6 IO Interface DMA Transfer and Secondary Storage

6.1 SECONDARY MEMORY

Disk

Magnetic disk:

- Magnetic disk is a storage device. The platter of the magnetic disk has both side recording (storage) surfaces. Getting the track is random access, whereas getting the desired sector of the track is sequential access.

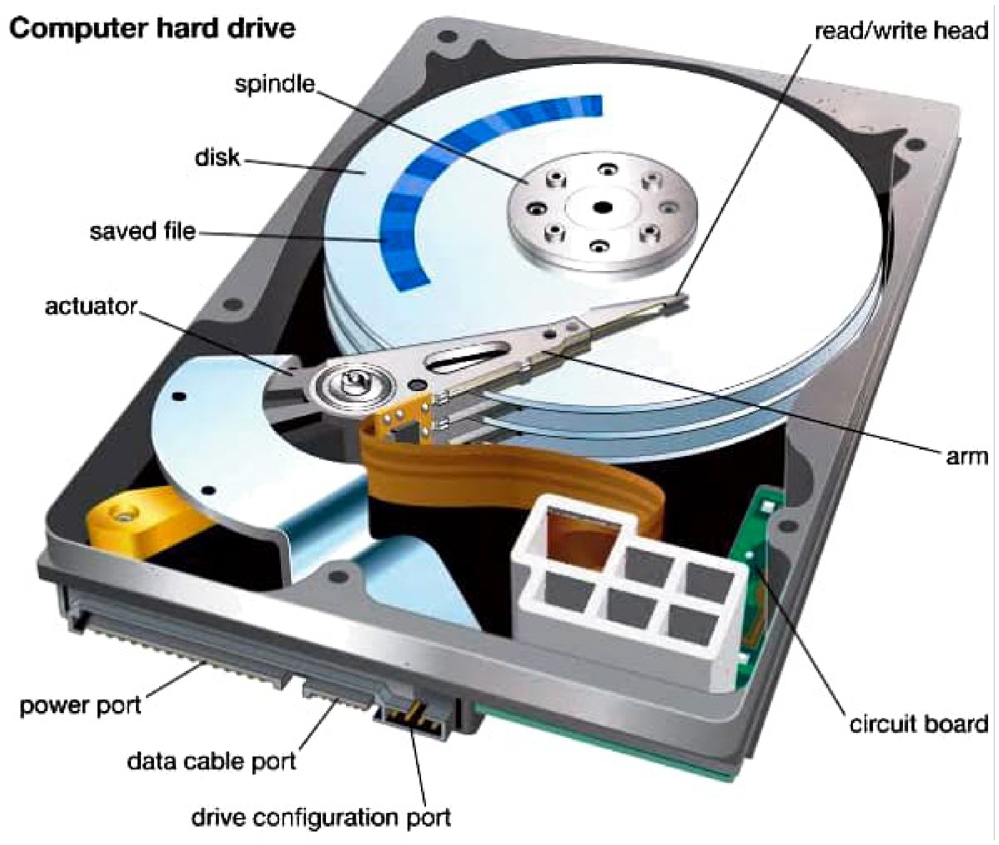


Fig. 6.1 Disk Hardware

Physical structure of the disk:

- Platter has two surfaces, both the surfaces are storage surfaces.
- All the platters are mounted with one common spindle.
- To perform the read/write operation, one pointer is needed at every surface. That pointer is called Read/Write head.
- All Read/Write heads are attached to one common arm assembly.
- Read/Write head can move only in two ways, either forward or backwards.

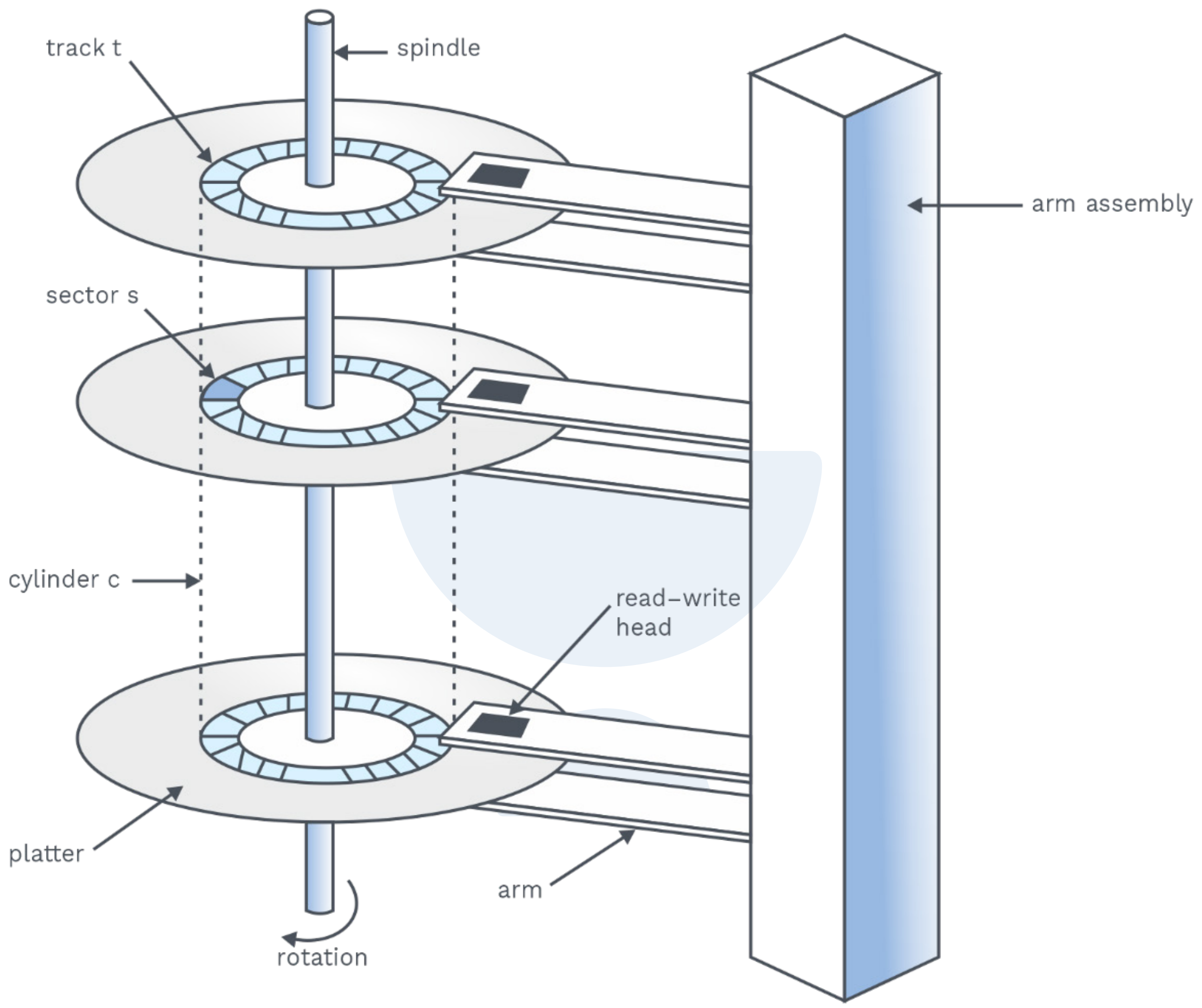


Fig. 6.2 Physical Structure of the Disk

Top view of the disk

- Top view of the disk is the top most surface.
- Surface contains tracks.
- Track contains sectors.
- Sector contains the data (in byte/word).
- Sector is the smallest unit of disk which can be read/write at once.
- Consecutive set of sectors is called cluster.

