Operating Systems

Quiz

Q. To recover system from deadlock, process which gets terminated is referred as a

a. terminated process
b. target process
c. victim process
d. all of the above

Q During Machine Boot, process to check peripherals connectivity is called as

A. BIOS
B. POST
C. Authentication
D. None of the above

Quiz

Q. The Boot sector files of the system are stored in which computer memory?

(a) RAM
(b) ROM
(c) Cache
(d) Register

Q. Which of the following statements are not correct about the main memory of a computer?

(a) In main memory, data gets lost when power is switched off.
(b) Main memory is faster than secondary memory but slower than registers.
(c) They are made up of semiconductors.
(d) All are correct

Q. Which of the following is the lowest in the computer memory hierarchy?

(a)Cache
(b) RAM
(c) Secondary memory
(d) CPU registers

Q. Which of the following has the fastest speed in the computer memory hierarchy?

(a)Cache
(b) Register in CPU
(c) Main memory
(d) Disk cache

Q. Which memory acts as a buffer between CPU and main memory?

(a) RAM
(b) ROM
(c) Cache
(d) Storage

Q. what is/are necessary and sufficient condition/s to occur deadlock.

a. resource can be allocated for any one process at a time → No Mutual ex.

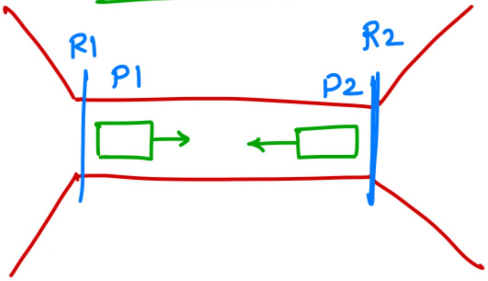
b. control of any resource cannot be taken away forcefully from a process → No pre-emp.

c. each process is holding one resource and requesting for a resource which is held by another process. → hold & wait.

d. circular wait

e. all of the above ✓

Deadlock.



The process are blocked indefintely indeadlock.

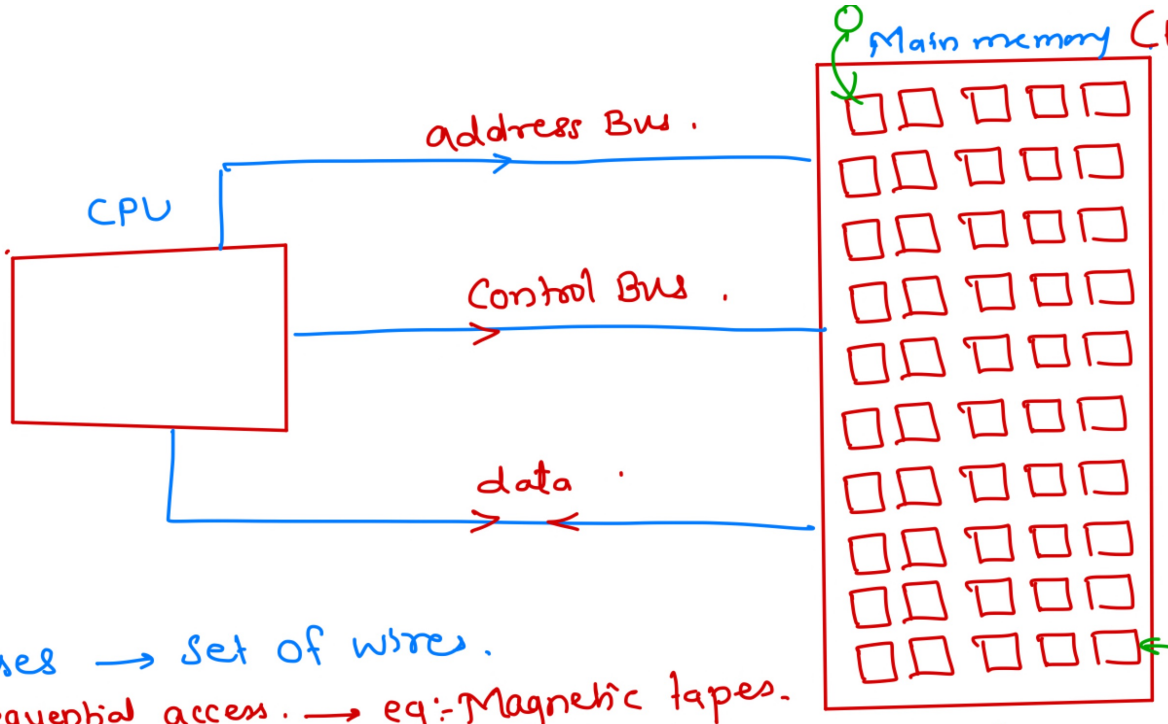
4 characteristic. Deadlock

1. No pre-emption →
2. No Mutual Exclusion
3. Hold & wait.
4. Circular queue.

deadlock prevention..

