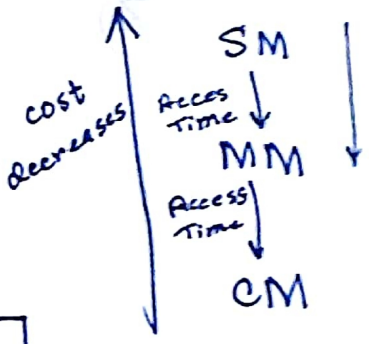
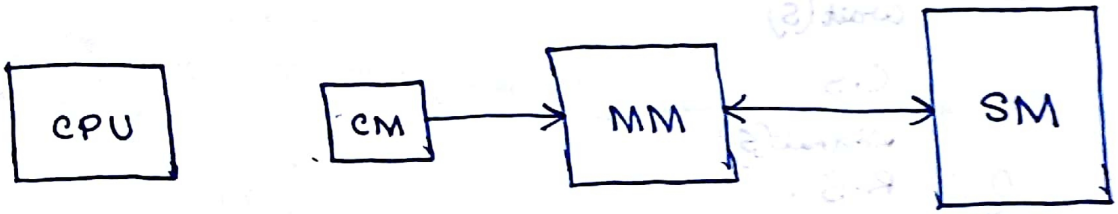
Main Memory

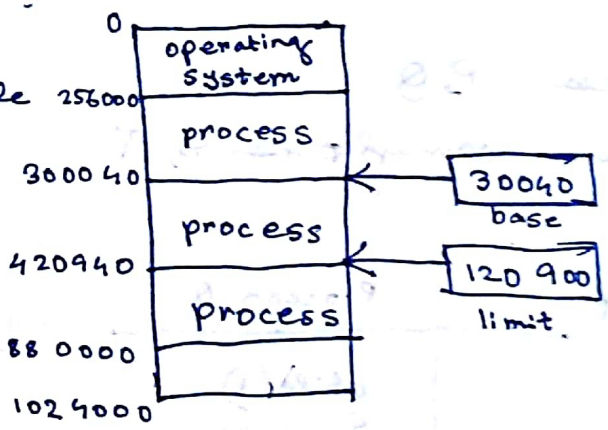




Base and Limit Registers

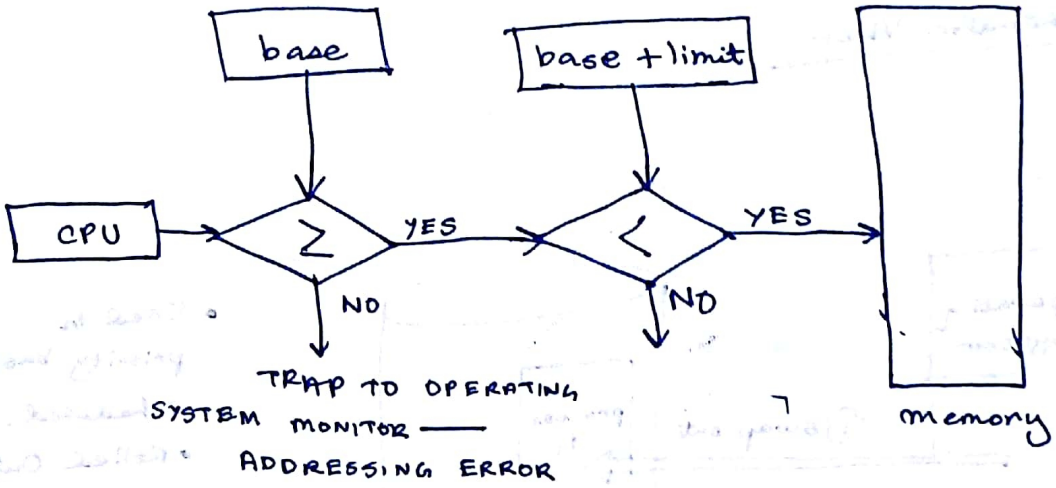
↳ Loaded in Kernal Mode

Base Reg: Holds the starting addres of the first process.



