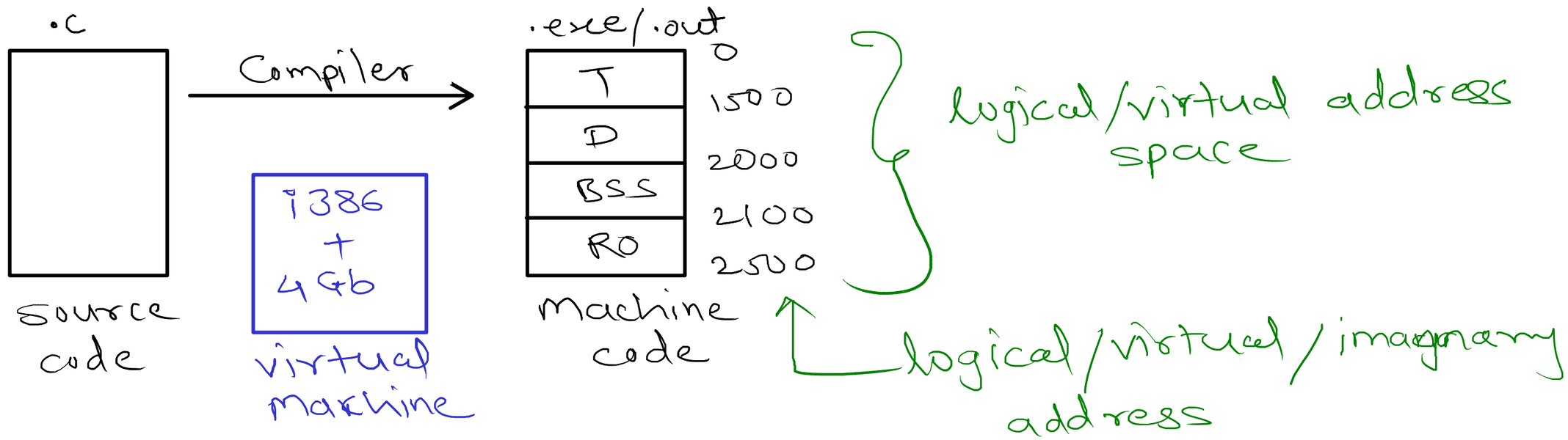
