Deadlock :

- indefinite waiting for resource is called as deadlock.
- if all four conditions hold true at same time deadlock will occur

1) Mutual Exclusion
2) No preemption
3) Hold & wait
4) Circular wait.

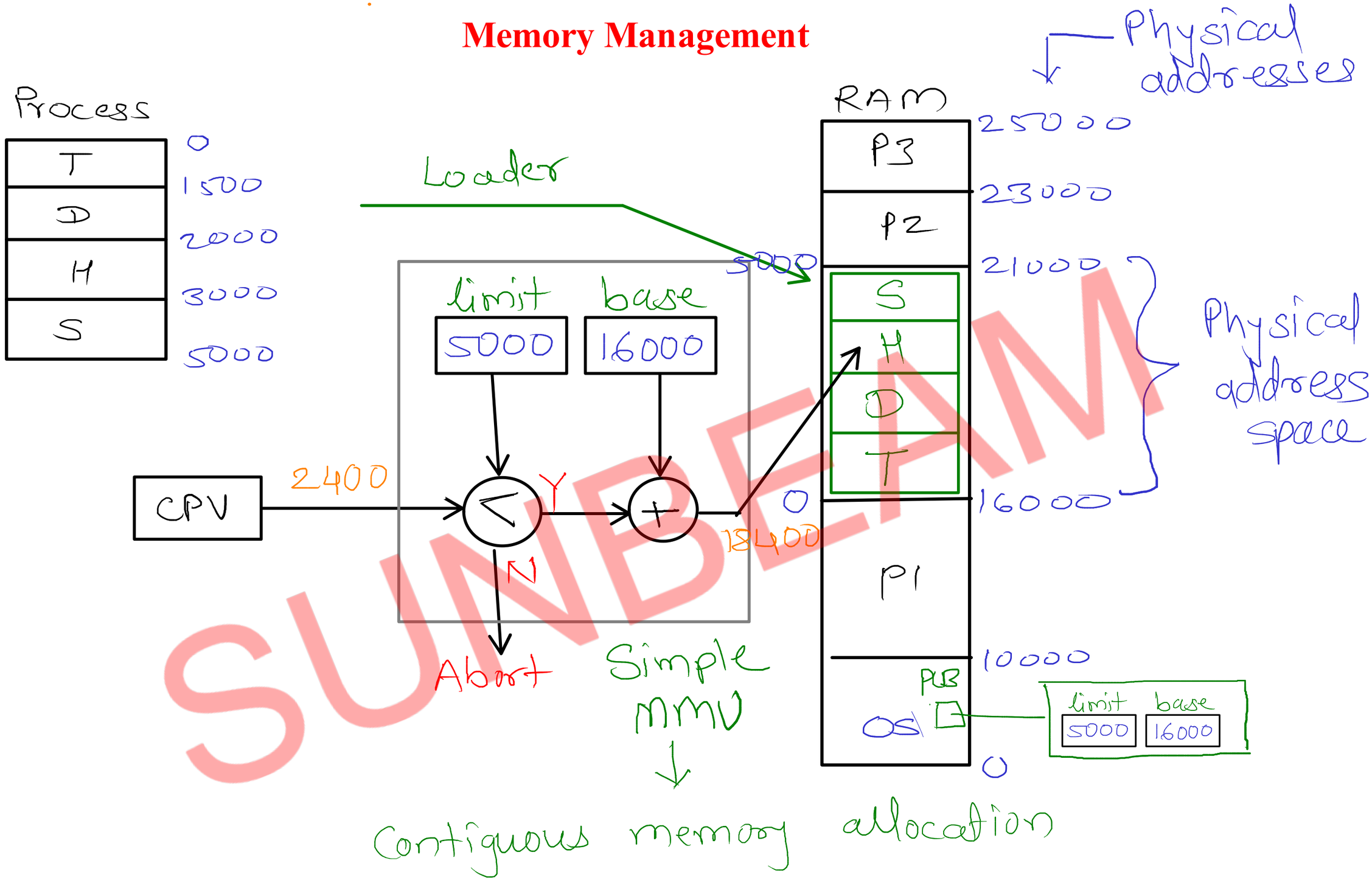
Prevention :
while writing code of OS we alway ensure that one condition out of four will hold false all the time

Avoidance
1) Banker's algorithm
2) Resource allocation graph
3) Safe state algorithm

