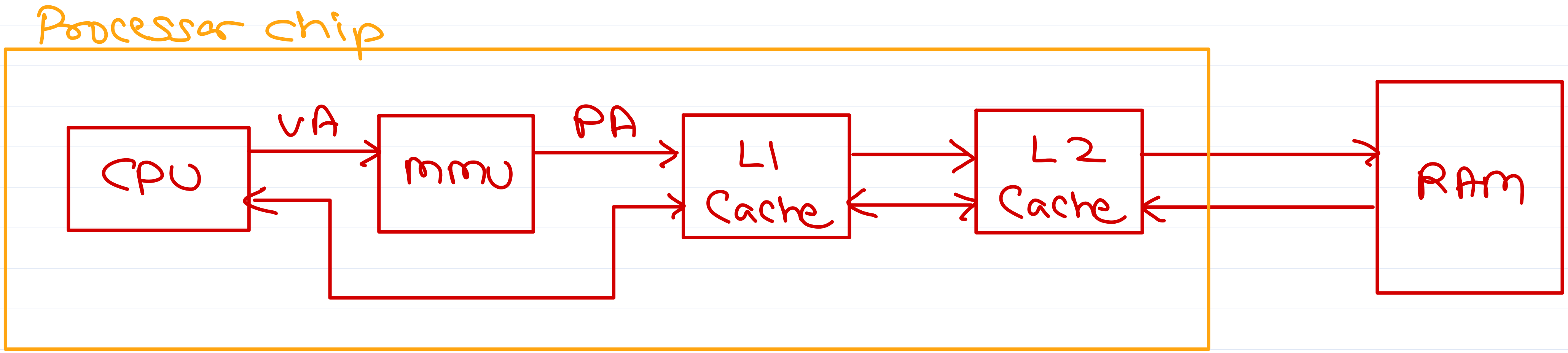
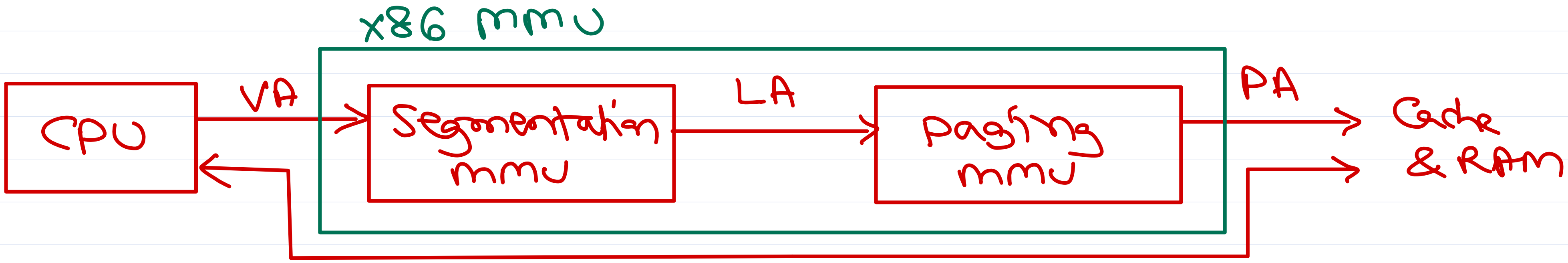
Embedded Operating Systems

Trainer: Nilesh Ghule


Processor chip
CPU
VA
MMU
PA
L1 Cache
L2 Cache
RAM


x86 MMU
CPU
VA
Segmentation MMU
LA
Paging MMU
PA
Cache & RAM

Virtual memory is the memory that can be given to a process.