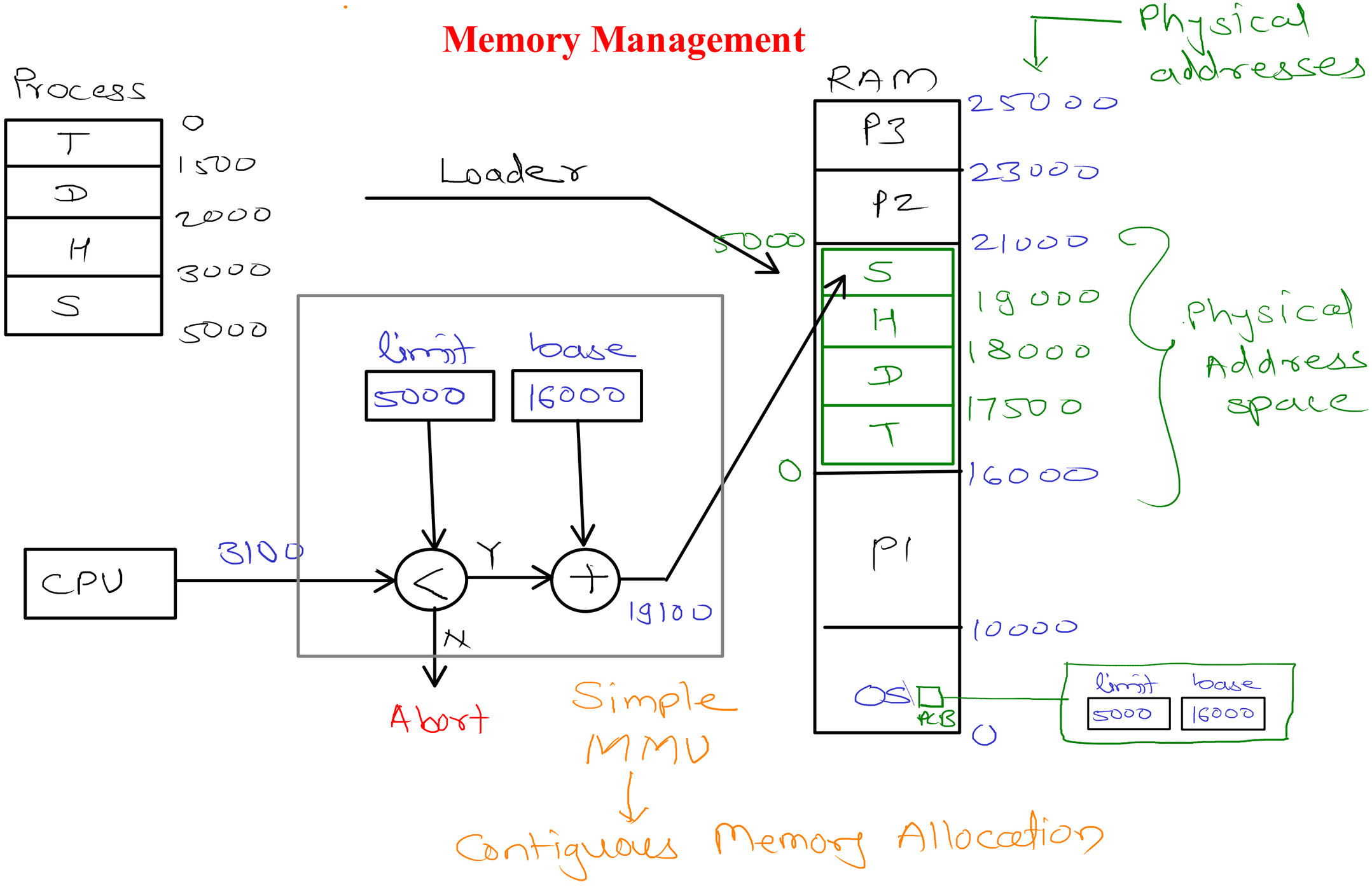
Memory Management



