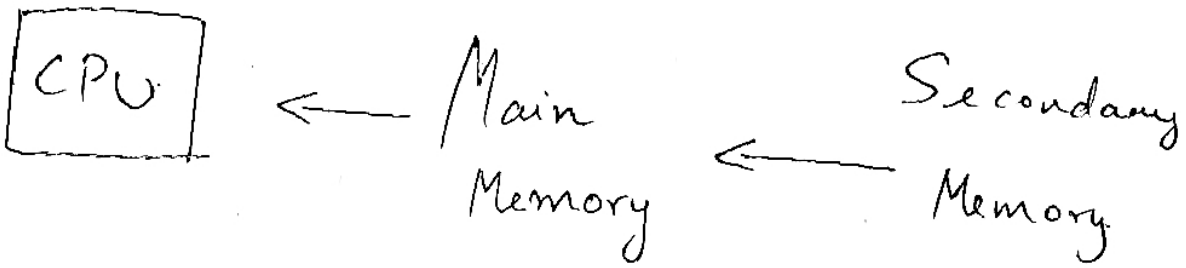
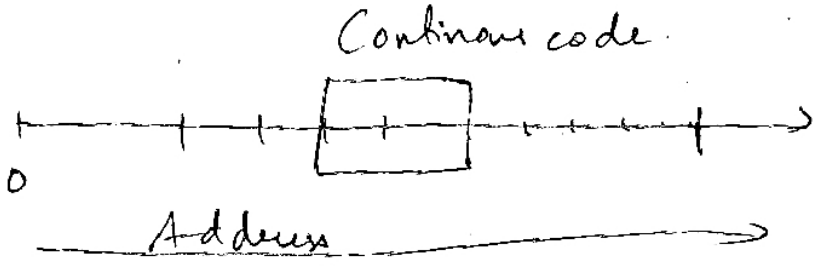
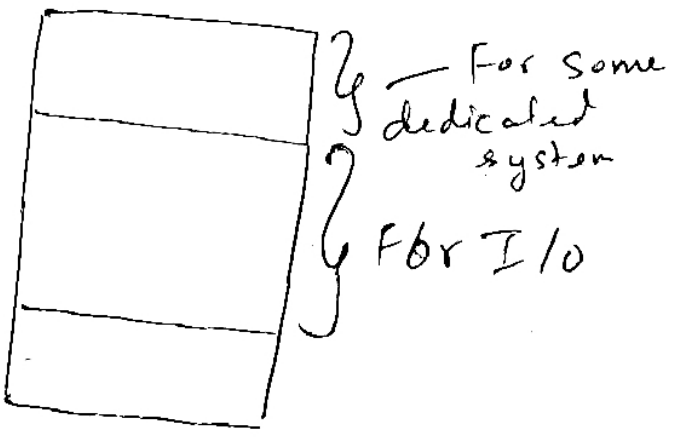
Memory Management. ①



- Generally, instructions / code is sequential.
  ⇒ We can therefore bring a lot of code (or data) beforehand.

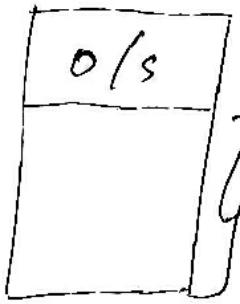


Bare System :
or
Dedicated system



All parts of the memory (main) are fixed.

These days we have the O/s residing in the memory.



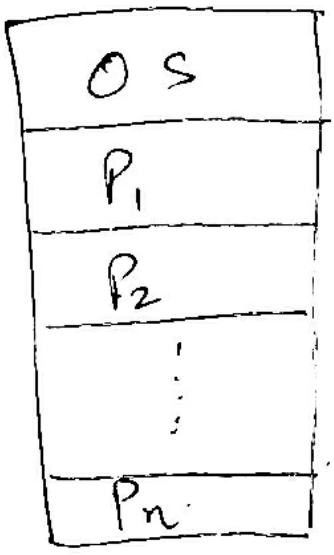
occupied by OS.

used by user area.

clear

Note: There is a distinction b/w OS area and user area.

