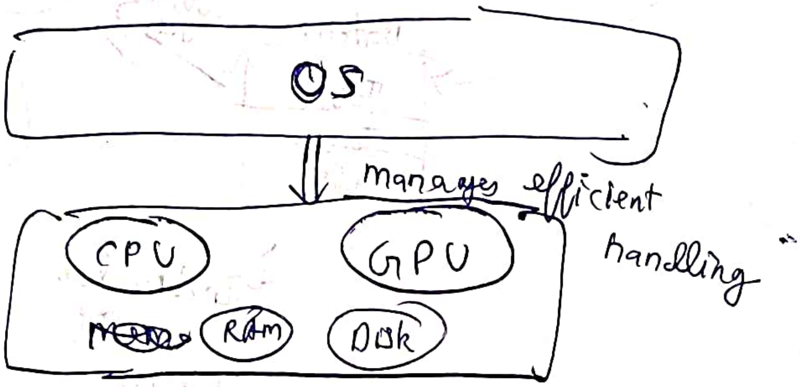
Operating System.

Lec1

Basics → It's an interface, which does resource management



Lec2

Goals of Operating System.

① max CPU utilisation → I want CPU to have minimum idle time

② Process Starvation should not happen. → P_1, P_2, P_3 ⇒ If P_1 is very long process then context switch happens and other processes get chances.

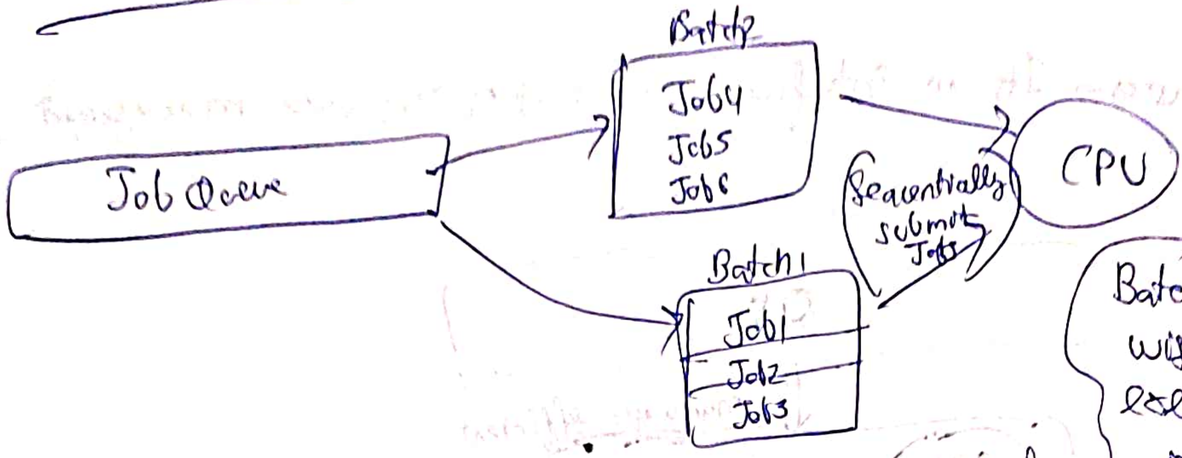
③ High Priority execution,
↓
any process having higher priority should get executed faster. → Like antivirus scan.

Types of OS

Single Process OS → 1 core, execute processes sequentially no context switching.

ex → MSDOS

② Batch Processing OS

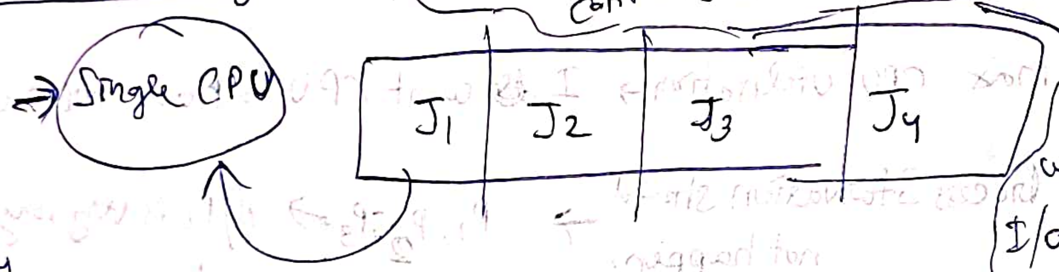


Batch wise execution is happening

Individual each Job will be executed sequentially

③ Multi-programming OS

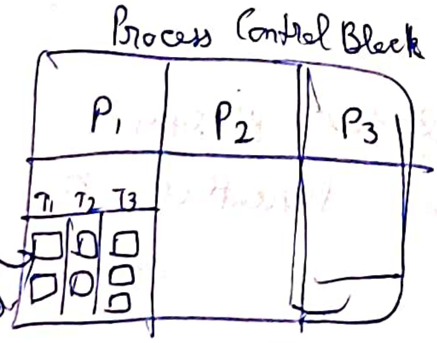
*** Important thing to remember is that, the context switch is happening only if a Job went for I/O and not using CPU



If J1 goes into I/O call then make context switch and let J2 do the work.

