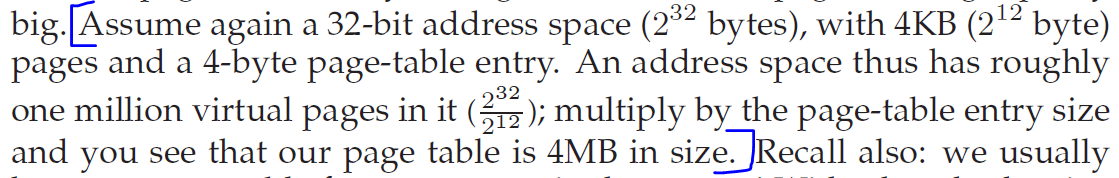
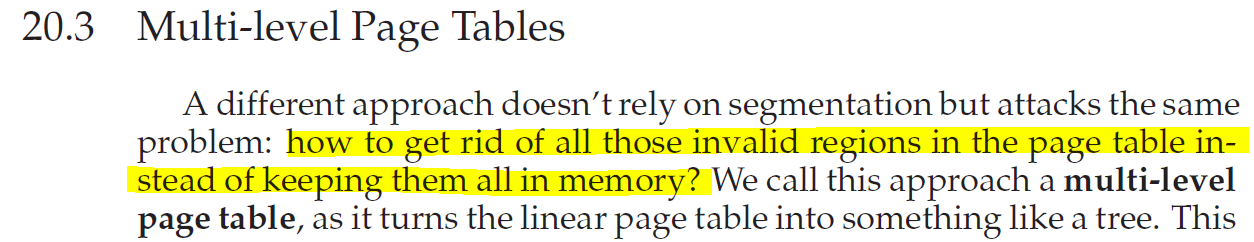
Chapter 9 Paging - smaller tables

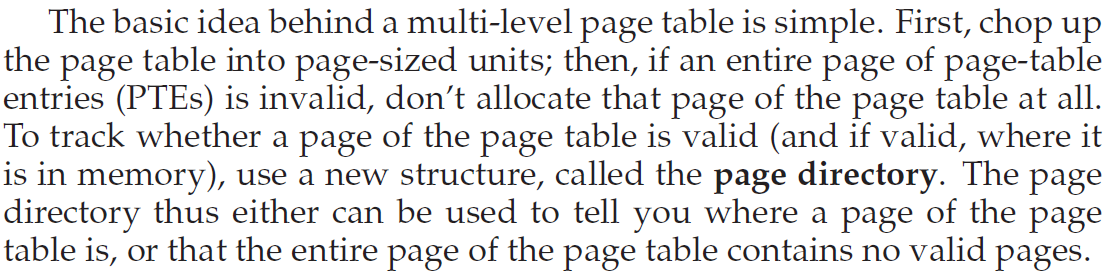
The problem with paging is that per-process page tables get too big.

