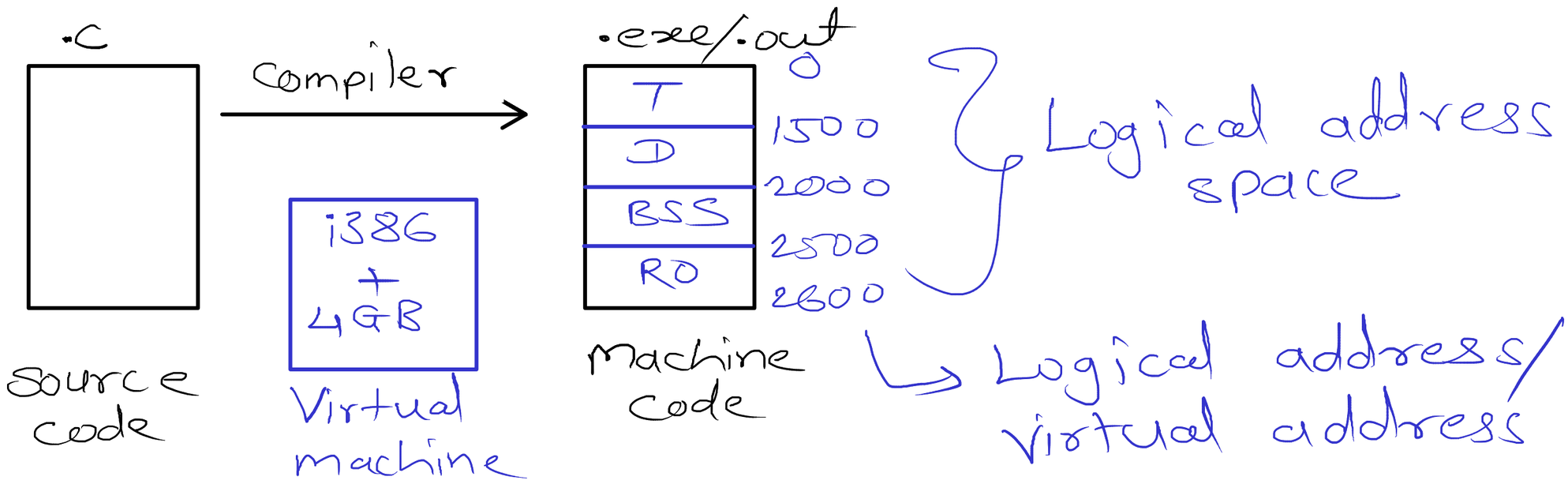
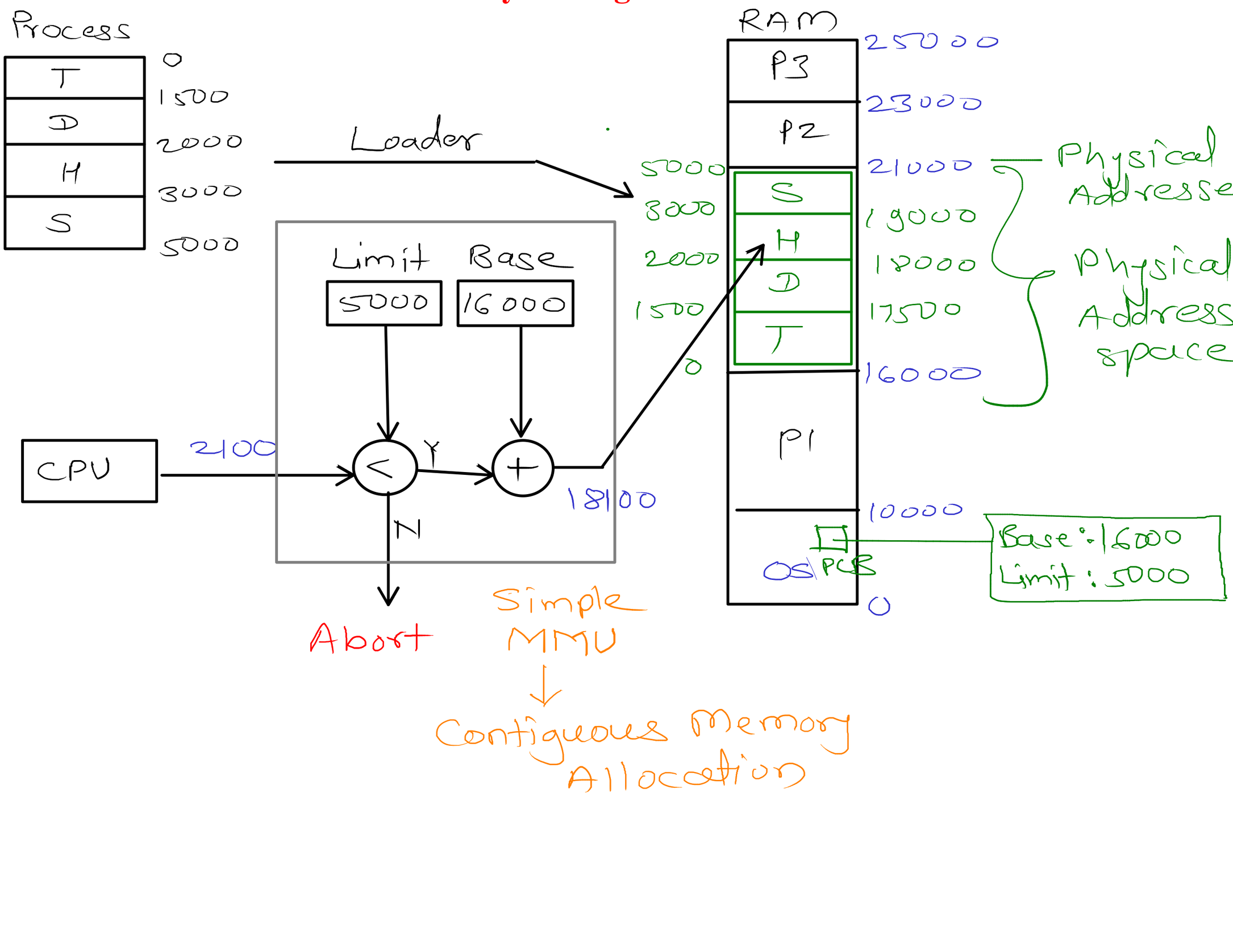
# Memory Management



