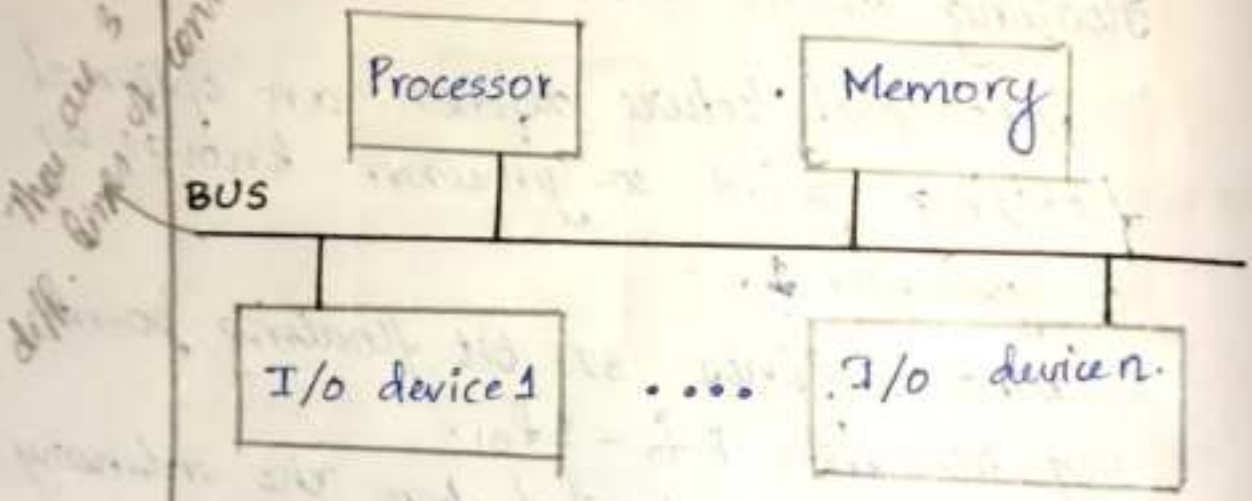
MODULE V

Accessing I/O Devices:

This includes several blocks such as processor, memory and a no: of I/O devices.



Processor: CPU.

I/O devices:- mouse, keyboard, printers --etc.

Different blocks are interconnected using special structures called buses to enable information exchange b/w different structures.

Interface / interaction b/w devices - processor memory and I/O devices is established using bus.

Buses are of three types: Address bus.
Data bus.
Control bus.

* Each of input-output device are assigned a unique set of address.

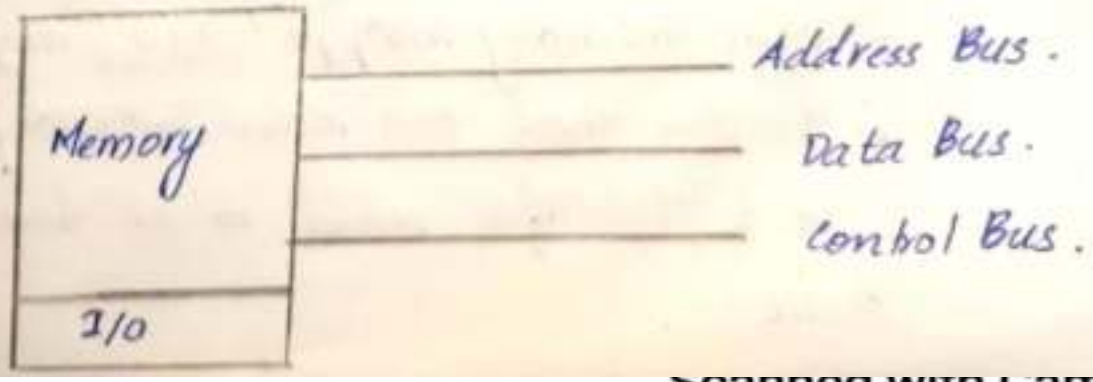
I/O Mapping

There are two methods to map I/O devices.

(i) Memory mapped I/O
(ii) Isolated I/O.

Memory mapped I/O

→ All the I/O device, processor, memory share the same address space.
→ I/O looks just like memory read/write.
→ No special commands for I/O
→ Large selection of memory access commands are available.



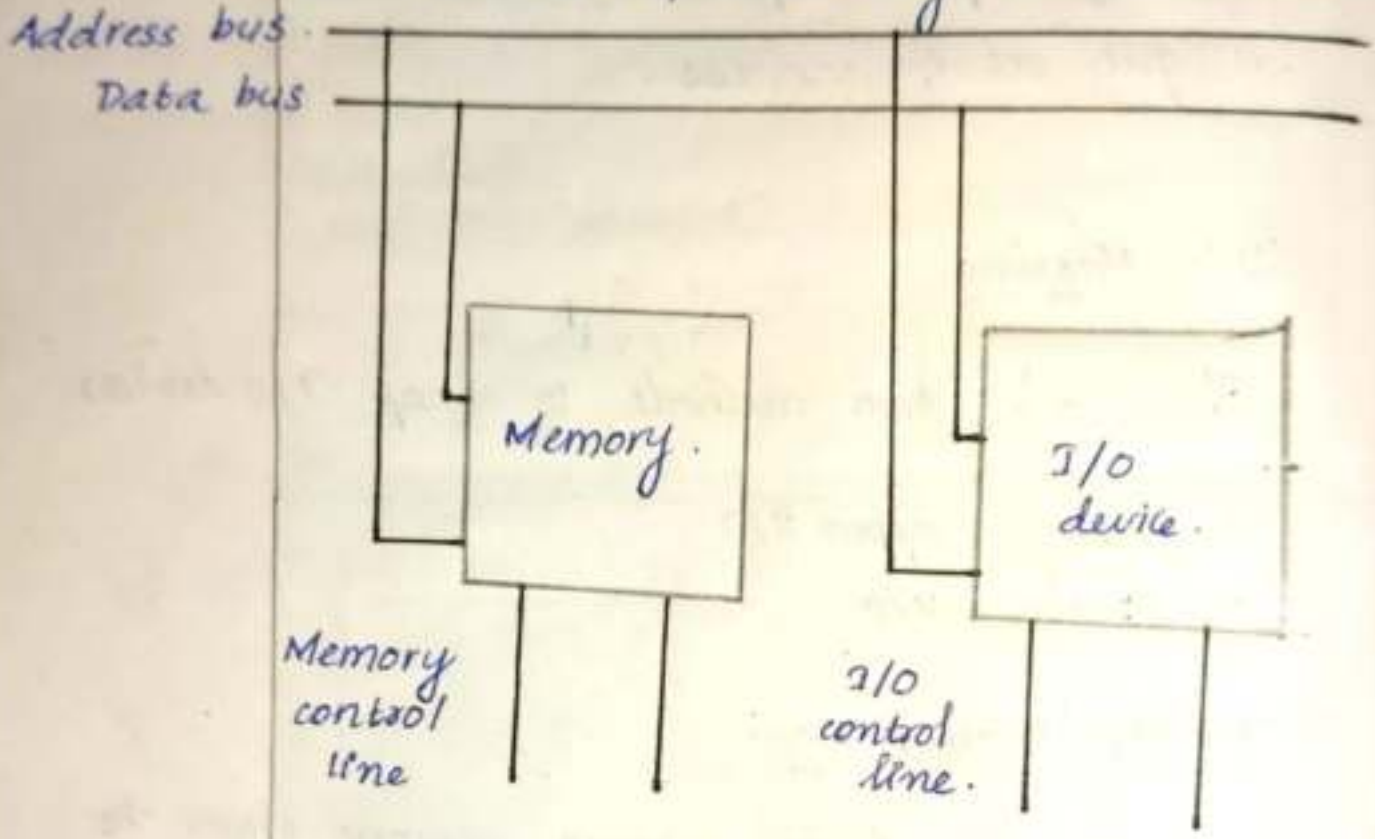
Isolated I/O

→ Seperate Address spaces.

→ Need I/O or memory select lines

→ Special commands are required for I/O device

→ Limited set of memory access commands.



MEMORY MAPPED I/O.

→ When I/O devices and memory share the same address space, the arrangement is known as memory mapped I/O.

→ With memory mapped I/O, any machine instruction that can accss memory can be used to transfer data to or from an I/O device.

Most computers make use of memory mapped I/O

→ Some processors like 8085 micro-processor uses seperate IN and OUT instructions to perform I/O transfers.

Therefore when building a computer based on ↔ these processors, the designer has the option for connecting I/O devices to use the special I/O address space or simply incorporating them as part of memory address space. i.e that can be made as isolated I/O or memory mapped I/O.

⇒ I/O Interface for an Input device:

The address decoder, the data and the status registers and the control circuitary are required to co-ordinate the I/O transfers ~~com~~ that constitute the device's Interface circuit.

Fig shows the interface in input ~~def~~ device.

