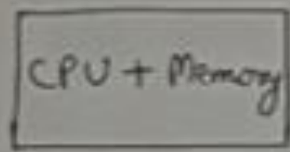
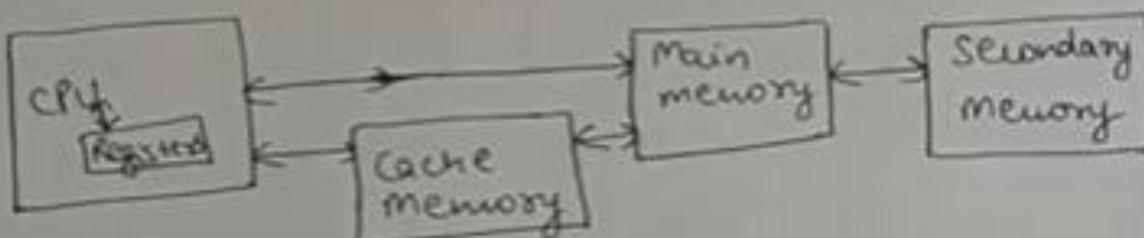
Memory Management



Computer

3 criteria to choose memory

① Size ② Access Time ③ Cost



Registers → It is a temporary Storage area built in a CPU. Access time of Register is below 10ns, and registers have lowest capacity ie of few KB of words.

Cache Memory → It is a high speed memory. The purpose of cache memory is to store those programs that are repeatedly used or likely to be used in the near future.

Memory Management Techniques

It is categorized into 2 types

- Contigous
  - fixed partition Scheme
  - Multiple partition Scheme
- non-contiguous
  - paging
  - Segmen-tation

→ **Fixed Partition Scheme**

- no. of partitions are fixed in memory.
- Size of each partition may be same or may be different.
- for eg:- we made 8 partitions in the memory. So, the no. 8 is fixed.

  In each partition only one process can be placed. In this case, we can place only 8 processes at a time. So, the degree of multiprogramming will be restricted.

If one of the partition is of size 50KB, and we have a process of size 20KB, in that case 30 KB will be wasted. This wastage of 30KB space is called as <u>Internal fragmentation</u>.

<u>Two problems in fixed partition scheme</u>

① Degree of multiprogramming is restricted

② Internal fragmentation.

→ To avoid these two problems we move to Variable partition scheme.

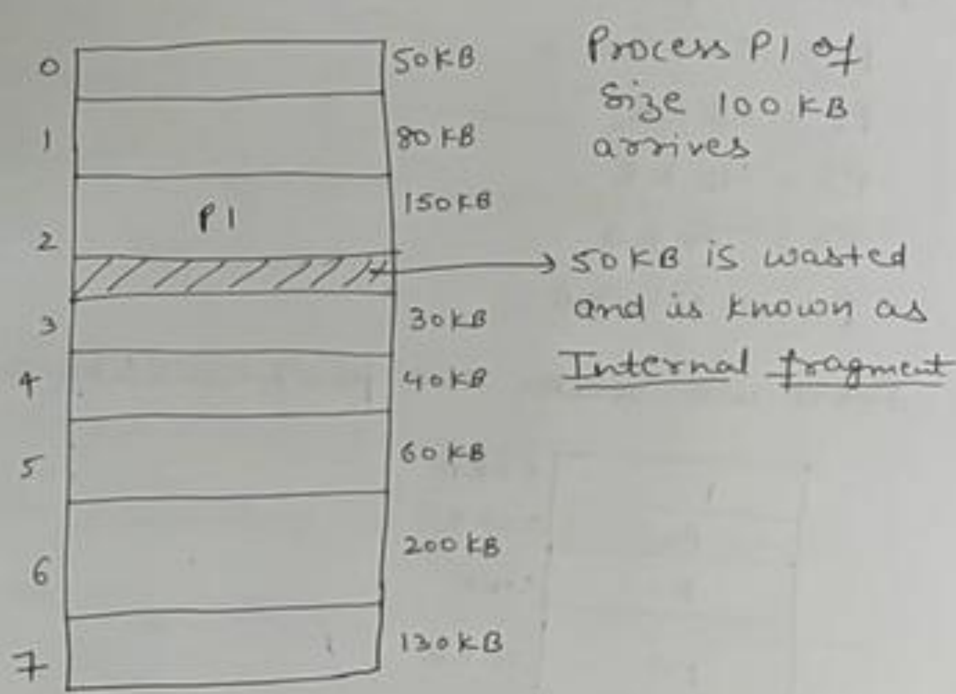
→ <u>Variable partition Scheme</u>

- In this, whenever the request of the process arrives accordingly partition is made in memory.
- For eg:- If a process of size 50KB arrives, a partition of 50KB is made in a memory.
- In this case, there is no restriction on degree of Multiprogramming, as we can make

as many partition as possible till the memory is free.

- Let us assume, after some various process are allocated and only 50KB space is available in memory. And after sometime, a process of size 80KB arrives, but we have only 50 KB space left, ~~so~~ Therefore, we cannot accomodate the process of size 80KB. Hence, this 50KB space is wasted and this wasted 50KB space is known as <u>External fragmentation</u>

→ <u>Example of fixed partition</u>



→ Example of Variable partition

If the request is like this -:

P1 = 50KB

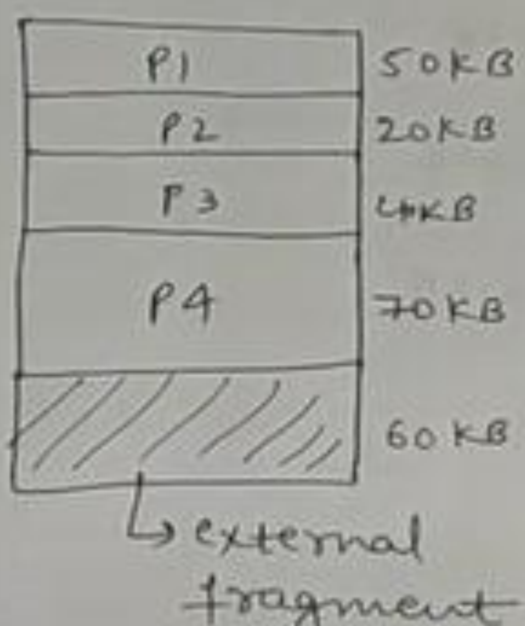
P2 = 20KB

P3 = 40KB

P4 = 70KB

P5 = 100KB

then we make the partitions like this -!



(only 60KB is left memory is full now)

Therefore, we can not allocate P5 of size 100KB.

→ Partition Allocation Methods

first fit

Best fit

worst fit

P1	100KB
	20KB
	30KB
	5KB
	10KB

P1 = 20KB

→ Partition Allocation Methods

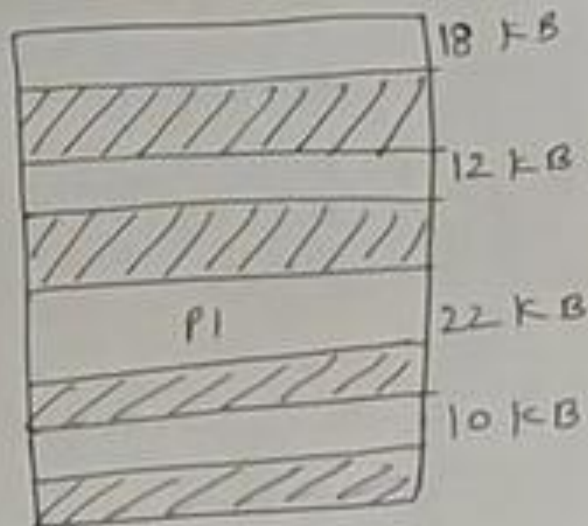
① First fit → In this, allocate the process in a partition which is first sufficient partition from the top of the memory.

② Best fit → In this, allocate the process in a partition which is the Smallest Sufficient partition among the free available partition. To find the Smallest Sufficient partition it requires to search all the free partitions in the memory.

③ Worst fit → In this, allocate the process in a partition which is largest sufficient among the free available partition. To find the largest partition it requires to search all the free partitions in the memory.

→ Example of first fit

$P1 = 20KB$



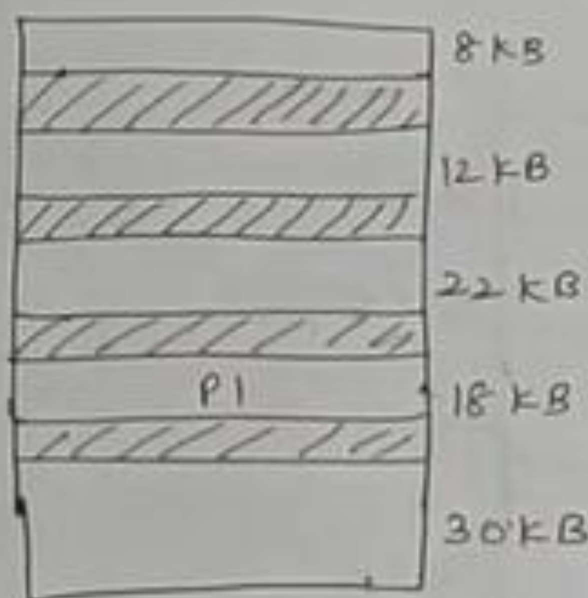
□ → free partition

▨ → busy partition

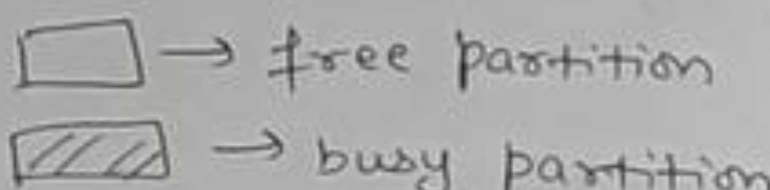
Suppose, Program P1 arrives of size 20KB.