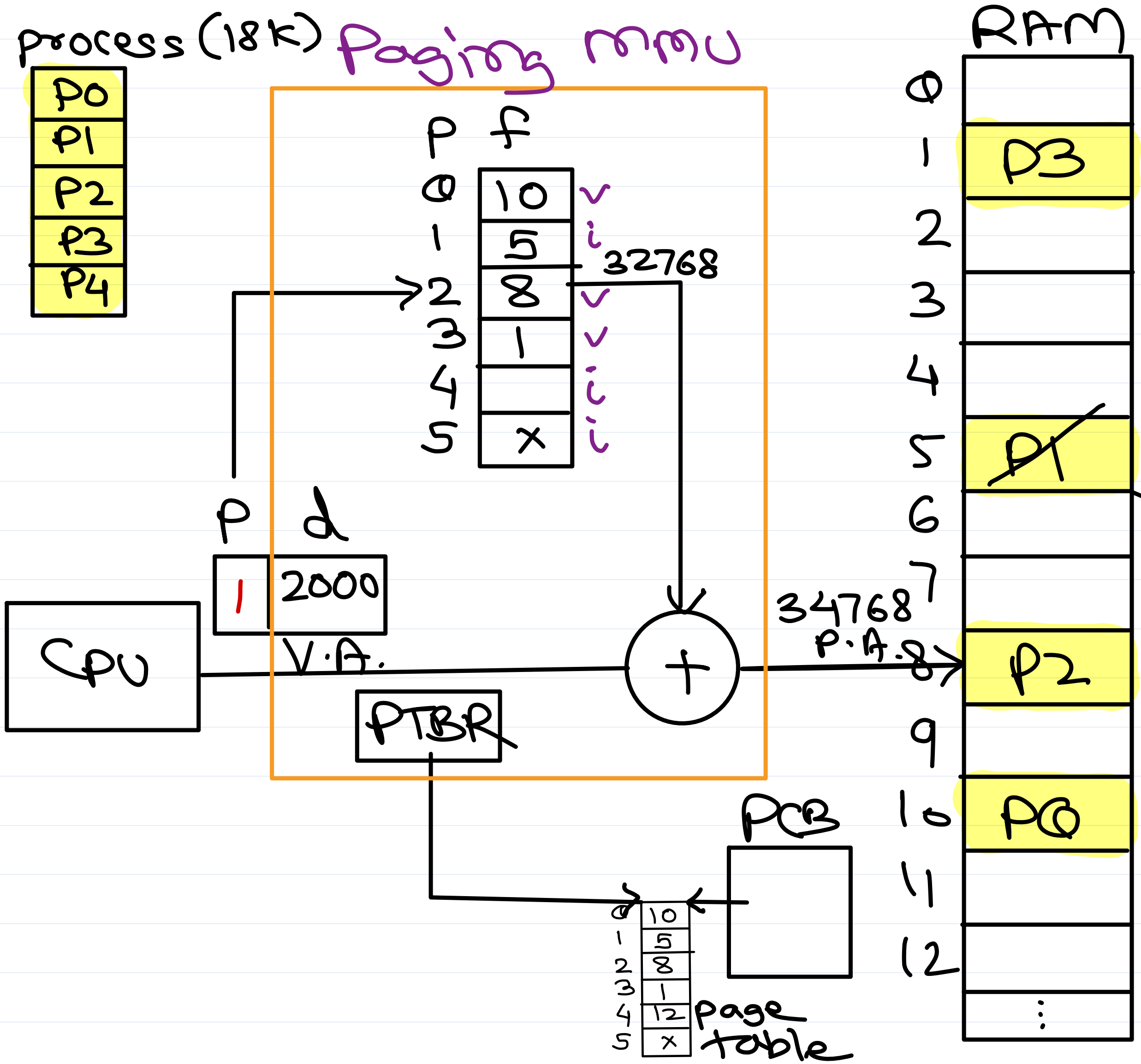
virtual memory = physical memory + Swap area − OS memory.
(RAM)

e.g. If RAM is 4GB and Swap is 2GB, then VM for a process = 4GB + 2GB − 1GB = 5GB.

↑ approx 1GB is needed for kernel.

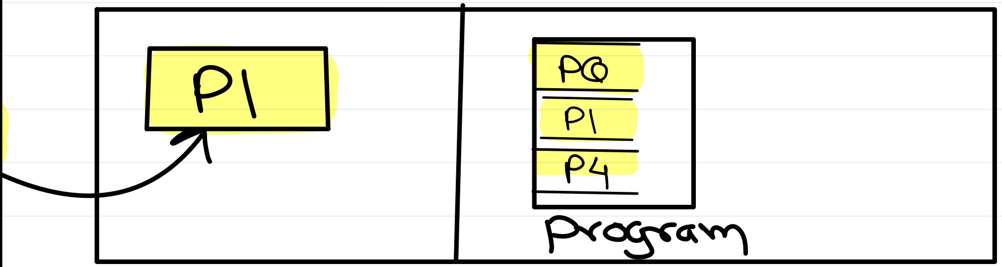
```
cmd> man 5 proc
              |→ overcommit_memory
              |→ overcommit_ratio
```

Page fault



When page requested by CPU is not present in main memory, then page fault will occur.

If page is in memory, then PTE is valid; otherwise PTE is invalid.



Page may not be in main mem:

1. page is not part of process's VAS.
2. page is not yet loaded.
3. page is swapped out.
4. page is not yet allocated.

When P.F. occurs, then OS's Page fault exception handler is executed

Page fault handling

① check if VA for which fault occured is valid (in VAS) or not. ← validity fault

if not, send SIGSEGV signal to process.

→ check from process's VAD list. If addr is not in range of any seg addr, then the addr is invalid.

② check if appropriate perms are available for requested page. ← protection fault

if not, send SIGSEGV signal to process.

→ check from process's VAD list. If seg perm are read only (r--) and write operation is done, then protection fault.

③ allocate an empty frame.