Memory Management



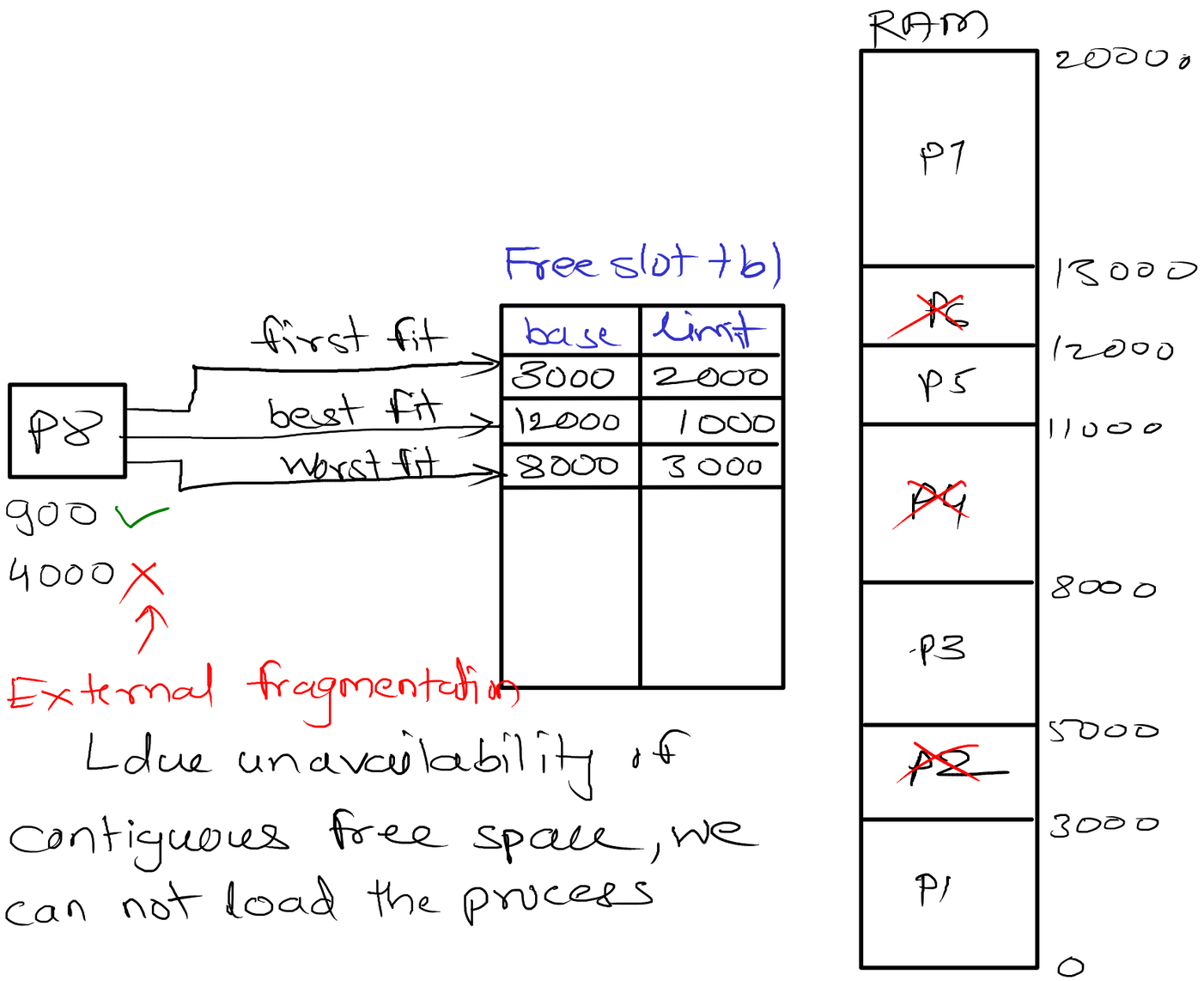
Fixed Partition

P8 → External fragmentation
- process can not be loaded into RAM due to unavailability of partition (free)

Internal Fragmentation
- process is not utilizing whole partition assigned to it, remaining space will be wasted

Limitations:
- max number of processes will be equal to number of partitions
- process size can be max size of partition

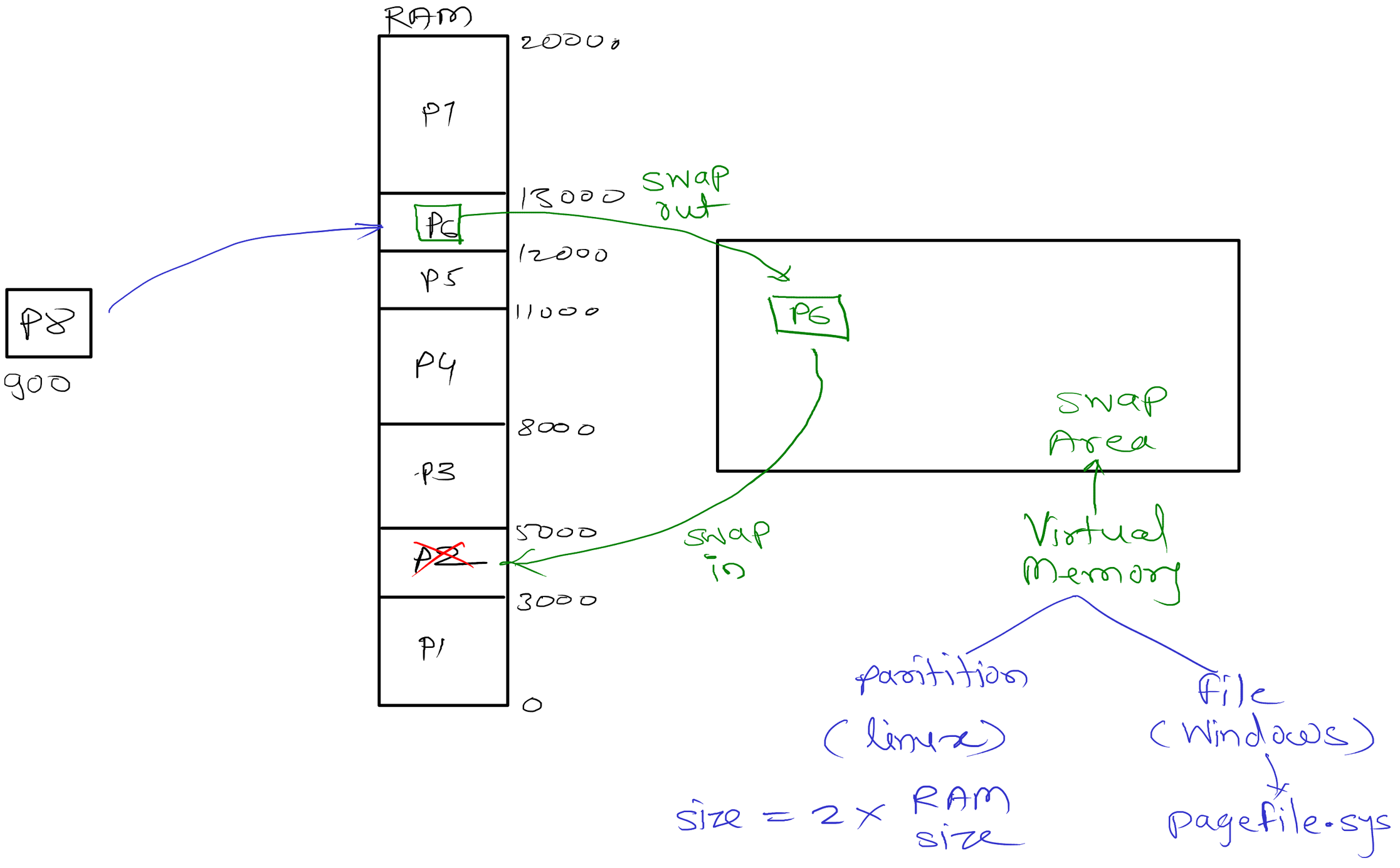
Dynamic Partition



Ldue unavailability of contiguous free space, we can not load the process

compaction :

– to create large free space processes are moved into RAM


RAM
20000
P7
13000
P6
swap out
12000
P5
11000
P4
8000
P3
5000
P2
3000
P1
0
P8
900
P6
swap Area
swap in
Virtual Memory
partition
(linux)
size = 2 x RAM size
file
(Windows)
pagefile.sys

Segmentation MMU

Demand segmentation:
— on demand segment is swapped in into RAM

Process

1	T	0
2	D	1500
3	H	2000
4	S	3000
		5000

	limit	base
1	1500	10000
2	500	17000
3	1000	14000
4	2000	20000

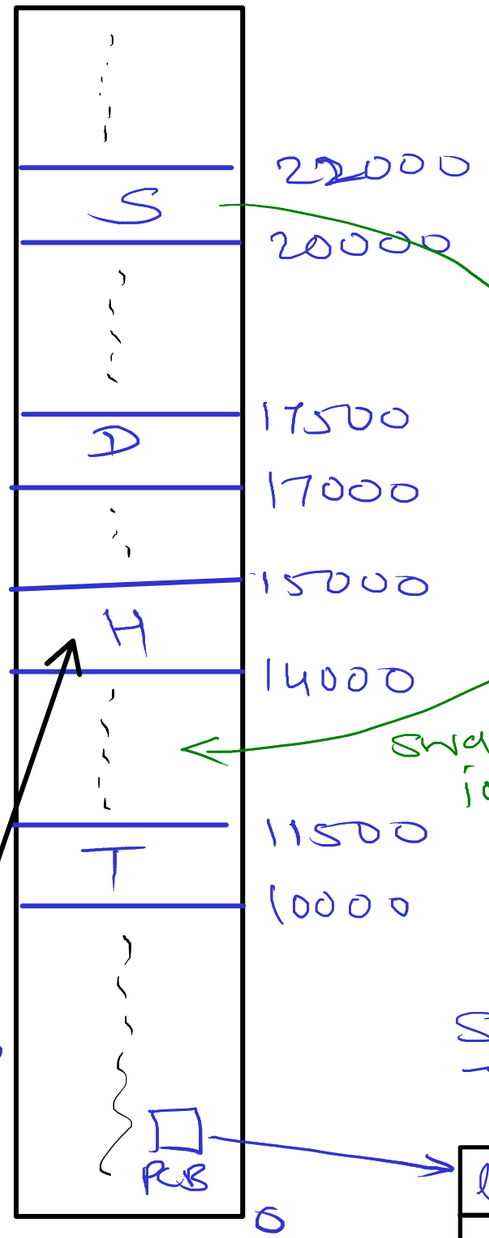
S d
3 | 900

CPU

< → Y → Abort

N

+ → 14900

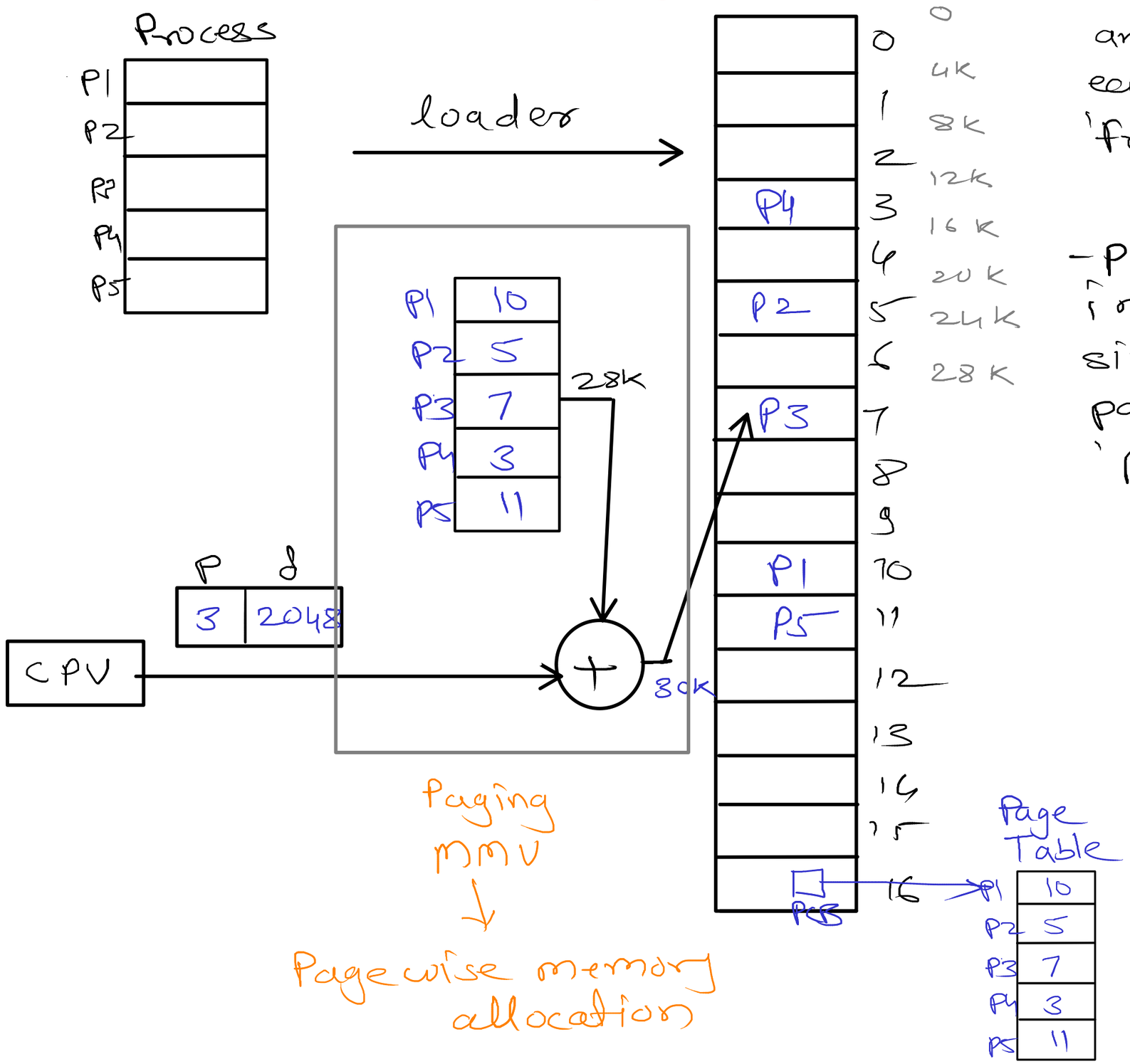


Segment Table

limit	base
1500	10000
500	17000
1000	14000
2000	20000

Segmentation MMU
↓
segmentwise memory allocation

Paging MMU



Paging MMU

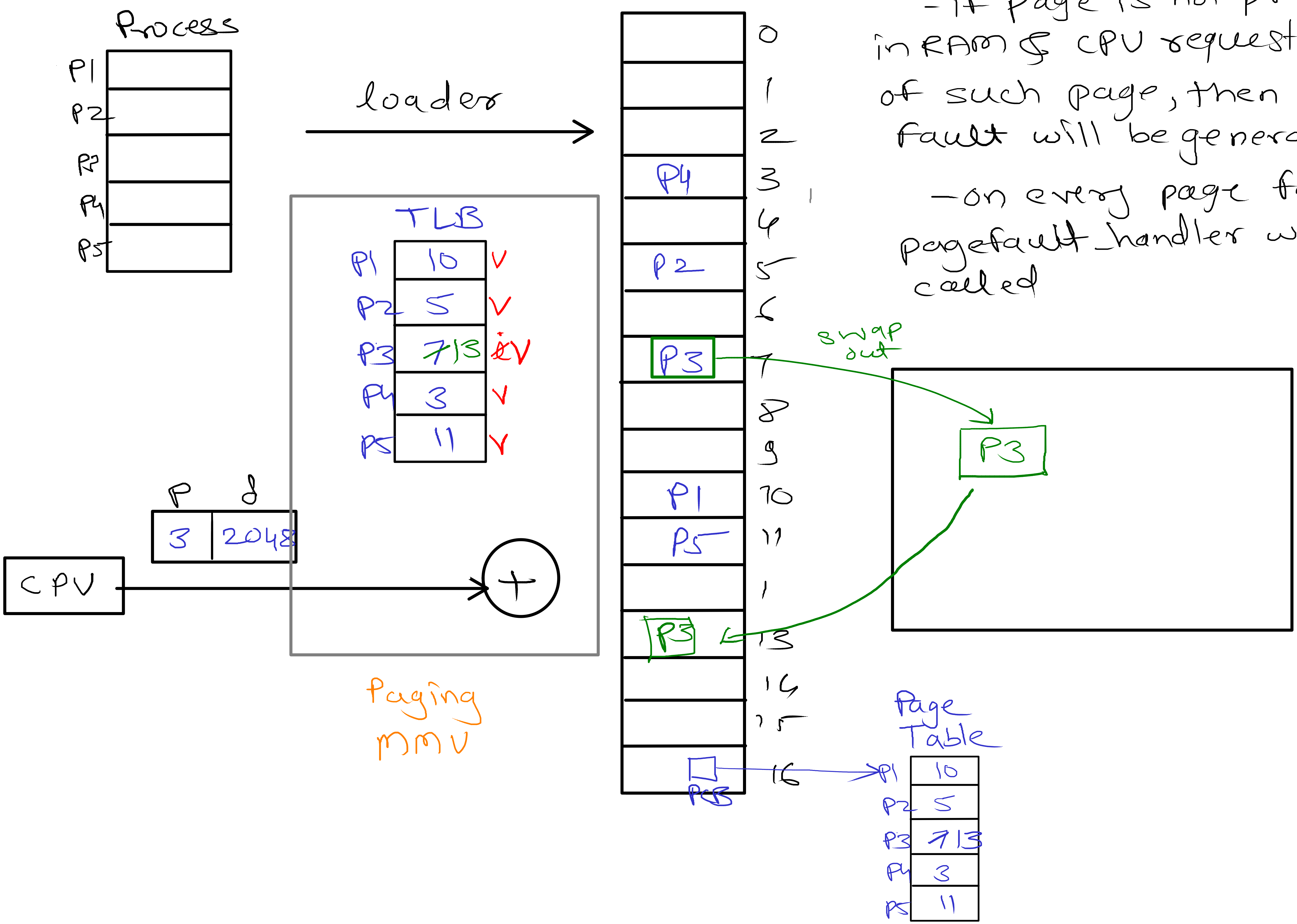
↓

Pagewise memory allocation

- RAM is divided into fixed and equal size partitions, each partition is known 'frame' / 'physical page'
  frame size = 4 kb
- process is also divided into partition equal to size of frame, each partition is known as 'page' / 'logical page'

Page fault

- if page is not present in RAM & CPU request addr of such page, then page fault will be generated.
- on every page fault, pagefault_handler will be called



```
pagefault_handler()
{
```

i) check request addr is valid or not
   if not valid — terminate process

ii) check whether you have read/write promission
   if no permissions — terminate process

iii) find free frame to swap in your page into RAM.

iv) swap in the page into RAM

v) Update page table entry

vi) re execute the instruction for which page fault was occured.

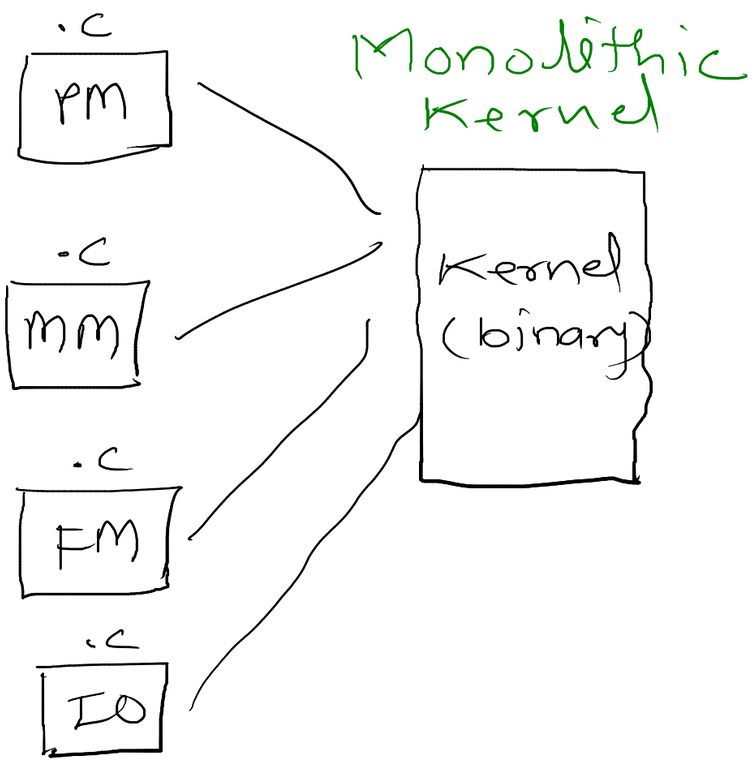
```
}
```

```
int main(void)
{
    char *ptr = "sunbeam";
    cout >> ptr;
    *ptr = 'F';  ---> error
    cout >> ptr;
    return 0;
}
```

```
char ptr[] = "sunbeam";
```

