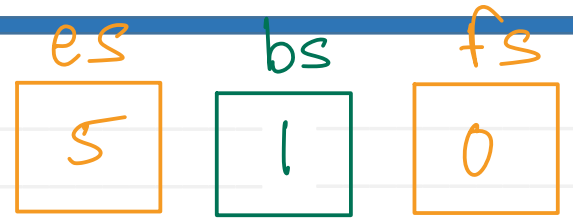

# Sunbeam Institute of Information Technology
# Pune and Karad

## Module - Concepts of Operating System

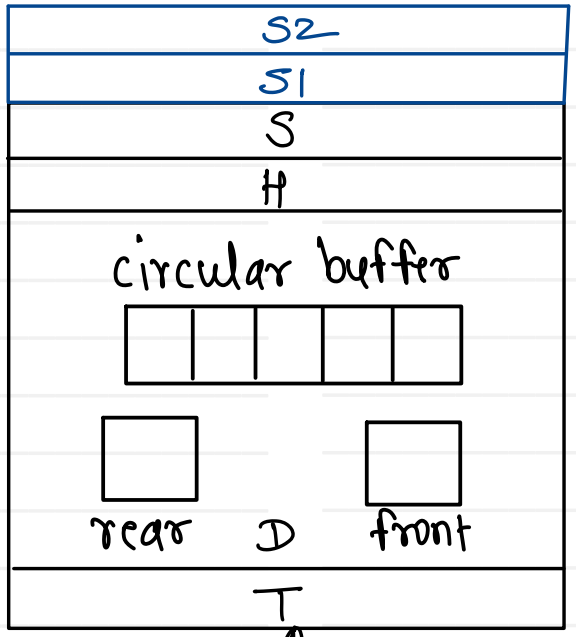
Trainer - Devendra Dhande

Email – devendra.dhande@sunbeaminfo.com

# Producer - Consumer




```
thread1_func( ) {
    while(1) {
        P(es)
        P(bs)
        V(bs)
        V(fs)
    }
}
```

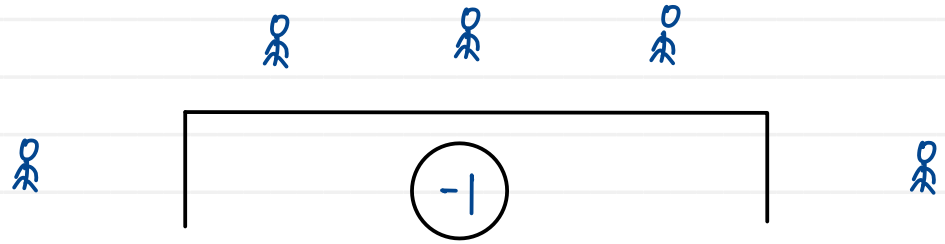



```
thread2_func( ) {
    while(1) {
        P(fs)
        P(bs)
        V(bs)
        V(es)
    }
}
```

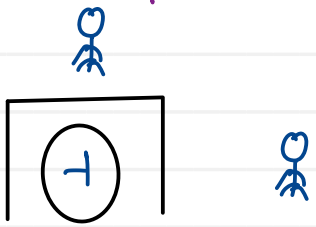
# Semaphore

- semaphore is internally a counter
- Operations :
  1. Dec / wait / P() :
     a. dec count
     b. if count < 0, then block the current process
  2. Inc / post / V() :
     a. inc count
     b. if someone is blocked on this semaphore, wake up one

1> Counting Semaphore




2. Binary Semaphore



# Mutex

Mutex = Mutual Exclusion

↳ one at a time

- lock/unlock operations are performed on mutex
- process who locks the mutex becomes owner of the mutex
- only owner can unlock the mutex

# Deadlock

- infinite waiting for a resource
- deadlock occurs only when below four conditions hold true at a time
  1. Mutual Exclusion
  2. No preemption
  3. Hold & wait
  4. Circular wait


**Prevention :**

While implementing OS, it is always ensured that 1/4 condition will hold false.