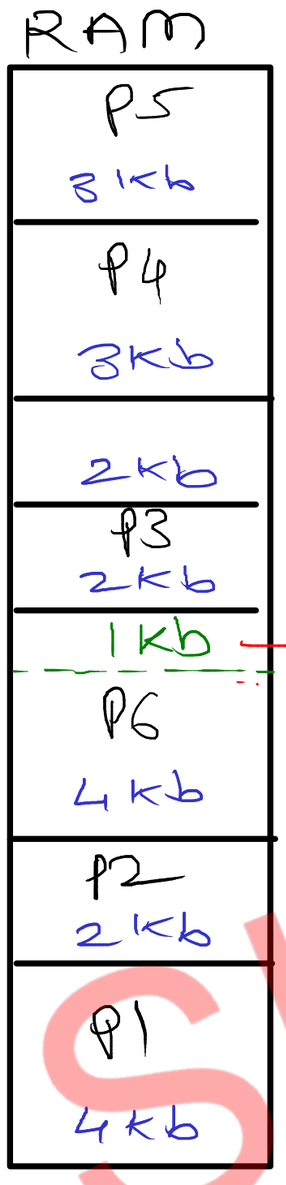
Recover
1) Resource preemption
2) Forceful termination

Memory Management

Memory Management



Fixed Partition



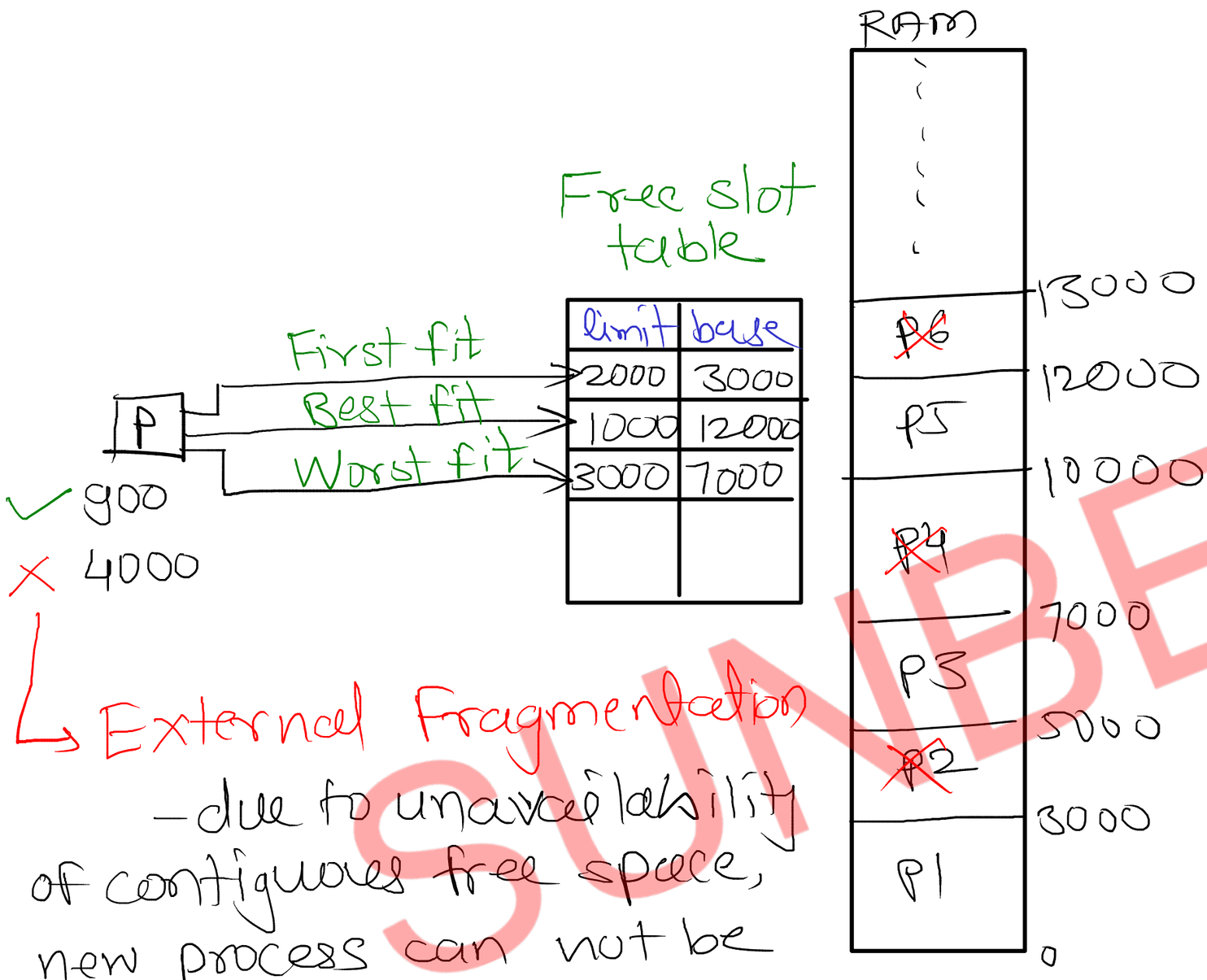
limitations:

1) Max size of process = max size of partitions
2) No. of processes = no. of partitions

Internal Fragmentation

- if process is not utilizing whole allocated space then, memory space is wasted.