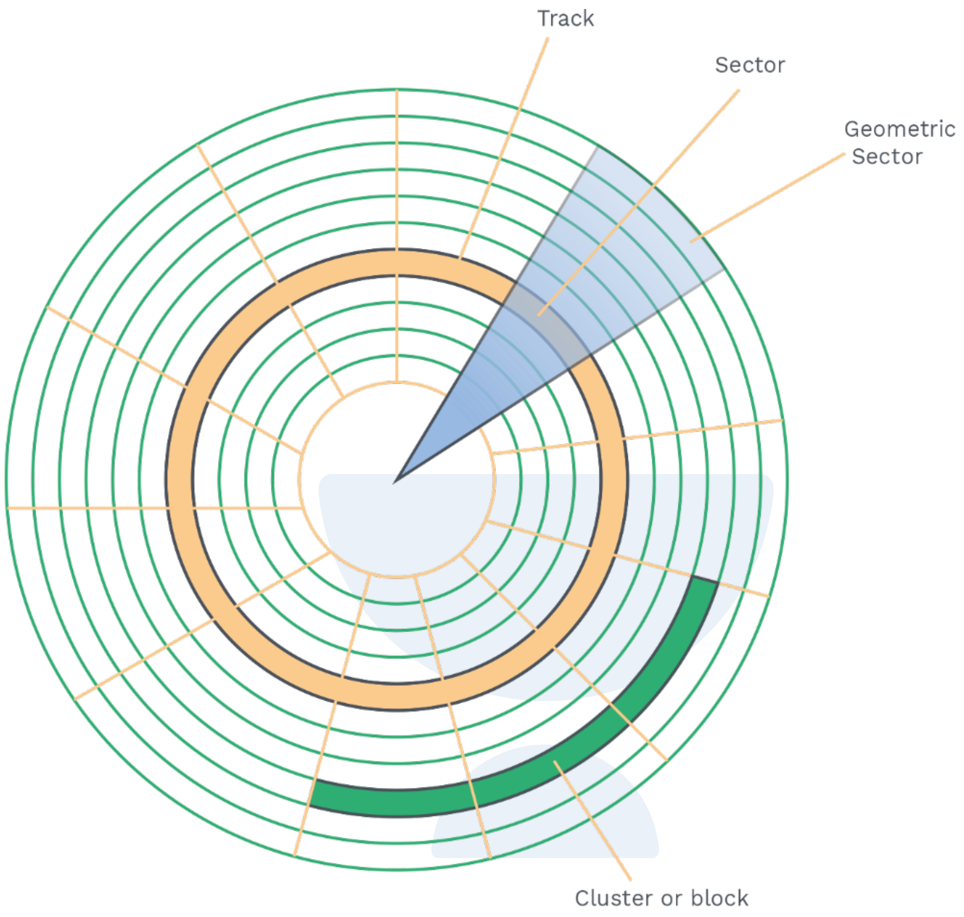


Fig. 6.3 Top View of the Disk

Magnetic disk capacity:

- If platters given instead of the surfaces

$$\boxed{\text{Capacity}\left(C\right) = 2 * \left(\begin{matrix}\text{Number of}\\ \text{platter}\end{matrix}\right) * \left(\begin{matrix}\text{Number of tracks}\\ \text{per surface}\end{matrix}\right) * \left(\begin{matrix}\text{Number of}\\ \text{sectors per track}\end{matrix}\right) * \left(\begin{matrix}\text{Capacity of}\\ \text{the sector}\end{matrix}\right)}$$

$$\boxed{\text{Capacity}\left(C\right) = 2 * P * T * S * B}$$

Where:

P = No. of Platters

T = No. of tracks per surface

S = No. of sectors per track

B = Data in bytes (capacity of sector)

Sector capacity:

There are two types of storage in a sector.

1) Sector having the constant capacity

2) Sector having the variable capacity (Not in Gate)

Sector having constant capacity:

Constant capacity means capacity of the sector is constant in each track.
The storage density varies from sector to sector.

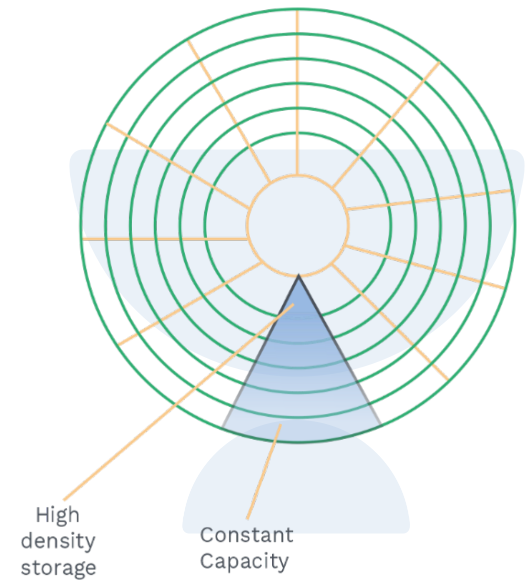


Fig. 6.4

Note:

Default constant capacity is used if each sector having constant capacity then capacity of disk is

$$\boxed{\text{Capacity of disk} = \left(\begin{matrix}\text{Number of}\\ \text{surfaces in a disk}\end{matrix}\right) * \left(\begin{matrix}\text{Number of tracks}\\ \text{per surface}\end{matrix}\right) * \left(\begin{matrix}\text{Capacity of}\\ \text{the sector}\end{matrix}\right)}$$

Disk access time:

- To calculate the access time of 1 sector of disk need to calculate:
  i) Seek time
  ii) Rotational delay/latency
  iii) Transfer time
  iv) Extra delay (if given)

- **Seek time:** Time required to move the read/write head over the desired track.
- **Rotational delay:** Time required to rotate the desired sector under read/write head. This time is taken as half rotation time on average.
- **Transfer time:** Time required to read or write one sector.

Access time:
Access time is the sum of all these delays, seek time, rotational delay, transfer time and extra time (If given in the question).

$$\boxed{\underset{\text{of 1 sector}}{\text{Disk Access time}} = \underset{\text{time}}{\text{Seek}} + \text{Rotational delay / latency} + \underset{\text{time}}{\text{Transfer}} + \underset{\text{delay}}{\text{Extra}}}$$

Rotational delay/latency:
Rotation latency/delay can be two types:

1) If current and target sector given:

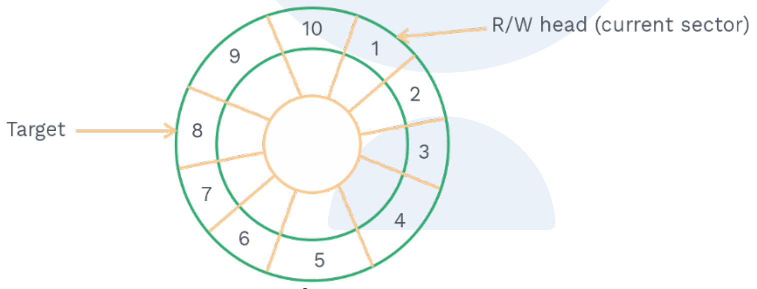


Fig. 6.5

Suppose the track contains 10 sectors, and the rotation time of the disk is 30 ms.

- To read the 8th sector data 7 sector rotation needed

$$\text{rotational delay} = 7 * \frac{30}{10} = 21\text{ms}$$

General formula:

$$\boxed{\text{Rotational latency / delay} = \frac{\text{Disk rotation time}}{\text{No. of sectors per track}} * \text{No. of sectors to rotate}}$$

2) If the current and target sectors have not given:
Use average rotational latency
So,

$$\boxed{\text{Rotational delay / latency} = \frac{\text{Disk rotation time}}{2}}$$

Note:

One track can be transferred, in one disk rotation.
Hence, time taken to transfer a sector -

$$\boxed{1 \text{ sector transfer time } = \frac{\text{Disk rotation time}}{\text{Total no. of sectors per track}}}$$

PRACTICE QUESTIONS

Q1 **Consider a disk with following parameters:**
Number of platters = 64
Number of tracks per surface = 1024
Number of sectors per track = 128
Capacity of the sector = 256 Bytes
Find the capacity of the disk and the number of bits required for addressing the disk?

Sol: **4 GB, 24 bits**

$$\boxed{\text{Capacity of the disk } (C) = 2 * P * T * S * B}$$

Capacity of the disk = 2 * 64 * 1024 * 128 * 256 bytes
= $2^{1+6+10+7+8}$
= 2^{32}
= 4 GB

- Addressing the disk means, number of bits required to represent each sectors.
  Number of sectors = 2 * 64 * 1024 * 128
  = 2^{24}
  Number of bits = $\log_2 2^{24}$
  = 24 bits

Q2 **Consider a disk with 64 surfaces, 4K tracks per surface, 2K sectors per track, and the sector contains 1024 bytes of data. Seek time of the disk is 12ms and the disk rotational speed is 6000 rpm (revolution per minute). Calculate the disk access time and disk transfer rate.**

Sol: **17.005 ms, 200 MBPS**

- Number of surfaces = 64
- Number of tracks per surface = 4K
- Number of sectors per track = 2K
- Capacity of the sector = 1024 bytes
- Seek time = 12 ms
- 6000 revolution = 60 sec
- 6000 revolution = 60000 msec
- 1 revolution = $\frac{60000}{6000}$ msec = 10 msec

Access time:

$$\text{Disk Access time} = \text{Seek Time} + \text{Rotational delay} + \text{1 sector transfer time} + \text{Additional delay}$$

$$= 12 + \frac{\text{Disk rotation time}}{2} + \frac{\text{Disk rotation time}}{\text{No. of sectors per track}}$$

$$= 12 + \frac{10}{2} + \frac{10}{2K}$$

= 12 + 5 + 0.005 (K = 1000 because we are calculating in time)

= 17.005 ms

- Disk Transfer Rate

In one disk rotation, one track can be transferred.

