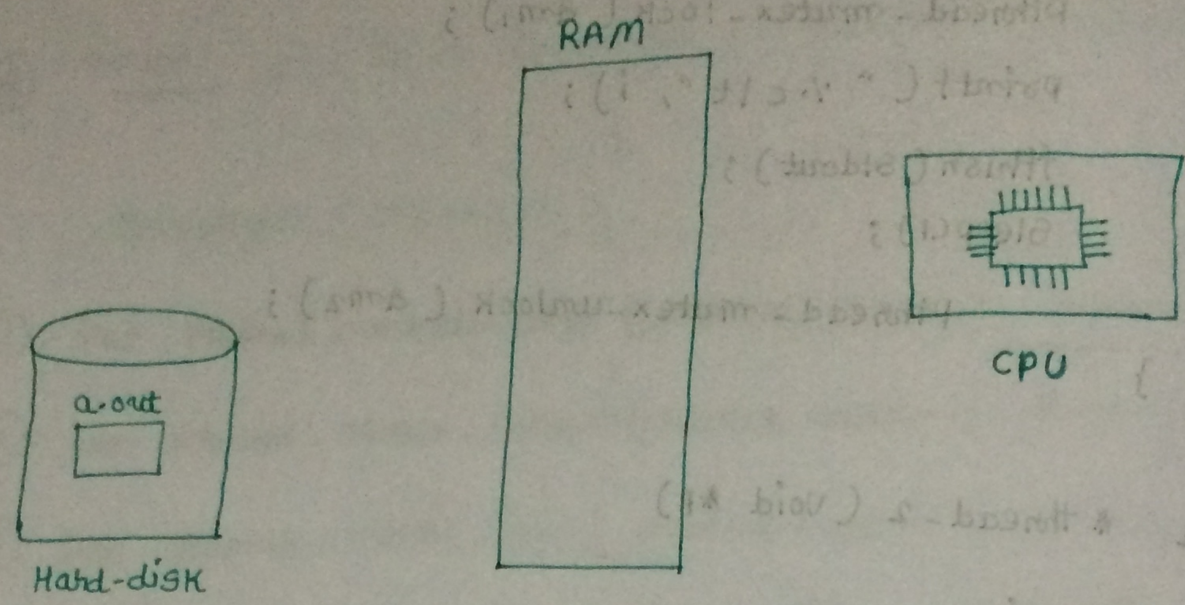
```
for (i='a'; i<='z'; i++)
{
    pthread_mutex_lock(&m1);
    printf("%c\t", i);
    fflush(stdout);
    sleep(1);
        pthread_mutex_unlock(&m2);
}
}
void *thread_2 (void *p)
{ char i;
  for (i='A'; i<='Z'; i++)
  {
    pthread_mutex_lock(&m2);
    printf("%c\n", i);
    sleep(1);
    pthread_mutex_unlock(&m1);
  }
}
main()
{
    pthread_t t1, t2;
    pthread_mutex_init(&m1, NULL);
    pthread_mutex_init(&m2, NULL);
    pthread_mutex_lock(&m2);

        pthread_create(&t1, NULL, thread_1, NULL);
        pthread_create(&t2, NULL, thread_2, NULL);
  pthread_exit(0);
}
```

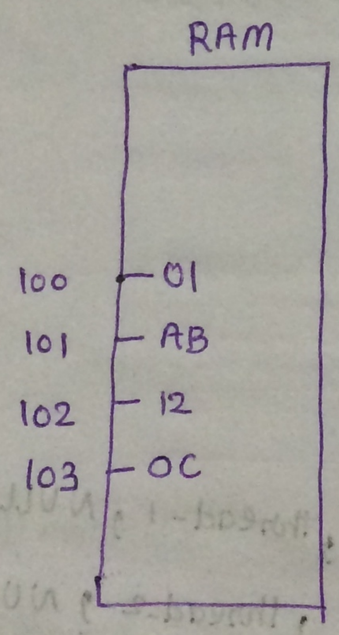
**** Memory Managment ****



⇒ memory manager work is manager the RAM.

⇒ memory manager has to manage the memory in a efficien way such that, there must a memory for the upcoming (new) process.

⇒ If the instruction and where instruction go and seat RAM is done of compile time only that type of code is called executable code.

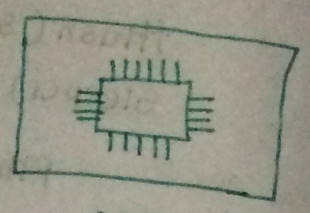


⇒ Cross compile is generate the absolute code.

