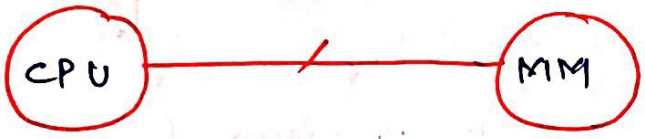
Multi-level caches :-

- Multi-level cache structure is used in the structure design to reduce the miss-penality.

- Miss-penality means time required to transfer the data from higher level to lower level when there is a miss in lower-level memory.

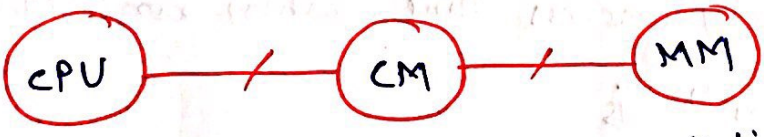
(a) Sys design without cache -



10 MR

10 times MM Accessed

(b) Sys design with cache -

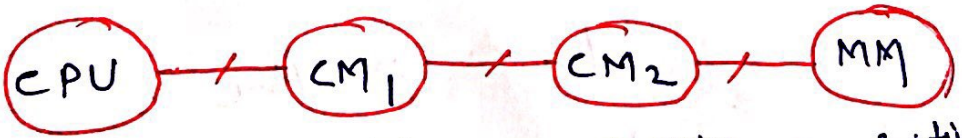


10MR

4-miss

4 times MM Accessed

(c) Sys design with multi-level caches -



10 MR

4 miss

2-miss

2 times MM accessed

- In multi-level cache design -

L_1 CM size < L_2 CM size

L_1 CM Access time < L_2 CM Access time

L_1 CM Association to CPU > L_2 CM Association to CPU

- Two kinds of miss rates are calculated in the multi-level cache design.

(i) GMR (Global miss rate) = $\frac{\text{no. of misses in cache}}{\text{Total no. of CPU generated refrences}}$

(ii) LMR (Local miss rate) = $\frac{\text{no. of misses in cache}}{\text{Total no. of access in the cache}}$

$GMR_{L_1} = \frac{4}{10} = 0.4$

$GMR_{L_2} = \frac{2}{10} = 0.2$

$LMR_{L_1} = \frac{4}{10} = 0.4$

$LMR_{L_2} = \frac{2}{4} = 0.5$

- Avg. memory access time is calculated in terms of hit time, miss rate (local) and miss penality is :

$$T_{avg} = \text{Hit Time}_{L_1} + (\text{miss-Rate}_{L_1} * \text{miss-penality}_{L_1})$$

$$\text{Miss-penality}_{L_1} = \text{Hit Time}_{L_2} + (\text{miss-Rate}_{L_2} * \text{miss-penality}_{L_2})$$

$$\text{Miss-penality}_{L_2} = \text{memory (MM) access time.}$$

- Avg. memory stalls created per inst^n is calculated as:

$$= \left(\text{no. of misses in } L_1 / \text{Inst}^n * \text{Hit-time } L_2\right) +$$

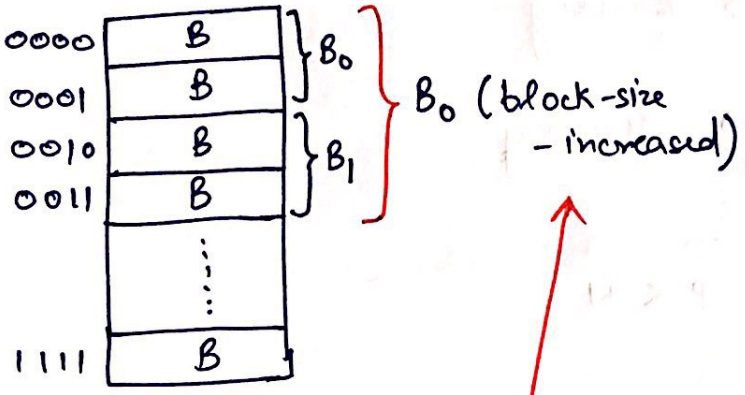
$$\left(\text{no. of misses in } L_2/\text{inst}^n * \underbrace{\text{MM access time}}_{\text{Miss-penality } L_2}\right)$$

Types of Cache-misses :-

(i) **compulsory-miss** :- (Cold-start miss or first reference miss)

This miss will occurs when the very first-time reference block is not present in the cache memory during the program execution. This misses can be minimized by increasing the block-size.

eg:-



To minimize the misses, block size (2B → 4B)

Prog:-

I_1: MOV r_0, [0000] (B_0);

I_2: MOV r_1, [0010] (B_1);

Exec:-

I_1: B_0 (miss)

I_2: B_1 (miss)

compulsory miss

New Exec:-

I_1 : B_0 (miss)

I_2 : B_0 (Hit)

* Now [0000] and [0010] both lie in B_0, block.

(ii) Capacity-miss :- This miss will occurred when the cache is full. These misses can be minimized by increasing the cache capacity.

(iii) Conflict-miss :-

(collision-miss or interference miss)

This miss will occurred when too many blocks are mapped into a same cache-line or cache-set. These misses can be minimized by double the associativity of the cache i.e

1-way → 2-way

2-way → 4-way

⋮

$1 < P < N$

Q. Consider cache-memory with a access time of 50 ns uses write-through protocol. MM access time is 200 ns. CPU generates 60% READ requests and remaining are used for WRITE op^n. $H_r = 80\%$. what is avg. memory access time when considering both or. READ & WRITE op^n.

