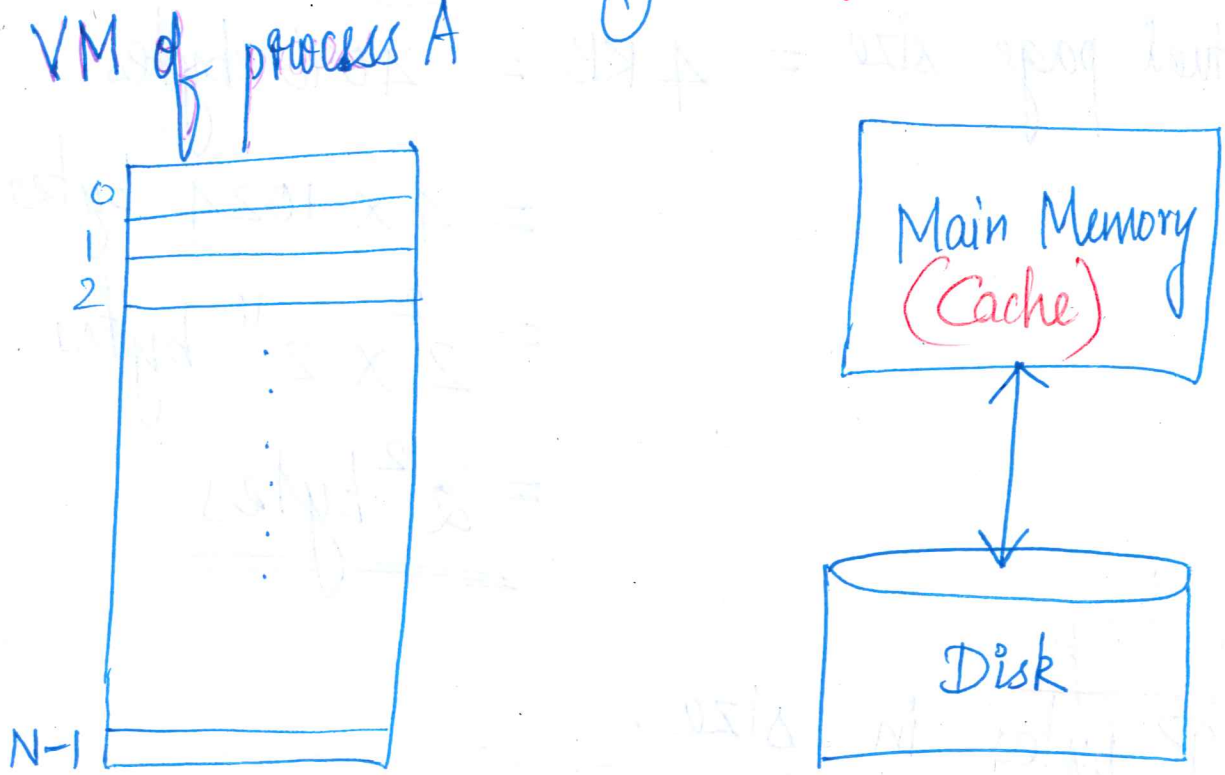
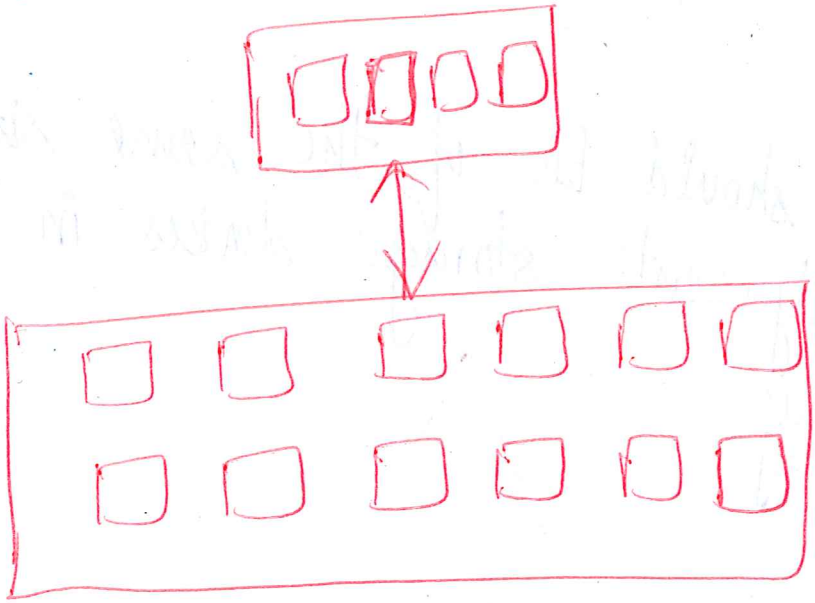
Lecture - 35

Virtual Memory



Remember "block size" in cache memories?



Block ≡ Virtual Page in VM system.

Each **Virtual Page** is $P = 2^p$ bytes in size.