# Memory Management



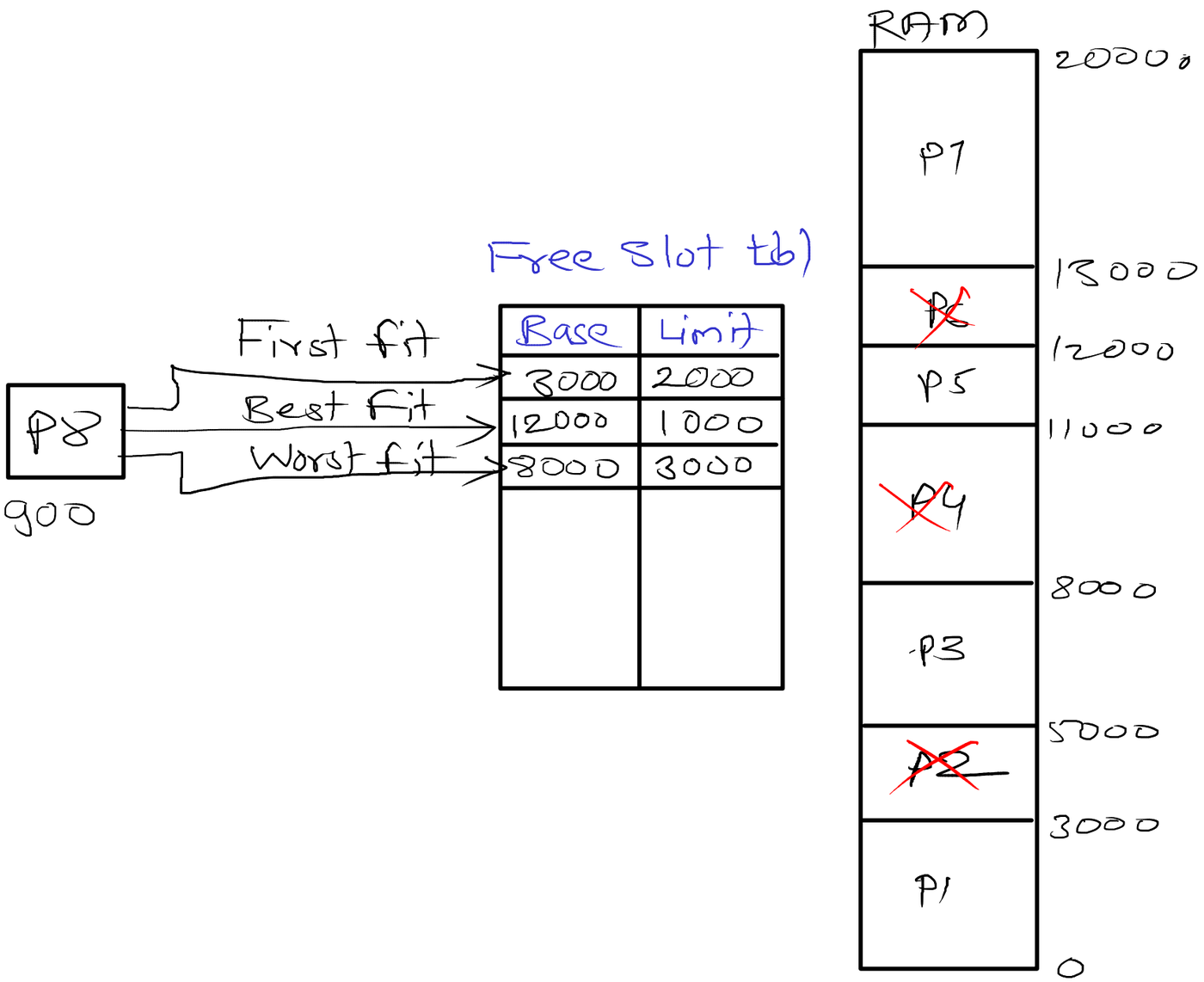
# Fixed Partition

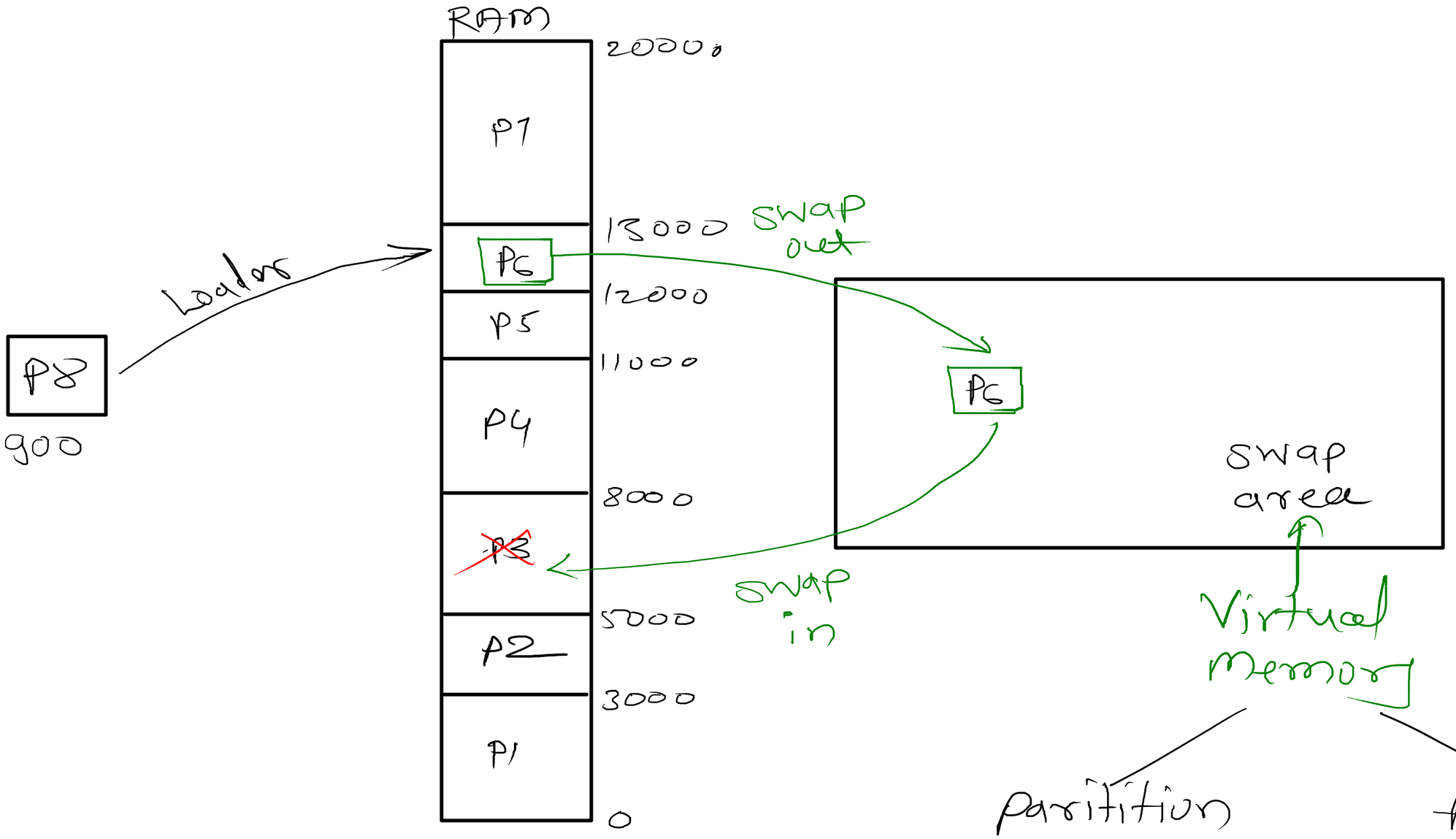

P8 → External fragmentation
