-> The printing characters consists of 26 - upper cases (A - Z)
26 - lower case letters
10 - numericals (0 - 9)
32 - special printable characters like %, *, $ ... etc.,

(ip) Control functions:-

Communication control characters (Text b/w remote terminals)
- NUL - NULL
- SOH - Start of heading
- STX - Start of text
- ETX - End of text
- EOT - End of transmission
- ENQ - Enquiry
- ACK - Acknowledge
- BEL - Bell

Format Effort (layout printing)
- BS - Back space
- HT - Horizontal Tab
- LF - Line feed
- VT - Vertical tab
- FF - Form feed
- CR - Carriage return

- SO - Shift out
- SI - Shift In
- SP - Space

- DLE - Data link escape
- DC1 - Device Control 1
- DC2 - " " 2
- DC3 - " " 3
- DC4 - " " 4
- NAK - negative Acknowledge
- SYN - Synchronous idle
- ETB - End of transmission block
- CAN - Cancel
- EM - End of medium
- SUB - Substitute
- ESC - Escape

Information separator (used in division of paragraph)
- FS - file separator
- RS - Record separator
- GS - Group separator
- US - unit separators
- DEL - Delete.

There are 3 - types of control characters.

(a) Format effort : are characters that control the layout printing which include familiar type cursor controls like
BS (Back space) ; HT ; CR ;

(b) Information separator :- are used to separate the data into division like paragraphs and pages which include
RS (Record separator) ; file separator (FS)

(c) Communication Control characters :- are useful during tx of text b/w remote terminals
Eg:- STX ; ETX which are used to frame a text message.

(b) Input-output interface :- is a method of transferring information b/w internal storage & external I/O devices.

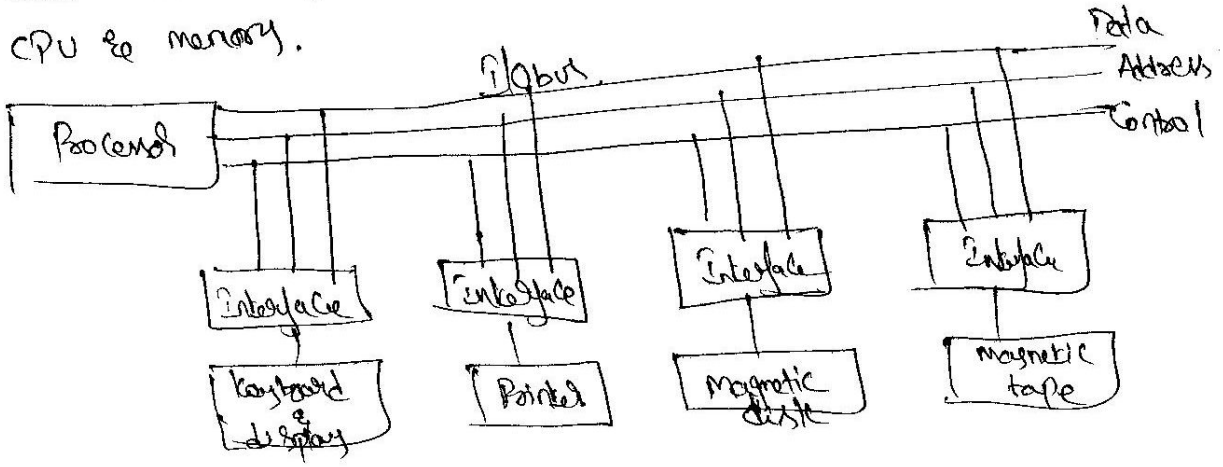
→ Major Difference b/w Peripheral devices & CPU is.

① Peripheral are electromechanical & electromagnetic devices and CPU & memory are electronic devices which have major difference from operation:

② The Data transfer rate of peripheral is usually slower than transfer rate of CPU. which needs Asynchronization mechanism.

③ Data codes and formats in peripherals differ from word formats in CPU & memory.

(c)



There are 4-types of Commands

(a) Control → is issued to activate Peripheral.

(b) Status → is used to test various status conditions

(c) Data output → responds by transferring data from bus into one of its registers

(d) data input → ↓

→ 3-ways that Computer buses used to Communicate memory/I/O.

① use two separate buses one for memory & other for I/O

② use one Common bus for both I/O but have separate Control lines for each.

③ use one Common bus for memory & I/O with Common Control lines.

(c) Isolated versus memory mapping I/O:-

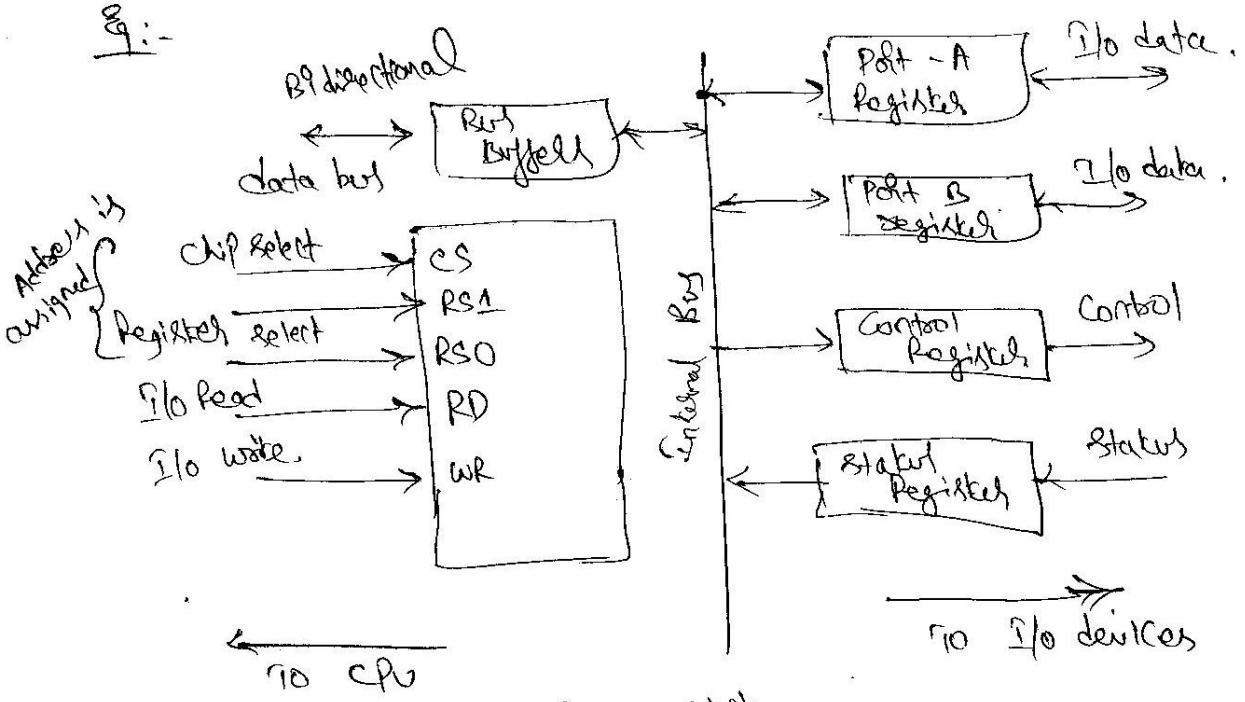
Isolated I/O is method for assigning address in common bus

→ the memory mapped I/O is to use the same address space for both memory and I/O.

→ The I/O Read & write & Enables control lines during I/O transfer.

→ memory Read & memory write Enables " " " I/O ".

Eg:-



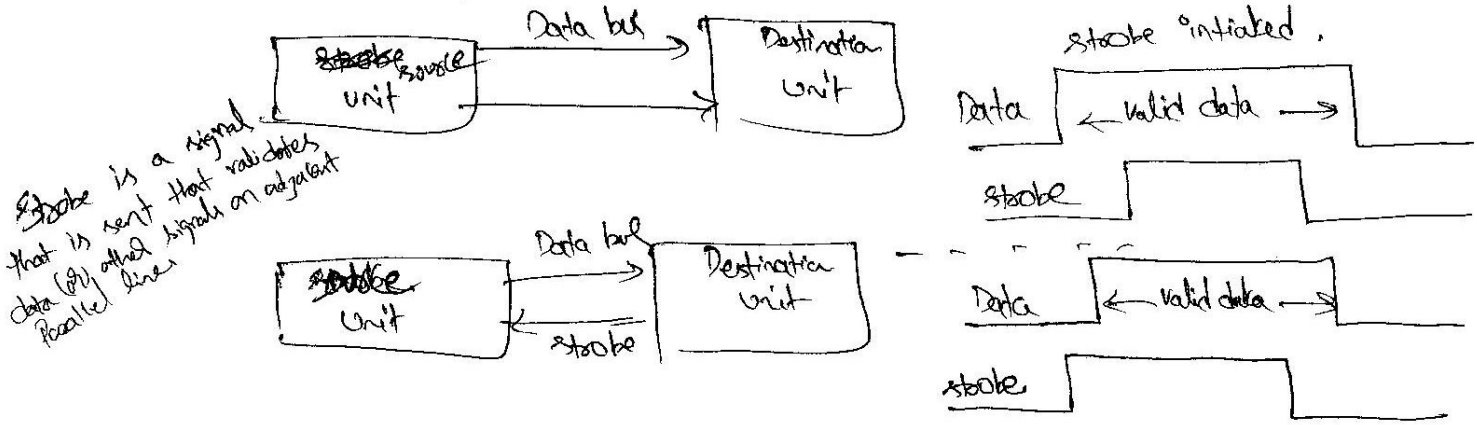
Chip Select CS	Register Select RS1	RS0	Register Selected
0	X	X	none: data bus in high impedance
1	0	0	Port - A Register
1	0	1	Port - B "
1	1	0	Control Register
1	1	1	Status Register.

(a) Asynchronous Data transfer :-

// // b/w two independent units require control signals be transmitted thru the communicating units to indicate the time at which being transmitted.

→ The strobe control method of Asynchronous data transfer employs a single control line to time each transfer.

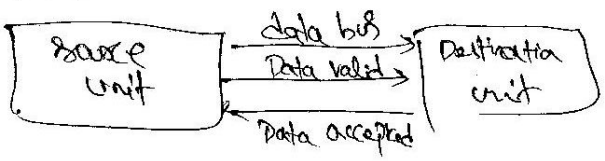
strobe may be activated either source (or) destination unit

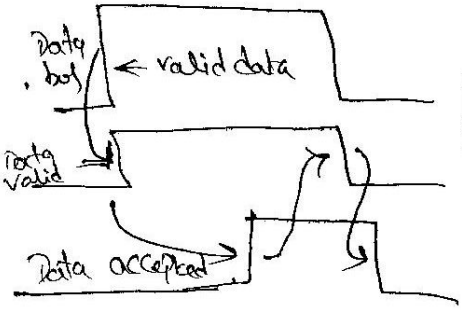


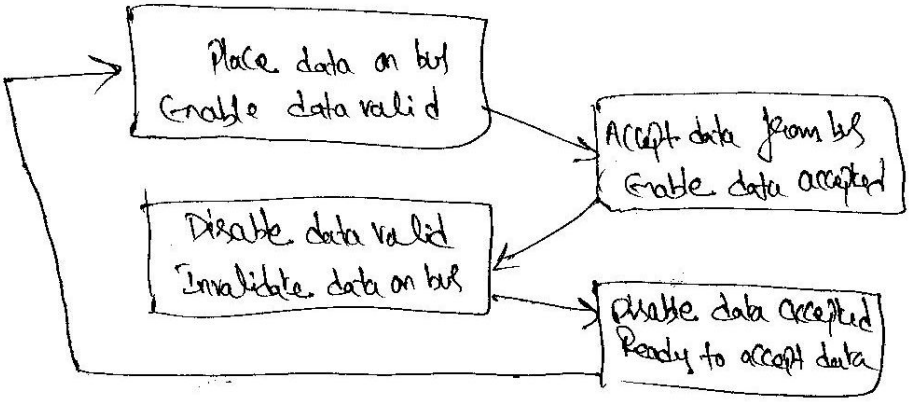
(b) Handshaking :-

The dis-advantage of strobe method is source/destination units initiates transfer has no way of knowing whether data is placed on bus (or) not

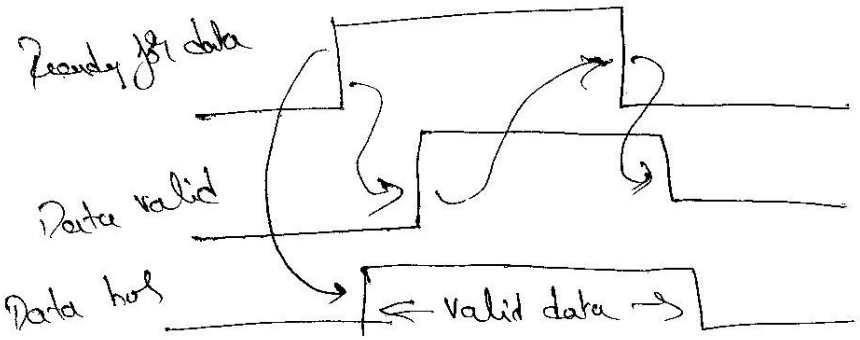
The handshaking methods solve by using (a) Data valid ; (b) Data accepted.