eg. Virtual page size = 4 KB = 4096 bytes

$= 4 \times 1024$ bytes

$= 2^2 \times 2^{10}$ bytes

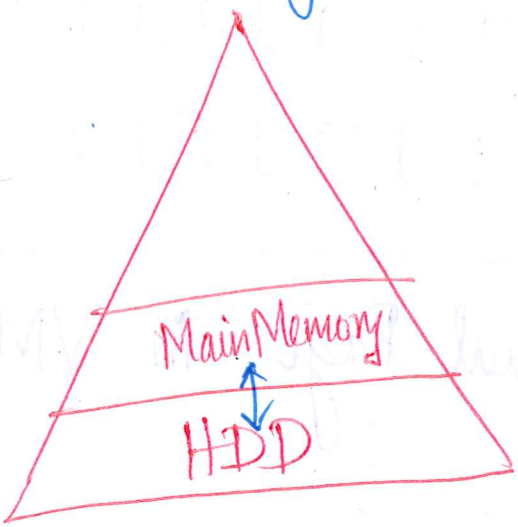
$= 2^{12}$ bytes

Physical Page

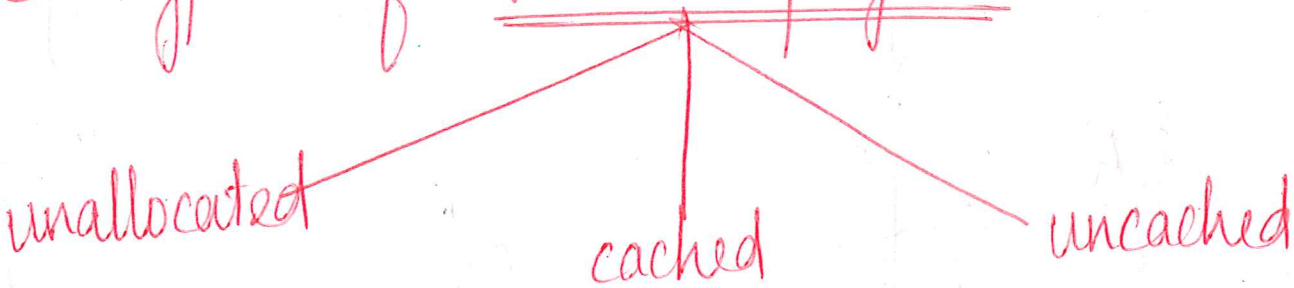
Also P bytes in size.

ie. Virtual page size = Physical page size

∴ Blocks should be of the same size between 2 adjacent storage devices in the memory hierarchy.



3 types of virtual pages (3)



Unallocated

* pages that are not yet created by the VM system.
* no data associated with them.
* do not occupy any space on disk.

Cached

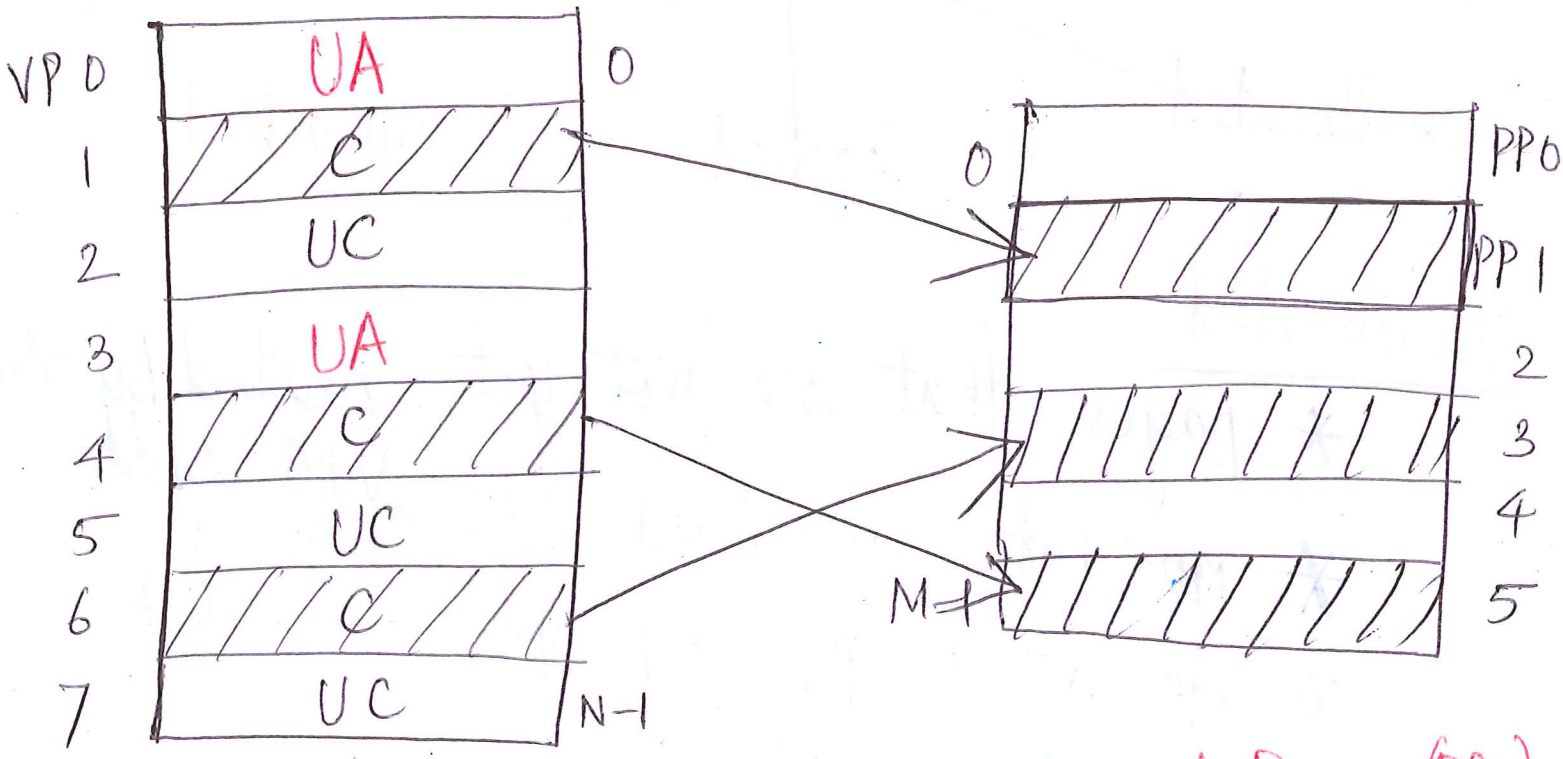
* Allocated pages that are currently cached in physical memory.

Uncached

Allocated pages that are NOT cached in physical memory.