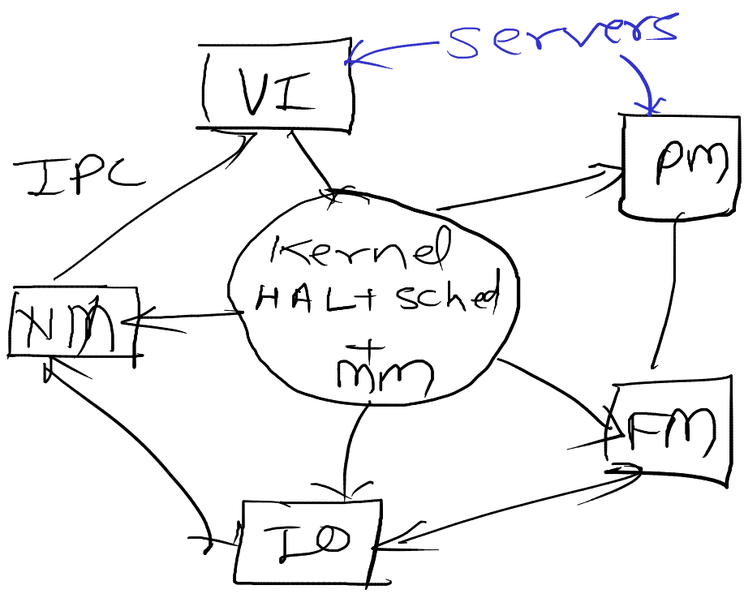
Types of Kernel

Monolithic Kernel



BSD UNIX

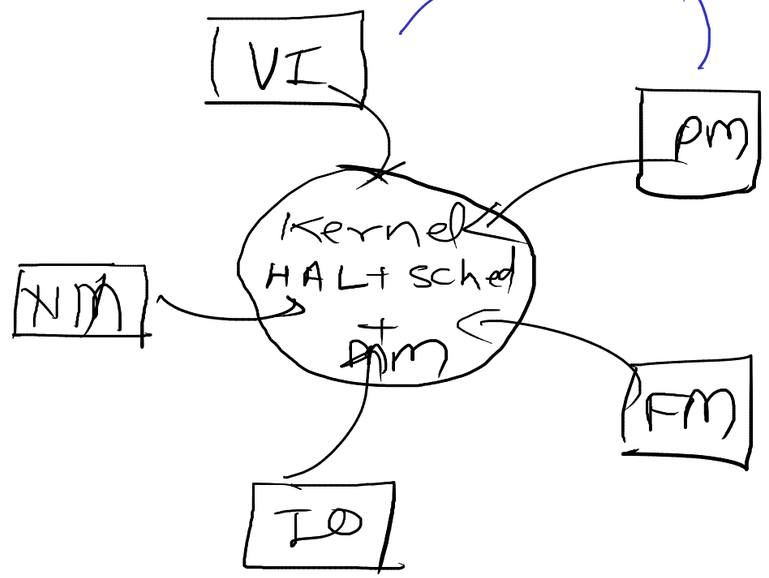
Micro Kernel



Symbian

Modular Kernel

dynamically loadable modules



Linux
Windows

Hybrid Kernel

BSD UNIX + MACH

Darwin

Nano Kernel

by RTOSes

Linux Kernel = Static component + Dynamic Component

1) Process management
2) CPU scheduling
3) Memory Management
4) IO sub system
5) Hardware Abstraction
6) system calls
:

/boot/ vmlinuz

Kernel.org

1) File system mgrs
2) Device drivers

Dynamically loadable moduks
(Kernel Objects)
(.KO)

/lib/modules/___

chmod +x demo01.sh

r - read
w - write
x - execute

levels
user/owner (u)
group (g)
others (o)

rw- rw- r--

chmod u+x demo01.sh
g-w

rw- rw- r--

110 110 100

6 6 4

chmod 664 demo01.sh

111 100 100

7 4 4

chmod 744 demo1.sh