Memory Management

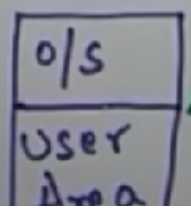
↳ It is the functionality of an operating system which manages primary memory and moves processes back and forth b/w main memory and disk during execution.

Functions of Memory Management

↳ i) Keep track of every memory location.
ii) Track of whether memory is allocated or not
iii) Track of how much memory is allocated.
iv) Takes decision which process will get
v) memory and when.
vi) updates the status of memory loc" when it is allocated or freed.

Protection

Process Can't Access O/S



Goals of Memory Management

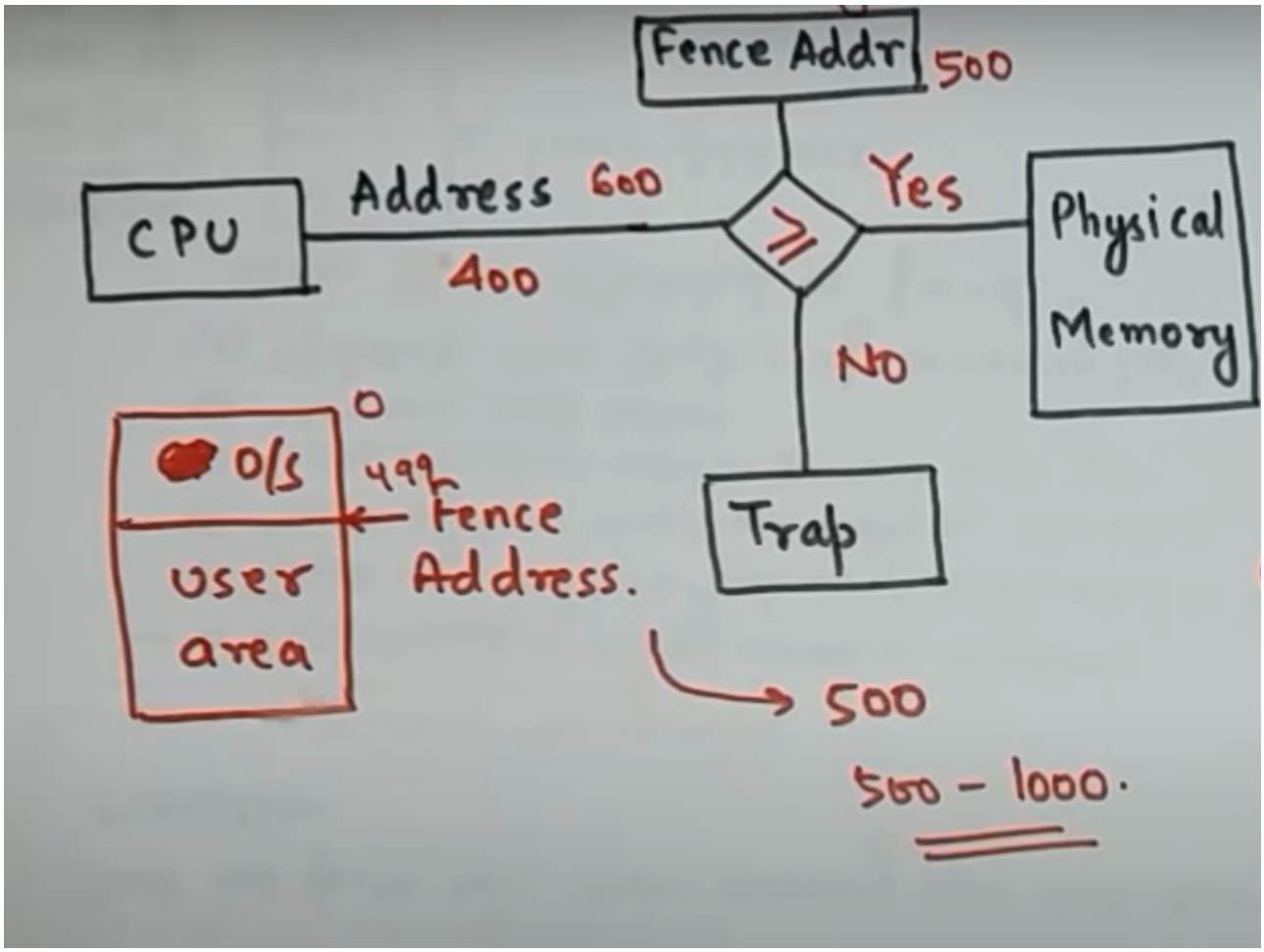
i) Space Utilization.
Fragmentation → Memory loss can occur.
∴ Tries to keep fragmentation as low as possible.