∴ ending address
= content of base reg.
+ content of limit reg.

Hardware Address Protection with Base and Limit Registers



- CPU generates logical address or Virtual address
- Main Memory has physical address
- Size of Virtual Memory is actually equal to the size of ~~main~~ secondary memory.

- Multistep Processing of a User Program — PPT

Dynamic Relocation using a Relocation Register

- The limit reg. is called reallocation register.

- Read PPT

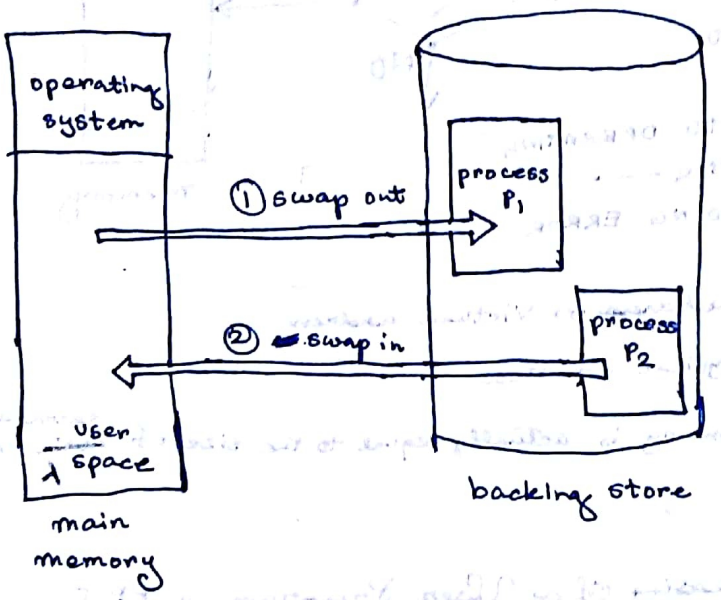
- Dynamic Loading: (Saves memory)

~~Suppose~~ In case of C, only when we call a function, it is loaded into main memory.

Tracking Memory Usage: bitmaps — PPT

Swapping

Schematic View



- Used in priority based scheduled.
- Rolled Out, Roll in

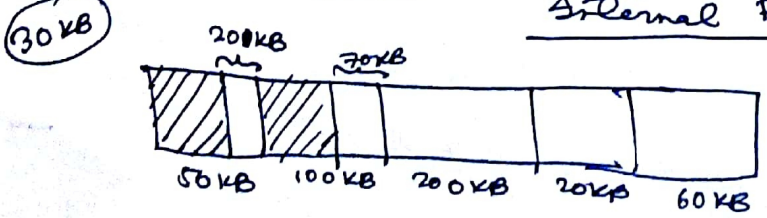
Overlay — PPT

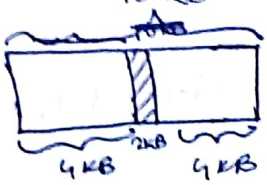
→ Instead of a process, only the instruction to be executed is brought in

~~Contiguous~~

Contiguous Allocation — PPT

- Fixed Size Partition → Suffers from External Fragmentation





We have a total of 5KB space, but we ~~cannot~~ cannot allocate space to P1 as the memory is not continuous here.

This is ~~External Fragg~~ External Fragmentation.

- Variable Partition
- Hole

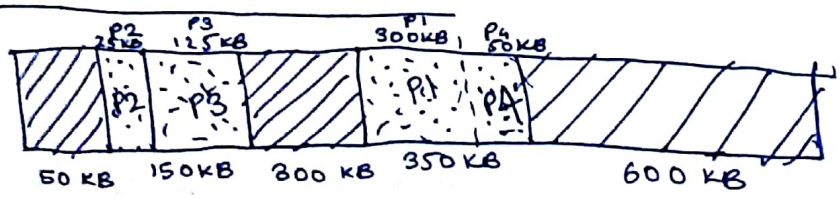
- Compaction / Garbage Collection / Coalescing of Holes — PPT