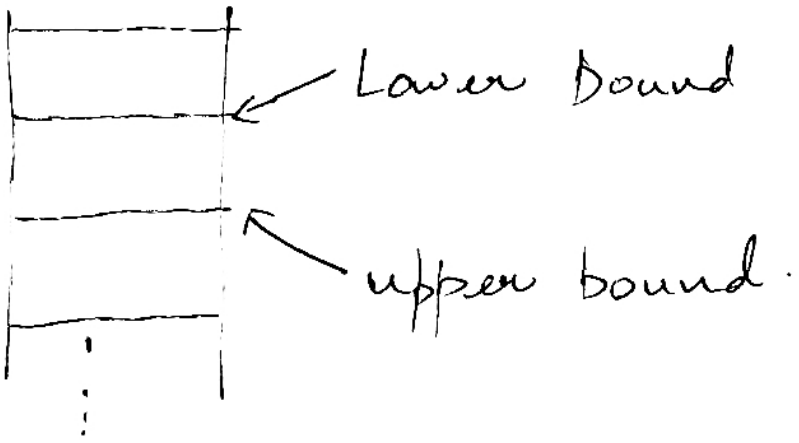
Multiprograming with fixed tasks



Switching b/w processes needs only IP change.

IP: Instruction Pointing. (3)

For every partition, there must be an upper & a lower bound.

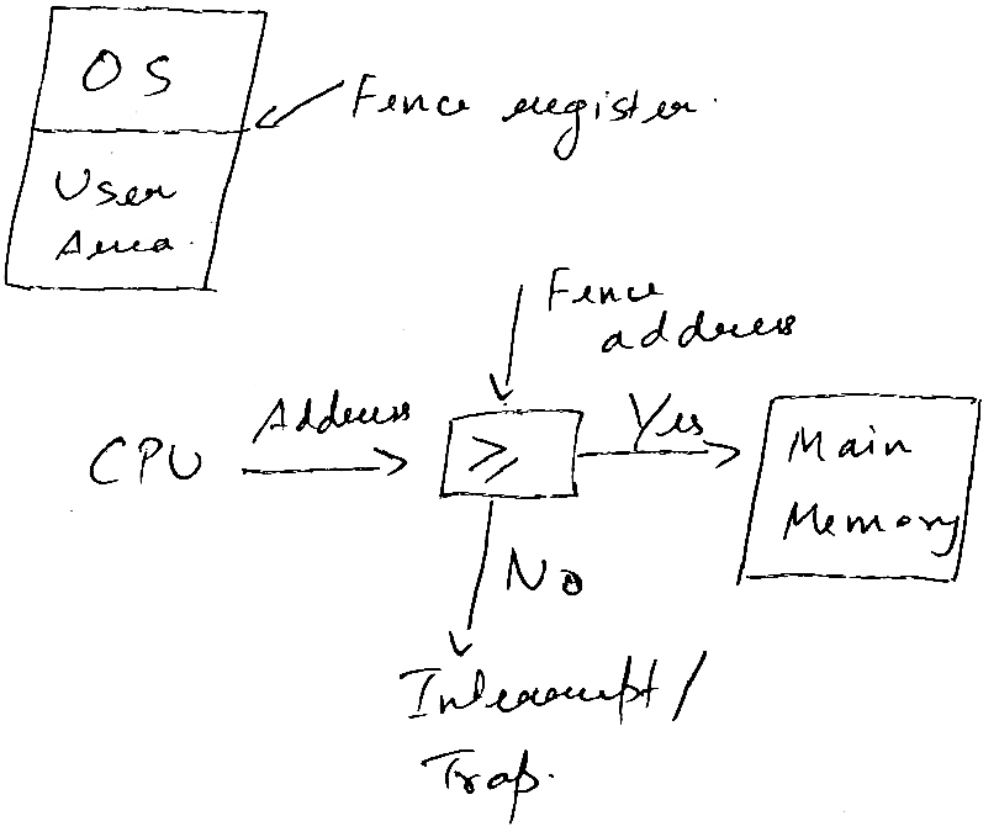


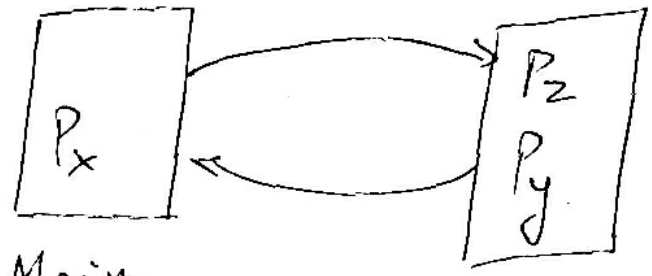
Q: How does CPU access something residing at the physical address.