In case of first fit, P1 (20KB) will be allocated in the ~~block~~ frame of size 22KB.

→ Example of Best fit

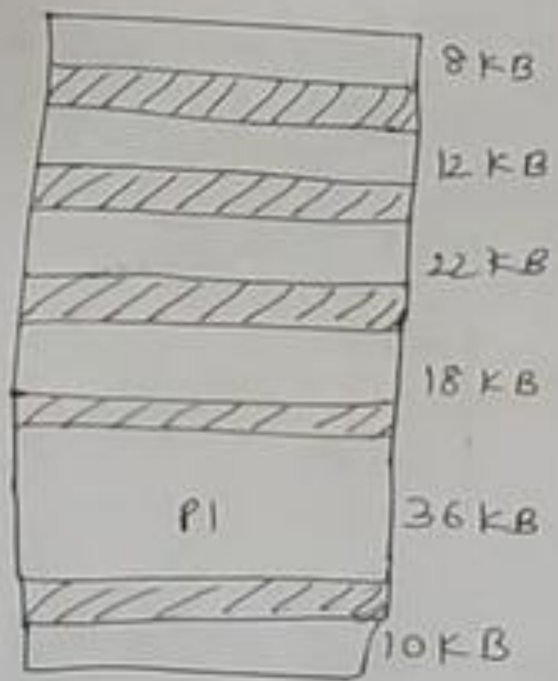


P1 = 16 KB



Suppose, P1 of size 16 KB arrives,
In case of Best fit, P1 will be placed at 18 KB partition.

→ Example of worst fit



$P1 = 16KB$

→ free partition

→ Busy partition

Suppose, P1 of size 16 KB arrives. In case of worst fit P1 will be ~~pa~~ placed at 36 KB partition.

Memory Management Techniques

It is categorized into 2 types

- Contigous
  - fixed partition Scheme
  - Multiple partition Scheme
- non-Contiguous
  - paging
  - Segmen-tation

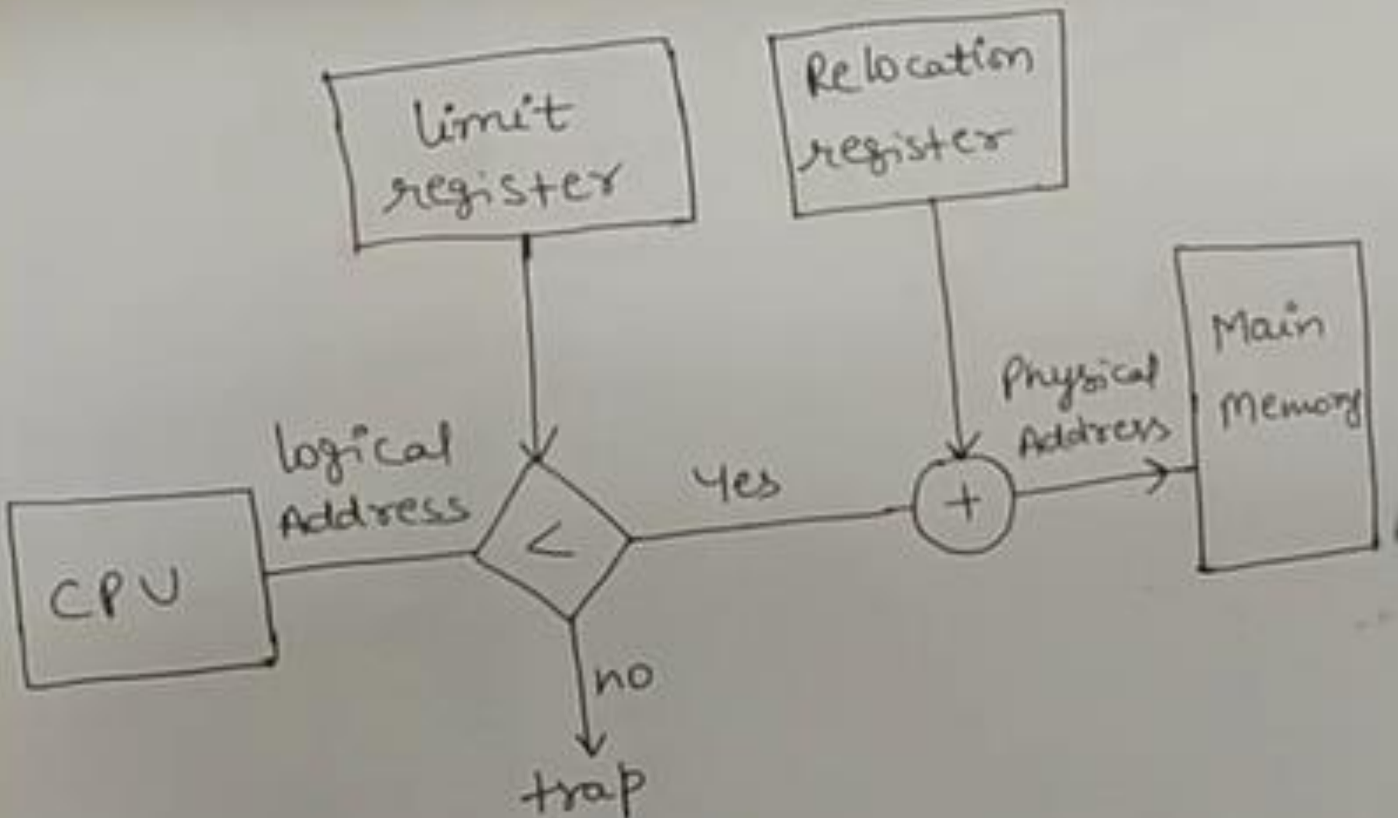
→ Fixed partition Scheme

- no. of partitions are fixed in memory.
- Size of each partition may be same or may be different.
- For eg:- we made 8 partitions in the memory. So, the no. 8 is fixed.
  In each partition only one process can be placed. In this case, we can place only 8 processes at a time.
  So, the degree of multiprogramming will be restricted.

→ Contigious Memory Allocation

Address Translation

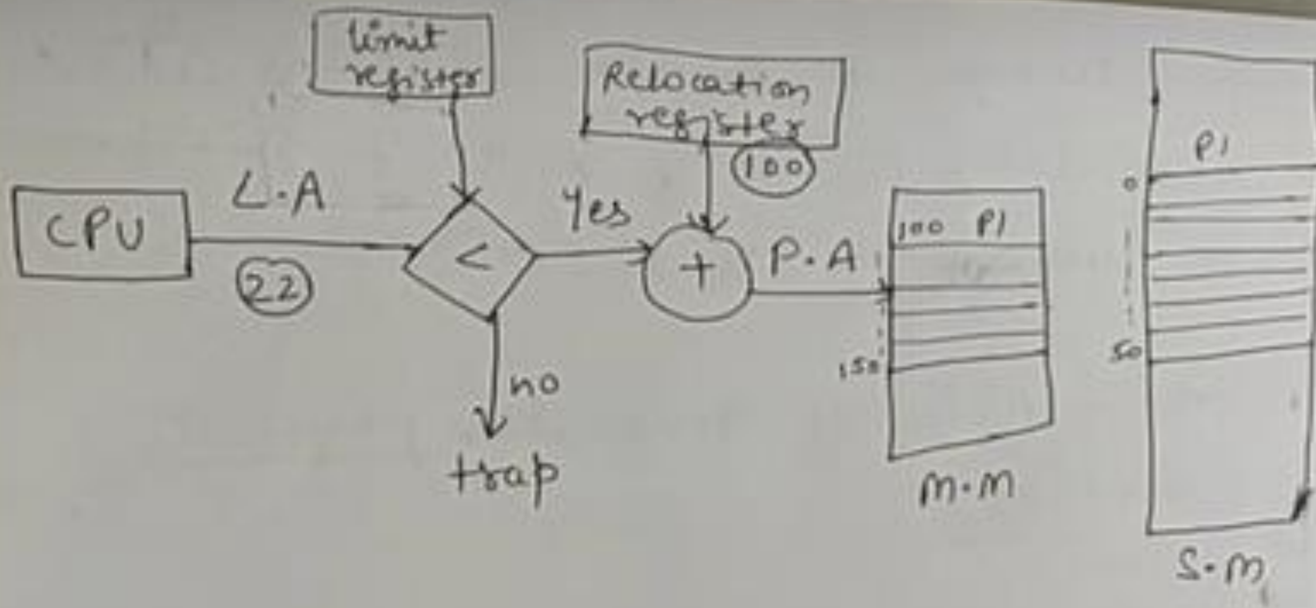
- logical Address to Physical Address



→ Two issues we face in Contiguous memory Allocation —:

① Fragmentation → internal, External

② we have to translate logical Address to physical Address.



