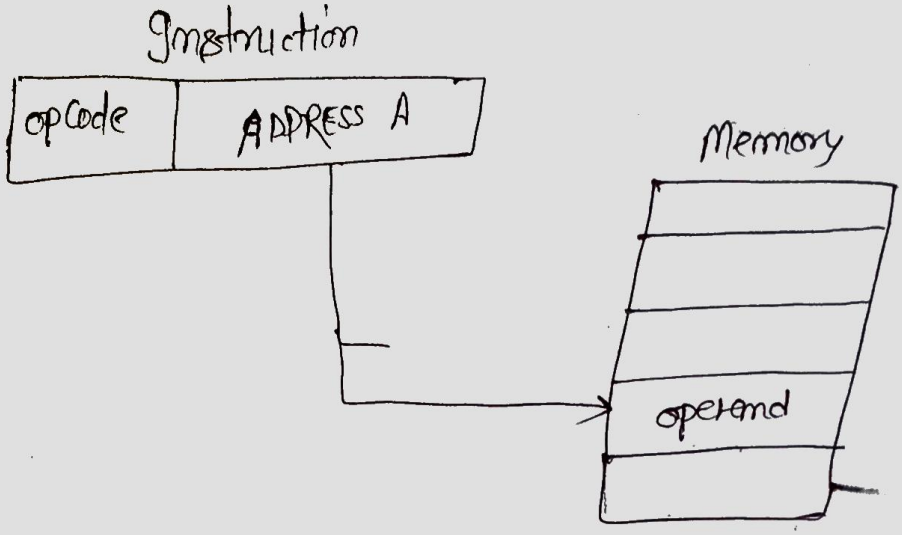
effective address (EA) = address field (A)

Note - An effective address is any operand to an instruction which references memory.

* The operand resides in memory and its address is given directly by the address field of the instruction.

eg. ADD A i.e Add Content of cell A to accumulator and look in memory at address A for operand.

* No addition Calculation to work out effective address.
* Limited address space.



③ Indirect Addressing :-

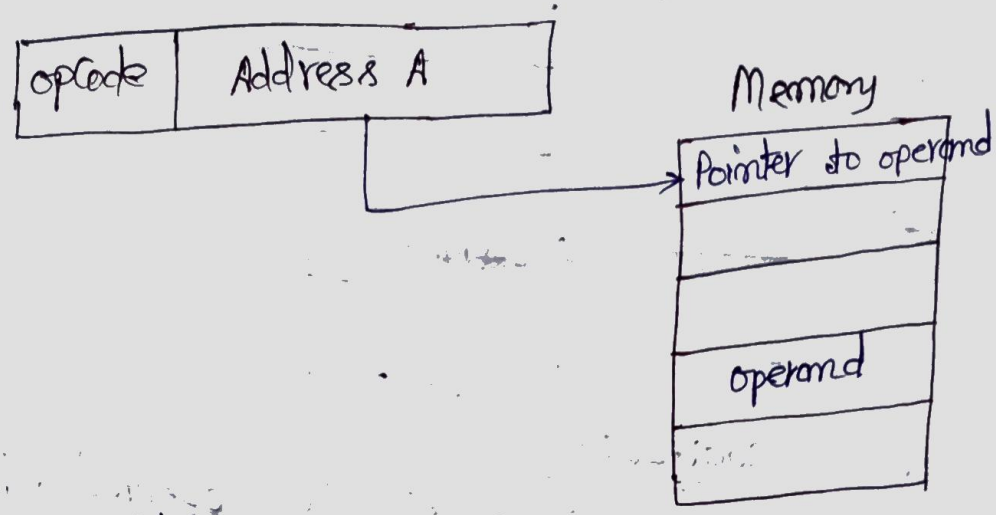
* In this mode the address field of the instruction gives the address where the effective address is stored in memory.

Effective address = address part of the instruction + Content of CPU register.