A: Use logical address and memory management unit.

(4)
Logical address is the address at wh an item appears to reside from the perspective of an executing applicat program.

Address Translation
Modern.
For ~~dedicated~~ systems





Main Memory (M.M.)

Secondary storage (S.S.)

If P_x is modified in M.M.
{
copy in S.S. should be updated
}

If P_x goes into trap mode
{
P_x must be swapped out
another process P_z should be swapped in
}

(4)

(5)

If P_x is finished

{ P_x in M.M. can be overwritten

by P_y (e.g.) since copy of

P_x is already there in S.S.

}.

If, we have only two partitions

1) OS area

2) Process area

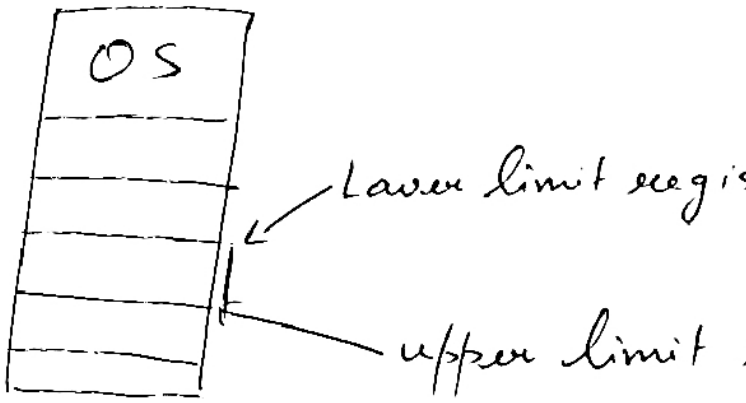
CPU utilization is bad.

↳ Many processes will go

for I/O.

etc.

Fix: Have multiple partitions in the main memory.