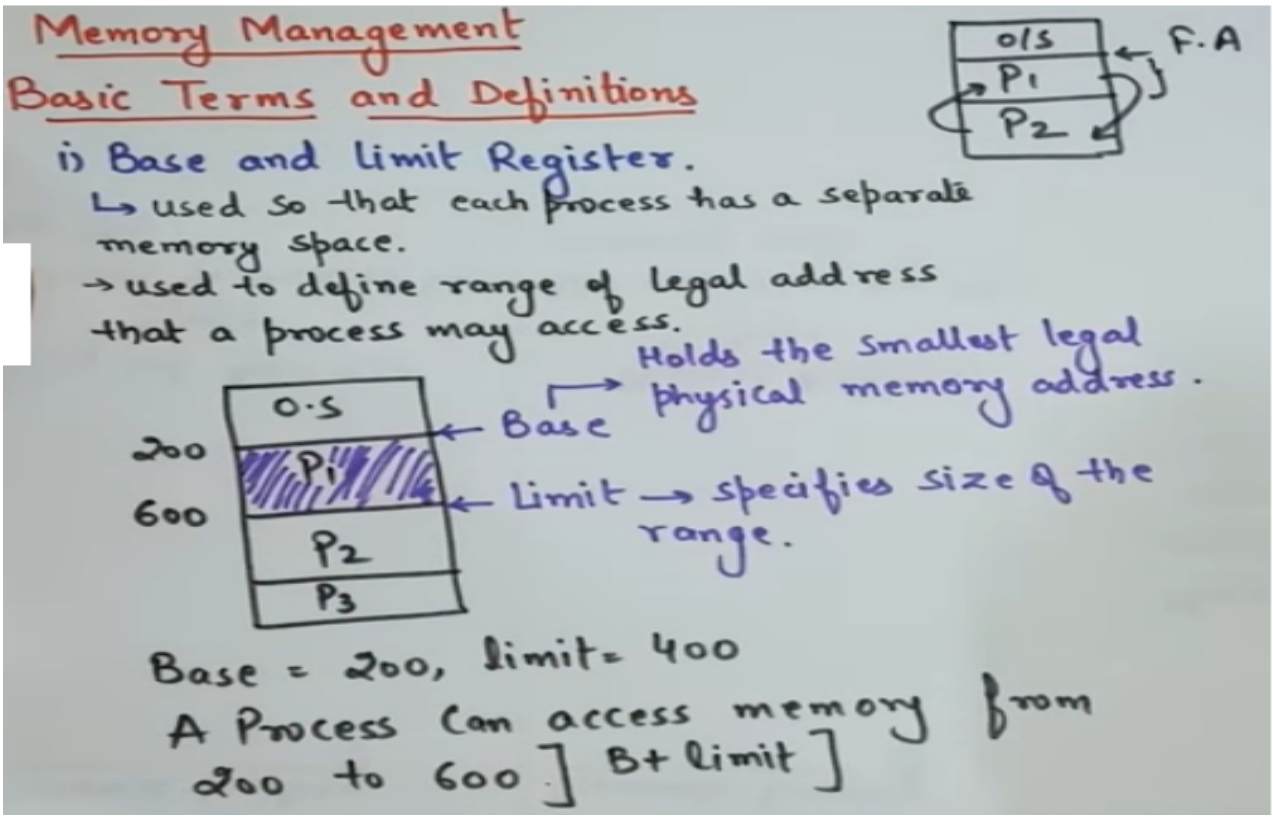
ii) Run Larger Program in Smaller Memory Area.

Prog = 1000 KB —— 500 KB.