Given :-

WT Protocal

$T_c = 50ns$

$T_m = 200ns$

$f_r = 60\%$

$f_w = 40\%$

$H_r = 80\%$

H_w = not given (Assume 1)

so $H_w = 1$

this means simultaneous access mem.

$$Tavg_r = H_r T_c + (1-H_r)(T_m + T_c)$$

$$= 0.8(50) + 0.2(50 + 200)$$

$$= 40 + 50$$

$$= 80ns$$

$$Tavg_w = T_m = 200ns$$

$$Tavg_{WT} = f_r \cdot Tavg_r + f_w \cdot Tavg_w$$

$$= 0.6 * 80 + 0.4 * 200$$

$$= 128ns$$

Q: Consider cache memory having the hit ratio for read & write op^n is 70% & 80% resp. CM access time is 40ns and MM access time is 400ns. When there is a miss in ~~cache~~ CM then 2 word block is copied from the MM to CM. CPU gnerates 40% READ req and remaining are used for WRITE op^n. What is the efficiency of a memory in terms of ~~MEANS~~ MWPS (million words per sec) using -

(a) write_through protocal

(b) Write-back protocal.

Given:-

$H_r = 0.7$

$H_w = 0.8$

$T_c = 40ns$

$T_m = 400ns$

Block-size = 2W

$f_r = 0.4$ (% of clean-bits)

$f_w = 0.6$ (% of dirty-bits)

since $H_w \neq 1$

↓

so, memory is hiiramical access

$$T_w \text{ (simultaneous - WRITE)} = \text{Max}\left(\text{word updation in CM}\ /\ \text{word updatin in MM}\right)$$

$$= \text{Max}(40ns\ /\ 200ns)$$

T_c / since 2W Block is transfered in T_m so for 1W → $T_m/2$

$$= 200ns$$

(a) $Tavg_r = H_r T_c + (1-H_r)(T_m + T_c)$

$= 0.7(40) + 0.3(40 + 400)$

$= 28 + 132$

$= 160 \text{ ns}$

$Tavg_w = H_w T_w + (1-H_w)(T_m + T_w)$

$= 0.8(200) + 0.2(400 + 200)$

$= 160 + 120$

$= 280 \text{ ns}$

$Tavg_{WT} = (f_r * Tavg_r) + (f_w * Tavg_w)$

$= (0.4 * 160) + (0.6 * 280)$

$= 232 \text{ ns}$

$\left(\eta_{WT} = \frac{1}{Tavg_{WT}} \text{ words/sec}\right) = 4.31 \text{ MWPS}$

(b) $Tavg_r = H_r T_c + (1-H_r)\left[\%_{\text{dirty bits}}(T_m + T_m + T_c) + \%_{\text{clean bits}}(T_m + T_c)\right]$

$= (0.7 * 40) + (1-0.7)\left[0.6(400+400+40) + 0.4(400+40)\right] = 232 \text{ ns}$

=

$$Tavg_{w} = HwTc + (1-Hw)\left[\%\ \text{dirty-bits}\ (Tm+Tm+Tc) + \%\ \text{clean bits}\ (Tm+Tc)\right]$$

$$= 0.8(40) + (1-0.8)\left[0.6(400+400+40) + 0.4(400+40)\right]$$

$$= \underline{168ns}$$

$$Tavg_{wB} = (f_r * Tavg_{wr}) + (f_w * Tavg_{w})$$

$$= (0.4 * 232) + (0.6 * 168)$$

$$= \underline{193.6ns}$$

$$\eta_{wB} = \frac{1}{Tavg_{wB}} = \boxed{5.16\ MWPS}$$

Q. Consider the following memory specification:

Mem	Access time	Hit-Ratio
(Inst^n) I-cache	20ns	0.6
(Data) D-cache	40ns	0.8
MM	200ns	0.9
SM	600ns	1

CPU generates 60% req to access the inst^n and 40% req to access the data. what is the total avg-memory access time when considering inst^n & data accesses.

$$T_{avg_{IR}} = H_{Ic}\,T_{Ic} + (1-H_{Ic})\,H_m\,(T_m + T_{Ic}) + (1-H_{Ic})\,(1-H_m)\,H_s\,(T_s + T_m + T_{Ic})$$

Inst^n READ

$$= (0.6 \times 20) + (0.4 \times 0.9 \times 220) + (0.4 \times 0.1 \times 820)$$

$$= \underline{124\text{ ns}}$$

$$T_{avg_{DR}} = H_{Dc}\,T_{Dc} + (1-H_{Dc})\,H_m\,(T_m + T_{Dc}) + (1-H_{Dc})\,(1-H_m)\,H_s\,(T_s + T_m + T_{Dc})$$

Data - READ

$$= (0.8 \times 40) + (0.2 \times 0.9 \times 240) + (0.2 \times 0.1 \times 840)$$

$$= \underline{92\text{ ns}}$$

$$T_{avg} = \left(f_{Inst^n} * T_{avg_{IR}}\right) + \left(f_{Data} * T_{avg_{DR}}\right)$$

$$= \boxed{111.2\text{ ns}}$$

Q. Consider computer system in which CPU generates 200 memory refrences (MR's), among them 60 miss op^n present in L1 cache & and 30 - miss op^n present in L2 cache. Hit - time of L1 memory is 4-cycles & Hit - time of L_2 - cache is 20-cycles. Miss penality of L_2 memory is 100 - cycles. What is the avg. memory access time.

$$LMR_{L_1} = \frac{60}{200} \; ; \qquad LMR_{L_2} = \frac{30}{60}$$

$$T_{avg} = \text{Hit Time}_{L_1} + (\text{miss-rate } L_1 * \text{miss-penality}_{L_1})$$

$$\text{miss-penality}_{L_1} = \text{Hit-Time}_{L_2} + (\text{miss-rate } L_2 * \text{miss-penality}_{L_2})$$

$$= 20_{\text{cycly}} + \left(\frac{30}{60} * 100\right)$$

$$= \underline{70 \text{ cycles}}$$

Memory access time

So, $$T_{avg} = 4 \text{ cycles} + \left(\frac{60}{200} * \left(20 + \left(\frac{30}{60} * 100\right)\right)\right)$$