1 Track Capacity = 2K * 1024

= 2^{11+10} = 2 MB

So, we can say, 2MB of data can be sent in 1 rotation (10 ms)

$$\begin{tikzcd}
\text{10 msec} \arrow[r, dash] \arrow[dr, dotted, dash] & \text{2 MB} \\
\text{1 sec} \arrow[r, dash] \arrow[ur, dotted, dash] & ?
\end{tikzcd}$$

$$= \frac{2\,\text{MB}}{10 * 10^{-3}} = \frac{2 * 10^3}{10}\,\text{MBPS (Mega bytes per sec.)}$$

= 200 MBPS

Q3 **Consider a hard disk with rotational speed of 3200 rpm. Average latency of disk is 8.4 milliseconds. A 1 MB file is stored in the hard disk. The hard disk contains 16 platters. Each surface contains 64 tracks. Each track contains 128 sectors. Each sector holds 64 B of data. What is the percentage of the disk is occupied by the file and disk data transfer rate (bandwidth), respectively?**

a) 4.75%, 13.98 MBPS **b) 6.25%, 15 MBPS**

c) 5%, 16 MBPS **d) 6.25%, 13.98 MBPS**

Sol: **d)**

Disk capacity = (No. of platters/surfaces*2)*(No. of tracks/surface)*(No. of sectors/track)*(No. of bytes/sector)

$= 16 \times 2 \times 64 \times 128 \times 64\text{B}$

$= 2^{4+1+6+7+6}\ \text{B} = 2^{24}\ \text{B} = 16\ \text{MB}$

Percentage of disk occupied by the file = $\dfrac{\text{File size}}{\text{Disk capacity}}$

$= \dfrac{1\text{MB}}{16\text{MB}} * 100\%$

$= 6.25\%$

Disk data transfer rate is calculated as

$\Rightarrow$ Number of surfaces*Surface rate

$\Rightarrow$ Number of surfaces* $\dfrac{\text{Track capacity}}{\text{Time taken by one revolution}}$

$$\Rightarrow \frac{32 \times 128 \times 64}{\left(\frac{60}{3200}\right)}\text{BPS} \quad \left[\begin{array}{l} \because 3200\text{ revolution} \rightarrow 60\text{ sec} \\ 1\text{ revolution} \rightarrow \dfrac{60}{3200}\text{ sec} \end{array}\right]$$

$$\Rightarrow \frac{32 \times 128 \times 64 \times 3200}{60}\text{BPS}$$

$\Rightarrow$ 13981013.33 BPS

$\Rightarrow$ 13.98 MBPS

$\therefore$ Option (d) is correct.

Previous Years' Question

Question: Consider a disk pack with 16 surfaces, 128 tracks per surface and 256 sectors per track. 512 bytes of data are stored in a bit-serial manner in a sector. The capacity of the disk pack and the number of bits required to specify a particular sector in the disk are respectively:

a) 256 Mbyte, 19 bits
b) 256 Mbyte, 28 bits
c) 512 Mbyte, 20 bits
d) 64 Gbyte, 28 bit

Sol: a) **(GATE-2007)**

Multiple sector access time:
Sequential access:
All sectors are in the form of cluster (same track)

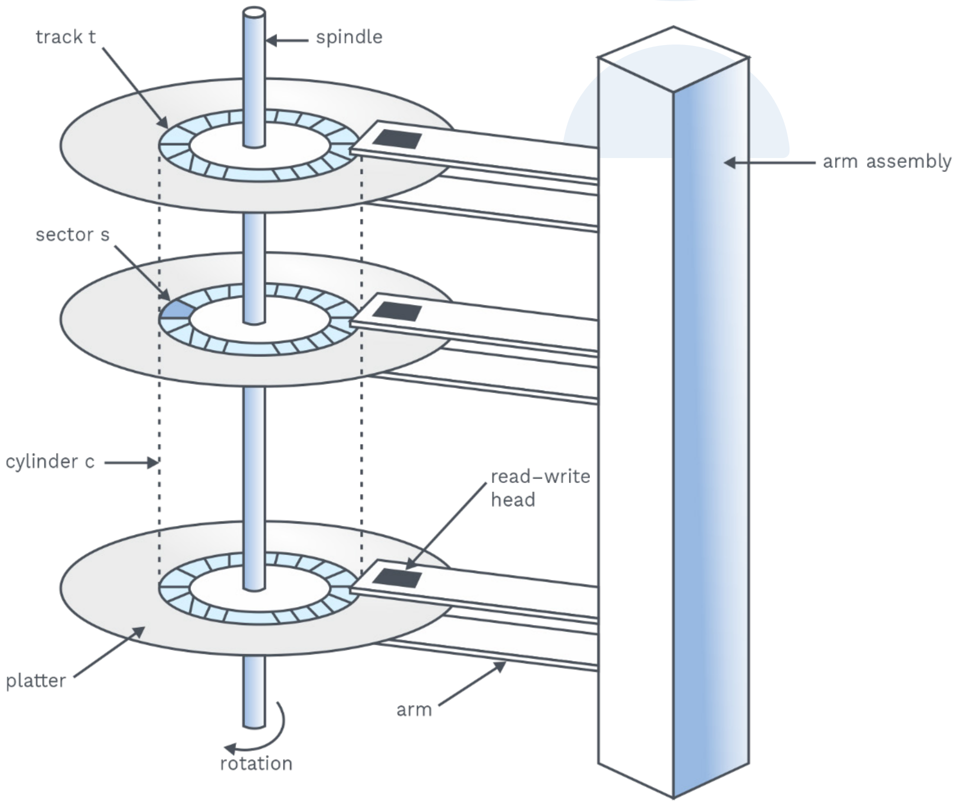


Fig. 6.6 Disk Access

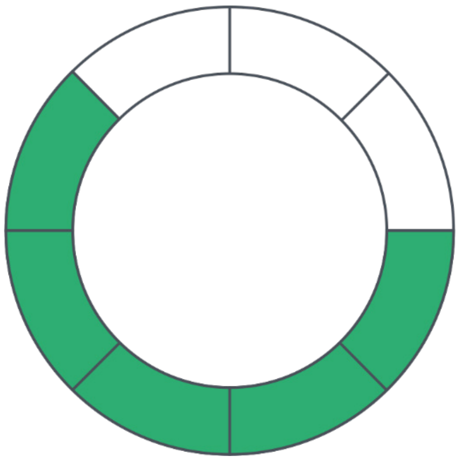


Fig. 6.7

Suppose, we need to transfer P sectors then access time is:

$$\boxed{\text{Access time} = \text{Seek Time} + \text{Rotational latency} + P * \text{1 sector transfer time}}$$

Random access time:

For random access, each time needs to calculate seek time, rotational time and sector transfer time. Because directly we are accessing the desired track.

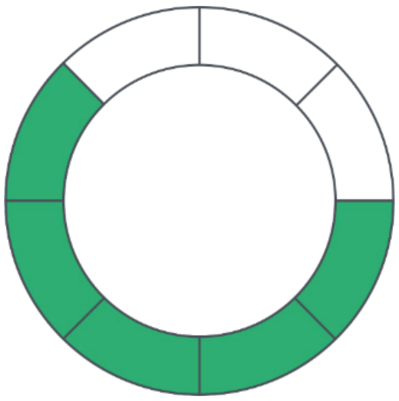


Fig. 6.8

To transfer the P sectors access time is:

$$\text{Access time} = P * \left[\text{Seek time} + \text{rotational latency} + \text{1sector transfer time}\right]$$

Previous Years' Question

Question: Consider a disk pack with a seek time of 4 milliseconds and rotational speed of 10000 rotations per minute (RPM). It has 600 sectors per track and each sector can store 512 bytes of data. Consider a file stored in the disk. The file contains 2000 sectors. Assume that every sector access necessitates a seek, and the average rotational latency for accessing each sector is half of the time for one complete rotation. The total time (in milliseconds) needed to read the entire file is:

a) 14020 **b)** 14000 **c)** 25030 **d)** 15000

Sol: a) **(GATE-2015)**

Cylinder:

- To save the seek time data stored and accessed in the form of a cylinder.
- Collection of all the tracks with the same radius is a cylinder.