Memory Placement Strategies

1) First-Fit
2) Best-Fit
3) Worst-Fit

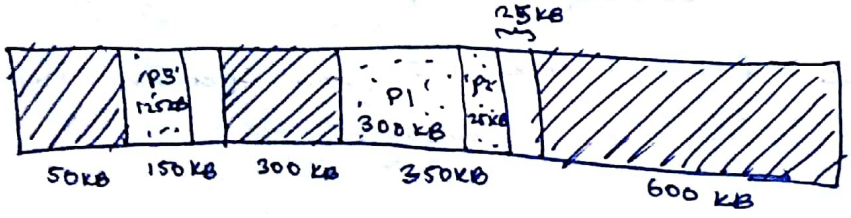
Variable Size Partition

First Fit:



$P_1 = 300$ KB
$P_2 = 25$ KB
$P_3 = 125$ KB
$P_4 = 50$ KB

Best Fit:



- Leaves Smallest Space

P4 cannot be allocated.

Example of External Fragmentation.

- In Variable Size Partition, Best Fit gives the worst performance usually.
- Best Fit is mostly required for

Worst Fit: Gives same result as First-Fit.

- Leaves the largest space

Fixed Size Partition

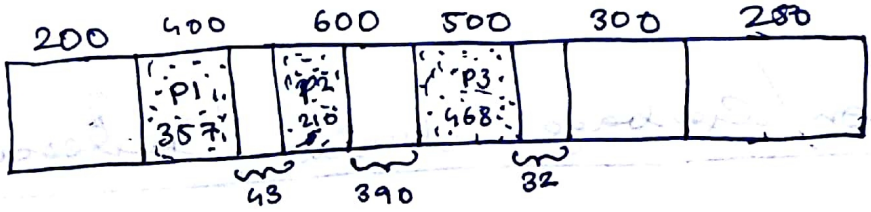
$P_1 = 357$ KB

$P_2 = 210$ KB

$P_3 = 468$ KB

$P_4 = 491$ KB

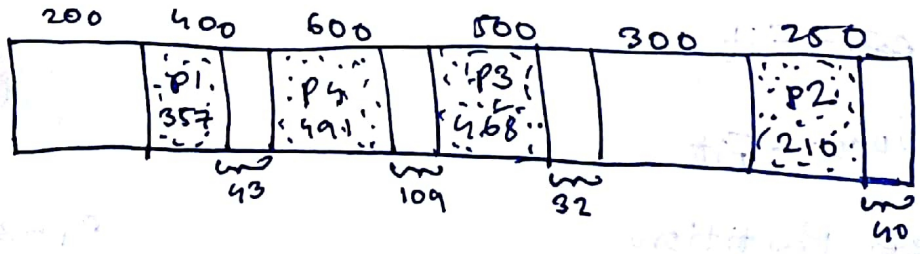
First Fit:



Can't Allocate P4.

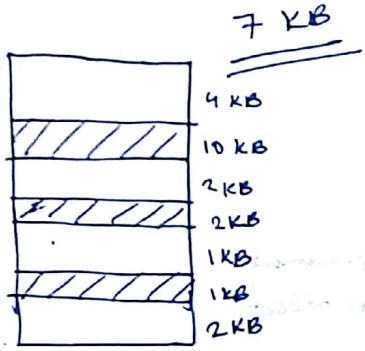
- Suffers from External as well as Internal ~~Fragment~~ Fragmentation.

Best Fit:

