Main memory:

What is the meaning of physical memory and virtual memory?

Physical memory is the actual real memory used in RAM. Virtual memory as the name suggests is not real. The OS uses virtual memory as a memory management technique in which non-contiguous memory is presented to software as contiguous memory. If the RAM falls short of memory to accommodate more running processes, the OS allocates a portion of your hard drive to act as though it were RAM. That's what is referred to as virtual memory.

Physical memory is the only memory that is directly accessible to the CPU. CPU reads the instructions stored in the physical memory and executes them continuously. The data that is operated will also be stored in physical memory in uniform manner.

Virtual memory is one classification of memory which was created by using the hard disk for simulating additional RAM, the addressable space available for the user. Virtual addresses are mapped into real addresses.

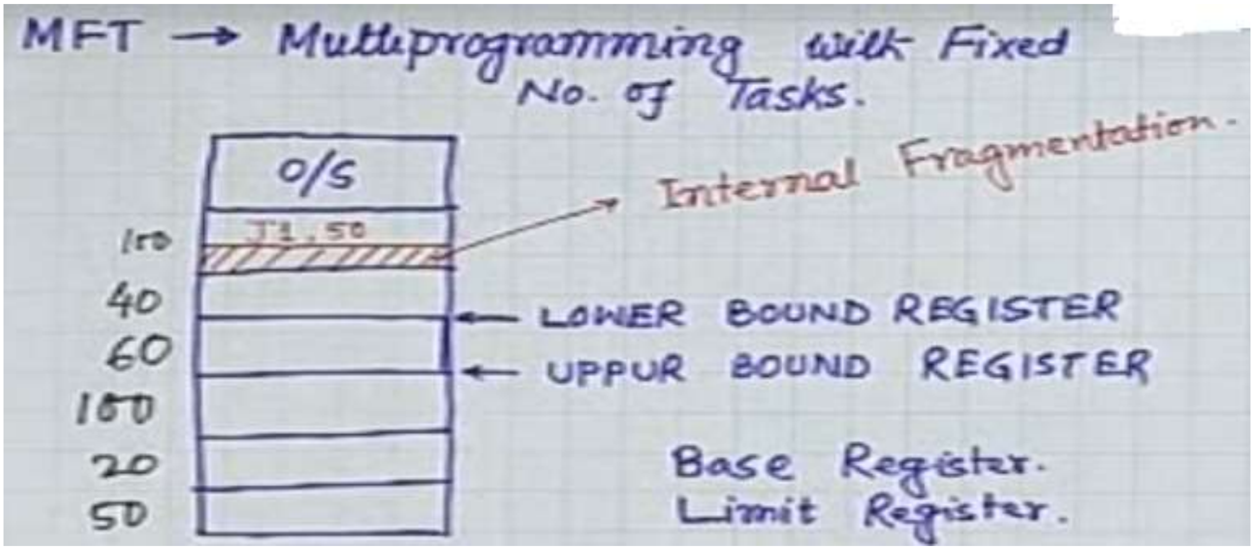
What is page and frame ??

In a memory management system that takes advantage of **paging**, the **OS** reads data from secondary storage in blocks called pages, all of which have identical size. The physical region of memory containing a single page is called a frame.

What is paging? Why paging is used?

OS performs an operation for storing and retrieving data from secondary storage devices for use in main memory. Paging is one of such memory management scheme. Data is retrieved from storage media by OS, in the same sized blocks called as pages. Paging allows the physical address space of the process to be non-contiguous. The whole program had to fit into storage contiguously.

Paging is to deal with external fragmentation problem. This is to allow the logical address space of a process to be noncontiguous, which makes the process to be allocated physical memory.



What is fragmentation in memory management?

As processes are loaded and removed from **memory**, the free **memory** space is broken into little pieces. It happens after sometimes that processes cannot be allocated to **memory** blocks considering their small size and **memory** blocks remains unused. This problem is known as **Fragmentation**.

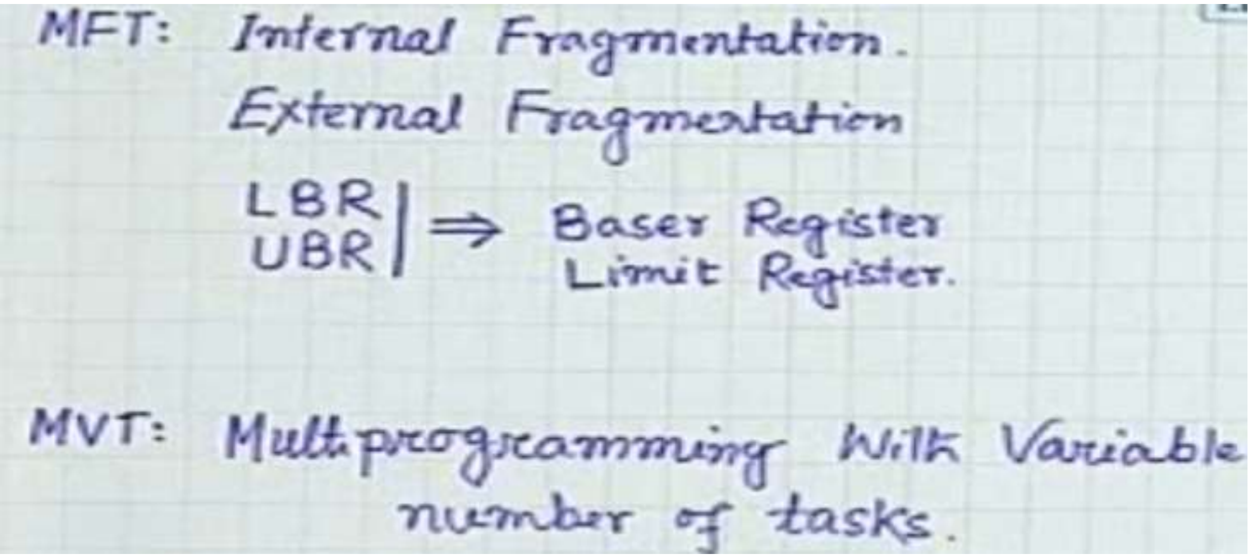
Fragmentation occurs in a dynamic memory allocation system when most of the free blocks are too small to satisfy any request. It is generally termed as inability to use the available memory.

What is external fragmentation & **internal fragmentation**?

External fragmentation- Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous so it cannot be used.
Internal **fragmentation-** Memory block assigned to process is bigger. Some portion of memory is left unused as it can not be used by another process.

In such situation processes are loaded and removed from the memory. As a result of this, free holes exists to satisfy a request but is non contiguous i.e. the memory is fragmented into large number of small holes. This phenomenon is known as **External Fragmentation.**

Also, at times the physical memory is broken into fixed size blocks and memory is allocated in unit of block sizes. The memory allocated to a space may be slightly larger than the requested memory. The difference between allocated and required memory is known as **Internal fragmentation** i.e. the memory that is internal to a partition but is of no use.