Input device & processor has to communicate with each other. for that there are 3 buses:

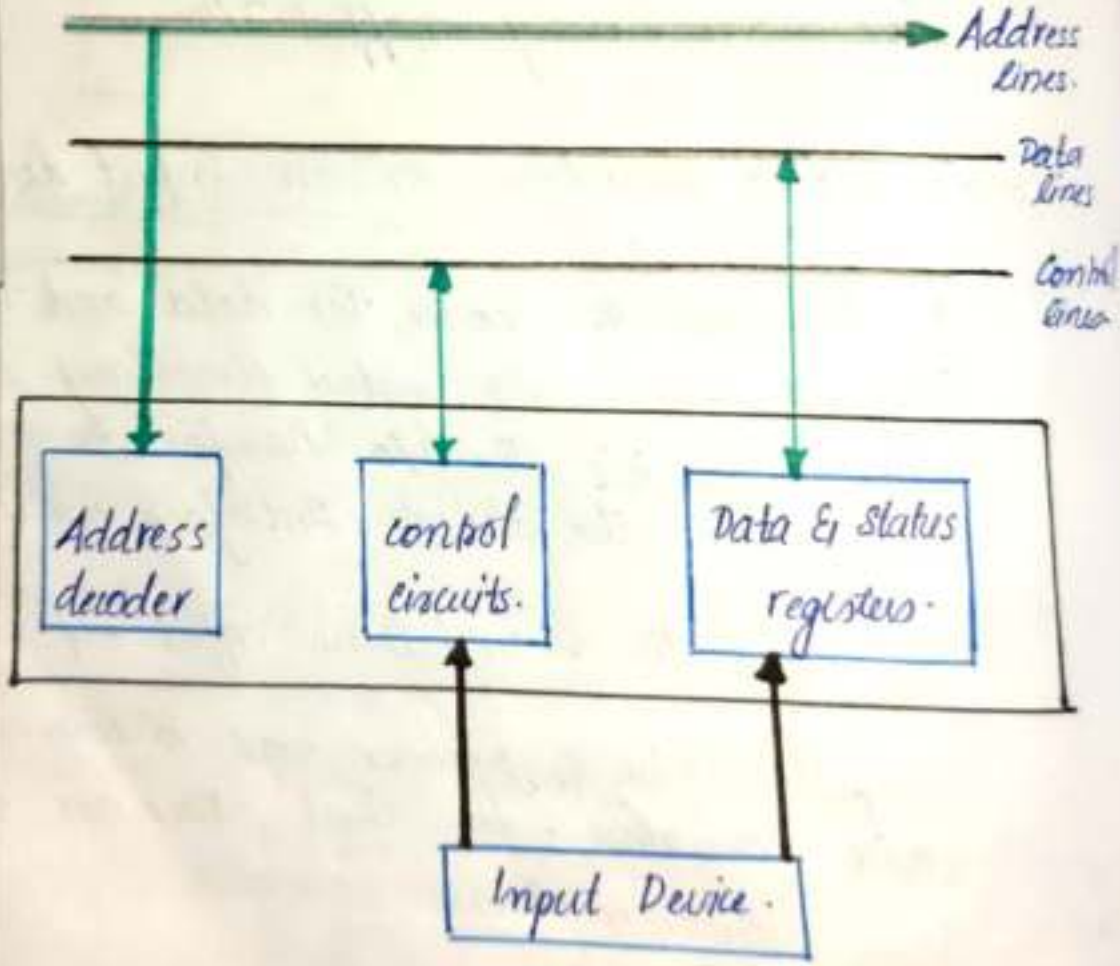
Address bus.
data bus.
control bus.

Each and every I/O devices has different unique specified address.

The processor identifies the device using the address.

To decode address we have address decoder.

The data inputted through the keyboard into the processor. and process can be acknowledged by the status register.



I/O interface can be done by using three techniques:

(i) Programmed I/O method.

(ii) Interrupt driven I/O method.

(iii) Direct memory access (DMA) control method.

Programmed I/O Method:

I/O operation b/w Input-output device is controlled using programs that is written in a process.

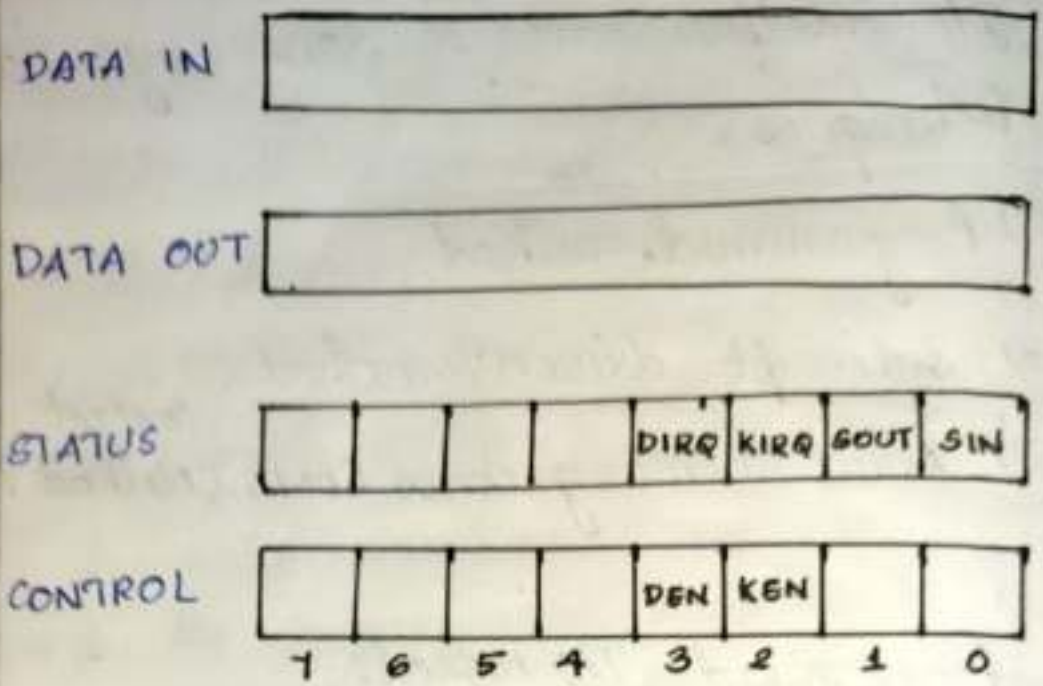
If we involve an i/p device & o/p device; then i/p device: data goes in to the ~~data~~ process

o/p device: data is taken out of the process.

two 8-bit registers are there data-in & data-out are there for this process.

Interfacing technique using programmed I/O method we make use of various registers like DATA-IN, DATA-OUT STATUS & CONTROL.

KIRQ & DIRQ in status register are used in conjunction with interrupts.



Example:

That shows how Intefrace b/w I/O devic and procesor is handled ~~b/w~~ two using programs.

This shows a program that read one line from the keyboard, stores in memory buffer and gives it back to the display.

	Move	#LINE, R0	Initialise memory pointer.
WAITK.	TestBit	#0, STATUS	TEST SIN
	Branch=0	WAITK.	wait for character to be entered.
	Move.	DATAIN, R1	Read character.
WAITD	Test Bit	#1, STATUS	Test SOUT.
	Branch=0	WAITD	wait for display to become ready
	Move	R1, DATAOUT.	Send character to display
	Move.	R1, (R0)+	store character & advance pointer.
	compare.	#$0D, R1	check if carriage return

⇒ No need to study.

```
Branch ≠0    WAITK      If not get another character.
Move.        #$0A,DATAOUT   Otherwise send it to
                            display.
Call         PROCESS.   Call a subroutine to
                        process the input
                        line.
```

The above example shows a program-controlled I/O, in which the processor repeatedly checks a status flag to achieve the required synchronisation between the processor and an input or output device. We say that the processor polls the devices.

Other two commonly used mechanisms for implementing I/O operation are interrupts & direct memory access.

Interrupts:

Interrupt is something that stops the normal execution of a program.

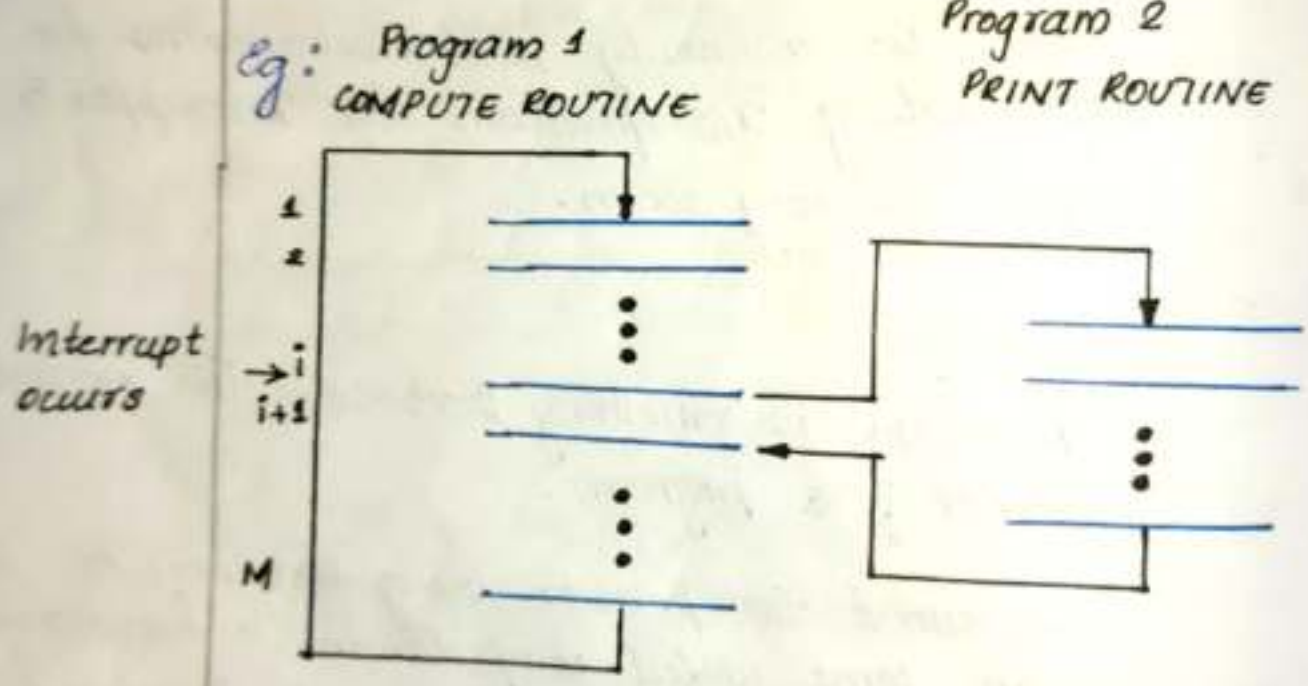
To avoid the processor being not able to perform some useful computations, a hardware signal called an interrupt to the processor can do it.

At least one of the bus control lines, called an Interrupt Request is dedicated for this purpose.

When an interrupt request is being send an interrupt - service Routine is executed.

→ The processor must communicate with the device that its request has been recognised and that the interrupt request may be removed. For this purpose an interrupt acknowledgement signal is used.

Eg:



Interrupt Service Routine & Sub-routine:

→ Treatment of interrupt-service routine is very similar to that of a subroutine.

→ A sub-routine performs function required by program from which it is called.

→ Interrupt service routine might not have anything common with the program that is being executed at the time when an interrupt request is received. The two programs may belong to different users.