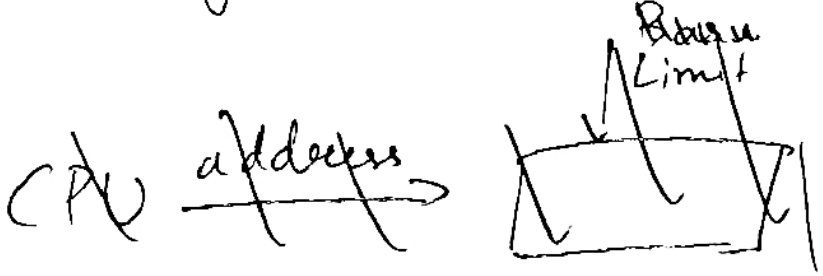


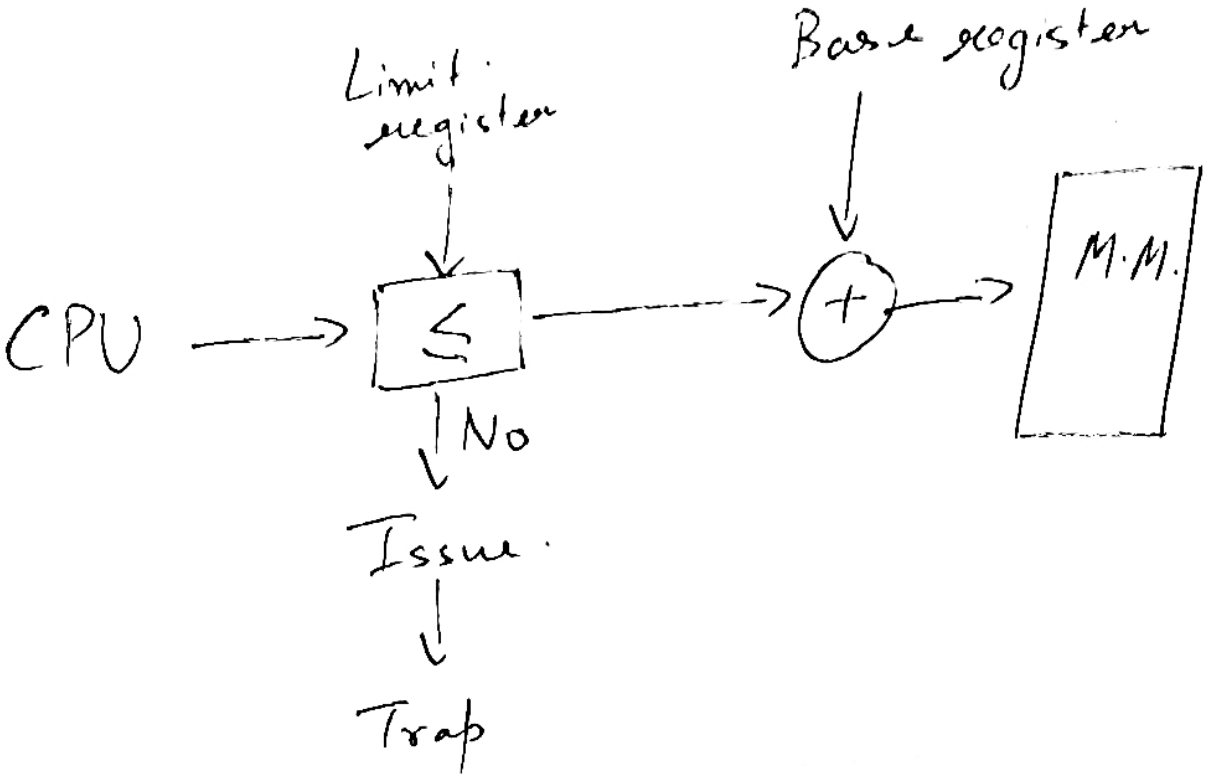
Modern systems use this idea.

They use the following:

1) Base register (starting point)
2) Limit register (Permissible range).

Memory access scheme now becomes:





Partition allocation.

Q: Multiple jobs have to assigned. memory in. How to do that?

Ans: Use memory allocation algorithms

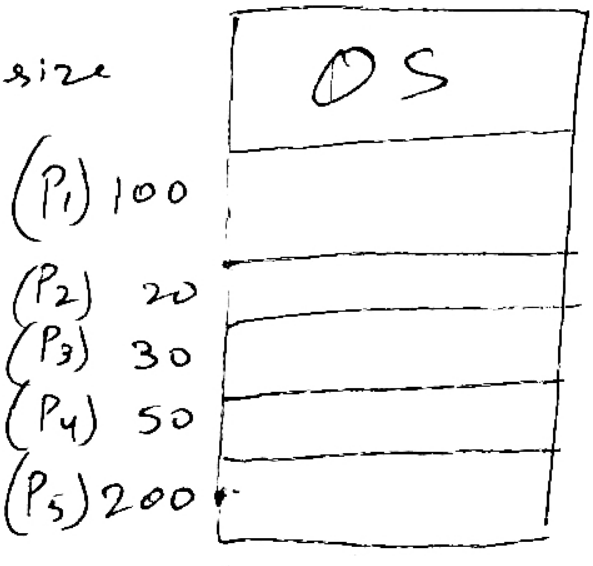
1) First fit algorithm

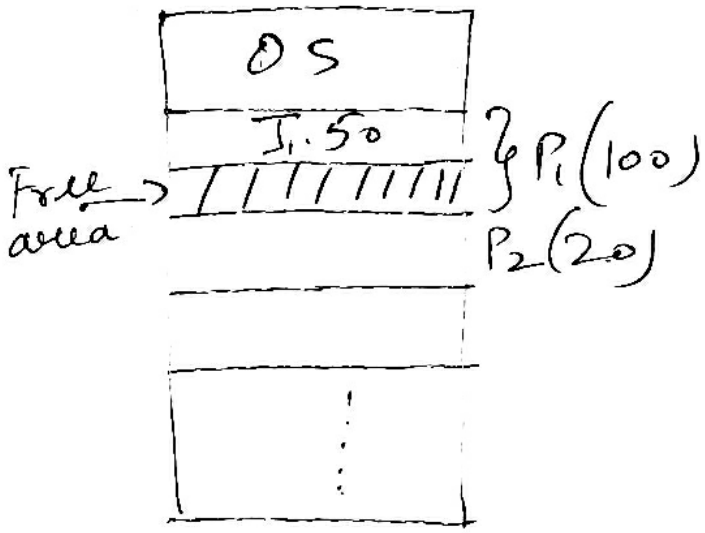
Scan memory partitions sequentially and find the partition with equal or more space than that required by a job.