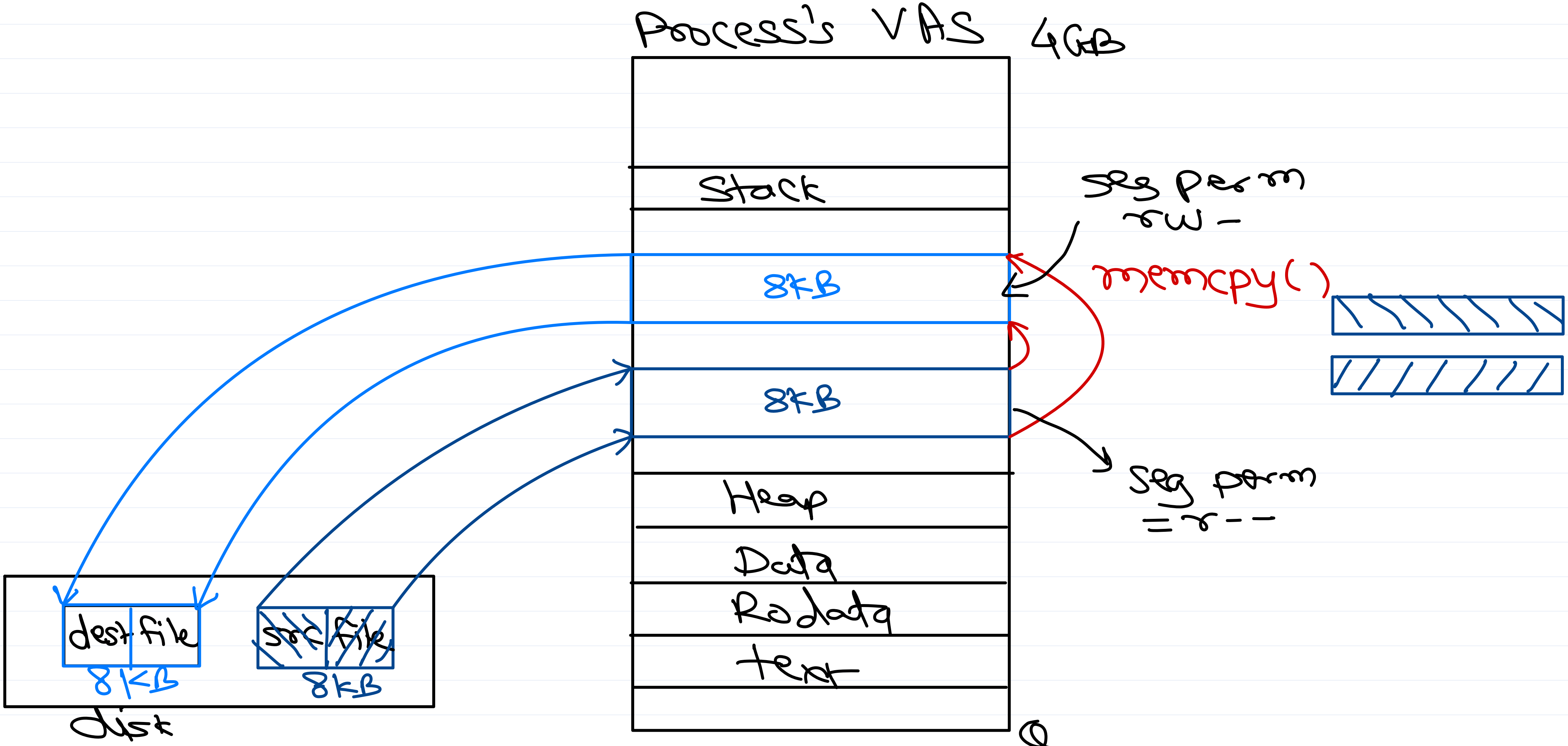
④ if page is on disk, then load it in that frame.

⑤ Modify Page table entry.