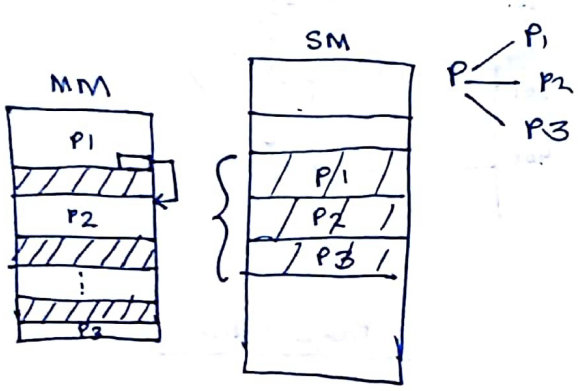


~~Worst~~

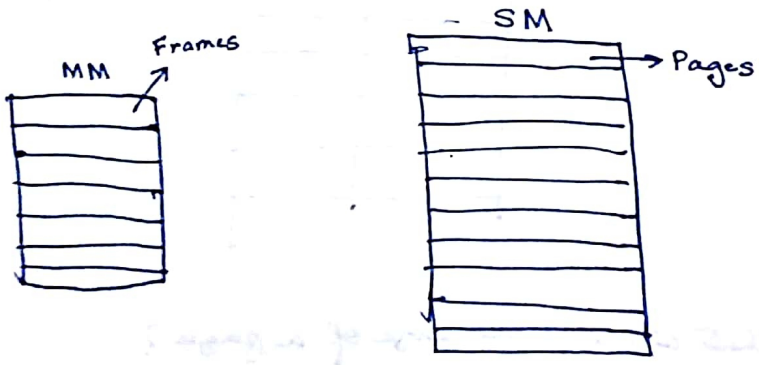
- **Rollout** → Process is taken out from main memory into secondary memory
- **Rollin** → Process is taken out from secondary memory into main memory.



Garbage Collection ~~collection~~ collects all the free spaces ~~so~~ that a process can be allocated space.



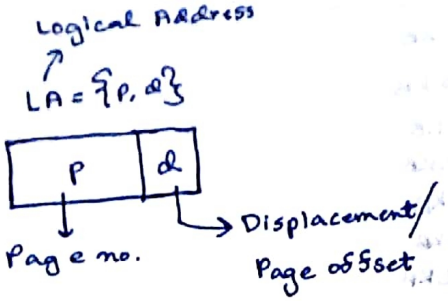
Paging

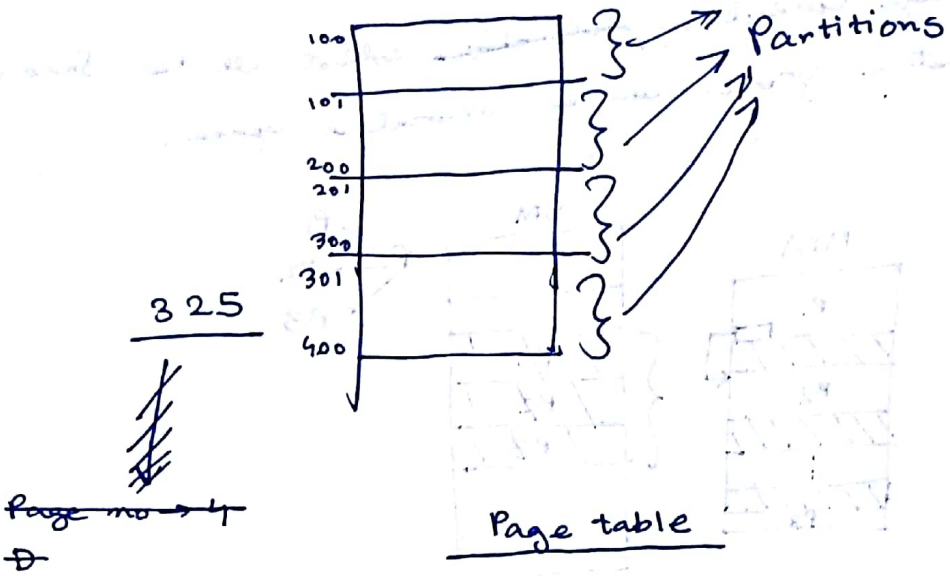


- Size of each page must be equal to size of each frame.
- What is the size of the virtual memory?

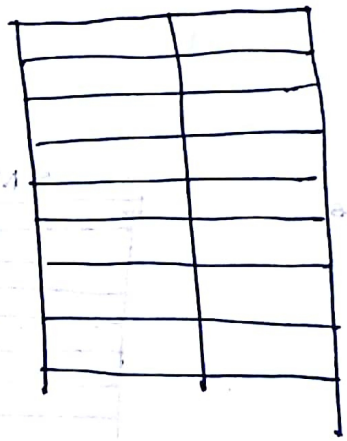
  Actually we cannot predict it. But we assume it to be the size of the secondary memory.