e.g. J_1 need 50 Kb

P_1 is free and has size 100 Kb

∴ Assign P_1 to J_1





Issue: The area in ▨ is wasted.

This is called as internal fragmentation.

2) Best fit algorithm.

Find the best size for the job.

Expensive takes time.

3) Worst fit

Find the largest block to fit the job (Bad).

4) Next fit algorithm.

Similar to first fit.

↳ It however starts scanning partitions from the node where it previously allocated the partition.

To accomplish partition assignment additional info has to be maintained.

i) Starting location of the partition.

ii) Size.

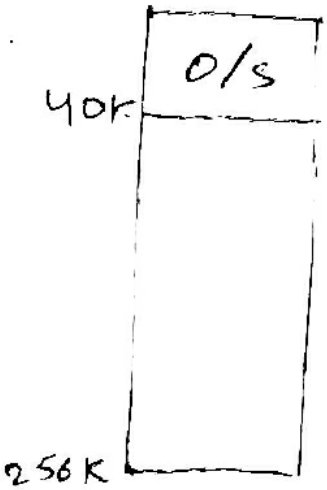
iii) Status.

The info is stored in partition allocation table (PAT).

The above table is used memory mgmt unit to assign partition.

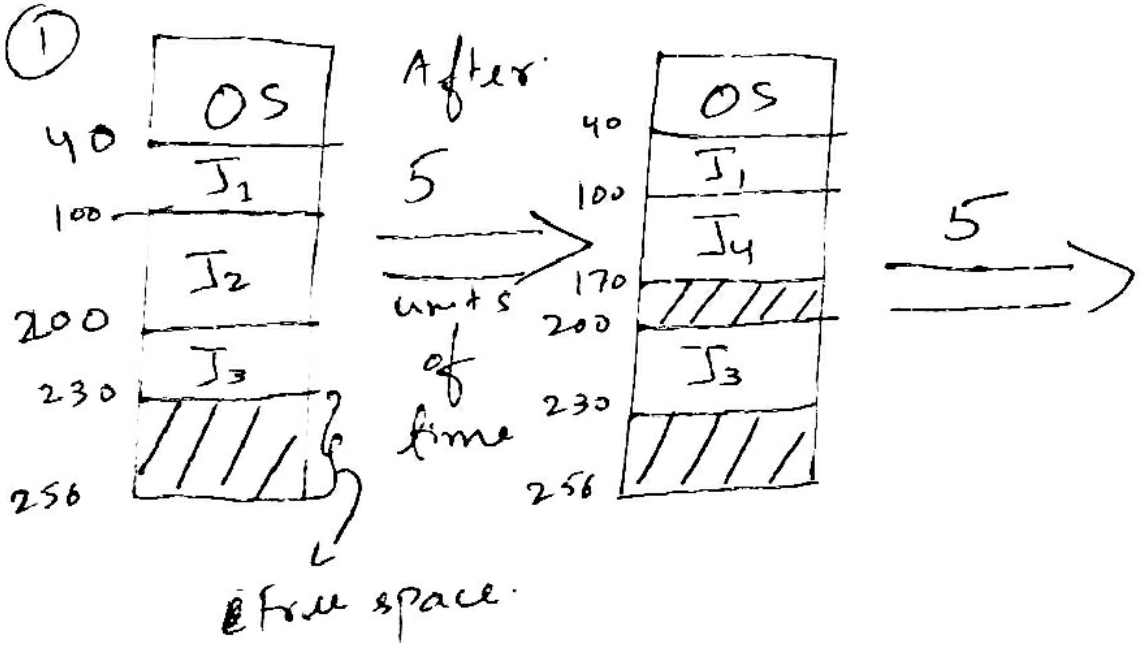
4. Example (Taken from Internet) (A)