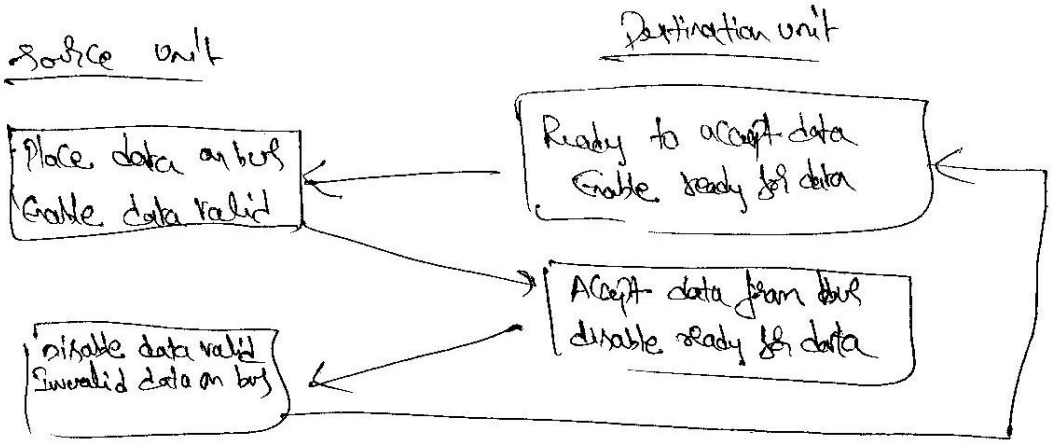




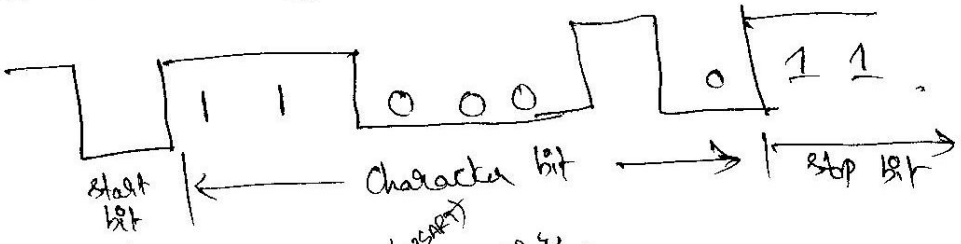


Source initiated transfer using handshaking

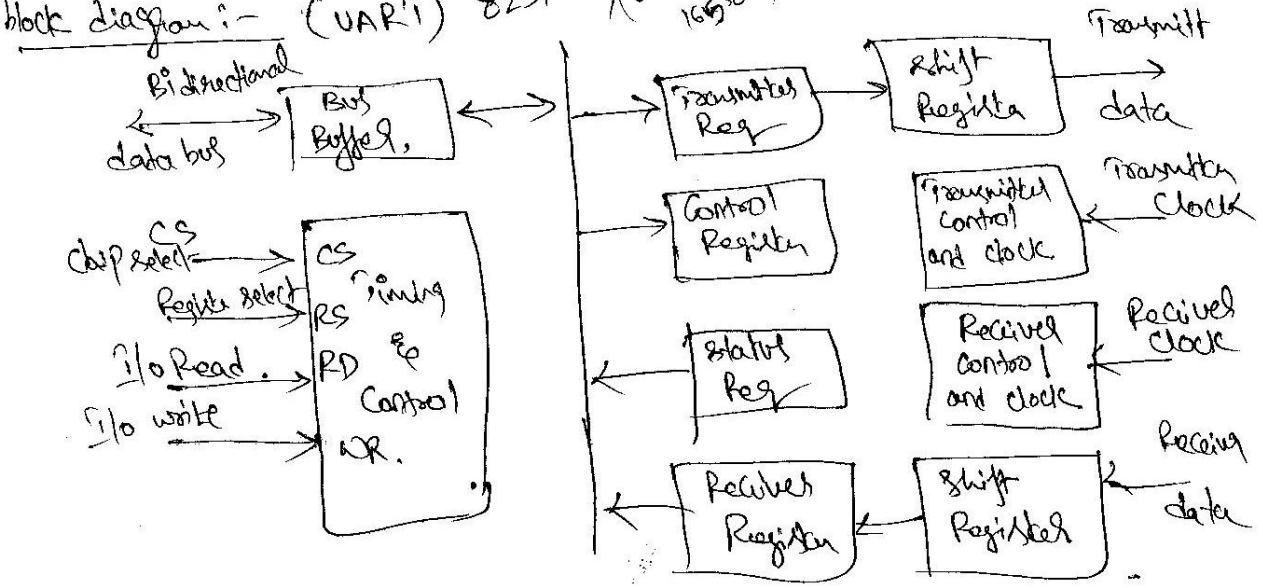




(c) Asynchronous Serial transfer:-



Block diagram:- (UART) 8251 (USART) (8250 & 16550)



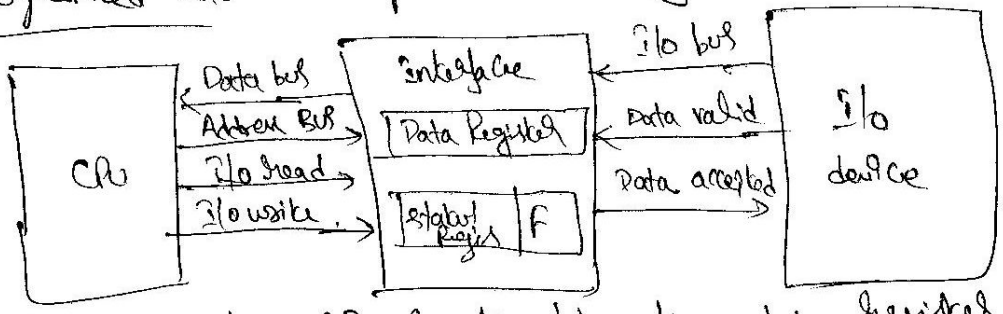
CS	RS	operation	Register selected
0	X	X	none data bus in high impedance
1	0	WR	Transmitter Register
1	1	WR	Control Register
1	0	RD	Received Register
1	1	RD	Status Register.

3) Modes of transfer:-

Data transfer to & from peripherals may be handled in 3-ways.

1. Programmed I/O
2. Interrupt - initiated I/O
3. Direct Memory Access (DMA)

1) Programmed I/O :- requires execution of several instructions by the CPU.



→ If flag = 1 the CPU reads data from data register otherwise it cleared

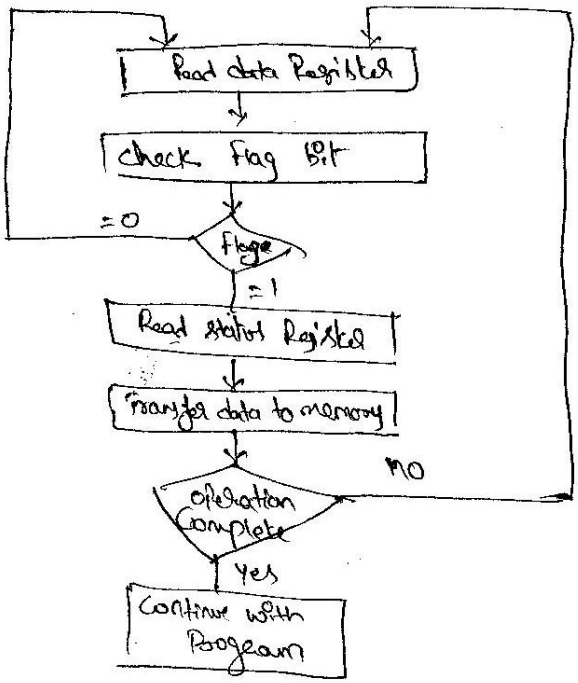
→ the transfer of each byte requires following steps

(a) Read the status register

(b) Check status of flag bit and branch to step 1 if not set (or) to step-3

(c) Read the data register

Flowchart



(3) Interrupt initiated I/O:- CPU responds to interrupt signal by storing the return address from the program Counter on to memory stack.

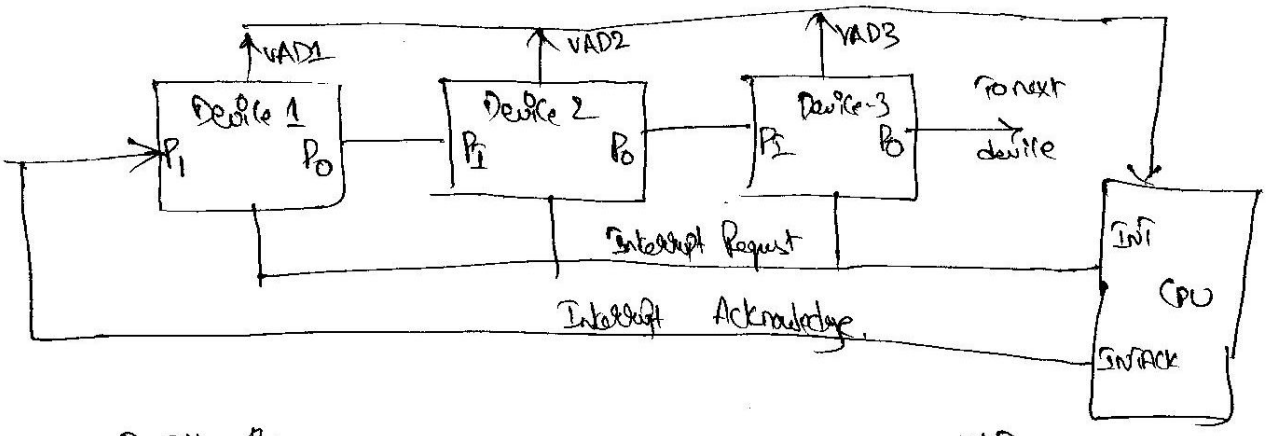
4) Priority Interrupt:- is a system that establishes a priority over various sources to determine which condition is to be serviced first. when two (or) more request arrives simultaneously.