<u>Explanation of this above diagram</u>

Suppose, we have a program P1 which contains 50 instructions in Secondary memory. CPU wants to execute instruction no. 22. CPU generates the logical address (LA) 22. And the programs which are to be executed will reside in the main memory. So, this P1 program is stored in the Main memory. Since main Memory generates the Physical address and CPU generates the logical address, we have to translate LA to PA.

So, when CPU generates the logical Address 22, we will check the limit register. Limit register contains the size of the program, in our example the size of the program is 50. So, we will check 22<50, if no it will go to trap, and if yes, then we will add the L.A (22) to relocation register to get the physical address. The relocation register contains the base address of the program in main memory. i.e in our example base address is 100. So, we add 100+22 ⇒ 122 → which is the physical address.

As, we use contigious memory allocation the instructions are stored in a continuous manner. In this way we

Can translate Logical address (LA) to physical Address (P.A)

→ Note:

1. Set of logical Addresses is known as Logical Address Space (L.A.S)

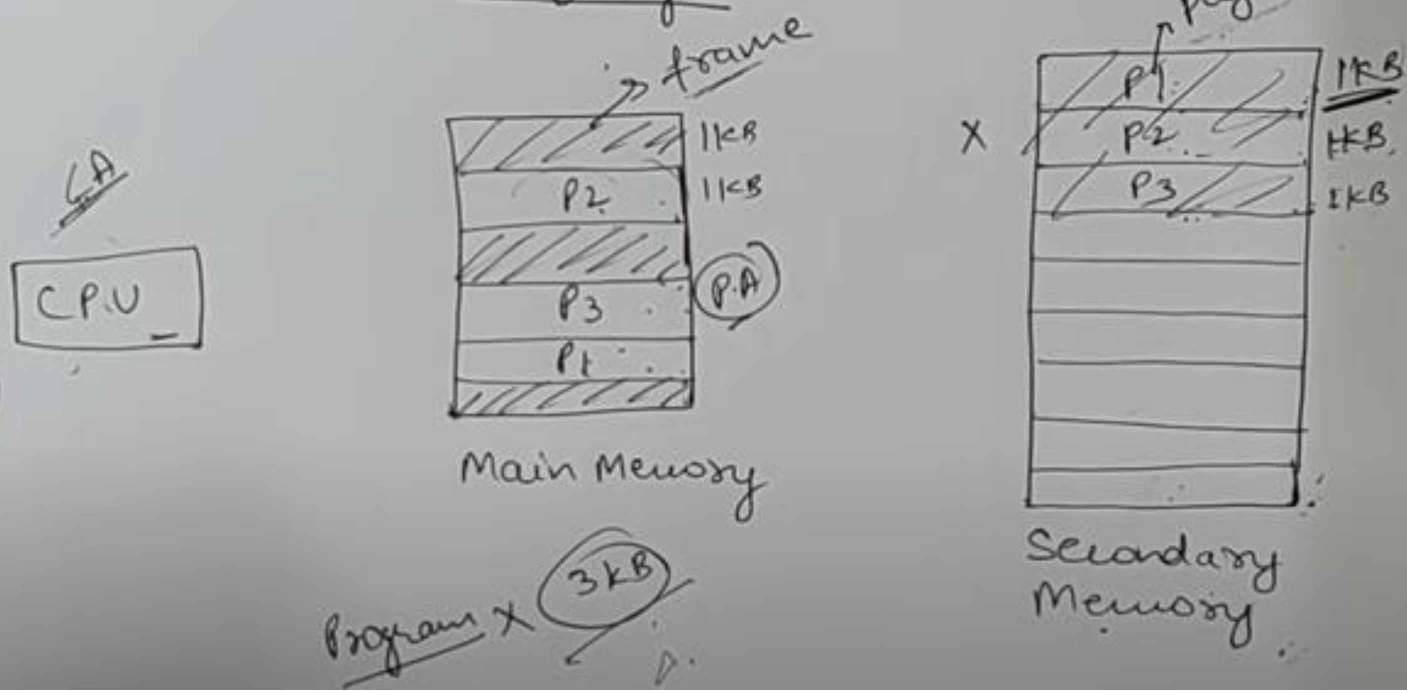
2. Set of physical Addresses is known as Physical Address Space (P.A.S)