- Datastructure for Paging → Page Tables (it is not a hardware)





Page table



- What will be the size of a page?
  It will always be power of 2.
- What's in a page table entry? — PPT
- Mapping Logical ⇒ physical address

$$4\,KB = 4 \times 1024 = 2^2 \times 2^{10} = 2^{12}$$

- Address Translation architecture – Paging with Direct Mapping

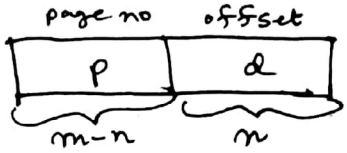
Example (PPT):

- PM = 32 bytes

  pg size = 4 byte

$n = 2$

$m = 4$



$2^m \rightarrow$ logical address space

$2^n \rightarrow$ page addressing unit

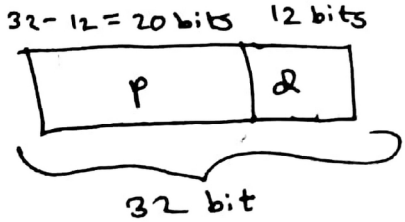
Example:

4KB = (4096) bytes pages

32 bit logical address

$2^d = 4096 \longrightarrow d = 12$

$2^2 \cdot 2^{10} = 2^n$



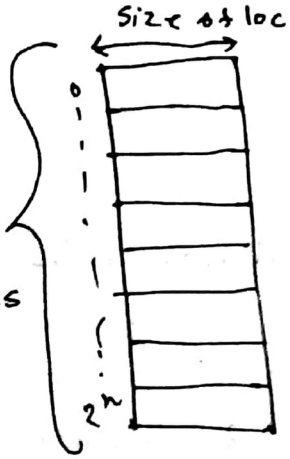
size of memory

= no. of location * size of location

10 bits

2^{10}

$= 1024 \times 1$ B



$2^{10} \times 2^2$

$= 4$ KB

16 bit logical address —

$\therefore 2^{16} \times 1\,B$

$= 2^6 . 2^{10} \times 1\,B$

$= 16\,KB \rightarrow$ Memory size

Memory size → 64 KB

↓

$2^6 . 2^{10} \times 1\,B$

↓

$2^{16} \rightarrow$ 16 bits logical address

$2^{10} = 1\,KB$

$2^{20} = 1\,MB$

$2^{30} = 1\,GB$

$2^{40} = 1\,TB$

$2^{50} = 1\,PB$

Problem:

$n = 2$

$m = 4$

Page size = 4 bytes

PM = 32 bytes

$\therefore$ 8 pages are there.

See PPT

- Paging helps to avoid external fragmentation, but not internal fragmentation.

Example:

page size = 2048 bytes

72,766 bytes

35 pages + 1,086 bytes

(2048 − 1086) = 962 bytes

- Free Frames — PPT

- Implementation of Page Table — PPT

- Paging Hardware with TLB — PPT

TLB is searched first instead of page table. If not hit, then search in page table.

- Effective Access Time

Hit ratio = 80% of TLB

Access TLB → 20 Nanoseconds

→ 100 Nanoseconds Access Memory

For TLB hit = 20 + 100 = 120

For TLB miss = 20 + 100 + 100 = 220

Effective access time

= 0.80 × 120 + 0.20 × 220

= 140 nanoseconds

- Shared Pages Example

- **Reentrant Code** —
  Code does not change during execution.