→ Before an interrupt service routine is executed, any information that might be altered during execution of that routine must be saved. This information must be restored before the interrupted program is resumed.

Interrupt Latency

The condition code flags and contents of any registers that are used by interrupted program and interrupt service routine must be saved and restored.

Saving registers increases delay b/w the time and an interrupt ~~is service~~ request received and start of an execution ~~program~~ of interrupt-service routine.

This delay is called as interrupt latency.

The processor saves only the contents of the program counter and the processor status register.

Any additional information that needs to be saved must be saved by the program instruction in stack.

priority

Identification of ~~Interr~~ Interrupting device.

Polling

→ The simplest way to identify interrupting device is to have interrupt-service routine poll at all I/O devices connected to the bus.

This method is easy to implement but this takes a lot of time interrogating all the devices.

→ A device requesting an interrupt may identify itself directly to the processor.

vector interrupts

The the processor immediately start executing the corresponding interrupt-service routine.

The address of device is already known to the processor & this address is called as vectored address & this process is called vectored interrupts.

Priority

Devices are assigned certain priority hence the processor can prioritise a device.

An interrupt request from high priority device should be accepted while the processor is servicing another request from a lower priority device.

Interrupt Priority:

→ The processor's priority is encoded in a few bits of the processor status word can be changed using program instructions that can be written into the program status register.

These are privilaged instructions, which can be executed while the process is running in the supervisor mode.

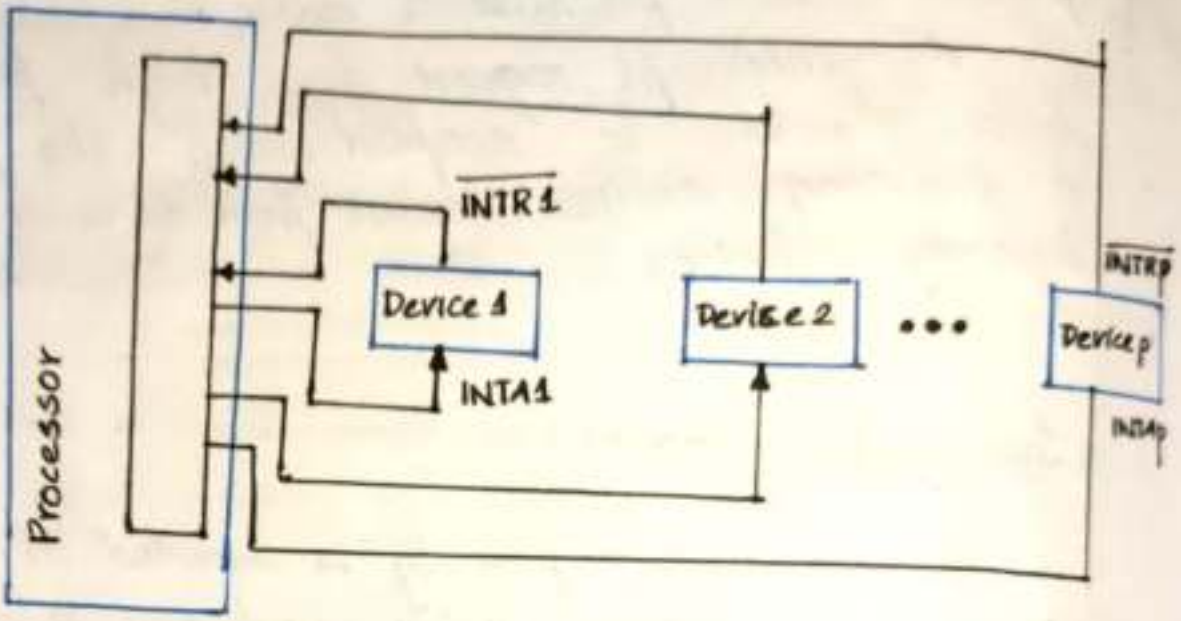
⇓

The processor is in supervisor mode only when executing operating system routines.

It switches to user mode before beginning to execute application program.

An attempt to execute a privilaged instruction while in user mode leads to a special type 1 interrupt.

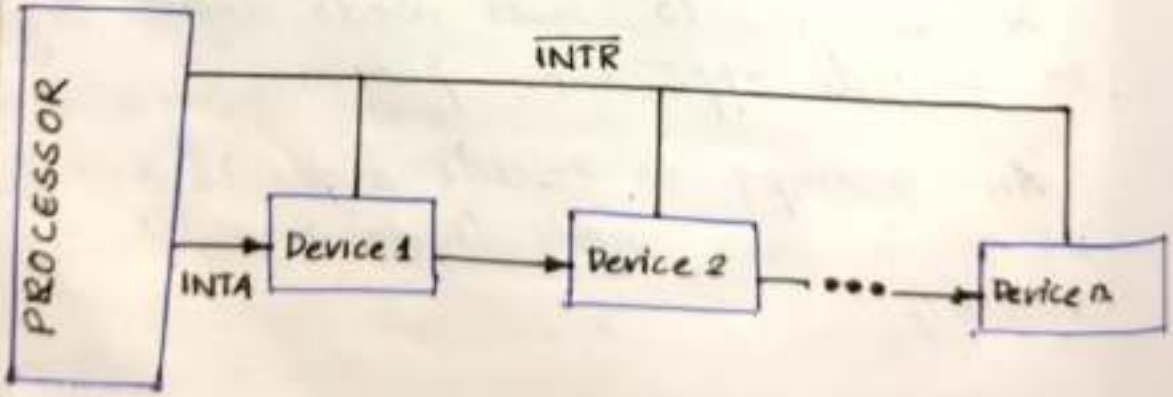
IMPLEMENTATION OF INTERRUPT PRIORITY



SIMULTANEOUS REQUESTS:

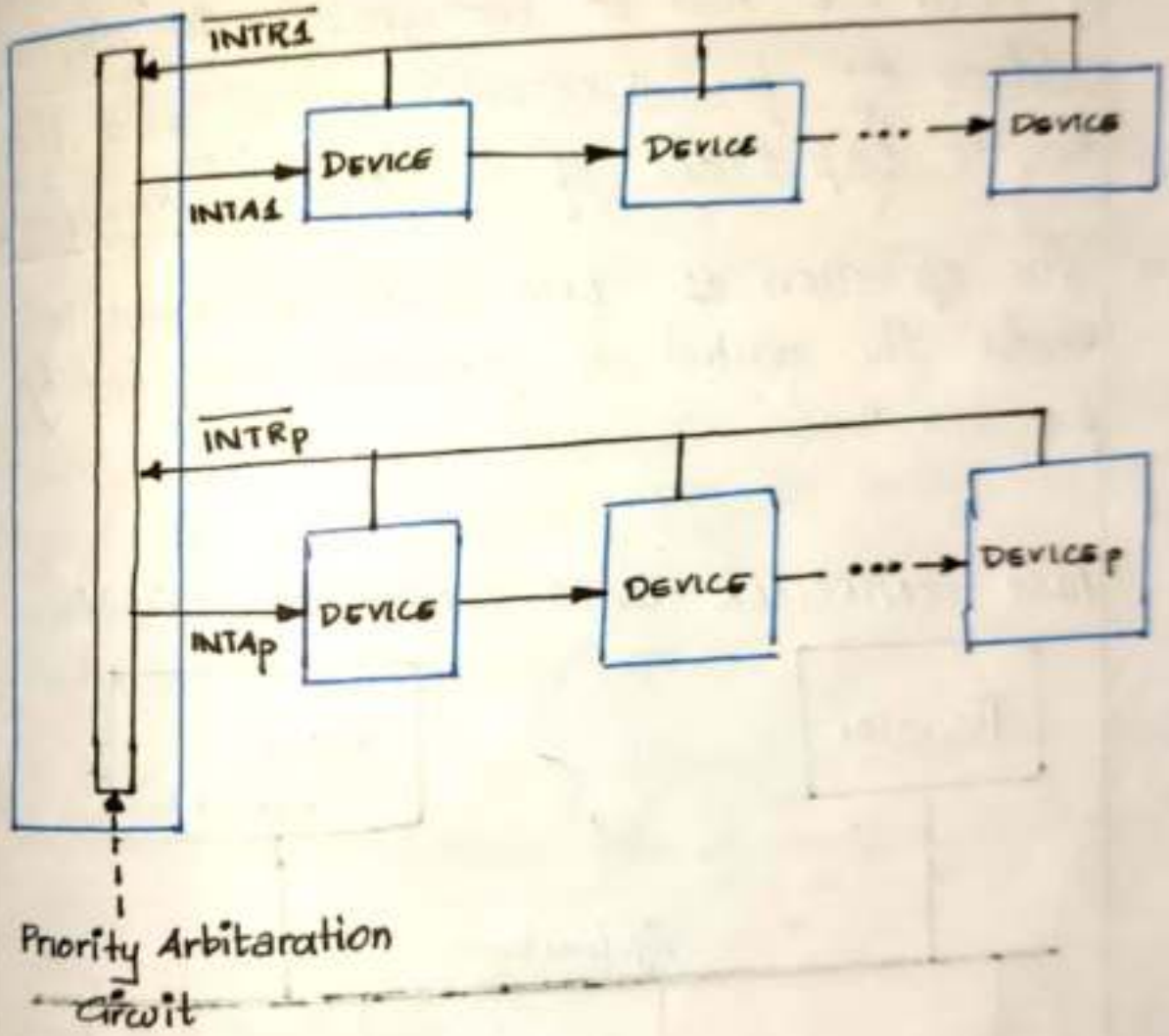
The problem of arrival of simultaneous interrupt requests from two or more devices is solved using priority.

Interrupt priority scheme with daisy chain



PRIORITY GROUP:

combination of priority interrupt scheme with daisy chain and individual interrupt request & interrupt acknowledge lines.