$$= \boxed{25\text{-cycles}}$$

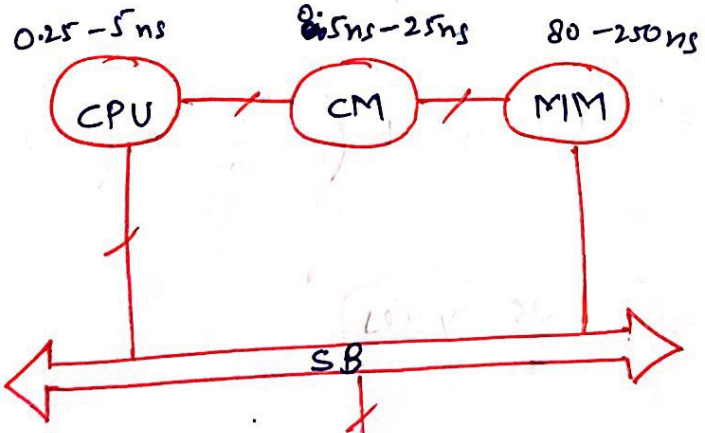
Input - Output (I/O) organization :-

- I/O devices are electro-magnetic component & CPU is an electronic component, so there is a difference existed in terms of operating modes, data-transfer rate and word formats.

- To synchronize the I/O properties with a CPU, high-speed interface chip is used called as I/O interface or I/O module.

- I/O interface chip is responsible for I/O op^n. ∴ in the computer design all the I/O devices are interfaced to CPU & via I/O interface chip.

(a) Sys design without I/O Interface logic -

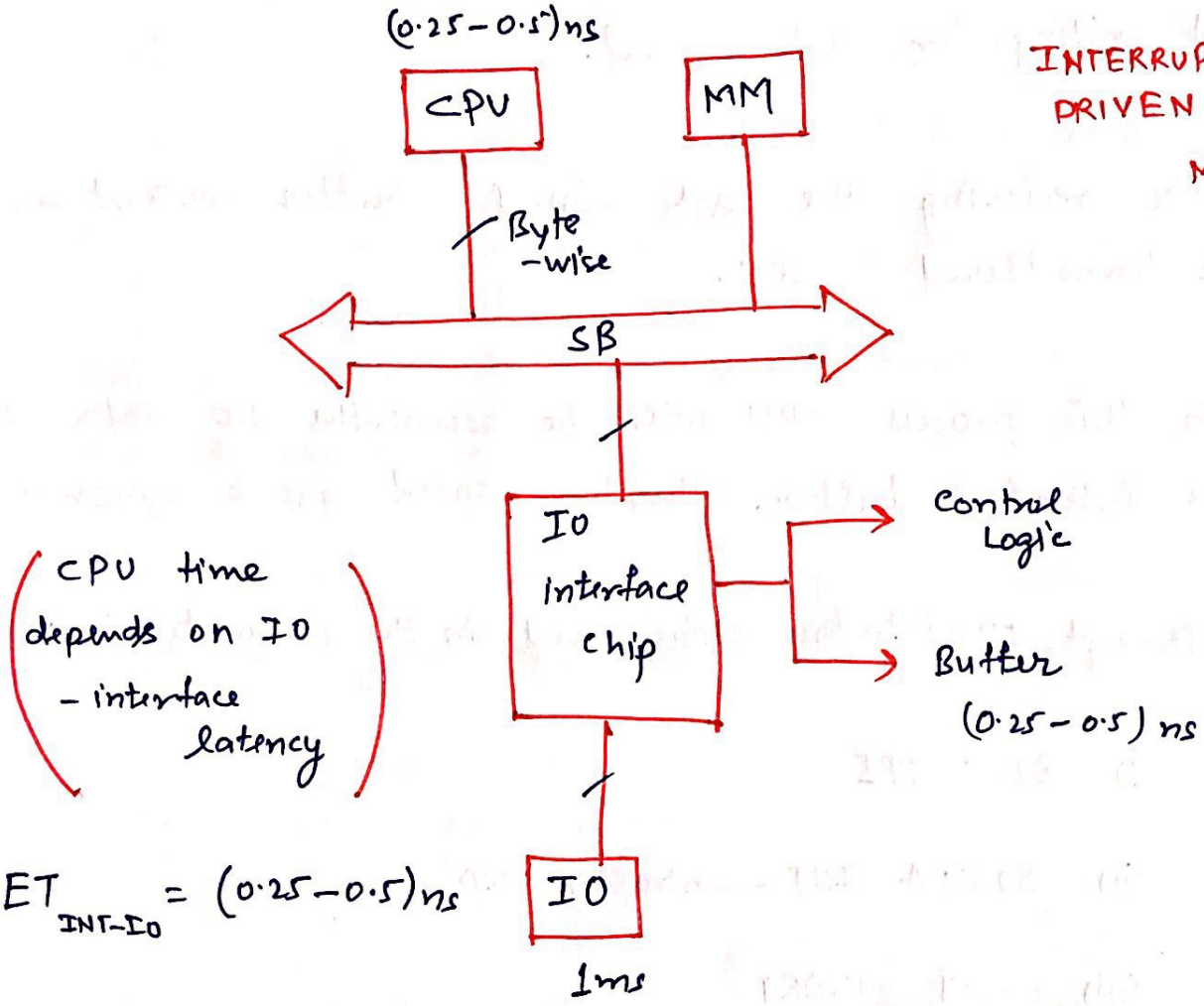


(PROGRAMMED - IO MODE)

(CPU time depends on I/O speed & data size)

1 KBps (1 ms)

(b) Sys design with I/O - interface chip logic



$ET_{INT-IO} = (0.25-0.5)ns$

Access Sequence :-

- CPU initializes the IO interface chip along with IO command later busy with other useful task.
- IO-interface control logic interprets the IO command and enables the IO opⁿ.
- Based on the speed of a IO-device consume the time to prepare the data, later transfer data to Interface-buffer.

- when the data is available in buffer then IO - interface chip to enables the interrupt signal to CPU and waiting for ack signal.

- After receiving the ack signal - buffer content will be transferred to CPU.

- In this process CPU will be accessing the IO - data from the interface buffer. Therefore speed - gap is syncronized.