- if free space is not available to load new process, then we can not load new process into memory.

1Kb → Internal fragmentation
- if complete partition is not utilized by your process, then remaining partition is wasted.

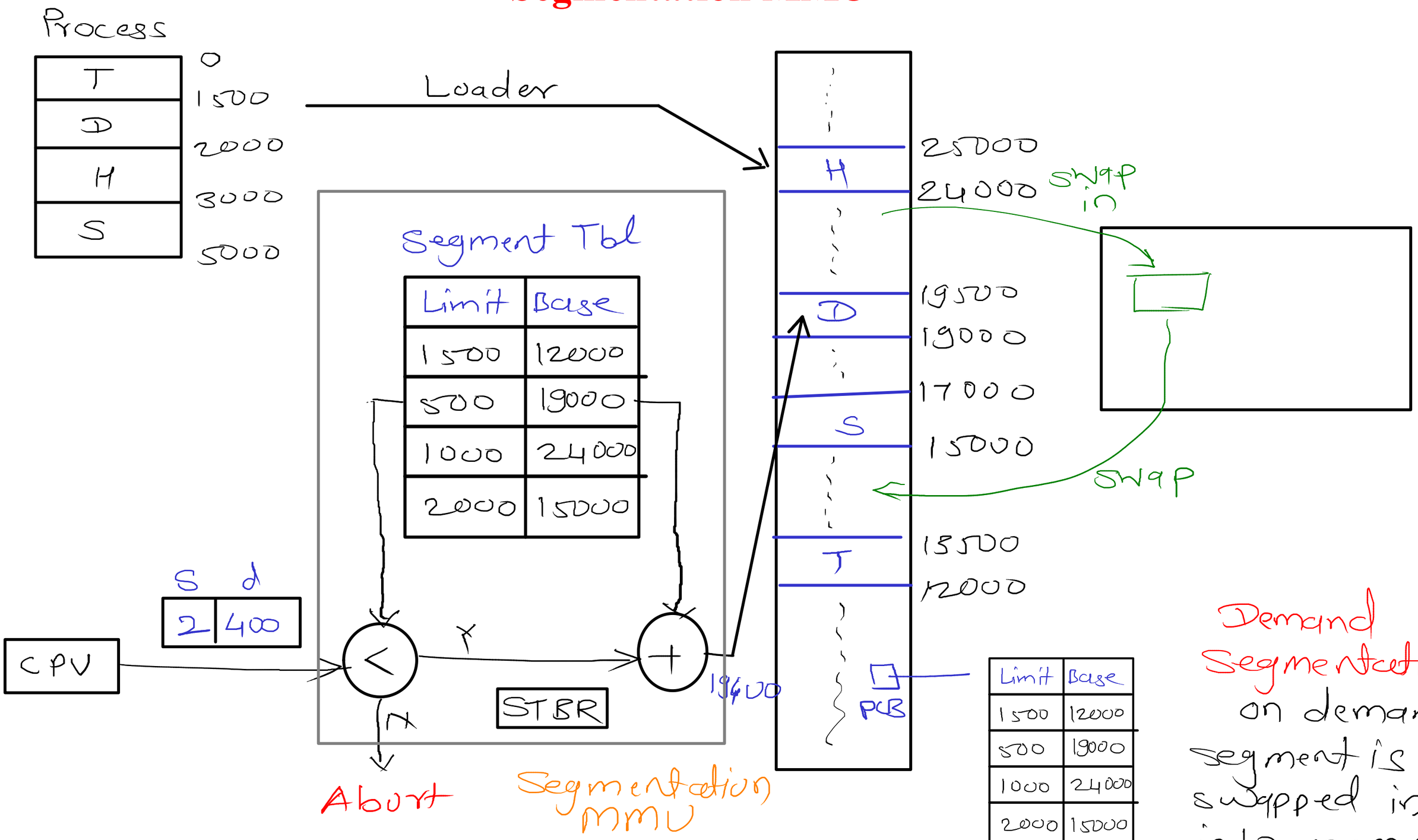
Limitation :
- max process size can be max partition size
- no. of processes are equal to no. of partitions.