DIRECT MEMORY ACCESS:

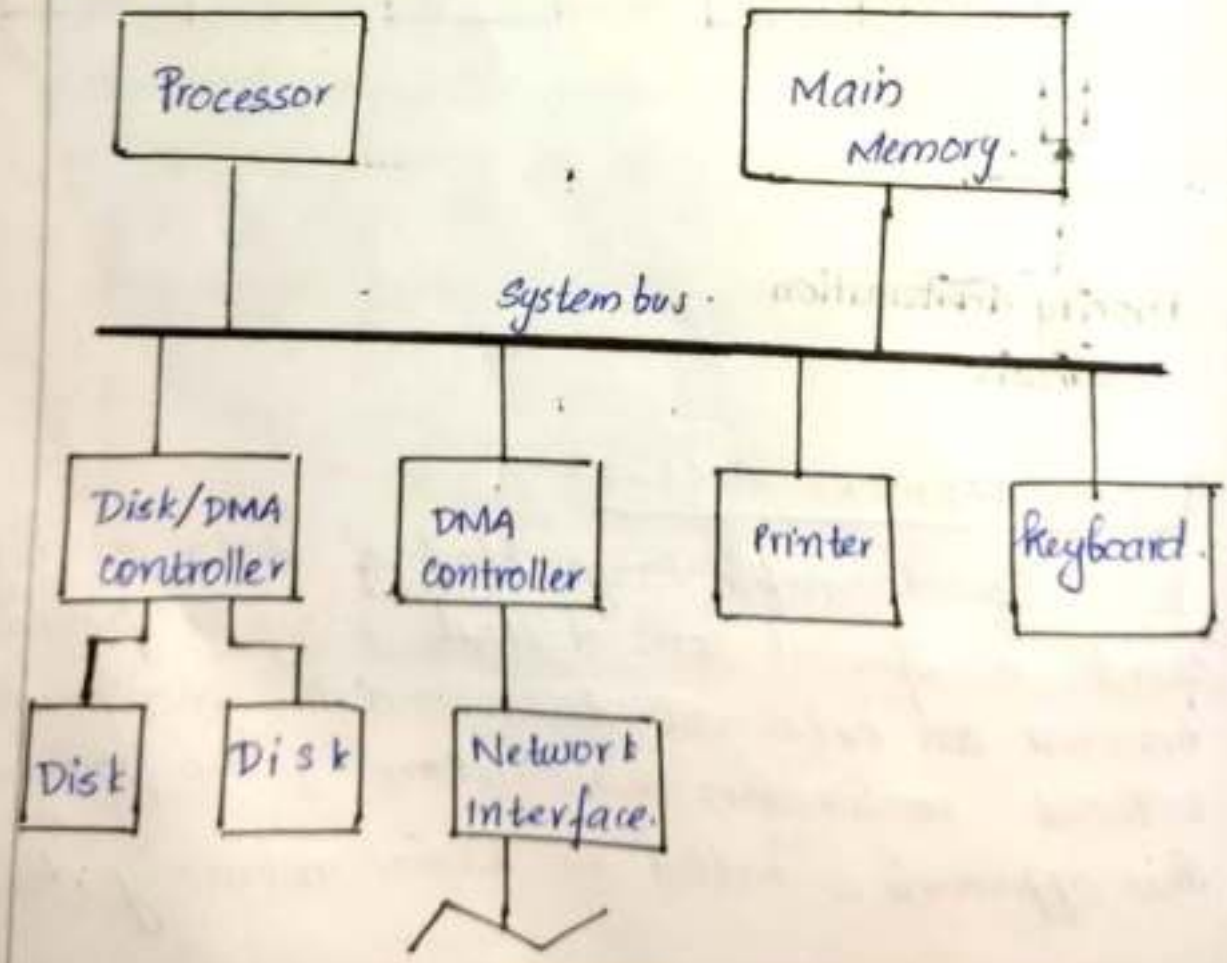
To transfer large block of data at high speed, a special control unit may be provided between an external device and the main memory without continuous intervention by the processor. This approach is called as Direct memory access.

→ DMA controller can directly handle data transfer between external device & main memory when data transfer is larg

→ The DMA controller has to transfer blocks of data. it has to increment the memory address by for successive word and it also has to keep track of number of transfers.

→ The operation of DMA control s must be under the control of program executed by the processor.

DMA CONTROLLER IN-A COMPUTER SYSTEM.



Memory Access Priority :-

Memory accesses by the processor and the DMA are interwoven. Request by DMA devices for using bus are always given higher priority than processor requests.

Top priority is provided for high-speed peripherals such as a disk, a high speed network interface ...etc.

Since the processor originates most memory access cycles, the DMA controller can be said to 'steal' memory cycles from processor. This interweaving technique is usually called cycle stealing.

DMA controller may transfer block of data without interruption. This is called block/burst mode.

Bus Arbitarisation :-

To avoid conflict between processor & DMA controller or two DMA controllers to use the bus at same time for accessing memory a arbitration procedure on bus is used.

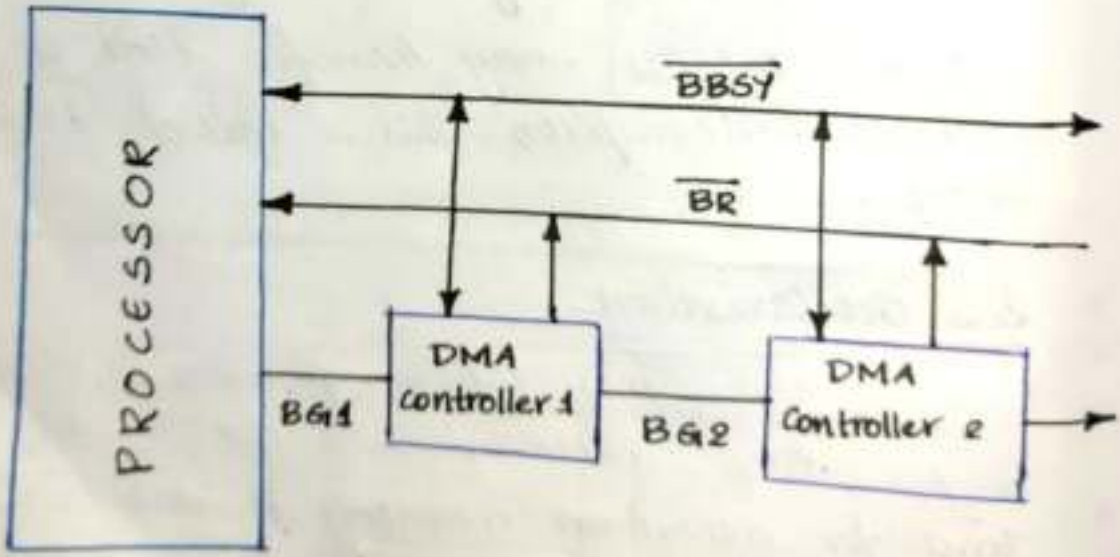
Bcuss of acarning bus - bus arbitration

Device that is allowed to initiate data transfer on bus at any given time is called bus master.

when current master becomes free of the bust, another device can acan the status.

Bus arbitration is the proces by which the next device to become the bus master take into account the needs of various devices by establishing a priority systems for gaining acan to bus.

CENTRACISED ARBITARISATION:



active low signal { BBSY - Bus Busy
BR - Bus request
Active high signal ← BG - Bus grant

Buses:

Bus protocol: Set of rules that govern behaviour of various devices connected to bus.

In synchronous bus: All devices derive timing info from common clock line.

Equal spaced pulses define equal time intervals on this line.

⇒ Each of these intervals constitute a bus cycle during which one data transfer can take place.

Interface Circuits: Eg: of input interface with processor

Keyboard to processor connection:

Whenever a valid key is pressed, valid signal changes from 0 to 1 causing ASCII code of key is loaded into DATAIN & SIN is set to 1.

{S-IN: Serial IN: status flag}

whenever 'datain' is set to 1, SIN is set to 1.

The status flag SIN is cleared to 0 when processor reads the contents of the DATAIN Register