- Different IO - interface chips used in the computer is :

  (i) 8255 PPI

  (ii) 8259A INT - controller chip

  (iii) 8251 USART

  (iv) 8257/8237 DMA

IO - modes :-

3 - different IO - transfer modes are used in the system design to transfer ~~design~~ data -

(i) Programmed - IO

(ii) Interrupt - driven IO

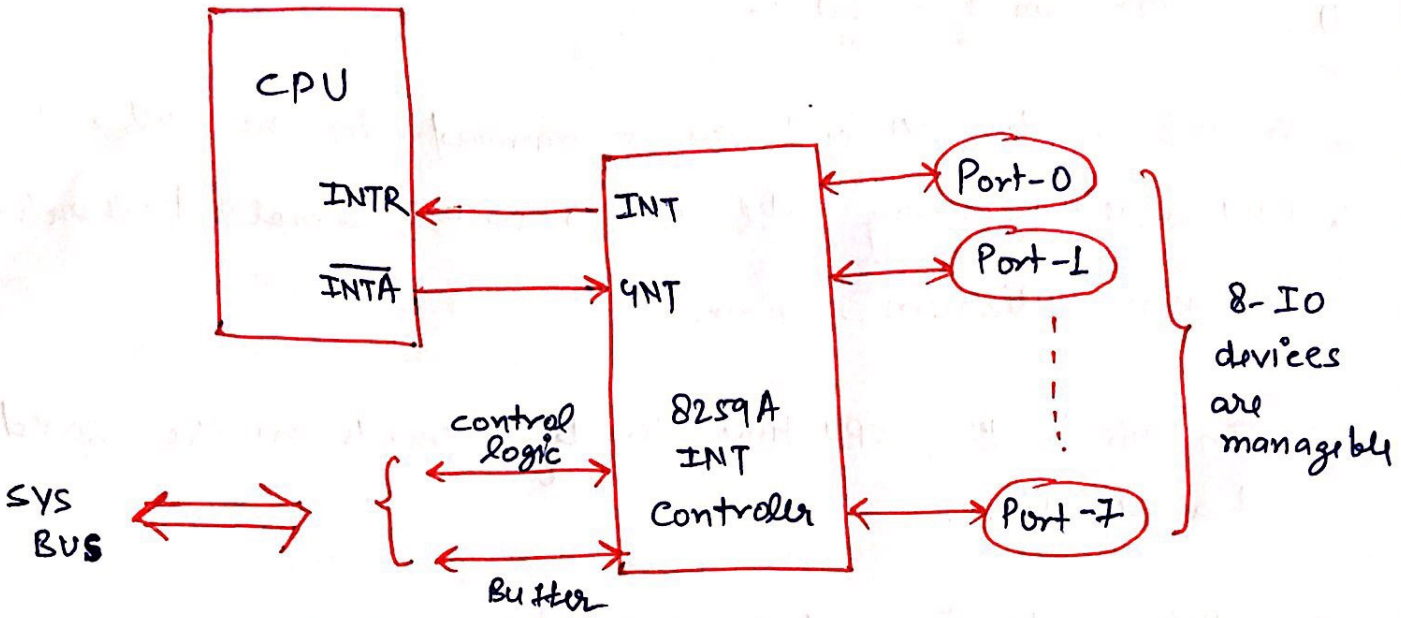
(iii) DMA

(i) Programmed – IO :-

- In this mode IO op^n are programmed in the CPU, so CPU will be waiting until the IO op^n is completed therefore processor utilization is poor.
- In this mode, CPU time directly depends on the speed of a IO device.
- This mode is used in the system design where application goals are important than the CPU time.

(ii) Interrupt – driven IO :-

- In this mode IO op^n are controlled by based on interrupt signals.
- In this mode, IO interface chip is responsible for IO op^n so CPU time is utilized for other useful task. Therefore processor utilization is efficient.
- In this mode CPU time depends on the latency of a IO interface rather than the speed of a IO device.

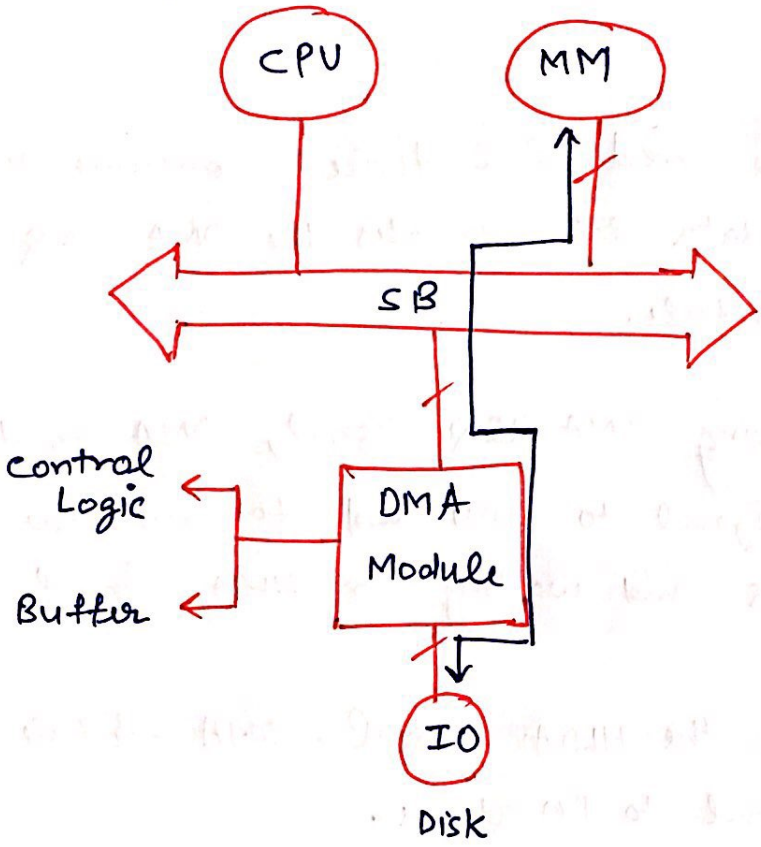


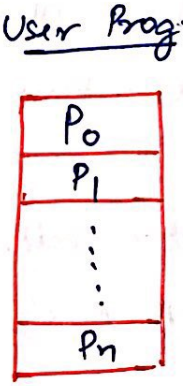
NOTE:- 8259 A chip is programmed into a cascading mode to manage more than 8 IO devices (max is 64 devices)