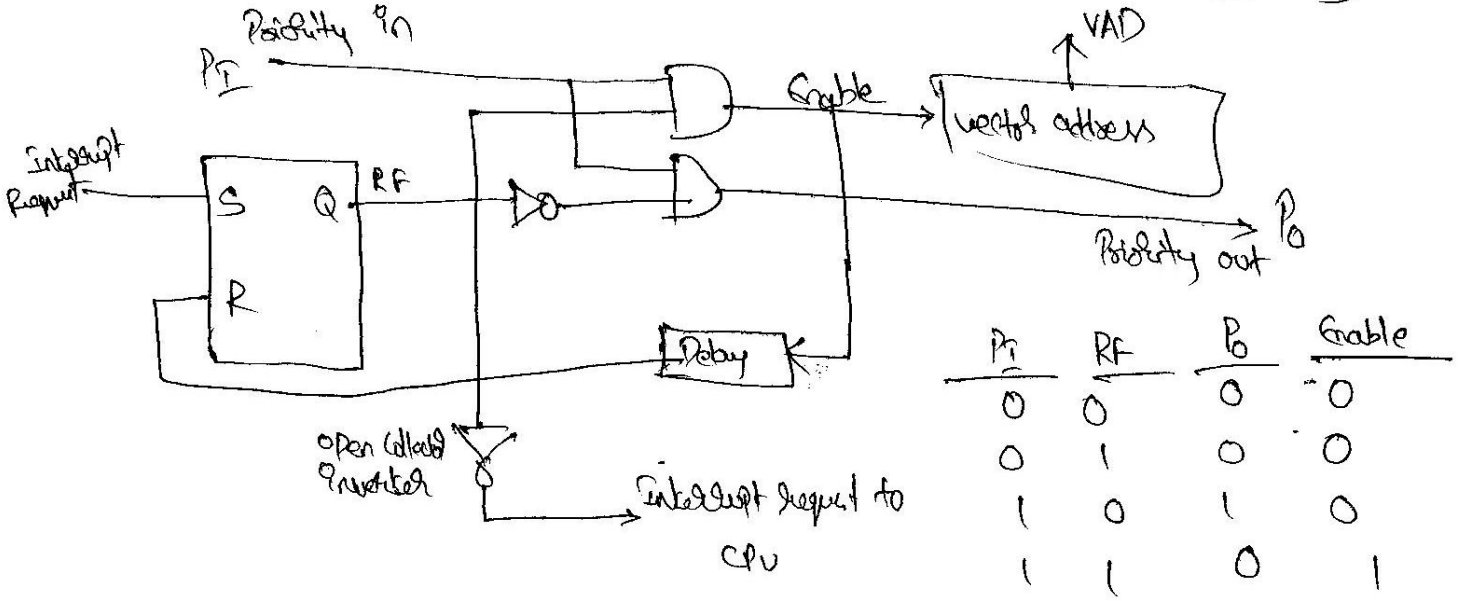
→ The High priority interrupt levels are assigned to request which is delayed (or) interrupted.

→ Devices with high speed transfer such as magnetic disks are given high priority
→ " " slow " " " " keyboards receives low "

(a) Daisy chaining Priority:-

→ method consists of serial connection of all devices that request interrupt
→ the device with high priority is placed in first position. followed by lower priority.
→ Each Devices has its own interrupt vector (VAD) in to data bus



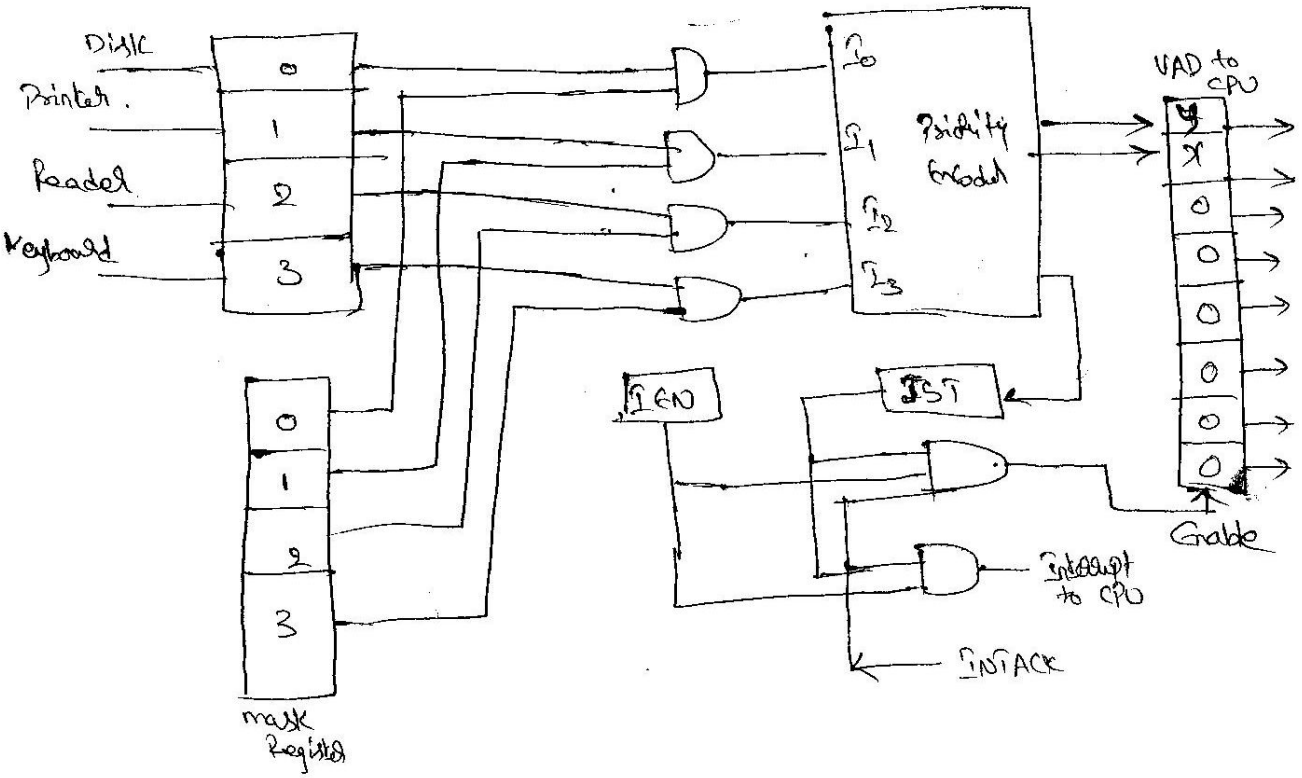


PI	RF	PO	Enable
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

Parallel Priority interrupt:-

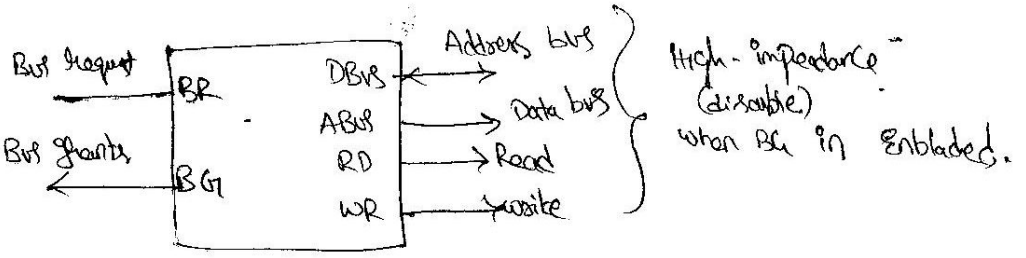
The parallel priority interrupt method uses a register whose bits are set separately by the interrupt signal from each device. Priority is established according to the position of the bits in register. In addition to interrupt register, the circuit may include a mask register whose purpose is to control the status of each interrupt request.

The mask register can be programmed to disable lower priority interrupts while a higher priority device is being serviced.



5) Direct Memory Access (DMA):-

The transfer of data btw a fast storage device such as magnetic disk and memory often limited by the speed of the CPU, removing the CPU from the path and letting the peripheral device manage the memory buses directly, the data formats of peripheral devices differ from memory and CPU data formats. This transfer technique is called direct memory Access (DMA)



→ The direct memory Access (DMA) I/o technique provides direct access to memory while the µP Processor is temporally disable.

→ I/o device is connected to system bus via a special interference circuit known as DMA Controller.

→ In DMA both CPU and DMA Controller have access to main memory via a shared system bus having data, address and control lines.

→ A DMA Controller temporally takeover the address bus, data bus, and control bus from microprocessor and transfers the data bytes directly from an I/o port and series of memory locations.

→ DMA transfer is also used to do high speed memory to memory transfers.

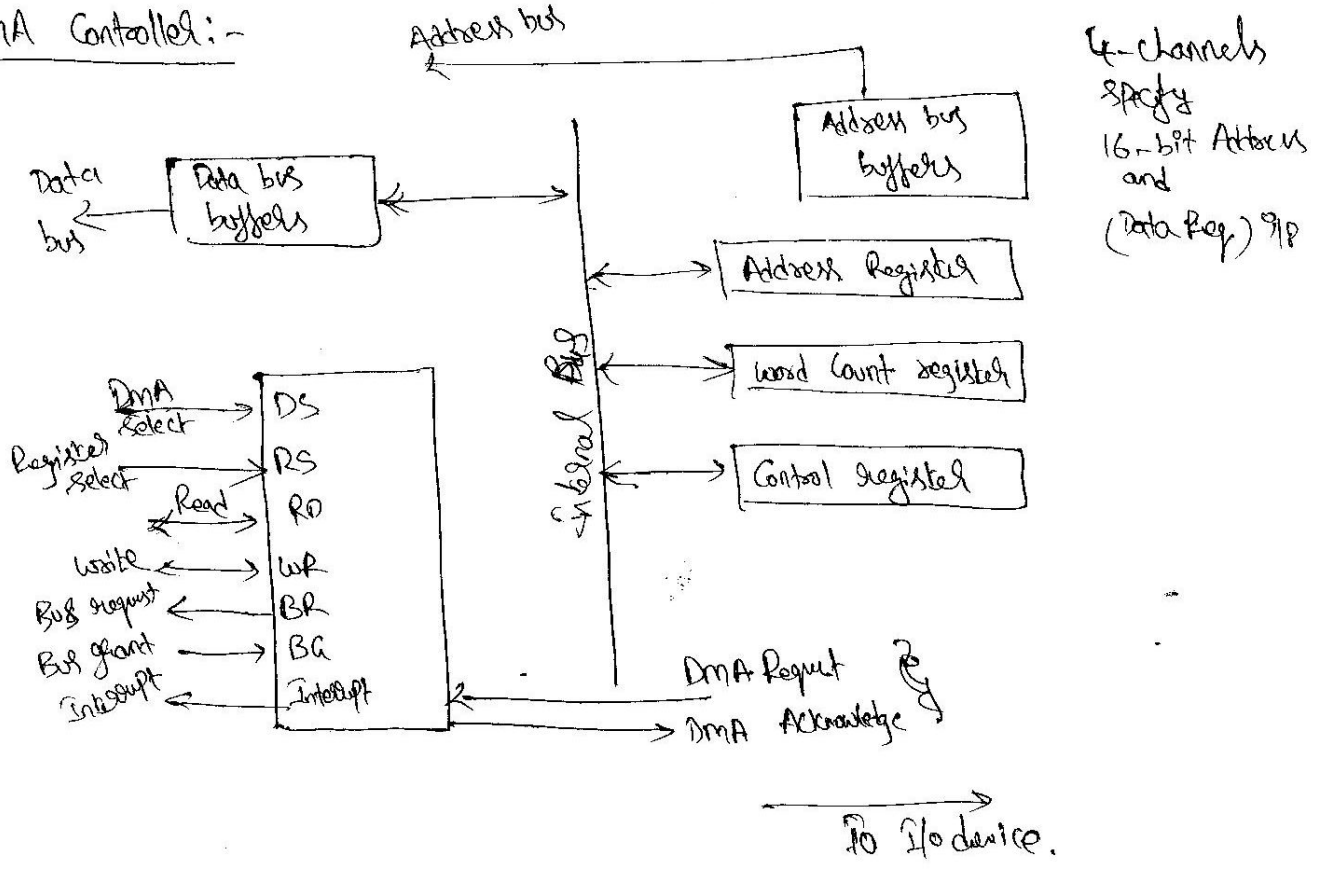
With out DMA:

→ Using Programmed I/o it is typically fully occupied for entire duration of Read (or) write operations and thus unavailable to perform other work.