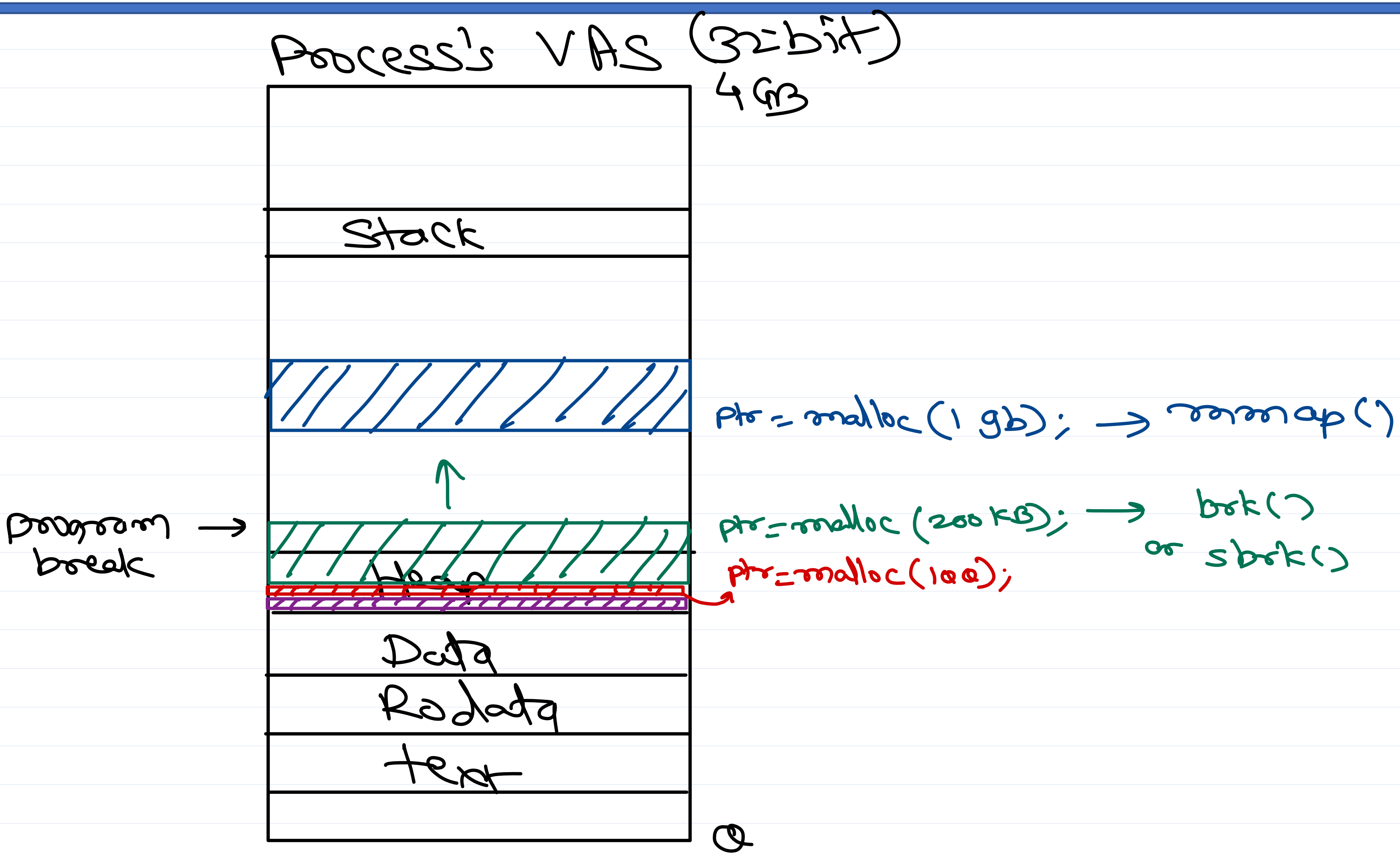
- frame address
- valid bit = 1

⑥ Restart the instruction at which page fault occurred.