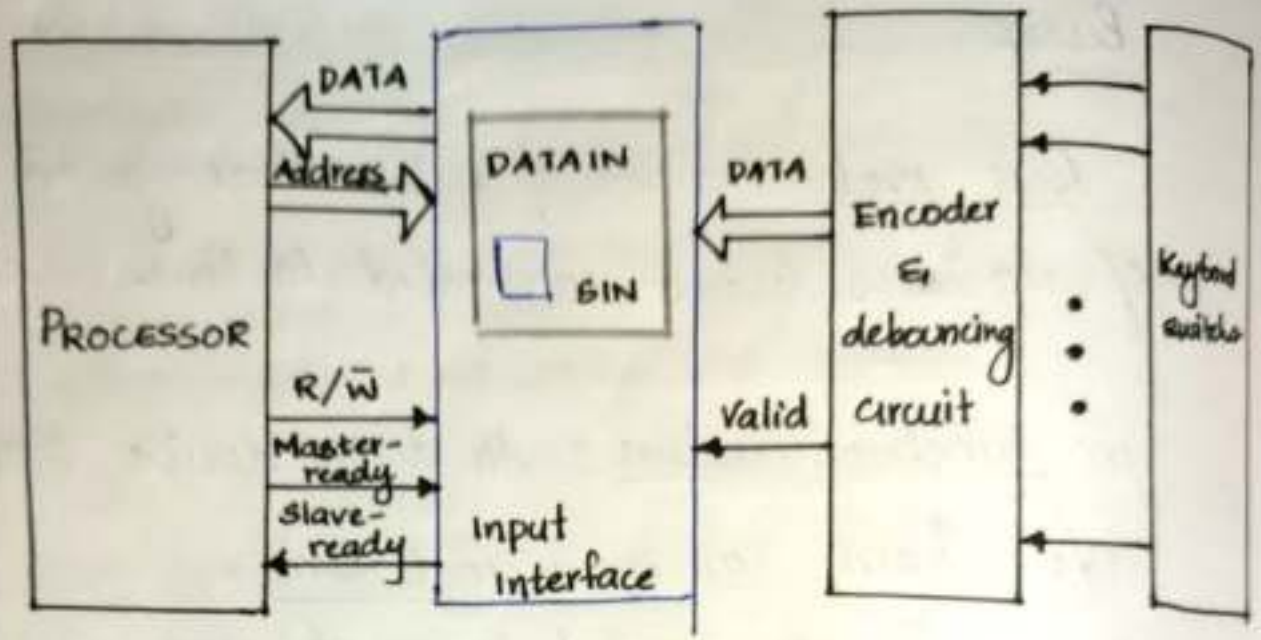


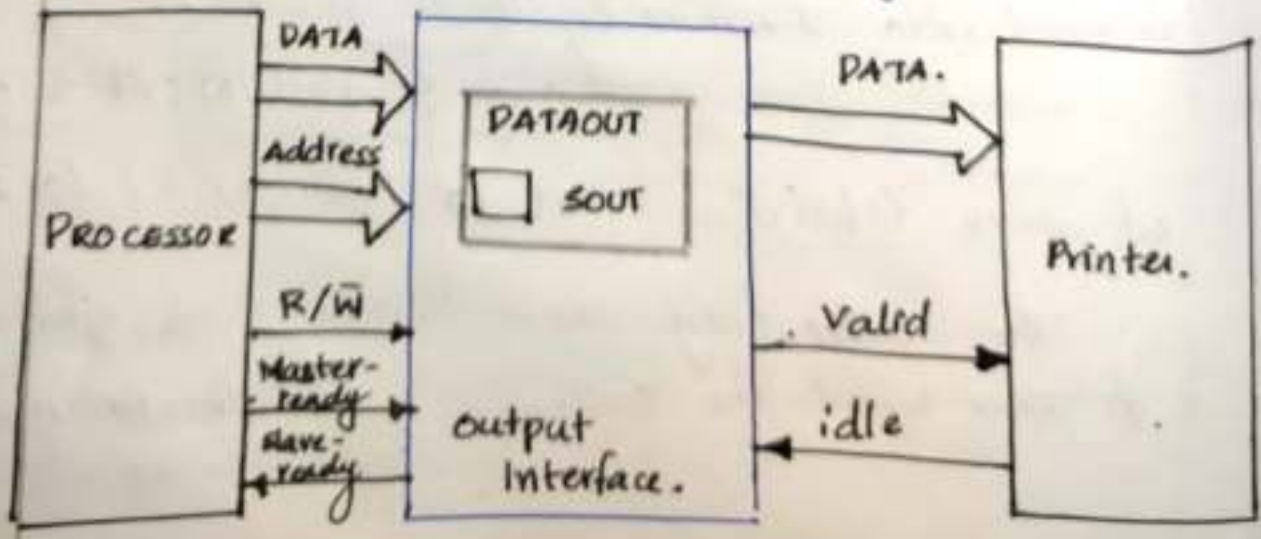
Fig: Eg: of Interface Circuits

⇒ Example of Output Interface with Processor

The SOUT flag is set to 1 when the printer is ready to accept another character.

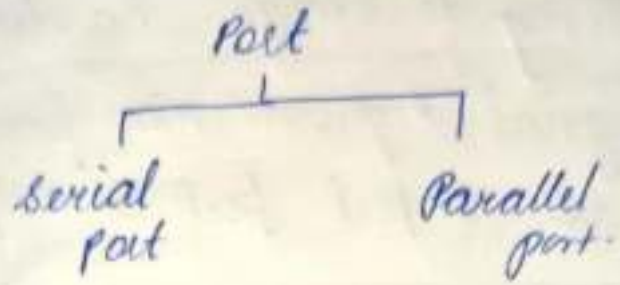
SOUT flag is set to 0 when new character is loaded into the DATAOUT by processor.

when printer becomes ready to accept a character it asserts an idle signal.



SERIAL PORT

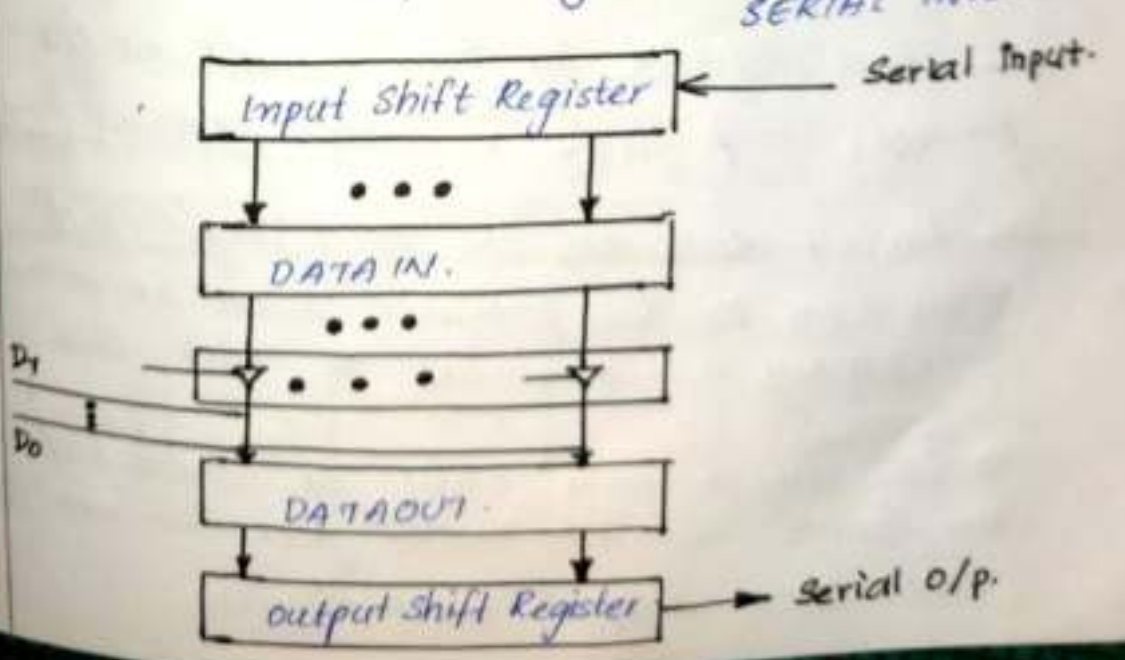
Port: Inter-connection between processor to input & output devices.



⇒ data transmitted one bit at a time.

⇒ capable of communicating in bit serial fashion on device side & bit-parallel ~~fasion~~ fashion on bus side.

⇒ Transformation b/w the parallel & serial format ~~pair~~ is acheived using shift registers that have parallel accen capability.



PARALLEL PORT:

→ used to send or receive data having group of eight bits simultaneously

→ According to usage, hardware and control signal requirements parallel ports are classified as input port & output port.

Input port: Receive data
output port: send data.

→ Parallel ports are easy to program & faster as compared to serial port.

Disadvantage: More no: of transmission line.

→ Not used for long distance communication

STANDARD I/O ~~INTO~~ INTERFACES:

→ Bus defined by the signals on the processor chip itself is called processor bus.

→ Devices that require very high speed connection to the processor, such as main memory is connected to this bus.

→ Few devices can only be connected to this bus hence mother board provides another bus that can support more devices.

Bridge: Two buses interconnected by a circuit

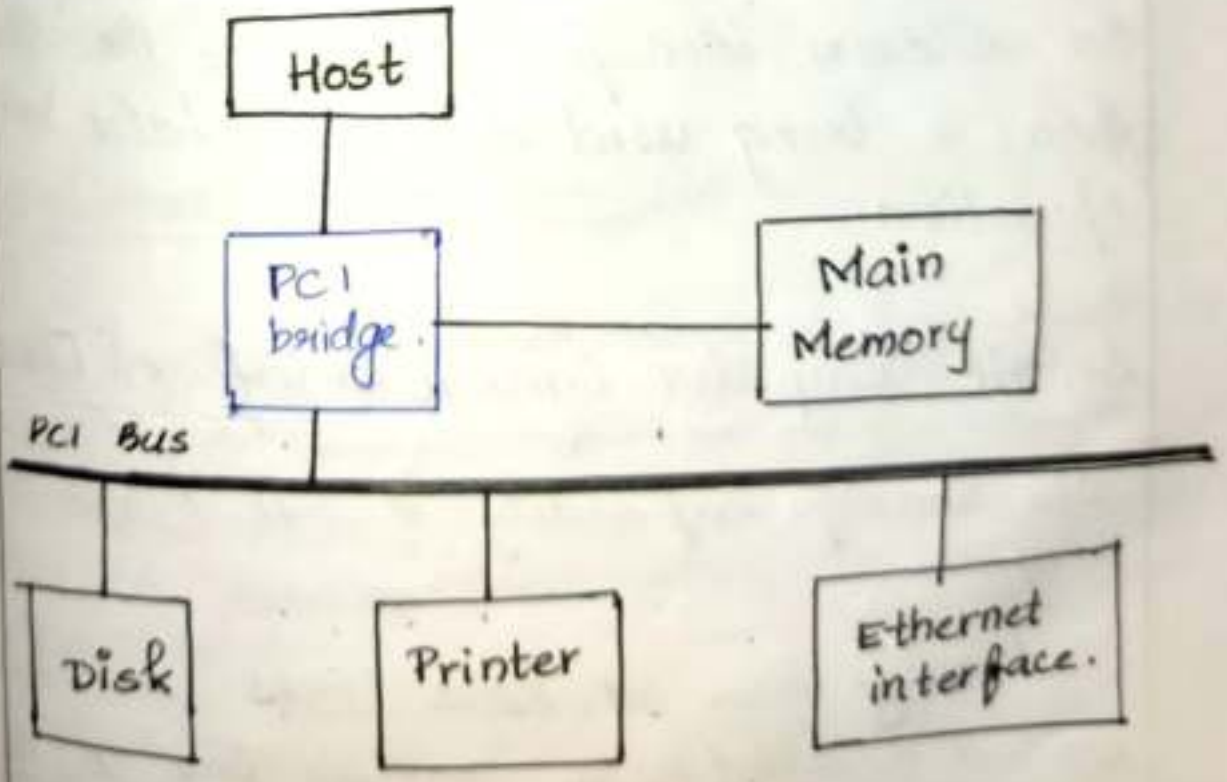
A bridge translates the signals & protocols of one bus into those of the other.

The structure of processor bus is closely tied to the architecture of the processor.

The expansion bus uses standardised signaling structure since it is not subjected to such limitations.

Eg: of standard I/O Interface: PCI.

⇒ PCI: Peripheral component Interconnect Bus



→ It supports functions found on processor bus bit is a standardised format that is independant of any particular processor.

→ Devices that are connected to the PCI bus appear to the processor as if they were connected directly to processor bus. They are assigned addresses in the memory addres space of the processor.