**Avoidance :**

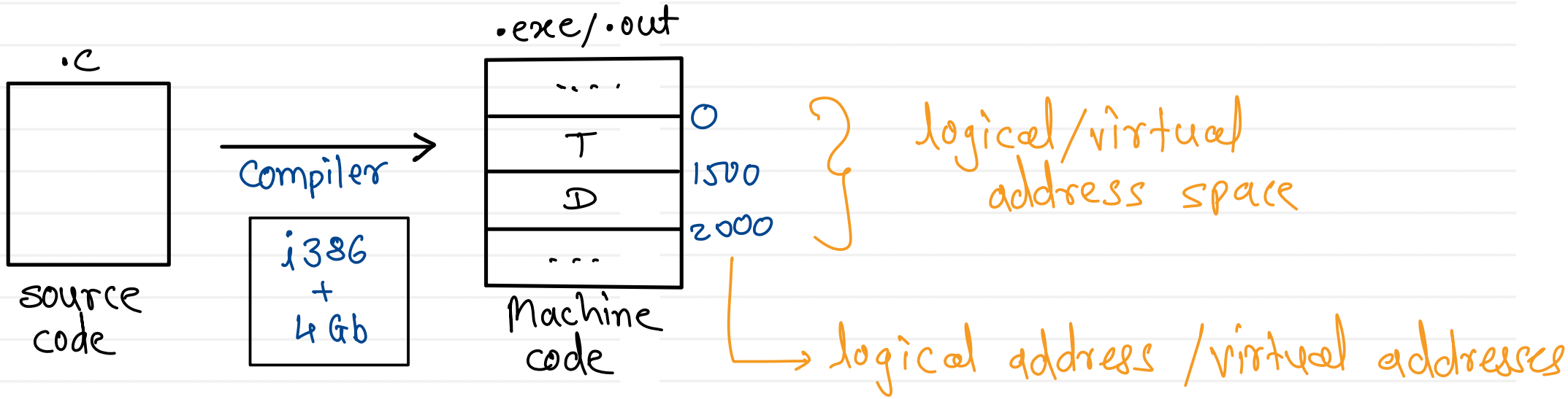
1. Banker's algorithm
2. Resourse allocation graph
3. Safe state algorithm

**Recover :**

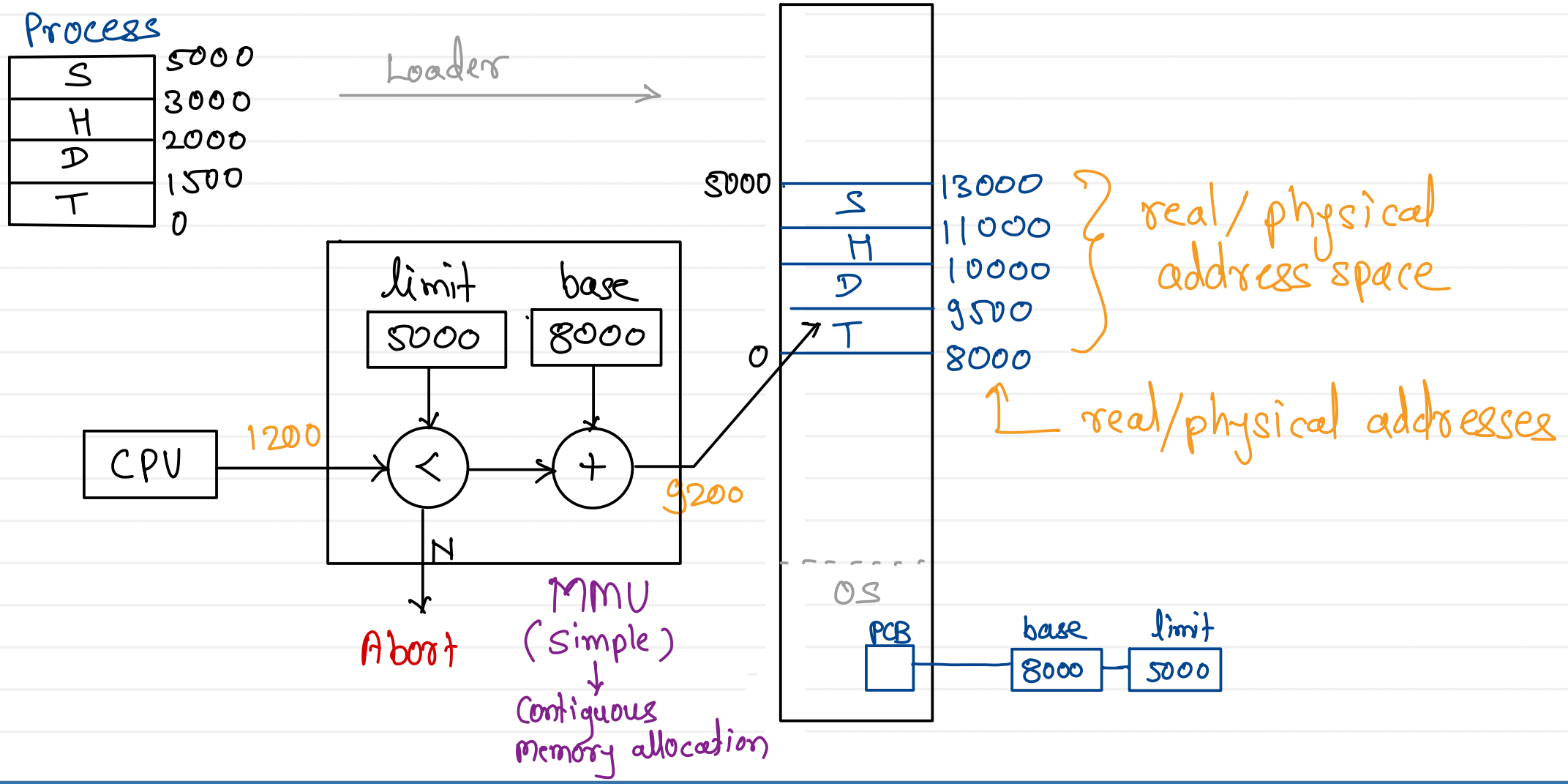
1. resource preemption
2. forceful termination of process