With DMA

→ The CPU initiates the transfer, does other operations while the transfer is in progress and receives an interrupt from DMA controller when operation is done.

DMA Controller:-



4-channels specify 16-bit Address and (Data Reg) 8/p

The (BR) Bus Request i/p is used by DMA Controller to request the CPU to relinquish Control of the buses.

The transfer can be made in several ways

1. Burst transfer : allows DMA Controller to transfer Entire block at a time.
2. Cycle Stealing :- allows " " " " one data at a time

DMA Controller :-

*) when (BG) Bus Grant is '0' the CPU can communicate with DMA Registers through the Data bus to read from (or) write to DMA Registers.

*) when BG is '1' the CPU has relinquished the buses and DMA can communicate directly with memory by specifying an address in the address bus and activating the RD (or) WR Control.

*) DMA Controller has 3-Registers (Address Register), (word Count Register), (Control register)

Address Register → Contains an address to specify the desired location in memory

" " is incremented after each register holds the no:- of words ~~word~~ ~~internally~~ to be transfered

It is decremented by one after each word transfer and internally tested for zero.

*) CPU initialize the DMA by sending ~~$~~ information through data bus :-

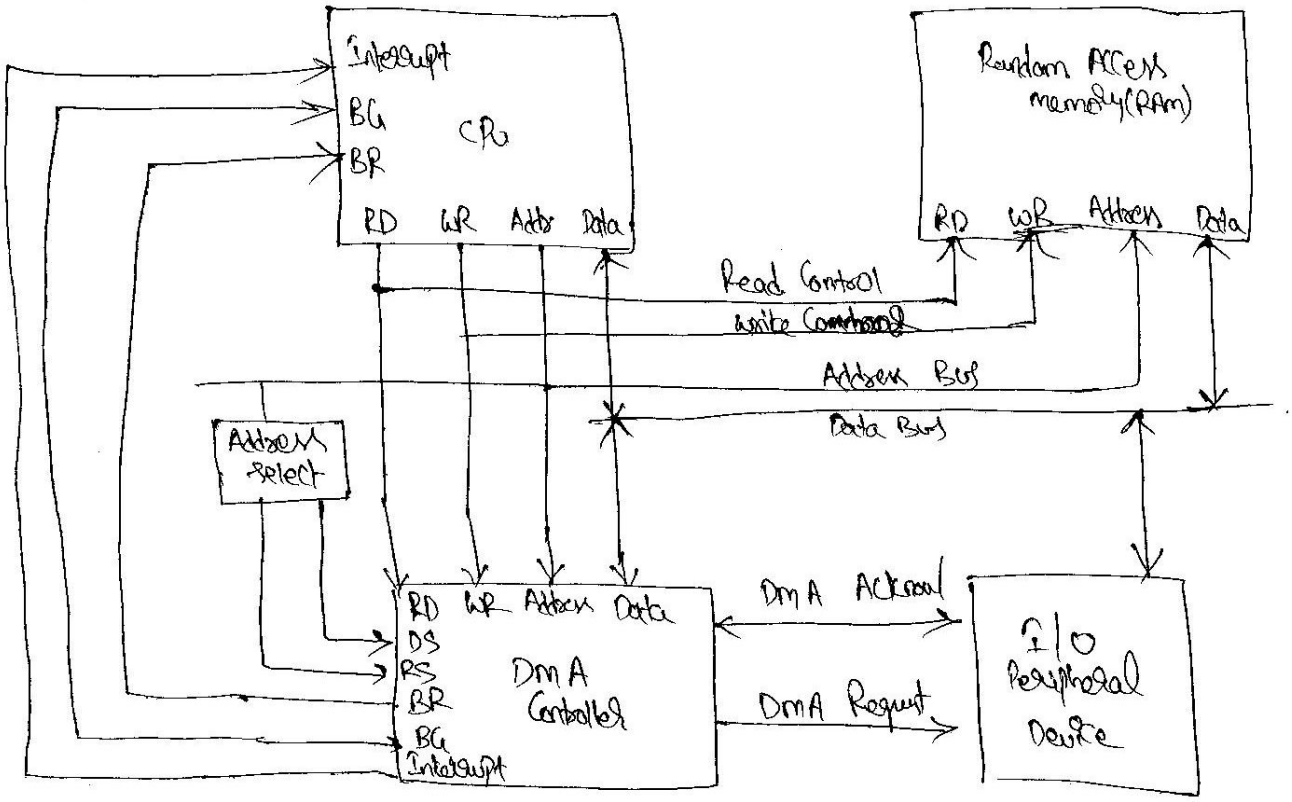
(a) The starting address of memory block where data are available (or) where data are to be stored.

(b) The word Count, which is the no:- of words in the memory

(c) Control to specify the mode of transfer such as read (or) write

(d) A Control to start the DMA transfer.

DMA transfer:-



The CPU communicates with the DMA through address and data buses as with interface unit.

→ The DMA has its own address, which activates DS and RS lines

→ When Peripheral device sends the DMA Request the DMA Controller Activates the BR line, informing the CPU relinquish the buses.

→ when peripheral device receives DMA Acknowledge it Puts a word in the Data bus (for write) (or) recall a word from the data bus (for Read).

6) **Multiprocessors:-**

↓ is an inter Connection of 2 (or) more CPU with memory & I/O devices

means CPU (or) Input-output processor (IOP)

→ A multiprocessor system with with common shared memory is classified as shared memory (or) tightly coupled multiprocessor.

Multiprocessors:- (CPU (or) I/P - O/P processors)

Inter Connection Structure:-

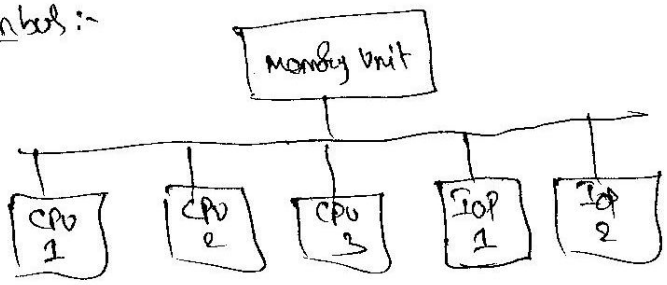
The way the components can have different physical configuration depending on no: of transfer path that are available b/w processor and memory in shared memory system.

→ Several physical form available for establish an inter connection n/w

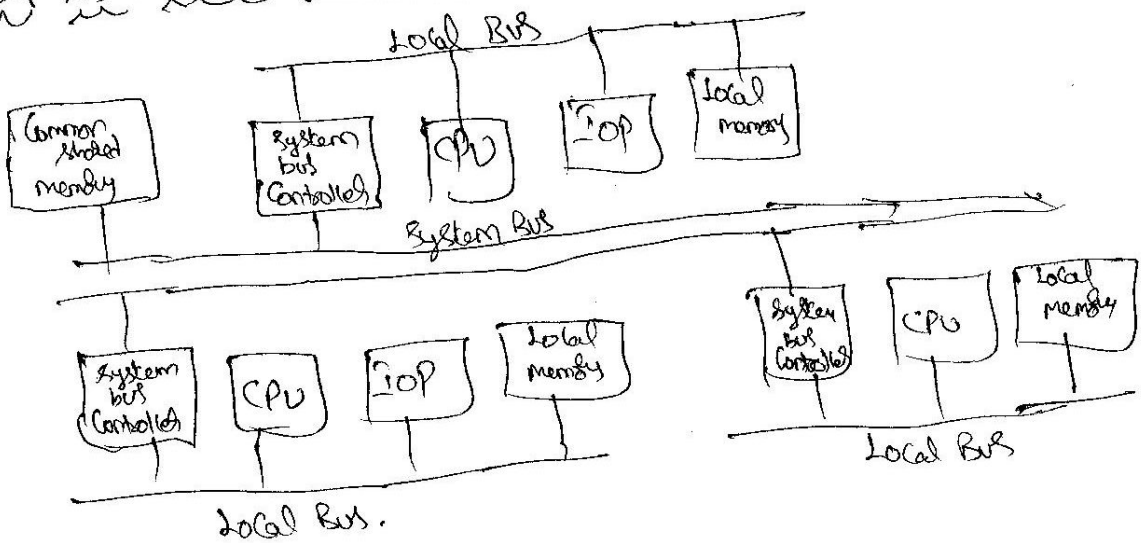
1. Time shared Common bus
2. multi port memory
3. Cross bar Switch
4. multi stage switching n/w
5. Hyper Cube system

(1) Time shared Common bus:-

only one processor can communicate with memory (or) another processor at any time.



System bus structure for multiprocessors:-



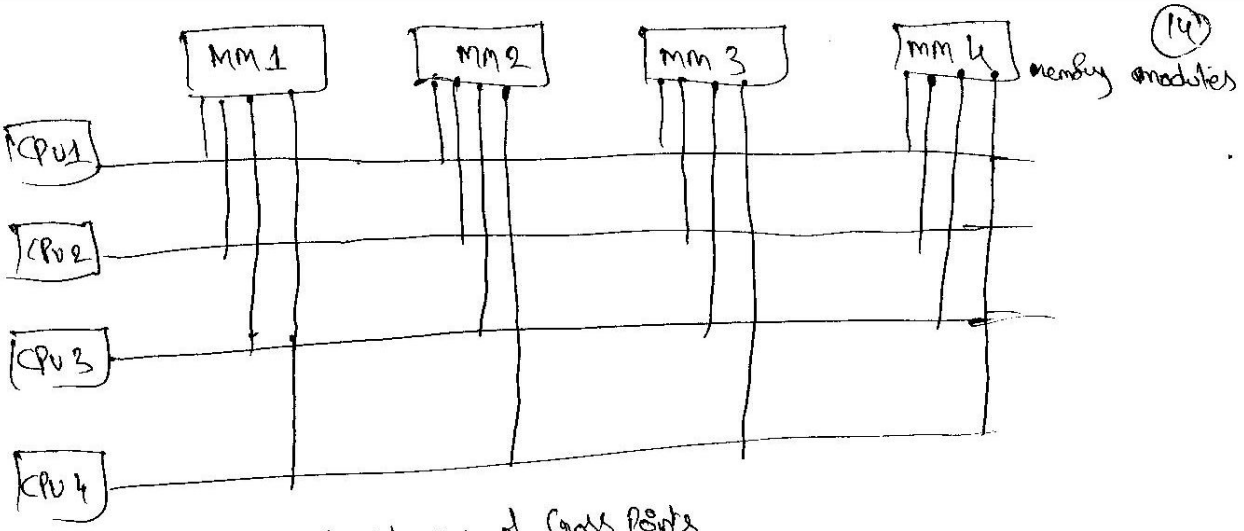
(2) Multi port memory:-

Employs seperate bus b/w each memory module and CPU.