When context switch is happening then, info ~~from~~ for P_1 should be saved somewhere to continue it

Process Control Block



here in the register the state is stored

we have

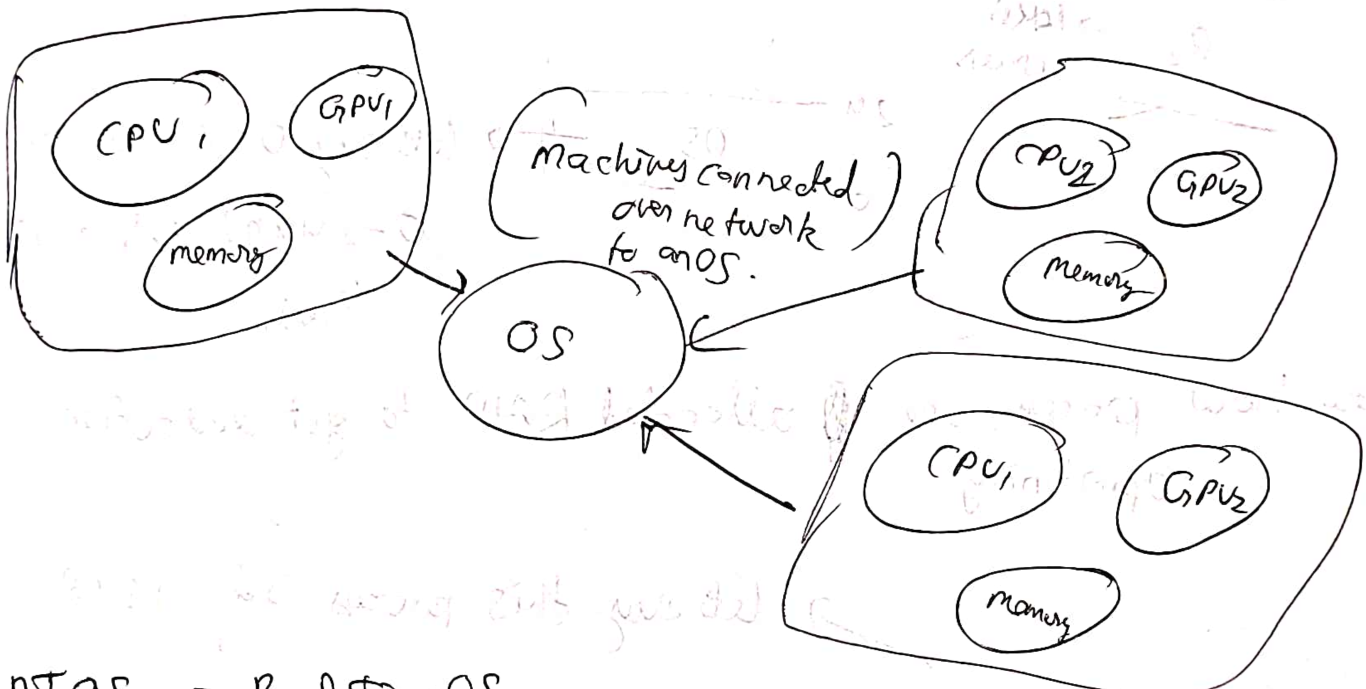
{Stack, Register, Program Counter}

④. Multitasking OS → Same as Multiprogramming
- context switching
- timesharing

But major difference is time sharing,

here even if the job is not going for I/O still after an interval OS will pull out that job and make it wait, and let other jobs have a chance

⑤ Distributed Centralized OS.