Eg. EA = [A] i.e look in [A] and look there for operand
* ADD (A) i.e add Contents of cell pointed to by Contents of A to accumulator.
* may be nested, Multilevel.

eg. EA = (((A))) i.e multimemory accessed to find operand

* Hence it is slower



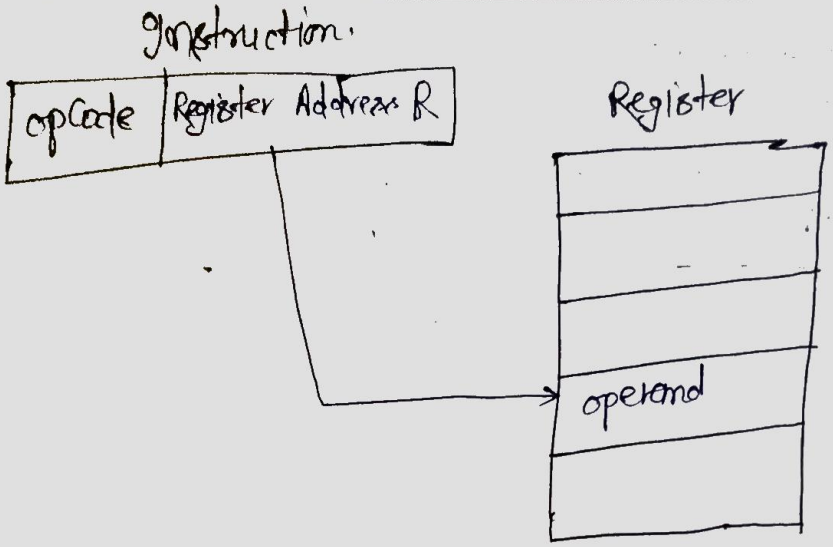
④ Register Addressing :-

* operand are in register named in address field.

* effective address = R
* limited number of register
* Very small address field needed, shorter instructions, faster instruction fetch
* No memory access.
* Very fast execution.
* Very limited address space.

eg. MOV AX, BX

i.e move value from BX to AX register.

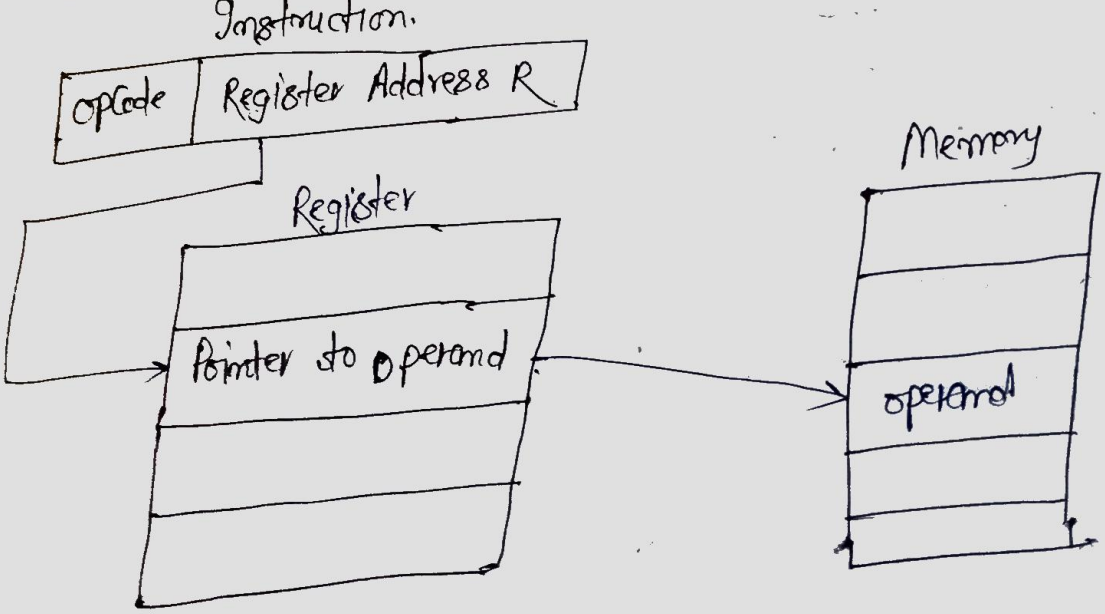


④ Register indirect address:-

* EA = [R] i.e operand is in memory cell pointed to by content of Register R.
* large address space (2n)
* one fewer memory access than indirect addressing.

eg. MOV [R1], R2

i.e value of R2 is moved to the memory location specified in R1.