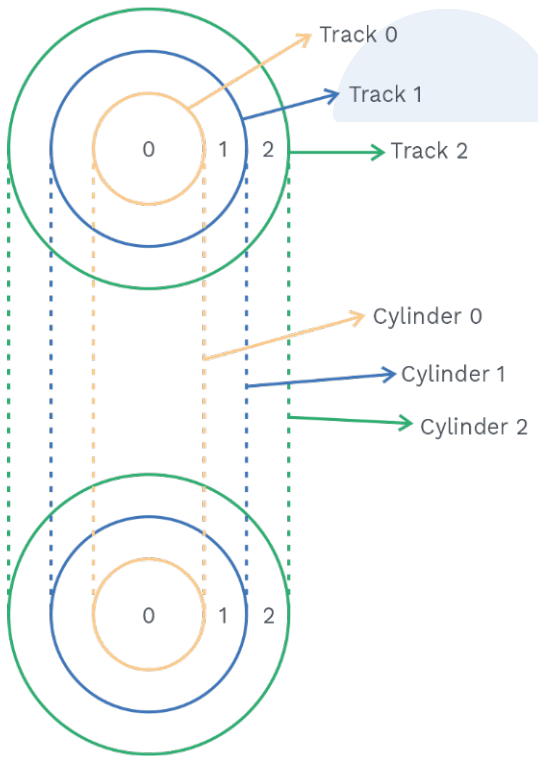


Fig. 6.9 Cylindrical View of the Disk

In the above diagram, total 3 tracks are present in the above surface.
So, we can say that

$$\boxed{\text{No. of tracks in the surface} = \text{No. of cylinders present in the disk}}$$

Disk addressing:

- Disk addressing is used to find the sequence number of the specified sector with the help of cylinders, surfaces and sectors.
- Direct sequence number of the sector is accessed with no time.

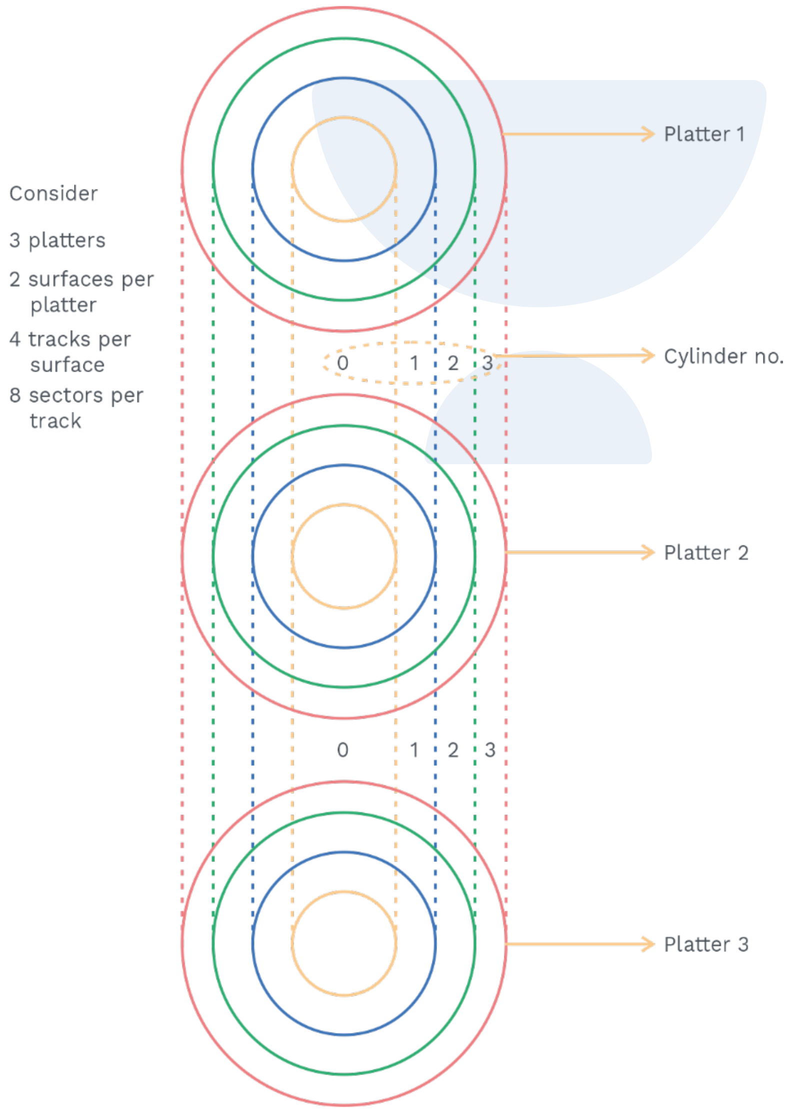
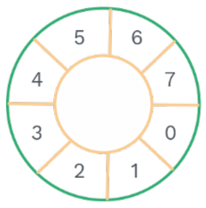


Fig. 6.10 Disk Addressing



8 sectors per track

Fig. 6.11

- 4 cylinders are present in the disk because no. of tracks per surface is 4.

Address representation:
<cylinder number, surface number, sector number>
Number of cylinders = 4 (0 to 3)
Number of surfaces = 6 (0 to 5)
Number sectors = 8 (0 to 7)

0th cylinder:

		address	
0th surface	<0, 0, 0>	0	
	<0, 0, 1>	1	
	⋮	⋮	
	<0, 0, 7>	7	
1st surface	<0, 1, 0>	8	
	<0, 1, 1>	9	
	⋮	⋮	
	<0, 1, 7>	15	
	⋮	⋮	
5th surface	<0, 5, 0>	40	
	⋮	⋮	
	<0, 5, 7>	47	(8 * 5 + 7)

1st cylinder:

$$0^{\text{th}}\text{ surface}\begin{cases} \langle 1, 0, 0\rangle & 48 \\ \vdots & \vdots \\ \langle 1, 0, 7\rangle & 55 \end{cases}$$

$$\vdots \qquad \vdots$$

$$5^{\text{th}}\text{ surface}\begin{cases} \langle 1, 5, 0\rangle & 88 \\ \vdots & \vdots \\ \langle 1, 5, 7\rangle & 95 \end{cases}$$

General formula:

To find address of the sector's sequence number

$$\text{Address} = \langle \underset{\substack{\text{Cylinder}\\ \text{number}}}{\underset{\downarrow}{D}} \quad , \quad \underset{\substack{\text{Surface}\\ \text{number}}}{\underset{\downarrow}{S}} \quad , \quad \underset{\substack{\text{Sector}\\ \text{number}}}{\underset{\downarrow}{T}} \rangle$$

$$\boxed{\text{Sector sequence number} = \left(D * \substack{\text{Number of sectors}\\ \text{per cylinder } (N_D)}\right) + \left(S * \substack{\text{Number of sectors}\\ \text{per track } (N_P)}\right) + T}$$

$$\boxed{\text{Sector sequence number} = (D * N_D) + (S * N_P) + T}$$

PRACTICE QUESTIONS

Q4 **Consider a disk with the following parameters:**
Number of platters = 6
Number of recording surface per platter = 2
Number of tracks per surface = 32
Number of sectors per track = 32
address of the sector is given as a triple <D, S, T>, where D is cylinder number, S is the surface number and T is the sector number. 0th sector representation (address) is <0, 0, 0>, the 1st sector address is <0, 0, 1>, and so on. The address <12, 8, 20> corresponds to sector number?

Sol: **4884**

No. of cylinders in the disk = No. of tracks per surface

= 32

No. of surfaces = 6 * 2

= 12

No. of sectors per

cylinder (N_D) = 12 * 32

= 384

No. of sectors per track (N_P) = 32

$$\begin{tikzcd} <12, 8, 20> \arrow[d] \\ <D, S, T> \end{tikzcd}$$

D = 12

S = 8

T = 20

Sequence No. = (D * N_D) + (S * N_P) + T

= (12 * 384) + (8 * 32) + 20 = 4884

Q5 **In the above question (Q3) if sequence no. of the sector is 4250. What is the address of the sector as a triple <D, S, T> format?**

Sol: **<11 0 26>**

N_D = 384

N_P = 32

$D = \left\lfloor \dfrac{\text{sequence No. of the sector}}{N_D} \right\rfloor \quad D = \left\lfloor \dfrac{4250}{384} \right\rfloor$

= 11

S = (sequence no. mod N_D)/N_P

= (3250 mod 384)

$S = \left\lfloor \dfrac{26}{32} \right\rfloor$

S = 0

T = (Sequence no. % N_D) % N_P

= (3250 Mod 384) Mod 32

= 26 Mod 32

T = 26

Triple format = <D S T> = <11 0 26>

Previous Years' Question

Question: A hard disk has 63 sectors per track, 10 platters each with 2 recording surfaces, and 1000 cylinders. The address of a sector is given as a triple (c, h, s), where c is the cylinder number, h is the surface number and s is the sector number. Thus, the 0th sector is addressed as (0, 0, 0), the 1st sector as (0, 0, 1), and so on. The address corresponds to the sector number:

a) 505035 **b)** 505036 **c)** 505037 **d)** 505038

Sol: c) **(GATE-2009 (2 Marks))**

Previous Years' Question

Question: Consider a hard disk with 16 recording surfaces (0–15) having 16384 cylinders (0–16383) and each cylinder contains 64 sectors (0–63). Data storage capacity in each sector is 512 bytes. Data are organised cylinder–wise, and the addressing format is. A file of size 42797 KB is stored in the disk, and the starting disk location of the file is. What is the cylinder number of the last sector of the file if it is stored in a contiguous manner?

a) 1281 **b)** 1282 **c)** 1283 **d)** 1284

Sol: d) **(GATE-2013 (2 Marks))**

I/O organisation:

- I/O is one of the major components to communicate with the CPU.
- Basic feature of the I/O is to exchange the data with other devices.