⇒ If binding is done at load time that code is called as a "relocatable code".

⇒ binding between opcode and memory pospond at runtime if you achieve seprate Hardware unit available is called MMU (memory managment Unit).

nagment ****



the RAM.
memory in a efficien
for the upcoming

go and seat
type of code

generate the

at load time
a

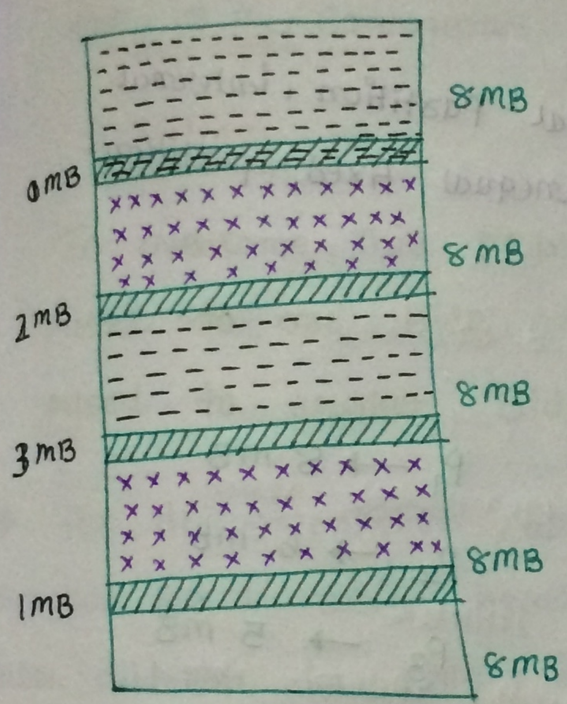
de and
t runtime
Hardware
d
t Unit).

Partioning

- Fixed
  - Equal size
  - unequal size
- Dynamic

① Fixed Equal Size :-

→ RAM are divided into fixed Partition.



$P_1 \rightarrow$ 8 MB (1MB Waste

$P_2 \rightarrow$ 6 MB (1MB Waste

$P_3 \rightarrow$ 5 MB (1MB Waste

$P_4 \rightarrow$ 7 MB (1MB Waste

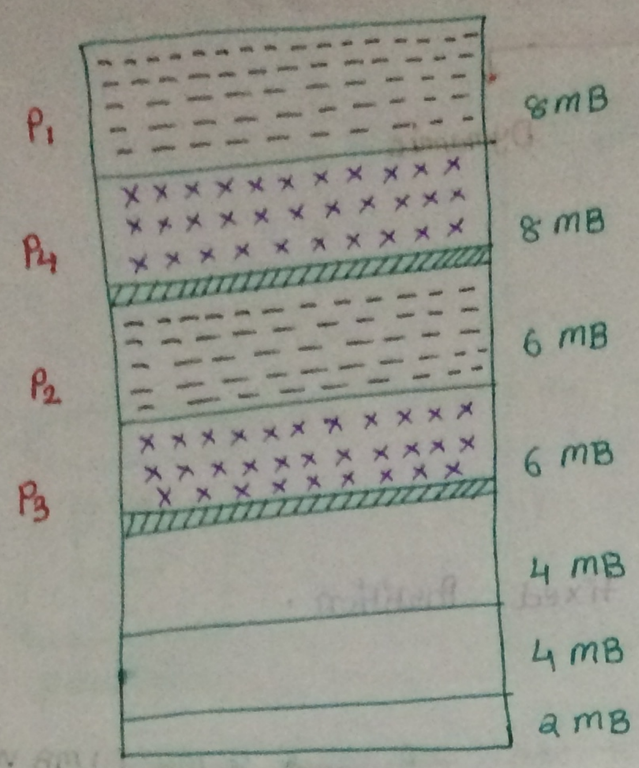
$2 + 3 + 1 = 6$

internal fragmentation

→ for a particular process more than required memory if you allocated that extra memory is called as internal fragmentation.

② Unequal Fixed Partition :-

→ Remove the internal fragmentation so use it.