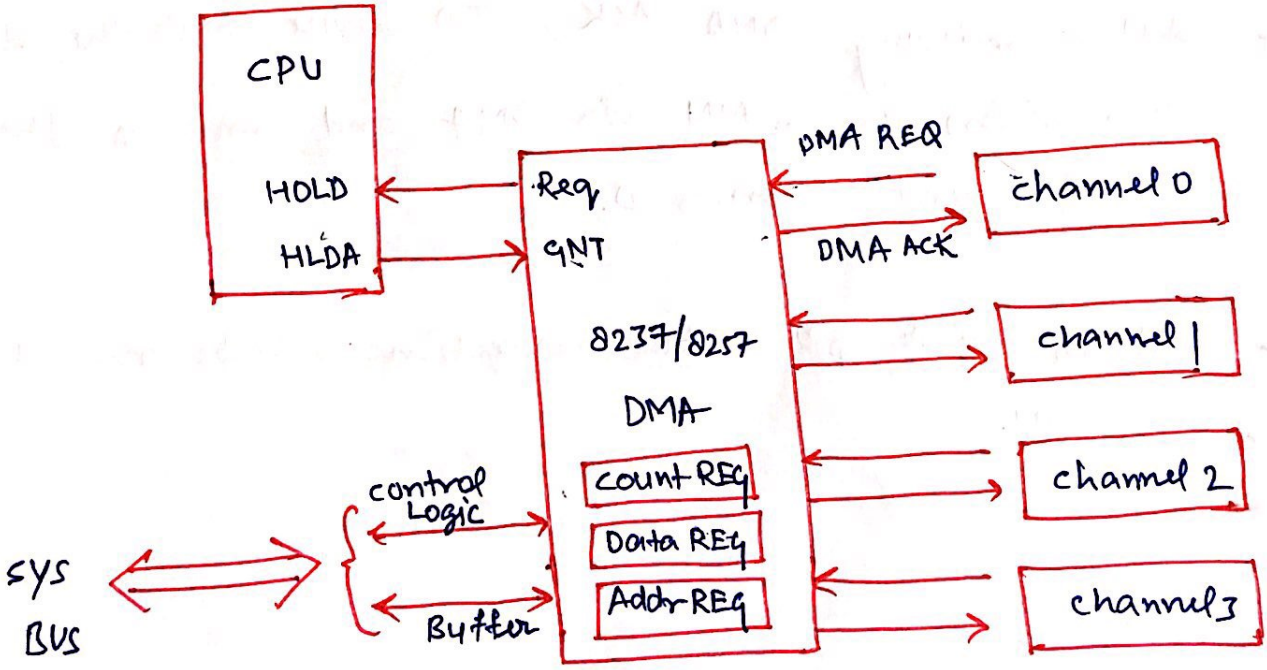
(iii) Direct-Memory Access DMA :-

- In this mode bulk amount of data will be transferred from IO to main memory without involvement of CPU.
- when the program size is greater than MM size then virtual memory concept is used to increase the address space. This concept states that use the secondary storage space to store the user program.
- Secondary storage is an electromagnetic storage so it is classified under the IO-category.

- In the computer design, secondary storage components are interface to system Bus via DMA module, therefore during the execution of a program, data will be transferred from IO to main memory via DMA without involvement of a CPU.







- CPU initializes the DMA module along with a IO command later busy with other task.

- IO command contains port address, mem addr, control signals and count value.

- DMA control logic interprets the command & enables the IO port.

- Based on the speed of a device, consume the time to prepare the data later enables the DMA REQ signal to DMA module.

- After recieving DMA REQ signal, DMA module enables the HOLD signal to CPU and to gain the control of a system BUS and waiting for HLDA signal.

- After recieving the HLDA signal, DMA module enables the DMA ACK signal to IO device.

- After recieving DMA ACK, IO device starts the data transmission to MM via DMA and continue the opⁿ until count becomes 0.

- After DMA opⁿ, Bus connection will be re-established to CPU.

NOTE:- In the DMA opⁿ CPU presents in two states -

(i) Busy state

(ii) Blocked or Hold state

- CPU is in Busy state until the IO device prepares the data.
- CPU is in Blocked state until the data is transferred to MM.
- Let x is a preparation time and y is transfer time then -