Virtual Memory

Physical Memory



Virtual Pages (VPs) stored on disk

Physical Pages (PPs) cached in DRAM.

UA – Unallocated
C – Cached
UC – Uncached

Main Memory (DRAM) as a Cache

1. Block size (Virtual Page size) = 4KB to 2MB

∴ Miss penalty is large!

Expense of accessing the first byte in disk.

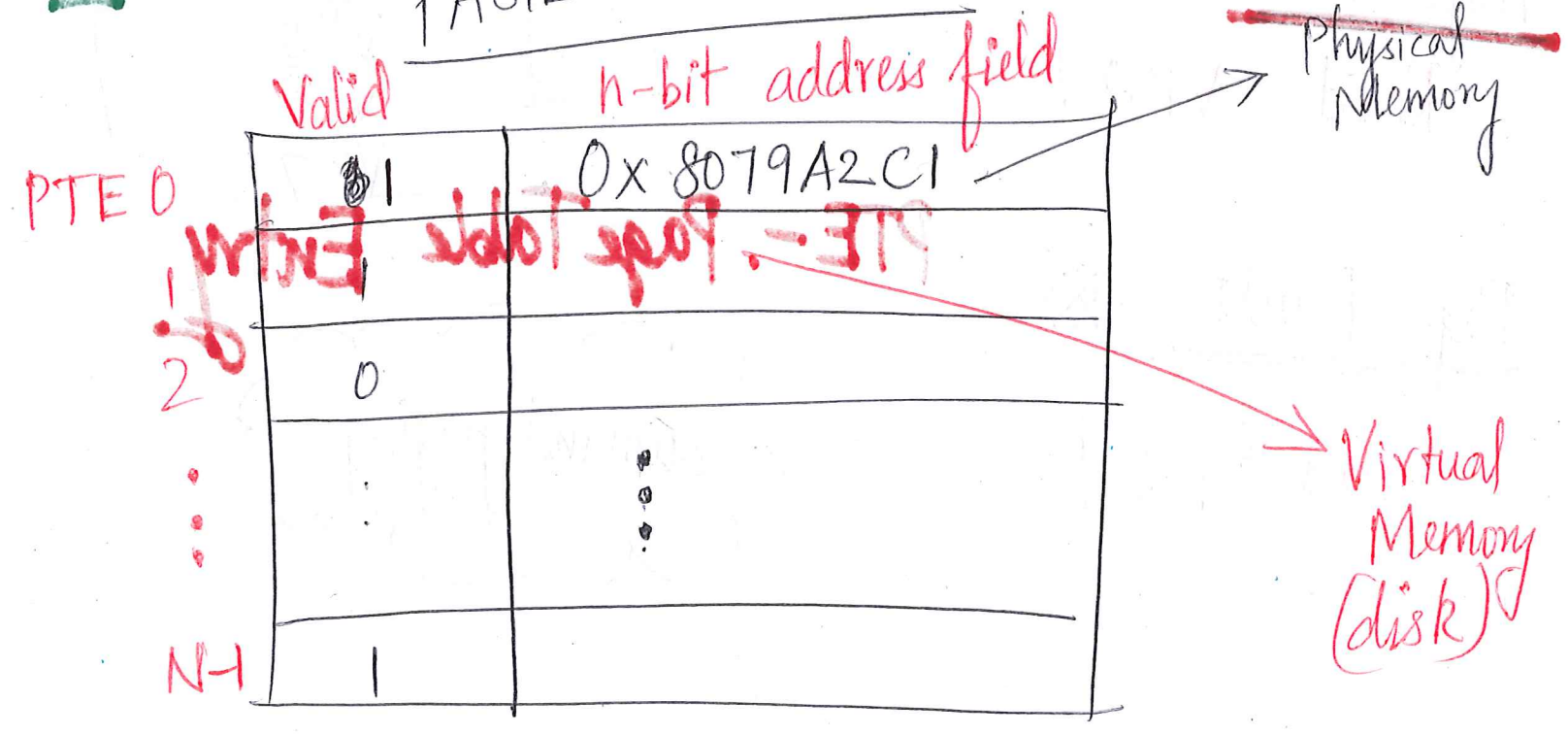
2. Assosiativity : fully associative!