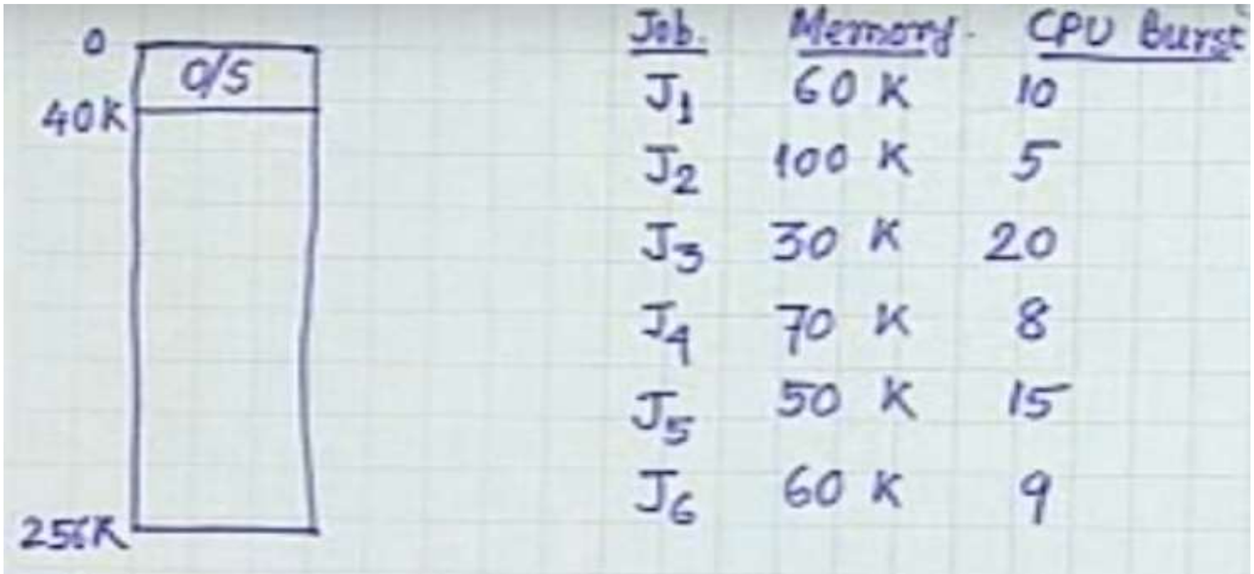


MFT: Internal Fragmentation.

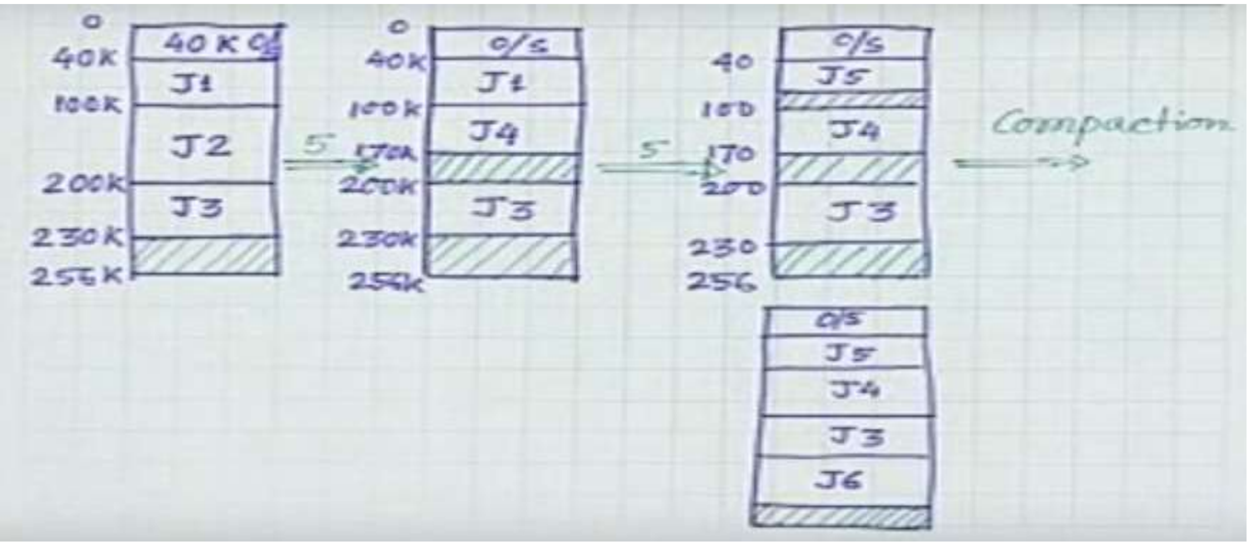
External Fragmentation

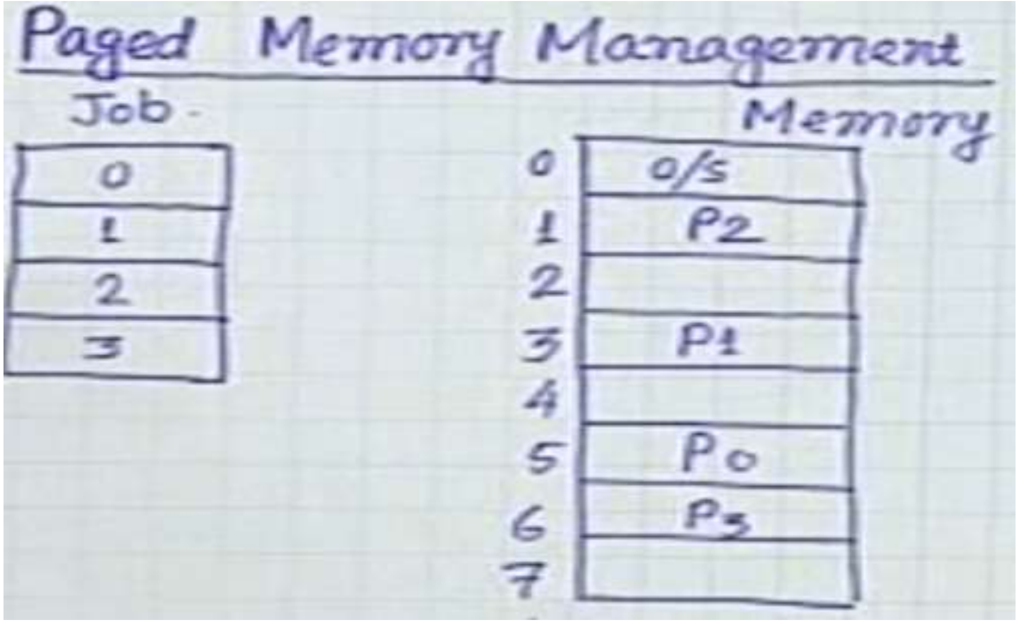
LBR | ⇒ Baser Register
UBR | ⇒ Limit Register.

MVT: Multiprogramming With Variable number of tasks.



Job.	Memory.	CPU Burst
J_1	60 K	10
J_2	100 K	5
J_3	30 K	20
J_4	70 K	8
J_5	50 K	15
J_6	60 K	9