Dynamic Partition



↳ External Fragmentation

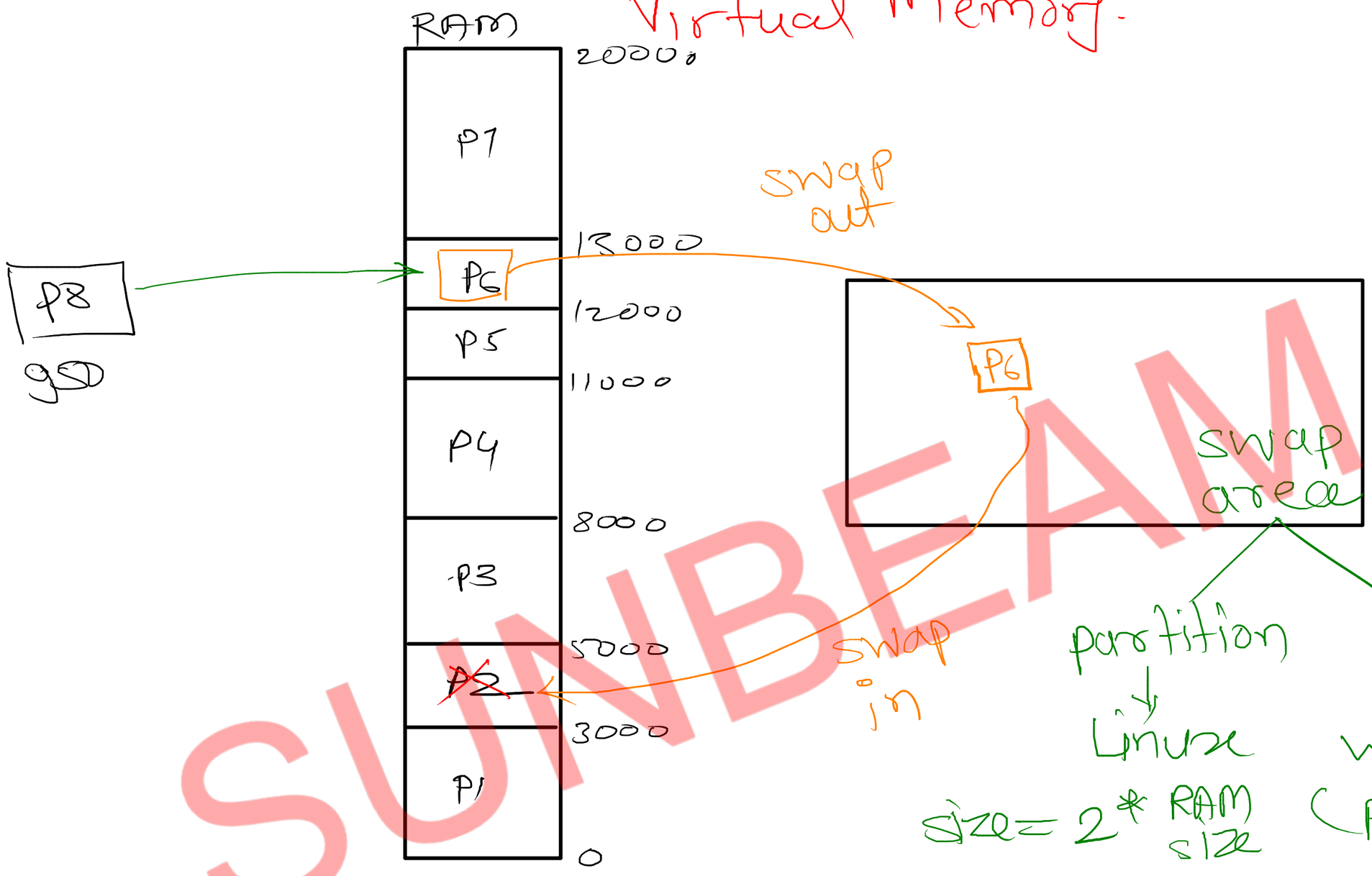
- due to unavailability of contiguous free space, new process can not be loaded into memory.