Variable number of tasks

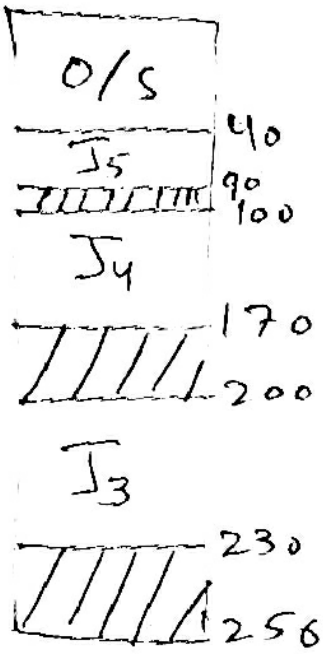


	Memory	B.T.
J_1	60	10
J_2	100	5
J_3	30	20
J_4	70	8
J_5	50	15
J_6	60	9

Say scheduling is FCFS



(B)

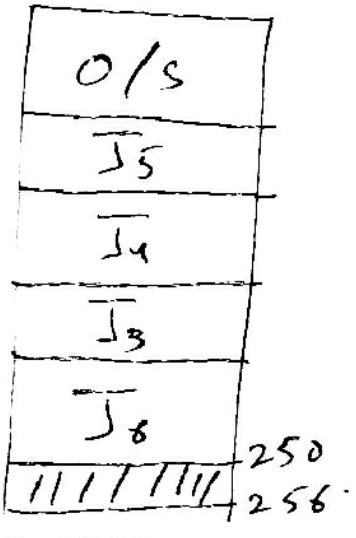


Total free memory 66 KB.

Memory requirement of J_6 60 KB.

However, no continous space is available

∴ do compaction.



Compaction makes things easy.