Hierarchical Page Tables — PPT

Segmentation — PPT

Segmentation Hardware — PPT

Dynamic Address Translation in a Segmentation / Paging System — PPT / Dittel (Book)

- Whenever CPU generates an address, it has 3 parts —
  $(s, p, d) \Rightarrow$ (segment no, page no, displacement)

Virtual Memory

Two types of Paging Policies
1) Demand
2) Predictive / Anticipative

- Demand Paging — PPT
  - **Lazy Swapper** — never swaps a page into memory unless page will be needed
- Valid-Invalid Bit — PPT
- Steps in Handling a Page Fault — PPT
- Performance of Demand Paging — PPT
  - Page Fault Rate
  - Effective Access Time (EAT)
- Predictive Paging (based on Temporal and Spatial

FIFO Page Replacement

1. reference string

* 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

page frames

#	7	0	1	2	0	3	0	4	2	3
	7	7	7	2	2	2	2	4	4	4
		0	0	0	0	3	3	3	2	2
			1	1	1	1	0	0	0	3
miss	m	m	m	m	H	M	M	M	M	M

Miss Rate = $\frac{15}{20} \times 100\% = 75\%$

Hit Rate = $\frac{5}{20} \times 100\% = 25\%$

2. reference string: 1 2 3 4 1 2 5 1 2 3 4 5 — Page frame = 3

	1	1	1	4	4	4	5	5	5	5	5	5
		2	2	2	1	1	1	1	1	3	3	3
			3	3	3	2	2	2	2	2	4	4
	m	m	m	m	m	m	m	H	H	m	m	H

Miss Rate = $\frac{9}{12} \times 100\%$

Hit Rate = $\frac{3}{12} \times 100\%$

- Belady's Anomaly — If we increase the number of page frame, it is also increasing the number of page faults.

No. of Page frames = 4, Reference

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	5	5	5	5	4	4
	2	2	2	2	2	2	1	1	1	1	5
		3	3	3	3	3	3	2	2	2	2
			4	4	4	4	4	4	3	3	3
m	m	m	m	H	H	M	M	m	M	M	M

∴ Miss = 10

Hit = 2

∴ With increase in the number of page frames, the number of miss also increases.

(Belady's Anomaly)

Optimal Page Replacement

Reference String:

1.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7
7	7	7	2		2		2			2		2					7
	0	0	0		0		4			0		0					0
		1	1		3		3			3		1					1

2. Reference string:

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	1	1	1	3	3	3
	2	2	2	2	2	2	2	2	2	4	4
		3	4	4	4	5	5	5	5	5	5
m	m	m	m	H	H	m	H	H	m	m	H

Miss = 7

Hit = 5

- In theory, optimal page replacement gives the least

LRU (Least Recently Used)

1.

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7
	7	7	7	2		2		4	4	4	0			1				
		0	0	0		0		0	0	3	3			3				
			1	1		3		3	2	2	2			2				

2.

Reference String: Page frame = 4

	0	2	1	6	4	0	1	0	3	1	2	1
	0	0	0	0	4	4	4	4	4	4	2	2
		2	2	2	2	0	0	0	0	0	0	0
			1	1	1	1	1	1	1	1	1	1
				6	6	6	6	6	3	3	3	3
	m	m	m	m	m	m	H	H	m	H	m	H

Miss = 8

Hit = 4

3. Optimal Replacement:

	0	2	1	6	4	0	1	0	3	1	2	1
	0	0	0	0	0	0	0	0	0	0	0	0
		2	2	2	2	2	2	2	2	2	2	2
			1	1	1	1	1	1	1	1	1	1
				6	4	4	4	4	3	3	3	3
	m	m	m	m	m	H	H	H	m	H	H	H