⑤ Relative Addressing:-

* In this mode the Content of Program Counter is added to the address part of the Instruction in order to obtain the effective address.

{ Effective address = address part of instruction + Content of PC }

* The address part of the Instruction is usually a signed number which can be either +ve or negative.

eg: EA = X + (PC)

i.e get operand from X byte away from current location pointed to by PC.

⑥ Indexed Addressing:-

* In this mode the Content of an Index register is added to the address part of the instruction to obtained the effective address.

* EA = address part of instruction + Content of index register.

* Index register is a special CPU register that Contain an index value.

* It is also used in array indexing.

EA = X + [R]

:- Processor organisation :-

Requirements placed on the processor :-

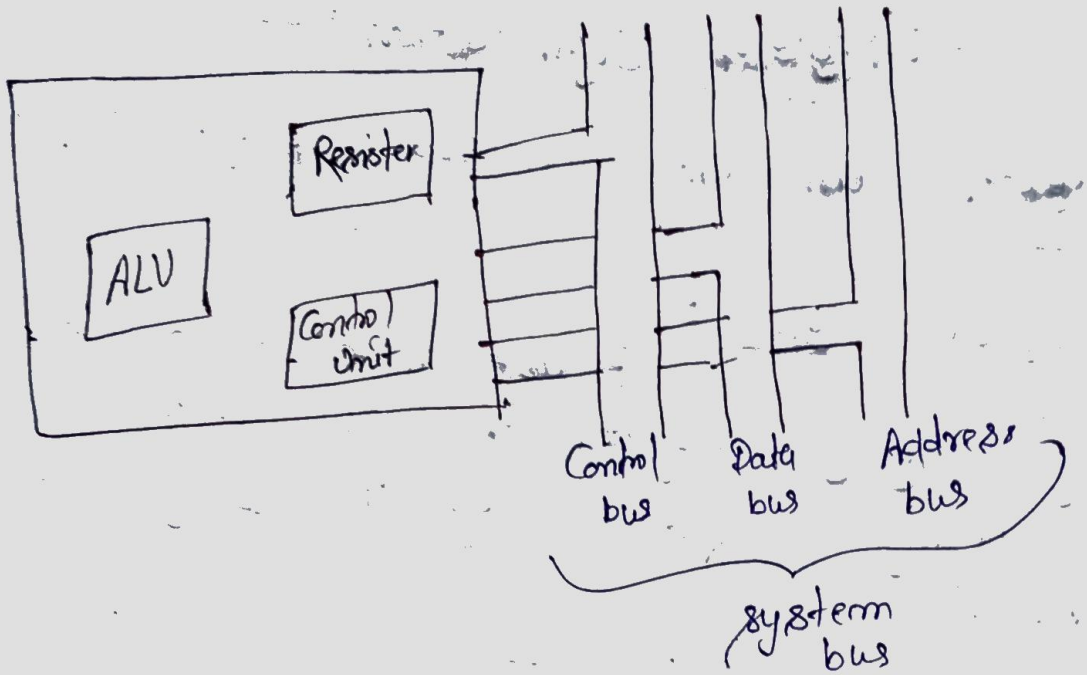
① **Fetch Instruction :-** The processor reads an instruction from memory (register, cache, main memory)

② **Interpret Instruction :-** Instruction is decoded to determine what action is required.

③ **Fetch data :-** The execution of an instruction may required data from memory or an I/O module.