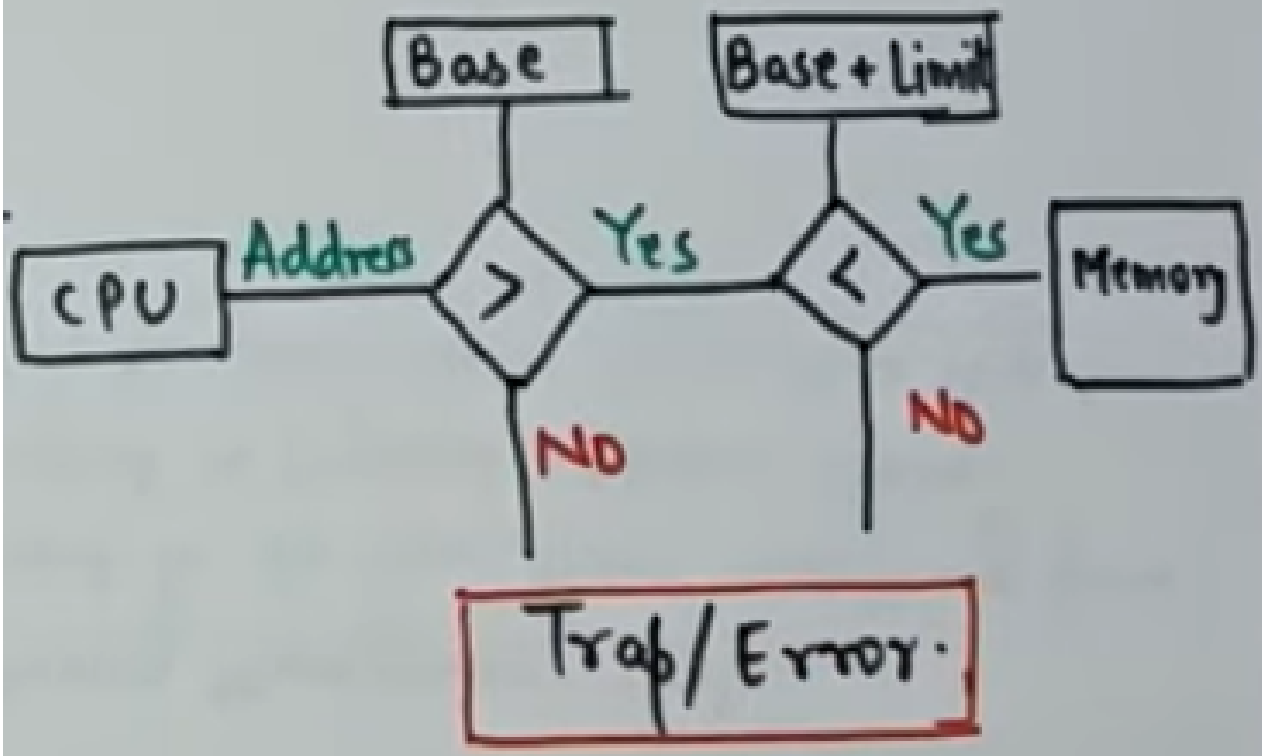
↳ Virtual Memory.

What is Fence address and why it is needed?



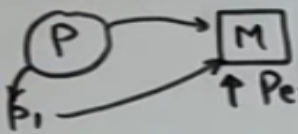


H/w Protection.

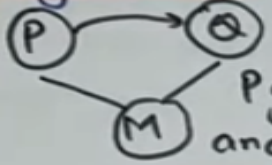


Dynamic loading

Complete Prog. is loaded into memory, but sometimes a certain part of prog. is loaded only when it is called by prog.



Dynamic linking



Initially only P gets loaded, and CPU links the dependent prog. (Q) to main executing prog. (P) when it is needed.

Logical Address Space

Address generated by the CPU. (virtual add.)

In Compile-time and address binding (LAS) and (PAS) are different in execution time binding.