↓

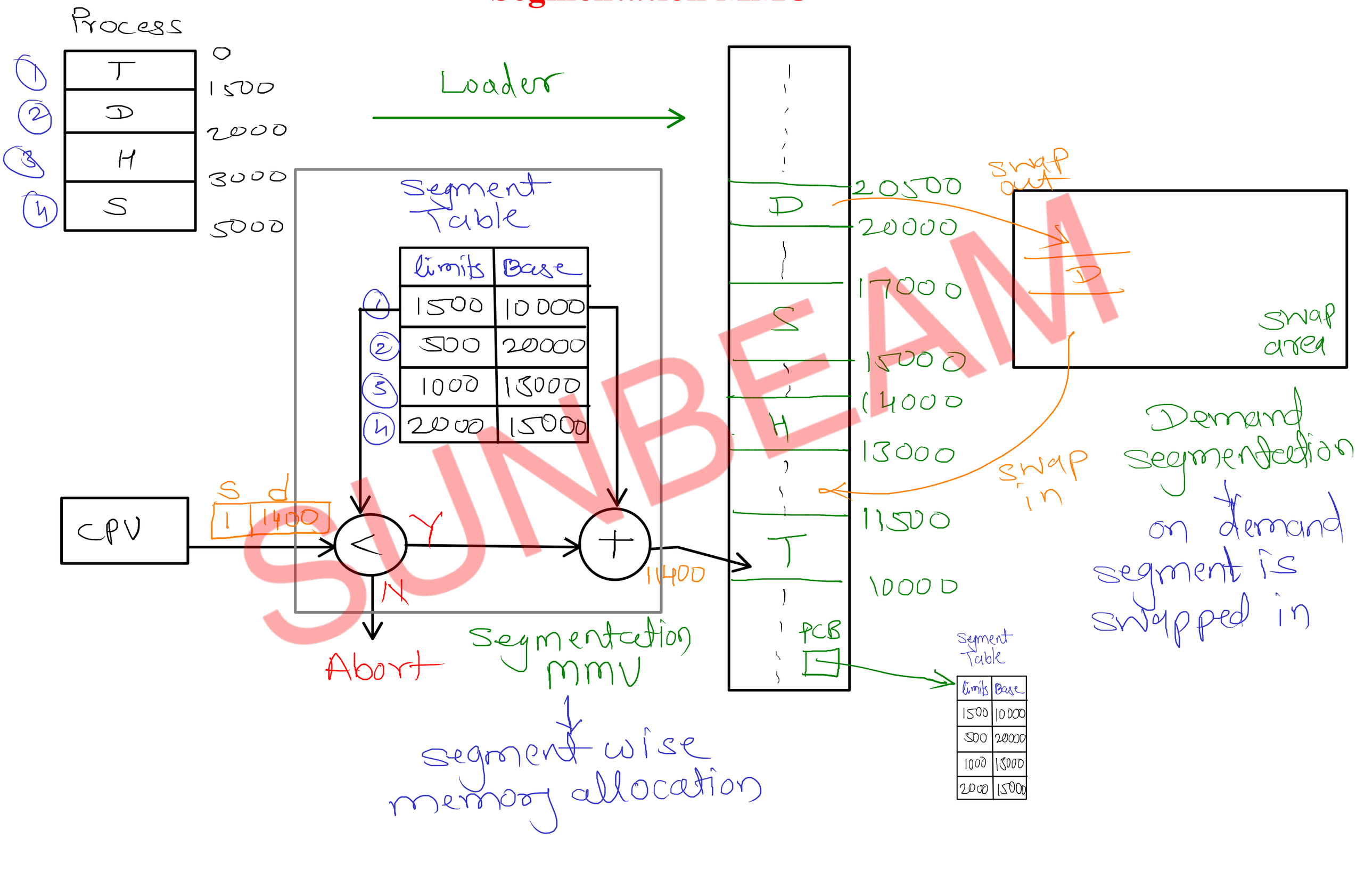
Compaction

- moving processes inside memory to create large free space

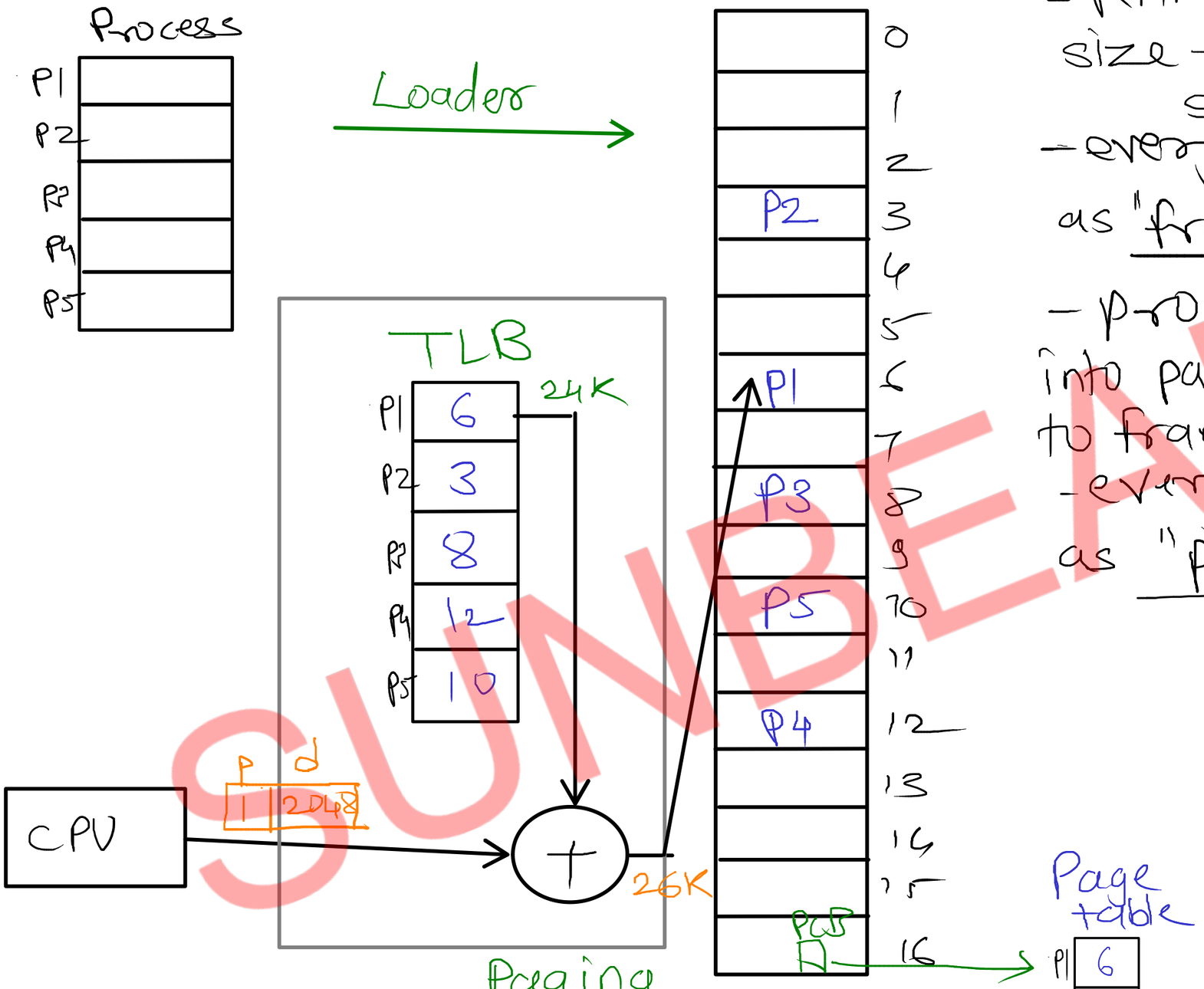
Virtual Memory.