⑥ RTOS → Real Time OS

ATC uses it

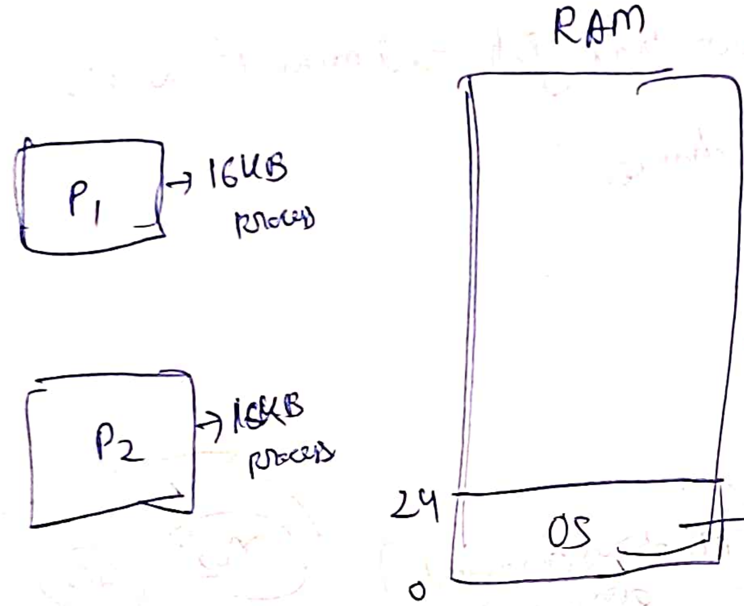
Error chance very low..

⑦ MultiProcessing OS → Same as Multitasking.
- context switch
- timesharing
- #CPUs ≥ 1

(Lecture 24) → Memory Management in OS..

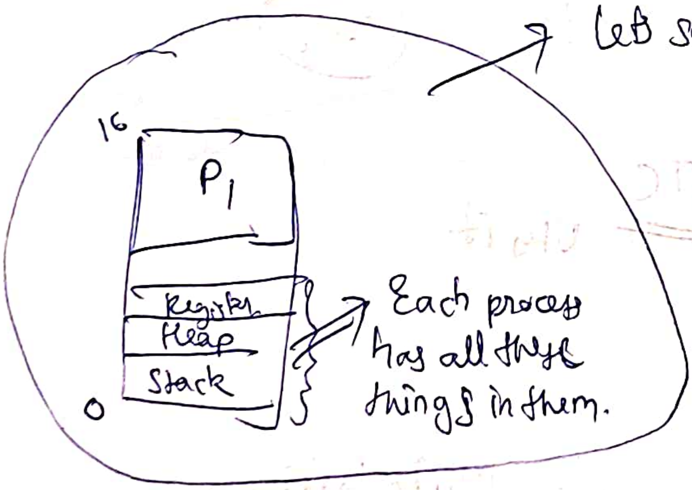
~~In multipro~~

In multi-programming model, we have multiple processes



→ Lets say OS is there in (0-24KB) OS is stored there.

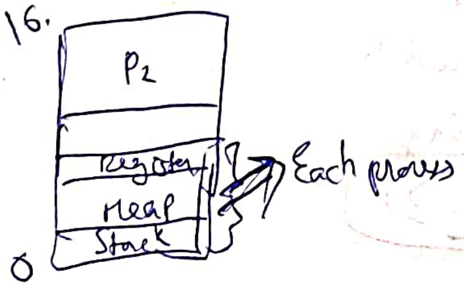
Now how processes are allocated RAM to get execution opportunity.



→ Lets say this process is of 16KB.