3. CPU generates Logical Address (LA)

4. Main memory Generates Physical Address (P.A).

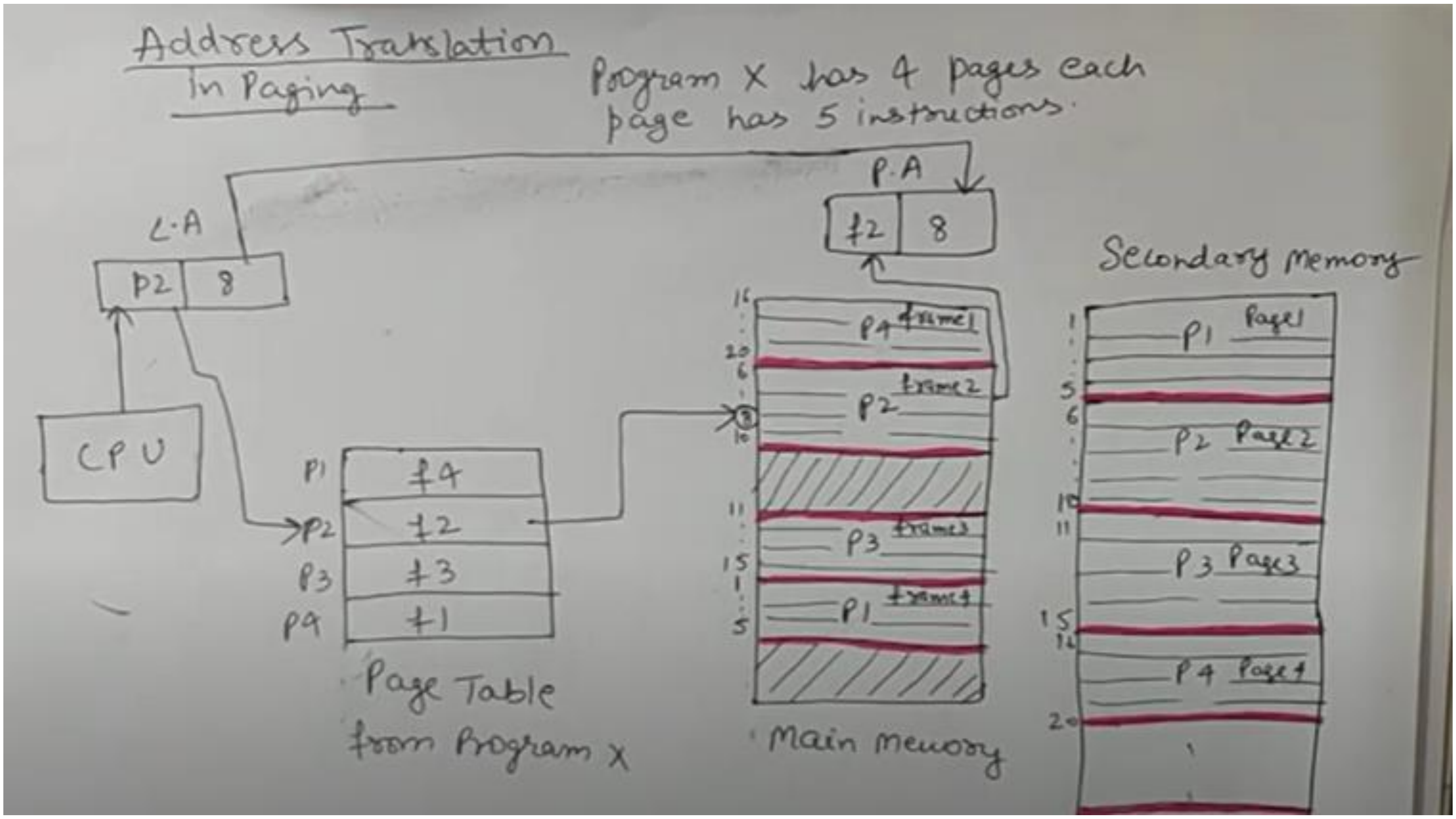
Non-Contiguous Memory Allocation

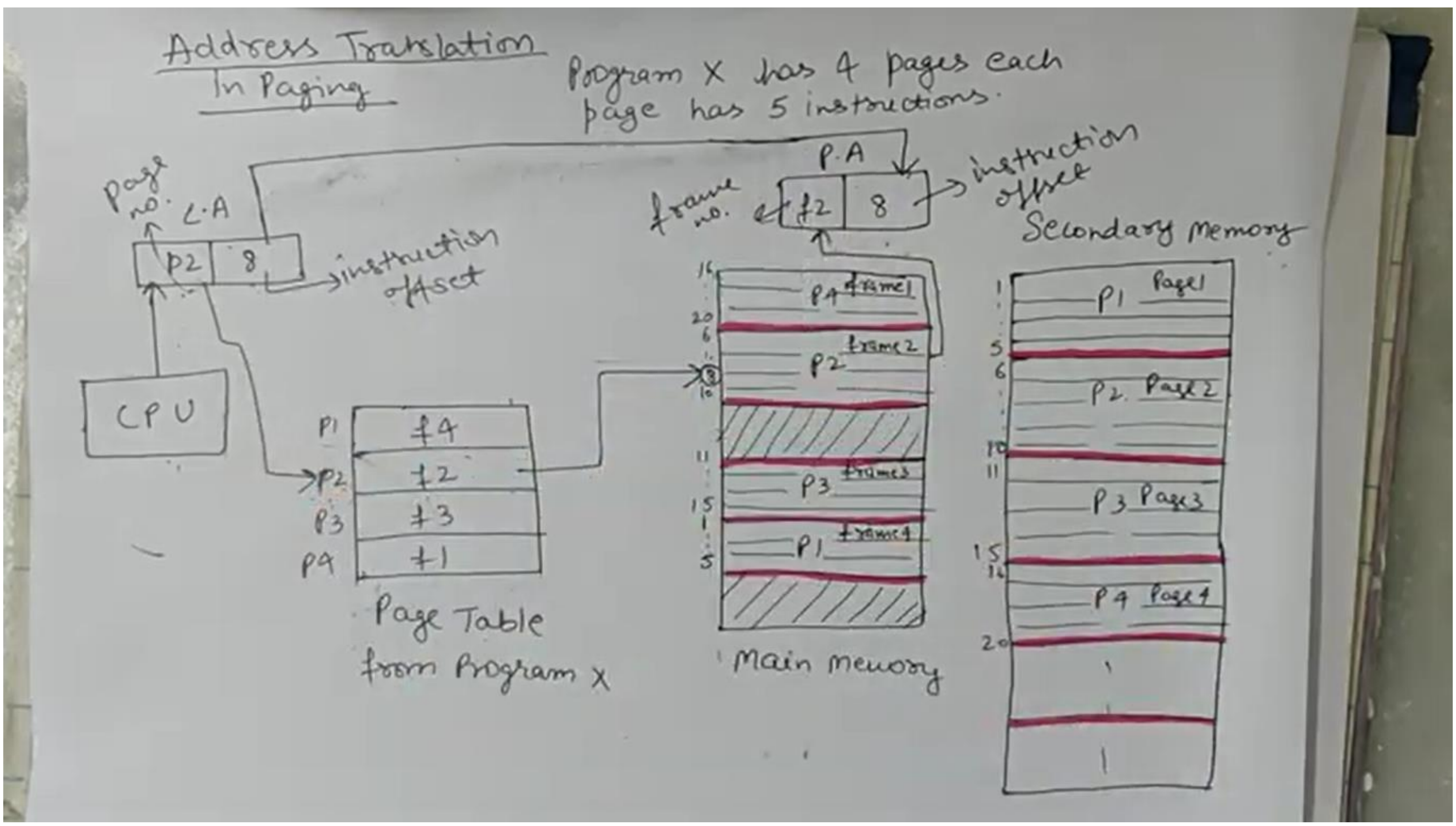
Paging



Defination of paging.

Paging is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory. Operating system reads data from secondary storage in blocks called <u>pages</u>, all of which have identical size. And the Blocks (identical size) in main memory are called <u>frames</u>.


Address Translation
In Paging
Program X has 4 pages each page has 5 instructions.
L.A
P2
8
P.A
f2
8
CPU
P1 f4
P2 f2
P3 f3
P4 f1
Page Table
from Program X
P4 frame1
P2 frame2
P3 frame3
P1 frame4
Main memory
Secondary memory
P1 Page1
P2 Page2
P3 Page3
P4 Page4

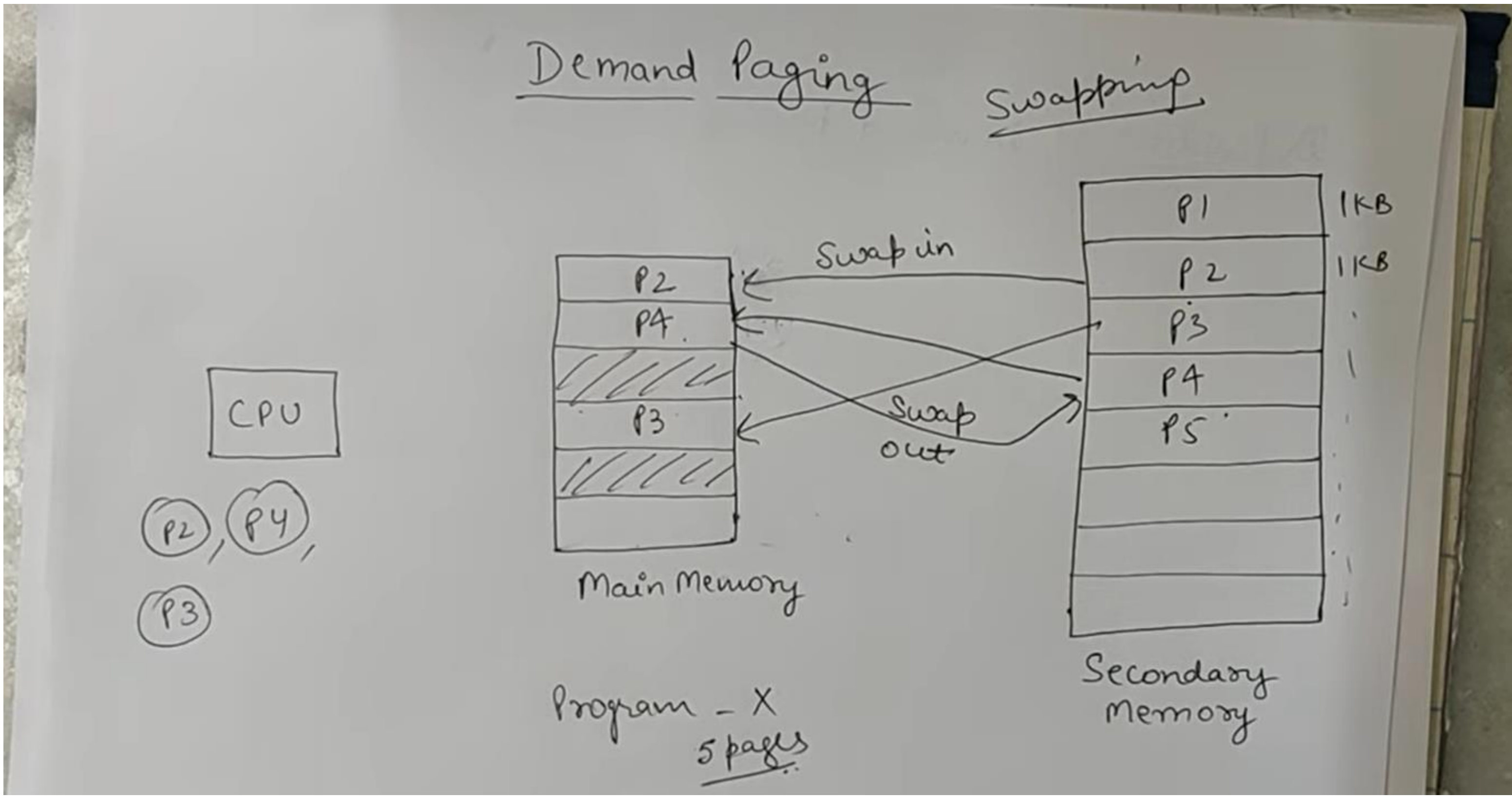

Address Translation
In Paging
Program X has 4 pages each page has 5 instructions.
page no.
L.A
P2
8
instruction offset
P.A
frame no.
f2
8
instruction offset
CPU
P1 f4
P2 f2
P3 f3
P4 f1
Page Table
from Program X
P4 frame1
P2 frame2
P3 frame3
P1 frame4
Main memory
Secondary memory
P1 Page1
P2 Page2
P3 Page3
P4 Page4

Foreg:- ...

Defination of Page Table – It is the data structure stored in Main Memory that stores the mapping between Virtual addresses and Physical addresses.