Miss = 6

Hit = 6

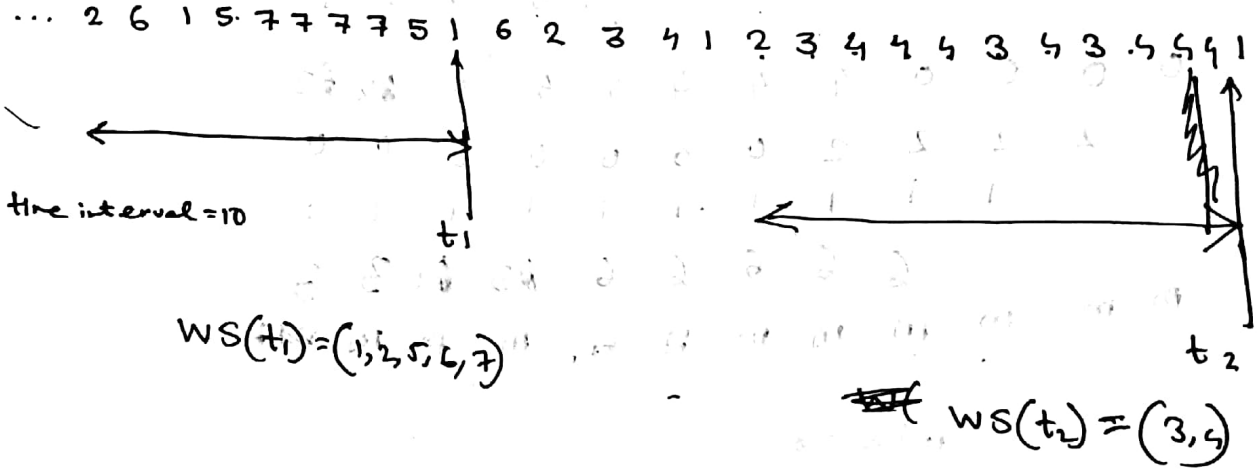
Thrashing

A process is busy swapping pages in and out.

- Thrashing reduces the degree of multiprogramming.
- Low CPU utilization

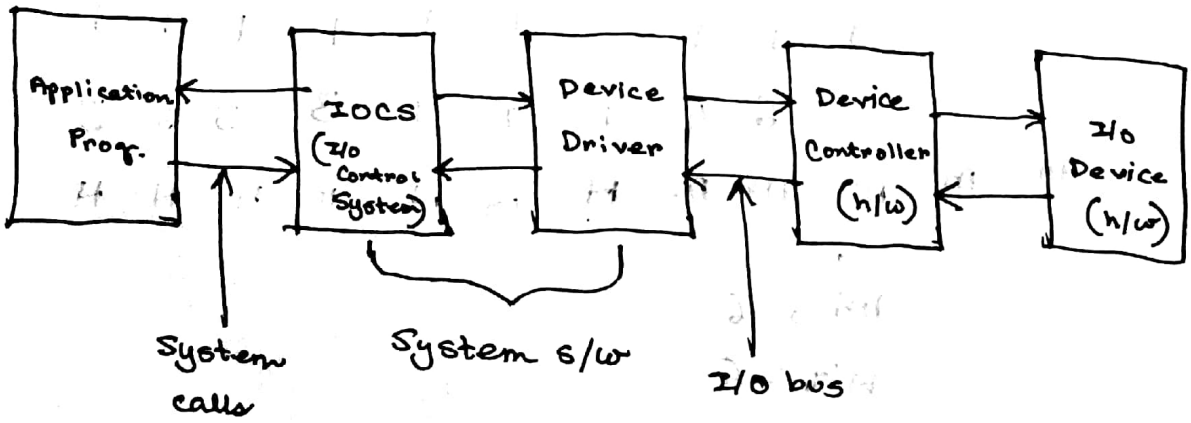
- Locality of Reference Principle — PPT / Book (Diagram)
- Working-Set Model — PPT