# Memory Management

- compiler always assigns logical / virtual / imaginary addresses to the functions are variables

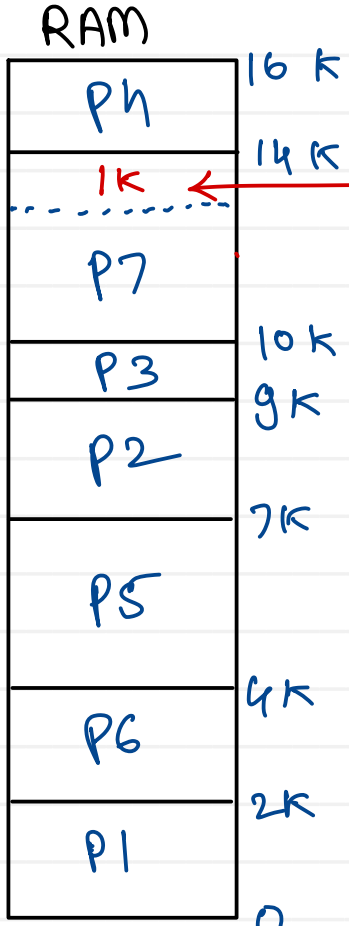


# Simple MMU


Process
S
H
D
T
5000
3000
2000
1500
0
Loader
5000
13000
11000
10000
9500
8000
real/physical address space
real/physical addresses
limit
base
5000
8000
CPU
1200
<
+
9200
N
Abort
MMU
(simple)
Contiguous memory allocation
0
OS
PCB
base
limit
8000
5000