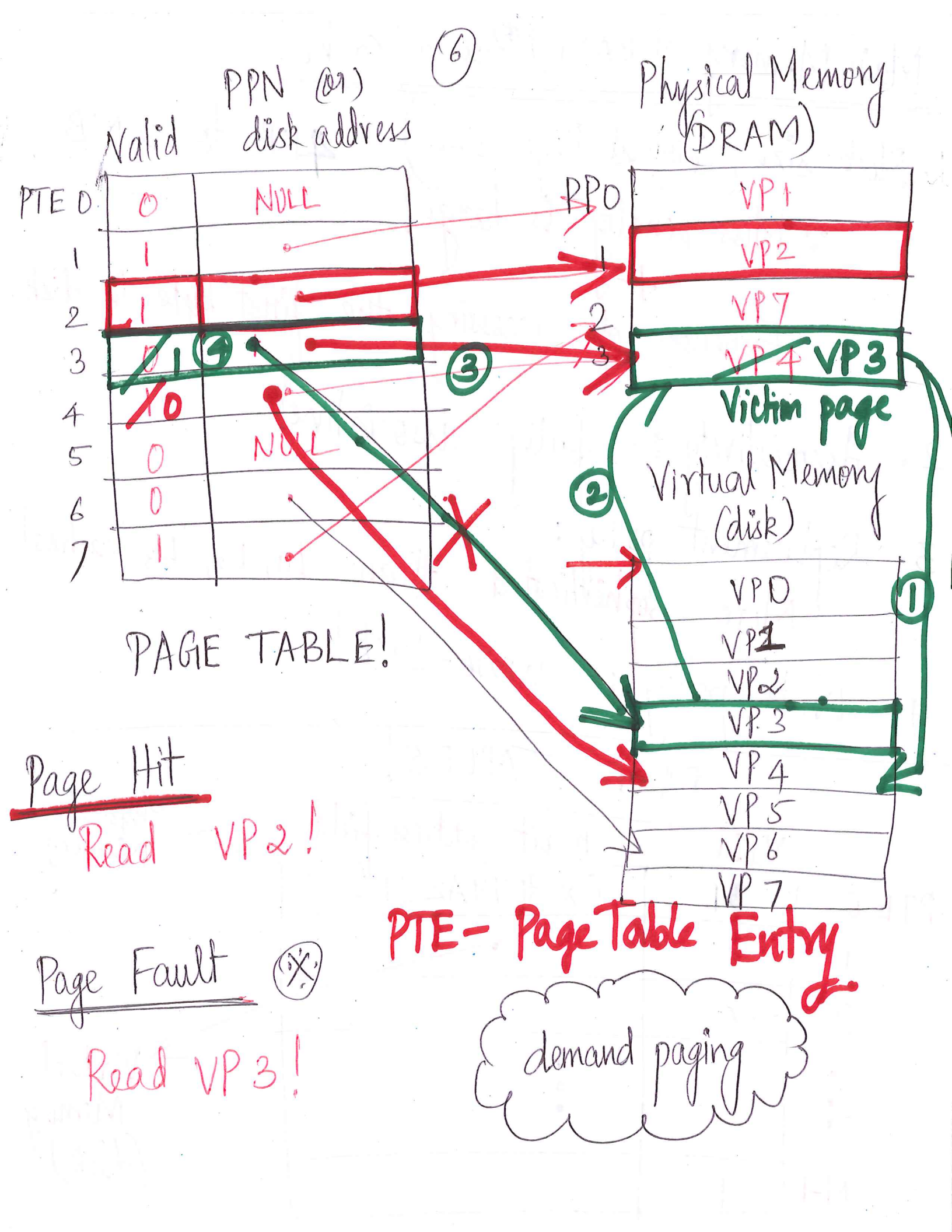
3. Replacement policy :

More sophisticated than L1, L2, L3 caches!

4. Write policy : write-back!

PAGE TABLES!

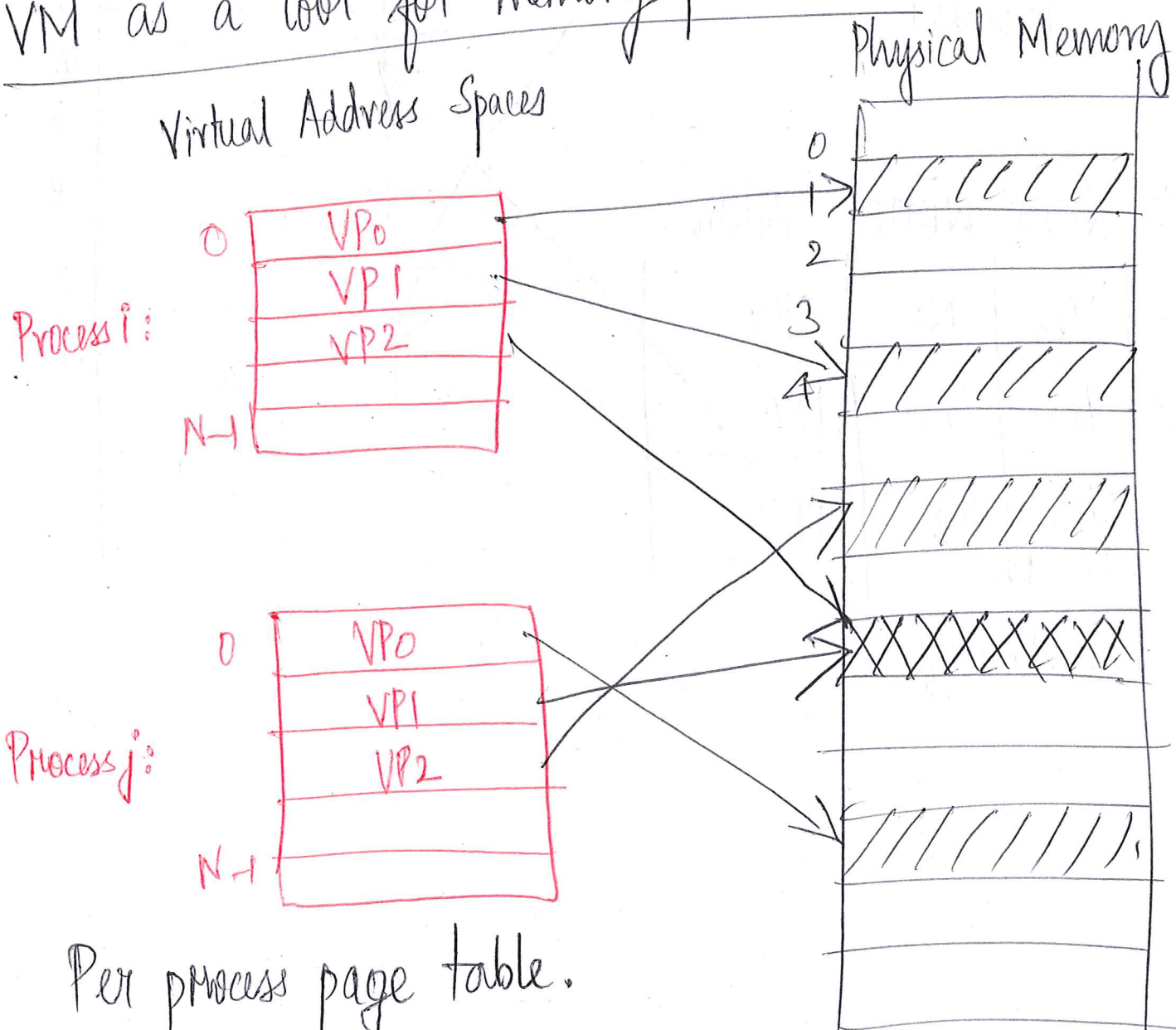



6
PPN (or) disk address
Valid
Physical Memory (DRAM)
PTE 0
1
2
3
4
5
6
7
0
1
1
0
1
0
0
1
NULL
NULL
PPO
1
2
3
VP1
VP2
VP7
VP4
VP3
Victim page
Virtual Memory (disk)
VP0
VP1
VP2
VP3
VP4
VP5
VP6
VP7
1
2
3
PAGE TABLE!
Page Hit
Read VP2!
Page Fault
X
Read VP3!
PTE - Page Table Entry
demand paging