$P_1 \rightarrow 8$ mB

$P_2 \rightarrow 6$ mB

$P_3 \rightarrow 5$ mB (1 MB Waste)

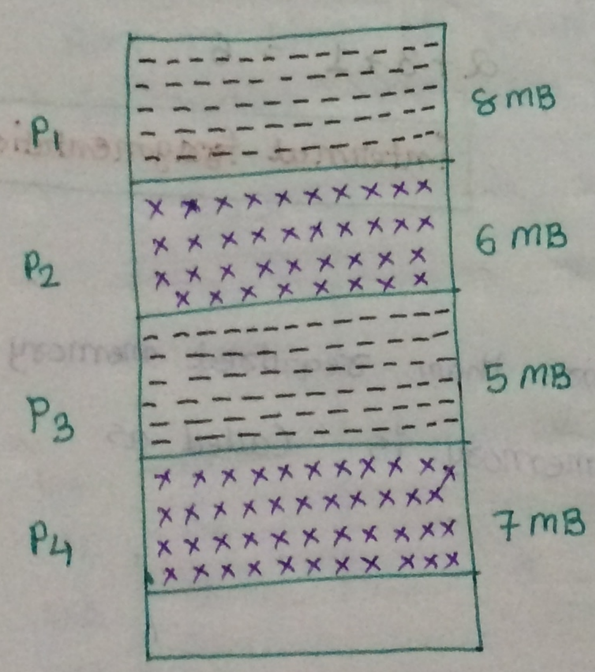
$P_4 \rightarrow 7$ mB (1 MB Waste)

$1+1 = 2$

internal fragmentation

⇒ So compare to fixed equal partition, internal fragmentation is less in unequal fixed partition.

③ Dynamic Partition :-



$P_1 \rightarrow 8$ mB

$P_2 \rightarrow 6$ mB

$P_3 \rightarrow 5$ mB

$P_4 \rightarrow 7$ mB

⇒ No internal fragmentation in dynamic partition.

→ New process is required 12 mB then this problem is called a "External fragmentation".

→ 8 mB
→ 6 mB
→ 5 mB (1 mB waste)
→ 7 mB (1 mB waste)

1 = 2

internal fragmentation

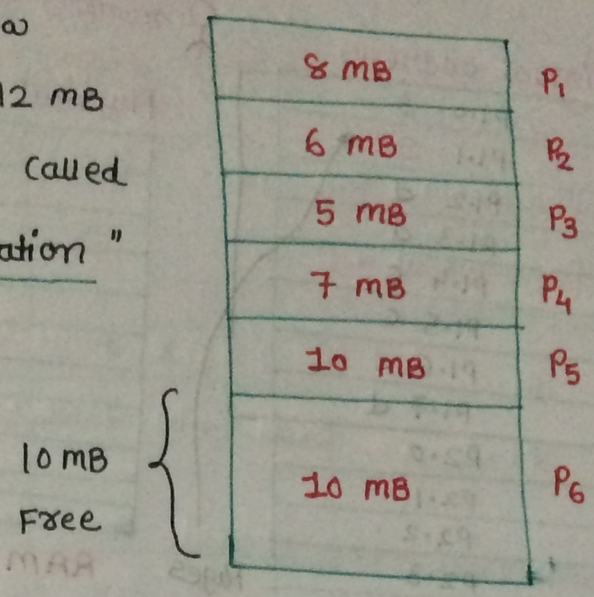
m, internal
ed partition.

mB
5 mB
7 mB

fragmentation
partition.

's problem

⇒ 10 mB Free Then New process is required 12 mB Then this problem is called a "External fragmentation"



⇒ For a process memory is there in the RAM but not in a Contageous fashion.

Solution :-

To overcome this problem all free space must be pused to one side while all the files must be pused to another side is called "defragmentation".

⇒ for defragmentation, all running process must be stopped for a while because they have to be given a new address in RAM, but due to this stoppage. throughput suffers.