$$\%\ \text{time}_{\text{CPU busy}} = \left(\frac{X}{x+y}\right) \times 100$$

$$\%\ \text{time}_{\text{CPU blocked}} = \left(\frac{Y}{x+y}\right) \times 100$$

- DMA module is operating under 3 modes -

(i) Burst mode

(ii) Cycle-stealing mode

(iii) Inter-leaving mode

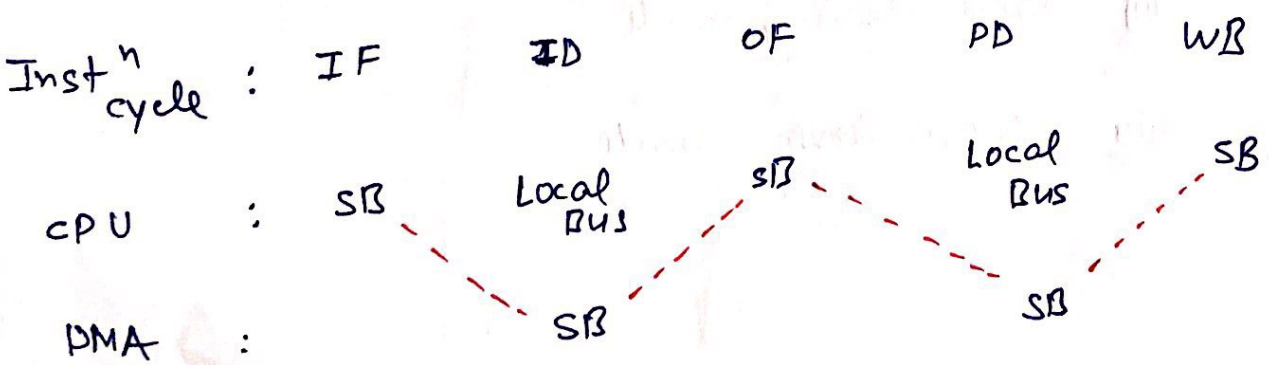
- In Burst mode of DMA, Bulk amount of data at a time will be transmitted to MM. Latter reciving the Sys BUS from the CPU. Therefore CPU waiting time is more

Taken the SB from the CPU → Bulk Data Transfer → Return the SB to CPU

- In cycle-stealing mode of DMA, data file is organized into a small units named as block, later transfer the data file to MM by req the BUS control multiple times. Therefore CPU will be suspended in a small time interval.

Taken the SB from CPU → Block Data Transfer → Return the SB to CPU

- In Inter-leaving mode of DMA, data will be transfered to MM in form of words when the CPU busy with local opn.



Q. Consider 10KBPs IO-device interface to 32-bit CPU in a programmed IO-mode. Data-transmission takes place b/w the CPU and I/O in a word-wise sequence. I/O interface overhead is 40μs. How much performance is improved when the device is operating under interrupt mode under programmed IO mode.

Prog - IO

10KB — 1sec

1W
(4B) ⇒ $\frac{4B}{10KB}$ sec = 400μs = ET_{Prog}

INT—IO

CPU time = IO - interface latency

= 40μs.

i.e ET_{INT} = 40μs

$$S = \frac{ET_{Prog}}{ET_{INT}} = \frac{400}{40} = 10$$

S = 10

Q. Consider 100 KBPs IO device interfaced to 32-bit CPU in a cycle stealing mode of DMA whenever 8W data is available in the buffers then it is transferred to MM. Machine cycle time is 2MS. How much % of CPU time is consumed in the DMA op^n.

$$\% \text{ time CPU is blocked} = \left(\frac{Y}{X+Y}\right) \times 100$$

X is preparation time
(Depends on IO-device speed)

100KB ⟶ 1sec

1B ⟶ $\frac{1}{100K}$

8W(32B) ⟶ $\frac{32}{100K}$ = 320 MS

X = 320MS

Y is transfer time (MM speed)
(1M/C cycle is used to transfer 1word data)

So 8 M/C cycles req. to transfer 8 words

Y = 8X 2

Y = 16MS

$$\%\ \text{time CPU blocked} = \frac{16}{320+16} \times 100$$

$$= \boxed{4.76\ \%}$$

Q. Consider 1MBPS IO-device interfaced to 32-bit CPU using DMA interface. DMA contains 8-bit count register 32-bit data reg & 24-bit addr reg. Data file size is 4KB. How many DMA cycles are required to transfer the data file to MM.

DMA opn is controlled by the count register-

count reg = 8-bit reg

= Maintains 256 counts (max)