Now CPU generates Logical address. The logical address consists of 2 parts — ① Page no. ② Instruction offset then CPU refers the Page Table and finds out which page no. is present at which frame no. and hence translate the Physical address. Physical address consists of two parts — ① frame no. ② Instruction Offset. foreg:- CPU wants to execute Page no. 1 and inside Page no. 1 instruction no. 5. i.e $\underset{L \cdot A}{\boxed{P_1 \mid 5}}$

P_1 is page no. and 5 is the instruction offset. now CPU refers Main memory and finds out (say P_1 ~~seen~~ is present at frame no. 4 (f_4). $\underset{P \cdot A}{\boxed{f_4 \mid 5}}$ The instruction offset is same 5. CPU goes to frame no. 4 and picks 5th instruction and executes it and gives the output. This is how we translate L·A to P·A in paging.

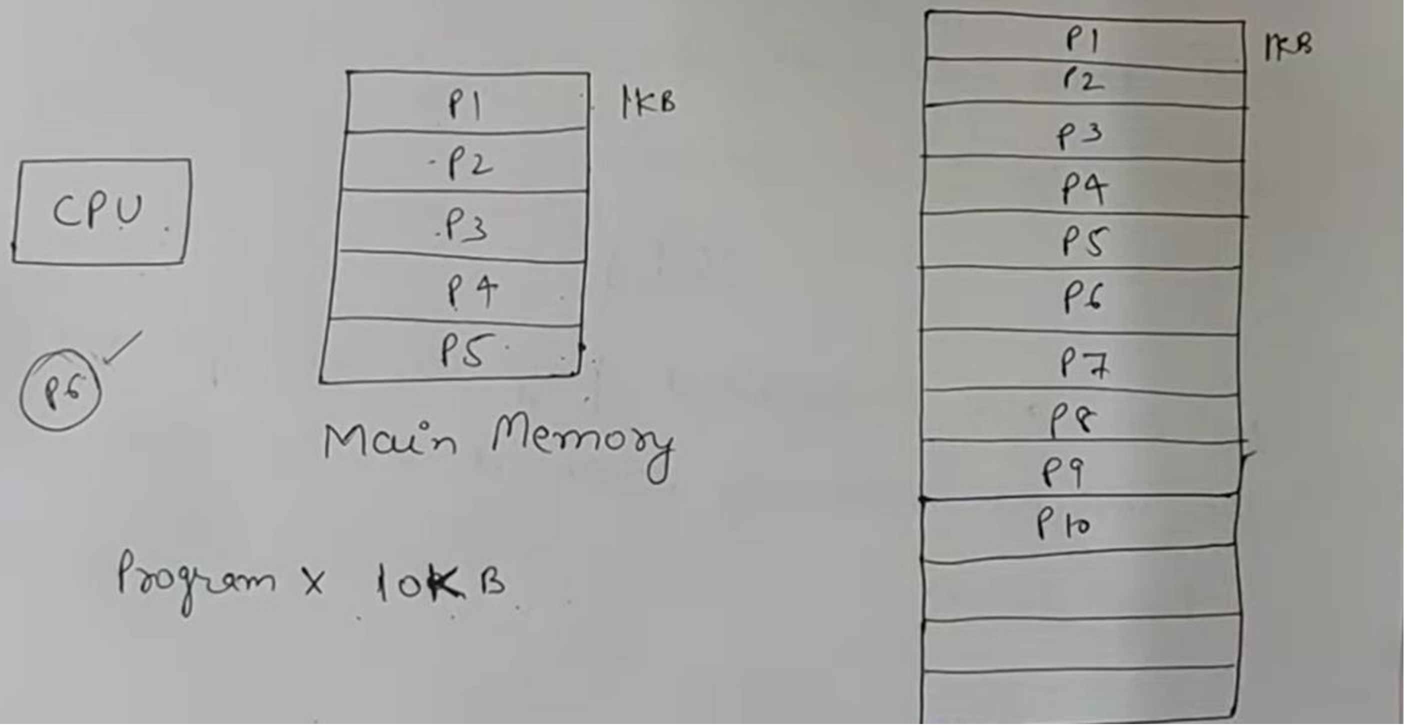


Demand Paging

The Program resides in the secondary memory, and pages are loaded in main memory only on demand not in advance.

Defination of Demand Paging → It is a type of swapping in which pages are copied from secondary memory to main memory when they are needed.