I/O devices (Peripheral devices):

- All devices that are connected to the CPU externally except the main memory are called as I/O devices.
- These devices are connected to the CPU through single bus arrangement.

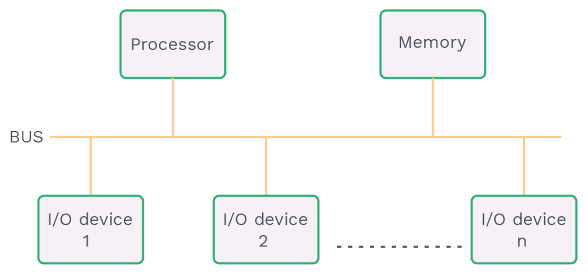


Fig. 6.12 Computer Architecture

- These I/O devices are categorised into three types:

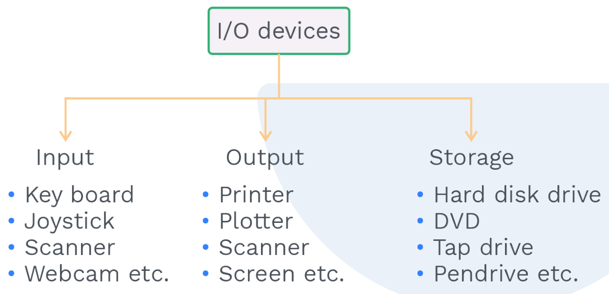


Fig. 6.13 Classification of I/O Devices

How the CPU connected to I/O device:

- CPU does not connect to the I/O device directly.
- CPU connected to the I/O device by I/O interface.
- I/O Interface is (Hardware + Software) interface. The software interface is nothing but a device driver.

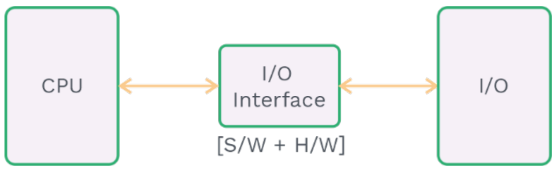


Fig. 6.14 I/O Interface

Need for interface:

- CPU is an electronic device, whereas I/O are electromagnetic device.
- For good synchronisation I/O interface is required because the CPU is fast in terms of speed, whereas I/O devices are usually slow.
- For conversion of format I/O interface is required because data codes of I/O devices are different from the word format in the CPU.
- Operations of I/O devices should not be overlapped (two or more devices can't perform an operation at the same time).
- **Versions of interface:** There are three versions:

i) I/O interface: I/O interface, which provides only interfacing.

ii) DMA controller: DMA controller provides I/O interface and DMA facility.
iii) I/O Processor: I/O processor provides DMA controller (I/O interfaces also) and I/O instruction execution facility.

Memory and I/O:
There are three ways that connect CPU to memory and I/O:
1) Separate buses for both
2) Common data, address bus
3) Common address, data and control bus

1) Separate buses for both:

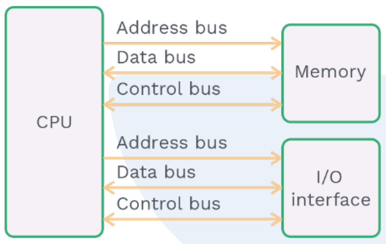


Fig. 6.15

- Separate buses for both I/O and memory.
- If the CPU wants to access memory, then CPU can access memory only, not I/O. Separate connections for both memory and I/O. So, it is costly.
- Easy to implement.
- CPU can perform operations either with I/O or with memory but not parallel.

2) Common data, address bus:

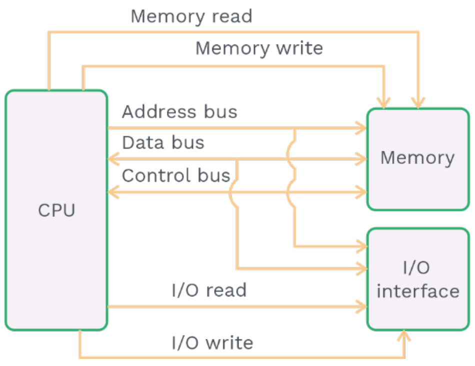


Fig. 6.16

- CPU communicates with memory and I/O by generating the address.
- If the CPU wants to access the memory, then the control signal of I/O will be disabled then the CPU can perform read/write operations in memory.

- CPU can distinguish between memory and I/O by control signals.
- CPU can perform either read operation or write operation at a time in memory or I/O interface.
- This type of communication is called isolated I/O or I/O mapped I/O, or port mapped I/O. Here address bus is common for both memory and I/O interface, but control signals are different, address can be same as we have common bus and control signals are the one to make the difference between them.

3) Common data, address and control bus:

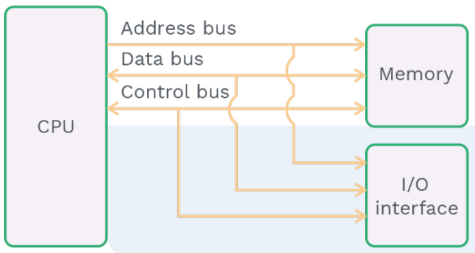


Fig. 6.17

- Here CPU uses common buses for both memory and I/O interface.
- CPU can communicate with memory by generating addresses here, some addresses of memory assigned for I/O devices.
- I/O device does not show their own addresses; they use memory addresses. This type of communication is called memory mapped I/O.
- This is difficult to implement, and this is cheap compared to I/O mapped I/O. Difference between memory mapped I/O and I/O mapped I/O.

	Memory mapped I/O	**I/O mapped I/O**
1)	I/O not having their own addresses but some addresses of memory is assigned to I/O devices	I/O mapped I/O have separate addresses for I/O
2)	Memory wastage	No memory wastage
3)	CPU distinguishes between memory and I/O using addresses	CPU distinguishes between memory and I/O using control signals.
4)	Number of addresses for memory will be less than I/O mapped as we are allocating some of the addresses to the I/O devices.	Number of instructions and addresses are more to access memory but less to access I/O.
5)	All memory access instructions and addressing modes are used for I/O also	Separate instructions and modes for I/O.

Table 6.1 Comparison between Memory Mapped and I/O Mapped I/O

Data transfer:

1 bit of data can be transferred at a time

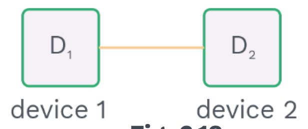


Fig. 6.18

Parallel data transfer:

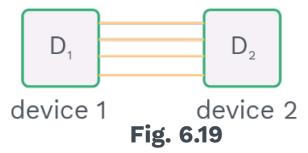


Fig. 6.19

Synchronous data transfer (serial):

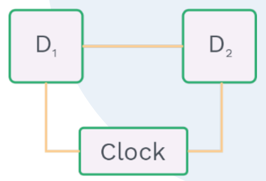


Fig. 6.20

Common clock is used for both devices to control the speed in synchronous data transfer.

Asynchronous data transfer (serial):

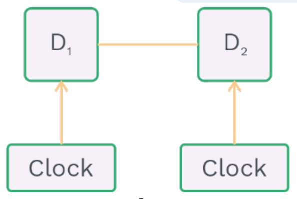


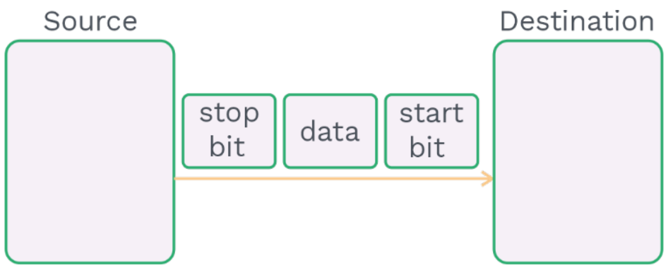
Fig. 6.21

- Here, we use separate clock for both devices.
- Here, devices have the communication but not internally (through clock).

