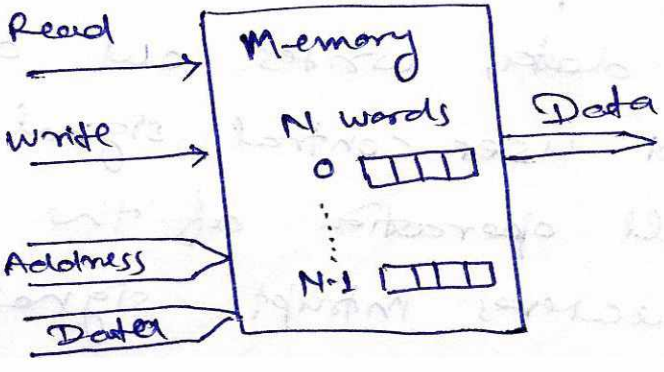
Functional Units and their Interconnection :-

A computer consist of a set of components or modules of three basic typs (processor, memory, I/O) that communicate with each other. Thus, there must be for connecting the modules.

The collection of paths connecting the various modules is called interconnection structure.

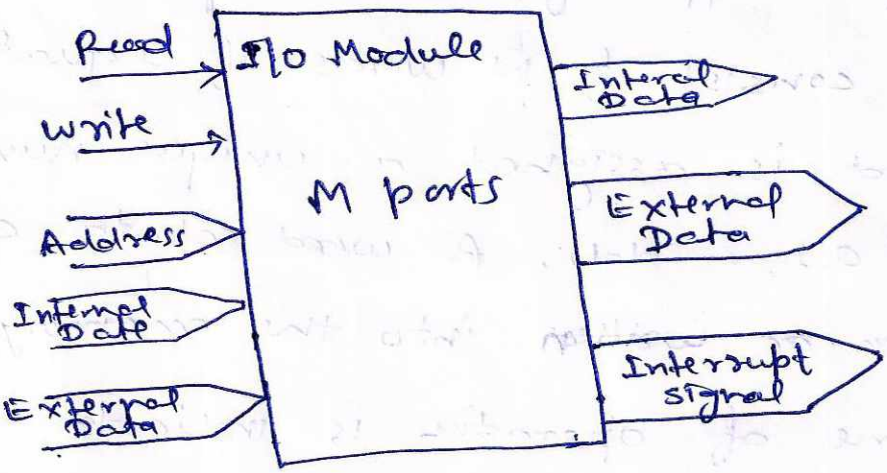
Functional Unit :-

(i) **Memory** - Typically, a memory module will consist of N words of equal length. Each word is assigned a unique numerical address (0, 1, ... N-1). A word data can be read from or written into the memory. The The nature of operation is indicated by read and write control signal. The location for the operation is specified by an address.



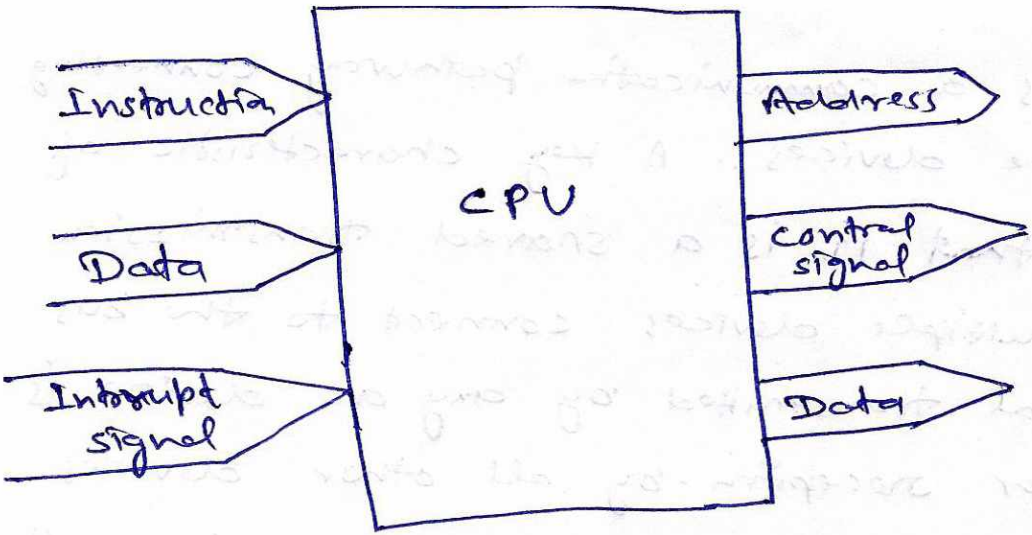
(ii) I/O Module :-

From an internal (to the computer system) point of view, I/O is functionally similar to memory. There are two operations, read and write. Further, an I/O module may control more than one external device. We can refer each of the interfaces to an external device as a port and give each a unique address (eg 0, 1 ... M-1) I/O module may be able to send interrupt signals to the processor.



(iii) Processors :-

The processor reads in instructions and data, writes out data after processing, and uses control signal to control the overall operation of the system. It also receives interrupt signal.



Computer Modules

The interconnection structure must following type of transfer.

1. Memory to processor
2. Processor to memory
3. I/O to processor
4. Processor to I/O
5. I/O to or from memory.

Bus Interconnection :-

A bus is a communication pathway connecting two or more devices. A key characteristic of a bus is that it 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 signal will overlap, thus only one device at a time can successfully transmit.

A bus consist of multiple communication pathways. Each line is capable of transmitting signals representing binary 1 and binary 0. Overtime a sequence of binary digit can be transmitted across a single line. Taken together, several lines of a bus can be used to transmit binary digits simultaneously (in parallel).

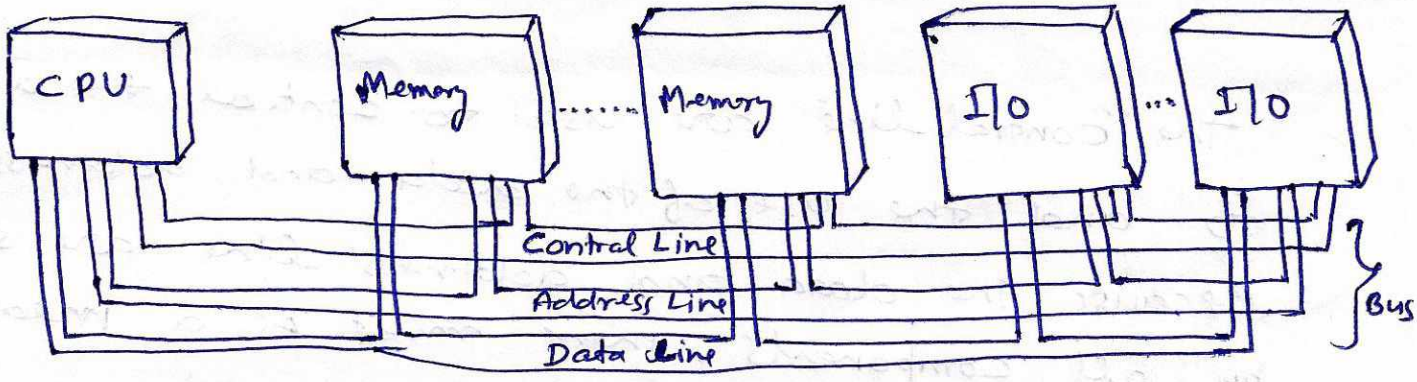
eg. an 8-bit unit of data can be transmitted over eight bus line.