Paged Memory Management

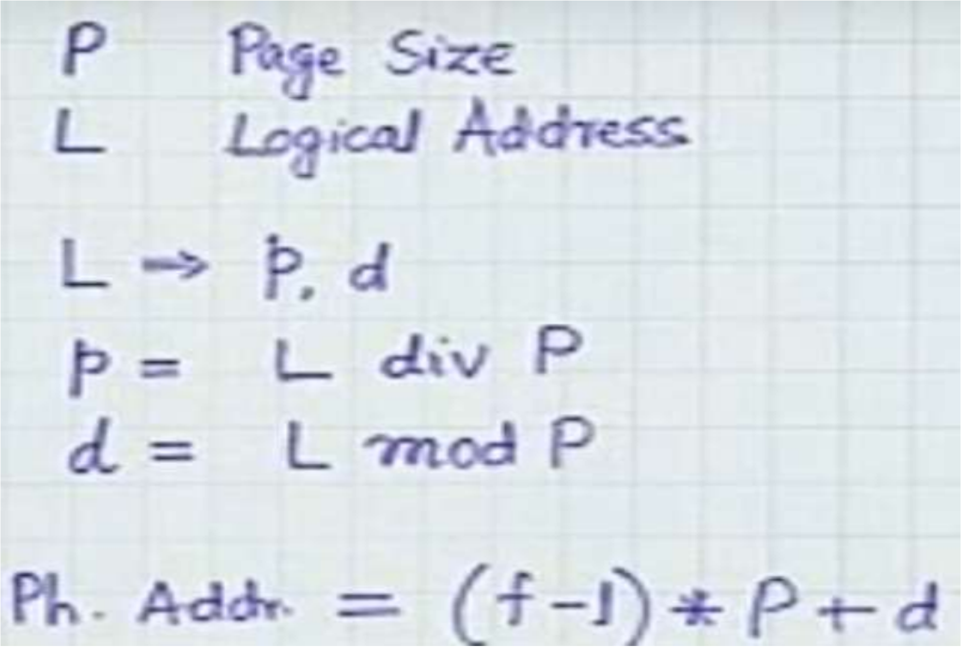
Job

0
1
2
3

Memory

0	O/S
1	P2
2	
3	P1
4	
5	P0
6	P3
7	

Logical address $P_0=5$, $P_1=12$, $P_2=10$, $P_3=33$; Page size P=10



P Page Size

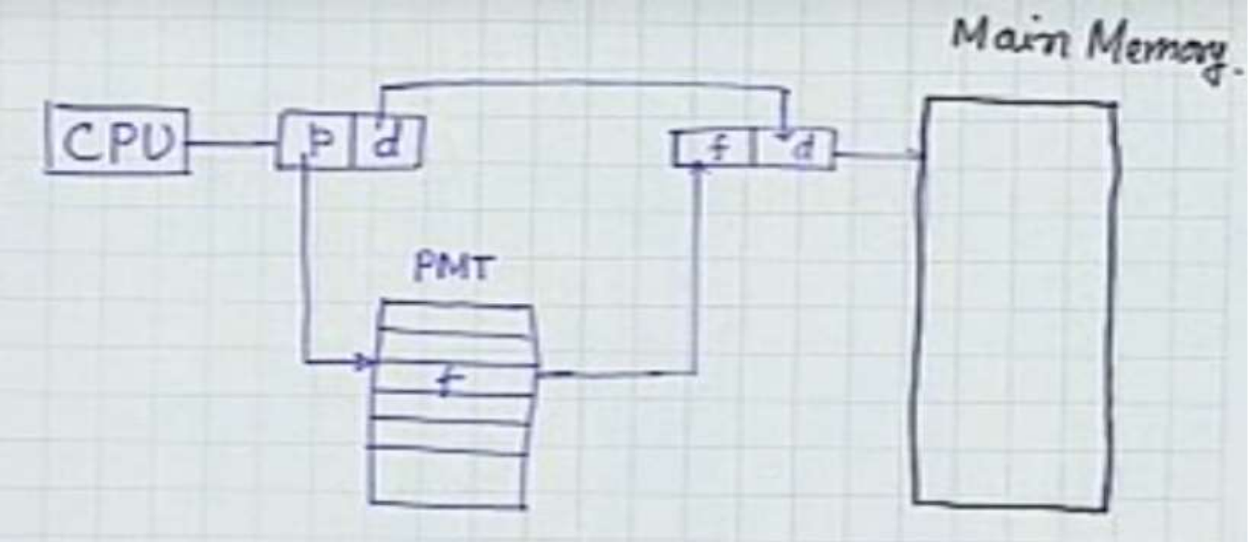
L Logical Address

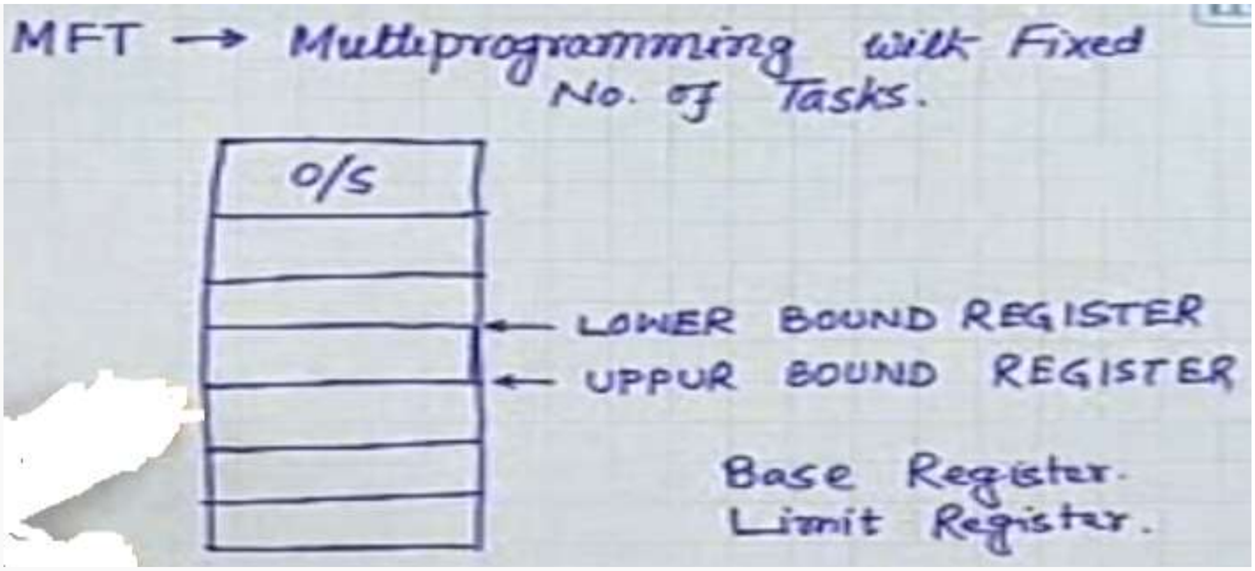
$L \rightarrow p, d$

$p = L \text{ div } P$

$d = L \text{ mod } P$

$\text{Ph. Addr.} = (f-1) * P + d$





Differentiate between logical and physical address.

Logical Vs physical address space

(1) An address generated by the CPU is commonly referred to as a logical a logical address. The set of all logical addresses generated by a program is known as logical address space. Whereas, an address seen by the memory unit- that is, the one loaded into the memory-address register of the memory- is commonly referred to as physical address. The set of all physical addresses corresponding to the logical addresses is known as physical address space.