Segmentation MMU

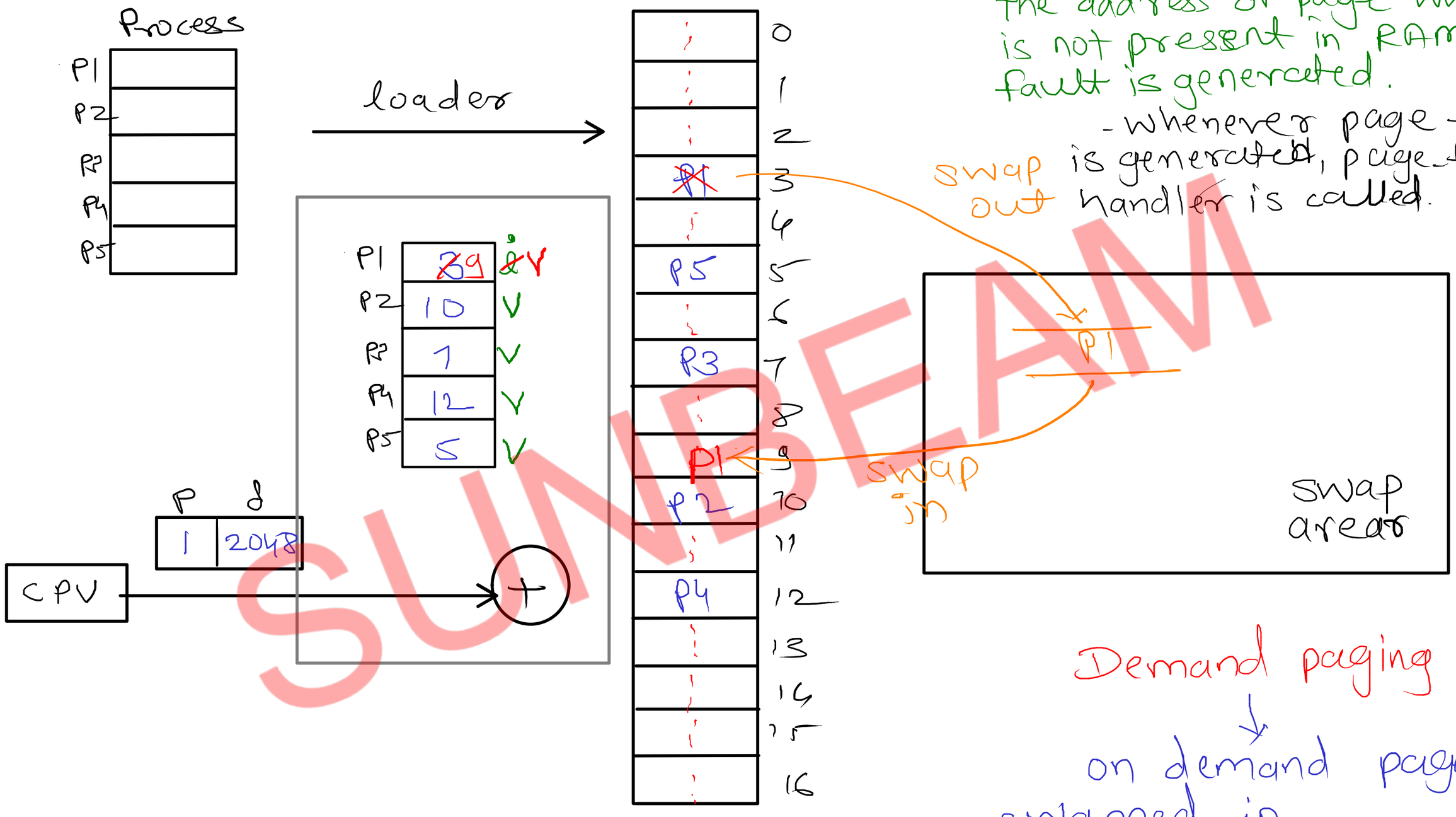


Paging MMU



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

Page fault :
- when CPU request for the address of page which is not present in RAM, this fault is generated.
- whenever page fault is generated, page fault handler is called.



Demand paging
↓
on demand page is swapped in

page_fault_handler( ) {

1) check address is valid or not.
   if address is not valid, abort the process

2) check for read/write access
   if don't have access, abort the process.

3) find the free frame to swap in requested page.

4) swapped in requested page inside memory and update validity bit & mapping in page table as well as TLB.

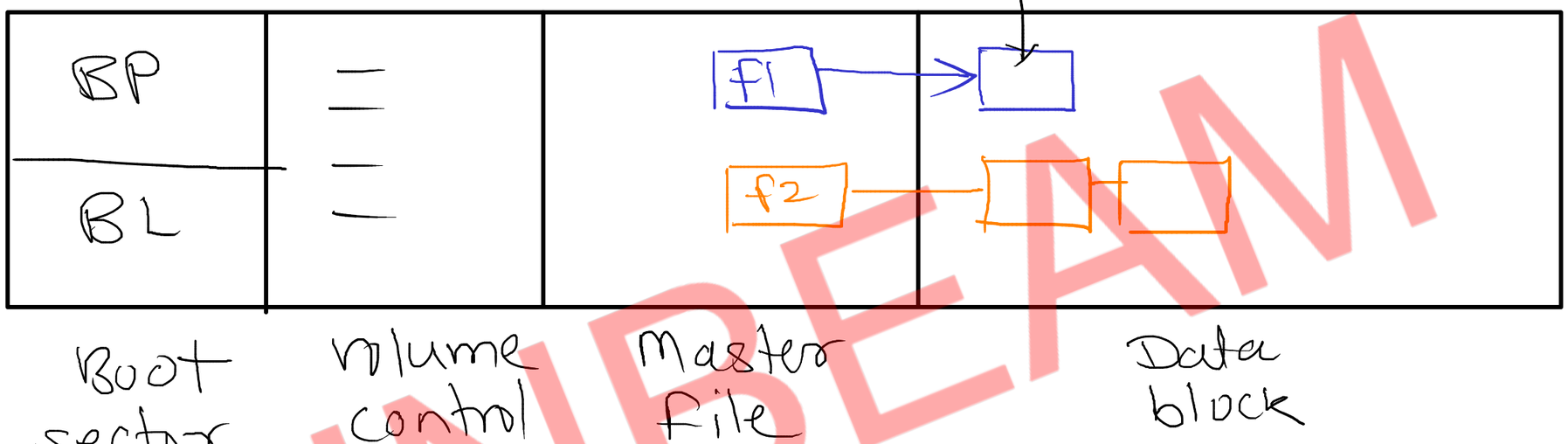
5) reexecute instruction for which page fault was occured.

}

File System

File = data + meta data

(data Block) (File Control Block)



↑ programs responsible for booting

↑ info about partition → label, size, filled, empty

File system — Organising files on hard disk partitions

OS Booting

1) Power ON
2) Load & Execute POST/BIST – check all hardwares
3) Execute bootstrap loader – find bootable partition of harddisk & load boot loader