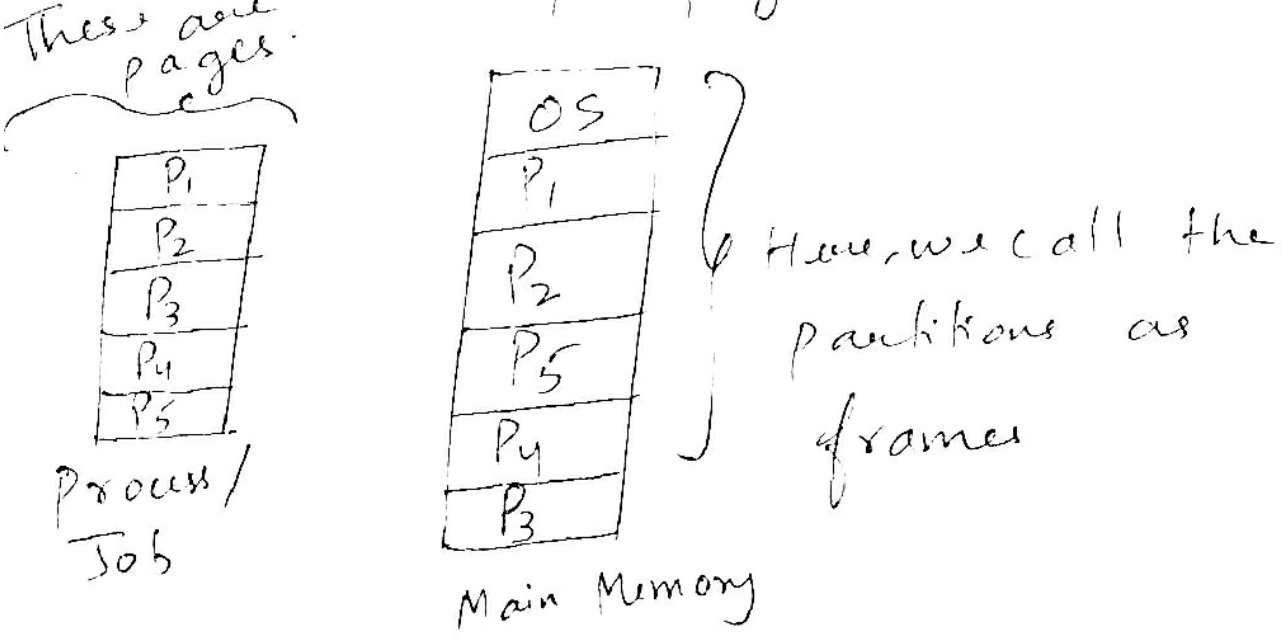
However, need more resources

- Compaction module
- Job/Process execution must pause.
- Additional info must be maintained.
  E.g. (1) which partitions are free.
  (2) Start & End address of each partition has to be rescheduled.
  (3) Entire memory structure is must be changed.

To Avoid the problems of compaction and maintaining additional information and fragmentation, we use page based memory mngmt.

Job : You divide this into multiple pages

Memory : You divide the memory into multiple pages.



P_i = i^{th} page.

Let assume that the page size is P

" " " " " number is p

and the offset within the page is d.

Let's say that the logical address generated is L.