Memory mapping

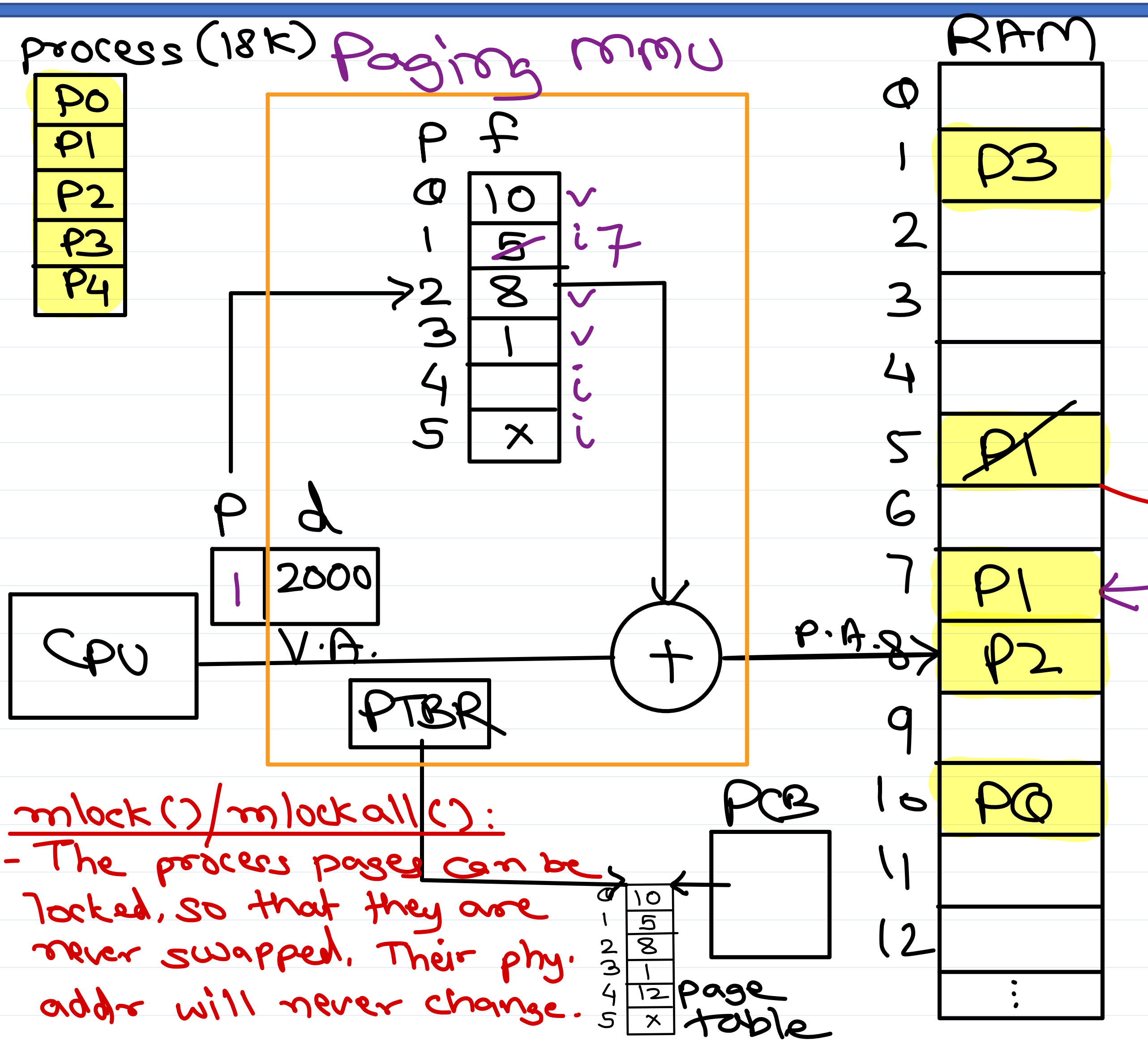


malloc()


Process's VAS (32-bit)
4 GB
Stack
ptr = malloc(1 gb); → mmap()
program break →
ptr=malloc(200 KB); → brk()
or sbrk()
Heap
ptr=malloc(100);
Data
Rodata
text
0

mlock()

- mlockall(MCL_CURRENT | MCL_FUTURE);

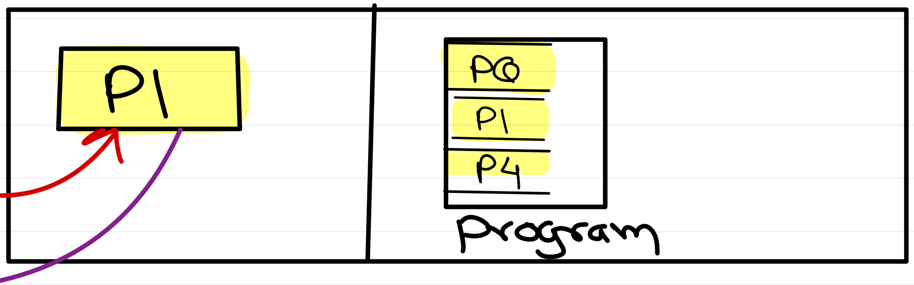


mlock()/mlockall():

- The process pages can be locked, so that they are never swapped. Their phy. addr will never change.

Virtual page

if process page & its frame addr may change (due to swapping).