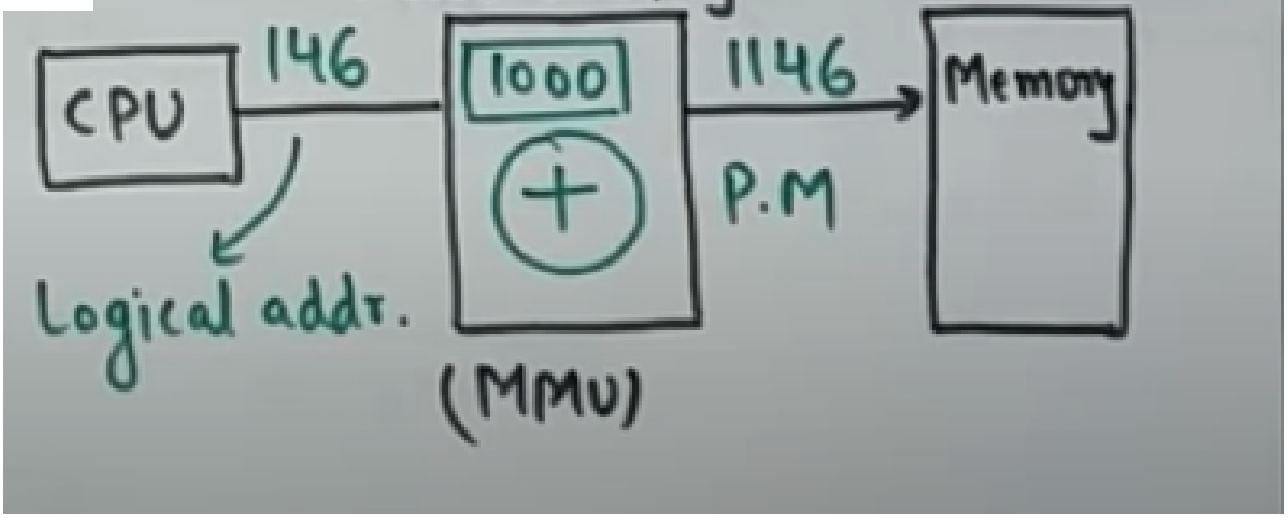
Physical Address Space

Address seen by the memory unit.

load-time memory (LAS) and (PAS) are same.

Memory Management Unit

- used to do run_time mapping from logical to physical address space.
- Can be done with Relocation Register.

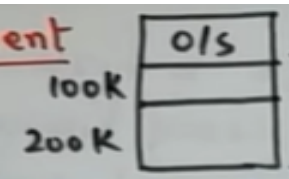


Memory Management Techniques

Contiguous	Non-Contiguous
- Centralized	Decentralized
- Prog./Process as a unit is completely stored.	- Prog. parts are distributed in memory.
- Partitioning.	- Paging
	- Segmentation
	- Virtual Memory.

Memory Management

Partitioning

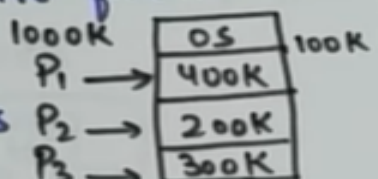


Memory is divided into two partitions. Resident / User

Partitioning → Fixed (Multiprogramming) with fixed tasks (MFT) processes.
Partitioning → Variable (MVT)

M.F.T :- No. of partitions are fixed and are of different sizes.

1 Partition = 1 Process

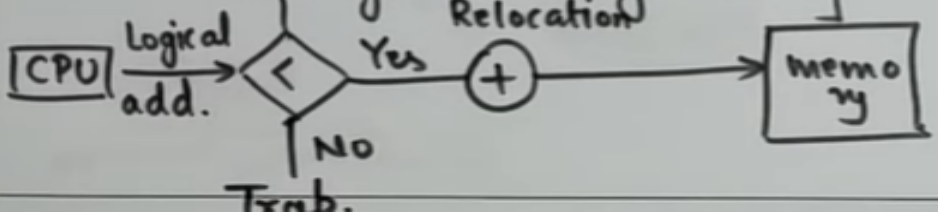


Protection < LBR, UBR

memory ≥ LBR
memory < UBR.

limit reg.

Relocation



Partition Allocation Policies

i) First Fit :- Allocates the process in the first free large enough partition.

ii) Best Fit :- In the Smallest possible memory partition.

iii) Worst Fit :- Largest possible memory partition.

iv) Next Fit :- Same as the first fit, but it doesn't start from first partition. It starts from the place where the last partition ended. [Fast Searching]

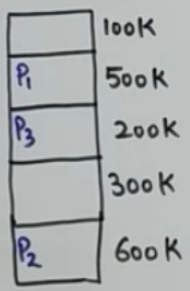
Solved Numerical on Partition Allocation

Ques.) Given 5 memory partitions of 100K, 500K, 200K, 300K and 600K (in order). How would each of first-fit, best fit and worst fit algorithm places processes of 212 KB, 417K, 112K, 426K (in order).