Computer system contains a number of different buses that provide pathways between component at various levels of the computer system hierarchy.

A bus that connect major computer components (processor, memory, I/O) is called the system bus.

Bus Structure :-

A system bus consists, typically about 50 to hundreds of separate lines. Each line is assigned a particular meaning or function. Although there are many different bus design on any bus the lines can be classified into three functional group.



Bus Interconnection Scheme

The "data line" provide a path for moving data between system modules. These lines collectively are called the data bus. The data bus may consist of from 32 to hundreds of separate lines, the number of lines being referred as the width of data bus.

Because each line can carry only 1 bit at a time, the number of lines determines how many bits can be transferred at a time.

The "address line" are used to designate the source or destination of the data on the data bus. For example, if the processor wishes to read a word (8, 16 or 32 bit) of data from memory, it puts the address of the desired word on the address line.

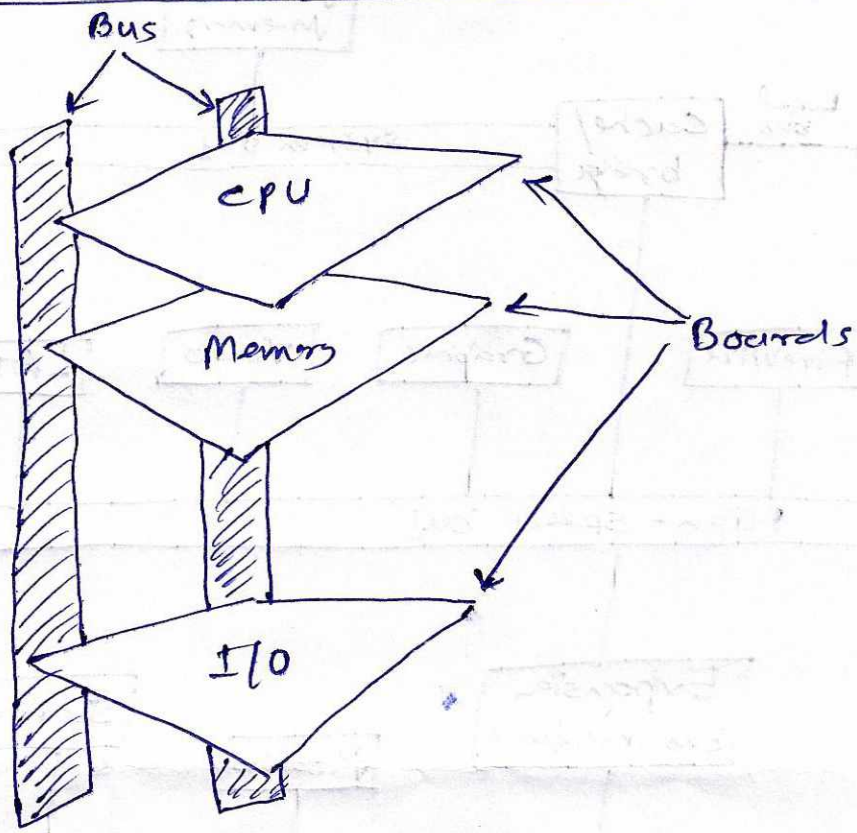
The address lines are generally also used to address I/O ports.

The "control line" are used to control the access to and the use of the data and address line. Because the data and address line are shared by all components, there must be a means of controlling their use.

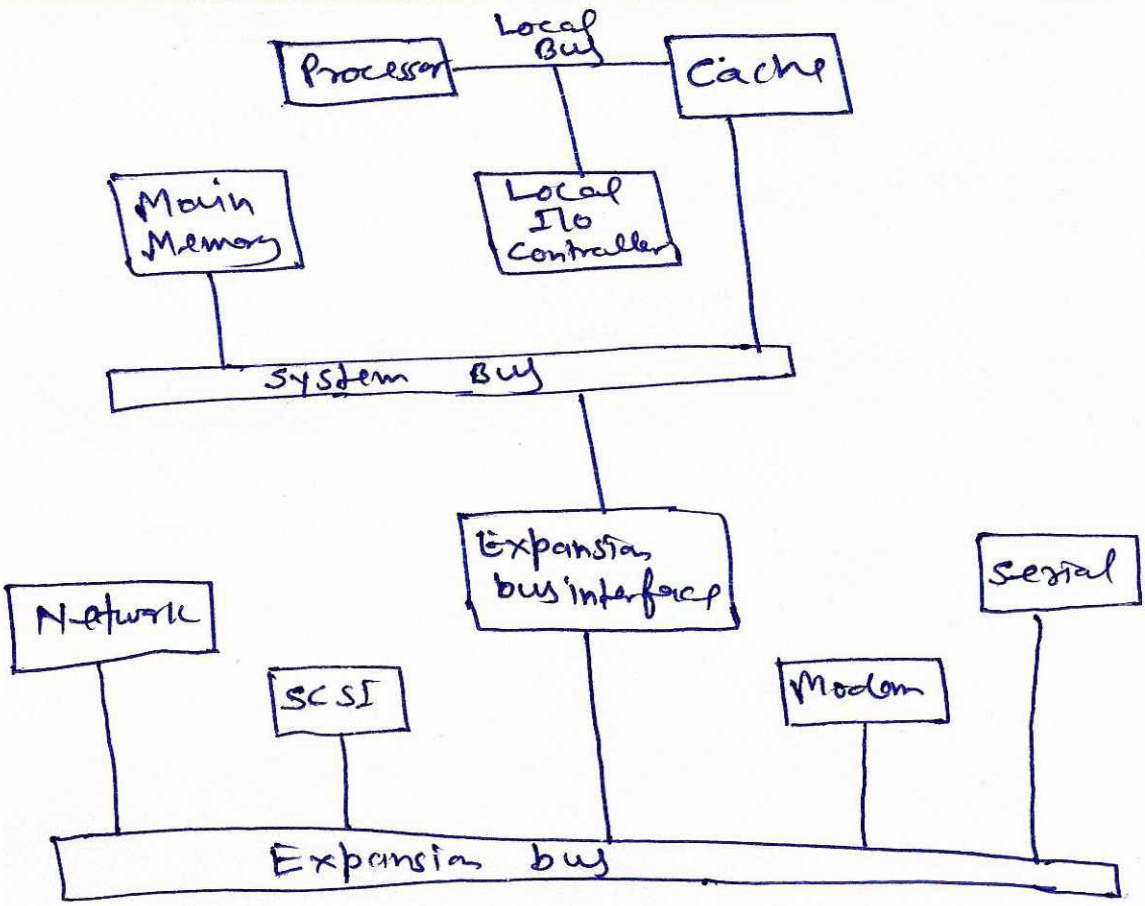
Typical control line include:

(i) Memory write
(ii) Memory read
(iii) I/O write
(iv) I/O read
(v) Transfer ACK
(vi) Bus request
(vii) Bus grant
(viii) Interrupt request
(ix) Interrupt ACK
(x) clock
(xi) Reset.

Physical Realization of a Bus Architecture :-

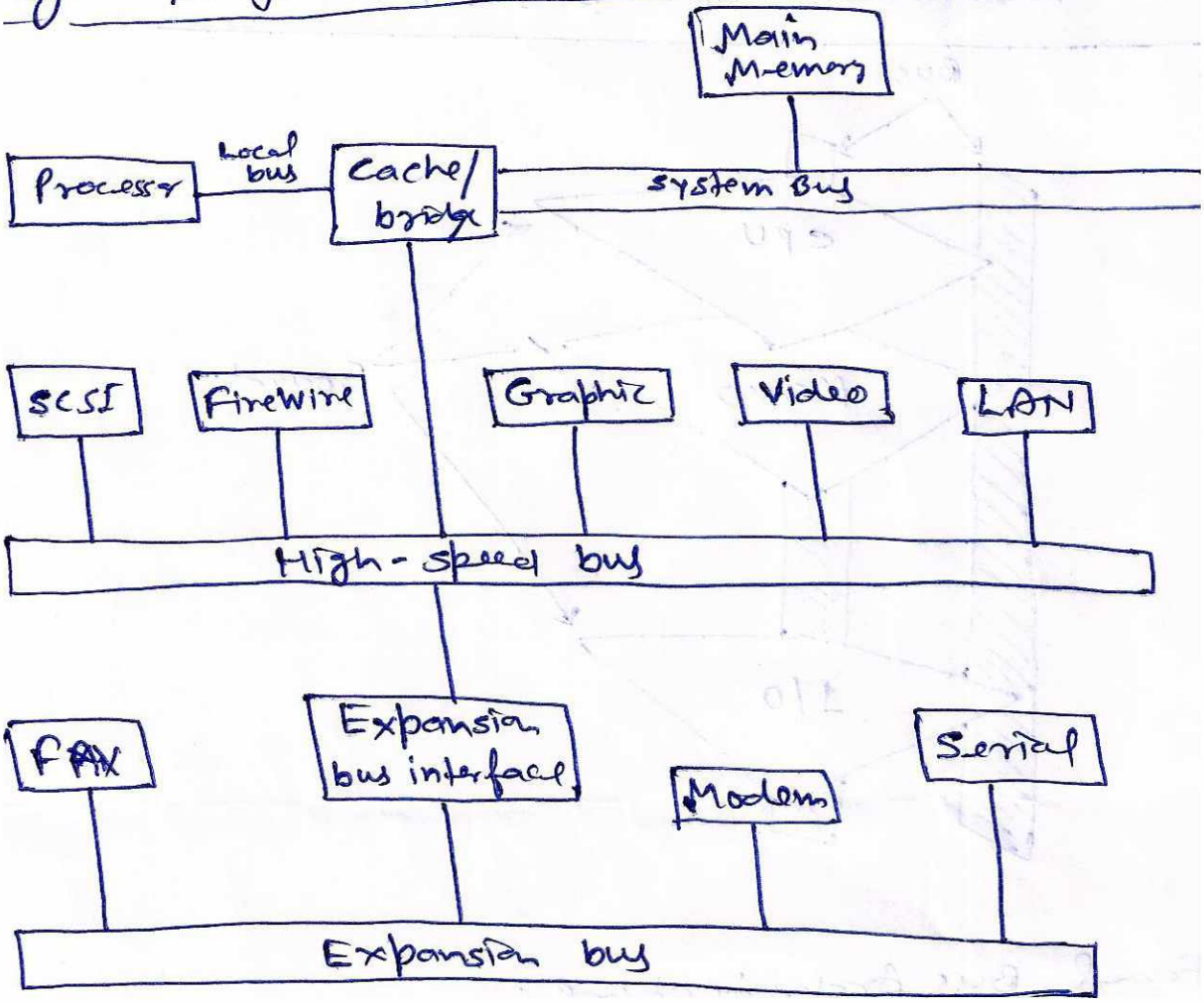


Traditional Bus Architecture :-



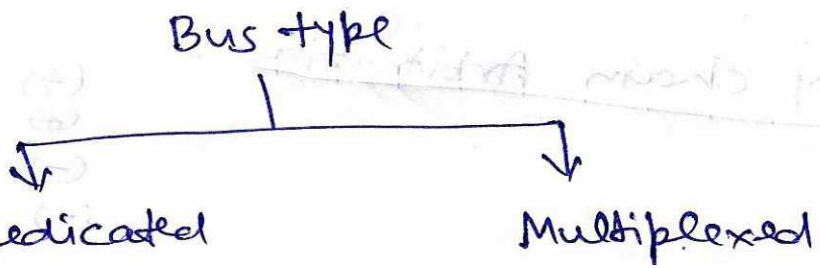
* SCSI - Small Computer System Interface

High - performance Architecture :-



Bus Type :- Bus line can be separated into two generic types: "dedicated" and "multiplexed". A dedicated bus line is permanently assigned either to a function or to a physical subset of computer components.

In multiplexed bus type address and data information may be transmitted over the same set of line using an Address Valid control line. At the beginning of a data transfer, the address is placed on the bus and the Address Valid line is activated. After that the address is then remove from the bus the same bus connection are used for the subsequent read or write data transfer.



Bus Arbitration :-

Multiple devices may need to use the bus at the same time so must have a way to arbitrate multiple requests.

Bus arbitration scheme usually try to balance

(i) Bus priority: the highest priority device should be served first

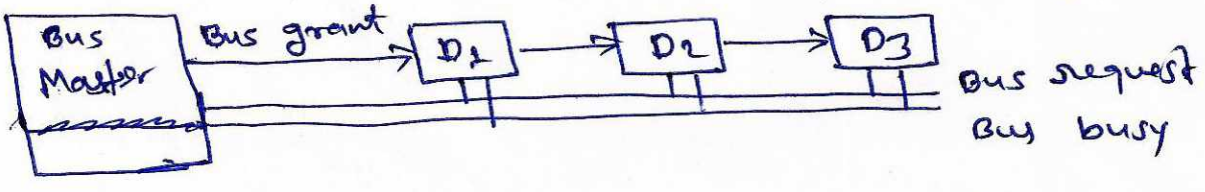
(ii) Fairness: even the lowest priority device should never be completely locked out from the bus.

Bus Arbitration scheme can be devided into four classes:

(i) Daisy chain arbitration
(ii) centralized paraller arbitration
(iii) Distributed arbitration by self selection.
(iv) Distributed arbitration by callision detection:

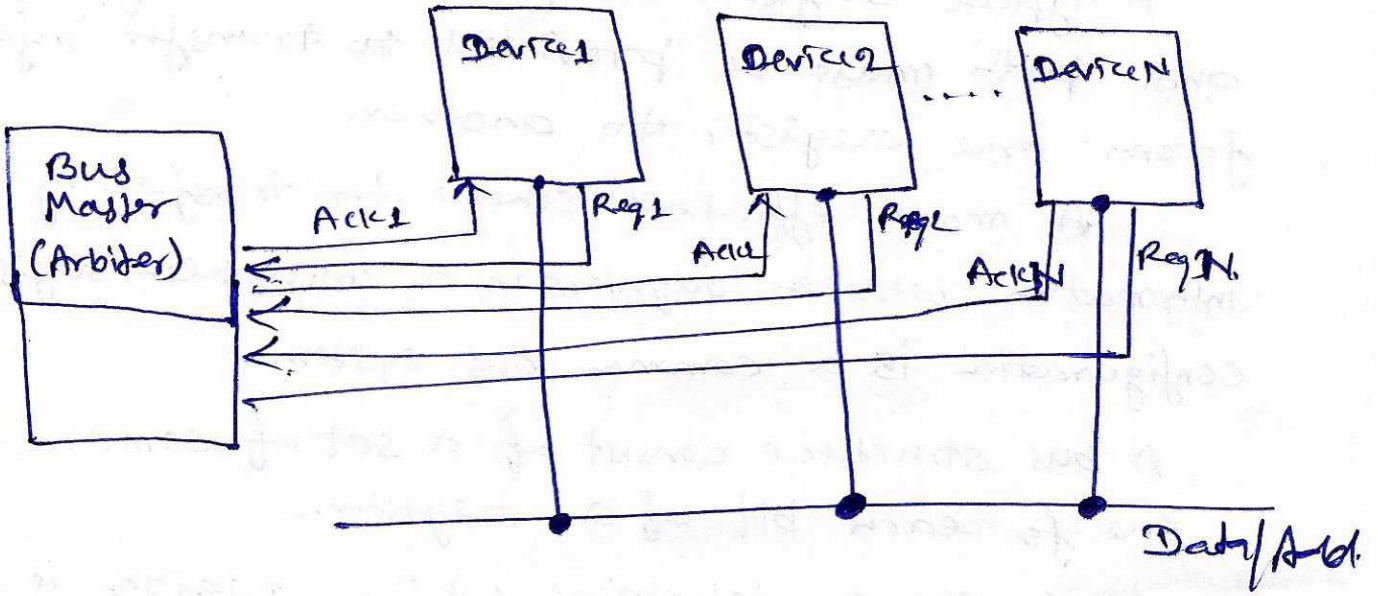
(i) Daisy chain Arbitration :-

(+) Simple
(+) Only three extra bus line
(−) Poor fault tolerance
(−) Delay



→ If bus not busy, make bus request
→ Master activates bus grant
→ If device get bus grant, mark bus busy.

(ii) Centralized, parallel Arbitration :-



(+) flexible, can assume fairness

(-) more complecated h/w

→ Used in essentially all processor memory buses and in high speed I/O buses.

(iii) Distributed arbitration by self selection: each device wanting the bus places a code indicating its identity on the bus.

(iv) Distributed arbitration by collision detection:

device uses the bus when its not busy and if a collision happen (because some other device also decides to use the bus) then the device tries again later.

Resister, Bus and Memory Transfer

A typical digital computer has many registers and path must be provided to transfer information from one register to another.

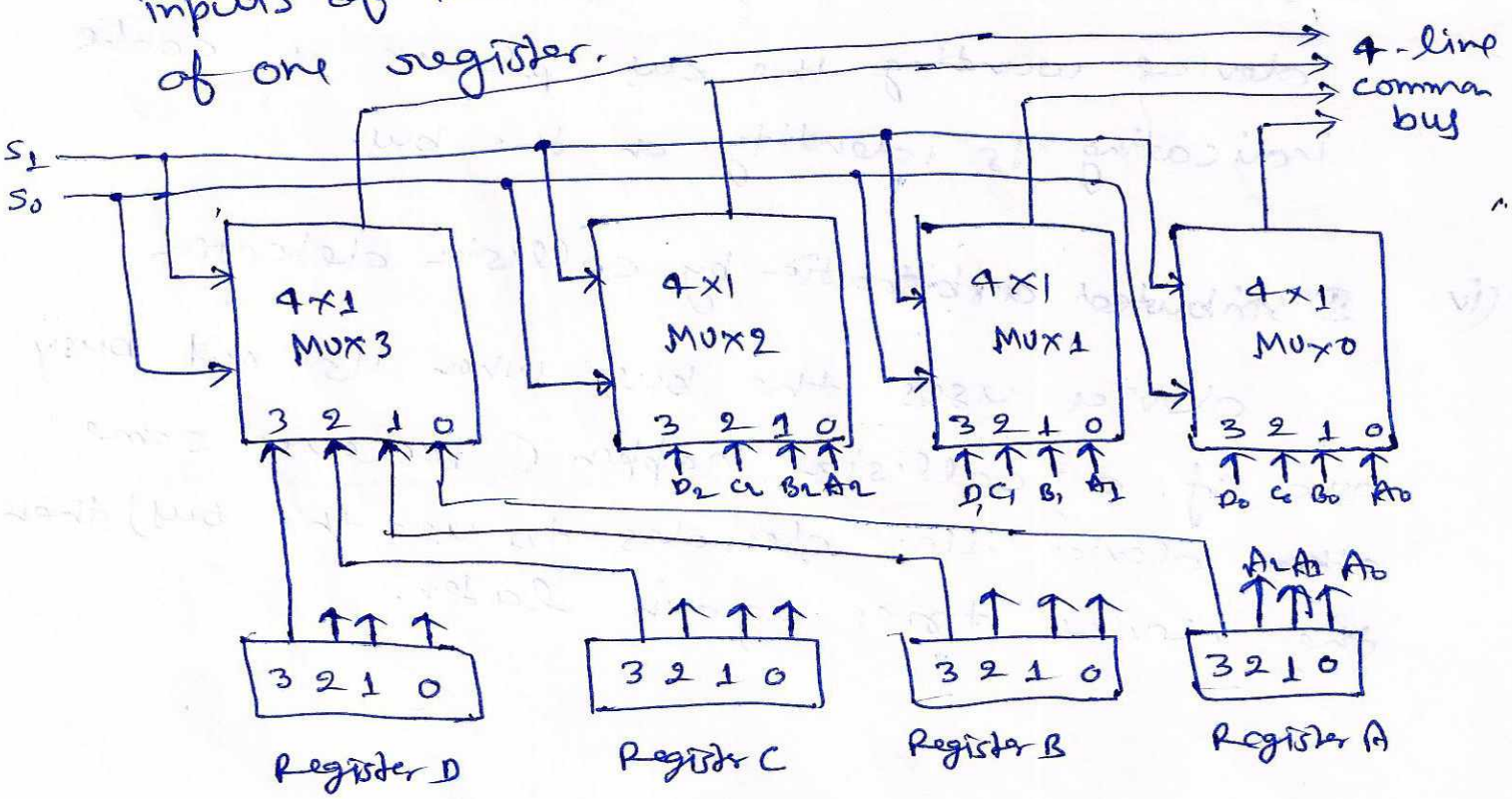
A more efficient scheme for transferring information between registers is a multiple-register configuration is a common bus system.

A bus structure consist of a set of common line, one for each bit of a register.

Control signal determine which register is selected by the bus during each transfer.

Multiplexers can be used to construct a common bus. Multiplexer select the source register whose binary information is then placed on the bus.

The select line are connected to the selection inputs of the multiplexers and choose the bit of one register.



Bus system for four Register

Memory Transfer :-

Consider a memory unit that receives the address from register, called the address register, symbalized as AR. The data are transferred to another register, called data register, symbalized by DR, the read operation can be stated as follows:

Read: $DR \leftarrow M[AR]$

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

The write operation transfers the content of a data register to a memory word M selected by the address. Assume that the input data are in register R_1 and the address is in AR, the write operation is as follows:

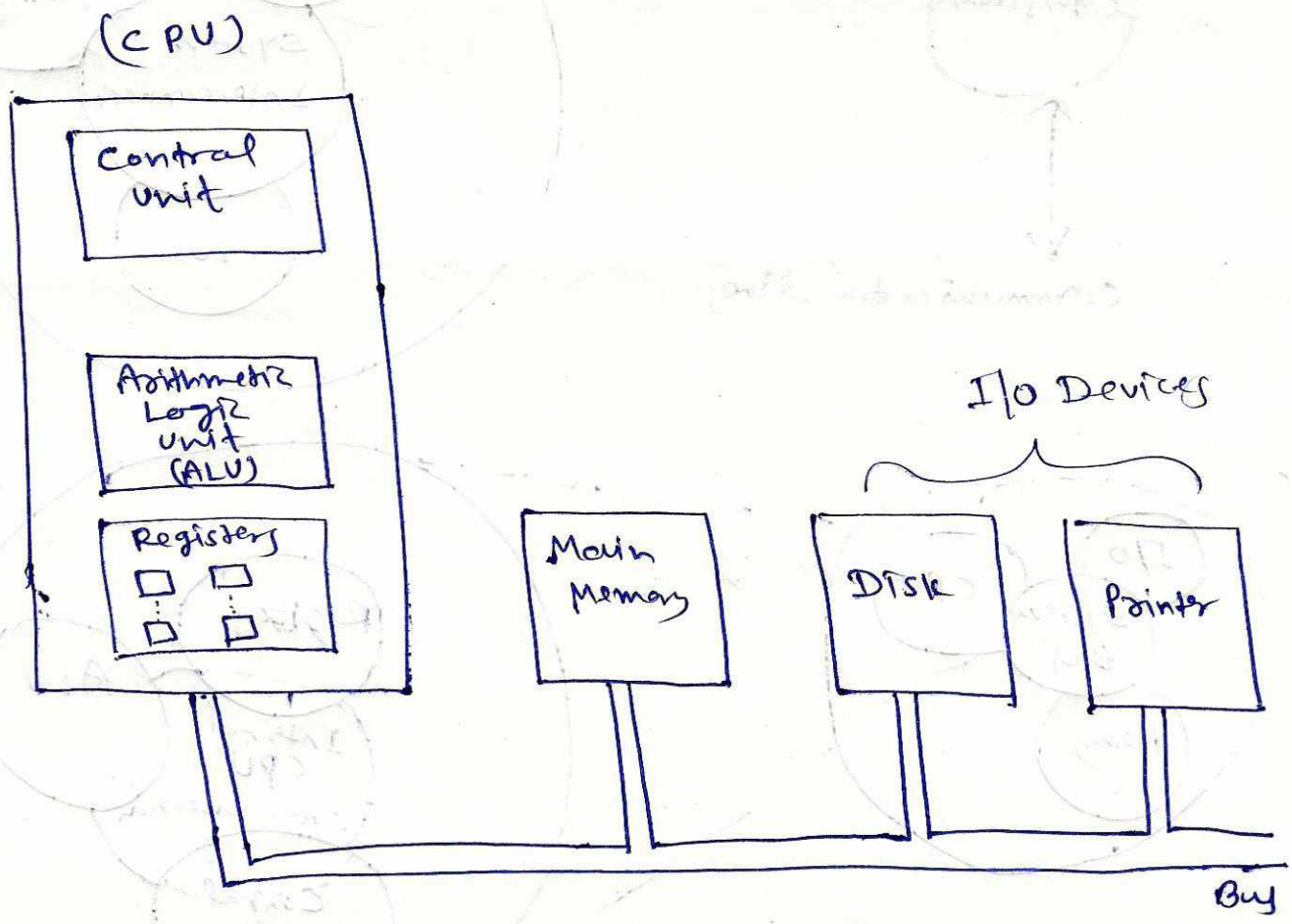
Write: $M[AR] \leftarrow R_1$

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

Processor (CPU) Organization

A processor must have three functional unit to be what we call a computer: a unit that perform arithmet and logic operation on data (ALU); a unit which remember the data (memory); and a unit which sequence the operation by ALU (control unit).

"Processor Organization" is a term describing how those three elements are implemented and how they interconnect to accomplish their task.



CPU organization

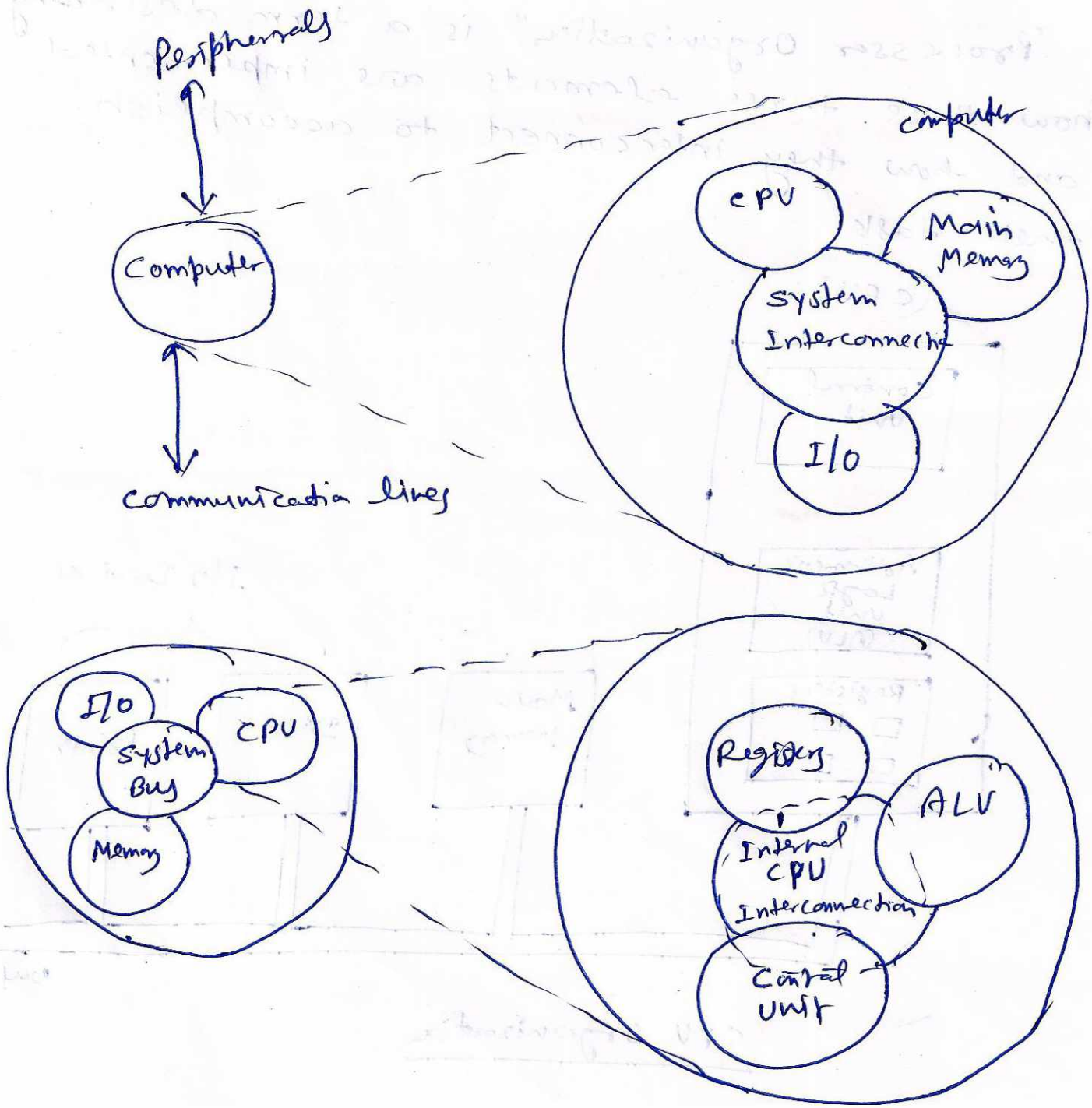
The major structural component of CPU are:

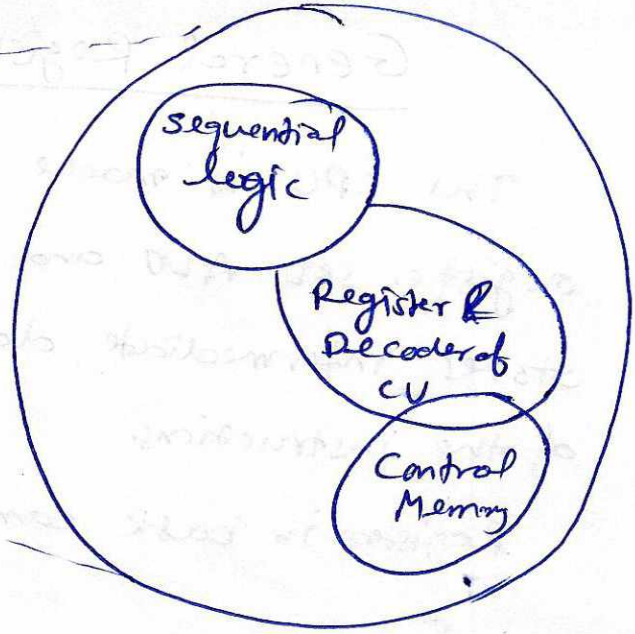
(i) Control Unit (CU) : control the operation of the CPU and hence the computer.

(ii) ALU : Performs computers data processing functions.

(iii) Register : Provide storage internal to the CPU.

(iv) CPU Interconnection : communication among the control unit, ALU and Registers.



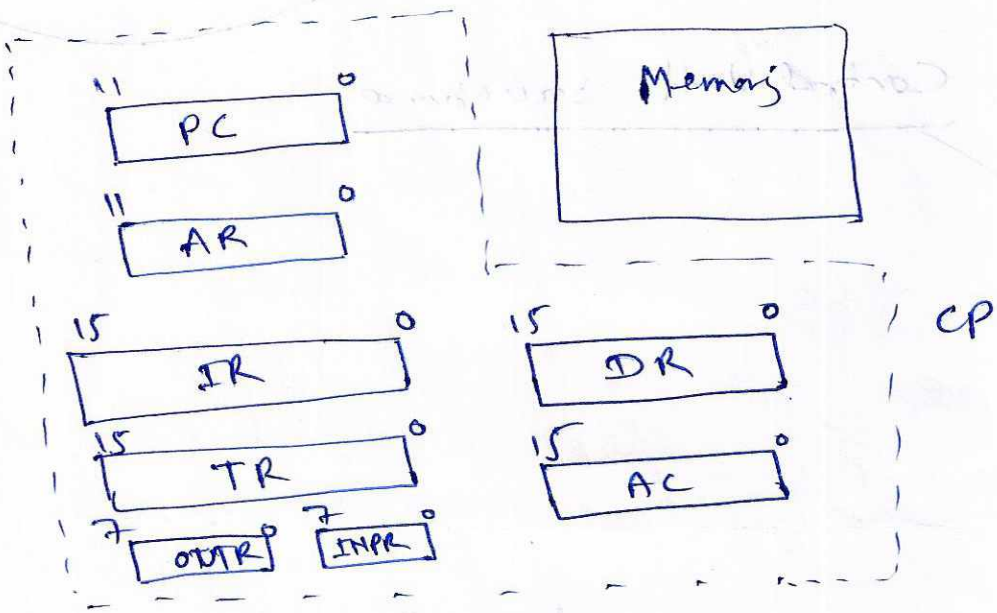


Control Unit Structure

General Register Organization

The CPU is made up of three major part: register set, ALU and control unit. The register set stores intermediate data used during the execution of the instructions.

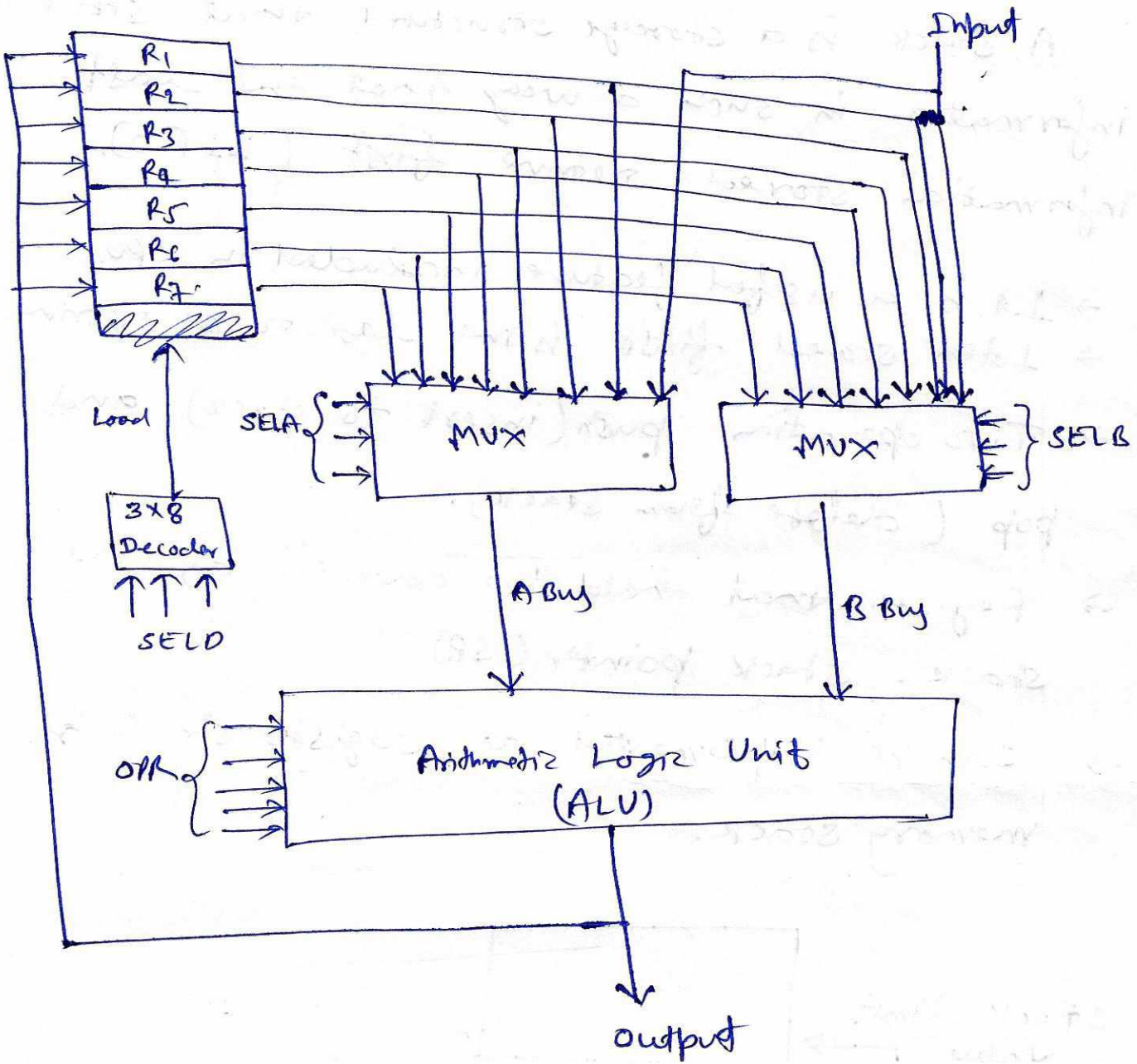
Registers in basic computer



List of General Registers :-

Reg symbol	No. of bit	Register	Function
DR	16	Data Reg	Hold operand
AR	12	Address Reg	Hold address for memory
AC	16	Accumulator	Processor register
IR	16	Instruction Reg	Hold instruction code
PC	12	Program counter	Hold addr of instruction
TR	16	Temp Reg	Hold Temp data
INPR	8	Input Reg	Hold input char
OUTR	8	Output Reg	Hold output char

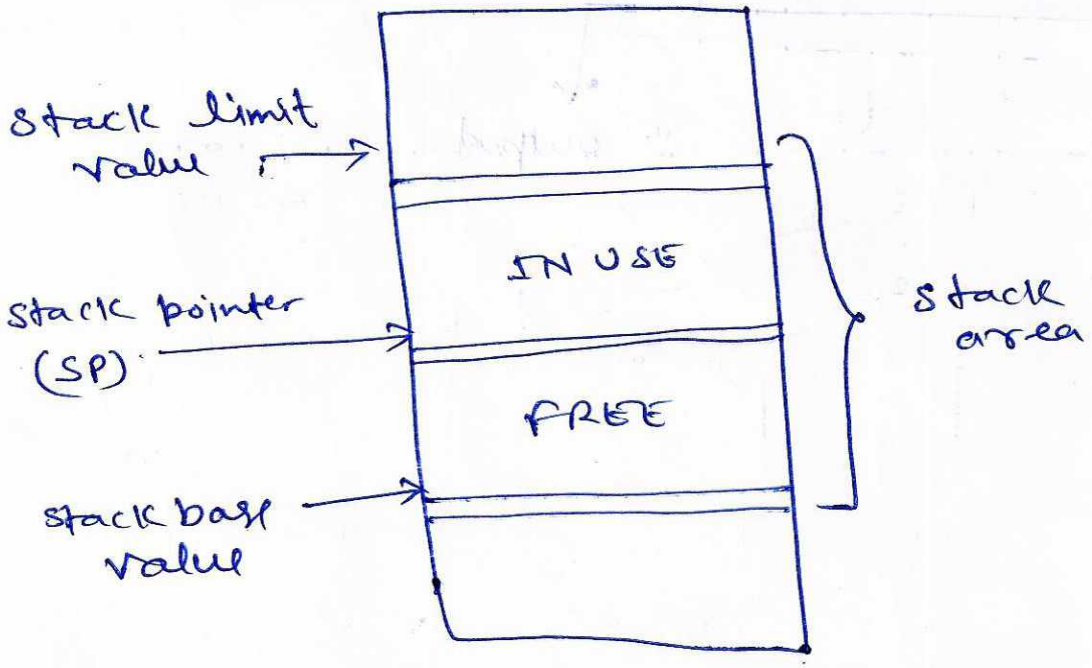
A bus organization for 7 CPU register are shown.



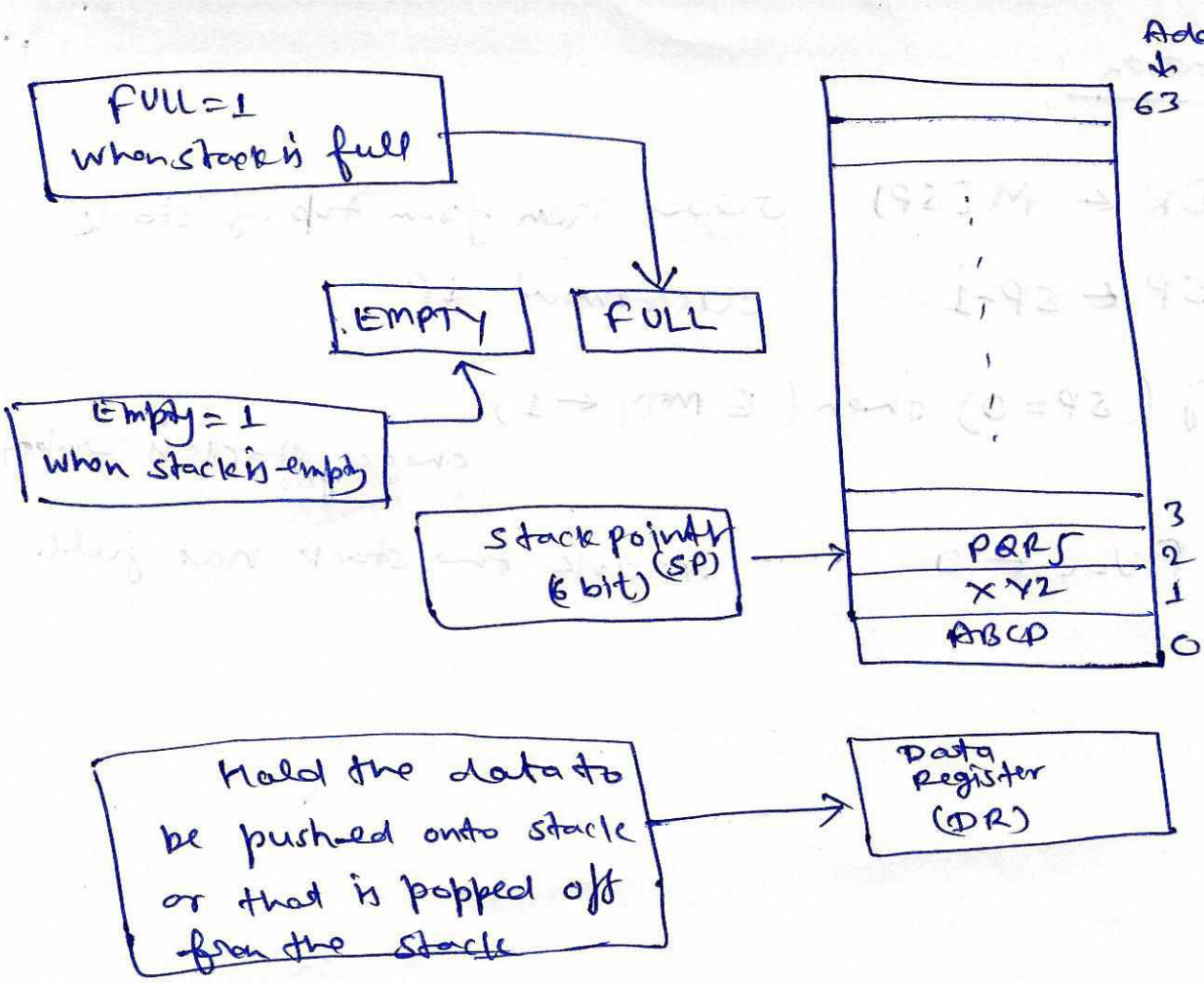
Stack Organization

A stack is a storage structure that store information in such a way that the last information stored retrive first (LIFO).

→ It is a useful feature included in CPU.
→ Item stored first is the last to be retrive.
→ Two operation push (insert to stack) and pop (delete from stack).
→ Register that hold the address of the stack - stack pointer (SP).
→ Can be implemented as register stack or memory stack.



Stack Organization



Block Diagram of 64 word stack

PUSH Operation :-

SP ← SP+1 increment stack pointer

M(SP) ← DR write item on top of stack

If (SP = 0) then (FULL ← 1) check if stack is full

EMTY ← 0 mark the stack not empty.

POP operation :-

DR ← M[SP) read item from top of stack

SP ← SP-1 decrement SP

If (SP=0) then (EMPTY ← 1) check if stack is empty

FULL ← 0 mark the stack not full.

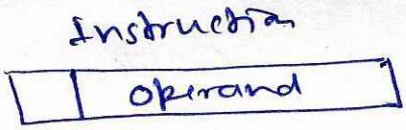
ADDRESSING MODE

The different kind of ways the programmer can refer to data stored in memory is known as addressing mode.

Most common addressing modes are:

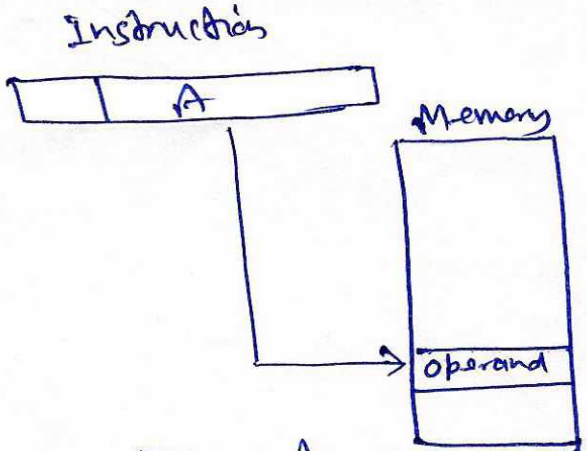
(i) Immediate
(ii) Direct
(iii) Indirect
(iv) Register
(v) Register Indirect
(vi) Displacement
(vii) stack

(i)



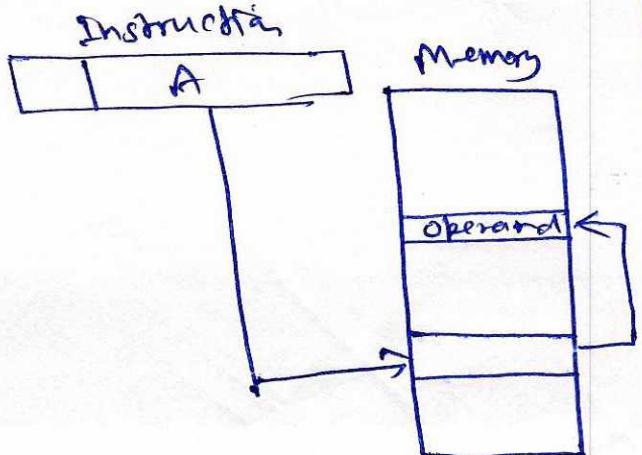
Immediate

(ii)



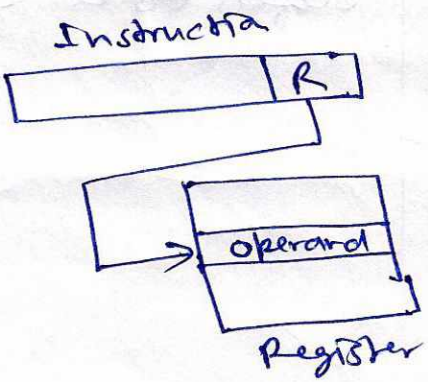
Direct

(iii)



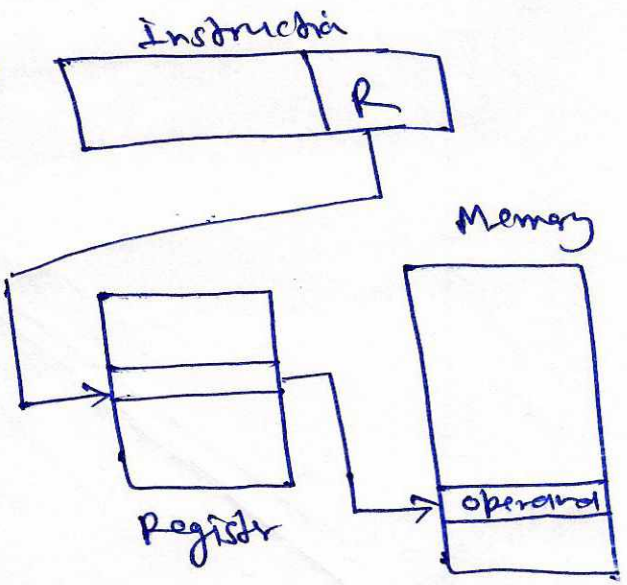
Indirect

(iv)



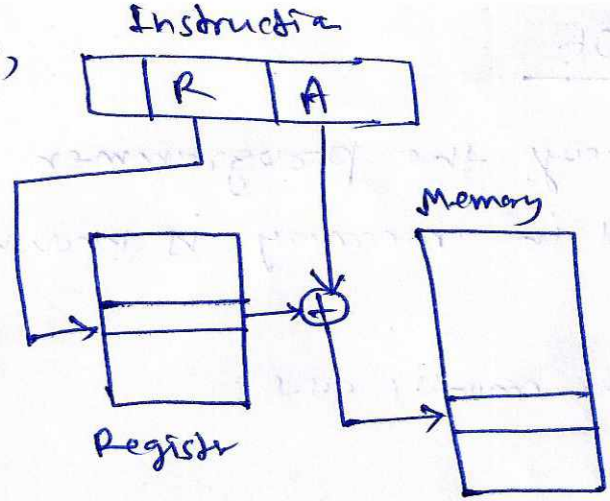
Register

(v)



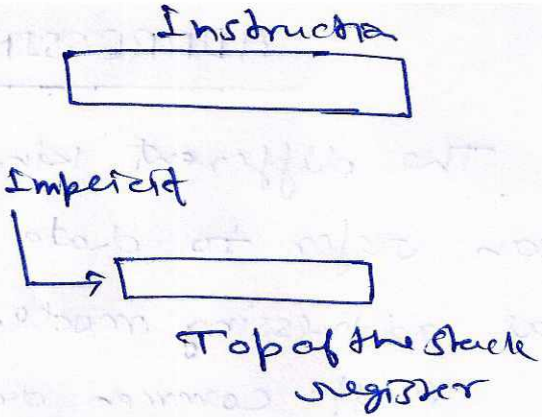
Register Indirect

(vi)



Displacement

(vii)



Stack

* A → content of an address field in the instructio.
** R → content of an address field in the instructio that refers to a register.