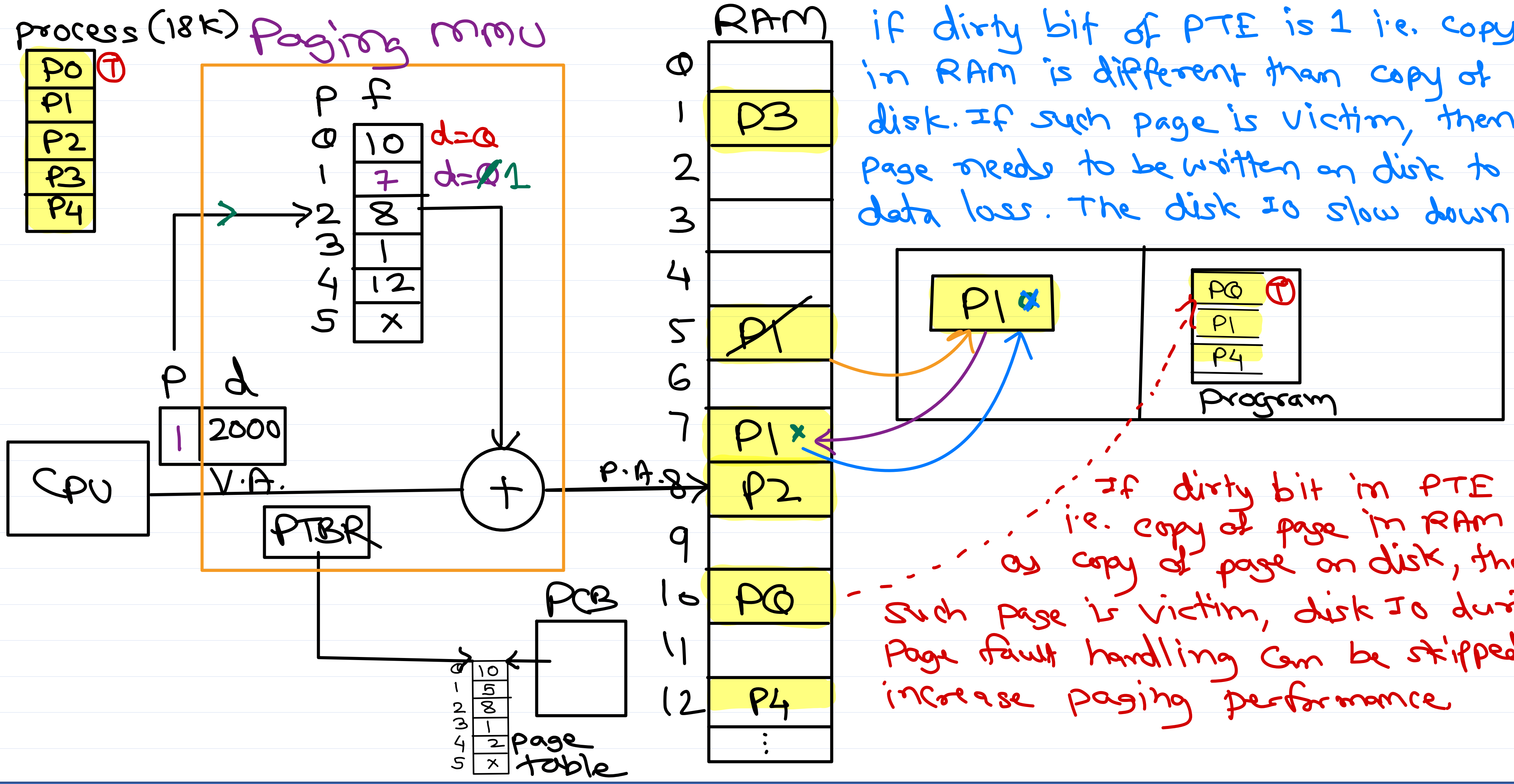


logical page

if process page & its frame addr never change. The page is never swapped (out & in).

Many kernel pages are logical i.e. never swapped out (e.g. pri page table, ...)

