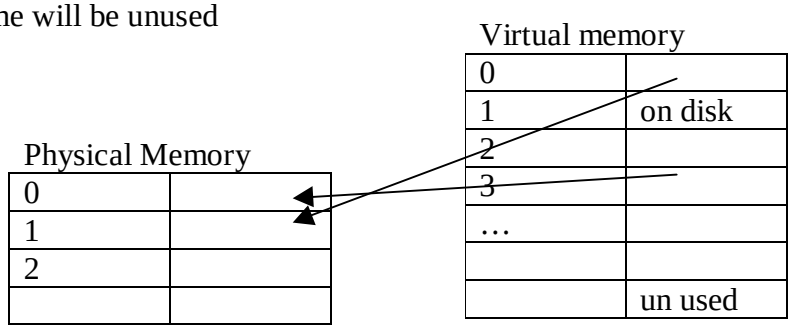
Lecture 29

July 18

Virtual Memory

- Modern PCs can address 2^{32} bytes = 4GB of memory
    - If we allow some "memory" to be stored on disk, there will be more space to use
- The larger "virtual memory could have parts in memory & other parts on disk
    - All of this should be invisible to the program
    - When a program does a memory read, it doesn't know if the data is even in memory
    - If its not in memory, it must be moved from disk into memory
        - Not necessarily in the same location every time
    - Every byte in the address space is given a "virtual address"
        - $0\text{-}2^{32}\text{-}1$ in a 21 bit arch.
        - Programs will use the virtual address
        - His will be translated to a "physical address"
            - If its in memory
- The virtual address space is broken into "pages"
    - E.g. with 4 KB pages
        - 4 GB – 2^{20} ~ 1 million pages
    - some of the pages will be in memory at a given time
    - some will be on the disk
    - some will be unused



- the space on disk that holds the VM ages is called the “swap space”
    - this is managed by the OS
    - the CPU doesn’t do disk access directly
    - the amount & location of the swap space is determined by the OS
    - the transfer memory <-> disk & keeping track of pages on disk is done by the OS
    - every time memory is accessed there must be a virtual → physical address translation
        - can’t be done by the OS (too slow)
        - need hardware support for this
        - the “memory Management Unit” (MMU)
    - the MMU’s job is to manage CPU <--> communication
        - virtual → physical translation
        - if the virtual address is m RAM, read & give the data back
        - if not, send a “page fault” to the OS
            - the OS is activated & must swap the required page into RAM
            - might have to move something else out to make room
        - once its in memory, the MMU can retrieve the data for the CPU