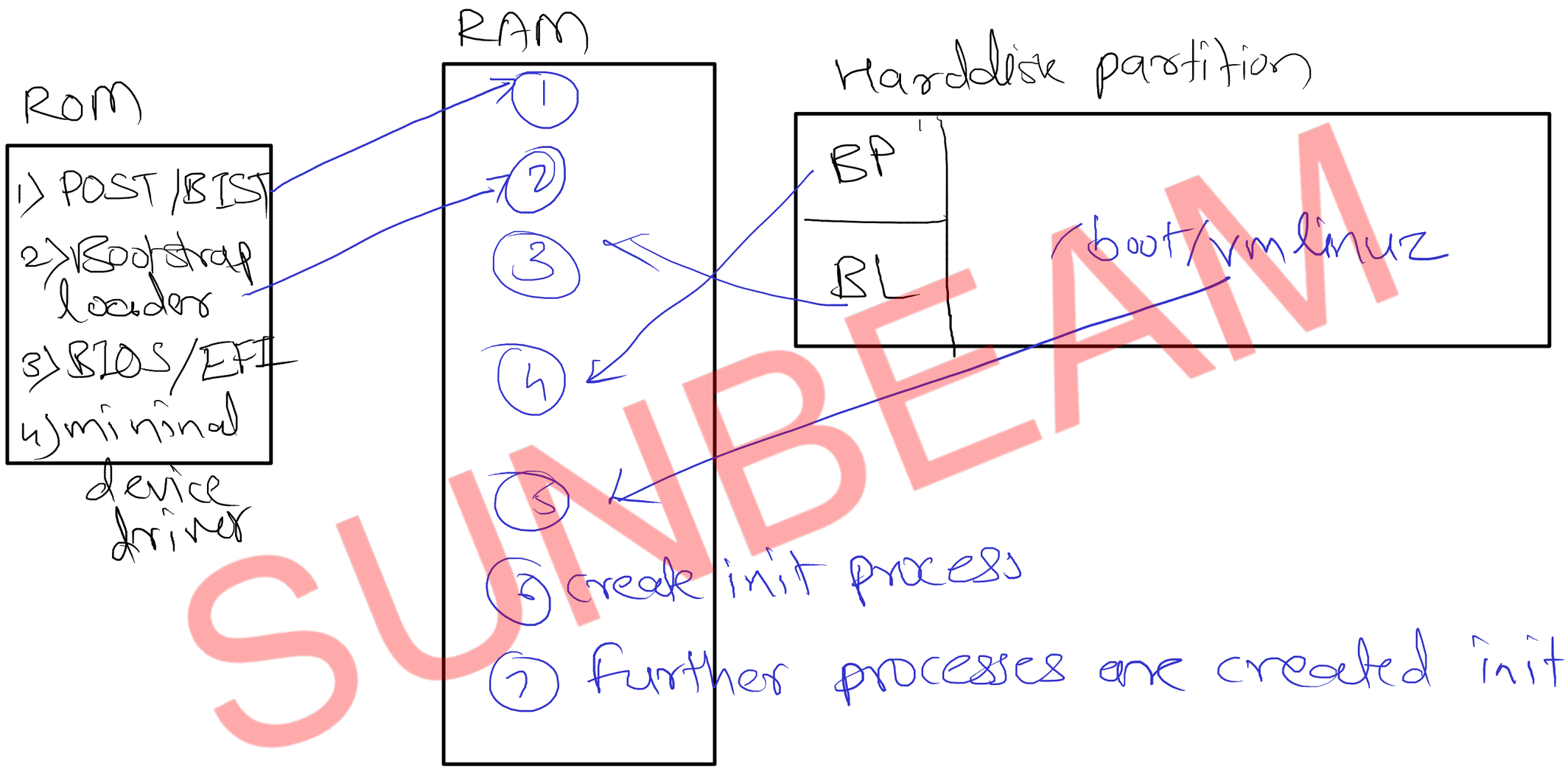
ROM

4) Execute bootloader – shows menu to the user & after selection, load corresponding bootstrap program
5) Execute bootstrap program – load Kernel image into memory.

boot sector of hdd partition

6) Kernel is self extracted inside RAM
7) first process of system is created (init/systemd)
8) further process & services are created by init

Booting Process



Types of kernel

1) Monolithik kernel – BSD UNIX
2) Micro kernel – Symbian, MACH
3) Modular kernel – Windows
4) Hybrid kernel – iOS – Darwin = BSD UNIX + MACH
5) Nano kernel – FreeRTOS