1. Resource allocation graph.

∴ 2. Banker's algoth



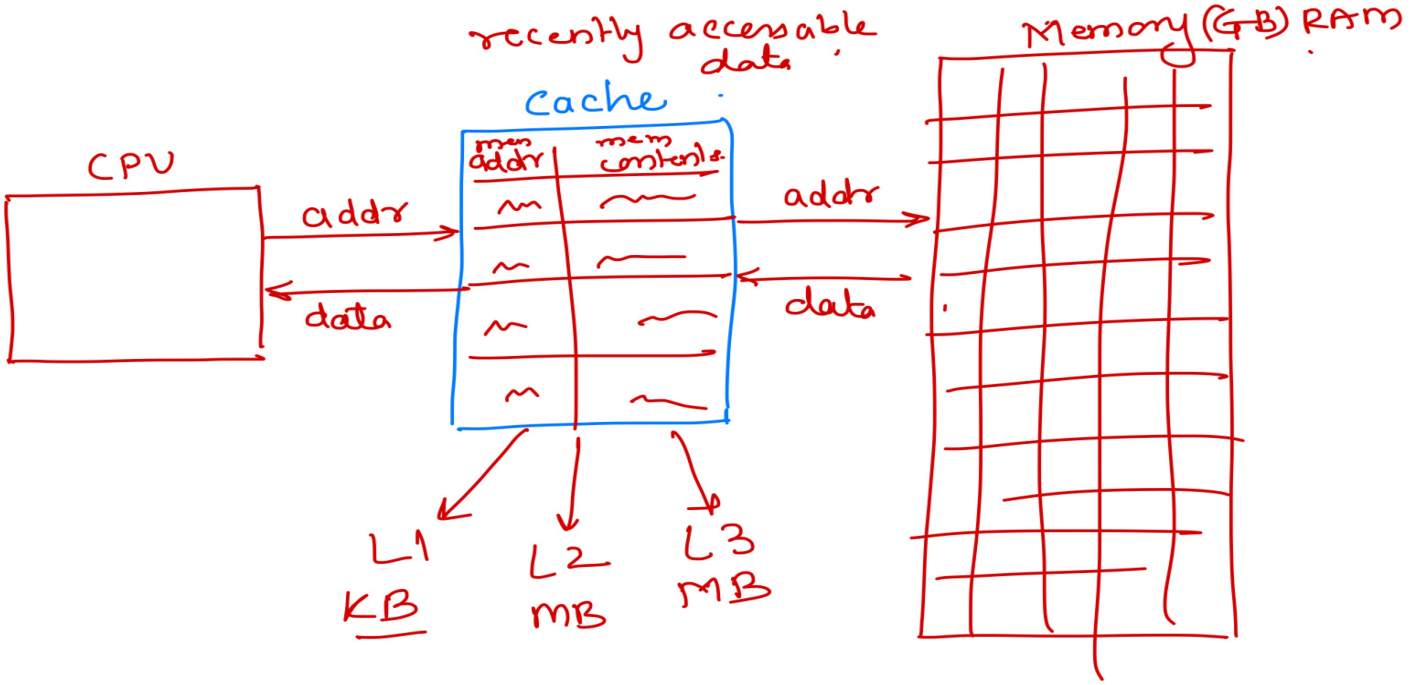
Buses → Set of wires.

① Sequential access. → eg:- Magnetic tapes.

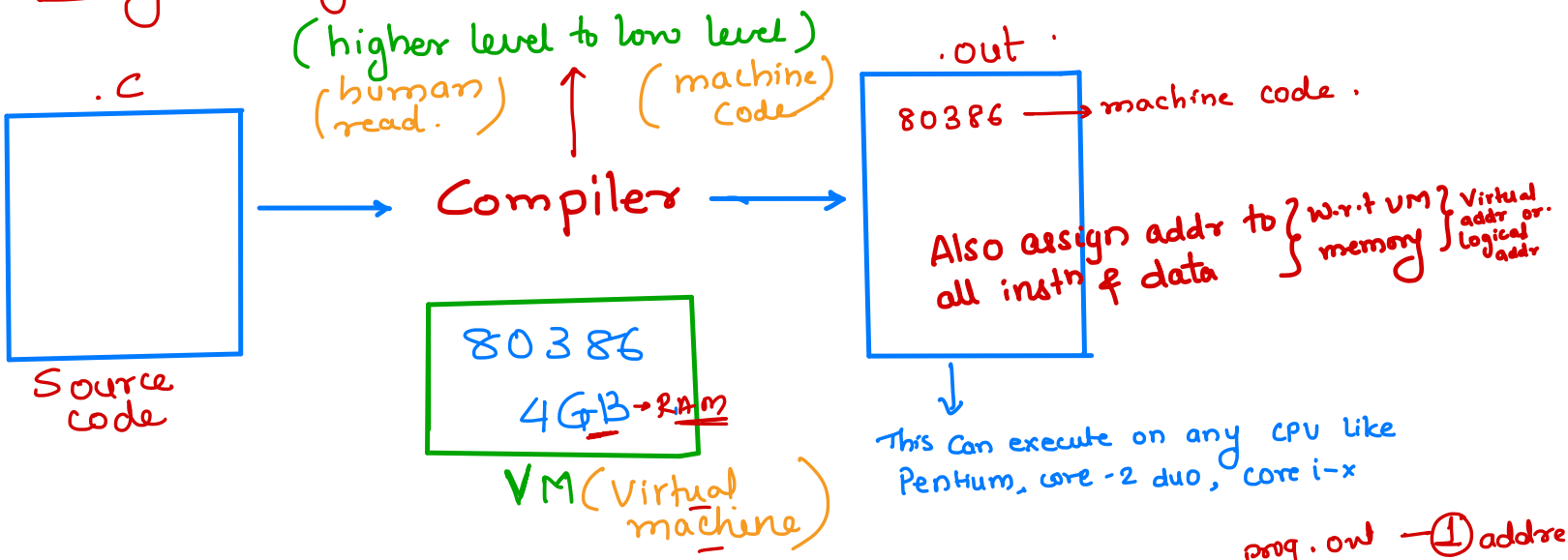
② Direct access → Block address → eg Hard disk

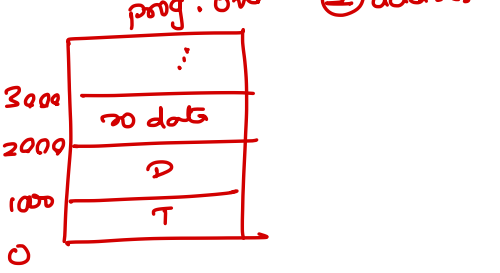
③ Random access → Byte address → eg RAM

④ Associative access → Key-value pair storage → eg Cache


recently accessable data
Cache
Memory (GB) RAM
CPU
addr
data
mem addr
mem contents
addr
data
L1
KB
L2
MB
L3
MB

Memory Management :