# Contiguous memory allocation

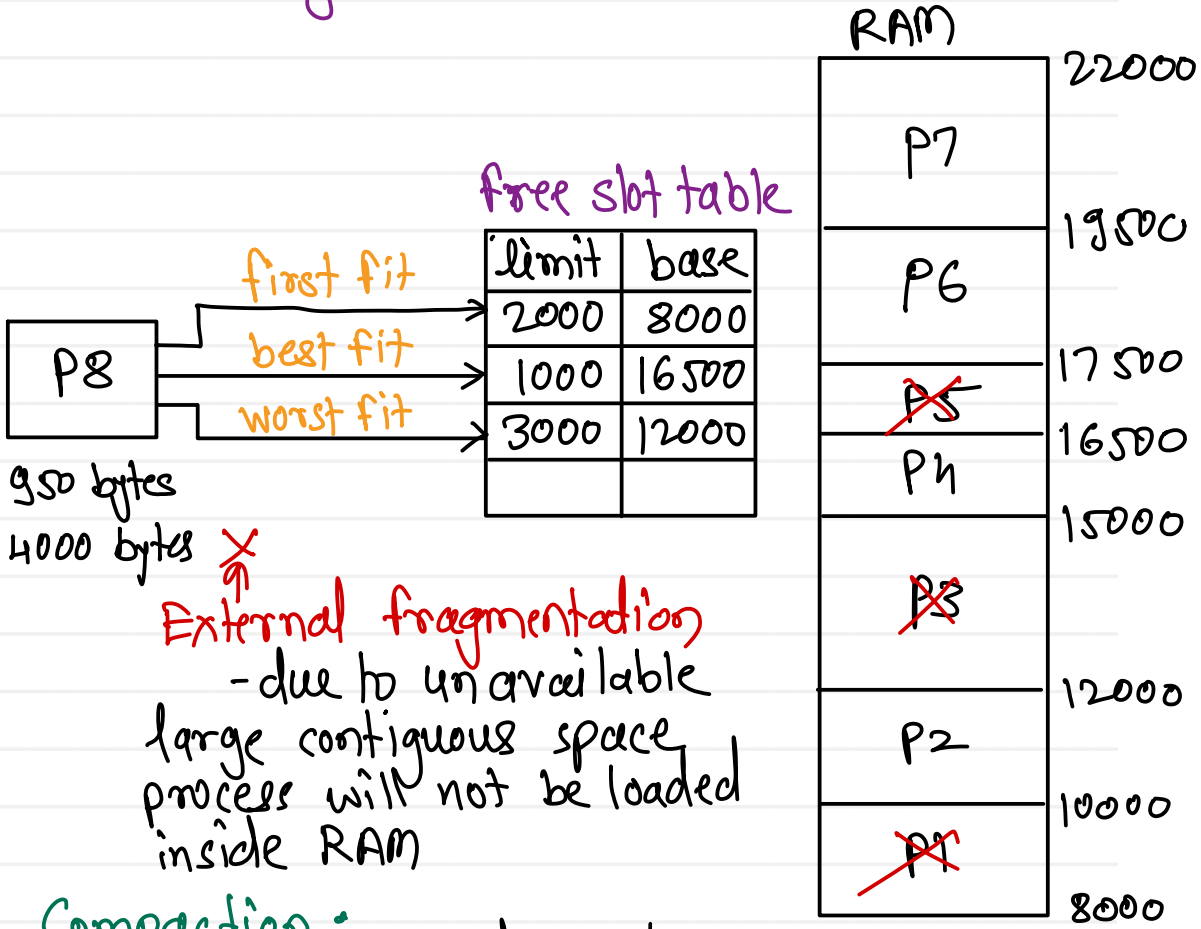
## Fixed partition




internal fragmentation:
process is not utilizing total space allocated to it. this leads to memory wastage

limitations :
1. No. of processes will be limited to no. of partitions
2. Max size of processes will be also limited to max size of partition.