Linux Kernel = static component + Dynamic Component + xServer for UI

static component
↓
MM, PM, HAL, Sched
system calls
IO subsystem

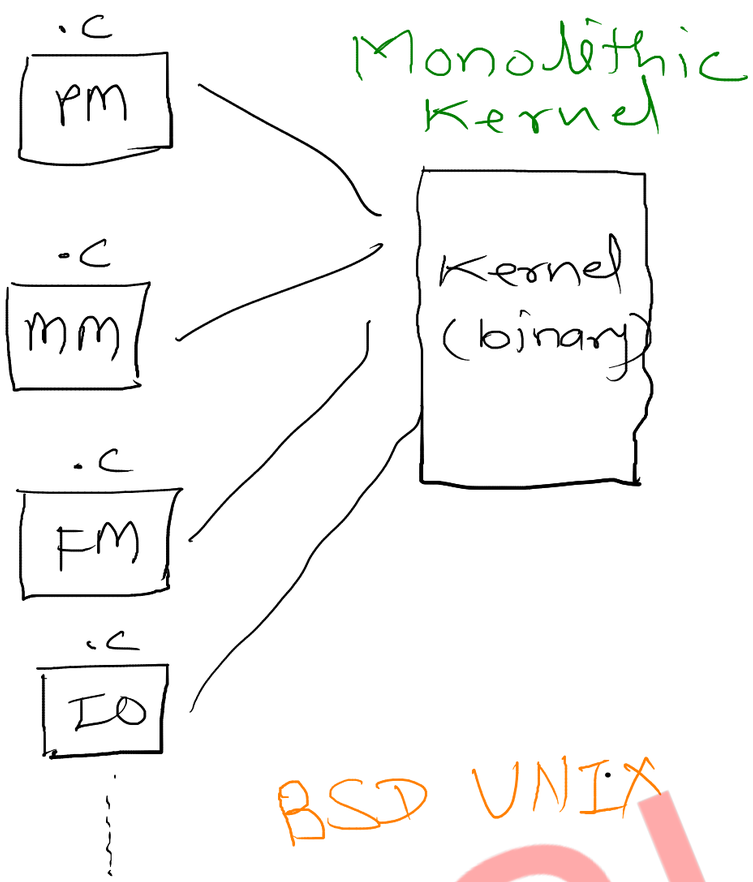
Dynamic Component
File system mgrs
device drivers

kernel
vmlinuz

dynamically loadable modules
(kernel objects)

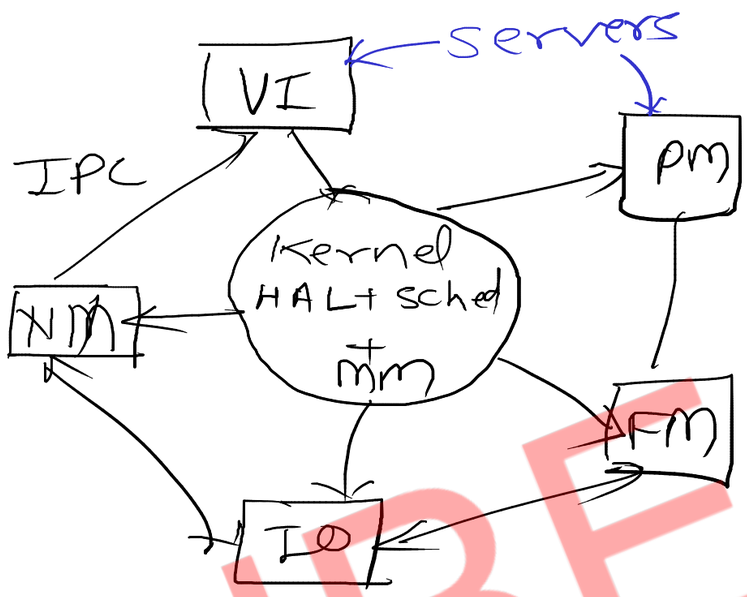
Types of Kernel

Monolithic Kernel



BSD UNIX

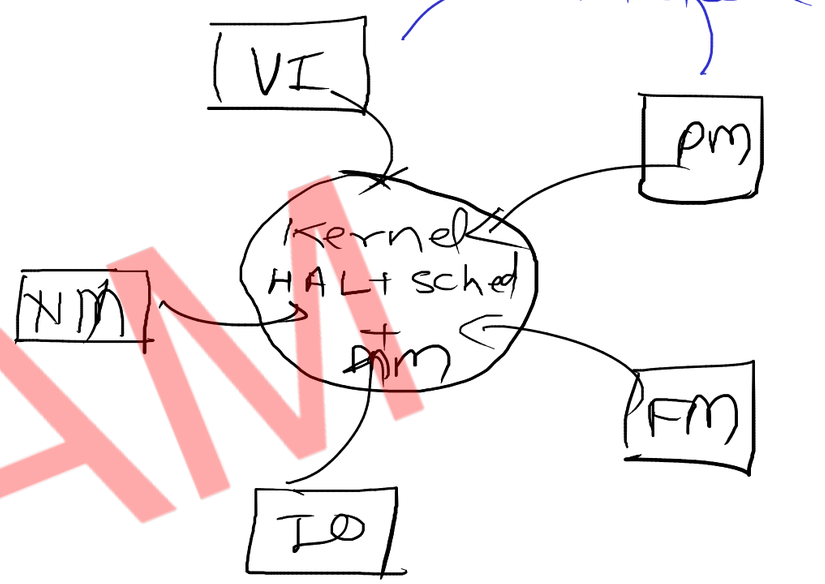
Micro Kernel



symbian
MACH

Modular Kernel

dynamically loadable modules



Windows

Hybrid Kernel { combination of two

Darwin = BSD UNIX + MACH

Nano Kernel } RTOS's