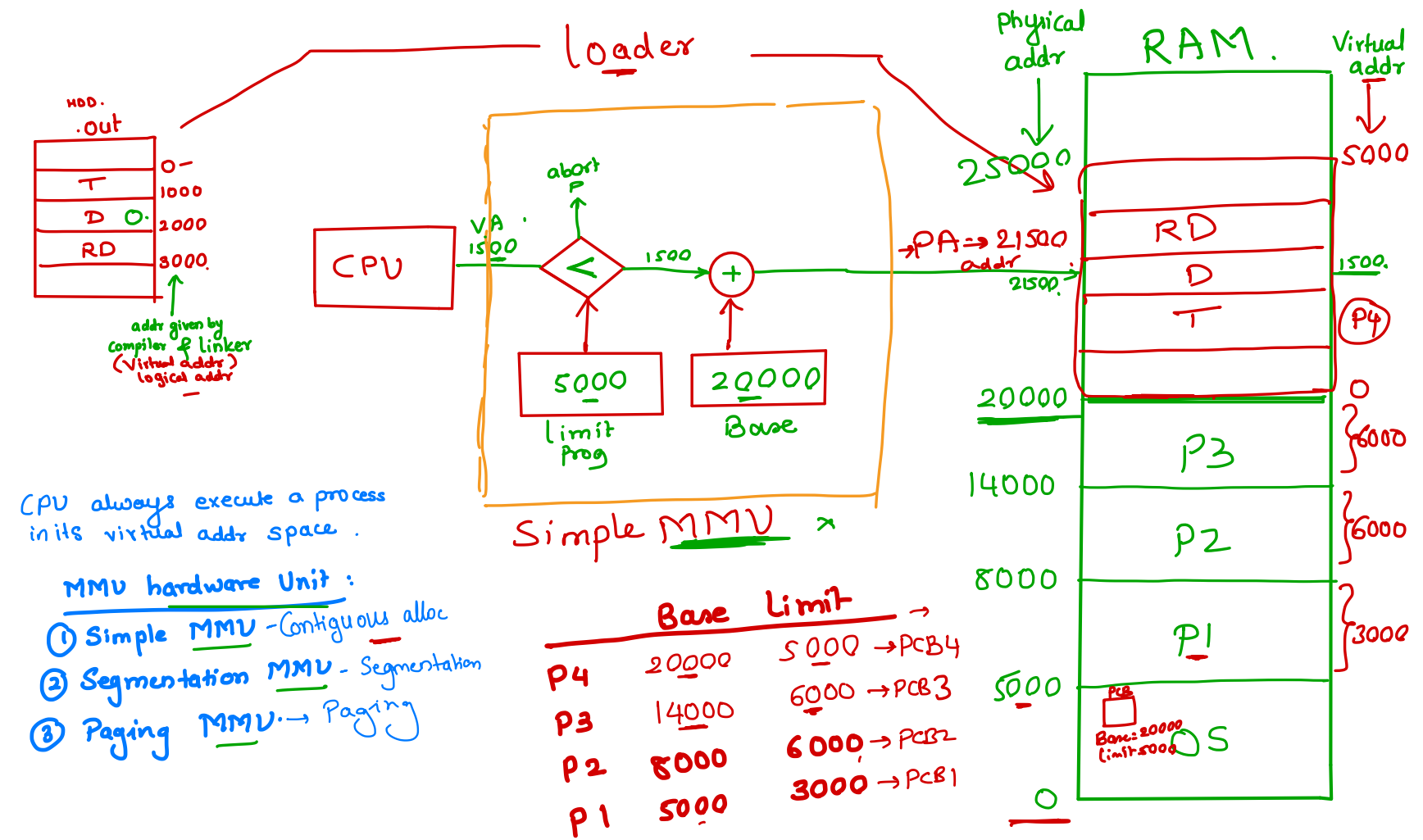




gcc -m32

↓

32 bit Compilation.



CPU always execute a process in its virtual addr space.

MMU hardware Unit :

1. Simple MMU - Contiguous alloc
2. Segmentation MMU - Segmentation
3. Paging MMU → Paging

	Base	Limit	
P4	20000	5000	→ PCB4
P3	14000	6000	→ PCB3
P2	8000	6000	→ PCB2
P1	5000	3000	→ PCB1

Memory Management

- Compiler convert code from high level language to low level language.
- Compiler assumes a low config machine, while converting high level code to low level code called as "Virtual Machine".
- Compiler and Linker assign addresses to each variable/instruction assuming that program will execute in VM RAM. These addressed are called as "virtual address" or "logical address". The set of virtual addresses used by the process is referred "Virtual address space".
- However while execution these addresses might be occupied by other processes. Loader relocates all instructions/variables to the address available in RAM. The actual addresses given to the process at runtime are called as "physical address" or "real address". The set of physical addresses used by the process is referred "Physical address space".
- CPU always executes a process in its virtual address space i.e. CPU always request virtual addressed (on address bus).
- These virtual addresses are verified and then converted into corresponding physical addresses by a special hardware unit called as "Memory Management Unit (MMU)".
- Simple MMU holds physical base address and limit (length) of the process. The base & limit of each process is stored in its PCB and then loaded into MMU during context switch.
- In multi-programming OS, multiple programs are loaded in memory.

Memory Management

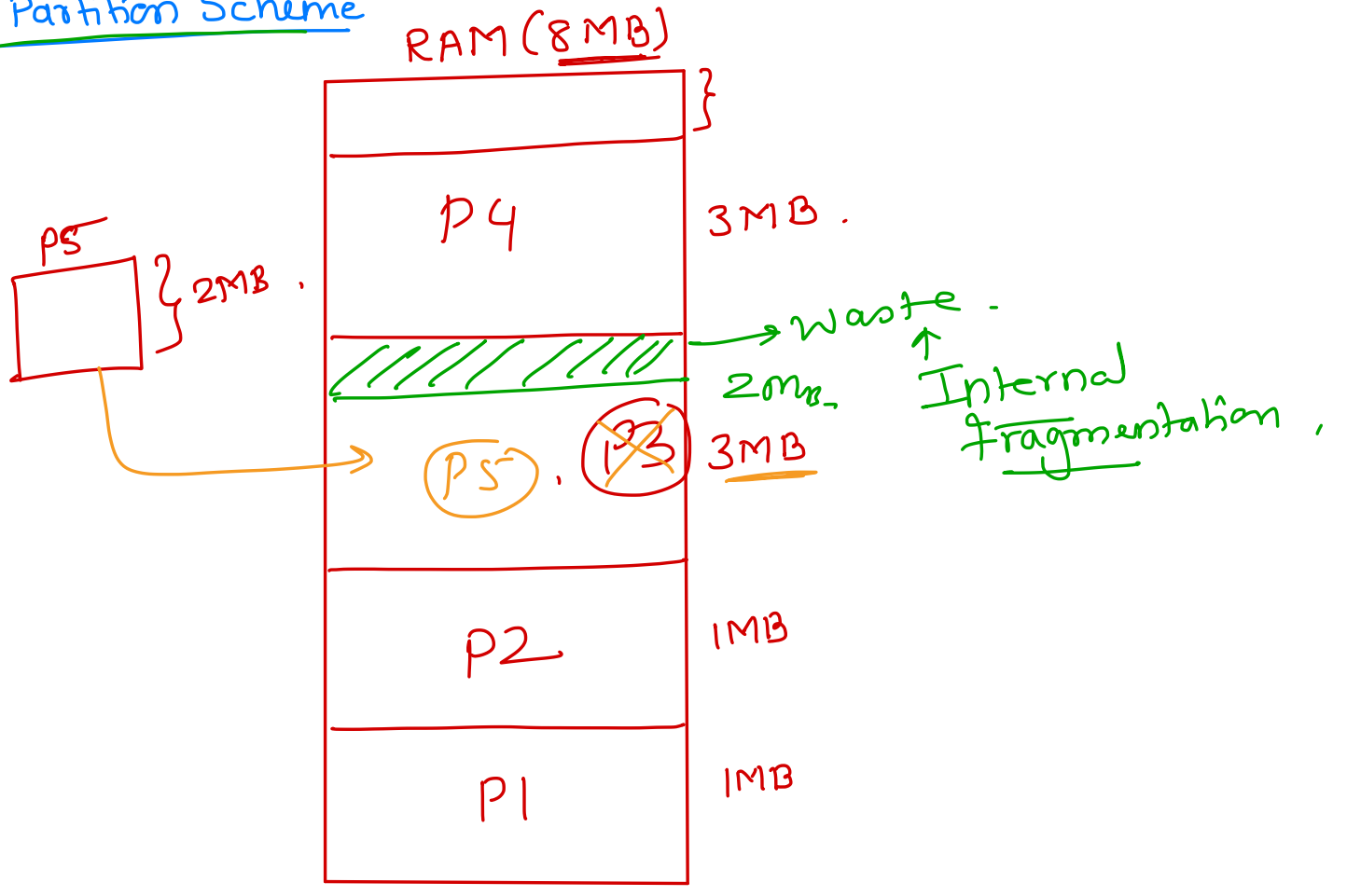
- RAM memory should be divided for multiple processes running concurrently.
- Memory Mgmt. scheme used by any OS depends on the MMU hardware used in the machine.
- There are three memory management schemes are available (as per MMU hardware).
  1. Contiguous Allocation
  2. Segmentation
  3. Paging

Contiguous Allocation

Fixed Partition

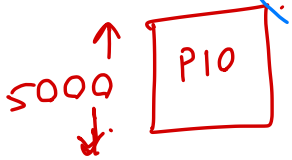
- RAM is divided into fixed sized partitions.
- This method is easy to implement.
- Number of processes are limited to number of partitions.
- Size of process is limited to size of partition.
- If process is not utilizing entire partition allocated to it, the remaining memory is wasted. This is called as "internal fragmentation".

Fixed Partition Scheme

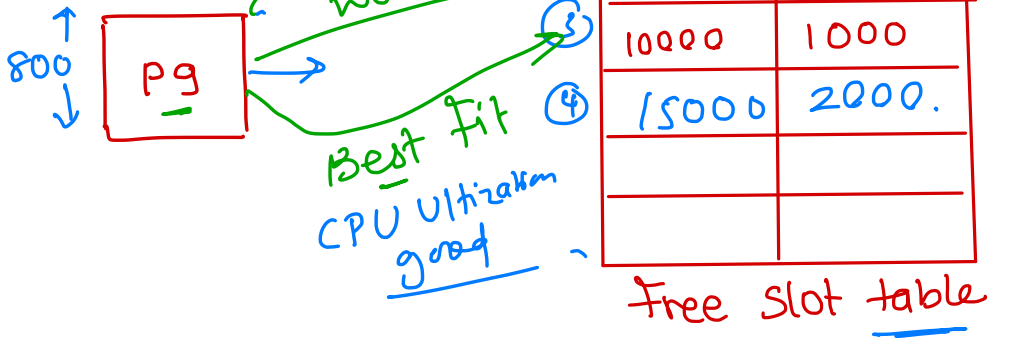


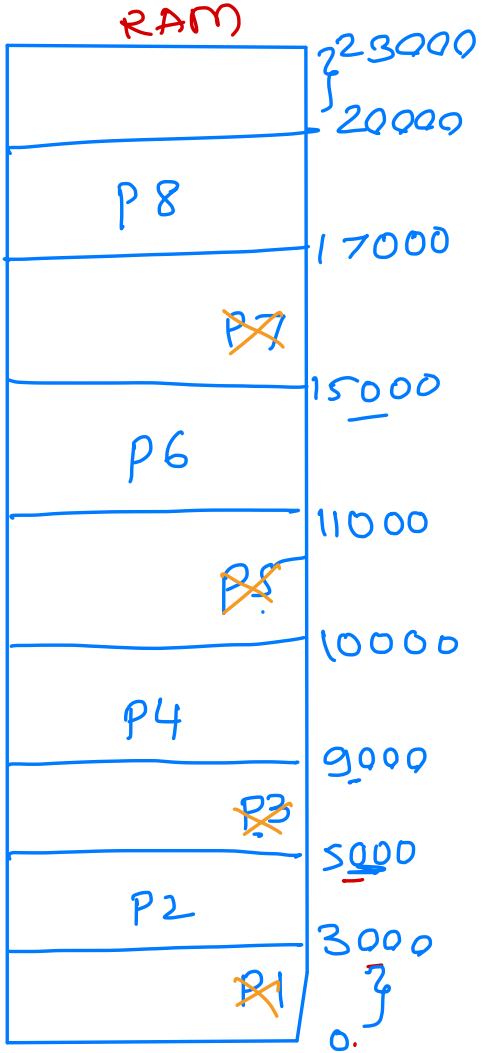
Variable Partition Scheme

Soln
Compaction : shifting processes in mem so that max contiguous free space is available



external fragmentation:
mem required is available but not contiguous.

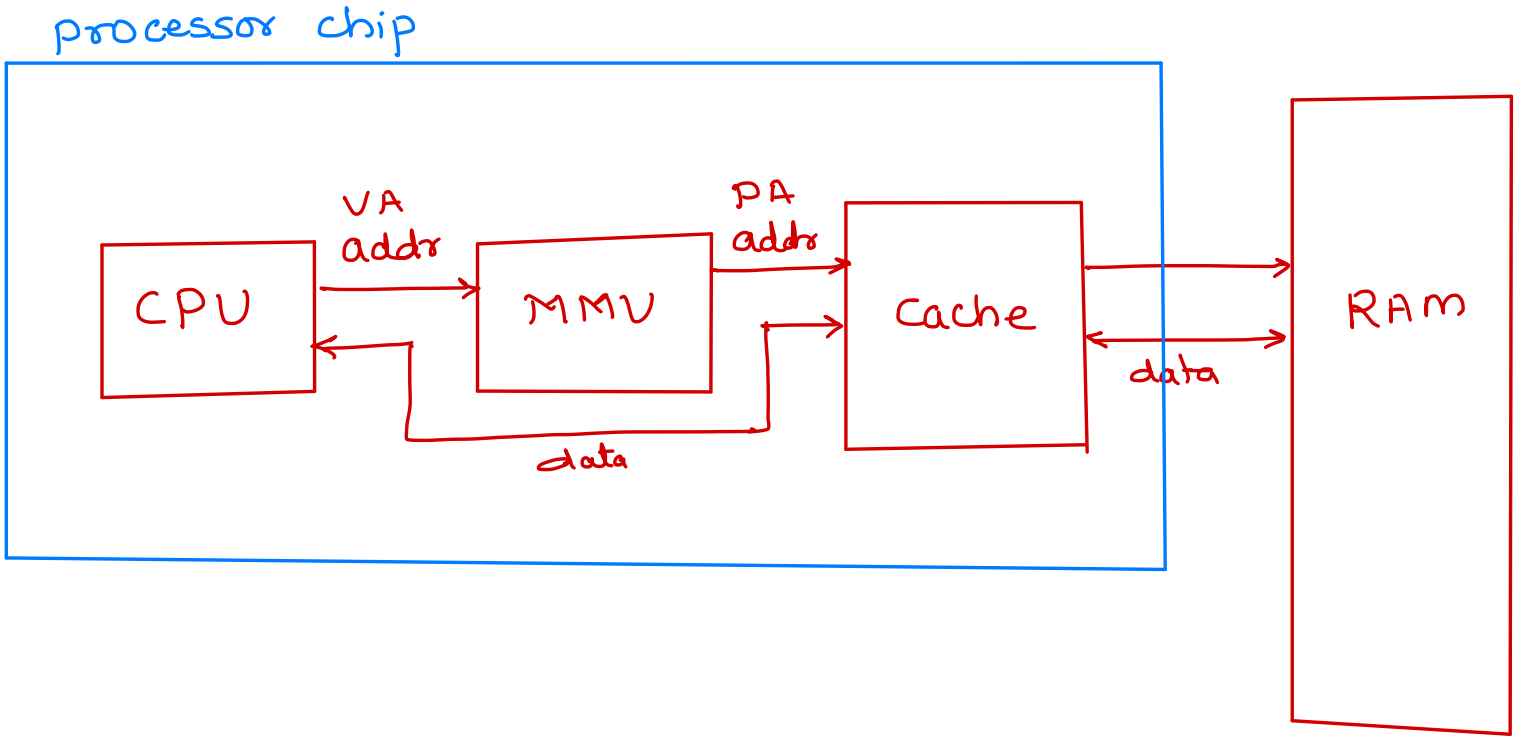




Memory Management

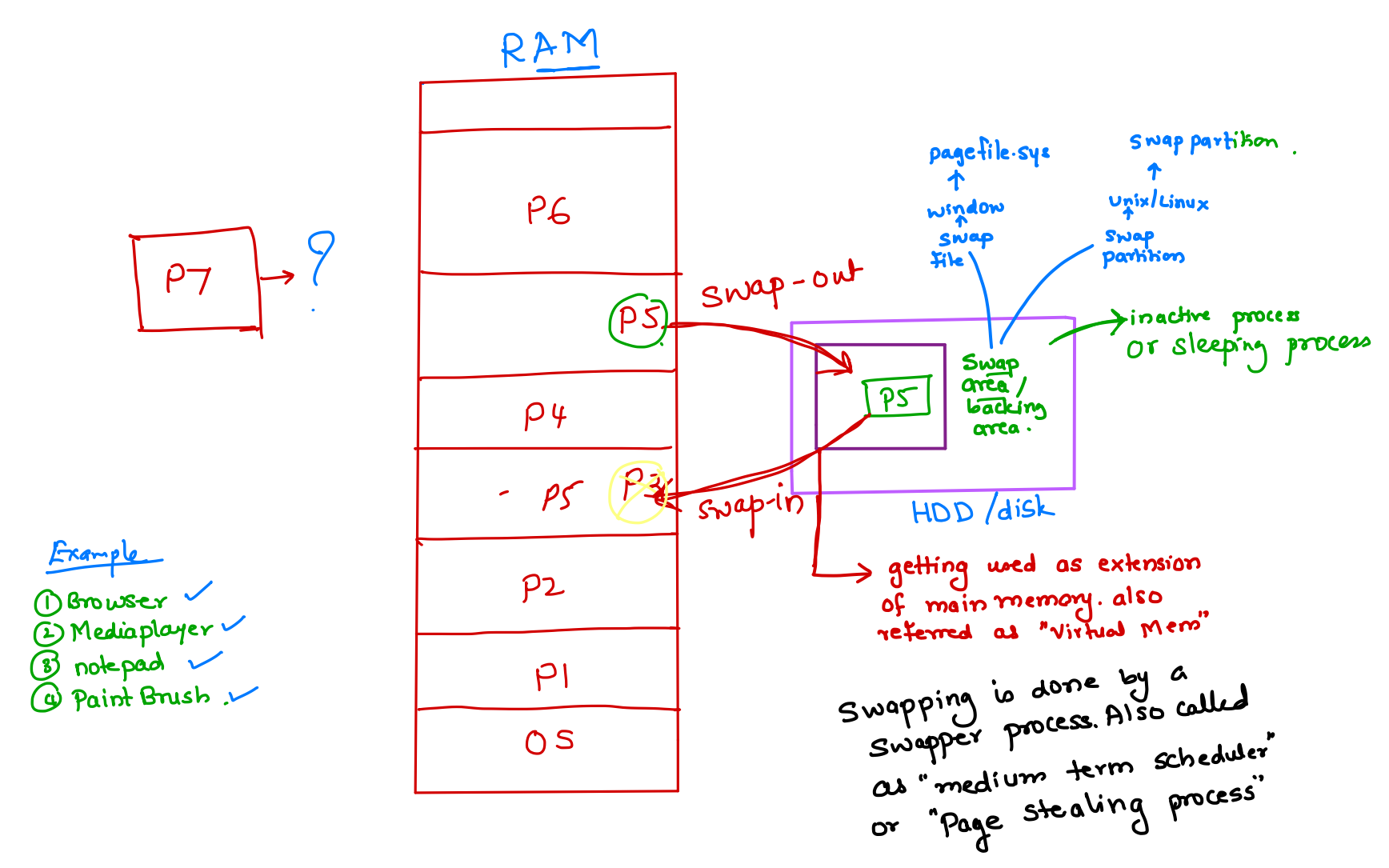
Dynamic Partition

- Memory is allocated to each process as per its availability in the RAM. After allocation and deallocation of few processes, RAM will have few used slots and few free slots.
- OS keep track of free slots in form of a table.
- For any new process, OS use one of the following mechanism to allocate the free slot.
  - First Fit: Allocate first free slot which can accommodate the process.
  - Best Fit: Allocate that free slot to the process in which minimum free space will remain.
  - Worst Fit: Allocate that free slot to the process in which maximum free space will remain.
- Statistically it is proven that First fit is faster algo; while best fit provides better memory utilization.
- Memory info (physical base address and size) of each process is stored in its PCB and will be loaded into MMU registers (base & limit) during context switch.


processor chip
CPU
VA
addr
MMU
PA
addr
Cache
RAM
data
data

Memory Management

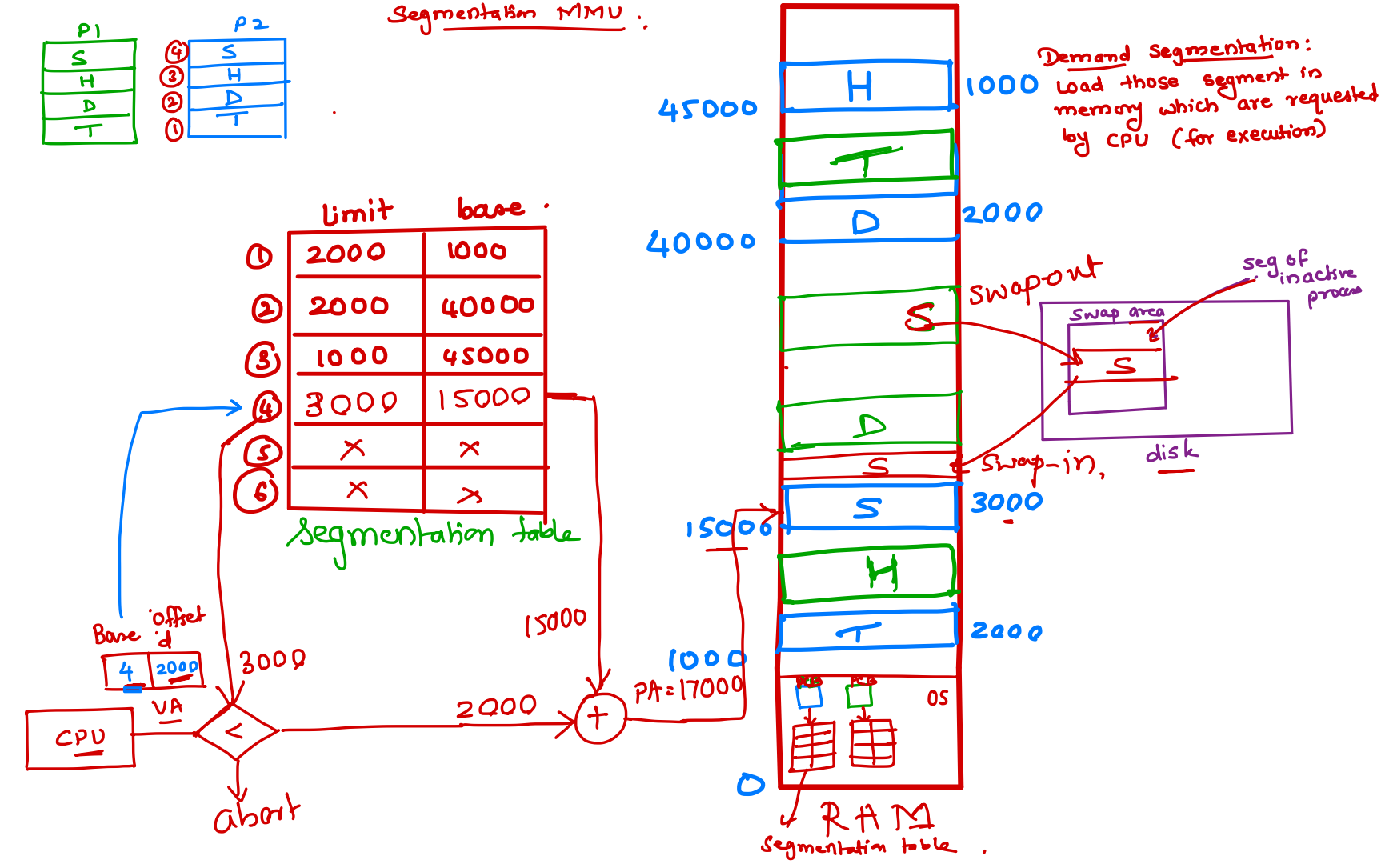
- CPU request virtual address (address of the process) and is converted into physical address by MMU as shown in diag.
- If invalid virtual address is requested by the CPU, process will be terminated.
- If amount of memory required for a process is available but not contiguous, then it is called as "external fragmentation".
- To resolve this problem, processes in memory can be shifted/moved so that max contiguous free space will be available. This is called as "compaction".


RAM
P6
P5
P4
P5
P3
P2
P1
OS
P7
?
swap-out
swap-in
pagefile.sys
window
swap file
swap partition
Unix/Linux
swap partition
P5
Swap area / backing area.
inactive process or sleeping process
HDD / disk
getting used as extension of main memory. also referred as "virtual Mem"
Example
1. Browser
2. Mediaplayer
3. notepad
4. Paint Brush
Swapping is done by a swapper process. Also called as "medium term scheduler" or "Page stealing process"

Memory Management

Virtual Memory

- The portion of the hard disk which is used by OS as an extension of RAM, is called as "virtual memory".
- If sufficient RAM is not available to execute a new program or grow existing process, then some of the inactive process is shifted from main memory (RAM), so that new program can execute in RAM (or existing process can grow). It is also called as "swap area" or "swap space".
- Shifting a process from RAM to swap area is called as "swap out" and shifting a process from swap to RAM is called as "swap in".
- In few OS, swap area is created in form of a partition. E.g. UNIX, Linux, ...
- In few OS, swap area is created in form of a file E.g. Windows (pagefile.sys), ...
- Virtual memory advantages:
  - Can execute more number of programs.
  - Can execute bigger sized programs.


Segmentation MMU :
P1
S
H
D
T
P2
4 S
3 H
2 D
1 T
Demand Segmentation :
Load those segment in memory which are requested by CPU (for execution)
limit
base
1 2000 1000
2 2000 40000
3 1000 45000
4 3000 15000
5 x x
6 x x
Segmentation table
Base
offset 'd'
4
2000
VA
CPU
<
3000
abort
15000
2000
+
PA=17000
1000
45000
40000
15000
0
H
T
D
S
S
D
S
S
H
T
1000
2000
3000
2000
OS
RAM
Segmentation table
swap-out
seg of inactive process
swap area
S
Swap-in
disk

Memory Management

Segmentation

* Instead of allocating contiguous memory for the whole process, contiguous memory for each segment can be allocated. This scheme is known as "segmentation".

* Since process does not need contiguous memory for entire process, external fragmentation will be reduced.

* In this scheme, PCB is associated with a segment table which contains base and limit (size) of each segment of the process.

* During context switch these values will be loaded into MMU segment table.

* CPU request virtual address in form of segment address and offset address.

* Based on segment address appropriate base-limit pair from MMU is used to calculate physical address as shown in diag.

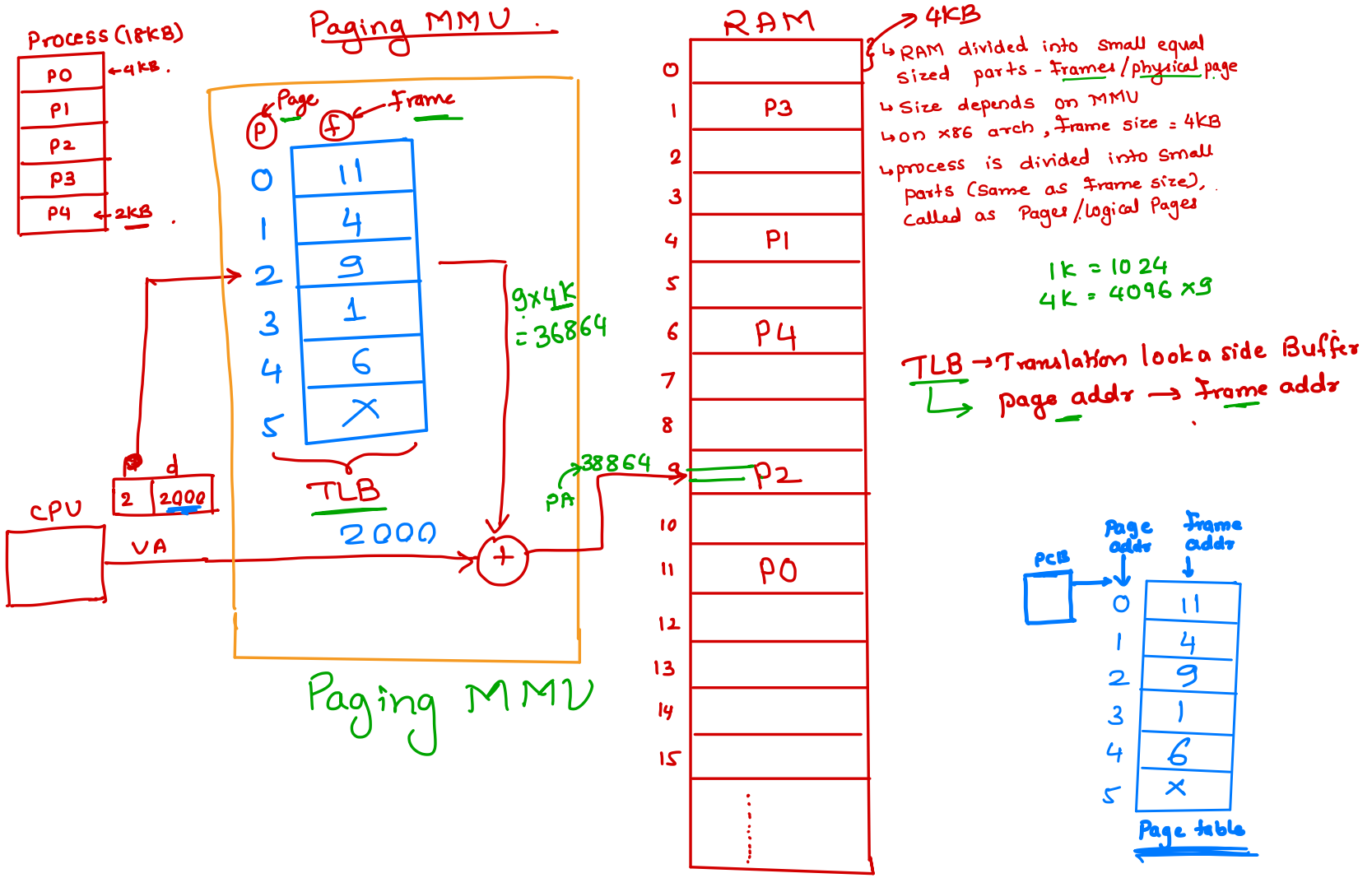
* MMU also contains STBR register which contains address of process's segment table in the RAM.

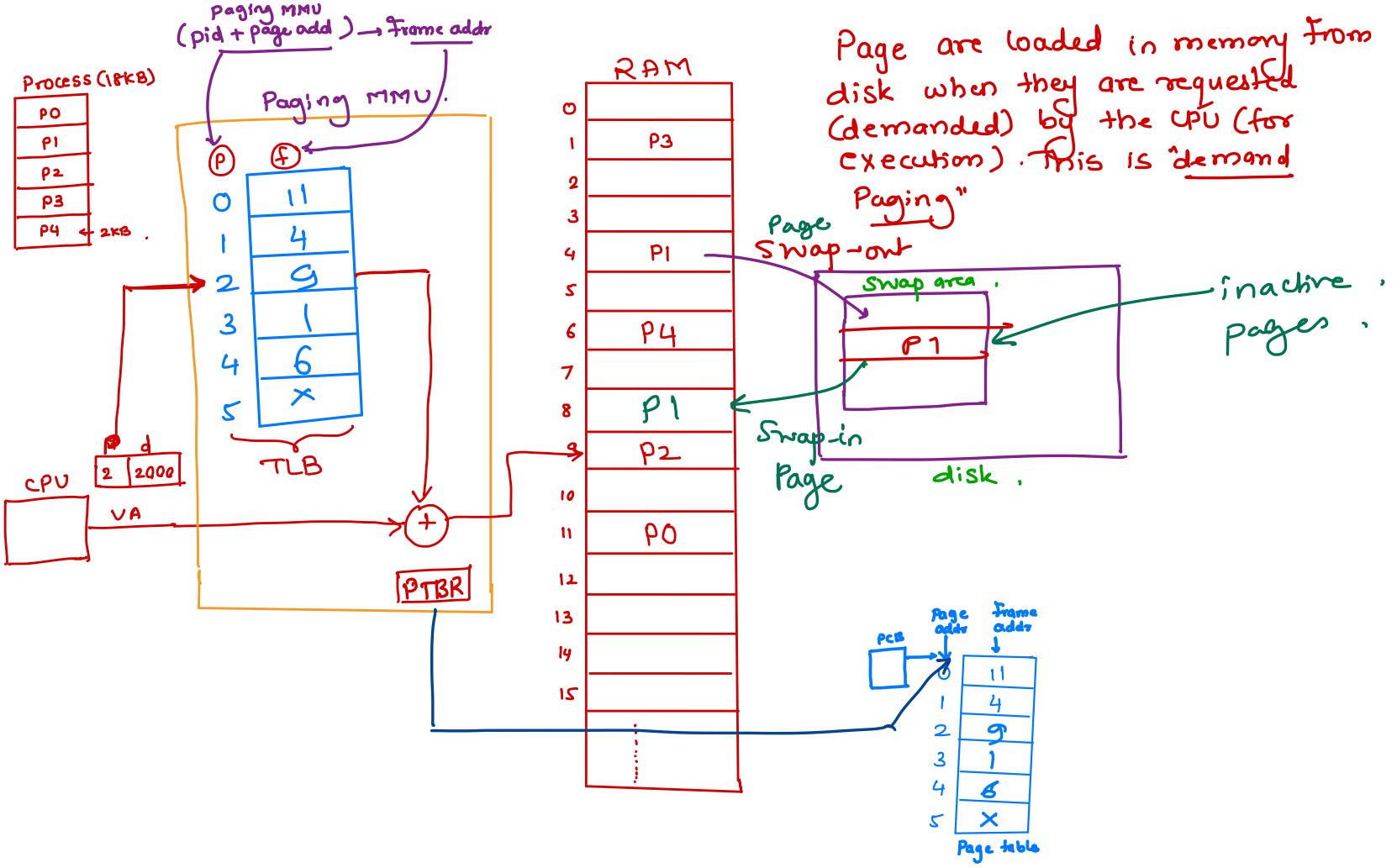
Demand Segmentation

* If virtual memory concept is used along with segmentation scheme, in case low memory, OS may swap out a segment of inactive process.

* When that process again start executing and ask for same segment (swapped out), the segment will be loaded back in the RAM. This is called as "demand segmentation".

* If segment is present in main memory, its entry in seg table is said to be valid. If segment is swapped out, its entry in segment table is said to be invalid.


Process (18KB)
P0
P1
P2
P3
P4
4KB
2KB
Paging MMU
Page
Frame
0 11
1 4
2 9
3 1
4 6
5 X
TLB
9x4K
= 36864
CPU
p d
2 2000
VA
2000
PA
38864
RAM
0
1 P3
2
3
4 P1
5
6 P4
7
8
9 P2
10
11 P0
12
13
14
15
4KB
RAM divided into small equal sized parts - frames / physical page
Size depends on MMU
on x86 arch, frame size = 4KB
process is divided into small parts (same as frame size), called as Pages / Logical Pages
1K = 1024
4K = 4096 x9
TLB → Translation look a side Buffer
page addr → frame addr
Paging MMU
PCB
Page addr
Frame addr
0 11
1 4
2 9
3 1
4 6
5 X
Page table


Paging MMU
(pid + page add) → frame addr
Process (18KB)
P0
P1
P2
P3
P4 ← 2KB
Paging MMU
p
f
0 11
1 4
2 9
3 1
4 6
5 X
TLB
p d
2 2000
CPU
VA
+
PTBR
RAM
0
1 P3
2
3
4 P1
5
6 P4
7
8 P1
9 P2
10
11 P0
12
13
14
15
Page are loaded in memory from disk when they are requested (demanded) by the CPU (for execution). This is "demand Paging"
Page Swap-out
swap area
P1
inactive pages
Swap-in Page
disk
PCB
Page addr
Frame addr
0 11
1 4
2 9
3 1
4 6
5 X
Page table

Memory Management

Paging

* RAM is divided into small equal sized partitions called as "frames" / "physical pages".
* Process is divided into small equal sized parts called as "pages" or "logical/virtual pages".
* page size = frame size.
* One page is allocated to one empty frame.
* OS keep track of free frames in form of a linked list.
* Each PCB is associated with a table storing mapping of page address to frame address. This table is called as "page table".
* During context switch this table contents are loaded into MMU.
* CPU requests a virtual address in form of page address and offset address. It will be converted into physical address as shown in diag.
* MMU also contains a PTBR, which keeps address of page table in RAM.
* If a page is not utilizing entire frame allocated to it (i.e. page contents are less than frame size), then it is called as "internal fragmentation".
* Frame size can be configured in the hardware. It can be 1KB, 2KB or 4KB, ...
* Typical Linux and Windows OS use page size = 4KB.

Memory Management

Demand Paging

* When virtual memory is used with paging memory management, pages can be swapped out in case of low memory.

- The pages will be loaded into main memory, when they are requested by the CPU. This is called as "demand paging".
- A page table entry (PTE) is valid if page is present in main memory , otherwise entry is invalid.
- What are the reason for invalid entry →page is swap out or entry is invalid
- If CPU requesting any page that is not in main memory(i.e. PTE is invalid), then it is a page fault exception.
- **Page fault → Page fault handler(OS generate)**

1. Check if address due to which page fault occurred is valid. If not , then abort the process.
2. Allocate an empty frame.
3. If page is on swap area, swap in page in that frame
4. Update PTE
5. Re-execute instruction at which page fault occurred.