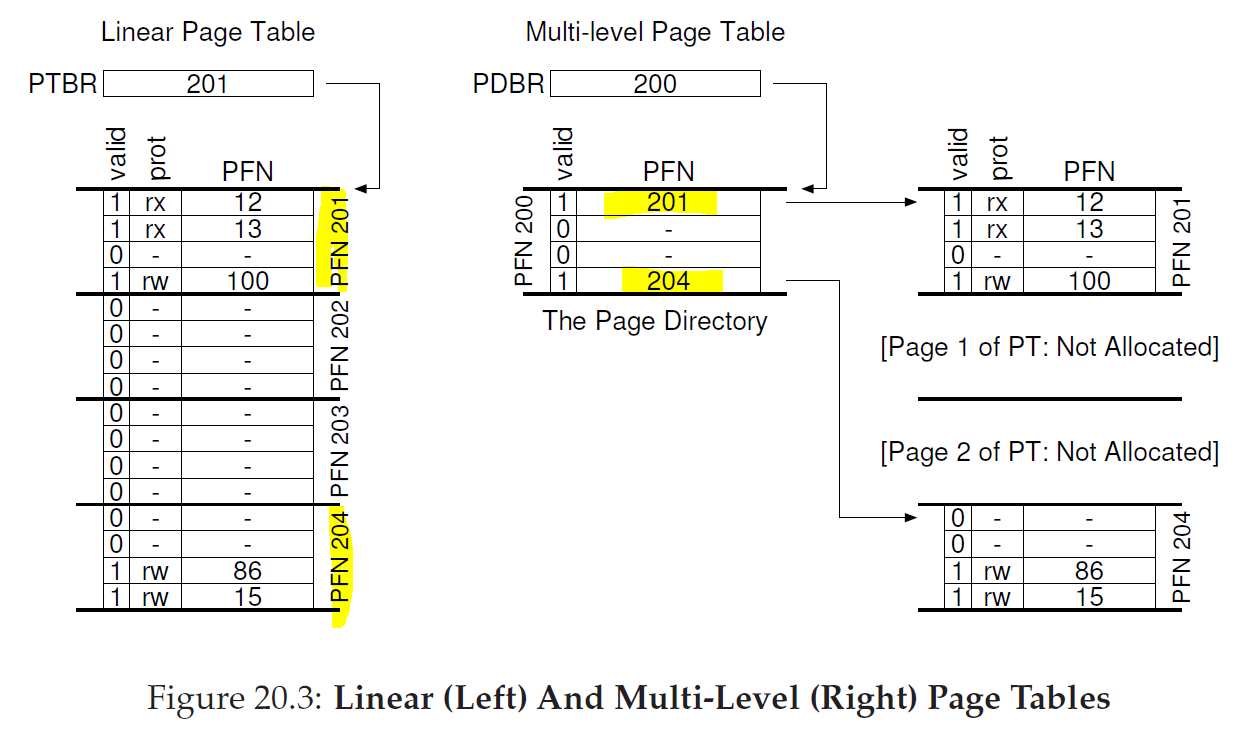


4KB pages means 12-bit offset and 20-bit VPN. 0xfffff = 1048575 bytes and if each PTE takes up 4 bytes, then the PT per process will be 4x1048575 = 4MB!

We could increase the page size to reduce the number of bits for the VPN, but that’s not a good solution - so we need another solution (ignore the hybrid approach).







Question – How many entries go in the PD (page directory)? Assume the following.

* 16 KB address space
* 64-byte page
* 8-bit VPN
* 6-bit offset
* Each PTE is 4 bytes
* Each PDE is ?? bytes (they don’t specify)
* A two-level page table

An 8-bit VPN means we need 2^8 = 256 PTEs meaning a PT size will be 4x256 = 1024 bytes, and if we divide that by the page size (64), we get 16. So, there will be 16 entries (PDEs) in the PD representing the PT for *this process.* If PDEs is 16 this means we need 4 bits to address them (2^4 = 16). Hence the *first* 4 bits out of the 8-bit VPN will be used for the PD index (1), while the remaining 4-bits will be used as an offset that will be appended to the PDE.PFN (which is our *index into* the PTE (2)) – *assuming that PDE is valid*.

In the homework, I assume the following:

* 32 KB address space
* 32-byte page
* 10-bit VPN
* 5-bit offset
* Each PTE is 1 byte (I assumed)
* Each PDE is also 1 byte (I assumed)
* A two-level page table

For this homework, see C:\Users\Alifa\Desktop\lc3-backup\eclipse\Ostep\src\chapter\MLPagingPD.java

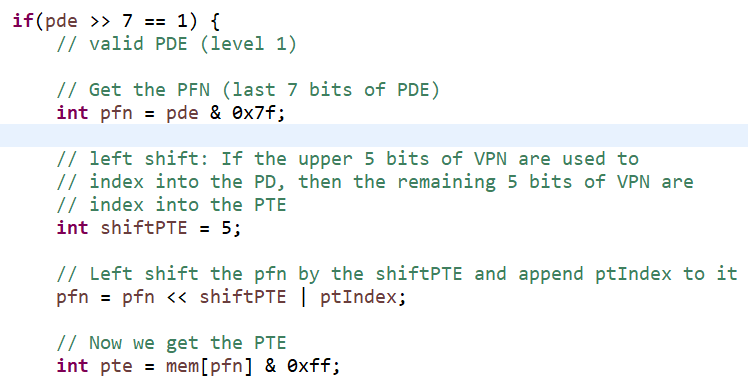
Hence, a 10-bit VPN means 1024 PTEs and if each PTE is 1 byte then dividing 1024 by 32 gives us 32 PDEs. Hence, we need 5 bits from the VPN as an index into the PD (1) while the remaining 5 bits of the VPN will be our offset which we will append to the PDE.PFN (being our index into the PTE – assuming that PDE is valid of course) (2).

(1)



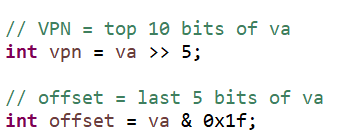


(2)

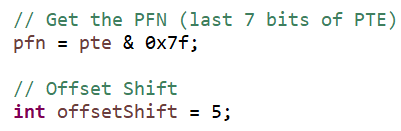


Now that we are in the PTE, we extract the PFN (4) (again) and append the offset (out of the VA we got earlier (3) and derive our Physical address (PA) (5)

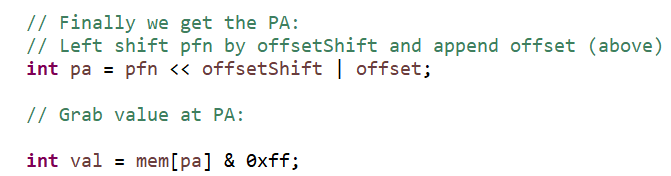
(3)

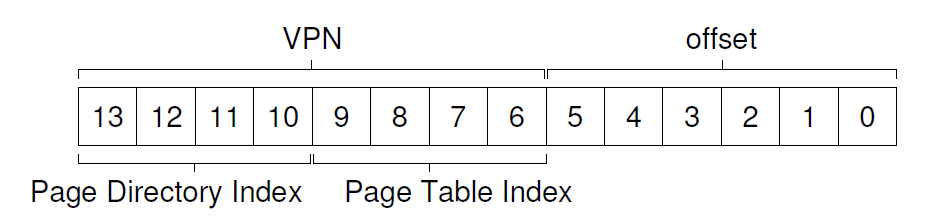


(4)



(5)





The following traces their example to convert VA 0x3f80.