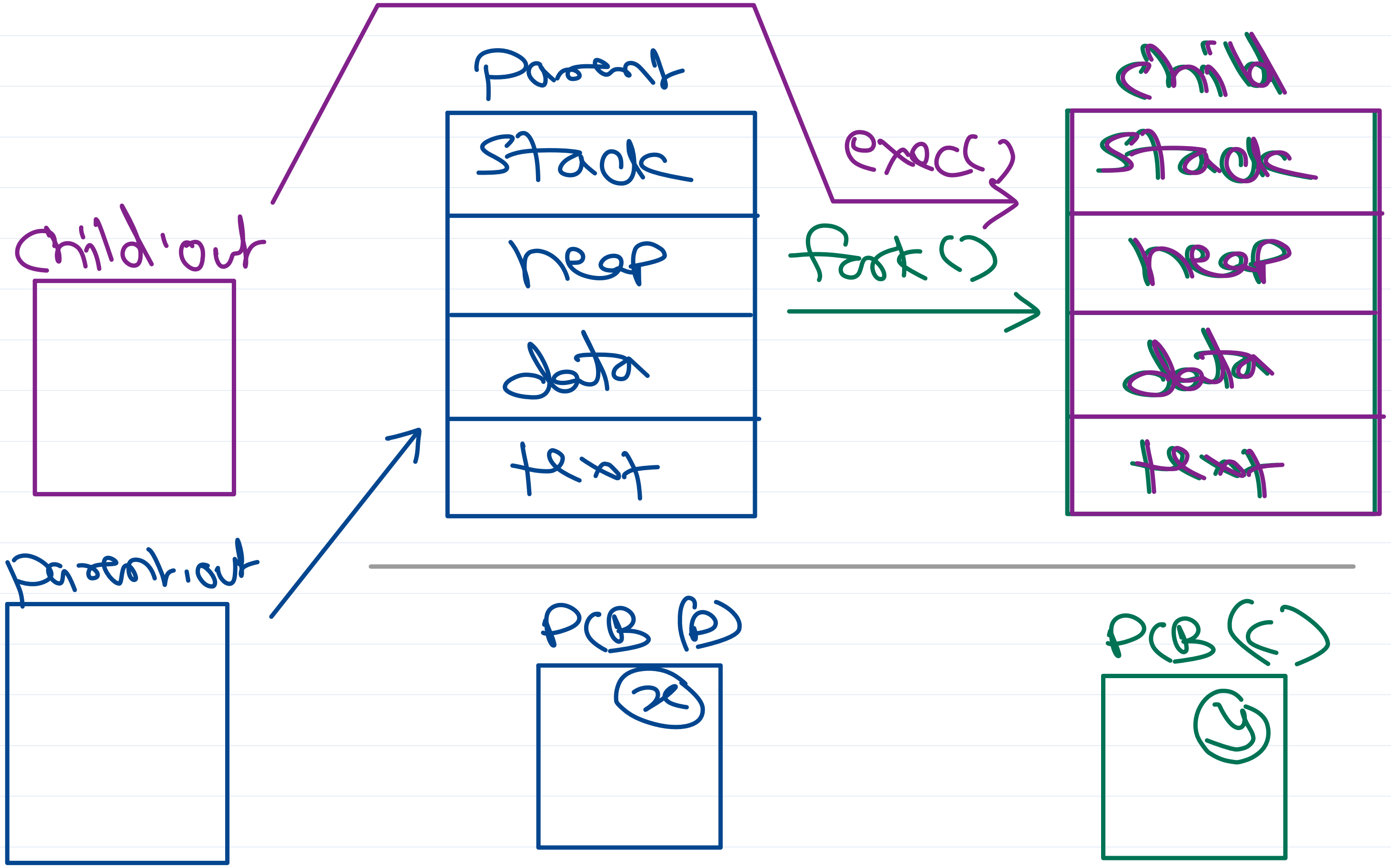
Dirty bit



if dirty bit of PTE is 1 i.e. copy of page in RAM is different than copy of page on disk. If such page is victim, then that page needs to be written on disk to avoid data loss. The disk IO slow down page fault handling.

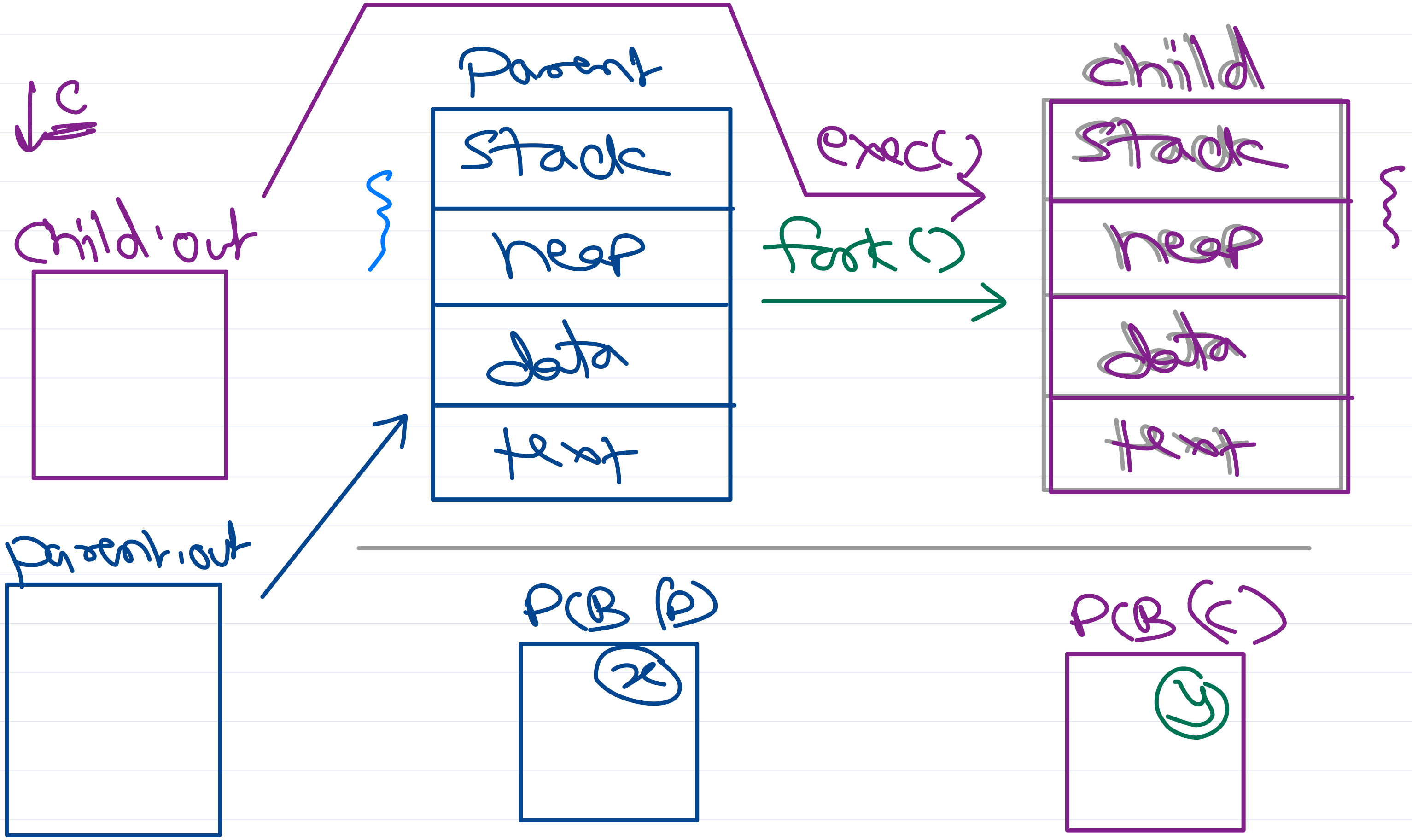
If dirty bit in PTE is zero i.e. copy of page in RAM is same as copy of page on disk, then if such page is victim, disk IO during page fault handling can be skipped to increase paging performance

```
ret = fork();
if (ret == 0) {
        exec("child.out", ...);
}
```

