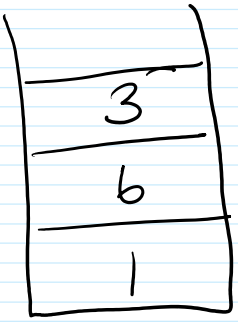
Alternative Page Replacement Policies

1. First in First Out

↳ Keep track of the order in which the pages came into memory.

Advantage: does not have to update stack when a page is reassessed.



Problem: Doesn't take into account principle of locality.

↳ no guarantee that the number of page faults will decrease if your memory size.

* Unlikely that you will see this *

2. Approximate LRU

↳ minimize amount of information that is maintained

↳ minimize the frequency at which information is updated.

Example → Add additional bit to page table.

→ Initialize all the bits to "0"

→ set the bit for a page to "1", when it's referenced.

→ replace one of the "0" pages.

→ Reset all bits to "0" once in a while.

↳ can do 2 bits, 3 bits, etc.

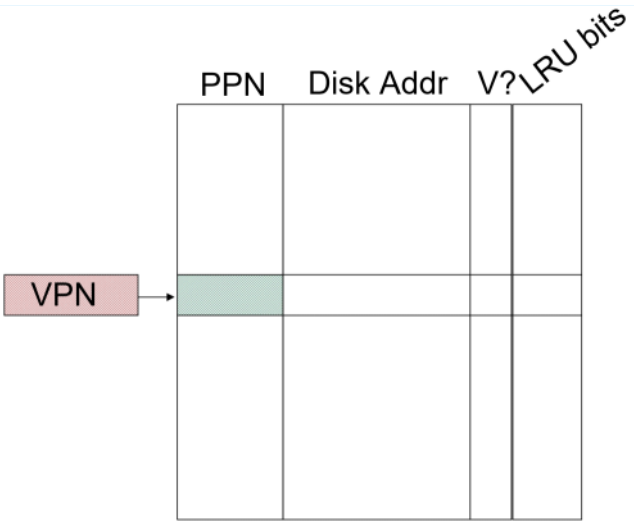
3. Random → randomly pick an "i" which is between 1 to n → replace page P_i

} works pretty well!

Page Table

PPN Disk Addr V? LRU bits

Page Table



Recall: Page Fault Handler

① Identify a slot in the main memory to be used.

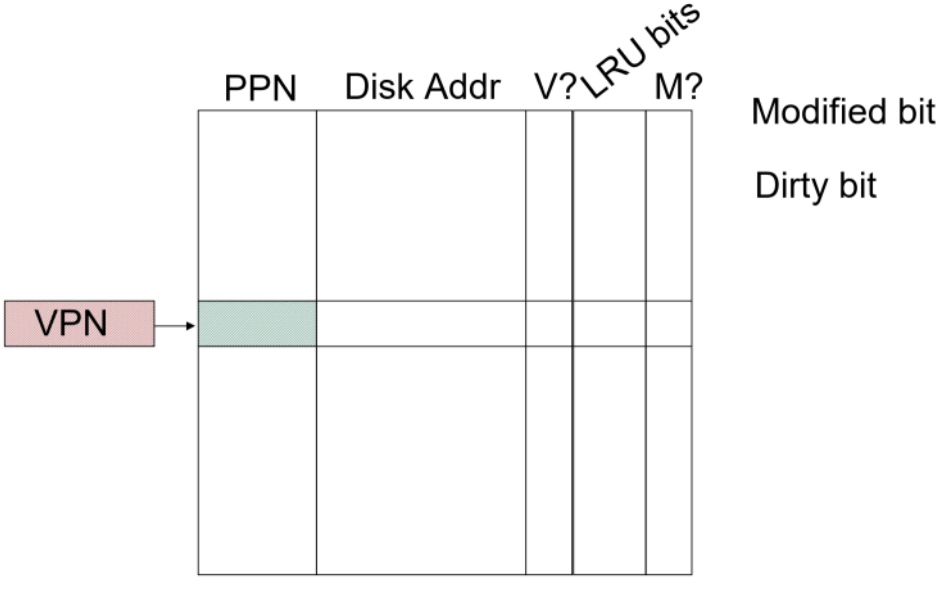
② Get page contents from disk

③ Update page table entry

Problem : The page identified by the page replacement policy might have been modified while it was in the main memory.

↳ It cannot be overwritten by the incoming page.

↳ first should be copied back to disk



Page Size

How big is a page in practice?

⇒ Trade off → larger page size

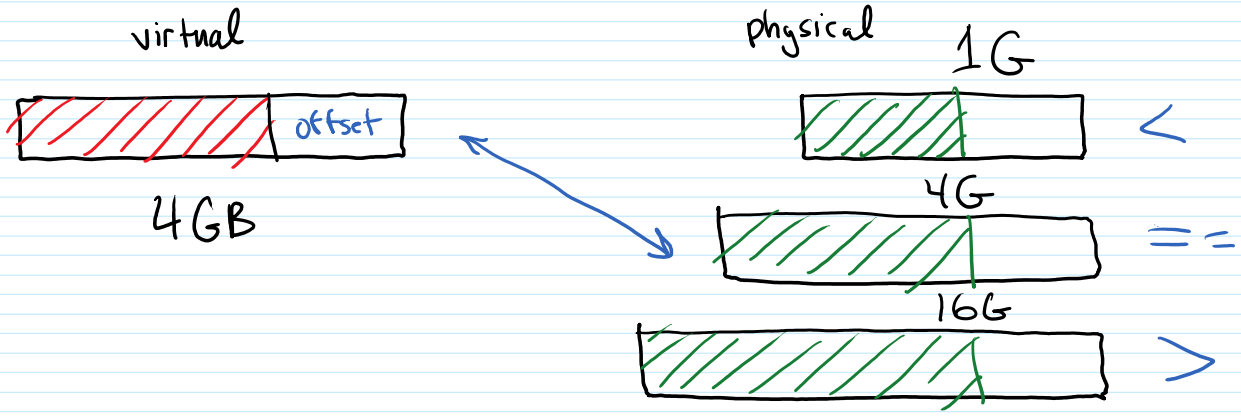
↳ the smaller page table.

↳ more unused memory space within a page.

⇒ Unit of transfer to hard disk → typically 512B (disk sector)

⇒ typical page size ~ 4kB

Virtual Space Size vs Physical Space Size



- CPU management