# Dynamic Partition




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

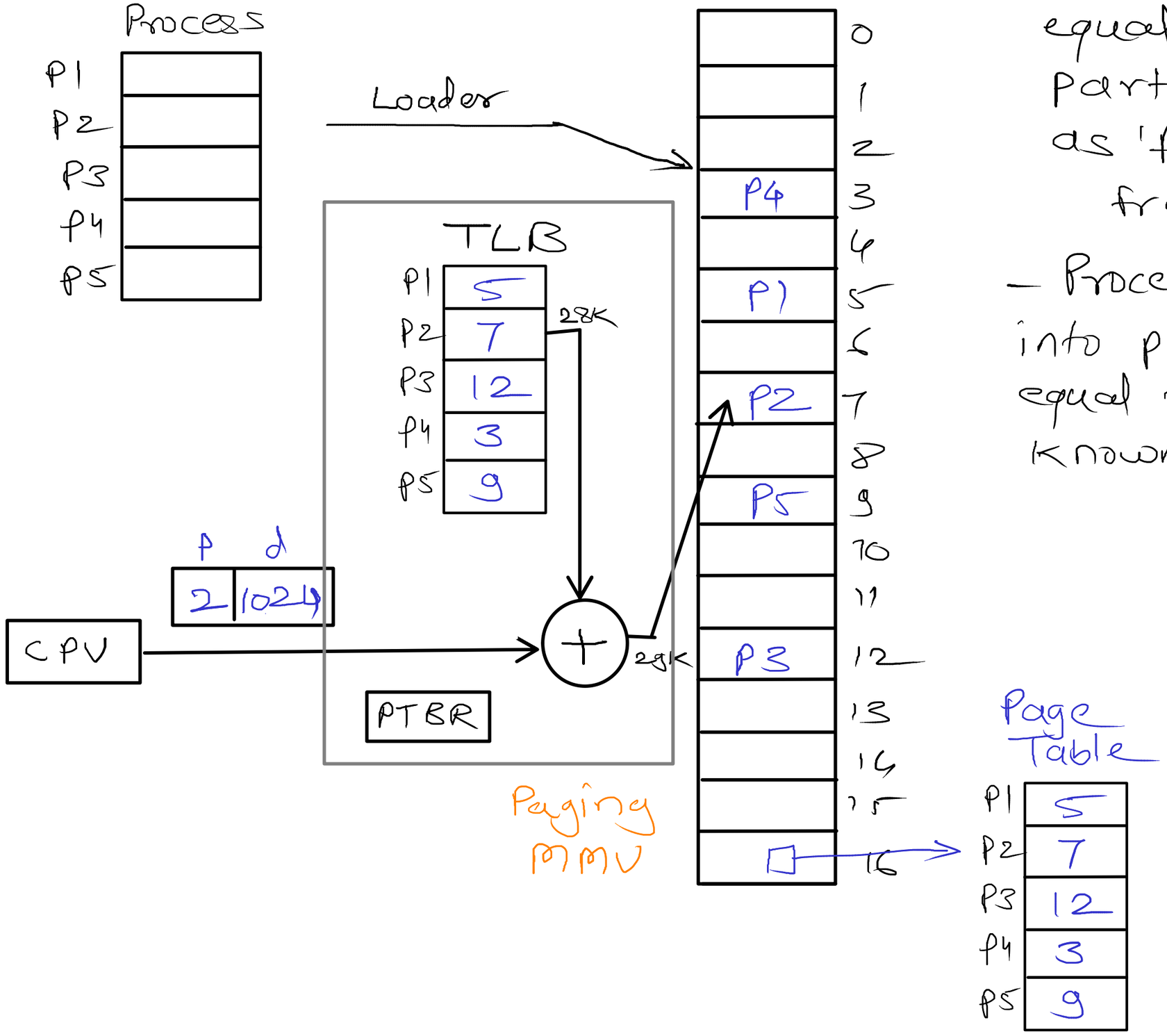
# Segmentation MMU




Demand Segmentation

on demand segment is swapped in into memory

# Paging MMU




- RAM is divided into equal & fixed size partitions, it is known as 'frame'/physical page
  frame size = 4 kb
- Process is also divided into partitions of size equal to frame, if is known 'page'/logical page

Process

P1
P2
P3
P4
P5

Loader

TLB

| | | |
|---|---|---|
| P1 | 5 | V |
| P2 | 7 | V |
| P3 | 12 | V |
| P4 | ~~3~~ 8 | ~~i~~ V |
| P5 | 9 | V |

p d

4 | 1024

CPU

PTBR

Paging MMU

0 1 2 P4 3 4 P1 5 6 P2 7 8 P5 9 10 11 P3 12 13 14 15 16

swap out

swap in

P4

Page Table

| | |
|---|---|
| P1 | 5 |