Page reference table



Mass Storage Systems

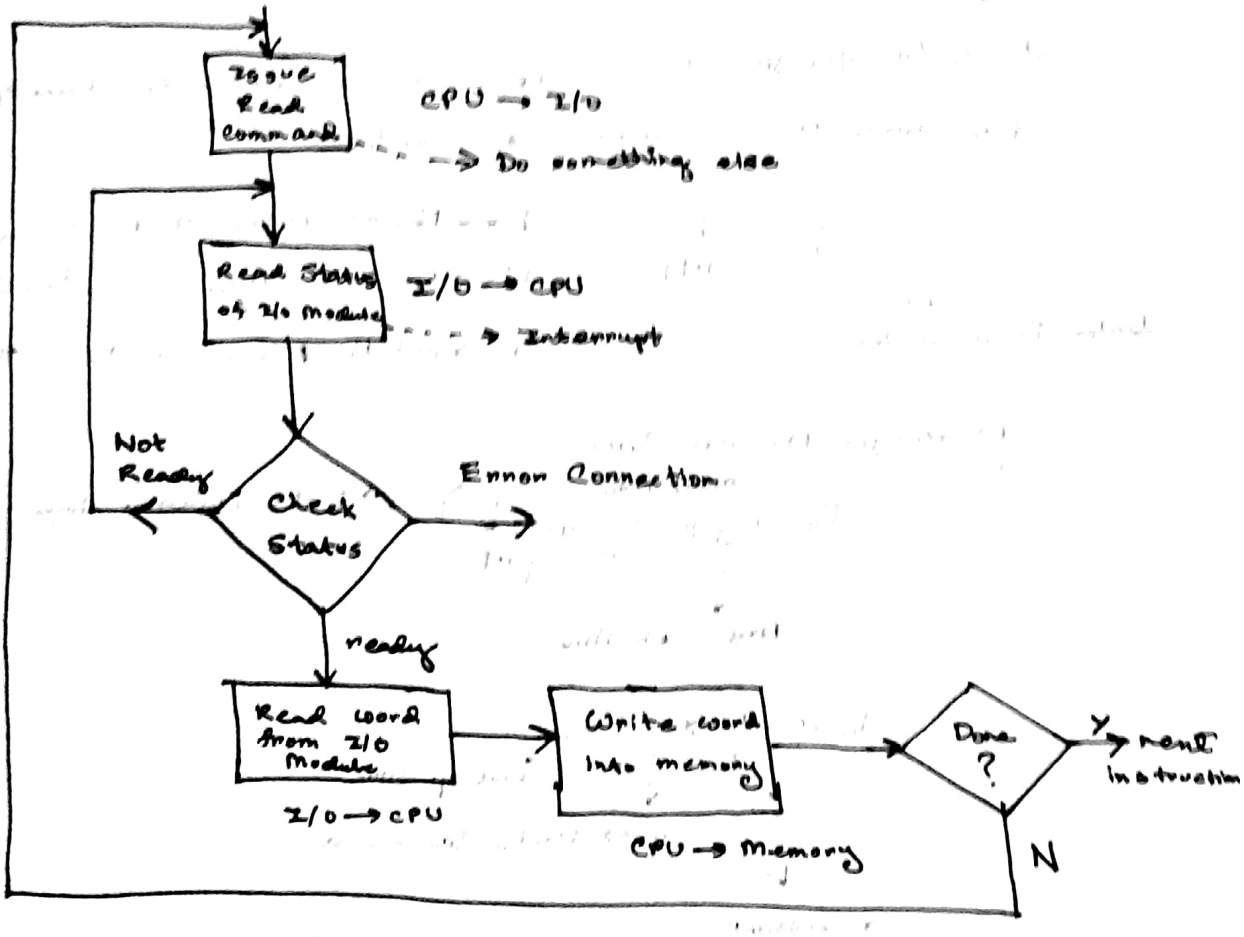
Structure of I/O Device



Organization of the I/O Function — PPT

i) Programmed I/O
ii) Interrupt I/O
iii) DMA Controlled

Programmed I/O



- DMA Controller — Read / PPT

- **Buffering** — PPT
  - — Without Buffer
  - — Single Buffer
  - — Double Buffer
  - — Circular Buffer

- **Disk Performance Parameters** — PPT
  - — Sector, Track, head
    - i) Wait for Device
    - ii) Wait for Channel
    - iii) Seek

- Positioning the Read/Write Heads — PPT

Data transfer time → How long do we take to transf data from one location to another.

$$T = \frac{b}{rN}$$

T → Data Transfer Time

b → No. of bytes to be trans

N → No. of bytes on a track

r → rotational speed in revolu

Average Access Time

$$T_a = \underbrace{T_s + \frac{1}{2r}}_{\text{Access Time}} + \frac{b}{rN}$$

$\frac{b}{rN}$ → Data Transfer Time

T_s → Avg. Seek Time

$$T_s = t' \cdot n + s$$

t' → a constant that depends on disk drive

n → no. of tracks traversed

s → start up time