In every count, 1 W Data is transmitted

Data size/DMA cycle = 256 W = 256 × 4B

= 1024B

= 1KB

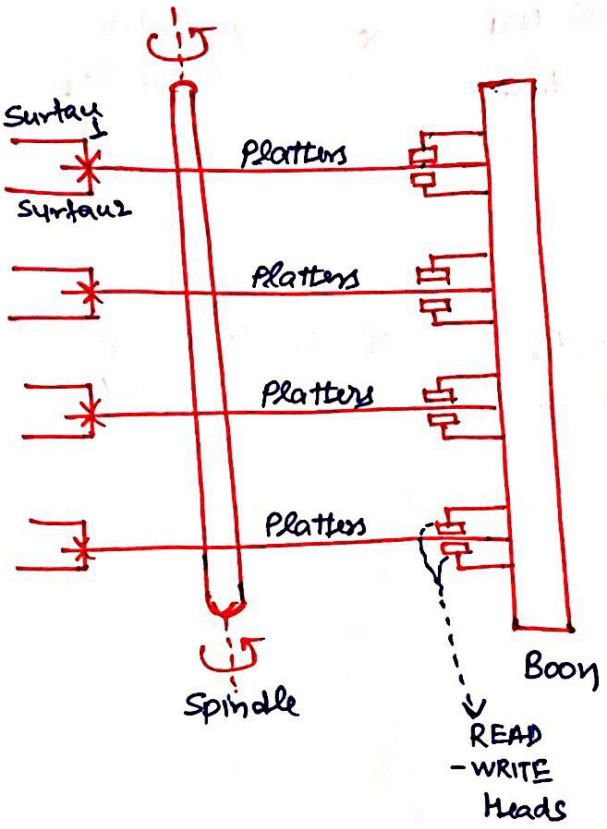
File-size = 4KB

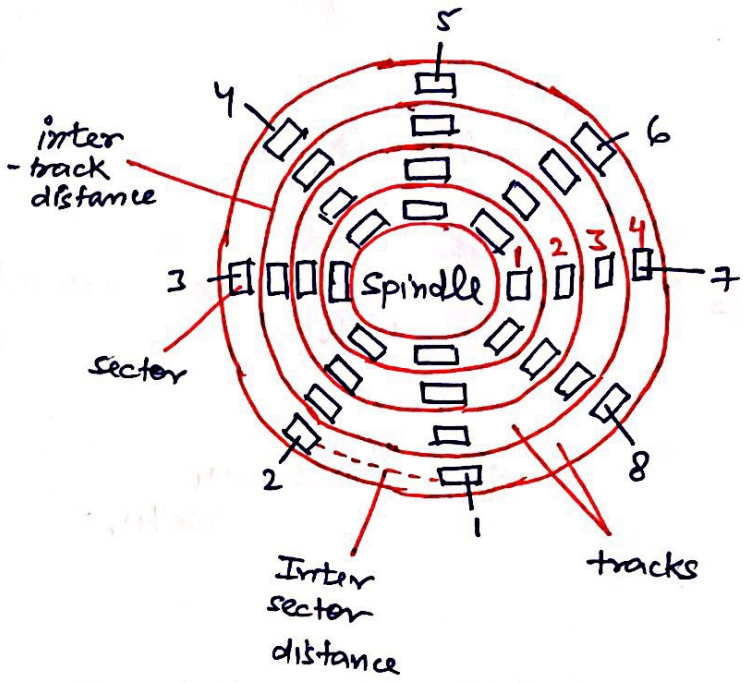
DMA cycles req for 4KB = 4 cycles.

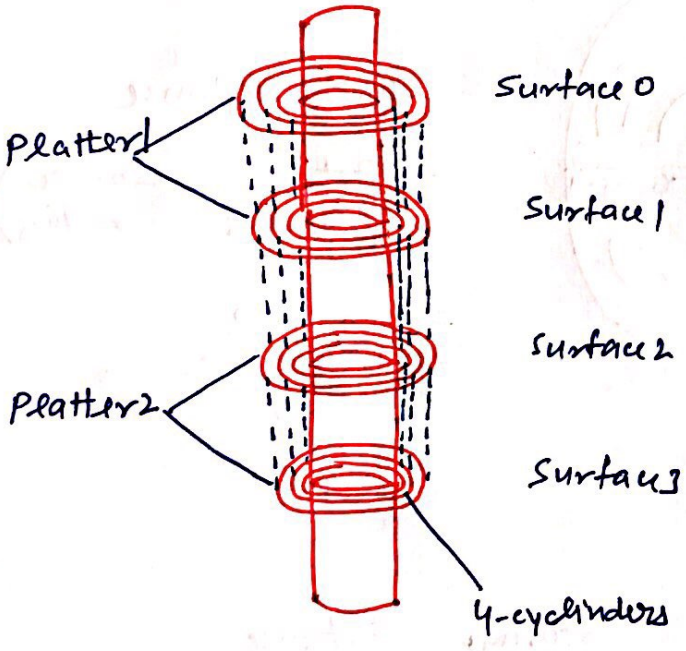
Hard Disk Structure :-

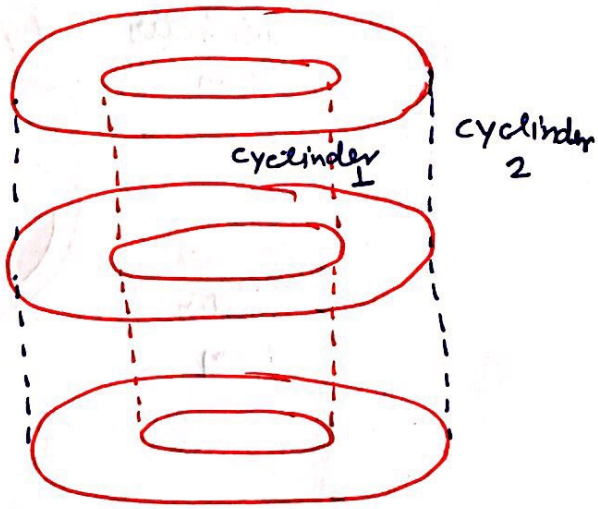
- It is a direct access electromagnetic storage component.
- It contains a bunch of magnetic coated platters to store the data.
- Each platters contains two surfaces. Each surface contains set of tracks.
- Each track contain set of sectors.
- Sector holds the data.
- In the hard disk each surface has its own READ - WRITE head, use to access the data.
- In the hard disk surfaces, tracks and sectors are the addressable units.
- In the hard disk cyclinder is formed by connecting the same track no. in all the surfaces.

Platters — surfaces — tracks — sectors — bytes









- no. cylinders in the disc = no. tracks in the surface
- Track capacity = no. sectors per track × no. bytes per sector
- Cylinder capacity = no. surfaces in disk × track capacity

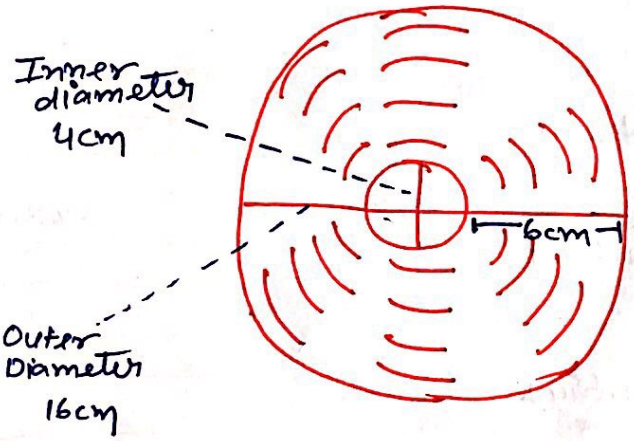
Disk capacity = no. of cyclinders in the disc × cyclinder capacity

Generalization:-

= no. Surfaces in the disk × no. of tracks per surface × sectors per track

× bytes per sector

Surface Analysis:-



Storage distance

= (Outer radius − Inner radius)

= 8 − 2

= 6cm

no. of tracks per surface = $\frac{\text{Storage Distance}}{\text{Inter track Distance}}$

= $\frac{6\text{ cm}}{0.2\text{mm}}$

= 300

- Different adjustments are required in the disk op^n -

(i) Seek time : Amount of time required to move the head point from the initial position to desired track in the surface.

(ii) Rotational Latency : Amount of time required to adjust the head point at the beginning of desired sector in the track. In best case Rotational latency is 0 because head point is adjusted during the seek time itself. In worst case, 1 - complete revolution is required to adjust the head points. In av. case half of the revolution is required to adjust the head point.