| P2 | 7 |
| P3 | 12 |
| P4 | ~~3~~ 8 |
| P5 | 9 |

## Page fault :

- will be generated when CPU will demand for address of invalid page
- when page fault will occur, page fault handler of OS will get call.

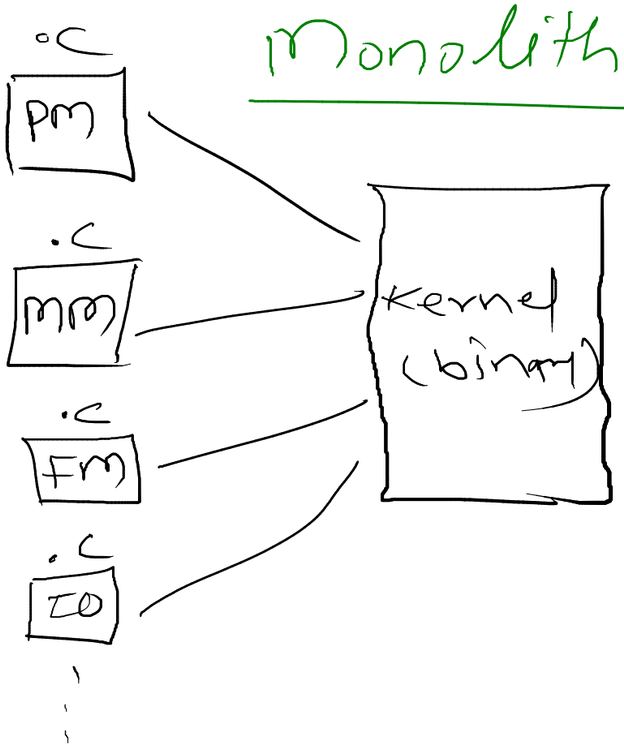
```
page_fault_handler( )
{
    1> check whether addr is valid/not
        if not valid — abort
    2> check whether you have read/write access
        if no permission — abort
    3> find free frame to swap in requested
       address page and swap in the same.
    4> Update logical & physical page mapping in
       TLB and page tbl.
    5> re execute the instruction for which
       page fault occured.
}
```

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

char ptr[] = "sunbeam";