PRACTICE QUESTIONS

Q6 **How many 8 bit characters can be transmitted per second over 14400 band serial communication links using parity synchronous mode of transmission with 2 start bits, 8 data bits, 1 stop and 1 parity bit?**

Sol: **1200 characters per second:**



To send data first bit is start bit (it can be either 0 or 1) last bit is stop bit (reverse of start bit).

- In 1 second 14400 bits can be transmit.

To send 1 character number of bits = start bits + data bits + parity bits + stop bits.

Number of bits = 2 + 8 + 1 + 1 = 12 bits

- Number of characters in one sec $= \frac{14400}{12}$

$= 1200$ characters per sec

Q7 **An asynchronous serial communication is employing 8 character bits, 2 parity bits, 2 stop bits and 2 start bits. To gain the rate of 800 characters/second what is the minimum transfer rate?**

a) 1400 bits per sec **b) 1400 bytes per sec**
c) 11200 bytes per sec **d) 6400 bits per sec**

Sol: **b)**

To send 1 character total (8 + 2 + 2 + 2) 14 bits need to transfer.

- For 1 character = 14 bits
- Number of character send in 1 sec = 800 character
- Number of bits in 1 sec. = 800 * 14 bits per sec.

$= \frac{800}{8} \times 14 \quad = 100 * 14$

$= 1400$ bytes per sec

Hence, option (b) is correct

Q8 **Using a parity synchronous mode of 8 bit character along with 2 stop bits, 1 parity bit and 2 start bits. What is the efficiency of the transmission line?**
a) 61.53% **b) 66.66%** **c) 62.50%** **d) None of these**

Sol: **a)**

Total bits per character = Start bits + data bits + parity bits + stop bits
= 2 + 8 + 2 + 1
= 13 bits

data bits = 8 bit

So,

Efficiency = $\frac{8}{13} * 100$ = 61.538

Hence, option (a) is correct.

Modes of transfer:

1) Programmed I/O or program controlled I/O
2) Interrupt driven or interrupted initiated I/O
3) Direct memory access (DMA)

Programmed I/O:

Overview:

- Whenever the CPU is executing a program related to I/O, it executes that instruction by issuing a command to a suitable I/O module.
- There is no furnishing through which I/O can tell the CPU about data transfer.
- With programmed I/O, I/O will perform the requested command and set the appropriate bits in the I/O status register.
- Further, I/O will not take any action to alert the CPU.
- I/O devices have their own status and wait.
- CPU executes the program periodically and checks the status of I/O devices related to CPU.
- If any I/O device has its status set, then only the CPU performs data transfer for it.

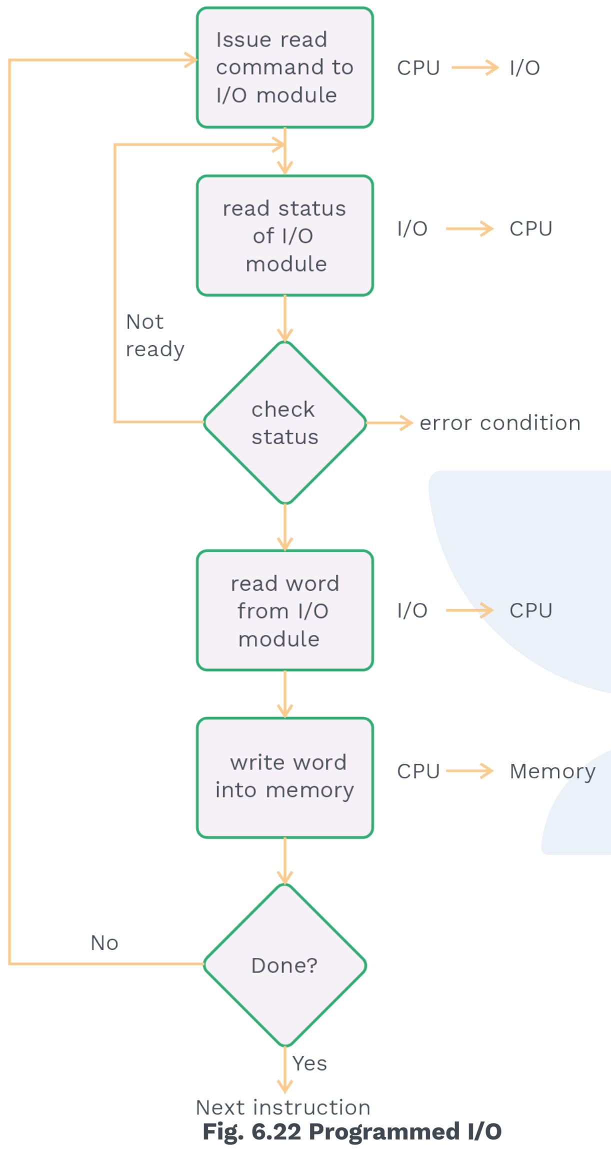


Fig. 6.22 Programmed I/O

Previous Years' Question

Question: The following are some events that occur after a device controller issues an interrupt while process L is under execution.

P) The processor pushes the process status of L onto the control stack.

Q) The processor finishes the execution of the current instruction.

R) The processor executes the interrupt service routine.

S) The processor pops the process status of L from the control stack.

T) The processor loads the new PC value based on the interrupt.

Which of the following is the correct order in which the above events occur?

a) QPTRS **c)** TRPQS

b) PTRSQ **d)** QTPRS

Sol: a) **(GATE-2018)**

$$\boxed{\text{Total time} = \text{Total time to check status of I / O device} + \text{Total data transfer time}}$$

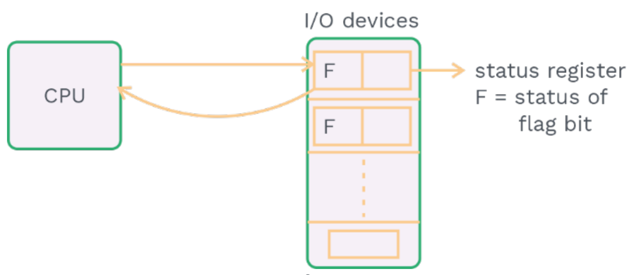


Fig. 6.23

- By default, time required by I/O device to read or share status register based on its own speed is 1 byte.

PRACTICE QUESTIONS

Q9 **Consider a device which is operating on program control mode of I/O with 100 KBPS (kilobytes per second) operating speed. The data transfer is 40 bytes from I/O. The size of status register is 4 bytes. The total time needed to perform the data transfer is ______ μs.**

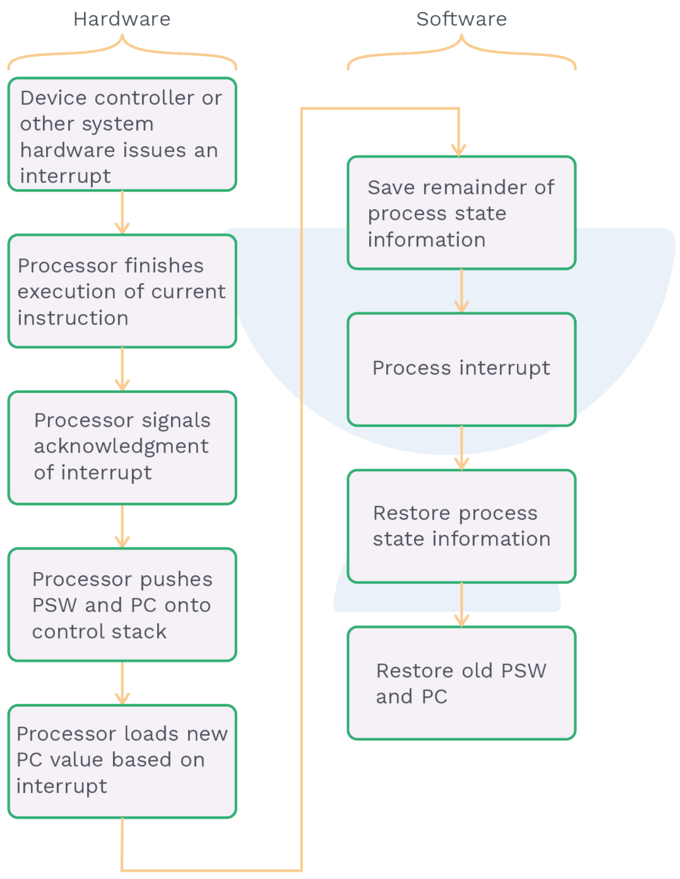
Sol: **440**

Programmed I/O

- Operating speed = 100 KBPS
  100 KB ––– 1 sec
  1 B ––– ?
  1 Byte transfer time = $\dfrac{1\text{B}}{100\,\text{KB}}$
  $= \dfrac{1000}{100}\,\mu\text{sec}$
  1 Byte transfer time $= 10\,\mu\text{sec}$
  Total time = Status check of I/O + data transfer time
  = 4 B status check + 40 B data transfer time
  = 4 * 10 + 40 * 10
  $= 440\,\mu\text{sec}$
  Note: If status register size not given take default 1 byte
- The problem with programmed I/O is CPU wastage is more because to transfer or reception of data CPU has to wait for long (CPU check periodically I/O devices).