→ This bus supports three independant address spaces: memory, I/O and configuration.

The I/O address space is intended for use with processors, such Pentium, that have a seperate I/O address space.

A 4-bit command that accompanies the address identifies which of the three spaces is being used in a given data transfer operation.

Small Computer System Interface: [SCSI].

The transfer capability of SCSI doubles almost every year.

It may have 8 data lines, in which case it is called as a narrow bus & transfers

data one byte at a time.

A wide SCSI bus has 16 data lines & transfers data 16 bits at a time

Devices connected to SCSI bus are not part of address space of processor in the same way as devices are connected to the processor bus.

The SCSI bus is connected to the processor bus through a SCSI controller.

This controller uses DMA to transfer data packets from main memory to the device & vice-versa.

A packet may contain block of data, commands from processor to device or status information of device.

Universal Serial Bus [USB]

⇒ Designed to meet several key objectives

+ Provide simple, low cost and easy to use interconnection system that overcomes difficulties due to limited no: of I/O ports available on computer.

— Accomodate wide range of data transfer characteristics for I/O devices, including

telephone and internet connections.

→ use convinience is enhanced through a "Plug and play" mode of operation

USB Structure:

→ A serial bus structure is choosen as it satisfies low cost and flexibility requirements.

→ clock and data information are encoded together and transmitted as a single signal.
∴ There are no limitations on clock frequency or distance arising from data skew.

→ The USB has tree structure to accomodate large no: of devices that can be added or removed at any time.

Node of tree : • Represents device.
• called hub
• Intermediate control point b/w the host & I/O device.

Root of tree : • called root hub
• connects entire tree to host computer.

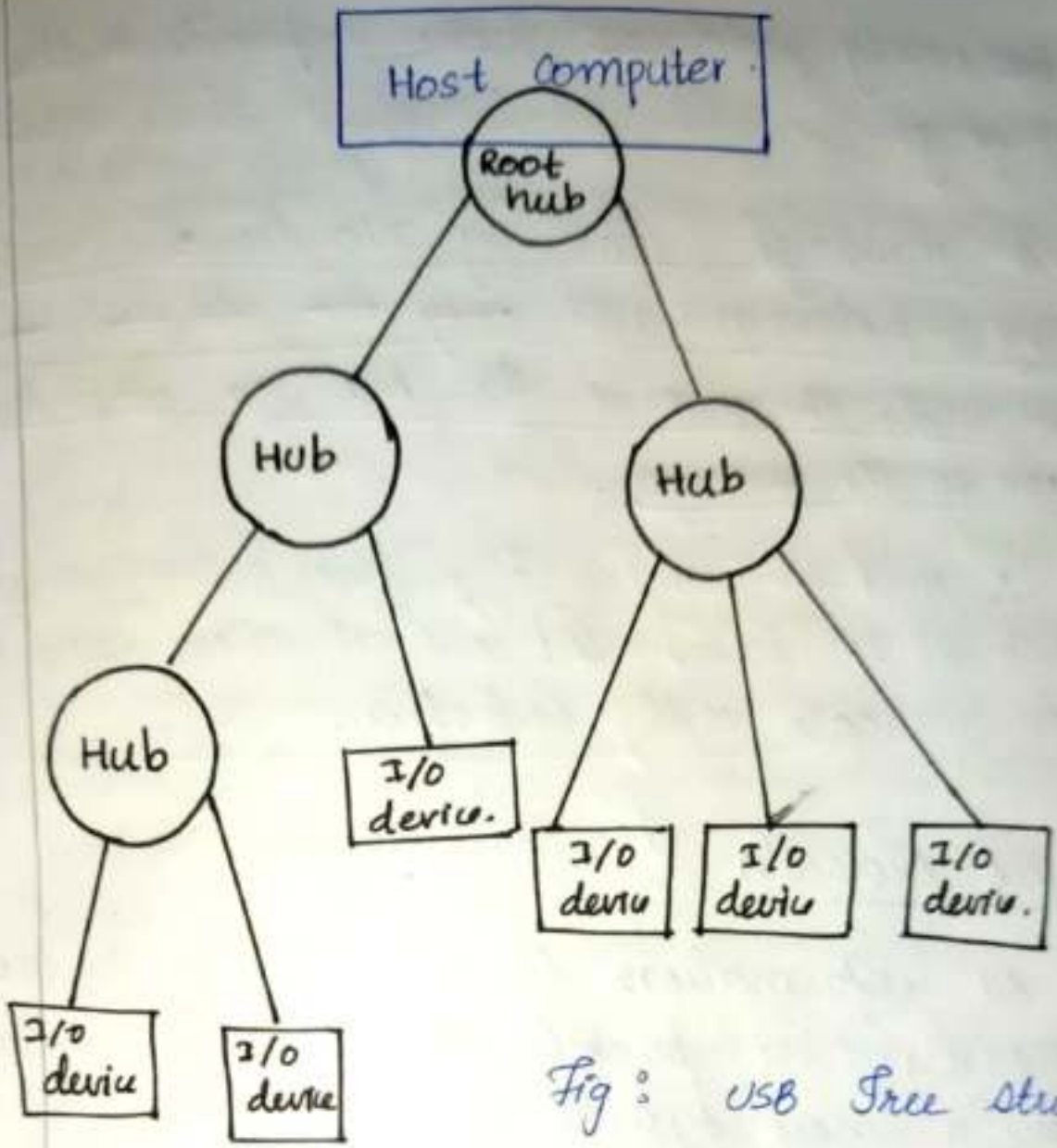


Fig: USB Tree structure.

- This uses simple pain-to-point serial links to connect many devices.
- Each hub has no: of ports were devices may be connected including other hubs.
- In normal operation, hub~~copies~~ copies the menage that it receive from upstream connection to all its downstream ports.

⇒ Menage sent ~~by broadcast is sent by~~ host computer is broadcast to all I/o device.

but only address device responds to the message.

⇒ A message sent by I/O device is sent only upstream and only the addressed device towards the root of the tree, no other devices can see the message.

∴ ~~Host~~ USB enables host to communicate with all I/O devices but will not allow devices to communicate with eachother.

USB Protocols

All informations transferred over the USB is organised as packets, where a packet consists of one or more bytes of information.

Information transferred by USB.

- Control packet
  - Perform tasks such as addressing device to initiate data transfer, acknowledging data has been received correctly, indicating error
- Data Packet
  - carry info: delivered to device.
  - Eg: I/P & O/P data are transferred inside data packets.

~~Memomory~~

Memory System

Memory Hierarchy :-

capacity, cost and speed of different types of memories play vital role in designing memory ~~of system for a~~ system for computer.

There is a tradeoff b/w three characteristics: cost, capacity and access time. All the three characteristics cannot be accomodated in a memory system since:

→ capacity ↑ access ~~tim~~ time ↑

~~cost~~ cost per bit ↓

and also vice versa: i.e access time ↓ capacity ↓

cost per bit ↑.

A designer may try to ↑ capacity since this ↓ cost per bit but then access time ↑ & this ↓ the performance.

This calls for memory hierarchy.

→ Memory hierarchy is to obtain highest possible access speed while ~~minimi~~ minimizing total cost of memory system.

→ To make it more economical low cost storage devices serve as a backup for storing the information that is not currently used by the CPU since all accumulated info: is not required by CPU at the same time.

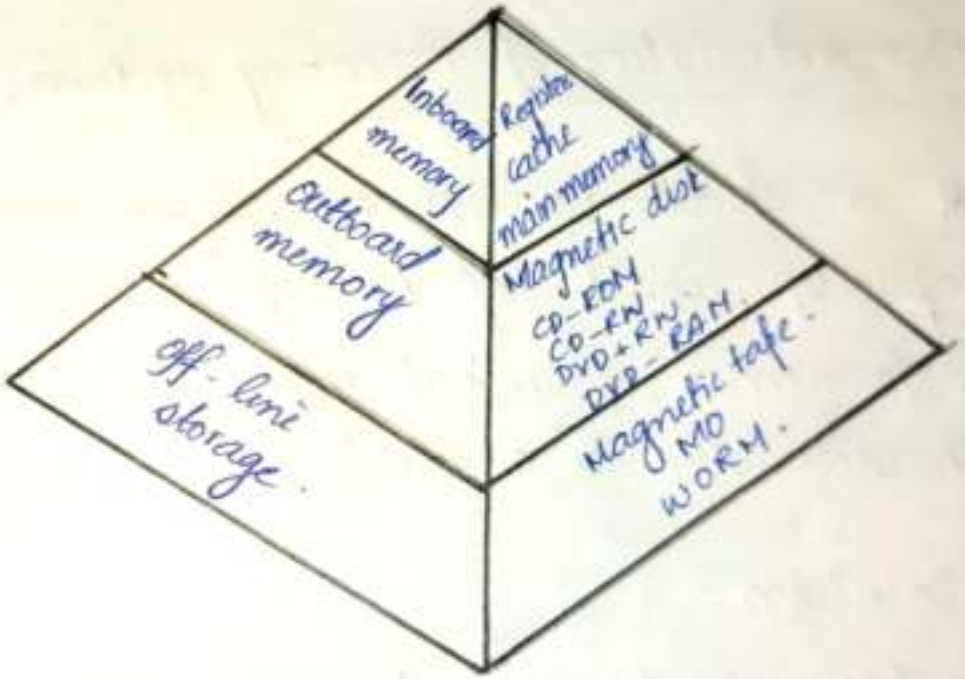
Devices providing backup storage is called auxillary memory.