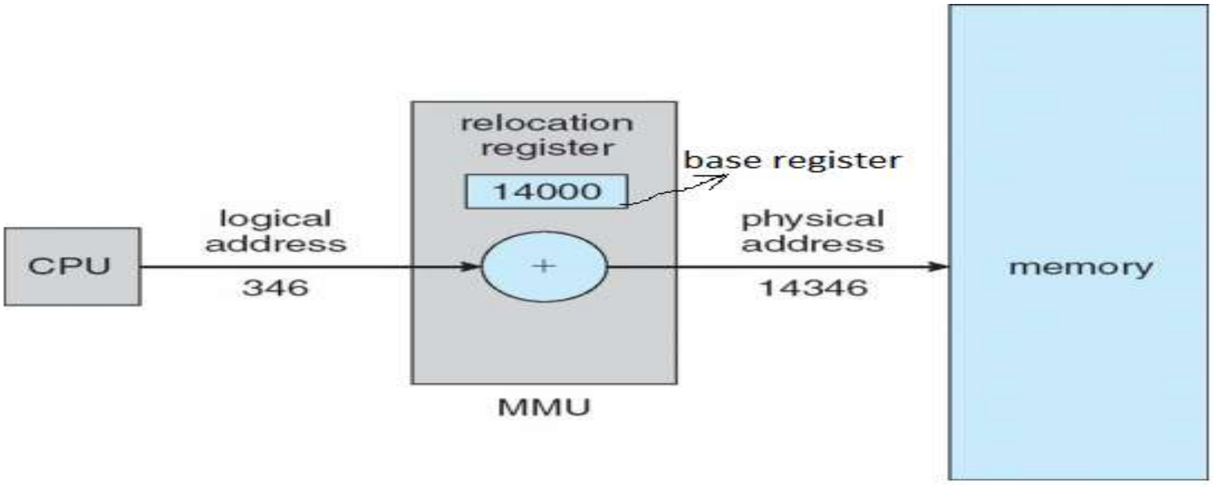
(2) The compile-time and load-time address-binding methods generate identical logical and physical addresses. However, in the execution-time address-binding scheme, the logical and physical-address spaces differ.

(3) The user program never sees the physical addresses. The program creates a pointer to a logical address, says 346, stores it in memory, manipulate it. The memory-mapping hardware device called the memory- management unit(MMU) converts logical addresses into physical addresses.

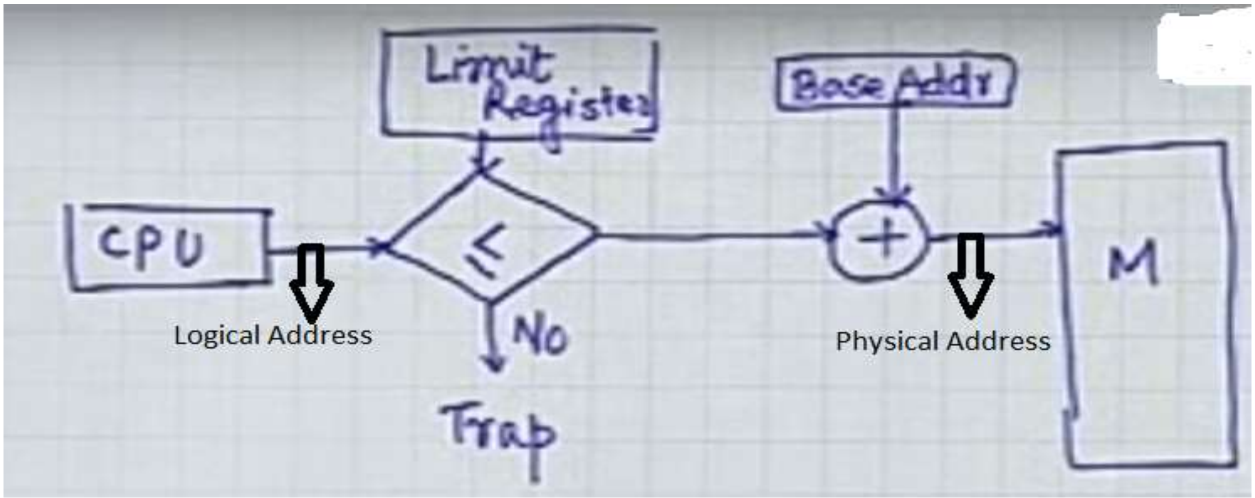
(4) Logical addresses range from 0 to max. User program that generates logical address thinks that the process runs in locations 0 to max.

Logical addresses must be mapped to physical addresses before they are used. Physical addresses range from (R+0) to (R + max) for a base/relocation register value R.

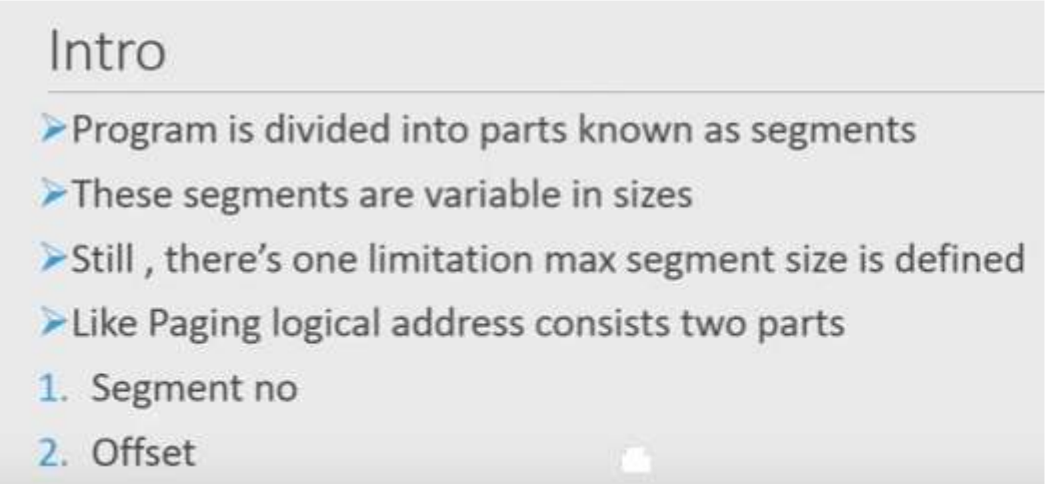
(5) Example:



Mapping from logical to physical addresses using memory management unit (MMU) and relocation/base register
The value in relocation/base register is added to every logical address generated by a user process, at the time it is sent to memory, to generate corresponding physical address. In the above figure, base/ relocation value is 14000, and then an attempt by the user to access the location 346 is mapped to 14346.



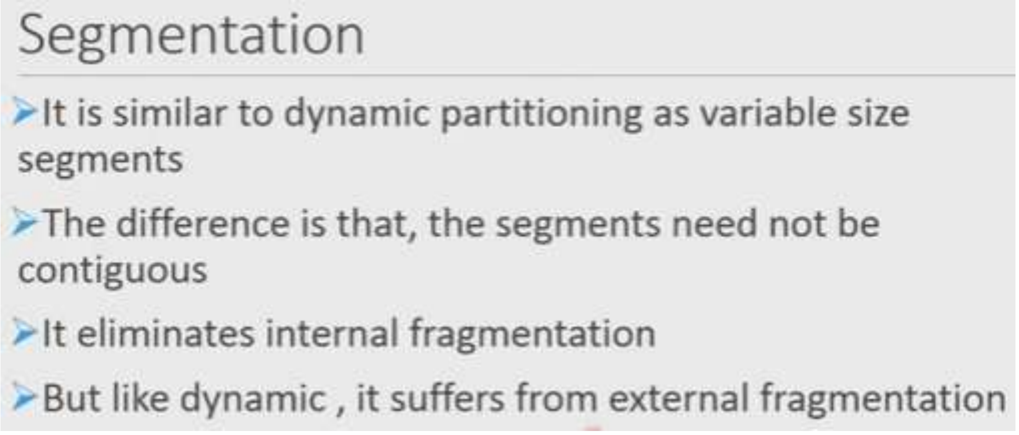
Segmentation:



Intro

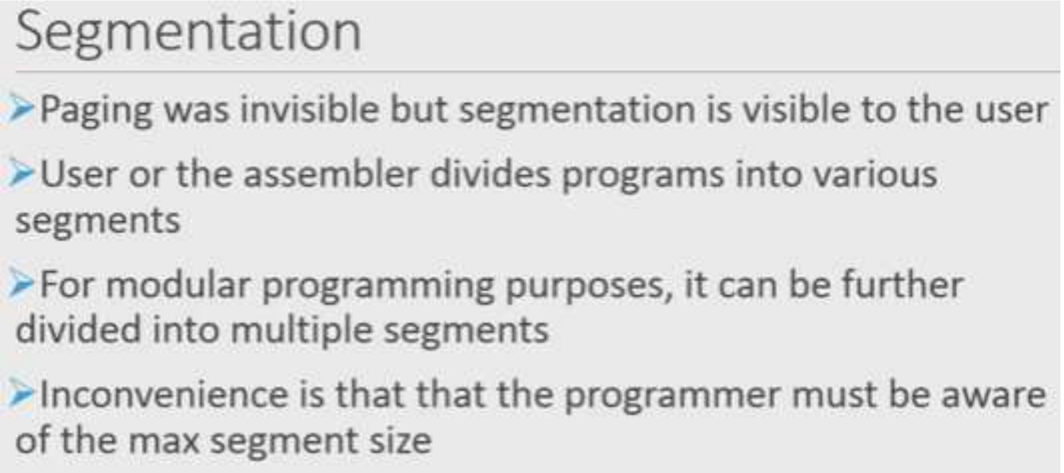
- Program is divided into parts known as segments
- These segments are variable in sizes
- Still , there's one limitation max segment size is defined
- Like Paging logical address consists two parts

1. Segment no
2. Offset



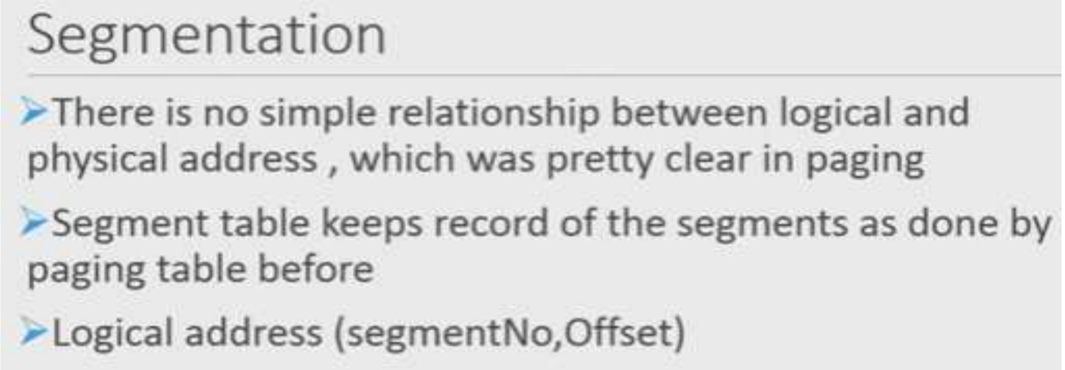
Segmentation

- It is similar to dynamic partitioning as variable size segments
- The difference is that, the segments need not be contiguous
- It eliminates internal fragmentation
- But like dynamic , it suffers from external fragmentation



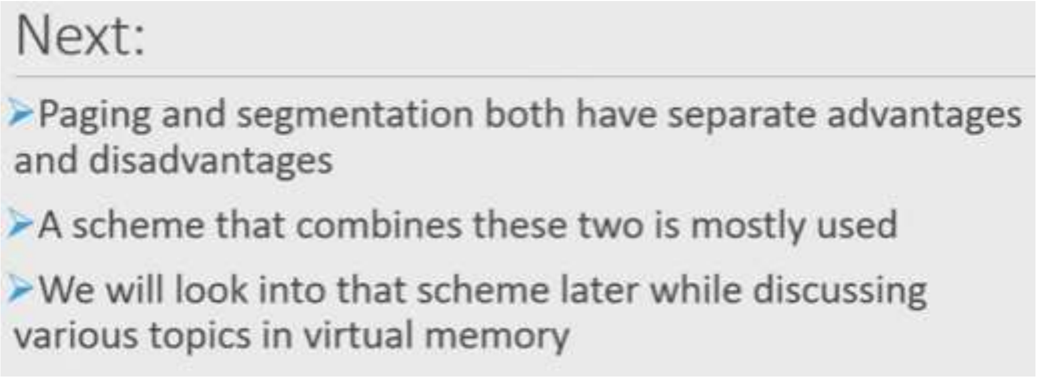
Segmentation

- Paging was invisible but segmentation is visible to the user
- User or the assembler divides programs into various segments
- For modular programming purposes, it can be further divided into multiple segments
- Inconvenience is that that the programmer must be aware of the max segment size