Interrupt driven I/O
Interrupt processing:
The sequence of steps that happens after an I/O device raises an interrupt are as follows:
- First CPU will complete the current instructions.
- Store the status of the current process in the stack.
- CPU will go to another process execution accordingly (stack).

- When execution is done then, resume the previous process by taking out the values from the stack.
- The occurrence of an interrupt is in CPU hardware and in software.
- When an I/O device completes an I/O operation, the following sequence of hardware and software events occurs.


Hardware
Software
Device controller or other system hardware issues an interrupt
Processor finishes execution of current instruction
Processor signals acknowledgment of interrupt
Processor pushes PSW and PC onto control stack
Processor loads new PC value based on interrupt
Save remainder of process state information
Process interrupt
Restore process state information
Restore old PSW and PC

Simple Interrupt Processing

Fig. 6.24 Interrupt Driven I/O

Vector address:
Vector gives the location of ISR or vector gives address or reference of ISR (interrupt service routine).

ISR:
A routine, execution of which services the interrupt.
There are two types of interrupts in computer system.

- Vectored interrupt
- Non vectored interrupt

Vector interrupt	Non vectored interrupt
Device sends interrupt signal and vector to the CPU	Device sends only interrupt signal Eg: Software Interrupt
CPU reach to ISR by using vector and execute it to provide service.	CPU first execute DSR (default service routine) to get ISR then CPU reach to ISR to service interrupt

Table 6.2 Comparison between Vector and Non Vector interrupt

Maskable interrupt:
Either the CPU can reject the interrupts, or CPU can accept the interrupts. Accepted interrupts some times kept in waiting (pending) according to priority.

Non maskable interrupt:
CPU always accepts these interrupts because these are high priority interrupt.

Internal interrupt:
CPU generates errors for itself during any instruction execution. CPU solve these errors first, then only CPU can execute the current instruction.
Example: Page fault, segmentation fault etc.

External interrupt:
If any device sends an interrupt to the CPU, these interrupts are also known as H/W interrupt.

Simultaneous interrupts:
CPU services the interrupt of the highest priority device first.

- To handle the priority, there are two solutions software solution and hardware solution.
- In a software solution, the CPU will run a program (software) first. Based on priority, the CPU resolve the interrupt.
- In hardware solution, there are two types.
- Serial solution (Daisy Chaining).
- Parallel solution.

Parallel solution not required for gate exam

Daisy chaining:

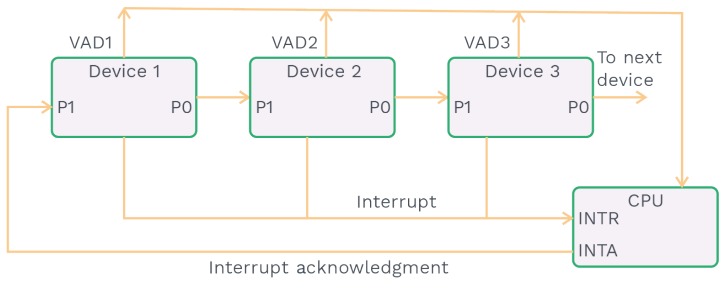


Fig. 6.25 Daisy Chaining

1) INTR: (Interrupt request):
   I/O device sends the interrupt request to CPU.
2) INTA (Interrupt acknowledgement):
   CPU sends the acknowledge (ack) to device 1.
3) If device 1 sent the interrupt then only device 1 accept the INTA and sends the vectored address (VAD) to the CPU by data bus.
4) If device 1 did not send the interrupt, then device 1 forward it to the next device whichever device sent the interrupt it to the CPU, that device accepted the INTA and send the vectored address to the CPU through a data bus and so on.
   Here, device 1 has the highest priority.
   Total time required in interrupt I/O

$$\boxed{\text{Time} = \text{Interrupt overhead} + \text{Service Time}}$$

Interrupt overhead:
Initial time for extra work before servicing the interrupt.

PRACTICE QUESTIONS

Q10 **Consider the processor which takes 10 cycles to service the interrupt. If the CPU runs on 20 MHz clock rate and interrupt overhead is 0.07 microseconds then total time CPU spends for interrupt service is _______ microseconds?**

Sol: **0.57**

Interrupt overhead = 0.07 μsec

Clock rate = 20 MHz

Number of cycles = 10

Total time = overhead time + Service time

Service time = 10 cycles

$$\text{Cycle time} = \frac{1}{\text{clock rate}} = \frac{1}{20}\,\mu\text{sec}$$

So, total time = overhead time + service time

$$= 0.07 + 10 * \frac{1}{20}$$

= 0.07 + 0.50

= 0.57 μsec

Q11 **Consider a CPU which takes 0.50 microseconds as interrupt overhead time when device generates interrupt to the CPU. An I/O device has a bandwidth of 40 KBPS.**

1) Calculate the total time in programmed I/O for 14 bytes data transfer.

2) Calculate the total time required in interrupt I/O for 14 bytes data transfer.

3) What is the estimate of performance gain of I/O transmission using interrupt driven I/O mode over programmed I/O mode?

Sol: **375 μs, 350.50 μs, 1.07**

Data transfer rate = 40 KBPS

Interrupt overhead = 0.50 μsec

1 sec = 40 KB

? = 1 B

$$\text{1 byte transfer time} = \frac{1\text{B}}{40\,\text{KB}}$$

= 0.025 ms

= 25 μsec

1) Programmed I/O:

Programmed I/O time = Status check time + data transfer time

Time = 1B transfer time (default) + data transfer time

= 25 + 14 * 25

= 25 + 350

= 375 μsec

2) Interrupt I/O:

Total time = interrupt overhead = service time

= 0.50 + 14 bytes transfer time

= 0.50 + 14 * 25

= 0.50 + 350

= 350.50 μsec

3) Performance gain (Speed up):

$$\text{Speedup} = \frac{\text{Older technique time}}{\text{New technique time}}$$

$$= \frac{375}{350.50}$$

= 1.07

DMA (Direct memory access):

- Enables data transfer between I/O and memory without the intervention of the CPU.
- To implement the DMA technique, a specific hardware is needed. This hardware is

called DMAC (DMA Controller).

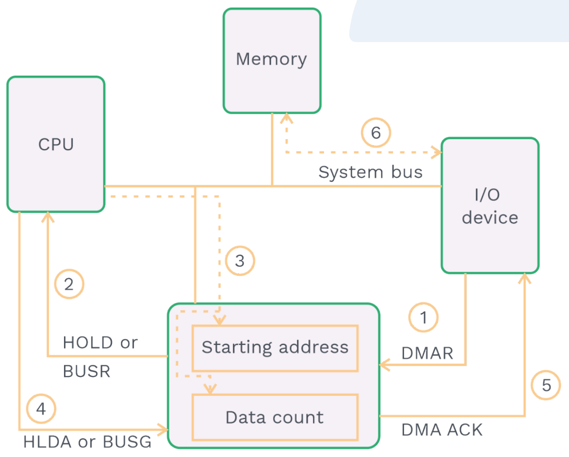


Fig. 6.26 Direct Memory Access

Steps to transfer the data between I/O and memory:

1) I/O device sends the DMAR (DMA request) to DMAC.

2) DMA controller sends Hold signal or BUSR (Bus request) to the CPU. Hold is nothing, but DMAC wants to hold the system bus control to send the data between memory and I/O.

3) If the CPU grant the HOLD, then the CPU will send the starting address and data count.
Starting address: Memory address, starting from where data is to be transferred.
Data count: Number of bytes or words to be transferred.

4) CPU sends the HLDA (Hold acknowledge) signal to the DMA controller.

5) DMA controller sends the DMA ACK signal to I/O device. We can say that DMAC is ready to transfer the data by the system bus.

6) Data is being transferred between I/O and memory through buses only. How does the starting address and data count value change?

		After 1 byte	**After 2 byte**	**.......**		**After 100 byte**
Starting address	2000	2001	2002		2099	2100
Data count	100	99	98		1	0

Table 6.3

- When the data count becomes 0, DMA transfer stops sending the data.
- DMA is a special purpose processor which is used to transfer the data between memory and I/O. Special purpose processor means DMAC can generate the address for memory, and it can generate control signals for both memory and I/O.