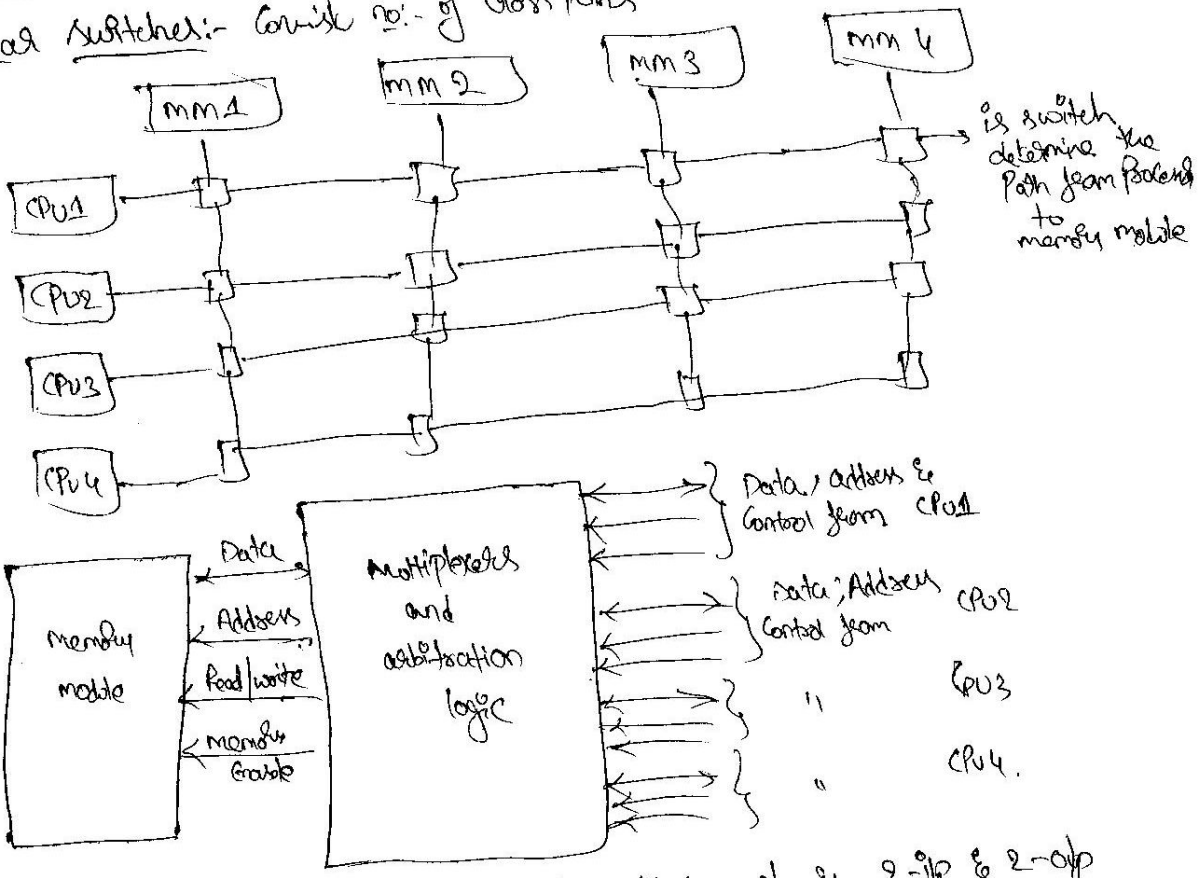
→ A processor bus consists of address, data and control lines required to communicate with memory.

→ Advantage of multiport memory organization is high transfer rate is achieved

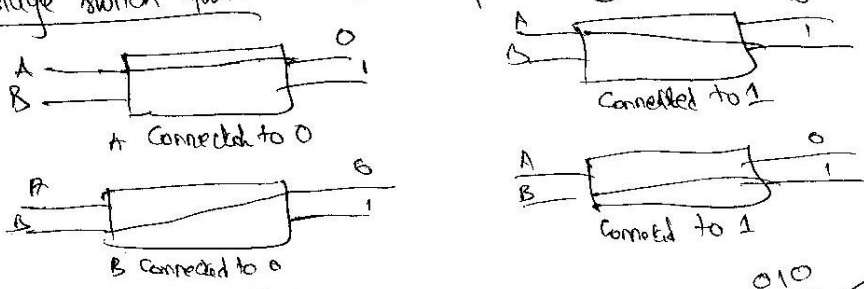
→ dis advantage :- requires expensive memory control logic and large no: of cables.



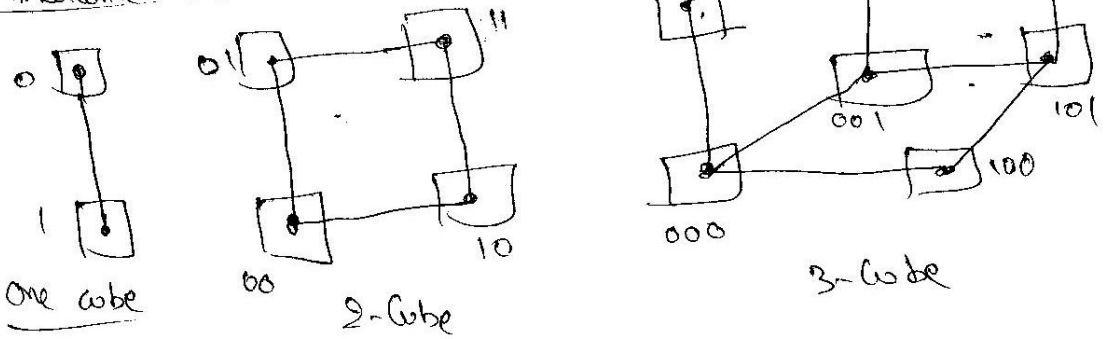
(3) Crossbar switches:- Consist no:- of cross points



(4) Multistage switch n/w:- Basic component of multistage n/w is 2-i/p & 2-o/p



Hyper Cube interconnection:-



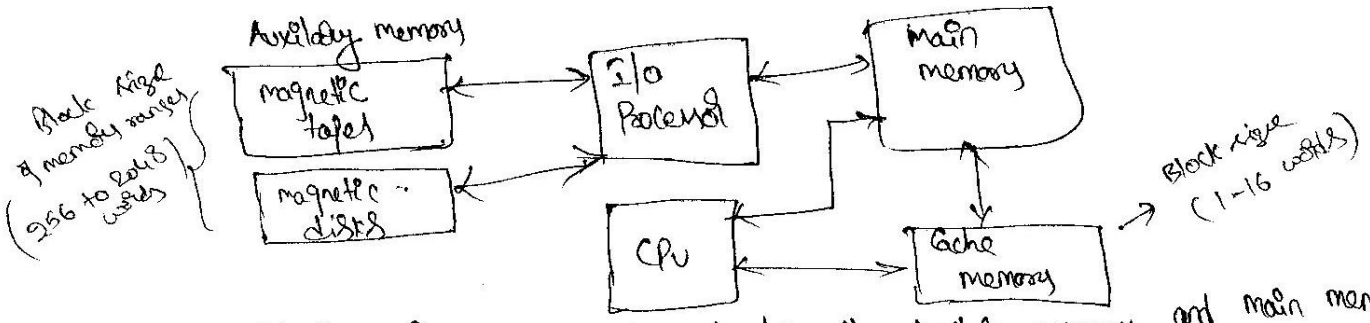
2) Memory Hierarchy:-

→ The memory unit communicate directly to CPU is called Main memory

→ Devices that provide backup storage are called Auxiliary memory

Eg:- magnetic disks & magnetic tapes. (used for storing system programs, large data files, and other backup information).

→ A special very high speed memory called Cache is sometimes used to increase the speed of processing by making current programs and data.



→ While I/O processor manages data transfer b/w Auxiliary memory and main memory the Cache organization is concerned with the transfer of information b/w main memory and CPU.

→ The Auxiliary memory has large storage capacity but has low access speed compared to main " .

→ The Cache memory is very small and has very high Access speed.

→ Typical Access time ratio b/w Cache & main memory is 1:7

Eg:- A typical Cache memory may have Access time 100ns
main " " " " 700ns
Auxiliary " " " " 1000 times than main memory.

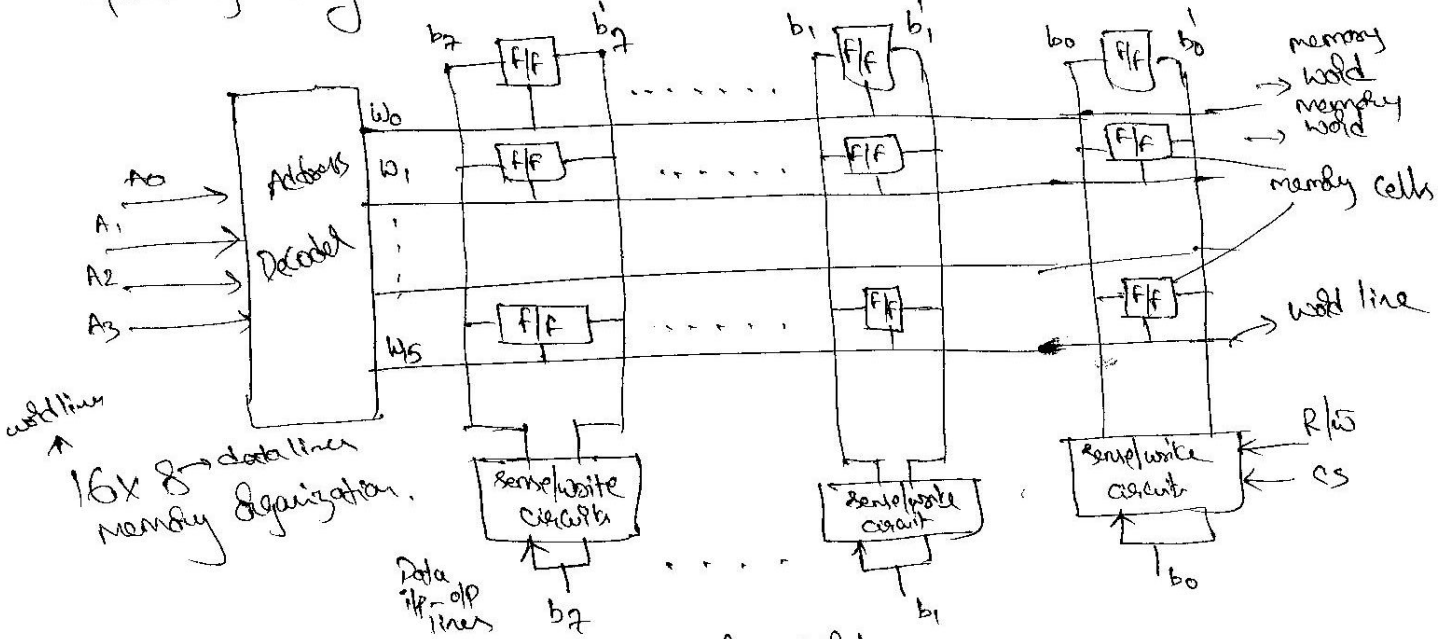
Ranges:- Block size in Auxiliary memory ranges 256 to 2048 words.
" " Cache " 1 to 16 words.

→ The part of the Computer system that supervises the flow of information b/w Auxiliary memory and main memory is called memory management system.

Main memory:- is Central storage unit in Computer system. which is relatively large and fast memory used to store Programs and data during Computer operation.

→ Internal Organization of memory chips:-

memory cells are usually organized in form of array in which each cell is capable of storing 1-bit information.



Each row of cells Constitutes memory word.

Read operation:- the information stored in the cells selected by a word line and transmits the information to o/p data lines.

→ write operation:- the circuit sense/write receives i/p information and store it in cells of selected word.

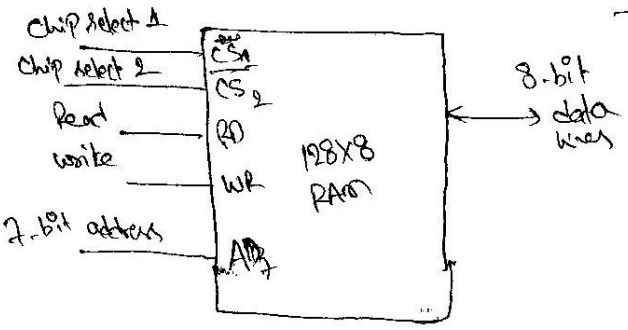
→ Integrated circuit RAM chips are available in 2-Operating modes → Static, → Dynamic

→ The static RAM Consists essentially of internal flip-flops to store binary information. the stored information remains valid as long as power is applied to unit.

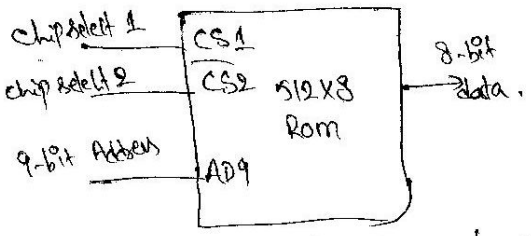
→ The Dynamic RAM stores binary information in form electric charges that are applied to Capacitors.