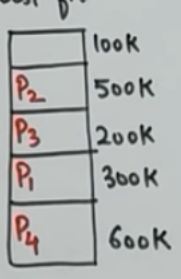
P_1 P_2 P_3 P_4

i) First Fit

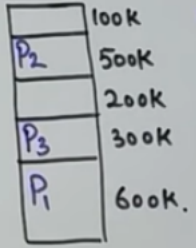
P_4 will not get allocated.



ii) Best fit



iii) Worst Fit P_4 will not get allocated.

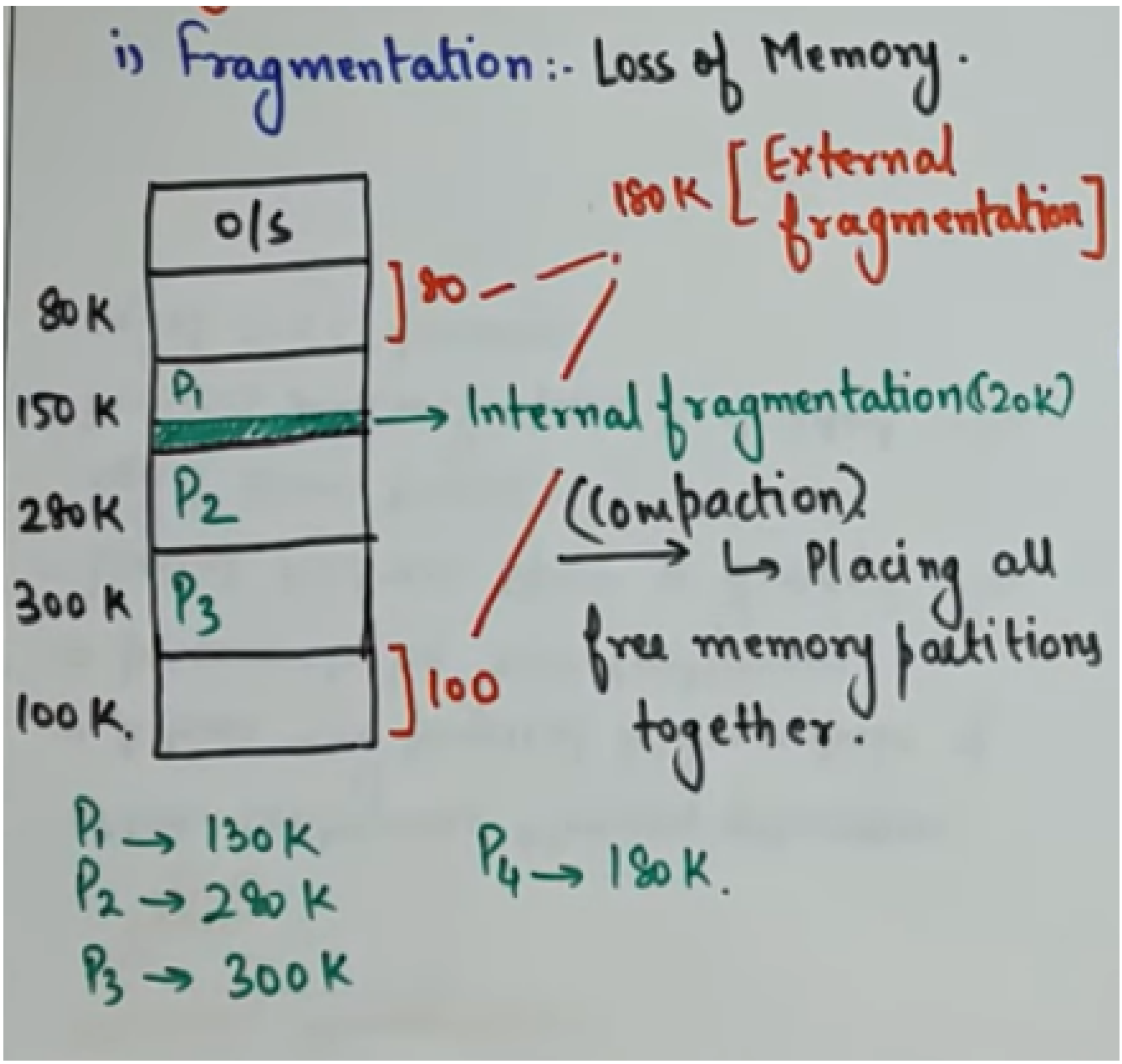


Analysis of MFT:

iii) There is no flexibility in maximum process size or maximum partition size.

iii) There is no flexibility in degree of multiprogramming.