Page fault

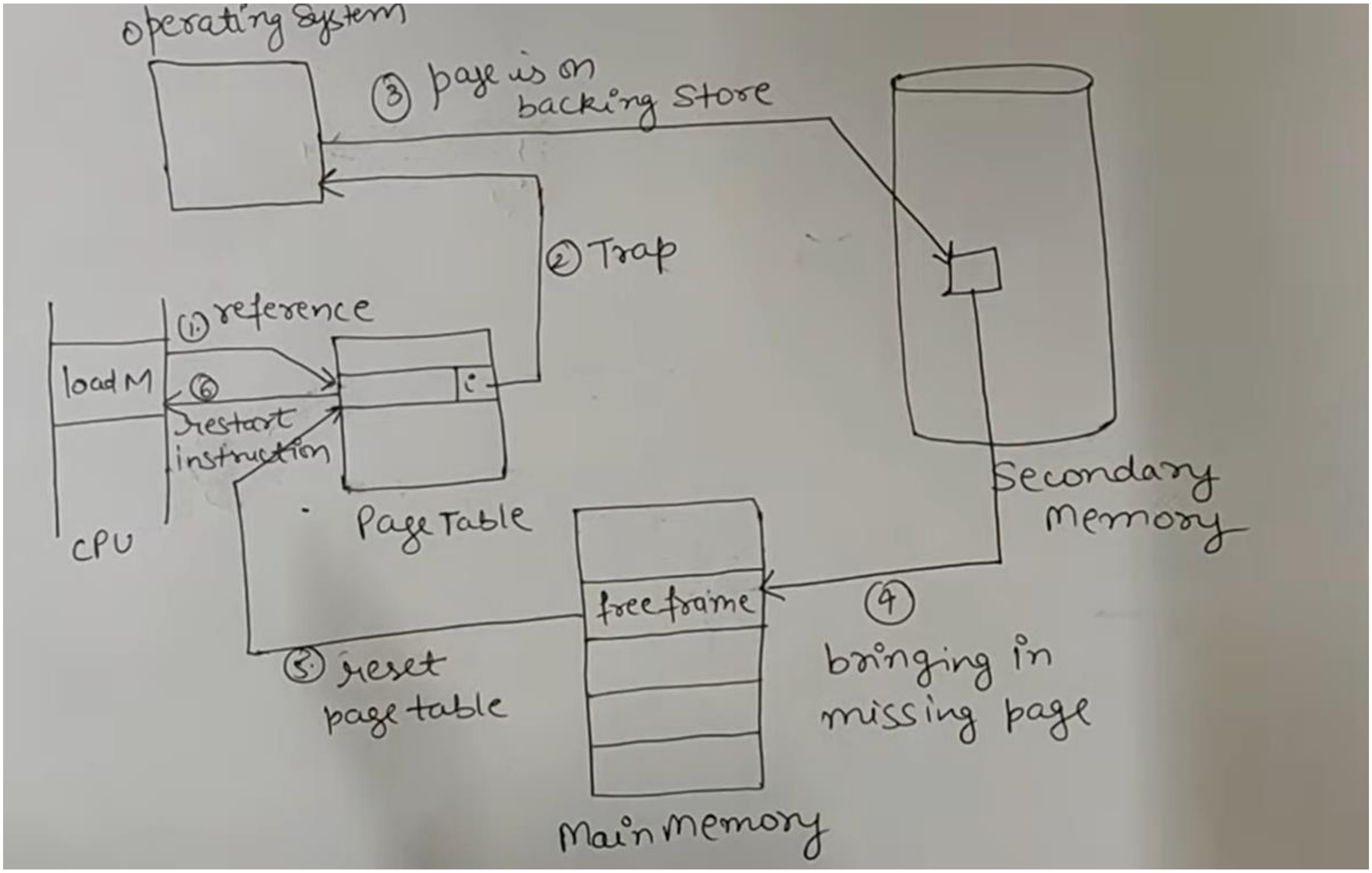


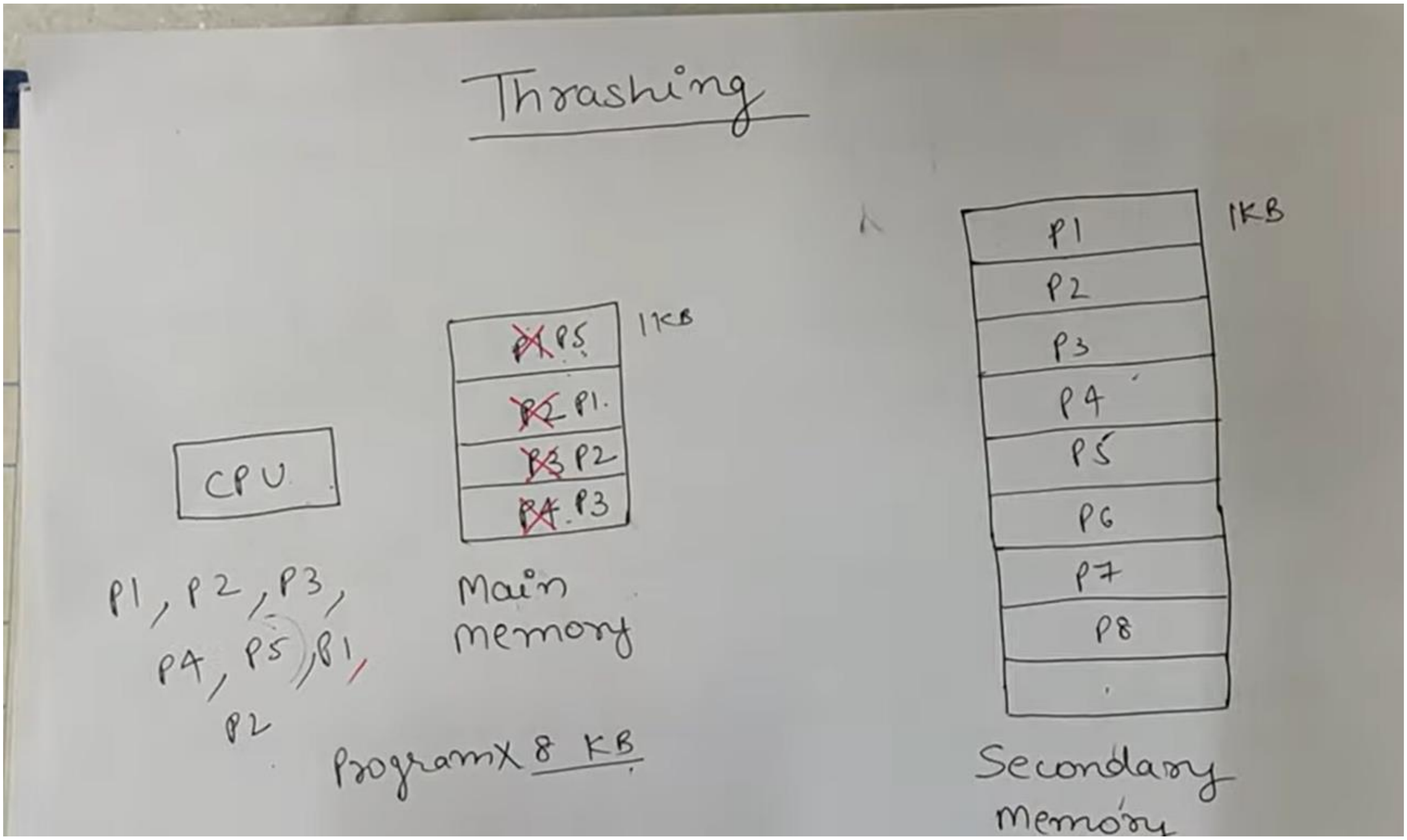
Defination of Page fault → when a processor wants to access a page, and that particular page is not ~~____~~ currently present in the main memory, then page fault occurs.

OR

when the demanded page is not present in the main memory then it is called as **Page fault**.

Page Fault Solution



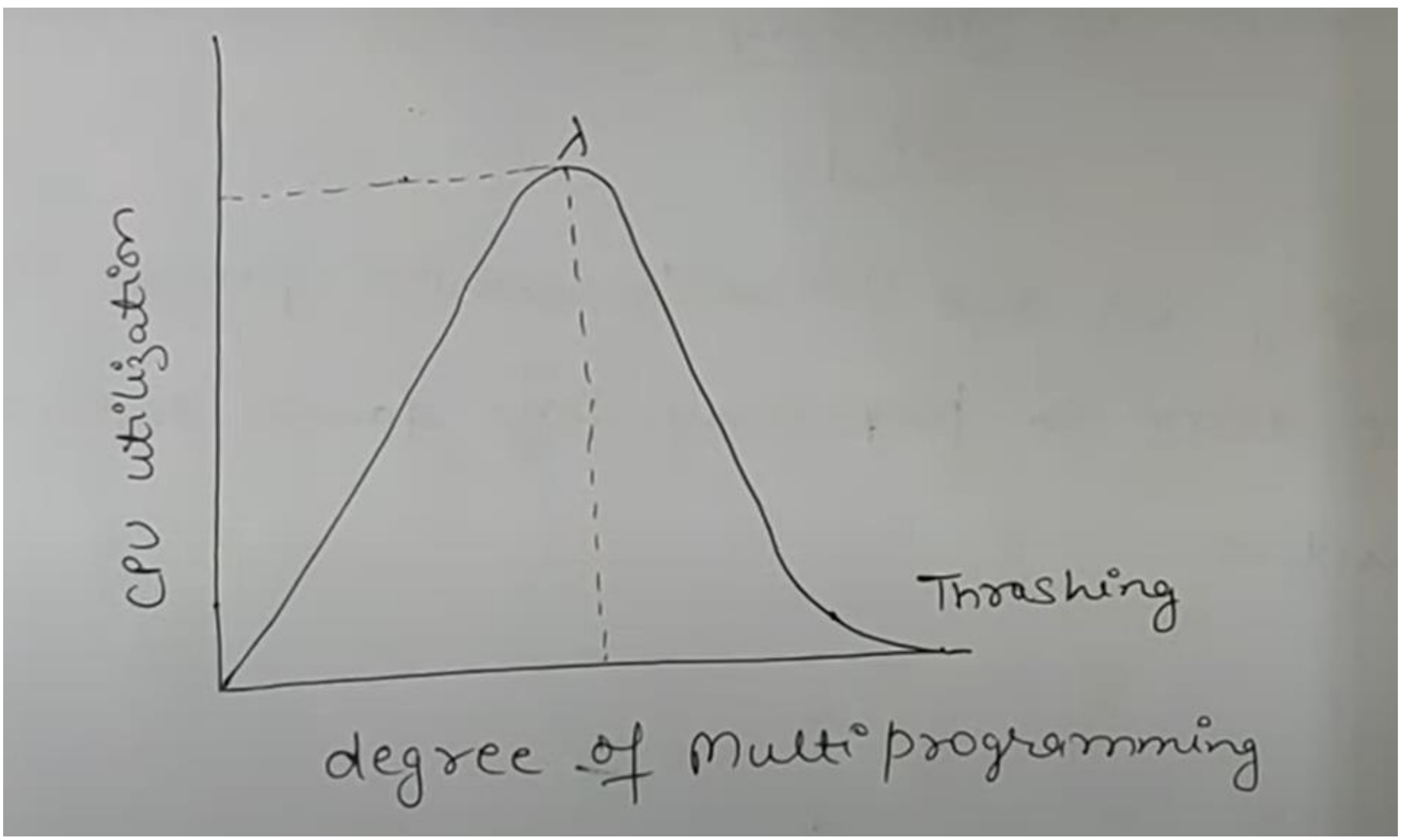


1. In the initial stage when we increase the degree of Multiprogramming, CPU utilization is very high upto λ.
2. Further, if we increase the degree of Multipr-ogramming, CPU utilization is drastically falling down.
3. This situation in the system is <u>Thrashing</u>.
4. Thrashing degrades the performance of System.

5. There can be a situation when main memory is full of pages that are accessed frequently. A page fault will occur if the required page is not present.

6. In order to make space for swapping in the required page, on of the frequently accessed page is swapped out.

7. Soon, the swapped out page is required for execution and this again results in



Degree of Multiprogramming: Number of pages are being added.

Defination of Thrashing → high paging activity is known as Thrashing.

OR

Thrashing is the Situation where process spends more time in processing page faults rather than execution.

Virtual Memory

- Virtual Memory gives an illusion to the programmer that programs of larger size than actual physical memory can be executed.
- Virtual memory doesn't really exists, but the part of secondary memory are made as virtual memory.
- Defination of Virtual Memory
  Virtual Memory is a feature in O.S, where large programs can store themselves in form of pages

Defination of Virtual Memory
Virtual Memory is a feature in O.S, where large programs can store themselves in form of pages
while their execution and only the required

pages or portions of processes are loaded into the main memory.

Virtual memory can be implemented by using -:

Demand Paging

Demand Segmentation

Race Condition Defination

when more thane one processes are executing the same code or resource or any shared variable in that condition there is a possibility that the output or the value of that shared variable is wrong, so for that all the processes doing race to say that my output is correct, this condition is known as Race condition.

Critical Section in O.S

do

{

$\boxed{\text{entry Section}}$ → controls the entry into C.S and gets a Lock on resources.

$\boxed{\text{critical Section}}$ -

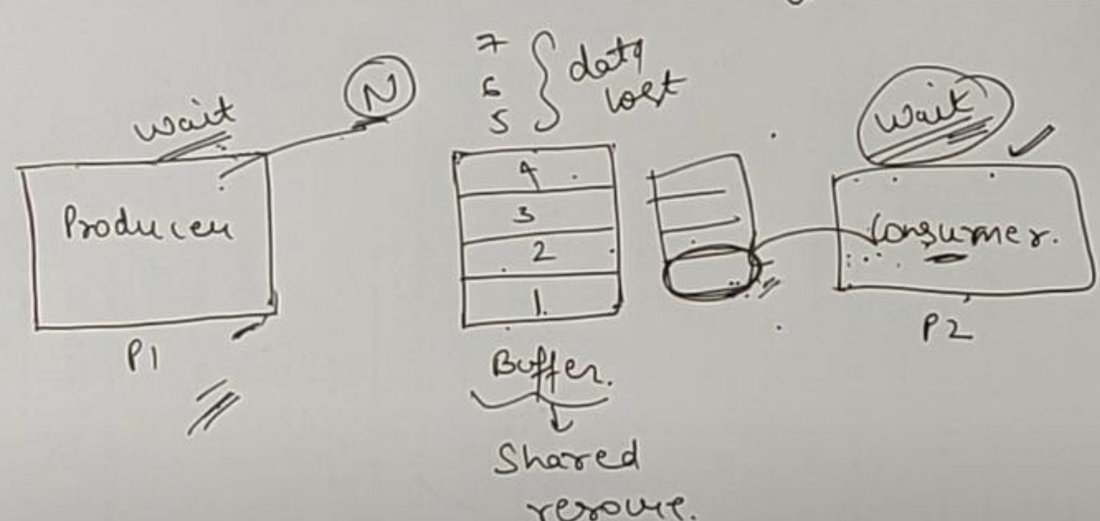
$\boxed{\text{exit Section}}$ → removes the Lock from the resources.

$\boxed{\text{Remainder Section}}$

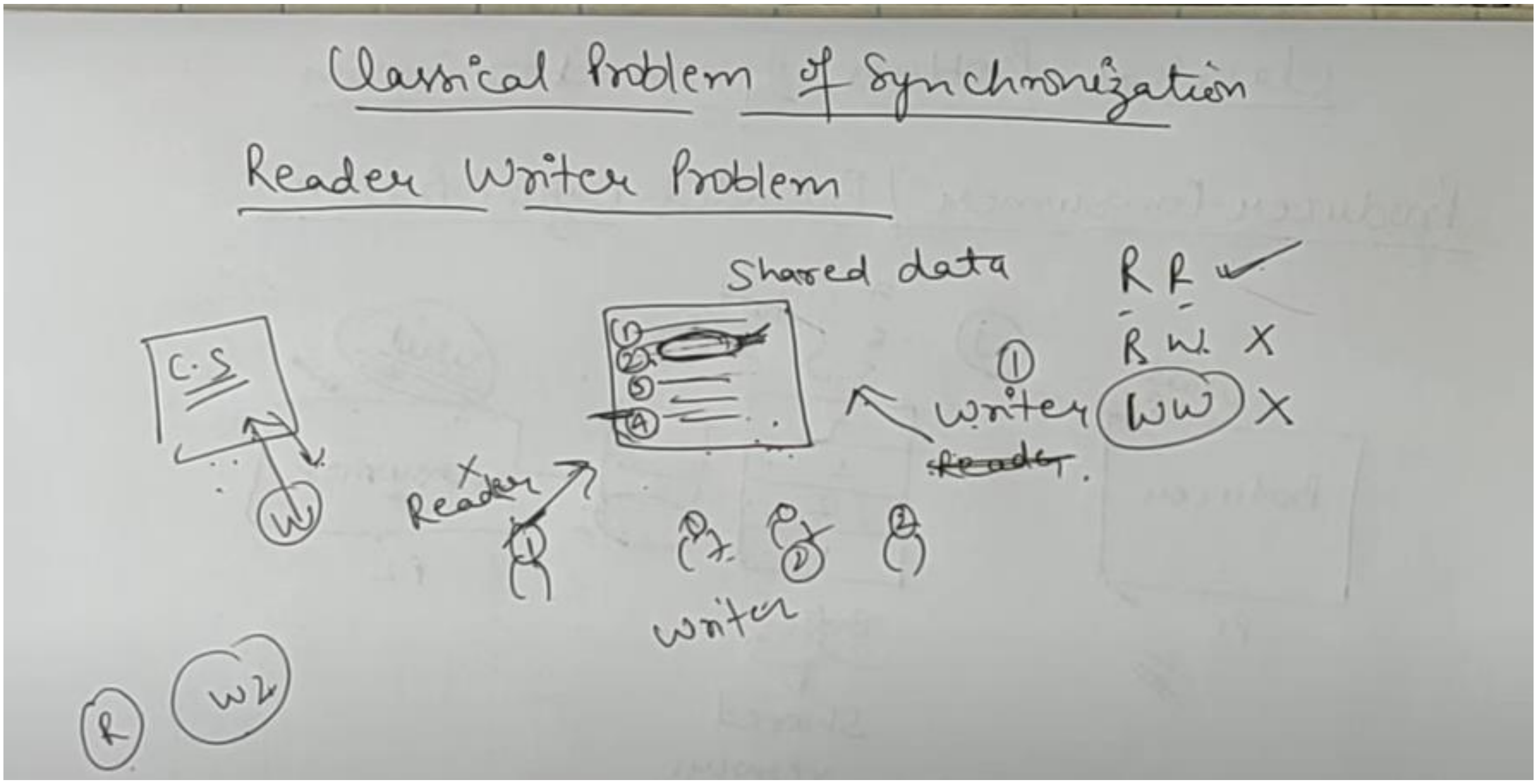
} while (True);

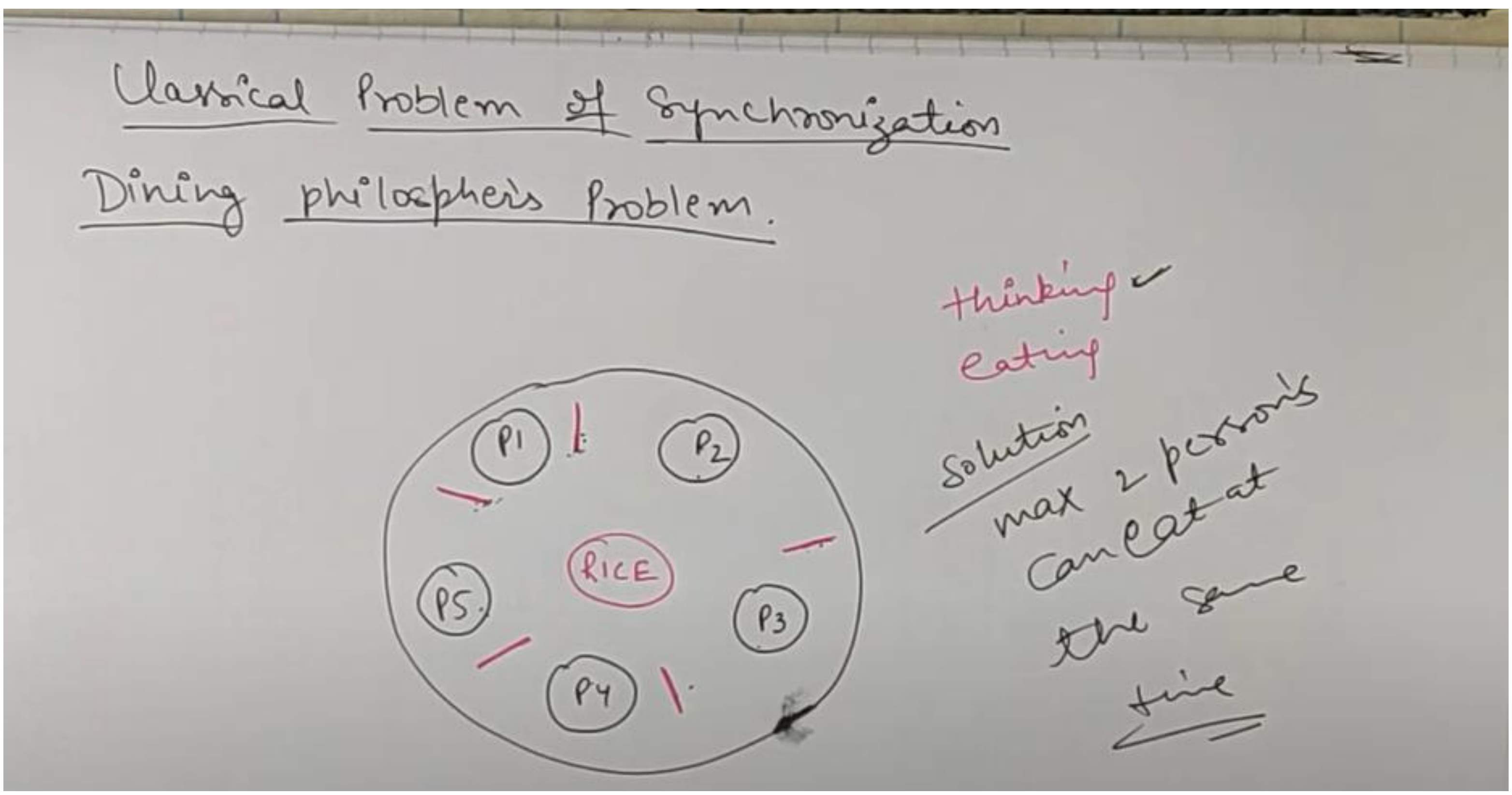
Classical Problem of Synchronization

Producer-Consumer / Bounded Buffer Problem



Producer/Consumer problem is solved using semaphores.





Semaphores in Operating System

Semaphore is an integer variable that solves the critical section problem. After initialization, it can only be accessed by two atomic operations —:

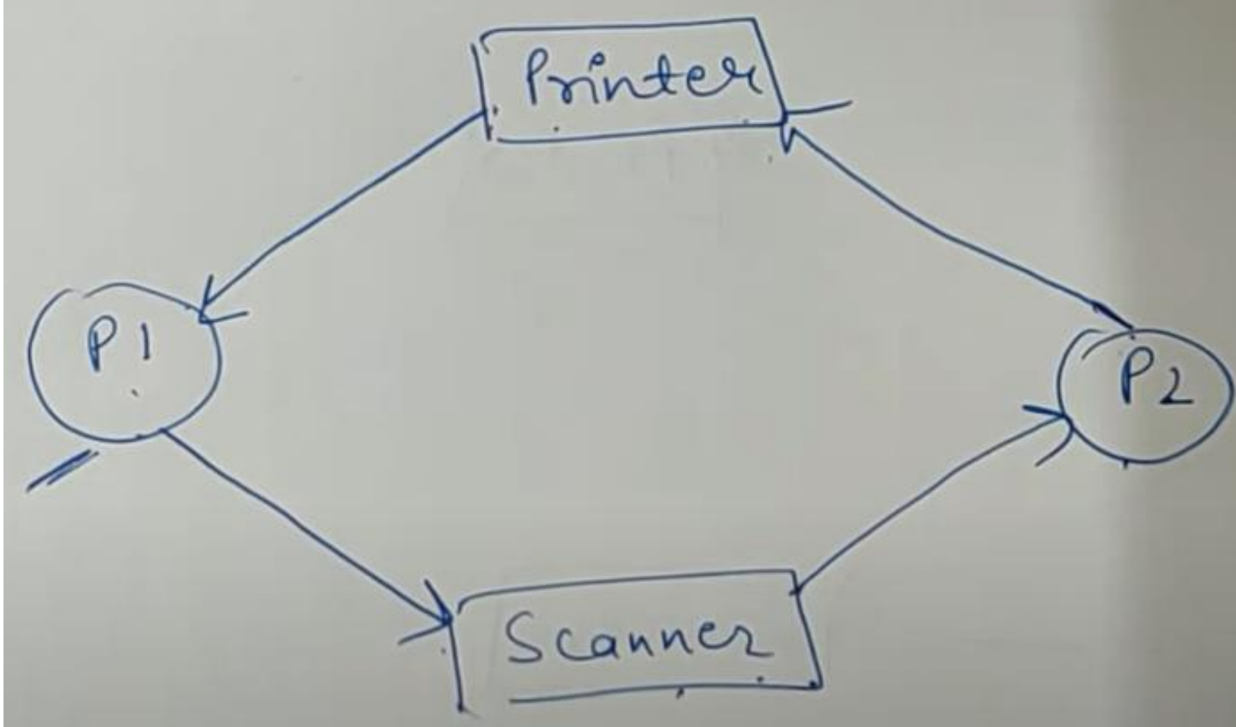
1. wait (s)

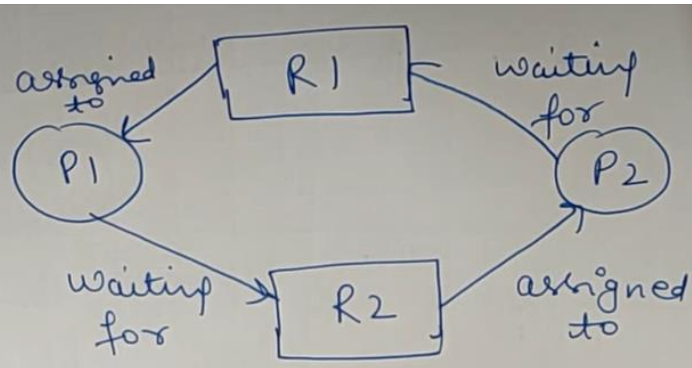
```
{ while (S<=0);
     S = S-1;
}
```

2. Signal (s)

```
{ S = S+1;
}
```

Deadlock in Operating system





Defination of Deadlock → Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process.

→ Four Necessary conditions for deadlock to occur

① Mutal Exclusion → one or more than one resource are non sharable (only one process can use it).

② Hold and wait → A process is holding at least one resource and waiting for another resource.

③ No Preemption → A resource cannot be taken from a process unless the process ~~[illegible]~~ releases that resource.

④ circular wait → A set of processes are waiting for

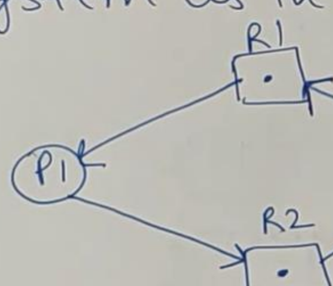
Resource Allocation Graph (RAG)

- Using RAG, deadlock can be easily detected.
- It is a Graph that represents the state of a system pictorially.
- It has vertices and edges.
- It has two types of edges —:
  1. Request edge $P_i \rightarrow R_j$
  2. Assignment edge $R_j \rightarrow P_i$

Rules to detect deadlock using RAG

Rule-1 → In RAG, where all the resources are single instance —:

(a) If a cycle is being formed, then the system is in a deadlock state.



(b) If no cycle is being formed, then the system is not in a deadlock state.

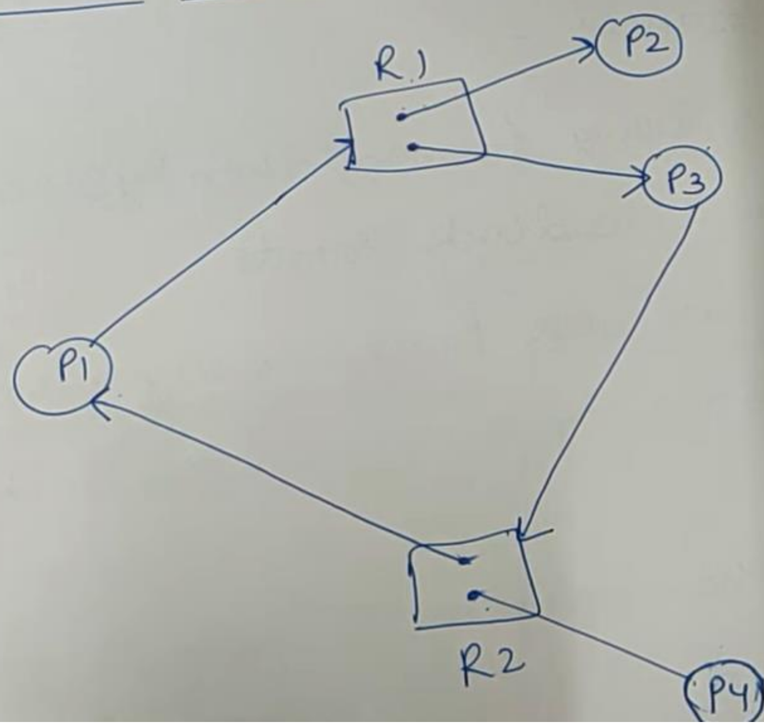
Rule-2 → In RAG, where all the resources are multiple instances :-

(a) If a cycle is being formed, then system may or may not be in a deadlock state.

In this case we use Banker's algorithm to confirm whether a system is in deadlock state or not.

(b) If no cycle, then system is not in a deadlock state.

RAG, with multiple instances but no deadlock



Banker's Algorithm (Deadlock Avoidance Algo)

Total A = 10 B = 5 C = 7

Process	Allocation A B C	Max need A B C	Available A B C	Remaining need A B C
P1	~~0 1 0~~	7 5 3	~~3 3 2~~ ~~2 0 0~~	~~7 4 3 P1~~
P2	~~2 0 0~~	3 2 2	~~5 3 2~~ ~~2 1 1~~	~~1 2 2 P2~~
P3	3 0 2	9 0 2	~~7 4 3~~ ~~0 0 2~~	~~6 0 0 P3~~
P4	~~2 1 1~~	4 2 2	7 4 5 0 1 0	~~2 1 1 P4~~
P5	~~0 0 2~~	5 3 3	7 5 5	~~5 3 1 P5~~
	7 2 5			

Safe seq: P2 → P4 → P5 → P1 → P3

Remaining need = Max need – allocation; for all

A1 (Available 1) = Total (A, B, C) – Allocation (sum (A, dim=1), sum (B, dim=1), sum (C, dim=1)); #dim 1 means column

Available for P1 = A1

Now execute the processing by looking at the remaining need and focusing on the available column;