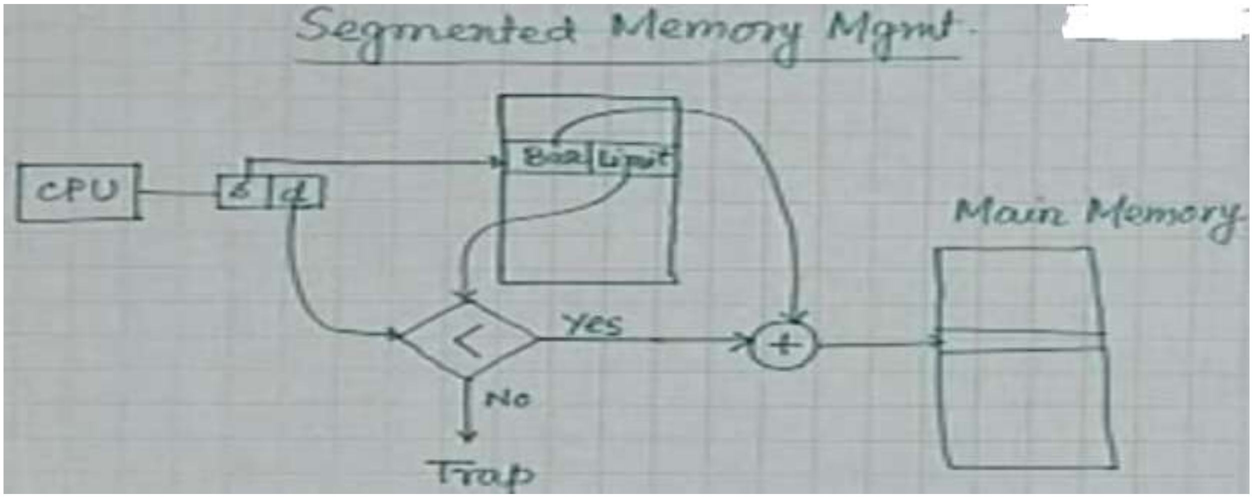
Segmentation

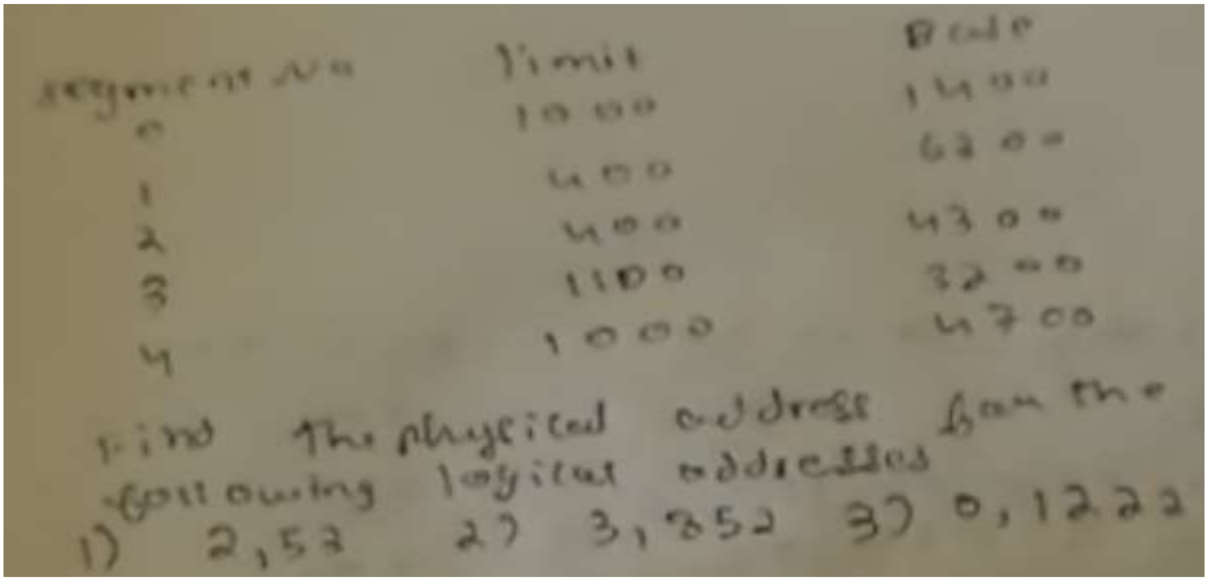
- There is no simple relationship between logical and physical address , which was pretty clear in paging
- Segment table keeps record of the segments as done by paging table before
- Logical address (segmentNo,Offset)



Next:

- Paging and segmentation both have separate advantages and disadvantages
- A scheme that combines these two is mostly used
- We will look into that scheme later while discussing various topics in virtual memory

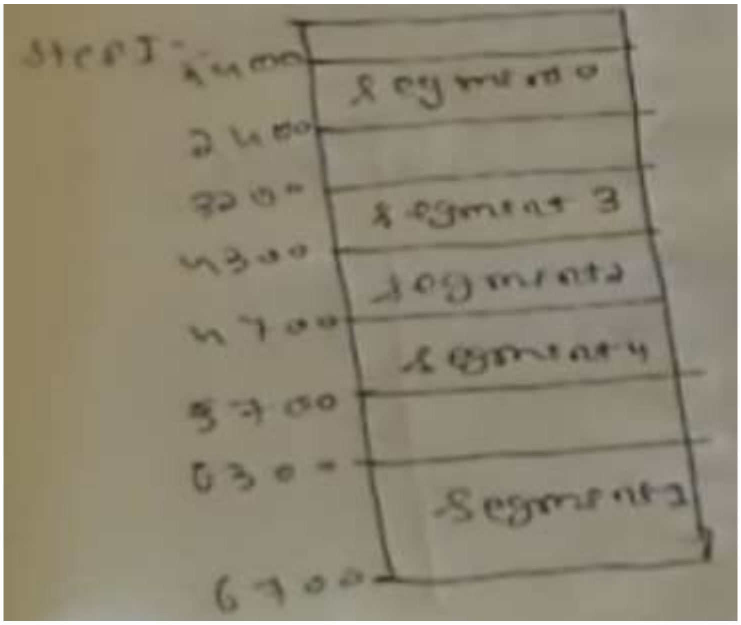




segment No	limit	Base
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

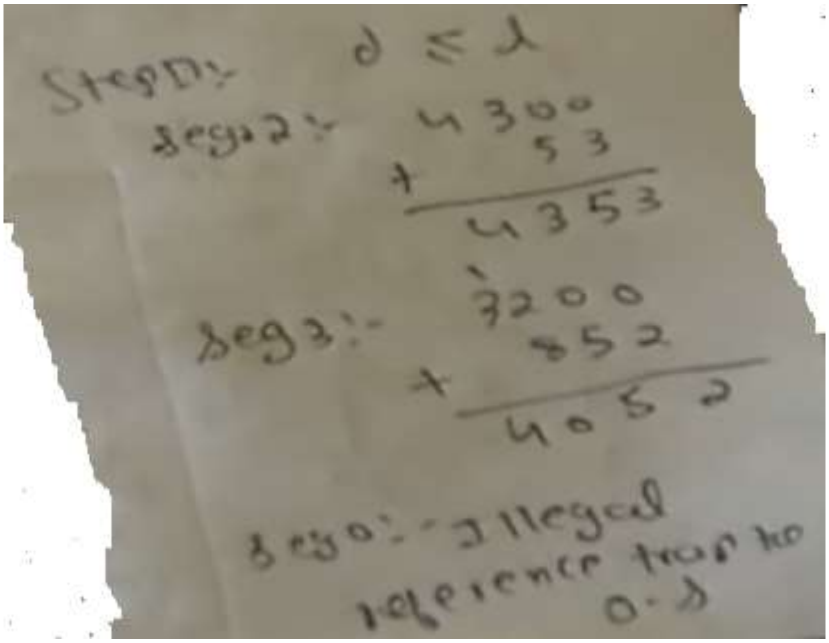
Find the physical address from the following logical addresses

1) 2,53 2) 3,852 3) 0,1222



Step I:-

1400	
	segment 0
2400	
3200	
	segment 3
4300	
	segment 2
4700	
	segment 4
5700	
6300	
	Segment 1
6700	



Step II:- $d \leq l$

seg 2:- $4300 + 53 = 4353$

seg 3:- $3200 + 852 = 4052$

seg 0:- Illegal reference trap to O.S

What is meant by contiguous memory allocation?

Contiguous memory allocation is a classical **memory allocation** model that assigns a process consecutive **memory** blocks (that is, **memory** blocks having consecutive addresses). **Contiguous memory allocation** is one of the oldest **memory allocation** schemes. When a process needs to execute, **memory** is requested by the process.

Memory Allocation Scheme:

→Fixed partition

→Dynamic partition

Dynamic Storage Allocation Problem:

How to satisfy a request of size ***n*** from a list of free holes?

n **First-fit**: Allocate the ***first*** hole that is big enough

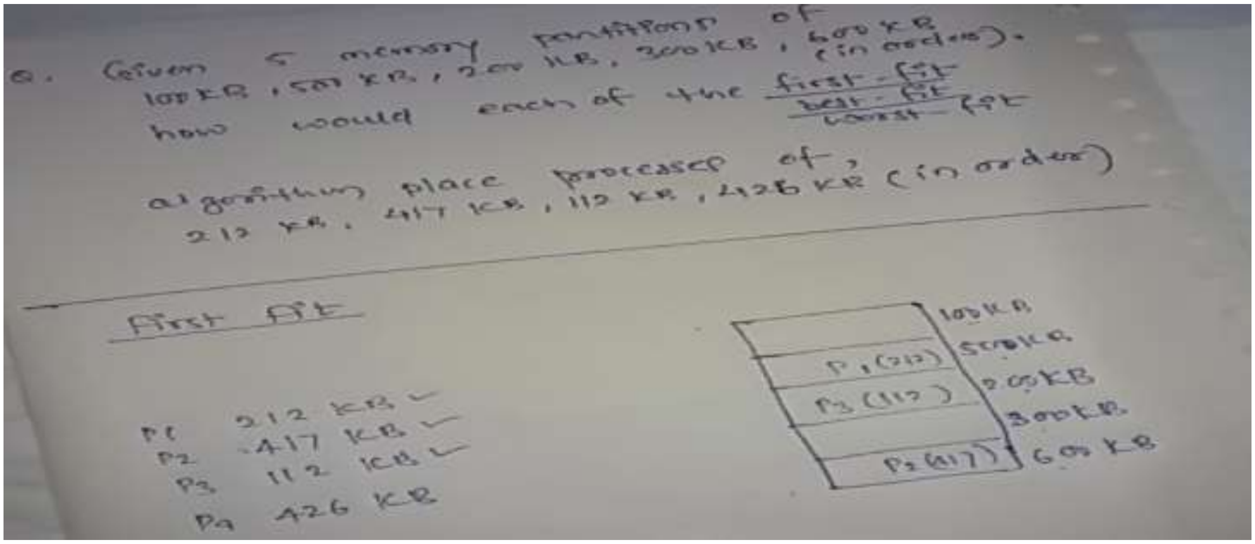
n **Best-fit**: Allocate the ***smallest*** hole that is big enough; must search entire list, unless ordered by size

l Produces the smallest leftover hole

n **Worst-fit**: Allocate the ***largest*** hole; must also search entire list

l Produces the largest leftover hole

n First-fit and best-fit better than worst-fit in terms of speed and storage utilization