Modes of DMA transfer:

There are three modes of DMA transfer

1) Burst Mode
2) Cycle Stealing Mode
3) Interleaving Mode

1) Burst mode:

In the burst mode, when DMAC gets the system bus control, a burst of data is transferred at one time then the CPU takes back the control. So that the CPU can continue the execution, the CPU will be blocked during burst (longer data) transfer.

2) Cycle stealing mode:

In the cycle stealing mode, I/O device takes some time to prepare word to be transferred. During word preparation time. CPU keep control of the buses. When the word is ready to transfer DMAC get the control of the buses for '1 cycle' in that only the prepared word is transferred to memory. After 1 cycle of time CPU takes back the control of the buses. CPU will be blocked during the word transfer.

3) Interleaving mode:

In this mode, when the CPU is busy with internal operations, or it does not need the buses, then, only the CPU gives the control of the buses to DMAC. CPU will not be blocked at all 0% CPU blocked during DMAC.

Assume,

- I/O device takes time to prepare the data = T_P
- Data transferred time to memory = T_T

$$\boxed{\%\,\text{time CPU is blocked} = \frac{T_T}{T_P + T_T} * 100}$$

CPU is blocked when the data is transferred because DMAC has control of the buses.

Previous Years' Question

Question: Consider a disk drive with the following specification: 16 surfaces, 512 tracks/surface, 512 sectors/track, 1 KB/sector, rotation speed 3000 rpm. The disk is operand in cycle stealing mode whereby whenever one byte word is ready it is sent to memory ; similarly, for writing, the disk interface reads a 4 byte word from the memory in each DMA cycle. Memory cycle time is 40 nsec. The maximum percentage of time that CPU gets blocked during DMA operation is:

a) 10 **b)** 25 **c)** 40 **d)** 50

Sol: b) **(GATE CS-2005)**

PRACTICE QUESTIONS

Q12 Assume a DMA controller that supports 6-bit count register. It is interfaced with a 32-bit CPU. A file of 1 KB size needs to be transferred to main memory using cycle steading mode. How many DMA cycles are required to complete the transmission?

a) 2 b) 8 c) 6 d) 4

Sol: d)

DMA supports 6-bit count register

$\Downarrow$

64 words an be transmitted to main memory in 1 DMA cycle

$\Downarrow$

64 × 4B can be transmitted to main memory in 1 DMA cycle

$$\therefore \text{ Number of DMA cycles required} = \frac{1\text{KB}}{64 \times 4\text{B}} = \frac{2^{10}}{2^{8}} = 2^{2} = 4$$

Q13 Consider 256 KBPS I/O device interfaced with 32-bit CPU using DMA module. Data is hot transferred to main memory until 64 word data is available is buffer machine cycle time is 2 μseconds. How much time will the DMA take to transfer 128 KB file to main memory? ________ (in milliseconds)

Sol: 1.024 ms

Amount of data to be available in buffer = 64 word for transmission in 1 DMA cycle

Word size of CPU = 32 bit = 2^{5} bits

File size = 128 KB = 2^{7+10+3} bit = 2^{20} bits

$$= \frac{2^{20}}{2^{5}} \text{ words}$$

$$= 2^{15} \text{ words}$$

$$\text{Number of DMA cycle required} = \frac{\text{File size}}{\text{Data transferred in 1DMA cycle}}$$

$$= \frac{2^{15}}{2^6} = 2^9$$

$\therefore$ Total time taken = Number of DMA × Machine cycle time
= 2^9 × 2 microseconds
= 2^{10} microseconds
= 1024 microseconds
= 1.024 milliseconds

Q14 **Consider 10 KBPS IO device interfaced to 32-bit CPU as cycle stealing mode of DMA. Whenever 16 word data is available in the buffer, it is transferred to main memory. Machine cycle time is 100 μs. What proportion of CPU time is consumed in this operation.**
a) 20% **b) 25%** **c) 80%** **d) 75%**

Sol: **a)**

Data Preparation Time (X):-

$$10\,\text{KB} \xrightarrow{\text{Prepared}} 1\,\text{sec}$$

$$1\,\text{W}\,(4\text{B}) \xrightarrow{\text{Prepared}} \frac{4}{10\,\text{K}}\,\text{sec}$$

$$16\,\text{W} \xrightarrow{\text{Prepared}} \frac{4}{10\,\text{K}} \times 16\,\text{sec}$$

= 6.4 ms

Data Transfer Time (Y):-

1 word data accessing from buffer to main memory happens in 1 machine cycle

$$1 \text{ word data} \xrightarrow{\text{transferred}} 100\ \mu\text{s}$$

$$16 \text{ word data} \xrightarrow{\text{transferred}} 1600\ \mu\text{s} \cong 1.6\ \text{ms}$$

Percentage of CPU time consumed = $\frac{Y}{X+Y} \times 100\% = \frac{1.6}{1.6+6.4} \times 100\%$

$$= \frac{1.6}{8.0} \times 100\%$$

= 20%

Q15 **Consider 16 KBPS IO device interfaced to 64-bit CPU using DMA interface in cycle stealing mode. DMA contains 4-bit count register. Machine-cycle is 2 ms. What are the percentages of block time and busy time of CPU respectively?**

a) 70%, 30% **b) 75%, 25%**

c) 80%, 20% **d) 90%, 10%**

Sol: **c)**

DMA contains 4 bit count register

The word data is transmitted per count

Data available in buffer before transmission = 2^4 words

= 16 words

= 16 × 8 B [1 word = 64-bit = 8 B]

= 2^7 B

= 128 B

Preparation Time (X):-

16 KB data is prepared in 1 sec

$$1\text{B} \longrightarrow \frac{1}{16\text{K}}\,\text{sec}$$

$$128\,\text{B} \longrightarrow \frac{1}{16\text{K}} \times 128\,\text{sec} \cong 8\,\text{ms}\,(\text{milli seconds})$$

Transfer Time (Y):-

1 word data accessing takes place in machine cycle (2 ms)

16 word data accessing will take in 16 × 2 = 32 ms

$\therefore$ Percentage time for which CPU is blocked = $\frac{Y}{X+Y} \times 100\% = \frac{32}{8+32} \times 100\%$

$= \frac{32}{40} \times 100\%$

= 80%

$\therefore$ Percentage of time for which CPU is busy = $\frac{X}{X+Y} \times 100\% = \frac{8}{8+32} \times 100\%$

$= \frac{8}{8+32} \times 100\%$

= 20%

Chapter Summary

Disk:

- Platter of the magnetic disks has both sides recording surfaces (storage surface).
- Getting the desired sector of the track is sequential access.
- Consecutive set of sectors is known as a cluster.
- Storage density varies from sector to sector when the sector have constant capacity.

Disk access time:

- Disk access time is nothing but access time, of 1 sector of the disk.
- To calculate the disk access time seek time, rotational latency and transfer time need to calculate.
- Rotational delay can be calculated in two ways:
  a) If current and target sectors are given.
  b) If current and target sectors are not given then we consider rotational delay as $\left(\frac{\text{Disk rotation time}}{2}\right)$.
- In one disk rotation, 1 track can be transferred.
- One sector transfer time is known as the transfer time of the disk.

Multiple sector access time:

- More than one sector can be accessed. There are two ways to calculate access time.
  a) Sequential access
  b) Random access
- In sequential access, the sectors are in the form of cluster (multiple sector with same track)
- In random access for each sector, need to calculate disk access time.

Cylinder:

- To reduce the seek time, data is stored in the form of a cylinder.
- Number of cylinders present in the disk is the same as the number of tracks on the surface.

Disk addressing:

- To get the sequence number of the sector disk addressing is used.
- Address representation of the disk is <cylinder number, surface number, sector number>. This address format is also known as the triple format of the disk.

I/O organisation

- I/O devices are connected to the CPU with a single bus.
- I/O interface is both software and hardware. Software is a device driver.
- I/O interface is required for good synchronisation, conversion, control the disturb between CPU and I/O and etc.
- There are three ways to connect the CPU to memory and I/O

  1) Separate buses for both memory and I/O.
  2) Common data bus and address bus.
  3) Common data bus, address bus and control bus.

Memory mapped I/O:

- To access the memory and I/O number of instructions and addresses are more.
- Memory wastage is more in memory mapped I/O.

I/O mapped I/O:

- To access memory number of instructions and addresses are more but less to access I/O.
- No wastage of memory.

DMA (Direct memory access):

- Without the interference from the CPU, data can be transferred between I/O and memory with the help of DMA.
- DMA generates an address for memory and also generates control signals for both memory and I/O.

  There are three modes of transfer, including DMA transfer.

  1) In the burst mode, CPU will be blocked during burst or block or longer data transfer.
  2) In cycle stealing mode, CPU will be blocked during word transfer.
  3) In interleaving mode, CPU will not be blocked (0% block).