(iii) Transfer time : Amount of time req to access the data from hard disk

(iv) Overhead : Additional delays in the disk op^n.

Total Time req to access the data from the harddisk (T_{avg} = Seek time + Avg Rotational latency + Transfer time + Overhead)

NOTE :- When the sector address is expressed in a 3D address format then sector no. in the hard disk is calculated as,

sector no = (i, j, k)

- i : cylinder no.
- j : surface no.
- k : sector no.

Sector no = $k + s(j + i * t)$

- k : count of a sector no. w.r.t surface
- j : count of respective surface
- i : count of current cylinder

Q. Consider hard disk with a rotational rate of 9800 rpm. Avg seek time of a disk is 8.4 ms. Disk contains 64 platters, surface contains 128 tracks, track contains 256 sectors. Sector holds 512 B data. 32 KB file is stored in the hard-disk.

Calculate the following:

(a) Disk Capacity.

(b) Time req to access file

(c) " " " 100 random sectors

(d) Disk - data transfer rate (bandwidth)

(a) Disk capacity = $64 \times 2 \times 128 \times 256 \times 512$

$= 2^{31}$ B

$= 2$GB

(b) Seek-time = 8.4ms

9800 revolution = 1 min = 60 sec

1 revolution = $\frac{60}{9800}$ s

$= 6.12$ ms

∴ avg. Rotational-latency = $\frac{6.12}{2} = 3.06$ ms

1-revolution time ⟶ 1-track data (256 sectors)

1 sector ⟷ 512 B

for 32KB ⟶ $\frac{32KB}{512}$ = 64 sectors

so, 6.12 ms ⟶ 256 sectors access

1 sector access = $\frac{6.12}{256}$ ms

so, time req for access 64 sectors data -

$$\text{Transfer time} = \frac{6.12\,ms}{256} \times 64$$

$$= 1.53\,ms$$

$$T_{avg} = (8.4 + 3.06 + 1.53 + 0)\,ms$$

$$= 12.99\,ms$$

(c) Random sectors accessing requires the adjustment to every sector.

Transfer time for one-sector $= \frac{6.12}{256}$

$= 0.0239\,ms$

T_{avg} for 1 sector $= (8.4 + 3.06 + 0.0239 + 0)\,ms$

T_{avg} for 100 random sectors $= (8.4 + 3.06 + 0.0239 + 0)\,100$

$= 1148.39\,ms$

(d) Bandwidth

Surface data rate : In 1 revolution time — 1 track data accessed

6.12 ms — 256 × 512 B data

so, Data-rate = $\frac{256 \times 512\ B}{6.12\ ms}$

Since Total = 128 surfaces are present

so Total data rate = $\frac{256 \times 512\ B}{6.12\ ms} \times 128$

Spatial Locality in the Memory :-

- Spatial Locality means adjacent data words accessing in the same block, in a sequence.
- In the array implementation prows spatial locality is realized in the memory structure.
- Array is an order set of homogenous data elements, stored in the memory in a row-wise sequence.

eg:- char a[4][4];

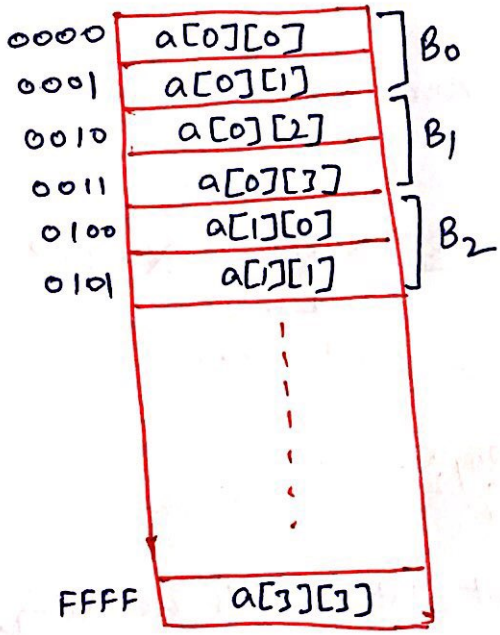
a[0][0], a[0][1], a[0][2], a[0][3]

a[1][0], a[1][1], a[1][2], a[1][3]

a[2][0], a[2][1], a[2][2], a[2][3]

a[3][0], a[3][1], a[3][2], a[3][3]

Memory storage



Assume block size = 2B

Array elements Accessing

(a) Row-major Indexing:- i.e a[0][0], a[0][1], a[0][2]

Assume CM is empty -

Access :-

a[0][0] : M ; B_0 (a[0][0], a[0][1]) → CM

a[0][1] : H ;

a[0][2] : M ; B_1 (a[0][2], a[0][3]) → CM

a[0][3] : H ;

Hit Follows miss H = 50%

Element size = 1B

Block size = 2B

no. of Elements/block = 2B/1B = 2

no. of Block/row = 2

no. of miss/row = 2

So, Total no. of misses/array = no. of rows * no. of misses/row

= 4 * 2

= 8

(b) Col-major Indexing :- i.e a[0][0], a[1][0], a[2][0], ...

Assume cM is empty

a[0][0] : M $B_0 \longrightarrow$ CM

a[1][0] : M $B_2 \longrightarrow$ CM

a[2][0] : M $B_4 \longrightarrow$ CM

a[3][0] : M $B_6 \longrightarrow$ CM

* Random Block references replace the existed Blocks in CM.

Every element in the column is miss.

Element size = 1B

Block size = 2B

no. of Elements/Block = $\frac{2B}{1B}$ no. of Element/col = 4

no. of Blocks/col = 4

so, no. of miss/col = 4

Total misses/array = no. of cols * no. of miss/col

= 4 * 4

= 16

Conclusion

Storage sequence	Access sequence	Spatial Locality
Row-wise	Row-wise	YES
Row-wise	Col-wise	NO
Col-wise	Row-wise	NO
Col-wise	Col-wise	YES