VM as a tool for Memory Management

1. Simplifying linking
2. " loading
3. " sharing
4. " memory allocation.

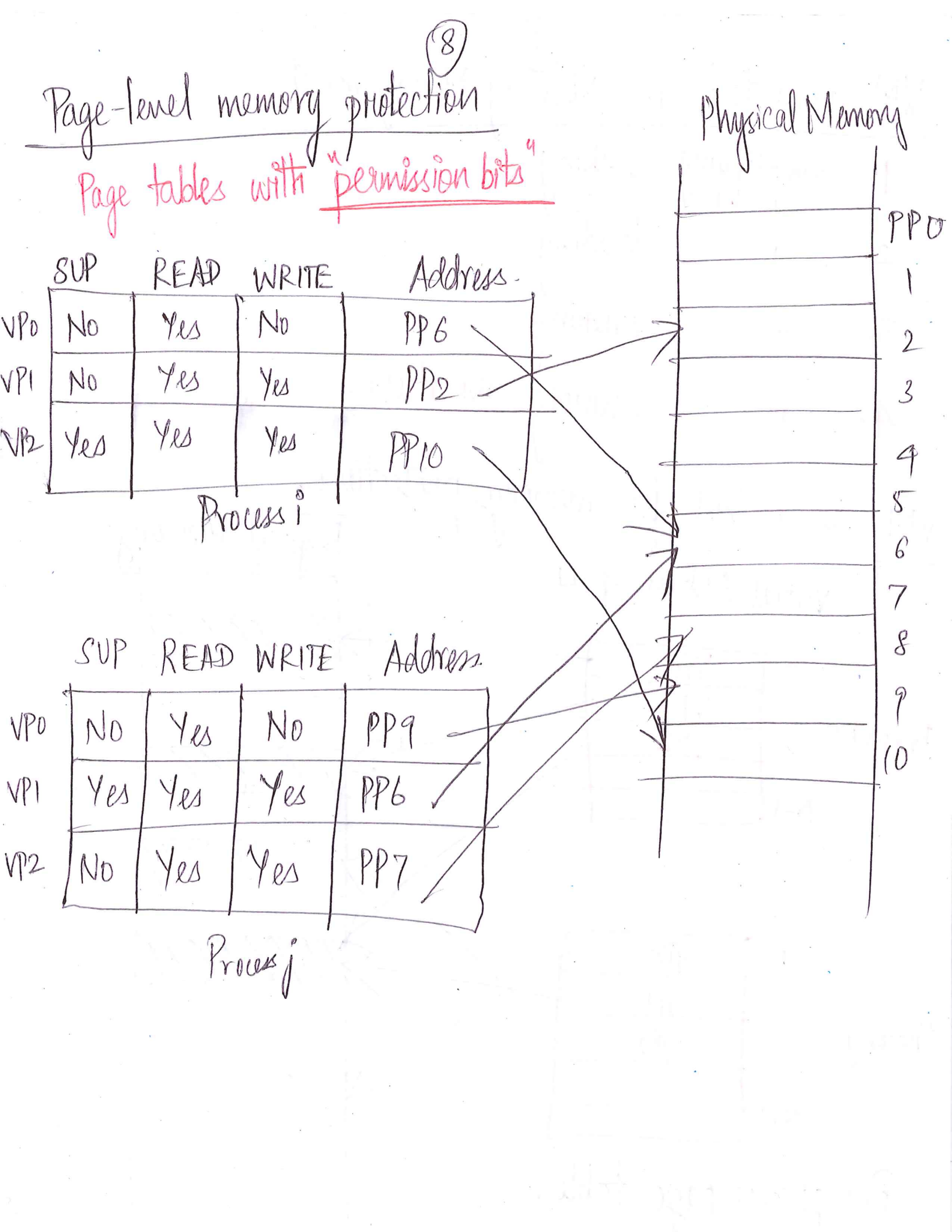
VM as a tool for memory protection.



Per process page table.

Page-level memory protection

Page tables with "permission bits"