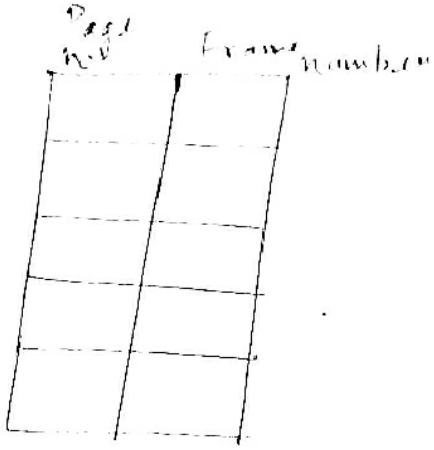
Now, using common sense

$$p = L/P$$

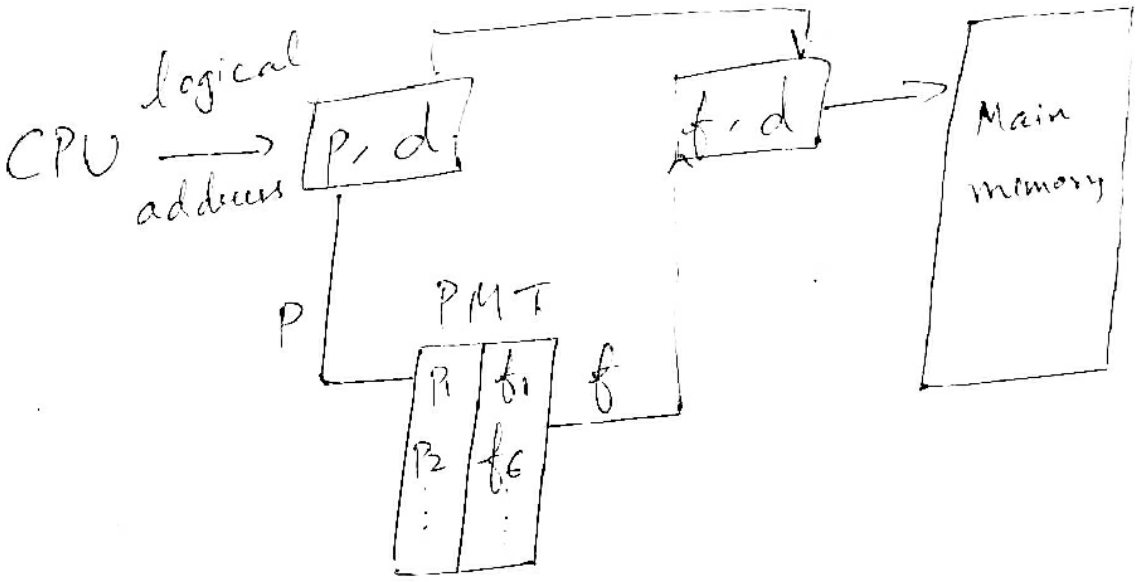
$$d = L \bmod P$$

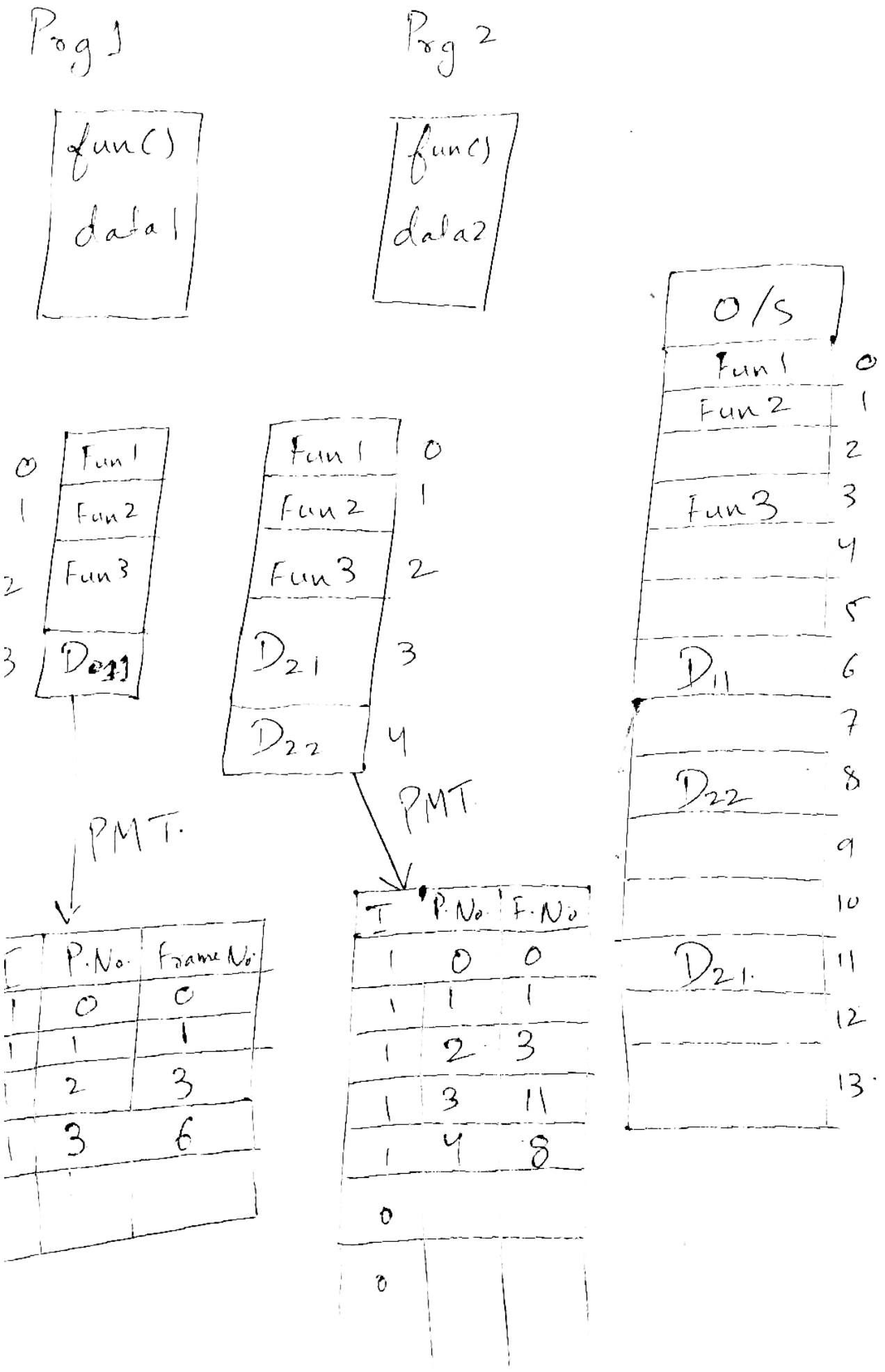
To connect pages (in CPU) with frames (in memory), we need a Page Map Table (PMT).

PMT



Hence the mapping becomes




Prg 1
Prg 2
fun()
data1
fun()
data2
O/S
Fun1
Fun2
Fun3
D11
D21
D22
PMT
PMT
P.No
Frame No
T
P.No
F.No

Pages can be shared given that fun() function does not modify itself while executing

If prog. changes itself then you have to load different copies of the same program.