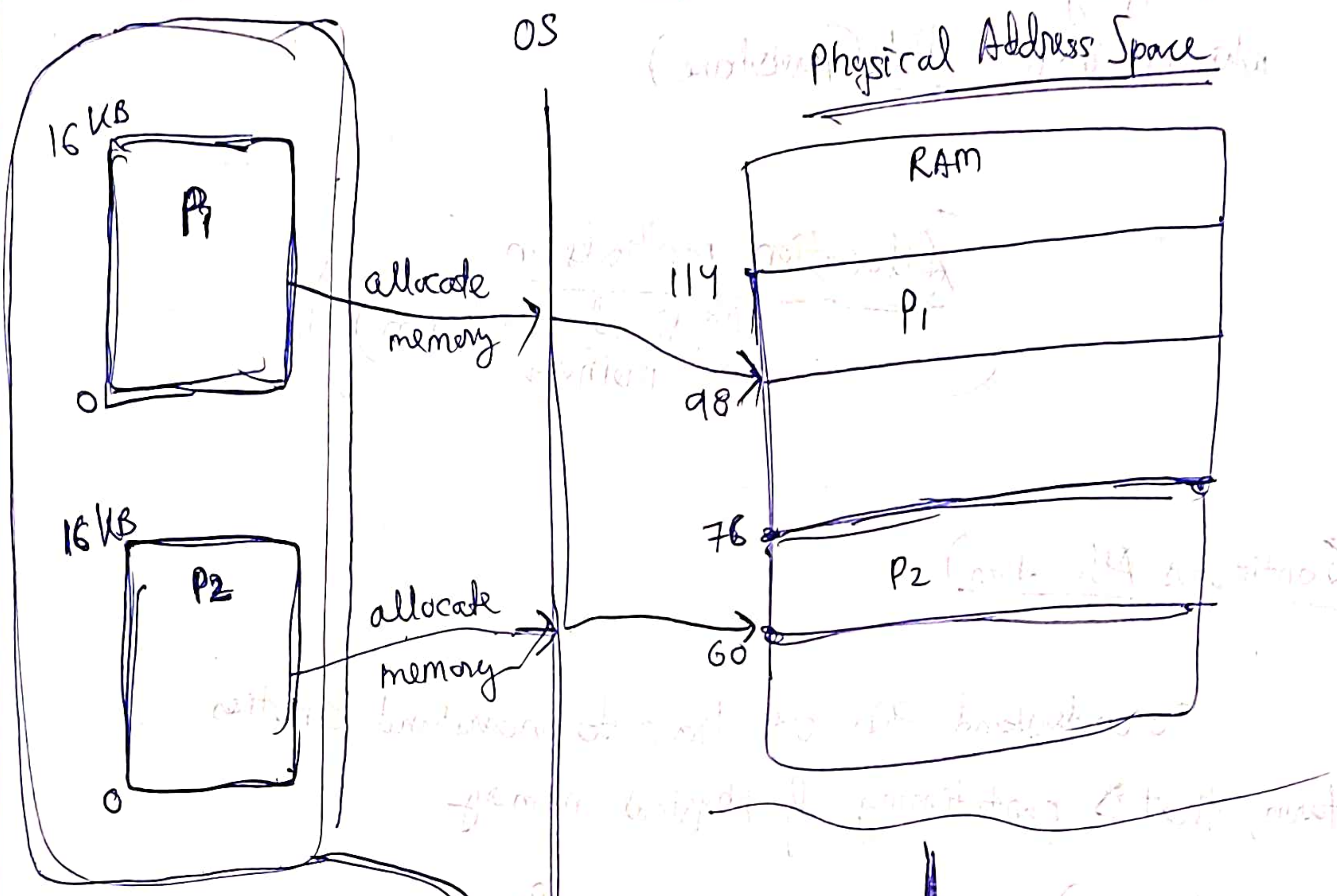
So this process is represented using Logical Address Space

$P_1 \rightarrow \begin{bmatrix} \text{Base} \rightarrow 0 \\ \text{offset} \rightarrow 16 \end{bmatrix}$



Now this process directly does not talk with the RAM.

it talks with the OS.



These memories are allocated to Logical Address or Virtual Address Space

Physical Address Space.

$B \to 0 \to 98$

offset $\to 16 \to 114$

Process 1 when this ~~translate~~

⇓

When Process 1 tries to access some block or address, outside the given range, OS will throw error and will not allowed it

This is how OS is maintaining Process Execution Isolation

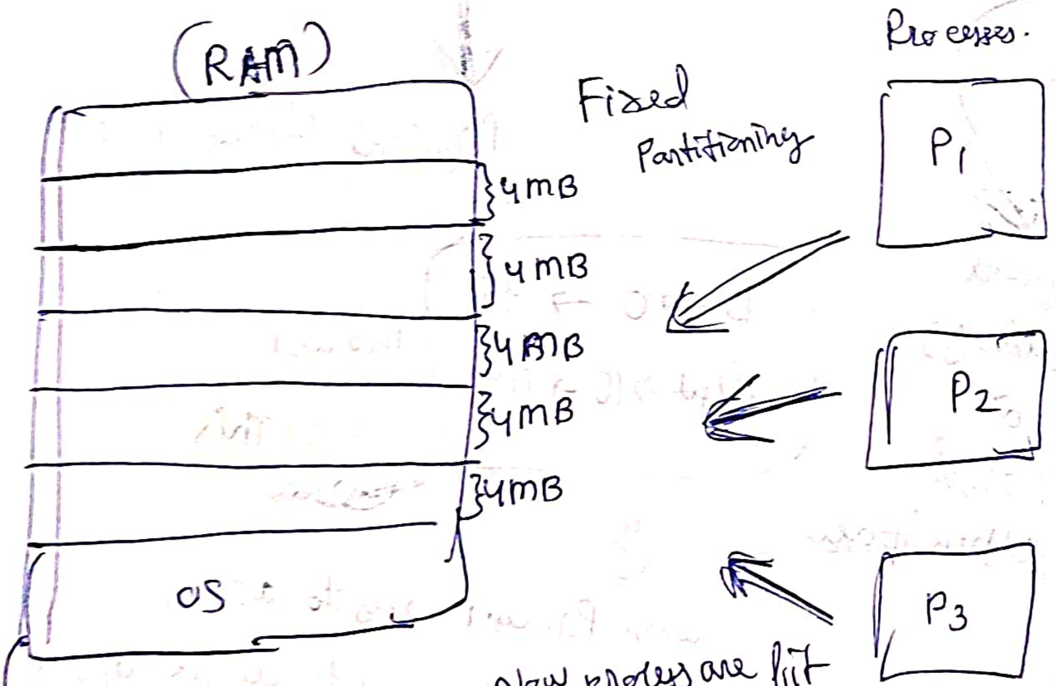
(Logical Address) → (Physical Address)

This mapping is stored in a [Memory Management Unit] which is inside disk (persistance)

Allocation methods in physical memory (RAM)

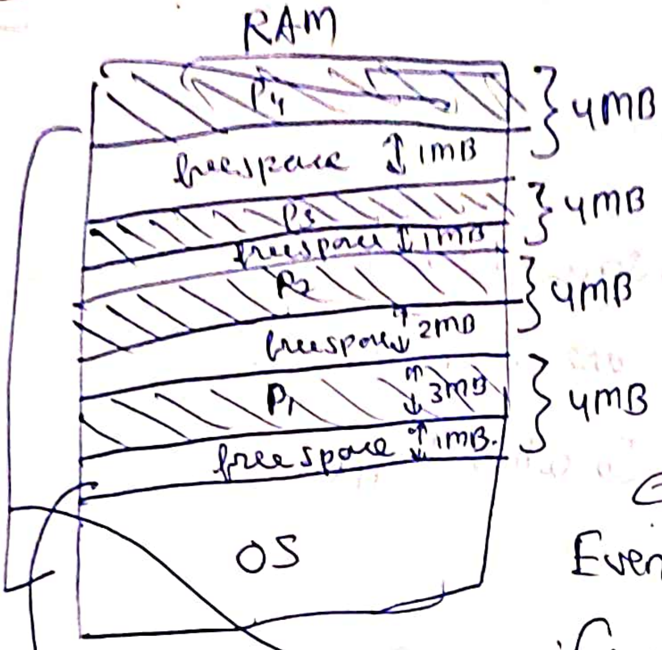
Continuous Allocation.

To understand this we have to understand another term, that is partitioning of physical memory



physical memory broken into chunks

Now process are fit into chunks.



Now lets say a process comes

$P_5 \rightarrow$ of 4MB

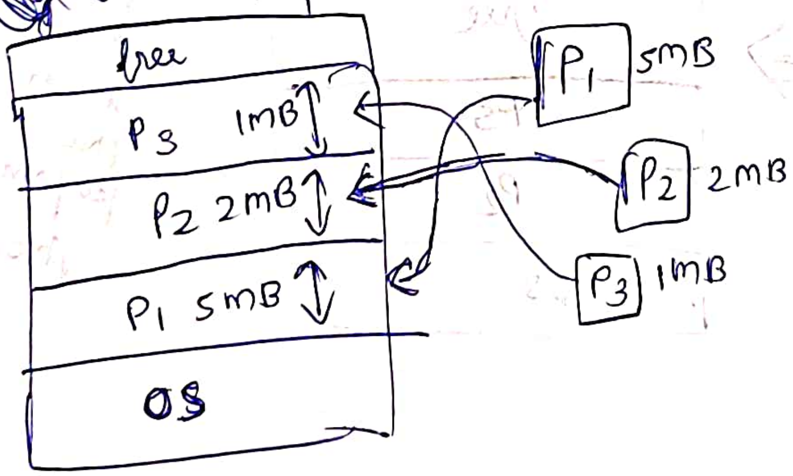
Eventhough we have a free space of $(1+1+2+1 = 5MB)$ still we are unable to allocate process P_5.

this left over space which we are unable to utilize is called internal fragmentation.

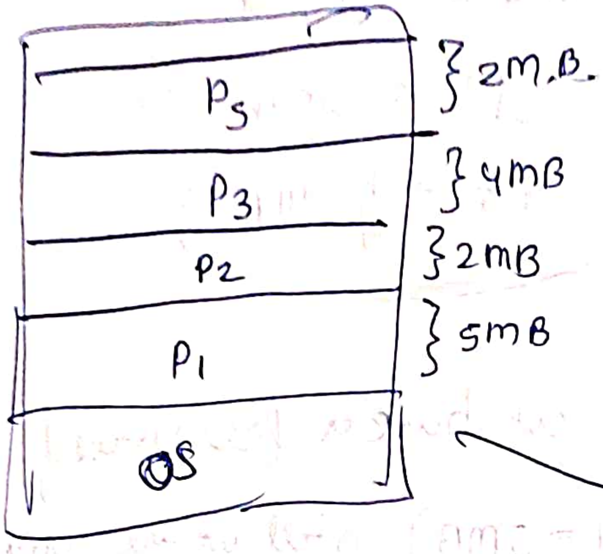
Remaining all the free spaces, is called external fragmentation.

To solve this problem.

We use dynamic partitioning, Basically in this approach, we dont partition in fixed chunks, but we partition ~~using~~ when a process is loaded into the system.

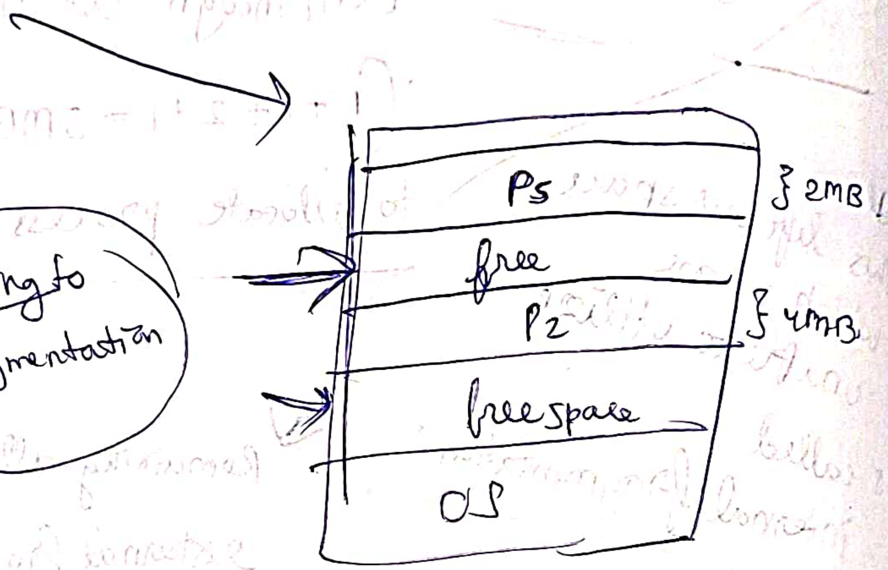


this was no internal fragmentation is happening.



RAM

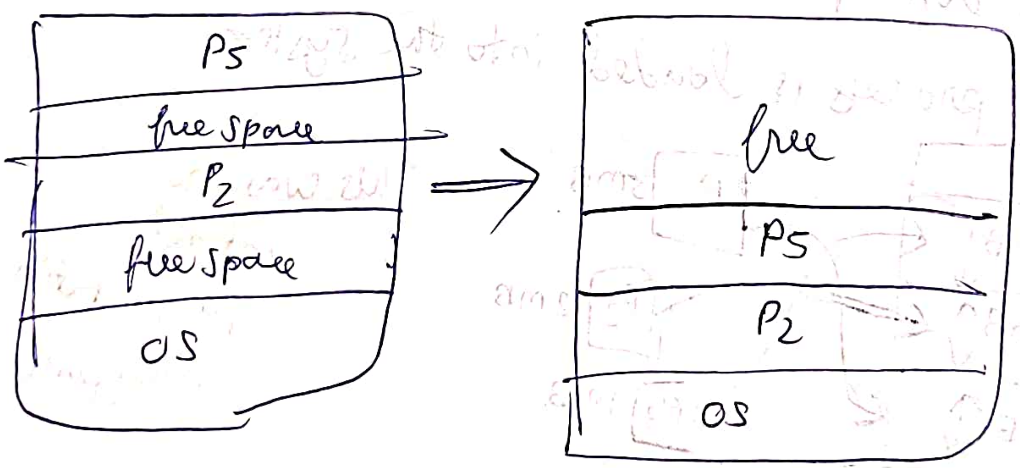
Lets say P3 and P1 are completed now, So what happens.



Leading to fragmentation

How OS resolves this, is using

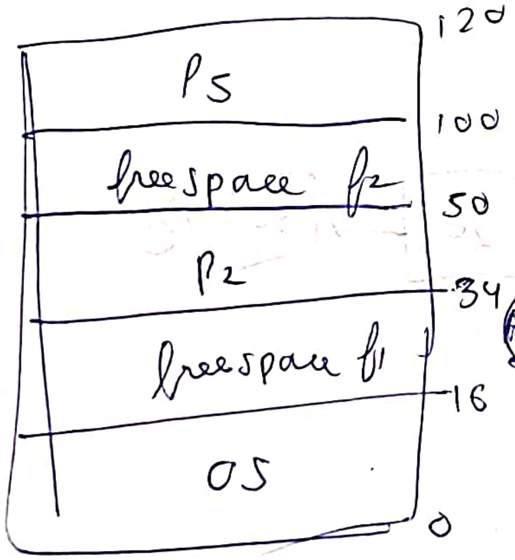
(Defragmentation / Compaction) just like SSTs and Cassandra,



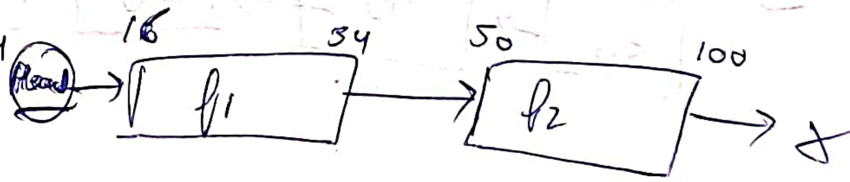
Well this takes a hit on the performance though.

Fixed Partitioning	Dynamic Partitioning
→ internal & external fragmentation	→ only ~~int~~ external fragmentation, no internal fragmentation
→ Degree of multiprogramming is less, as we saw even though 5MB ~~process~~ ~~as~~ memory is free, still process is not getting executed.	→ Degree of multiprogramming is more as ~~con~~ after Defragmentation, new process is allocated RAM.
→ Limit on size of process as chunk size is there	→ No limit on size of process as no chunk size, dynamic partitioning.

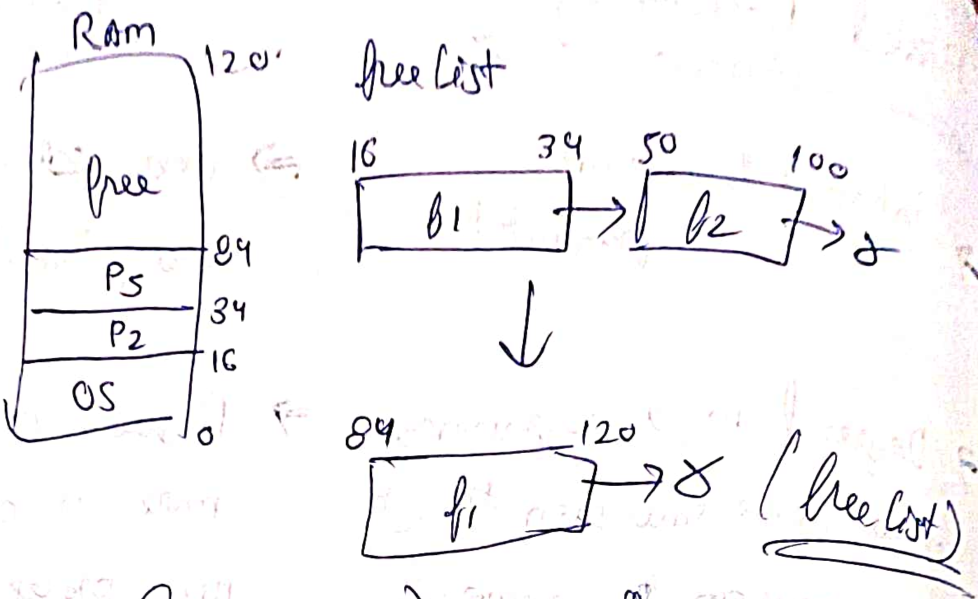
Lecture 25 : How OS manages free space?



So OS maintains a Doubly Linked List called Free List



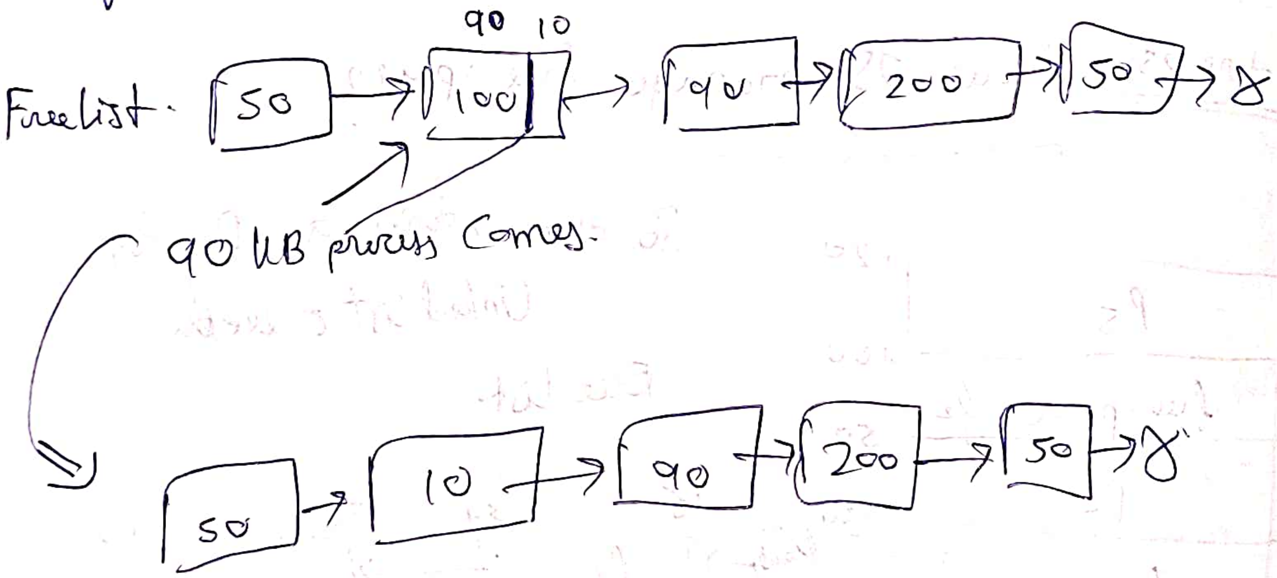
Now if compaction happens, the free list changes.



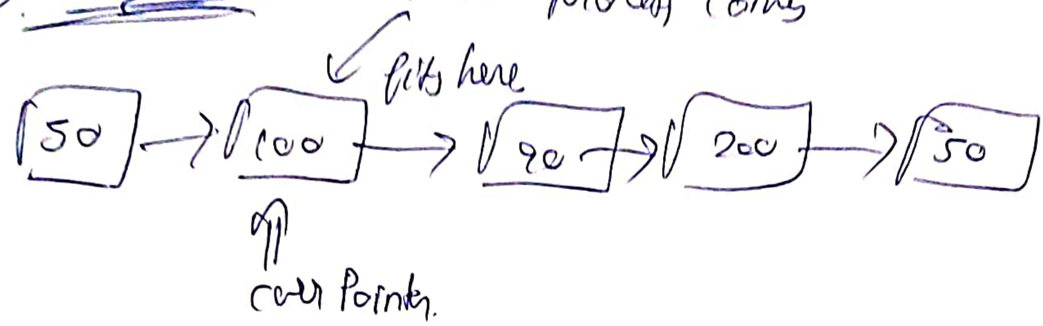
Now lets say before (compaction / defragmentation) happens a process comes.

How will the allocation take place?

① First fit (fit into first node which satisfies size).

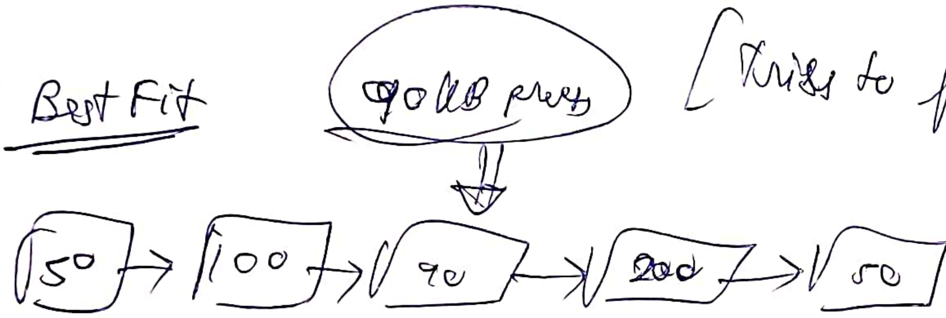


② **Next Fit.** 90 KB process comes



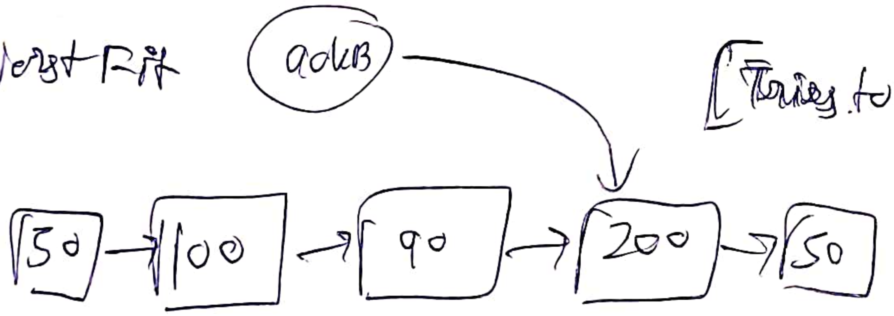
Next process whichever comes will try to fit and linear search begins from curr Pointer or the pointer where last entry was there, Not from the start

③ **Best Fit**



[Tries to fit in such a way to minimize internal fragmentation.)

④ Worst Fit



[Tries to fit where max internal fragmentation is happening].

Best Fit → low internal fragmentation.
high external fragmentation

Worst fit → high internal fragmentation
low external fragmentation.