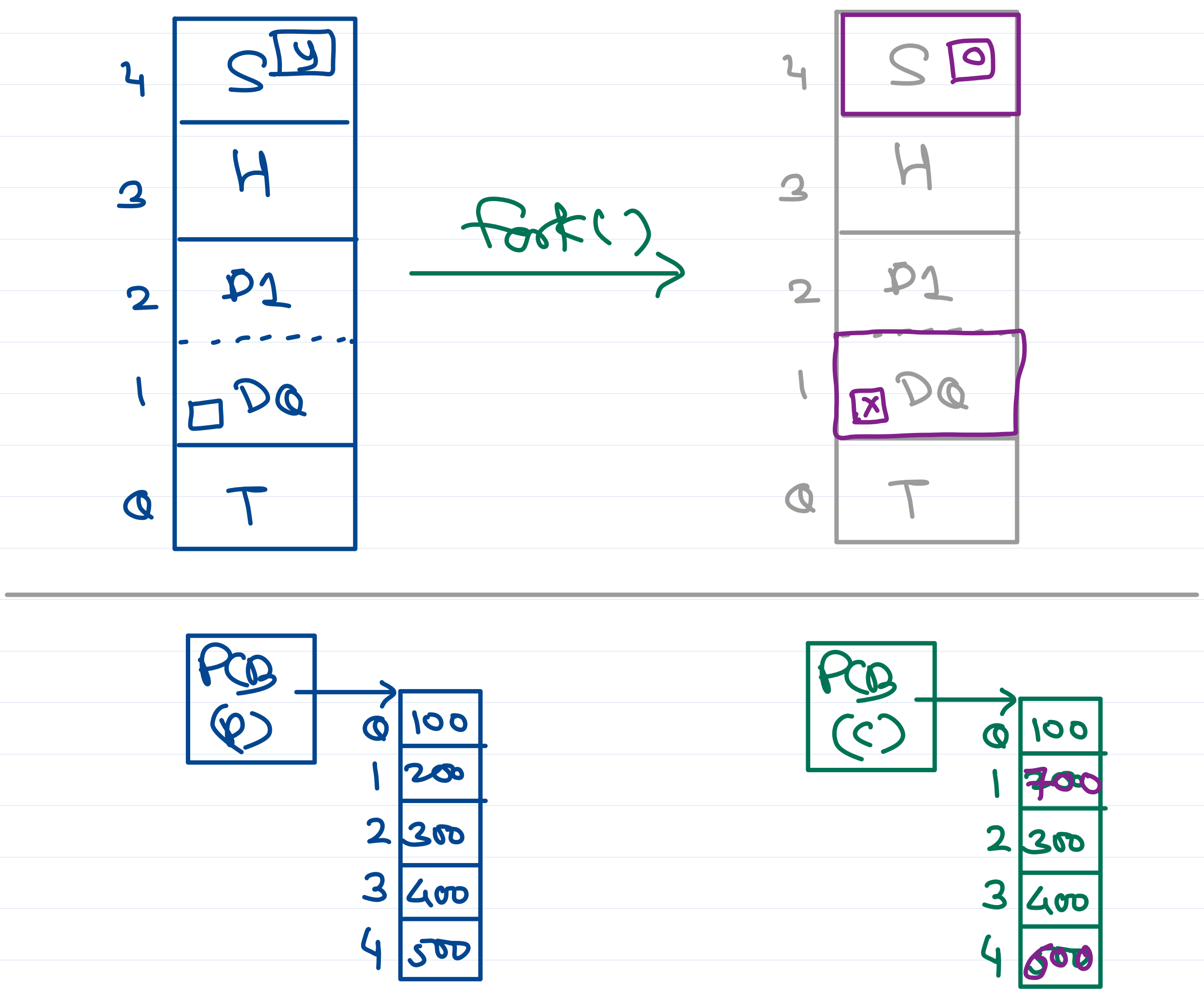


vfork()

```
P ↓ ret = vfork();
     if (ret == 0) {
P'   ↓ exec("child.out", ...); ↓C
     }
P ↓
```



Copy-on-write



```
ret = fork();
if(ret==0) {
    gvar++;
    =
}
```

SUNBEAM

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>