→ the Ram portion of main memory is needed for storing an intial Program Called boot strap. loader.

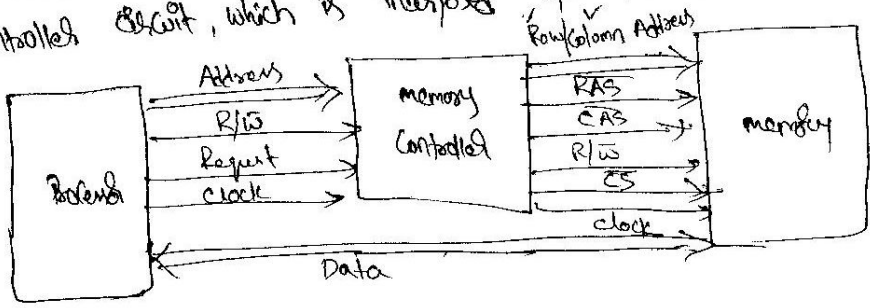
RAM and ROM Switches:-



$\overline{CS1}$	CS2	RD	WR	function	State of data bus
0	0	x	x	Inhibit	High impedance
0	1	x	x	Inhibit	High "
1	0	0	0	Inhibit	High "
1	0	0	1	write	Input data to RAM
1	0	1	x	Read	output data from RAM
1	1	x	x	Inhibit	High impedance



Memory Controller:- The required multiplexing of address bits is usually performed by a memory controller circuit, which is interposed b/w processor and the dynamic memory.



→ **Virtual Memory:-**

Is used to increase the apparent size of physical memory.

→ Some time data may be stored in physical memory location that have address different from those specified by the program.

In such cases address generated by processor is referred as virtual (or) logical address.

→ The virtual address is mapped into physical memory where data are actually stored. mapping function is implemented by a special memory unit called memory management unit.

UNIT-4

THE MEMORY SYSTEM

* Some basic concepts

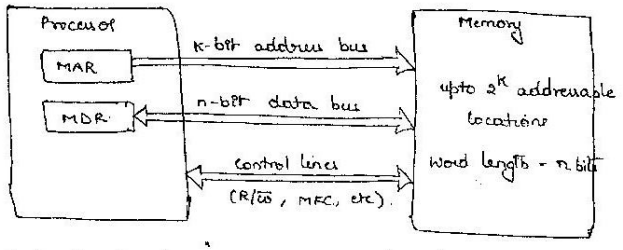
→ Maximum size of memory that can be used in any computer is defined by addressing scheme.

Eg: Computer that generates 16-bit address is capable of addressing upto 2^{16} = 64K (Kilo) memory locations.

32 bit address ⇒ 2^{32} = 4G (giga) memory locations

40 bit address ⇒ 2^{40} = 1T (tera) memory locations

→ connections between processor & memory is shown below:



Data transfer between processor and memory takes place through 2 processor registers, MAR (Memory Address Register) and MDR (Memory Data Register)

If MAR = K-bit register ⇒ K-bit address bus

⇒ 2^K addressable locations

If MDR = n-bit register ⇒ n-bit data bus

⇒ word length = n-bits.

⇒ during a memory cycle, n-bits of data are transferred between memory and processor.

Word length defines the no. of bits stored or retrieved in 1 memory access. Control lines include R/$\overline{W}$ (Read / write) and MFC (Memory Function completed) for co-ordinating data transfer.

Other control lines can be included to ~~indicate~~ indicate the no. of bytes to be transferred.

- Processor loads data from memory by loading the address of the required memory location into MAR register and setting R/$\overline{W}$ line to 1.
- Memory responds by placing data from the addressed location onto data lines, and confirms this action by asserting MFC signal.
- Upon receipt of MFC signal, the processor loads the data on the data lines into MDR register.

b) Write Operation

- Processor writes data into memory location by loading the address of this location into MAR and loading data into MDR.
- Set R/$\overline{W}$ = 0 to indicate a write operation.
- Word in MDR is copied into memory location indicated by MAR.

If Read or Write operation involves consecutive address locations in the memory, then Block Transfer operation can be performed, in which the 1st location address is only sent.

→ Measure of speed of memory units is defined by

a) Memory Access time

b) Memory Cycle time

- Memory Access time

It is the time that elapses between the initiation of an operation and the completion of that operation.

Eg:- time between Read and MFC signals.

- Memory Cycle time

It is the minimum time delay required between 2 successive Read operations

a) How many chips are needed, and how should their address lines be connected to provide a memory capacity of 1024 bytes?

Sol: Memory capacity = 1024 bytes = 1024 × 8.

Size of each RAM chip = 1024 × 1

⇒ We requires 8 chips with their address lines connected in parallel.

b) How many chips are needed to provide a memory capacity of 16K bytes? Explain in words how the chips are to be connected to the address bus?

Sol: Memory capacity = 16 K bytes = 16 × 1024 × 8.

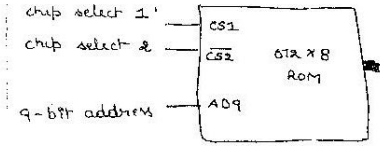
Given size of RAM = 1024 × 1

No. of chips required = $\frac{16 \times 1024 \times 8}{1024 \times 1}$ = 16 × 8 = 128 chips

No. of address lines required = 14 ($2^4 \times 2^{10} = 16 \times 1024$).

4 of which to lines specify address in each chip and 4 lines are decoded into 16 chip-select i/p's.

* ROM chips



Hence, ROM is read only, data bus can only be in o/p mode.

The 9-bit address specify any one of the 512 bytes stored in ROM.

For ROM chip to operate, CS1 = 1 and $\overline{CS2}$ = 0, otherwise, data bus is in high-impedance state.

There is no need for read & write control because the can only read. ⇒ when chip is enabled, the byte selected by the address lines appears on the bus. (data bus).

(Pb) A ROM chip of 1024 × 8 bit has 4 select i/p's and operates from a 5V power supply. How many pins are needed for IC package? Draw a block diagram and label all i/p and o/p terminals in the ROM.

Sol: 1024 × 8 ROM is constructed from ~~8 512 × 8 ROMs~~ with 4 select i/p's and 5V power supply.

⇒ No. of select i/p's = 4.

No. of power supply pins = 2 (V_{cc}, GND)

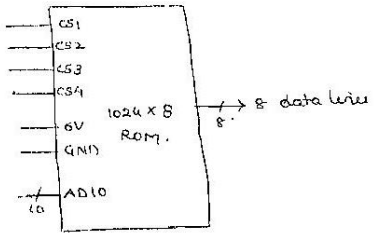
(9.16)

1024 × 8

⇒ No. of address lines = 10.

No. of o/p's = No. of data lines = 8.

⇒ Total no. of pins in 1024 × 8 ROM = 4 + 2 + 10 + 8 = 24 pins.



* Memory Address Map

The designer of a computer system must calculate the amount of memory required for the particular application and assign it to either RAM or ROM. The interconnection between memory and processor is then established from the knowledge of the size of the memory needed and type of RAM or ROM chips available.

The addressing of memory can be established by means of a table that specifies the memory address assigned to each chip. The table is a pictorial representation of assigned address space for each chip in the system and is called memory address map.

(Pb) Design a computer system which needs 512 bytes of RAM and 512 bytes of ROM.

Sol: Size of RAM chip = 128 × 8

Size of ROM chip = 512 × 8

⇒ To design a memory of 512 bytes of RAM and 512 bytes of ROM, one require 4 ($\frac{512}{128} = 4$) RAM chips and 1 ROM chip.

Total no. of address locations = 512 in RAM + 512 in ROM = 1024 = 2^{10}.

⇒ no. of address lines required = 10.

CPU has 16 address lines ⇒ higher order 6 address lines 16 to 11 is always 0's.

Address lines 8 and 9 act as chip-select for the 4 RAMs.

Distinction b/w RAM and ROM is done by address line 10.

Component	Hexadecimal address	Address Bus									
		10	9	8	7	6	5	4	3	2	1
RAM 1	0000 - 007F	0	0	0	x	x	x	x	x	x	x
RAM 2	0080 - 00FF	0	0	1	x	x	x	x	x	x	x
RAM 3	0100 - 017F	0	1	0	x	x	x	x	x	x	x
RAM 4	0180 - 01FF	0	1	1	x	x	x	x	x	x	x
ROM	0200 - 03FF	1	x	x	x	x	x	x	x	x	x

CS2 — Particular RAM selected — 7 lower Addr. bits

CPU

Address

16 11 10 9 8 7-1 RD WR Data bus

Decoder 3 2 1 0

CS1, CS2, RD, WR, AD7 — 128 x 8 RAM 1

CS1, CS2, RD, WR, AD7 — 128 x 8 RAM 2

CS1, CS2, RD, WR, AD7 — 128 x 8 RAM 3

CS1, CS2, RD, WR, AD7 — 128 x 8 RAM 4

CS1, CS2, AD9 — 512 x 8 ROM

Memory Connection to CPU

$\frac{9}{8} = 512$ :: 9 - Addr. Lines

$\frac{7}{8} = 128$ 7 - A.L

(9.18) ... address buses.

→ The lower-order lines in the address bus select the byte within a particular chip and other lines in the address bus select a particular chip through its chip select i/p's.

→ Each RAM receives 7 lower-order bits of the address bus to select one of 128 possible bytes.

→ A particular RAM is selected from lines 8 and 9 in the address bus. This is done through a 2x4 decoder whose outputs go to CS1 inputs in each RAM chip.

8 9
0 0 — RAM 1
0 1 — RAM 2
1 0 — RAM 3
1 1 — RAM 4

→ RD and WR o/p's from the microprocessor are applied to i/p's of each RAM chip.

→ Selection between RAM and ROM is achieved through bus line 10. RAMs are selected when bit line is 0 and ROM is selected when 1.

→ other chip select input in the ROM is connected to RD control line for the ROM chip to be enabled only during a read control operation

→ Address bus lines 1 to 9 are applied to the i/p address of ROM without going through the decoder. This assigns addresses 0 to 511 to RAM and 512 to 1023 to ROM.

→ The data bus of the ROM has only an o/p capability, whereas the data bus connected to the RAMs can transfer information in both directions.

(P6) A computer employs RAM chips of 256 x 8 and ROM chips of 1024 x 8. The computer system needs 2K bytes of RAM, 4K bytes of ROM and four interface units, each of 4 registers. A memory-mapped I/O configuration is used. The two highest-order bits of the address bus are assigned 00 to RAM, 01 to ROM and 10 to interface units.

a) How many RAM and ROM chips are needed?

b) Draw a memory-address map for the system.

c) Give the address range in hexadecimal for RAM, ROM and interface.

Sol: a) Size of RAM chips = 256 x 8.

Required RAM capacity = 2K x 8

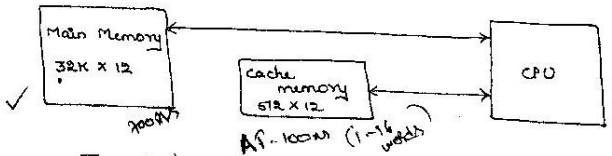
⇒ No. of RAM chips = $\frac{2K \times 8}{2 \times 1024}$ = 4

Cache memory:-

When a program loop is executed, the CPU repeatedly refers to the set of instructions in memory that constitute the loop. Everytime a given subroutine is called, its set of instructions are fetched from memory. Thus loops and subroutines tend to localize the references to memory for fetching instructions.

The references to memory at any given interval of time tend to be confined within a few localized areas in memory. This phenomenon is known as property of **Locality of Reference**.

If the active portions of the program and data are placed in a fast small memory, the average access time can be reduced, thus reducing the total execution time of the program. Such a fast and small memory is called **Cache Memory**. It is placed between CPU and main memory as shown below.