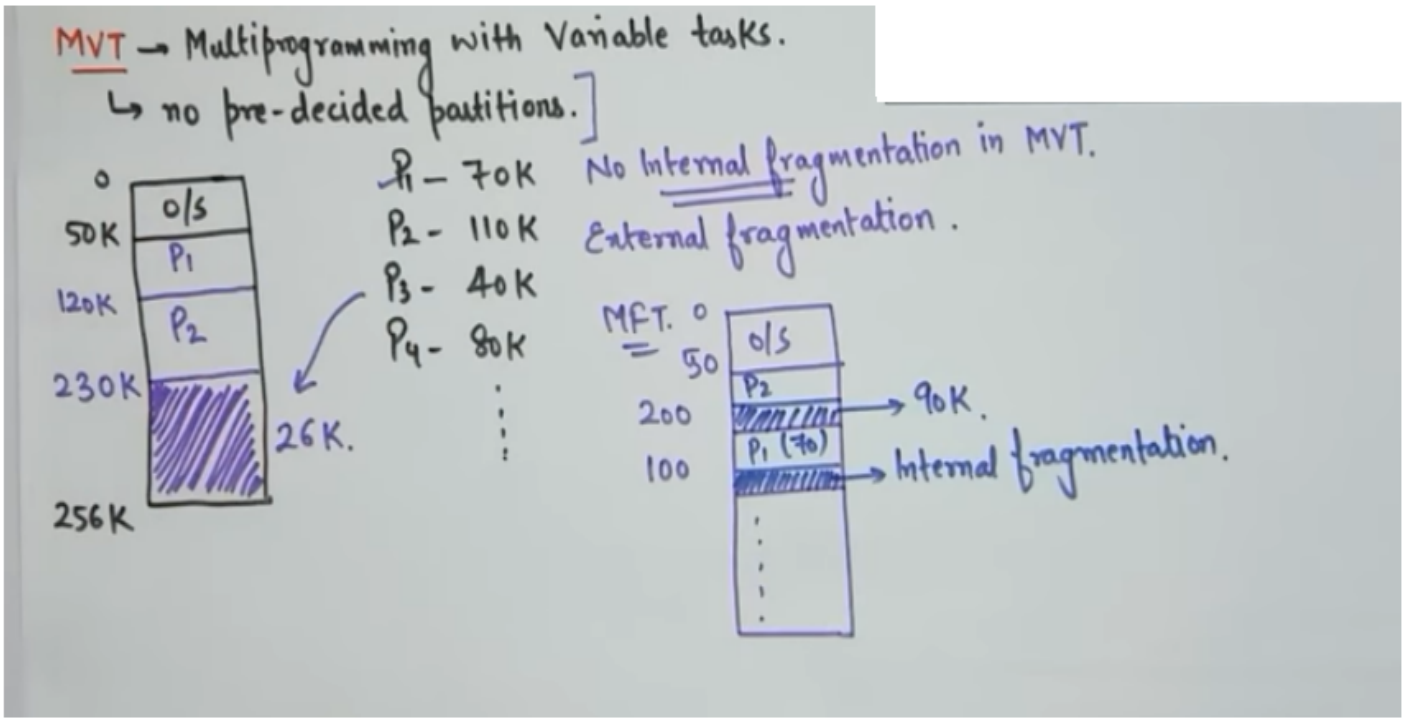
Fragmentation:



External Fragmentation → Arises when unable to accomodate a Process of size even we have sufficient size of memory but it is not contiguous.

Internal Fragmentation Arises when a Process of size smaller that allocated partition is placed in that partition, that leaves behind small amount of unused memory.