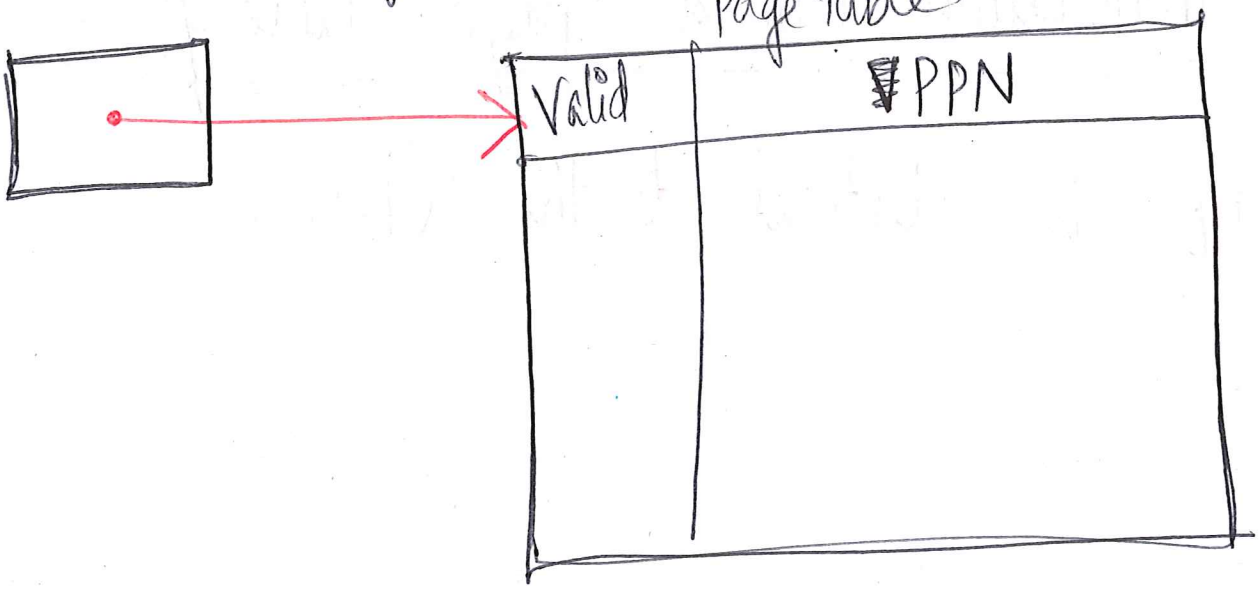
Address Translation

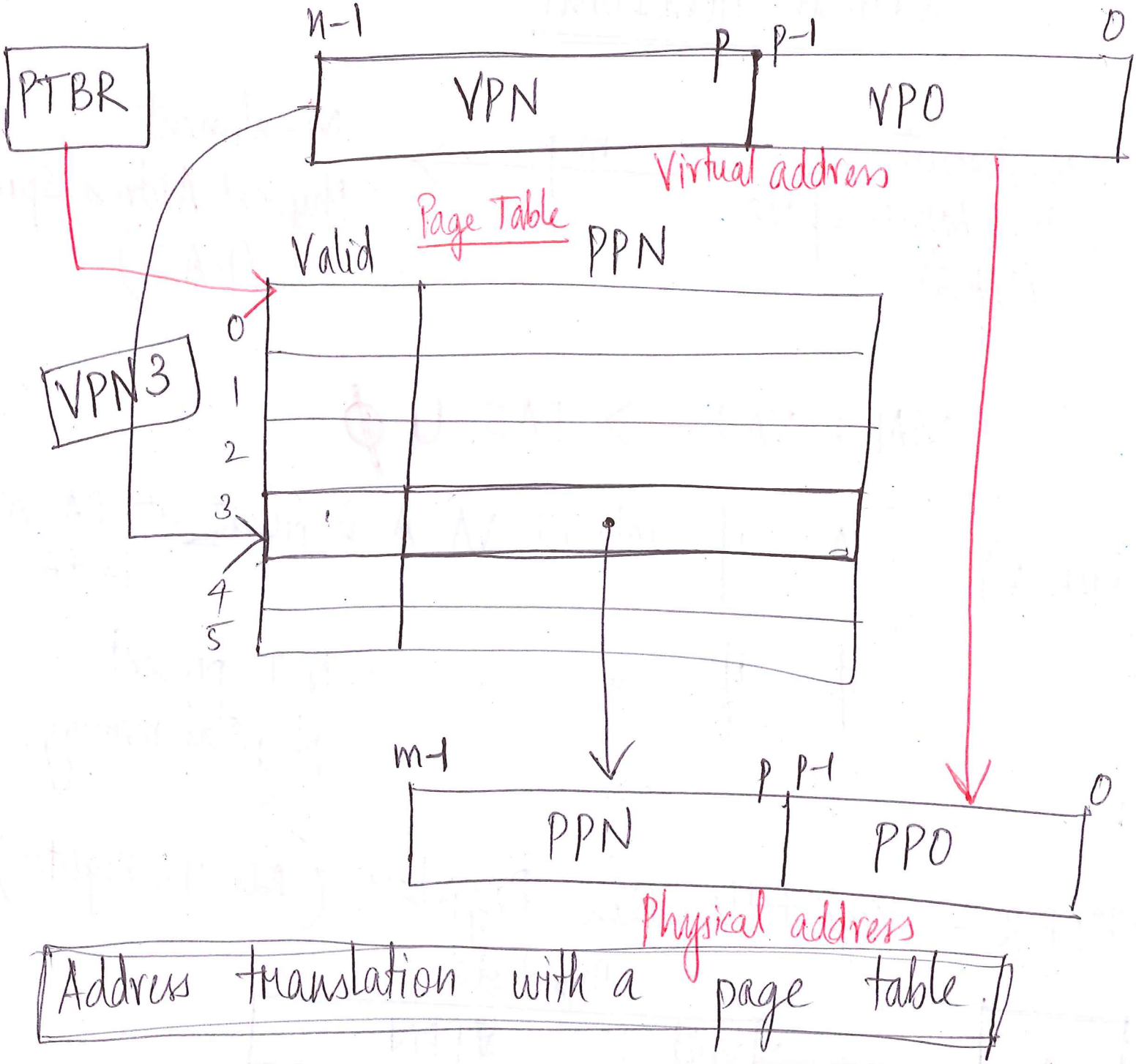
N-element Virtual Address Space (VAS) —map→ M-element Physical Address Space (PAS)

MAP : VAS → PAS ∪ ϕ

$$MAP(A) = \begin{cases} A' & \text{if data at VA } A \text{ is present at PA } A' \text{ in PAS.} \\ \phi & \text{if " " " NOT present in physical memory.} \end{cases}$$

PTBR - Page table Base Register. (CPU Register)

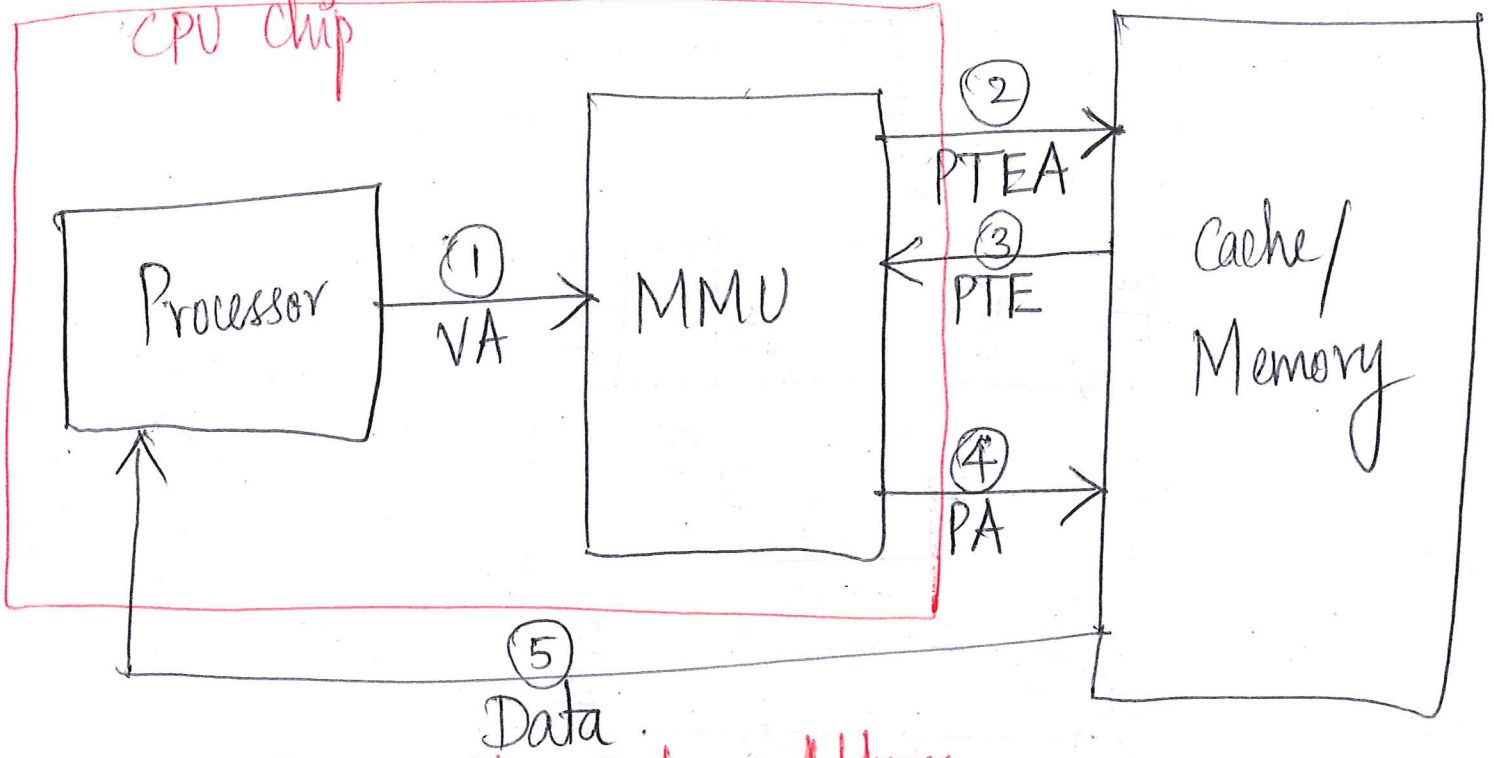




Address translation with a page table.

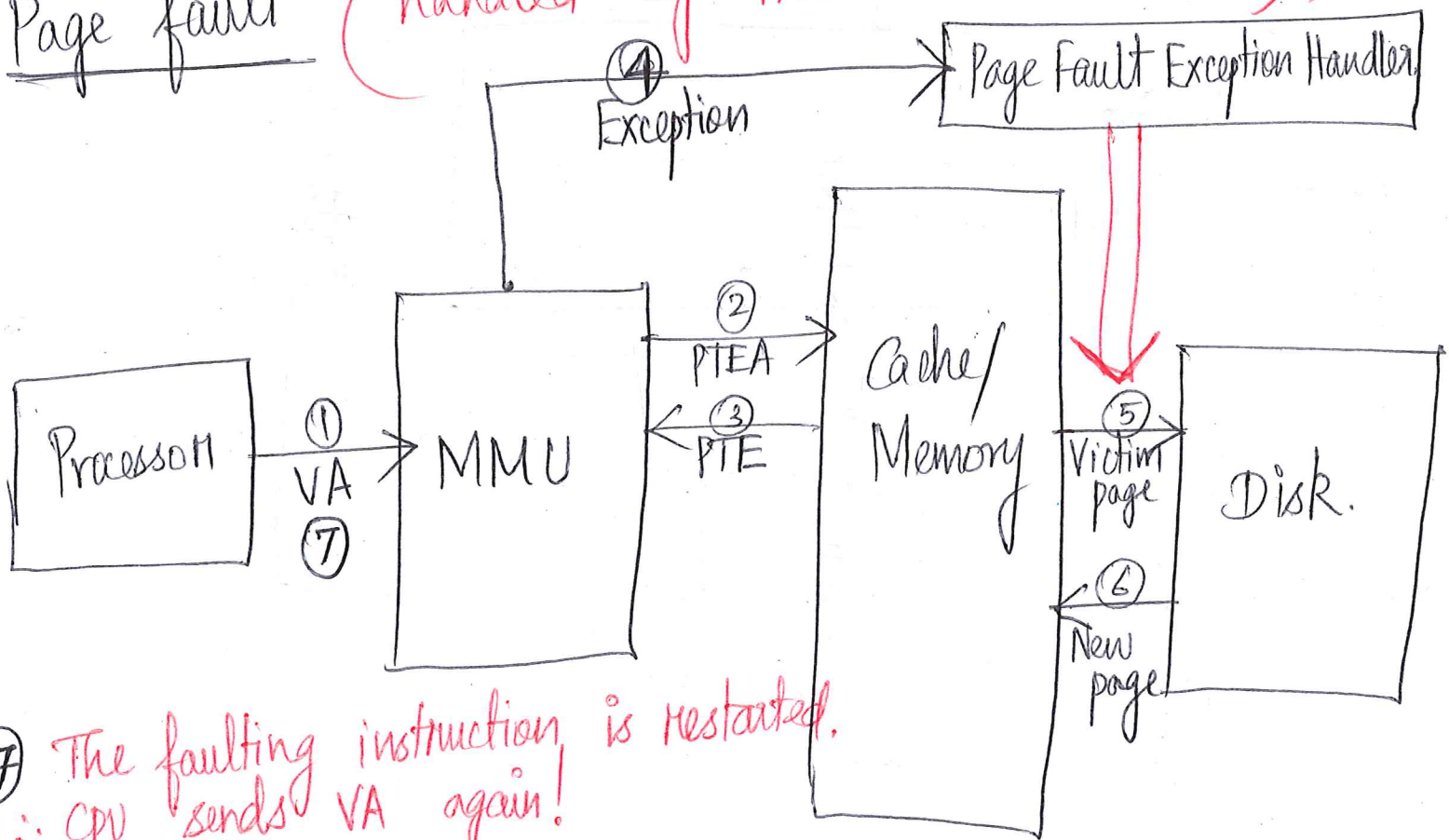
NOTE: PPO is identical to the VPO.

Page hit (handled entirely by the hardware) (11)



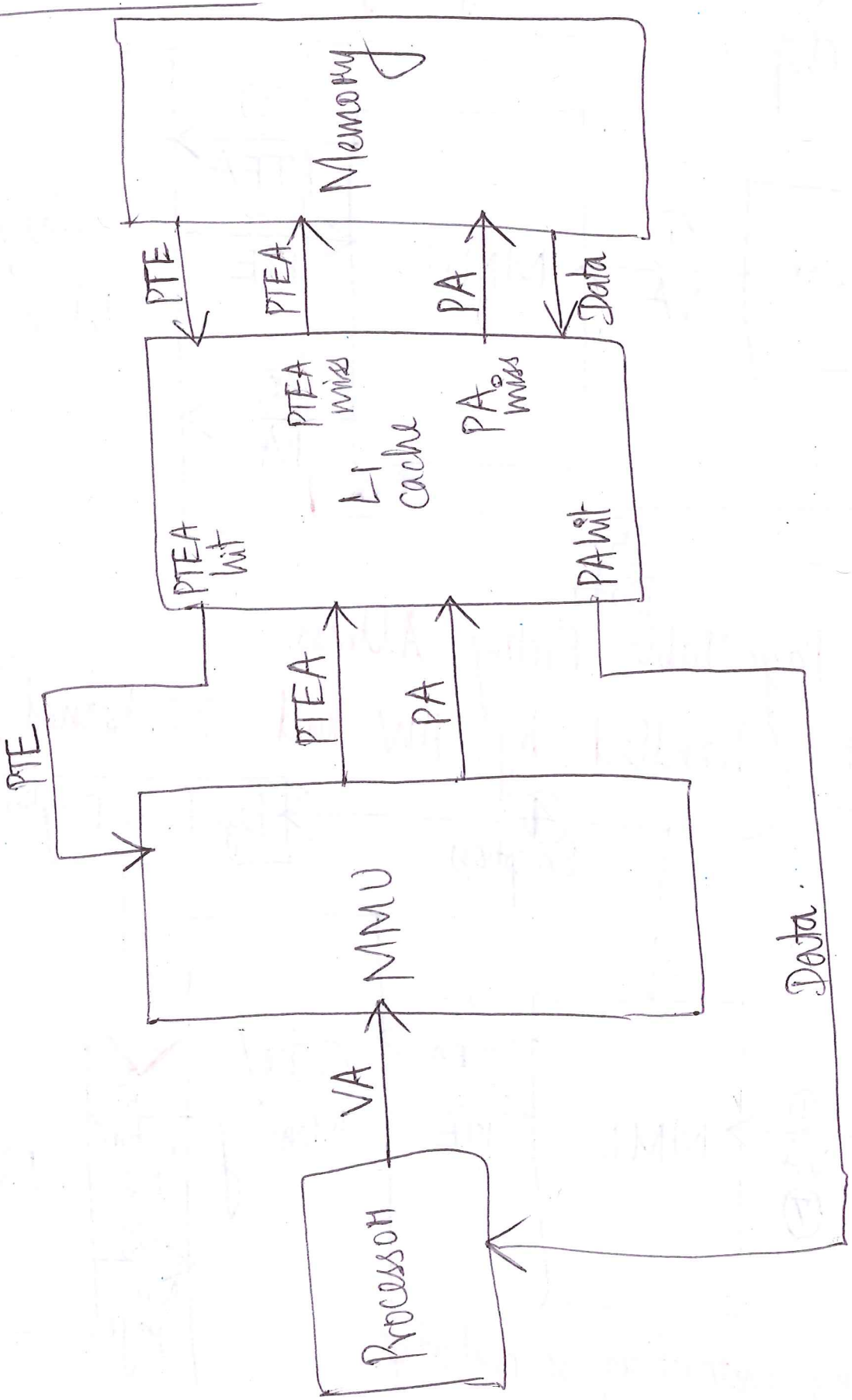
PTEA - Page Table Entry Address.

Page fault (handled by HW and OS Kernel).



(7) The faulting instruction is restarted.
∴ CPU sends VA again!

Caches + VM


Memory
PTE
PTEA
PA
Data
PTEA miss
PA miss
L1 cache
PTEA init
PA init
PTE
PTEA
PA
MMU
Data
VA
Processor