④ **Process data :-** The execution of an instruction may require performing some arithmatic and logical operation on data

⑤ **Write data :-** The result of an execution may require writting data to memory or I/O module



Components of Processor:-

* The major components of the processor are an arithmetic and logic unit and a control unit (CU).
* ALU does the actual computation or processing of data
* Control unit control the movement of data and instruction into and out of the processor, controls the operation of ALU.
* Register consists of a set of storage locations.
* The data transfer and logic control paths are indicated, including an element labeled internal processor bus.

-: stack organisation :-

There are two type of stack organisation.

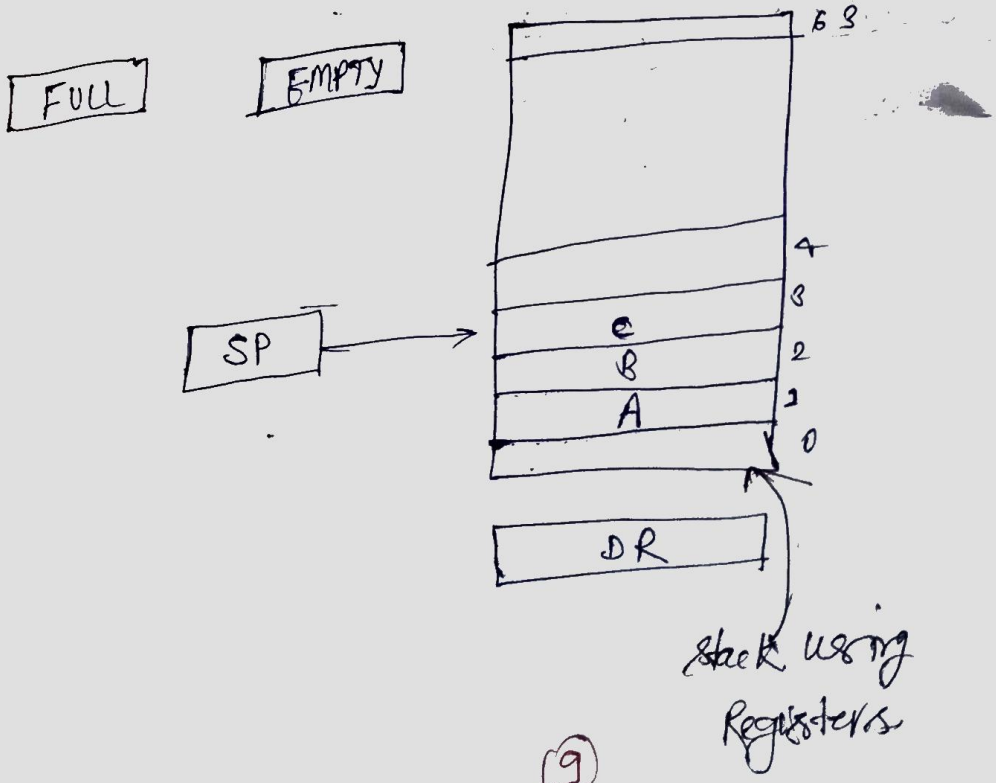
(1) Register stack :- built using register

(2) Memory stack :- logical part of memory allocated as stack

* A stack is a storage device that stores information in such a manner that the item stored last is first item retrived (LIFO).

* The register that hold the address for the stack is called a stack pointer (SP) because its value always point at the top item in the stack.

(1) Register stack :-



we also use Registers FULL and EMPTY along with SP. If stack is full then FULL = 1 and is not full then Register FULL = 0. Similarly when stack is empty Register EMPTY = 1 and if not then EMPTY = 0. And one more register we use that i.e DR (data reg.) through which data is transfered to stack

*PUSH operation

SP ← SP + 1

M[SP] ← DR

if (SP = 0) then (FULL ← 1)

EMPTY ← 0

* POP operation.

DR ← M[SP]

SP ← SP − 1

if (SP = 0) then (EMPTY ← 1)

FULL ← 0

———— * ————