The cache access time is less than the access time of main memory by a factor of 5 to 10. The cache is the fastest component in the memory hierarchy and approaches the speed of CPU.

The fundamental idea of cache organization is that by keeping the most frequently accessed instructions and data in the fast cache memory, the average memory access time will approach the access time of the cache.

Although cache is only a small fraction of the size of main memory, a large fraction of memory requests will be found in the fast cache memory because of the locality of reference property of programs.

Basic operation of cache: When the CPU needs to access memory, the cache is examined. If the word is found in the cache, it is read from the fast memory. If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word.

transferred from main memory to cache memory. The block size may vary from 1 word to 16 words.

Performance of cache memory is frequently measured in terms of a quantity called **hit ratio**. When the CPU refers to memory and finds the word in cache, it is called **hit**. If word is not found in cache, it is in main memory and it counts a **miss**.

$$\text{Hit Ratio} = \frac{\text{No. of hits}}{\text{Total CPU references to main memory}}$$

$$= \frac{\text{No. of hits}}{\text{No. of hits} + \text{No. of misses}}$$

Hit ratio of 0.9 is experimentally seen. This hit ratio verifies the validity of the locality of reference property.

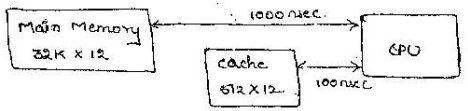
(Pb) The access time of a cache memory is 100nsec and that of main memory 1000nsec. It is estimated that 80% of memory requests are for read and the remaining 20% for write. The hit ratio for read accesses only is 0.9. A write through procedure is used.

a) what is the average access time of the system considering only memory read cycles?

b) what is the average access time of the system considering both read & write requests?

c) what is the hit ratio taking into consideration the write cycles?

Sol a)



Hit ratio of read accesses = 0.9.

⇒ probability of hit = 0.9, probability of miss = 0.1.

If hit, data is accessed from cache ⇒ 100 nsec

If ~~hit~~ miss, data is accessed from main memory after checking cache ⇒ 100 nsec + 1000 nsec = 1100 nsec.

Average access time for read accesses = 0.9 × 100 nsec + 0.1 × 1100 nsec

= 200 nsec

= % of read cycles × Avg. access time for read cycles
+ % of write cycles × Avg access time of write cycles

= 0.8 × 200nsec + 0.2 × 1000nsec

since write-Through procedure updates both main memory & cache simultaneously ∴ Avg access time = 1000nsec

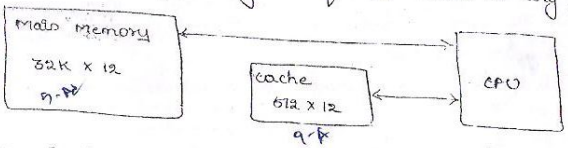
c) Hit ratio considering write cycles = 0.8 × 0.9 = 0.72

The transformation of data from main memory to cache memory is referred to as a **mapping procedure**. Three types of mapping procedures are of practical interest when considering the organization of cache memory:

1. Associative mapping.
2. Direct mapping
3. Set-Associative Mapping.

Consider, the main memory can store 32K words of 12 bits each. The cache is capable of storing 512 of these words at any given time.

32K = $2^5 \times 2^{10}$
Address = 15-bits

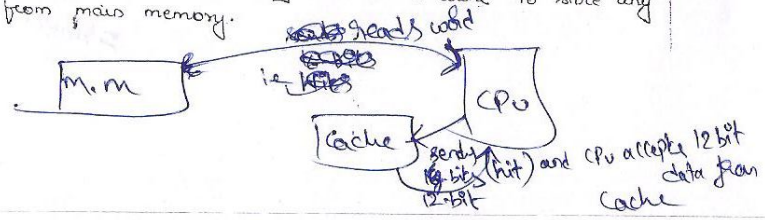


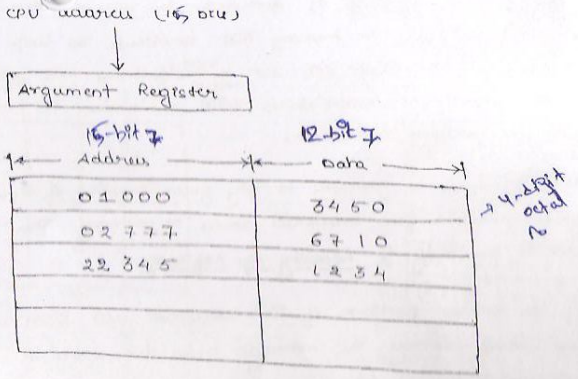
CPU communicates with both memories. It first sends a 15-bit address to cache. If there is a hit, the CPU accepts 12-bit data from cache. If there is a miss, the CPU reads the word from main memory and the word is then transferred to cache.

a) Associative Mapping.

The fastest and most flexible cache organization uses an associative memory.

Associative memory stores both address and data of the memory word. This permits any location in cache to store any word from main memory.





The 15-bit address is shown as 5 digit octal number and the 12-bit data is shown as 4 digit octal number.

A CPU address of 15 bits is placed in the argument register and the associative memory is searched for a matching address.

1) If address is found, the corresponding 12-bit data is read and sent to CPU.
2) If no match occurs, the main memory is accessed for the word. The address-data pair is then transferred to the cache.
3) If cache is full, an address-data pair must be displaced to make room for a pair that is needed and not presently in cache. The decision as to what pair is replaced depends on the replacement algorithm.

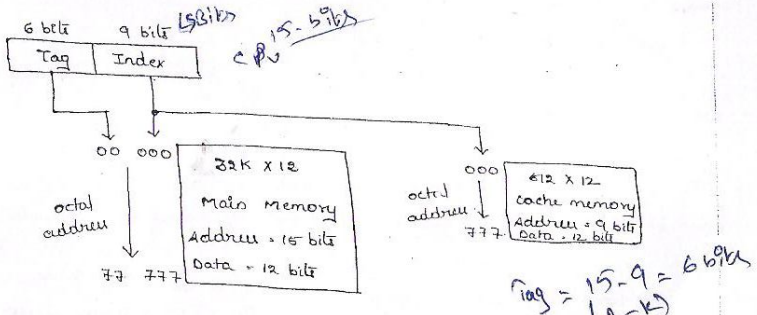
In general, FIFO or LRU replacement algorithms are used.

Disadv:- Associative memories are expensive compared to random-access memories because of the added logic associated with each cell.

Random-access memories are used in
1) Direct Mapping
2) Set-Associative Mapping

Direct mapping

The addressing relationships between main memory and cache in direct mapping is shown below



tag = 15 - 9 = 6 bits
(n - k)

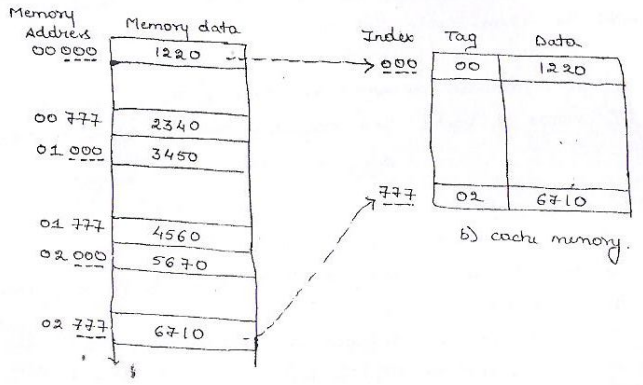
The CPU address of 15 bits is divided into 2 fields. The least significant 9 bits constitute the index field and the remaining 6 bits form the tag field

No. of bits in the index field = No. of address bits required to access cache memory.

In general, There are 2^k words in cache and 2^n words in main memory. The n-bit memory address is divided into 2 fields: k bits for index field and (n-k) bits for the tag field.

The direct mapping uses n-bit address to access main memory and k-bit address to access cache memory.

The internal organization of words in the cache is shown below:



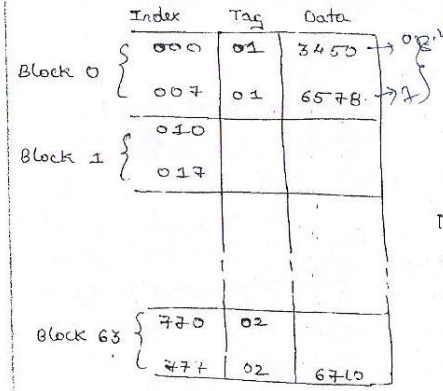
Each word in the cache consists of the data word and its associated tag. When a new word is first brought into the cache, the tag bits are stored alongside the data bits. When the CPU generates a memory request, the index field is used for the address to access the cache.

The tag field of the CPU address is compared with the tag in the word read from the cache. If the two tags match, there is a hit and the desired data word is in cache. If there is no match, there is a miss and the required word is read from main memory. It is then stored in the cache together with the new tag, replacing the previous value.

Disadv:- We cannot store two words with same index and different tag values. ⇒ Hit ratio will fall if two words with same index and different tag value are accessed repeatedly.

However, this possibility is minimized by the fact that such words are relatively far apart in the address range (multiples of 512 in our example) based on "locality of reference".

Direct mapping shown above uses a block size of 1 word. The same organization but block size of 8 words is shown below.



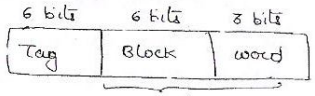
Tag field stored in the cache is common for all 8 words in the same block.

The index field is now divided into 2 parts:

1) block field
2) word field

In a 512-word cache, There are 64 blocks ($\frac{512}{8} = 64$) of 8 words each.

⇒ block field = 6 bits
word field = 3 bits



memory of 1K words. The cache uses direct mapping with block size of 4 words.

a) How many bits are there in the tag, index, block and word fields of the address format?

b) How many bits are there in the each word of cache, and how are they divided into functions? Include a valid bit.

c) How many blocks can the cache accommodate.

Sol: Size of main memory = 64K × 16. = $2^{16} \times 16$.

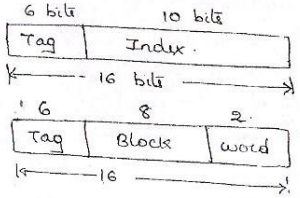
⇒ No. of address bits = 16 bits

Size of cache = 1K words = 1024 words.

⇒ No. of address bits = 10 bits

Block size = 4 words.

a)



Index = no. of bits in cache address.

Tag = 16 − Index.

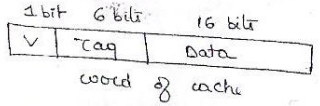
Block = 4 words = 2^2.

⇒ word field = 2 bits

Block field = Index − Bword.

= 10 − 2 = 8 bits

b)



word of cache

word of cache = 1 + 6 + 16

= 23 bits

Valid bit is used when working with write-back policy.

c) Cache can accommodate 2^8 blocks as block field = 8 bits

= 256 blocks of 4 words each.

c) Set-Associative Mapping

Disadvantage of direct mapping is that two words with the same index in their address but with different tag values cannot reside in cache memory at the same time.

Set-associative mapping is an improvement over the direct-mapping organization in that each word of cache can store two or more words of memory under the same index addr.

(9.28)

Index	Tag	Data	Tag	Data
000	01	3450	02	6670
777	02	6710	00	2340

Two-way set-associative mapping cache.

A two-way set-associative mapping accommodates 2 words of main memory in each word of cache.

** In general, a set-associative cache of "set size K" will accommodate "K" words of main memory in each word of cache.

In above example, the words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000. Illy, words at addresses 02777 and 00777 are stored in cache at index address 777.

When CPU generates a memory request, the index value of the address is used to access the cache. The tag field of the CPU address is then compared with both tags in the cache to determine if a match occurs. The comparison logic is done by an associative search of the tags in the set similar to an associative memory search: ⇒ The name "set-associative"

Hit ratio increases as set-size increases because more words with the same index but different tags can reside in cache.