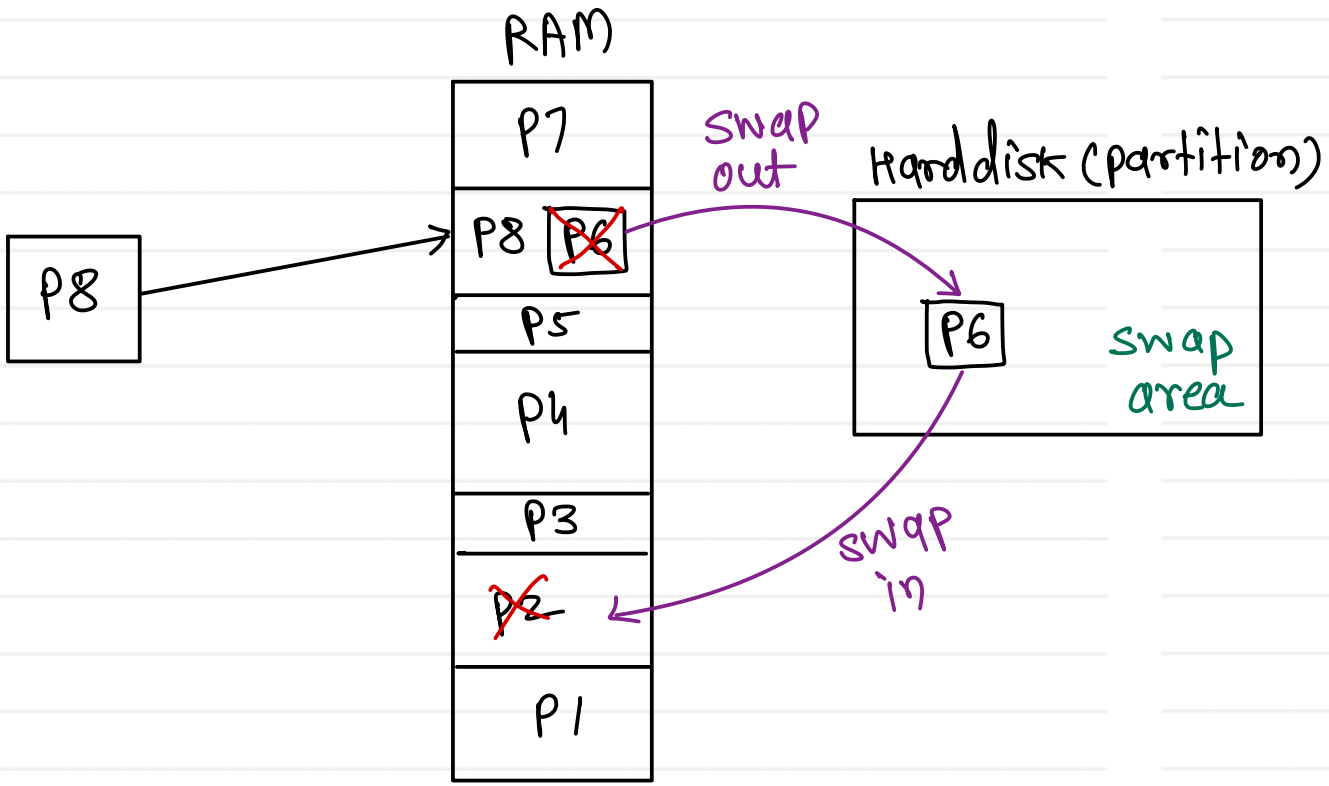
## Dynamic Partition




External fragmentation
- due to unavailable large contiguous space process will not be loaded inside RAM

Compaction :
processes are moved inside RAM to create large contiguous free space

# Virtual memory




swap area

partition → Linux

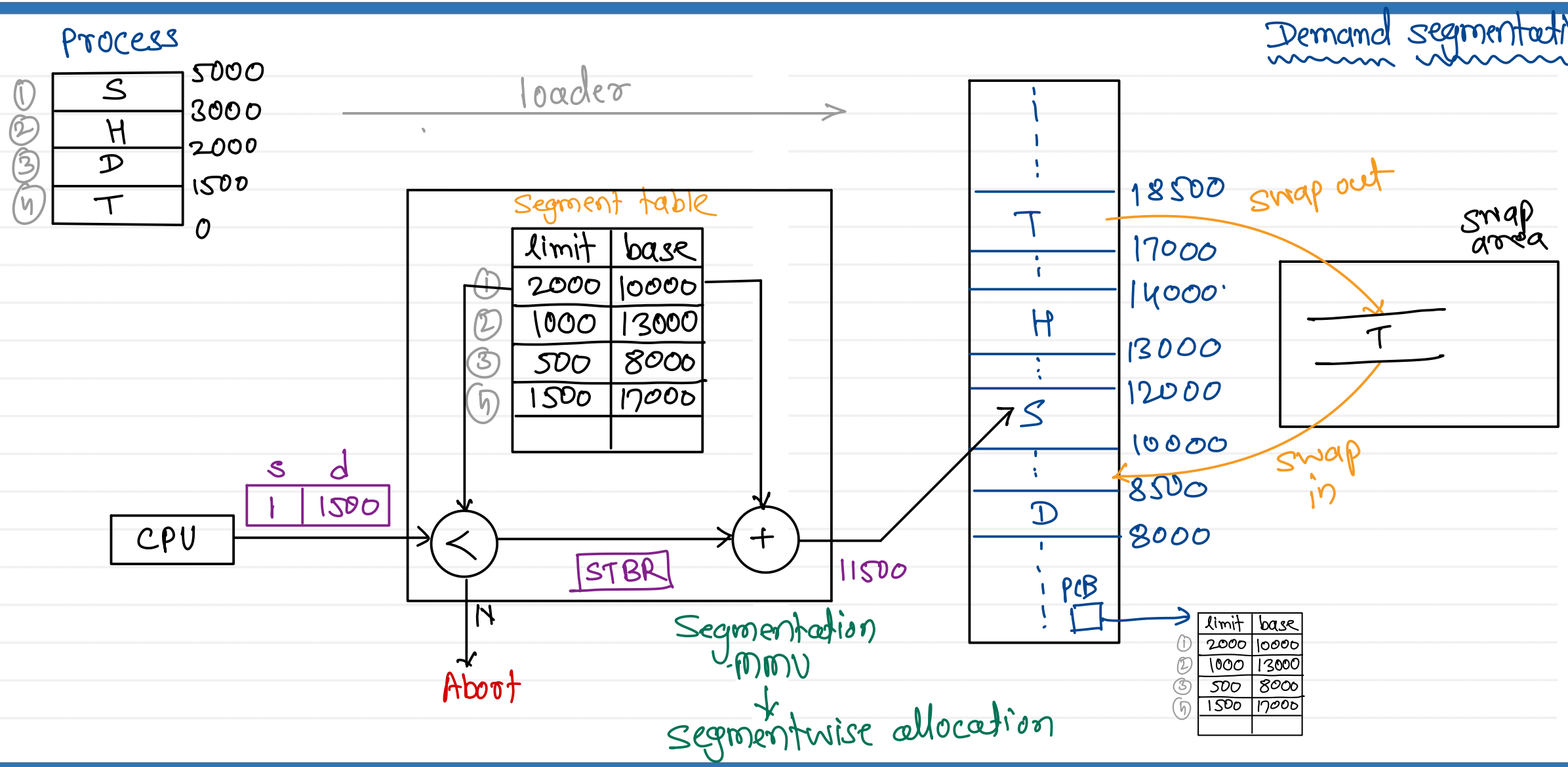
file → Windows (pagefile.sys)