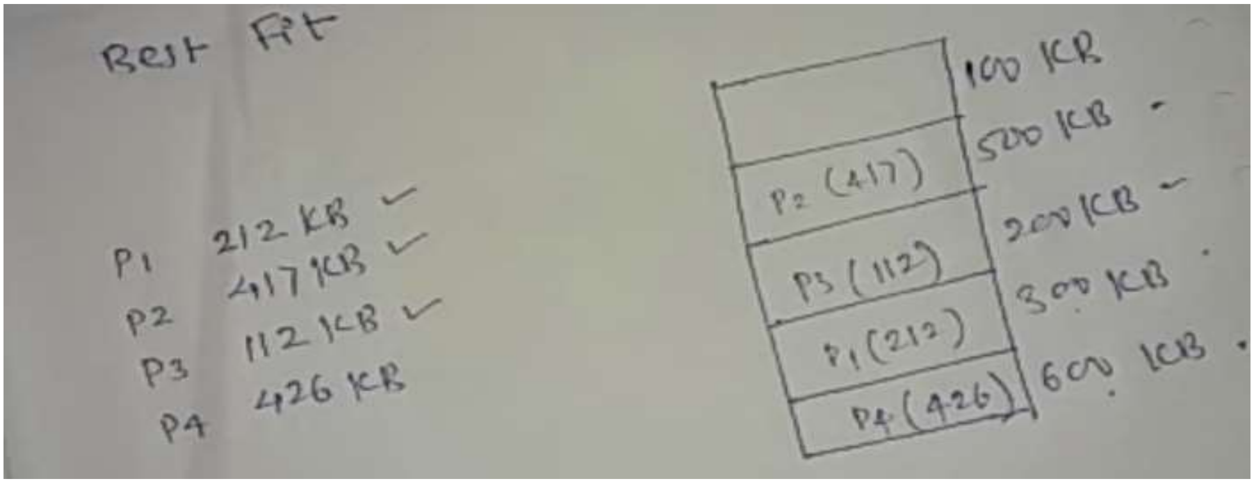


Q. Given 5 memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, 600 KB (in order). how would each of the first-fit / best-fit / worst-fit algorithm place processes of, 212 KB, 417 KB, 112 KB, 426 KB (in order)

First Fit

P1 212 KB ✓
P2 417 KB ✓
P3 112 KB ✓
P4 426 KB

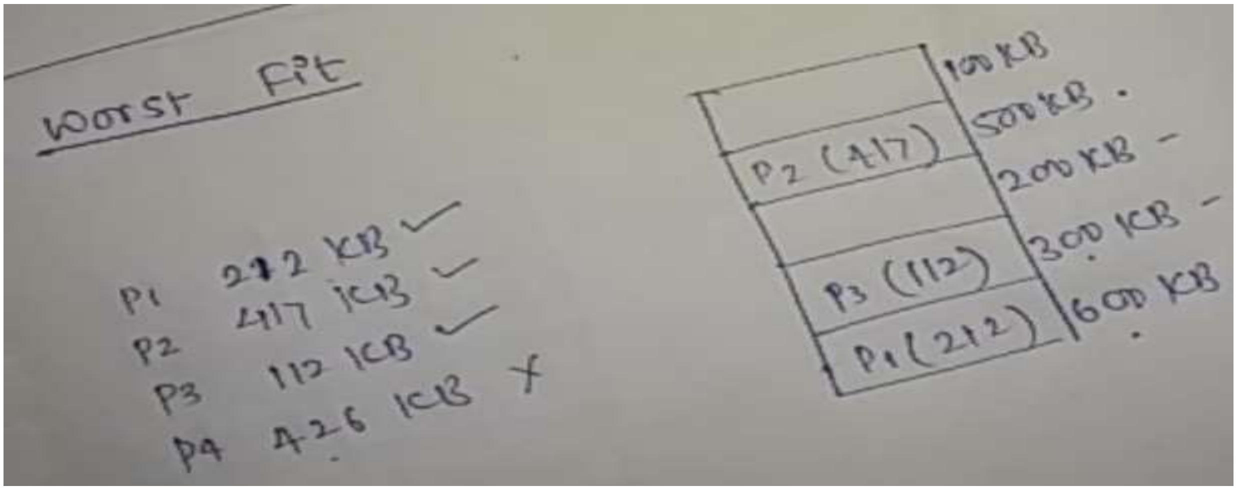
	100 KB
P1 (212)	500 KB
P3 (112)	200 KB
	300 KB
P2 (417)	600 KB



Best Fit

P1 212 KB ✓
P2 417 KB ✓
P3 112 KB ✓
P4 426 KB

	100 KB
P2 (417)	500 KB ·
P3 (112)	200 KB -
P1 (212)	300 KB ·
P4 (426)	600 KB ·

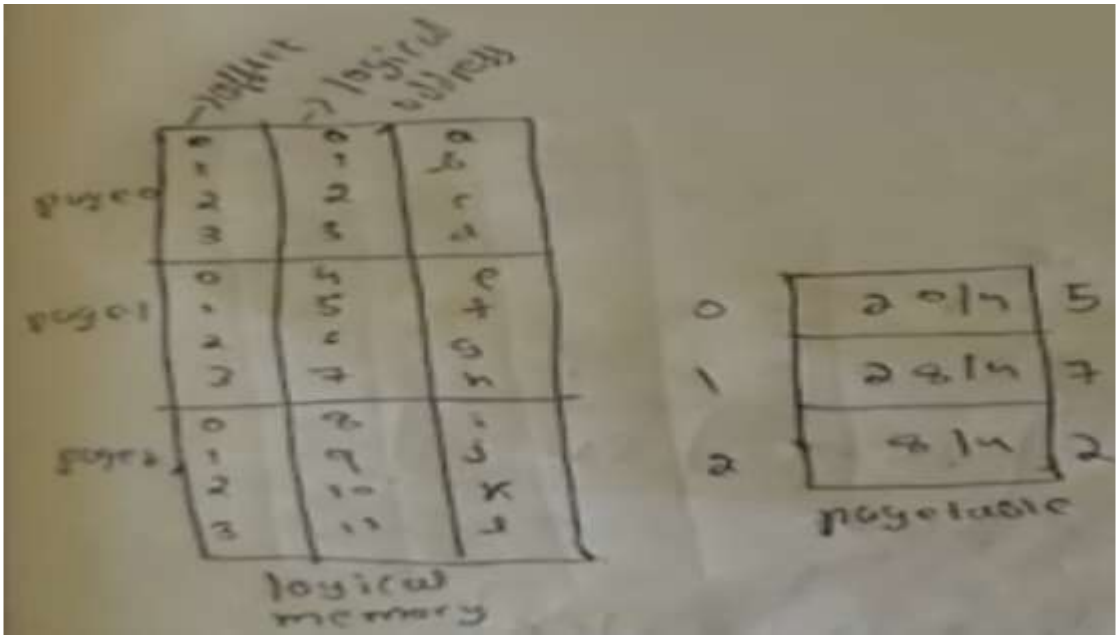


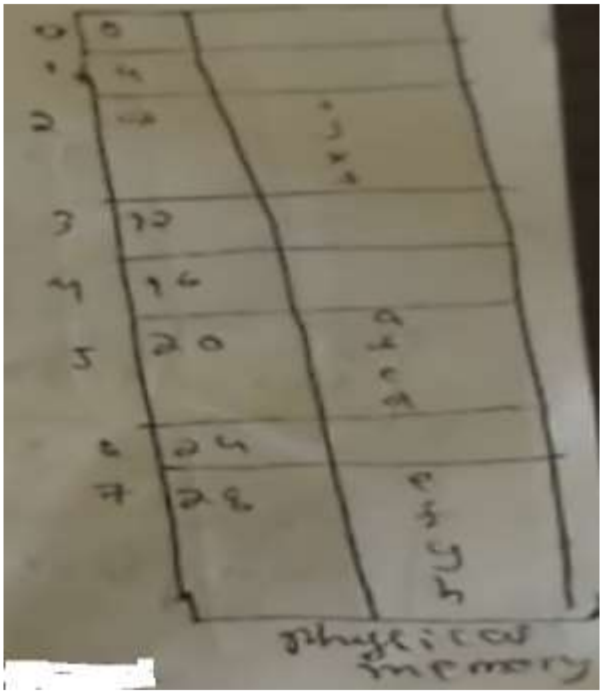
Worst Fit

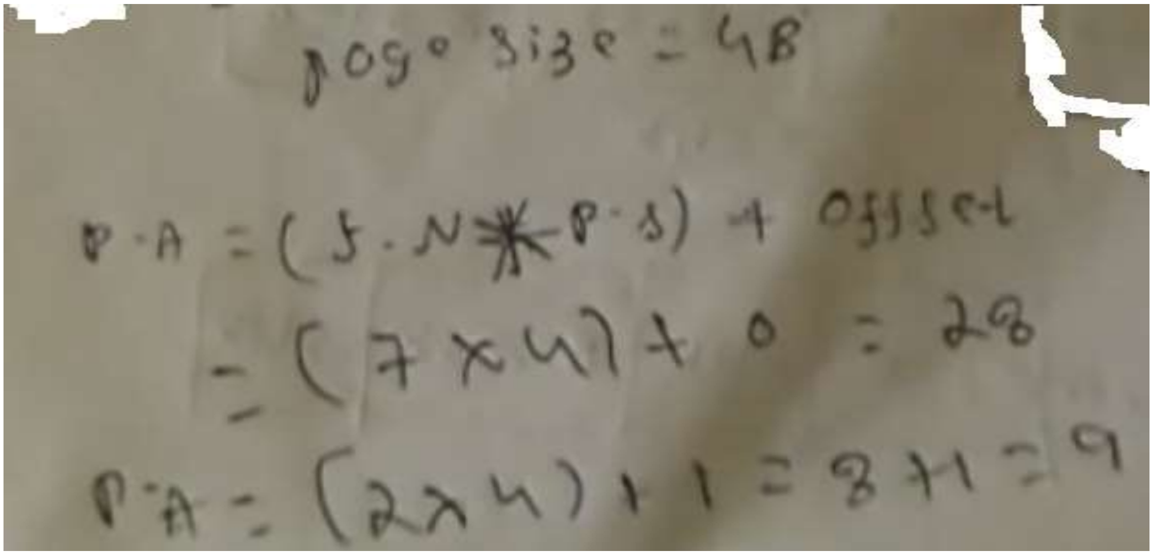
P1 212 KB ✓
P2 417 KB ✓
P3 112 KB ✓
P4 426 KB ✗

	100 KB
P2 (417)	500 KB ·
	200 KB -
P3 (112)	300 KB -
P1 (212)	600 KB

Physical Address Finding by page frame:







page size = 4B

$P \cdot A = (f \cdot N * P \cdot S) + offset$

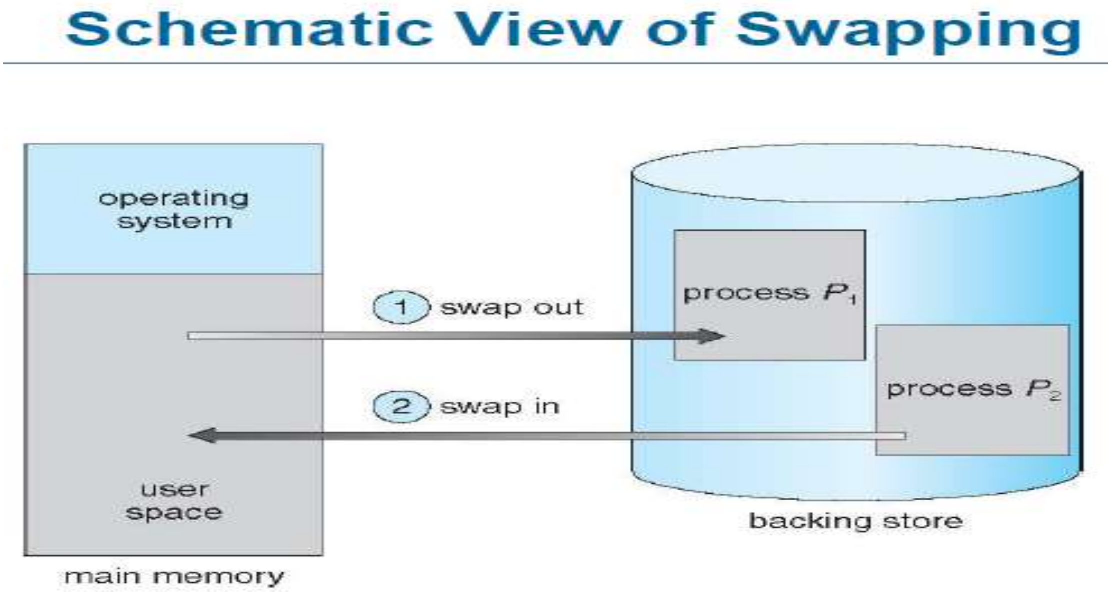
$= (7 \times 4) + 0 = 28$

$P \cdot A = (2 \times 4) + 1 = 8+1 = 9$

What is swap on a computer?

A **swap** file (or **swap** space or, in Windows NT, a pagefile) is a space on a hard disk used as the virtual memory extension of a **computer's** real memory (RAM). Having a **swap** file allows your **computer's** operating system to pretend that you have more RAM than you actually do.

To replace pages or segments of data in memory. **Swapping** is a useful technique that enables a computer to execute programs and manipulate data files larger than main memory.
The **operating system** copies as much data as possible into main memory, and leaves the rest on the disk.



What is address bind?

Address binding is the process of mapping the program's logical or virtual **addresses** to corresponding physical or main memory **addresses**. In other words, a given logical **address** is mapped by the MMU (Memory Management Unit) to a physical **address.**