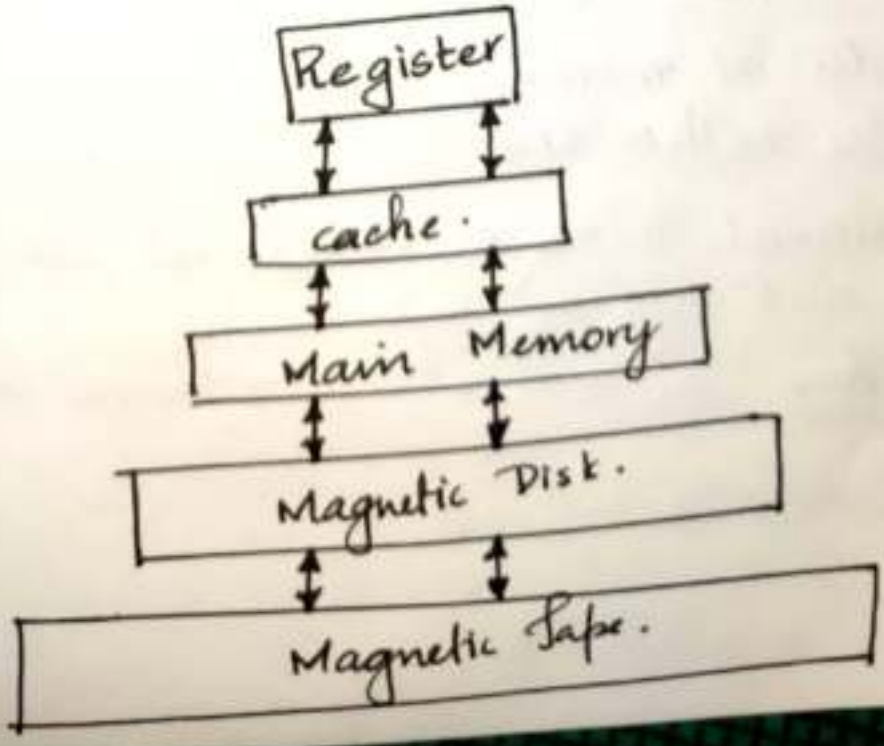
Memory hierarchy system consists of all storage devices employed in computer from the slow by high capacity auxillary memory to a relatively faster main memory; to an even smaller cache memory.

→ Main memory occupies central position. A special very high speed memory called cache is used to ↑ the speed of processing by making current programs & data available to CPU at a rapid rate.

cache - intermediate memory [It is a virtual memory]

cache is used for storing segments of programs currently being executed in the CPU & temporary data frequently needed in the present calculations.





As we go down the hierarchy

→ cost per bit ↓

→ capacity of memory ↑.

→ Access time ↑.

→ Frequency of access of memory by processor ↓

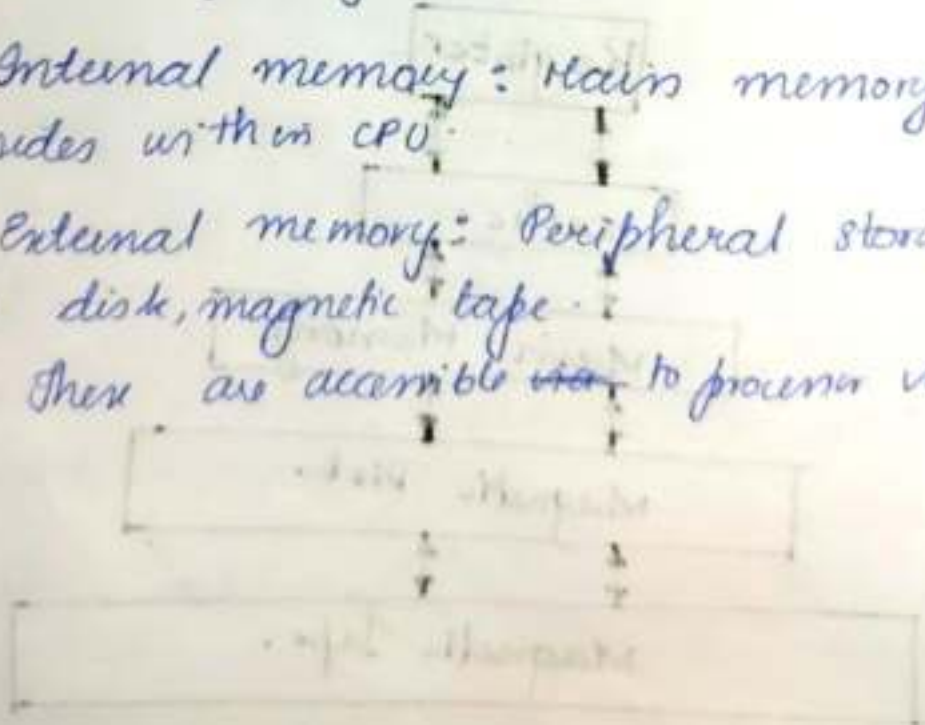
characteristics of memory system:

It is characterised based on their location, capacity, unit of transfer, Access method, performance, physical type, physical characteristics & organisation.

→ location:

- Processor memory: Memory that lie with processor. Eg: Register
- Internal memory: Main memory that resides within CPU.
- External memory: Peripheral storage devices like disk, magnetic tape.

These are accessible ~~thro~~ to processor via i/o controllers



Capacity :

Word size : Expressed in terms of words or bytes

No. of words ; common word lengths : 8, 16, 32 bit or bytes. bits.

unit of transfer

Internal : unit of transfer = no: of data lines into & out of the memory module.

External : They are transferred in block which is larger than word.

Addressable unit : . smalled locations that can be uniquely addressed
. word internally
. cluster on magnetic disk.

Access Method:

Sequential access : → In this access this must start from beginning and read through a specific linear sequence.
→ Access time depends on position of records & previous location.
Eg : Tape.

Direct Access : → Individual blocks of records have unique address based on location.

→ Access accomplished by jumping to general vicinity plus a sequential search to reach location

eg: disk.

Random Access : → Time to access a given location is independant of the sequence of prior accesses & is constant.

→ Any location can be randomly selected and directly addressed & accessed.

eg RAM.

Associative Access: → Random access type of memory that enables one to make a comparison of desired bit of locations within a word for a specified match and to do this for all words simultaneously.

eg: Cache.

Performance:

Acces time:

For random acces memory acces time is time taken for memory to perform read & write operation.

For non-random acces memory, it is time required to position read/write mechanism.

Memory cycle Time:

Total time required to store next memory acces ~~location~~ operation from previous memory acces operation.

Memory cycle Time = Acces time + Transient Time.

Time required for memory to recover before next acces = cycle time in acces + Recovery.

Transfer Rate: Rate at which data can be transfered in & out of memory unit.

For random acces $\Rightarrow R = \frac{1}{\text{cycle Time}}$.

For Non-random accem $\Rightarrow T_n = T_a + \frac{N}{R}$

T_n - av. time to read/write N bits

N: No. of bits.

R - Transfer rate in bits per second.

- Physical types :
  - Semiconductor – RAM
  - Magnetic – Disk & Tape
  - Optical – CD & DVD

- Physical characteristics
  - → ~~Dec~~ Decay - Info decays mean data lost.
  - → Volatality - Info decays when electric power is switched off.
  - → Erasable - Permission to be erase
  - → Power consumption - Amount of power consumed

Internal organisation of memory chip.

A memory chip is capable of storing 1-bit of information.

A number of memory cells are organised in the form of a matrix to form memory chip

Organisation of memory:

- → RAM organisation.
- → ROM organisation.

⇒ RAM Organisation:

- Most semi-conductor memories are packaged as chips.
- These memory chip can store information in the range from 64kB to 1MB.

There are several memory organisation of chips among which common are.

(i) 2D - organisation:

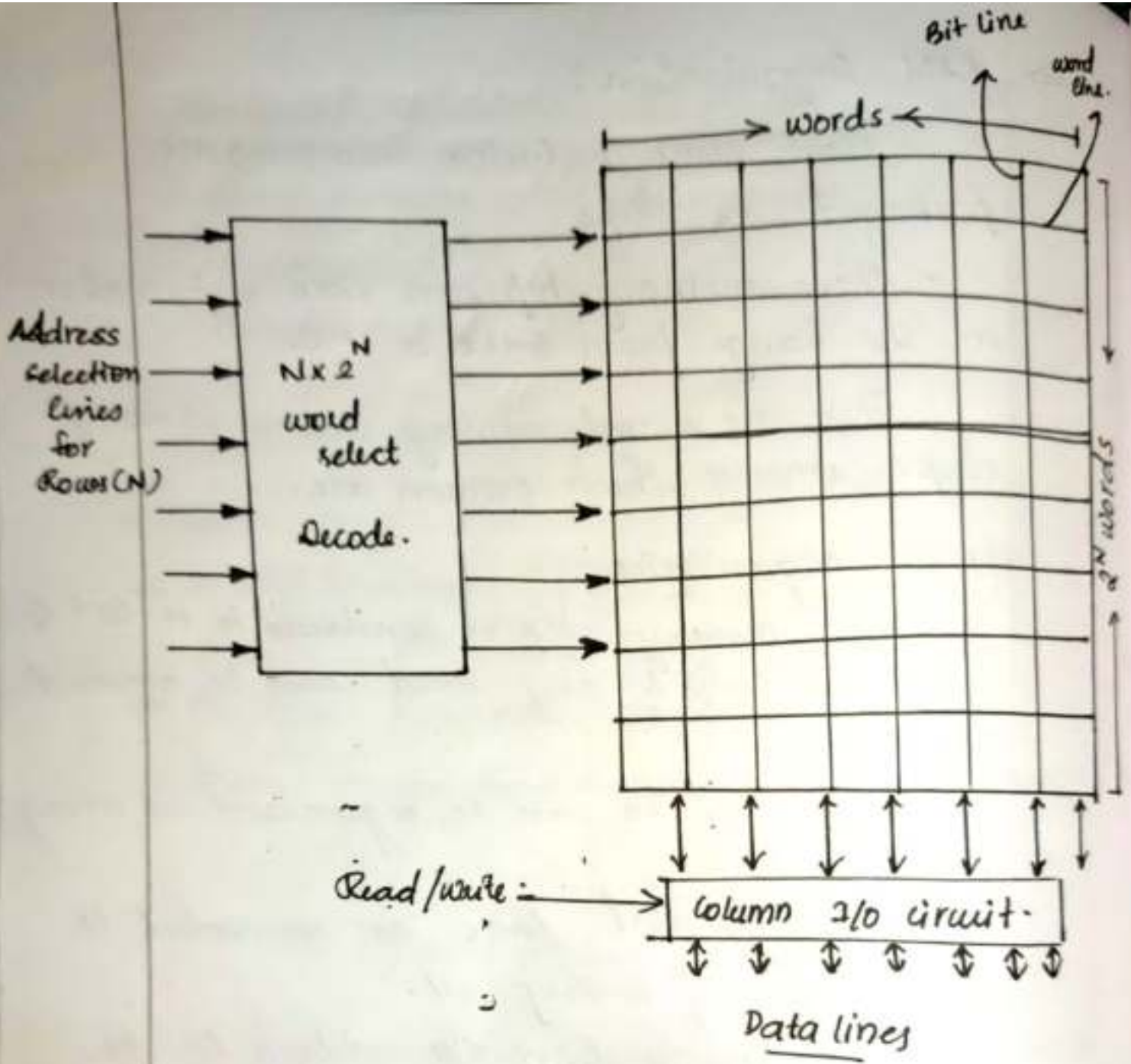
Here memory chip is considered to be list of words in which any word can be accessed randomly.