size = 2 * RAM size

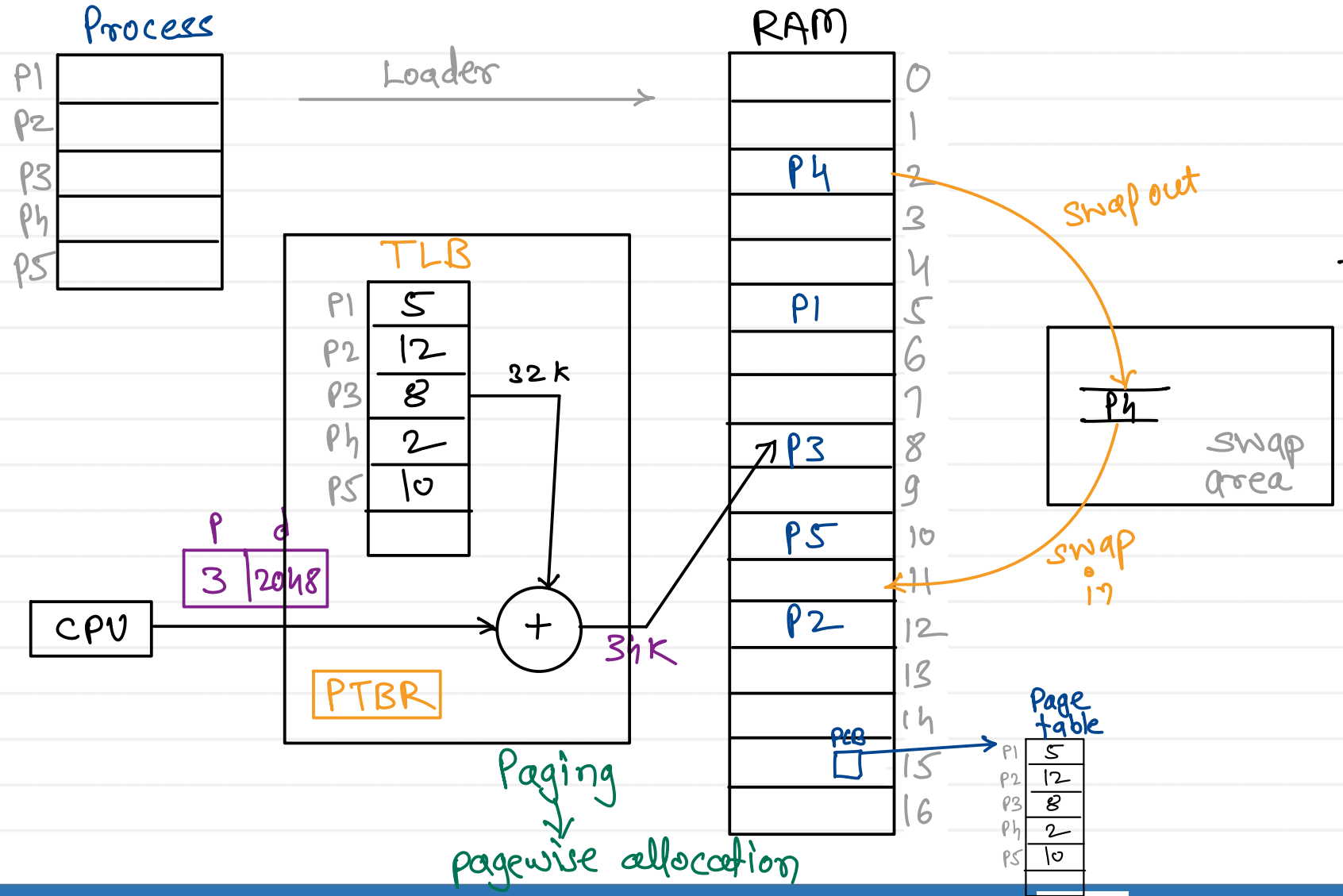
virtual memory

- extension to RAM (memory)
- this memory is formatted like RAM
- it is like RAM but not RAM thats why it is called as virtual memory.
- processes are only kept into this area (inactive)
- processes never executes inside virtual memory.