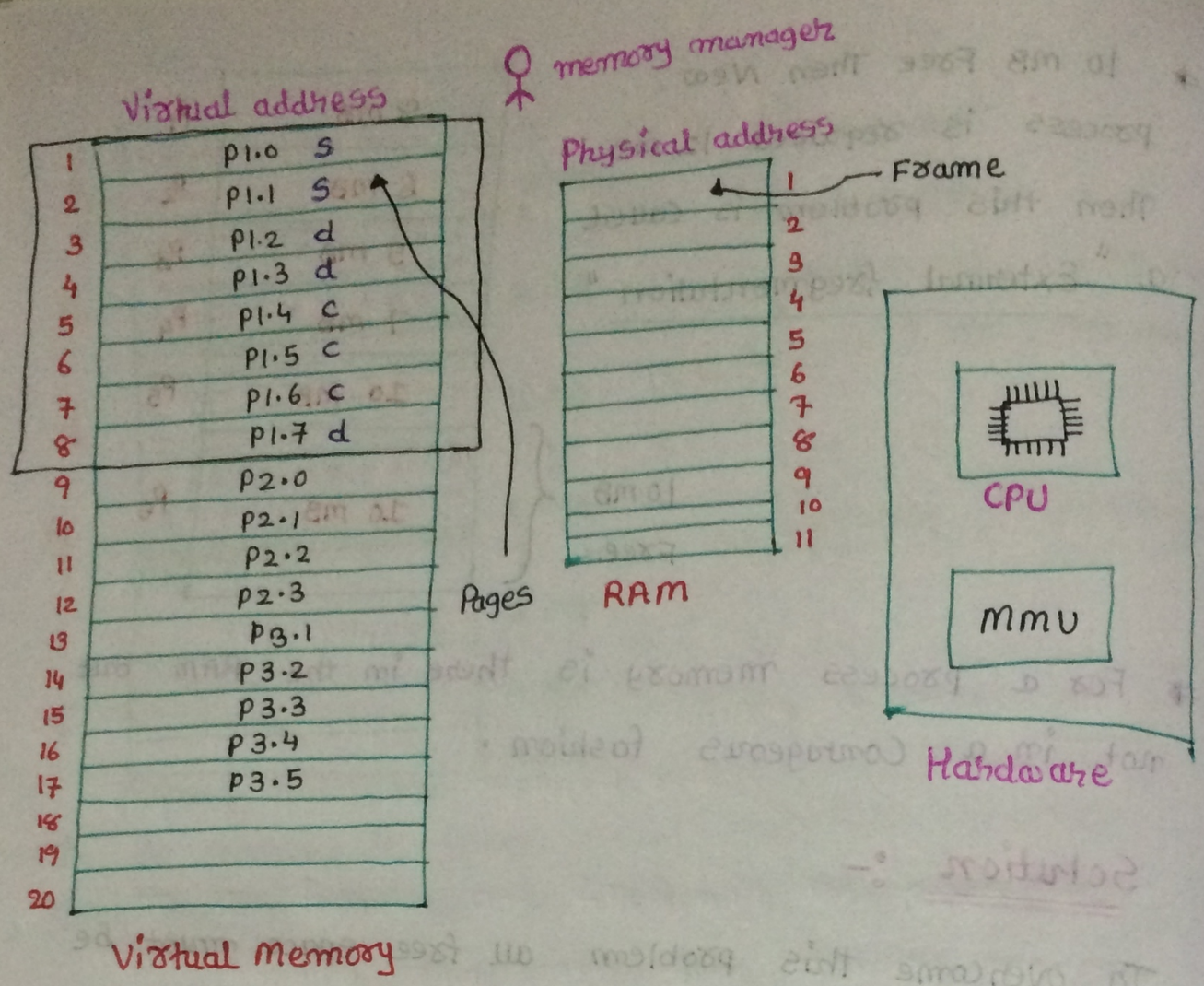
→ no. of process completed per unit time.

⇒ Pages and frames are same.

⇒ no. of pages are more than no. of frames.

⇒ In virtual memory small-small parts is called Pages.

⇒ In RAM small-small parts is called frames.



⇒ ./a.out is run, the process is loaded into virtual memory (RAM).

⇒ logical address consits pagenumber and offset.

⇒ CPU give logical to the MMU.

⇒ MMU is convert the logical address into physical address.

⇒ ex:- logical address is 610

6th page 10th Instruction

⇒ Memory manager maintain one table is called Page table.

Page table

Page Number	Frame Number
1	–
2	–
3	–
4	–
5	–
6	– 1
7	–
⋮	–

→ Page is not present the mmu send a signal to the memory manager for a demand page.

→ If required page is not available in frame is called demanded page.

→ Loading the page into the frame is called as

Page - In.

Page table

Page No.	Frame. No.
1	4
2	–
3	–
4	–
5	–
6	2
7	–
8	–
9	1
10	–
11	–
12	–
13	3
14	–

P2.0
P1.5
P3.1
P1.0

⇒ Process is in RAM not in Contigeous.

⇒ process is in Virtual memory in contigeous.

⇒ unloading the page from the frames is called **Page - Out.**

⇒ pl.0 is modifiable page.
pl.5 is read only page.

⇒ New page is insert the memory manager is check the old pages.

⇒ The page which can be modifiable is called **dirty Page**, must be replaced.

⇒ all the frames are filled with the pages the MMU is demand New pages then memory manager is help of medium term schedular it is moved in swap area and New page adjust in a RAM.

Command :-

```
$ getconf PAGESIZE
```

Show the size of page.

** Stack operation **

1) PUSH → Inserting data into Stack.
2) POP → retrying data from a Stack.