Memory in 2D can be organised as array of words.

The horizontal lines are connected to select input of binary cell.

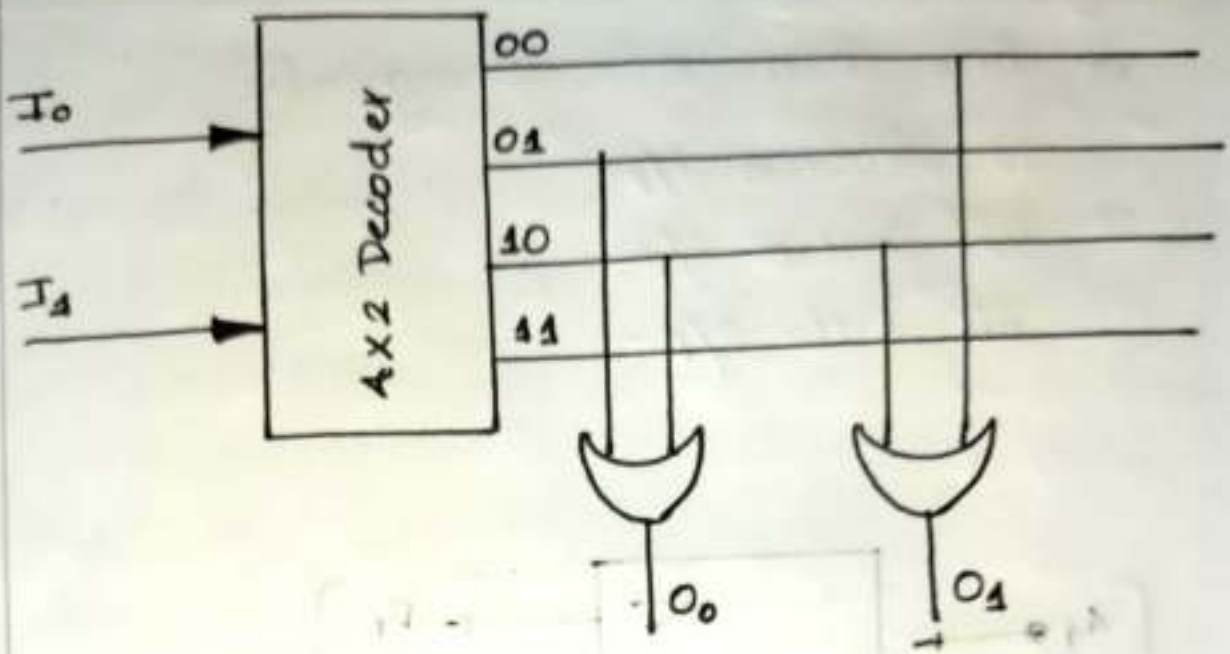
On write operation the addres decoder selects the required word and bit lines are activated for value 0 or 1 according to data line values and complete write operation.

On read operation the value of each bit line is passed through a sense amplifier & passed to data lines. RAM uses 2D chip organisation.



ROM organisation :

→ ROM is abasically a decoder with t i/ps & 2^t addres lines & n o/p lines where no: of o/p '0' are decided by no: of or gates.



Input		output	
I_0	I_1	O_0	O_1
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

Using hardwired combinational circuit ROM can be created

on applying i/p $I_0=0$, $I_1=0$

o/p $O_0=0$, $O_1=1$

on applying i/p $I_0=0$; $I_1=1$

o/p $O_0=1$ & $O_1=0$ as 01 line of decoder is selected. This logic is used for

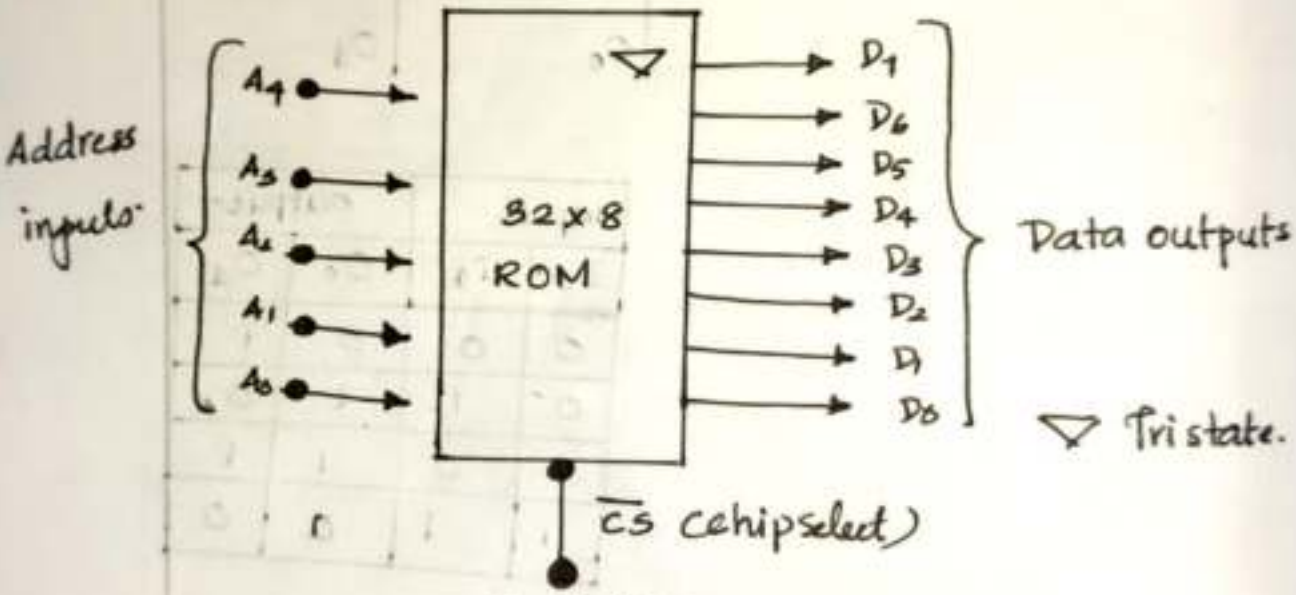
constructing ROMs with higher capacity

It has three sets of signals:

i) Address i/p.

ii) Control i/p

iii) Data o/p.



The above shown ROM can store 32 words since it has 5 address lines and can have $2^5 = 32$ possible words, each word contains 8-bits since there are 8 data outputs.

Data outputs of most ROM integrated circuits are tristate outputs to permit the connection of many ROM chips to the same data bus for memory expansion the control i/p $\overline{CS}$ stands for chip select. chip select enables or disables data. ROM o/ps. of the

Many ROMs have two or more control i/p. that can be active inorder to enable the data outputs.

In some ROM ICs one of the control inputs is used to place the ROM in a low power standby mode when it is not being used. This reduces current drawn from systems power supply.

Organisation of memory unit:

Memory unit is organised in two different ways:

Physical organisation of computer memory. and Logical organisation of computer memory

P.T.O.

Physical organisation of computer memory:

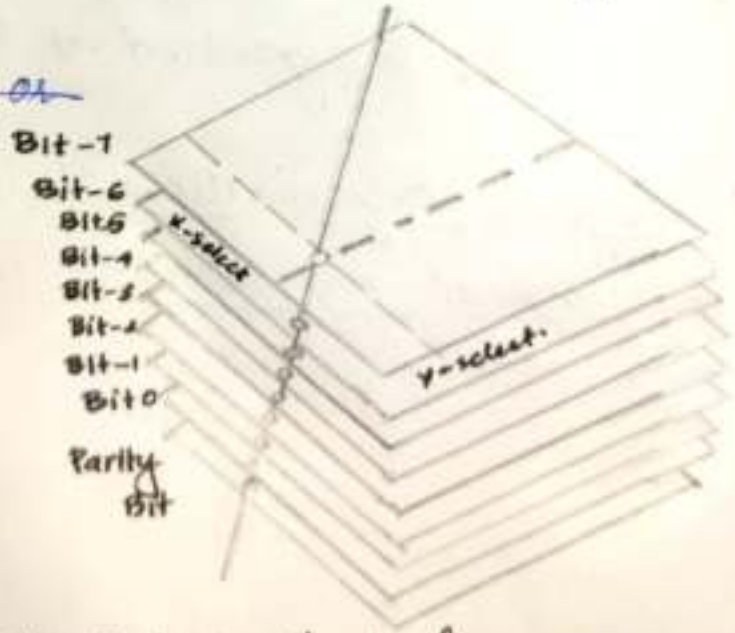
→ Over past 20 years, magnetic core has been major form of computer memory. After the recent development of semiconductor memory most computers now use it.

The major deciding factors are speed & cost

Semiconductor memories are faster, but until recently ~~expensive~~ it has been more expensive.

Core memories have been an important ~~no major~~ form of computer main memory.

~~Logical~~ ~~or~~



Core planes stacked together to form memory module.

Logical Organisation of Computer Memory -

All (current) computer memories are composed of basic units which can be in either of two possible states.

By arbitrarily assigning a label to each of these two states, we can store information in these memory devices.

Traditionally we assign label 0 to one state and label 1 to the other state.

If we write 1 into a specific cell then a subsequent read of that cell should return a 1. and if we write 0, 0 should be returned.

Each word of memory has an address. Address of word is the bit pattern which must be put into the memory address register in order to select word in the memory module.

In general every possible bit pattern in memory address register will select a diff. word in # memory.

Thus maximum size of memory unit (no: of different words) depends upon the size of memory address register.