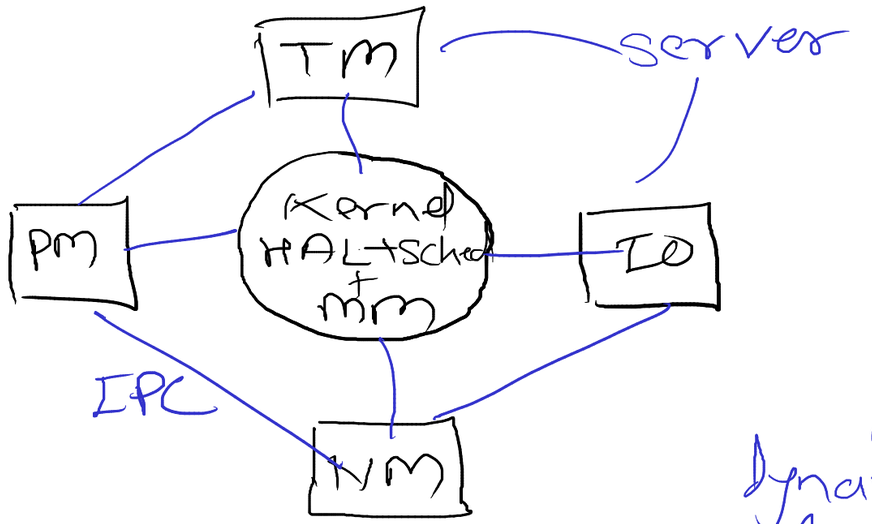
# Types of Kernel

## Monolithic



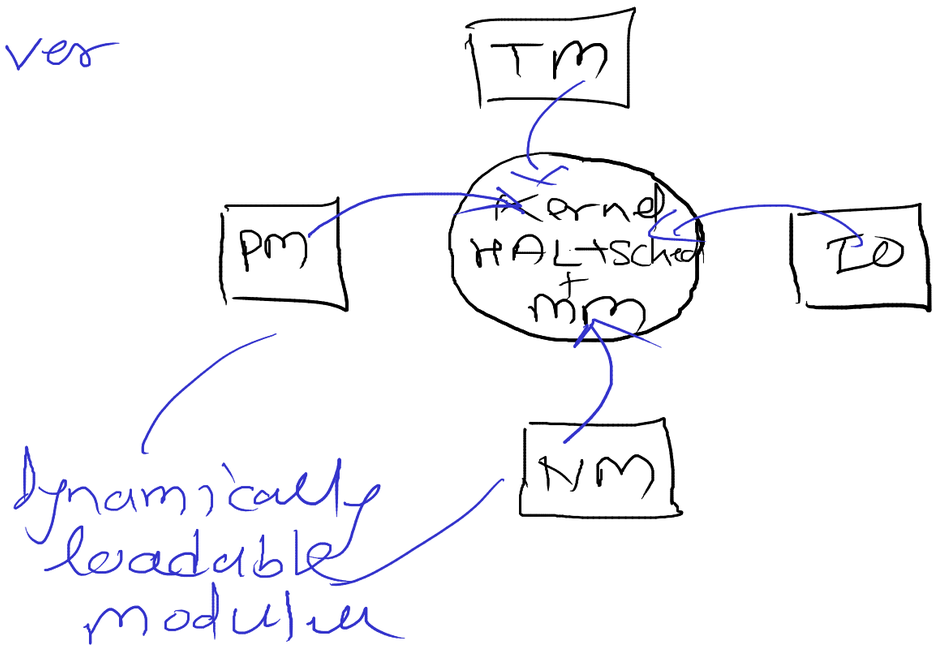

BSD UNIX

## Micro




Symbian

## Modular kernel




Linux
Windows

## Hybrid kernel

MAC → BSD UNIX + MACH

Darwin

Linux Kernel = Static + Dynamic

1) PM
2) MM
3) CPU sched
4) IO subsystem
5) HAL
6) system calls

/boot/vmlinuz

kernel.org

1) File system mgr
2) Device driver

Dynamicaly loadable Module (Kernel Objects)

/lib/modules/____/....

r — read
w — write
x — execute

user = rw— = 110 = 6
group = rw— = 110 = 6
others = r— — = 100 = 4
= 664

111 = 7
110 = 6
000 = 0
760

chmod +x file.txt
-x
+w
-w
+r
-r

chmod u+x file.txt
u-x

u - user
g - group
o - others

chmod 760 file.txt