However, increase in set size increases the number of bits in the words of cache and requires more complex comparison logic.

Replacement Algorithms

When a miss occurs in a set-associative cache and the set is full, it is necessary to replace 1 of the tag-data items with a new value.

The most common replacement algorithms are

i) Random Replacement

ii) LRU (least Recently used

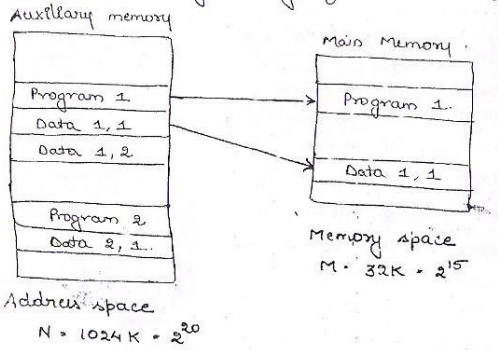
cache is not replaced by another word unless the valid bit is set to 1 and a mismatch of tags occurs. If valid bit happens to be 0, the new word automatically replaces the invalid data.

* Virtual memory

Virtual memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space is available, equal to the totality of auxiliary memory.

Virtual memory is used to give programmers the illusion that they have a very large memory at their disposal, even though the computer actually has a relatively small main memory.

Consider a system with a main memory capacity of 32K words (2^{15} words) and auxiliary memory of 1024K words (2^{20}).

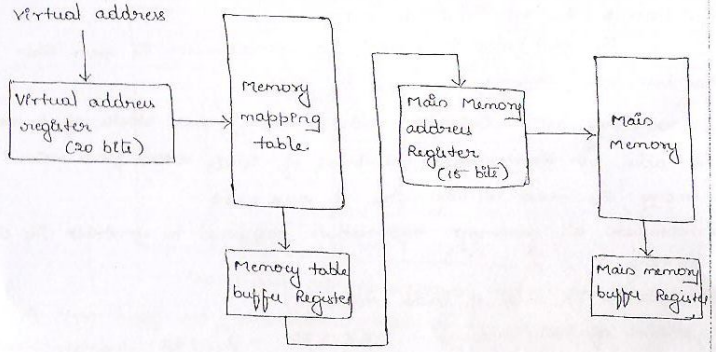


Address in auxiliary memory is called **virtual address**. An address used by the programmer is virtual address and set of all virtual addresses is called **Address space**.

An address in main memory is called **Physical address**. The set of all physical addresses is called **Memory space**.

The address space is allowed to be larger than memory space in computers with virtual memory.

Each address referenced by the CPU (virtual address) goes through an address mapping from the so-called virtual address to a physical address in main memory. A table is thus needed to map a virtual address of 20 bits to a physical address of 15 bits as shown below

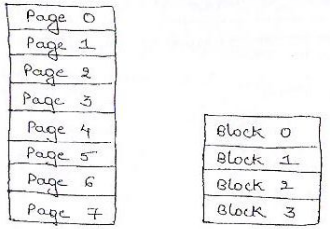


The mapping table may be stored in a seperate memory or in main memory. If stored in a seperate memory, an additional memory unit is required as well as one extra memory access time. If stored in main memory, the table takes space from main memory and two accesses to memory are required with the program running at half speed. A third alternative is to use associative memory.

The address space is broken down into groups of equal size called **Pages** and the physical memory or memory space is broken into groups of equal size called **Blocks**. The size of page & block are same.

Portions of programs are moved from auxiliary memory to main memory in records equal to size of a page.

Consider a system with memory space of 4K and address space of 8K. If we split each into groups of 1K words, we get 8 pages and 4 blocks as shown below



Address space
N = 8K = 2^{13}

Memory space
M = 4K = 2^{12}

At any point of time, 4 pages of address space can reside in main memory in any 1 of the 4 blocks.

facilitated if each word virtual address is considered to be represented by 2 numbers:

- a page number address
- a line number within a page.

In a computer with 2^p words per page, p-bits are used to specify a line number and remaining higher-order bits of the virtual address specify the page number.

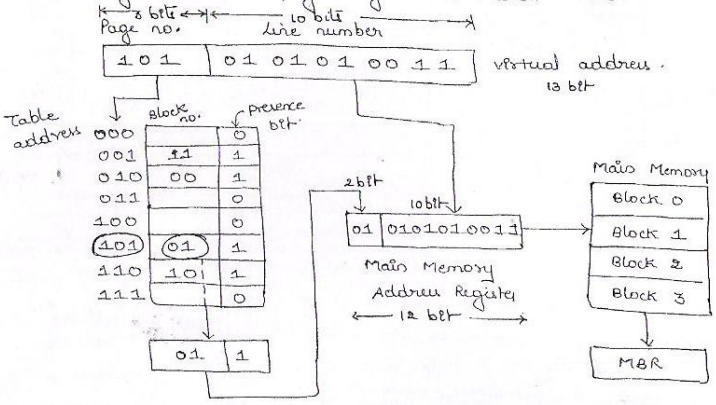
Eg:- N = 8k. $= 2^{13}$.

page size = 1024 words $= 2^{10}$.

$\Rightarrow$ line no = 10 bits

page no = 13 − 10 = 3 bits to specify 1 of the 8 pages.

Memory table in a paged system is shown below.



Memory page table.

The memory page table consists of 8 words, one for each page. The address in the page table denotes the page number and the content of the word gives the block number where the page is stored in main memory.

The table shows that pages 1, 2, 5 and 6 are now available in main memory blocks 3, 0, 1 and 2 respectively.

A presence bit in each location indicates whether the page has been transferred from auxiliary memory into main memory. A "0" in the presence bit indicates that this page is not available in main memory.

the CPU references a word in memory. It is a

Virtual address of 13 bits. The 3 higher-order bits of the virtual memory address specify the page no. and also an address for memory-page table.

The content of the word in the memory-page table at the page number address is read out into memory table buffer register. If **presence bit = "1"**, the block no. thus read is transferred to the two higher-order bits of the main memory address register. The line no. from the virtual address is transferred into the lower-order bits of the memory address register. A read signal to main memory thus transfers the content of the word in main memory to main memory buffer register which is then used by CPU.

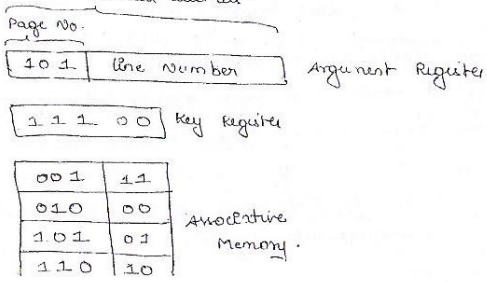
If **presence bit = "0"**, it signifies that the content of the word referenced by the virtual address does not reside in main memory. A call to OS is then generated to fetch word from auxiliary memory and place it into main memory.

Disadv: A random access memory page table is inefficient wrt storage utilization. Here, we need 8 words of memory, 1 for each page, but atleast 4 words will always be marked empty because main memory cannot accommodate more than 4 blocks.

In general, a system with "n" pages and "m" blocks require a memory-page table of "n" locations of which upto "m" locations will be marked with block numbers and all others will be empty.

Sol:- A more efficient way is to organize the page table in the construct with no. of words equal to no. of blocks in main memory. In this size of memory is reduced & it is fully utilized.

This is implemented using **associative memory** - with each word in the memory containing a page no. together with its corresponding block number.



Page replacement

A virtual memory system is a combination of hardware and software techniques. The memory management software system handles all the software operations for the efficient utilization of memory space.

It must decide

1. which page in main memory ought to be removed to make room for a new page.
2. when a new page is to be transferred from auxiliary memory to main memory.
3. where the page is to be placed in main memory.

When a program starts execution, 1 or more pages are transferred into main memory and the page table is set to indicate their positions. The program is executed from main memory until it attempts to reference a page that is still in auxiliary memory. This condition is called **page fault**.

When a page fault occurs in a virtual memory, it signifies that the page referenced by CPU is not in main memory. A new page is then transferred from auxiliary memory to main memory. If main memory is full, a page is removed from main memory to make room for the new page.

The policy for choosing pages to remove is determined from the **replacement algorithm**. The goal of a replacement algorithm is to try to remove the page least likely to be referred in the immediate future.

Two common replacement policies are

1. FIFO (first in first out)
2. LRU (least Recently used).

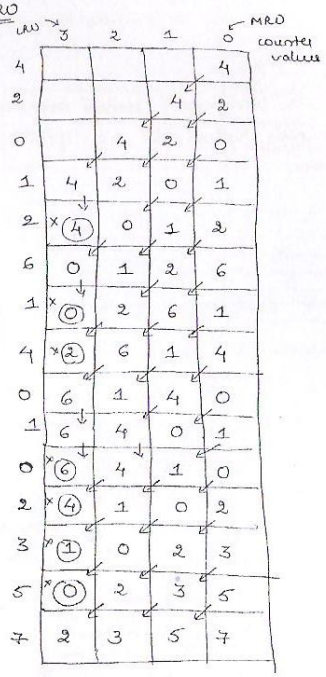
(Pb) A virtual memory system has an address space of 8K words, a memory space of 4K words, and page + block size of 1K words. The following page references occur during a given interval of time.

4 2 0 1 2 6 1 4 0 1 0 2 3 5 7

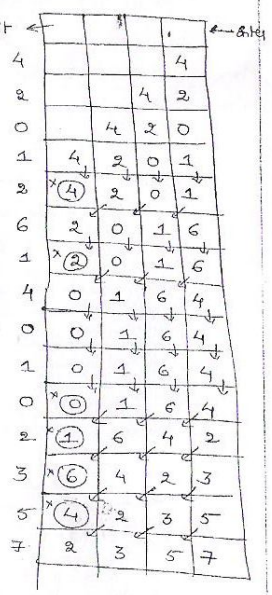
Determine the four pages that reside in main memory after each page reference if the replacement algorithm used is a) FIFO b) LRU.

Page ref: 4 2 0 1 2 6 1 4 0 1 0 2 3 5 7

a) LRU



b) FIFO



(Pb) A virtual memory has a page size of 1K words. There are 8 pages and 4 blocks. The associative memory page table contains the following entries.

Page	Block
0	3
1	2
4	1
6	0

Make a list of all virtual address in decimal that will cause a page fault if used by CPU.

Sol: Address range

available to main memory
- 0 — 0 to 1023 (1K-1)
- 1 — 1024 (1K) to 2047 (2K-1)
- 2 — 2K to (3K-1)
- 3 — 3K to (4K-1)
- 4 — 4K to (5K-1)
- 5 — 5K to (6K-1)
- 6 — 6K to (7K-1)
- 7 — 7K to (8K-1)

These pages are not available in main memory ⇒ a page fault occurs if these addresses are accessed by CPU.

UNIT - 4

Input - output Organization & Memory System

1) Input - output organization:-
   - ↳ Peripheral Devices → ASCII alphanumeric characters, Control functions, I/O interface, Isolated vs memory mapping
   - ↳ Asynchronous Data transfer.
   - ↳ Modes of transfer
   - ↳ Priority interrupt
   - ↳ Direct memory Access (DMA)
   - ↳ Serial Communication.
   - ↳ Input - output Processor.
2) Memory System:-
   - → Memory hierarchy
   - → main memory
   - → Auxiliary memory
   - → Associative memory
   - → Cache memory
   - → Virtual memory
   - → Memory management hardware.

1) Input - output organization:-

(a) Peripheral Devices:- I/O devices attached to Computer are called Peripherals * Peripherals like keyboards, display units, pointers, magnetic disks; (ex) Hard disks magnetic tapes.

(i) ASCII alphanumeric characters:-
(American Standard Code for Information Interchange) it uses 7-bits Code. 128 - characters.
($b_1 - b_7$)
(as shown in Table) //

→ out of 128 - characters the ASCII Code Contain 94 - characters Can be printed and 34 non printing characters used for various Control functions.