# Segmentation MMU



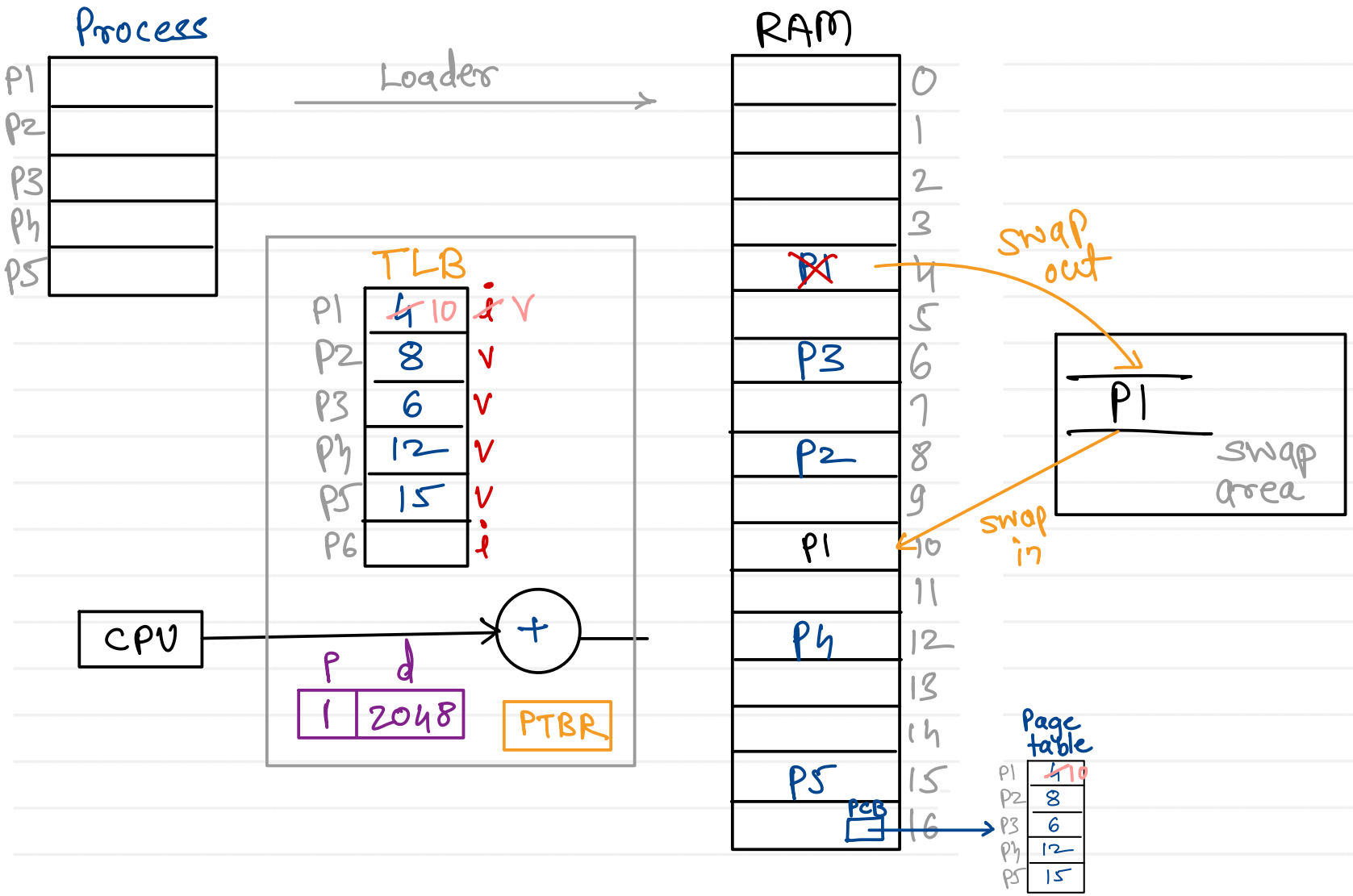
# Paging MMU




- RAM is divided into fixed equal size partitions
  size = 4kb (4096 bytes)
  "frame" / "physical page"
- Process is also divided into partitions of size equal to frame size.
  "page" / "logical page"

Demand paging :
- on demand page is swapped in.

# Page fault




**Page fault :**

Whenever CPU request for the address of some invalid entry of page table, this fault is generated.

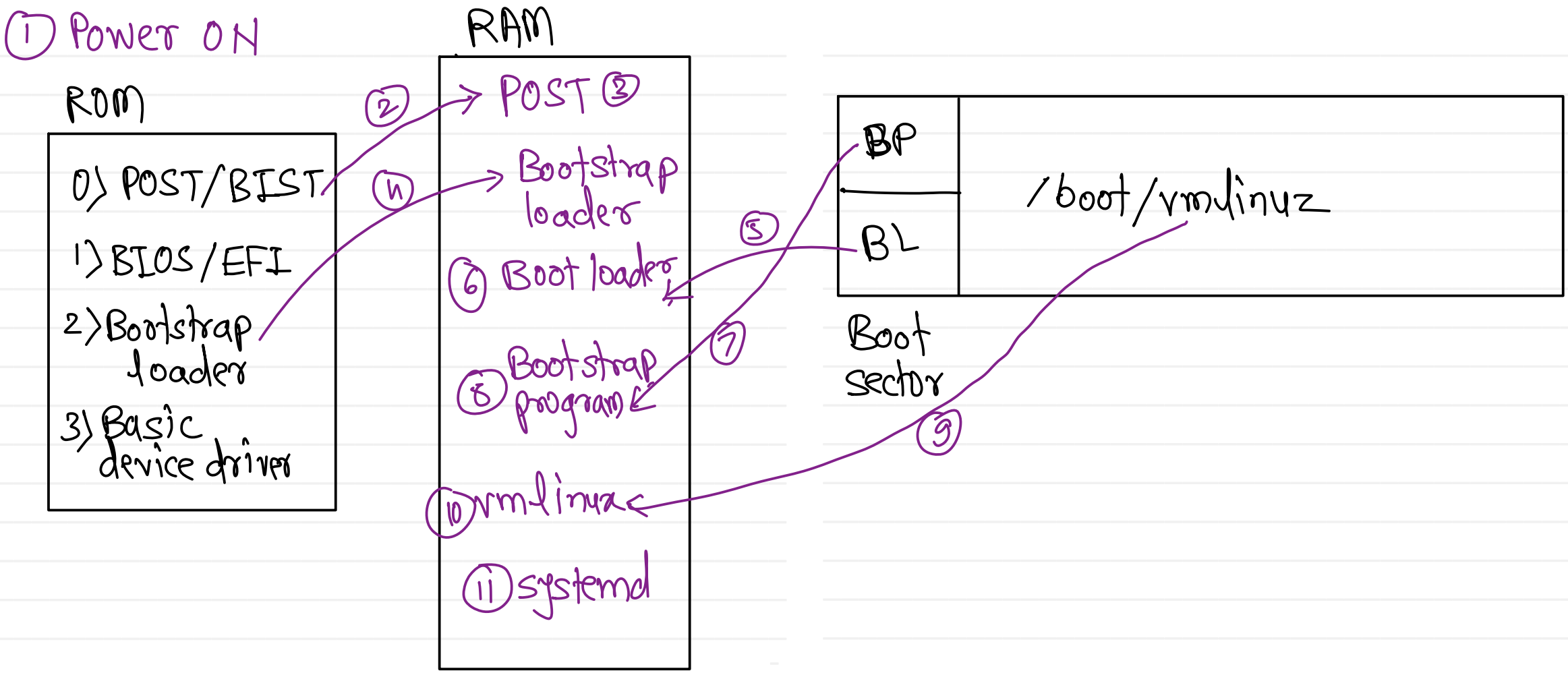
on every page fault, page fault handler of OS is called

```
pagefault_handler( ) {
  1. validate the address
  2. check for read/write perm
  3. find free frame in RAM
  4. swap in page into free frame
  5. update mapping in page table and TLB.
  6. re execute the command for which page was occured
}
```

**Thrashing :** frequent swap in & swap out of pages

**Solution :** increase the size

# Booting


1) Power ON
ROM
0) POST/BIST
1) BIOS/EFI
2) Bootstrap loader
3) Basic device driver
RAM
POST
Bootstrap loader
Boot loader
Bootstrap program
vmlinuz
systemd
BP
BL
/boot/vmlinuz
Boot sector
2
3
4
5
6
7
8
9
10
11


# Thank you!!!

Devendra Dhande

devendra.dhande@sunbeaminfo.com