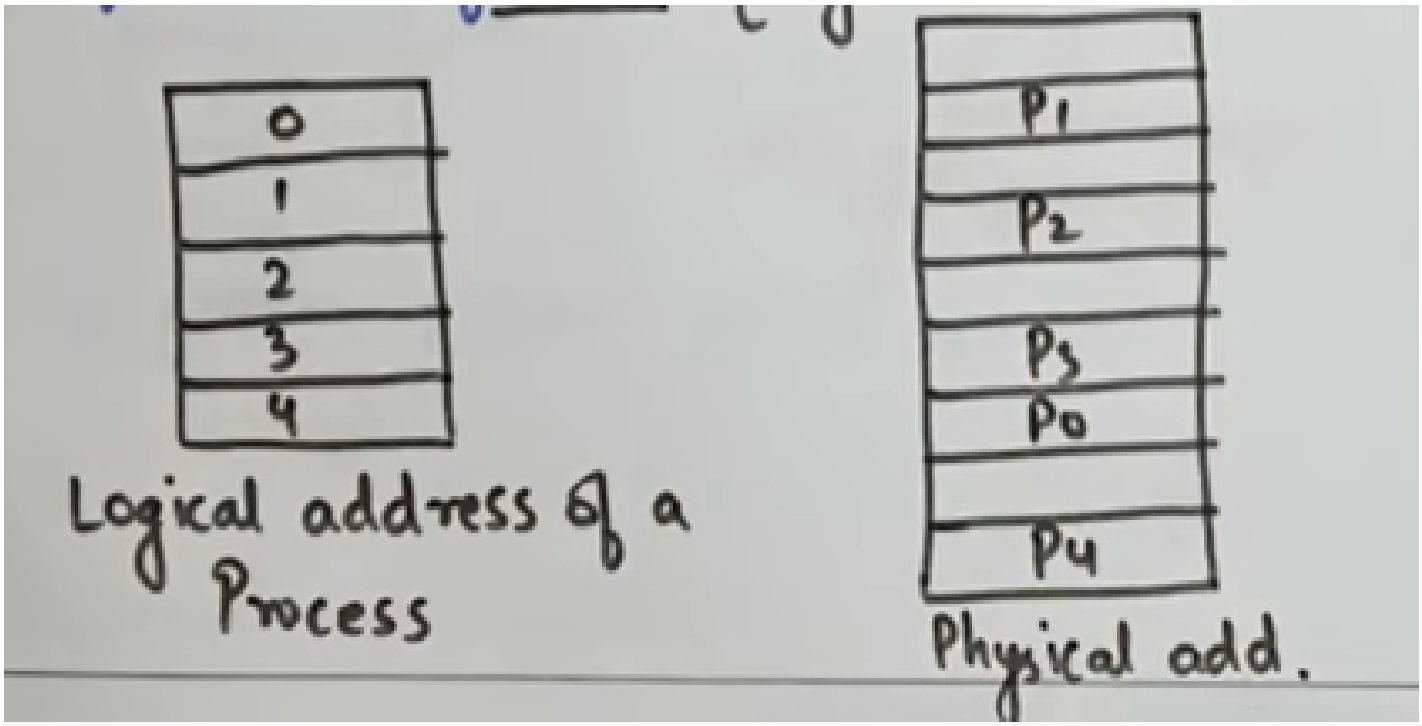
To Remove the Internal Fragmentation, we use MVT (Multiprogramming with Variable Task)



Paging

→ Non-Contiguous Memory allocation.
→ Allows the physical Address space of a process to be non-contiguous.
- Logical Address space is divided into equal size pages.
- Physical Address space is divided into equal size frames.

→ Non-Contiguous Memory allocation.
→ Allows the physical Address space of a process to be non-contiguous.
- Logical Address space is divided into equal size pages.
- Physical Address space is divided into equal size frames. [Page Size = Frame Size]



Page Size = P

L: Logical address

L = (p, d) → displacement

↓ → offset within that page.

page no.

ize]

14 → (1, 4) → (p, d)

↓

(2, 4)

Page Size = P

L = Logical address

L = (p, d) → d: displacement; offset within that page.

p: page no.

Phys. addr.

Size]

14 → (1, 4) → (f, d) → (f-1) * P + d

(2, 4)

p = L (div) P

d = L (mod) P

P = 10 , p = 14 div 10 = 1] (1,4)

(14) d = 14 mod 10 = 4

Hardware Architecture of Paging: