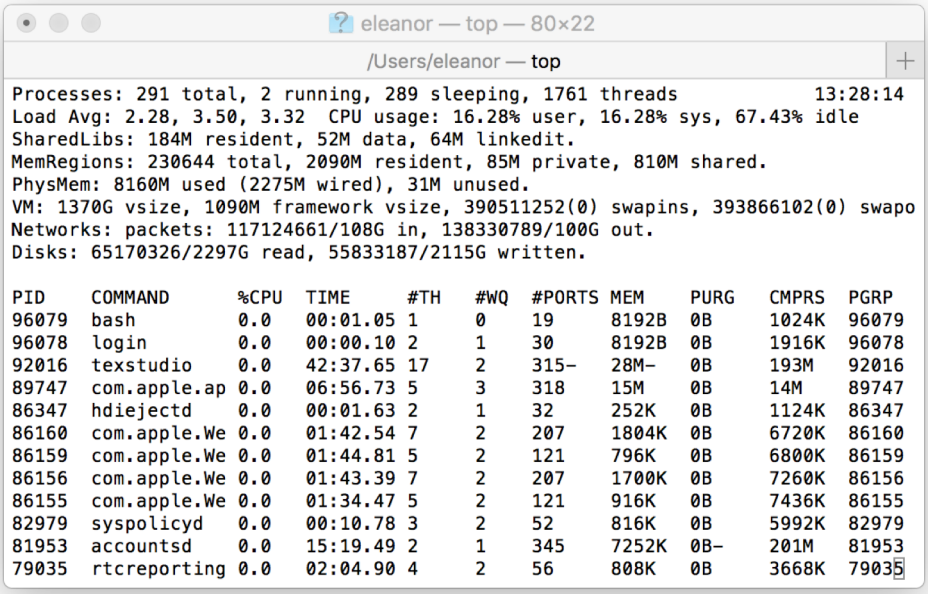
Lecture 15: Virtual Memory

CS 105

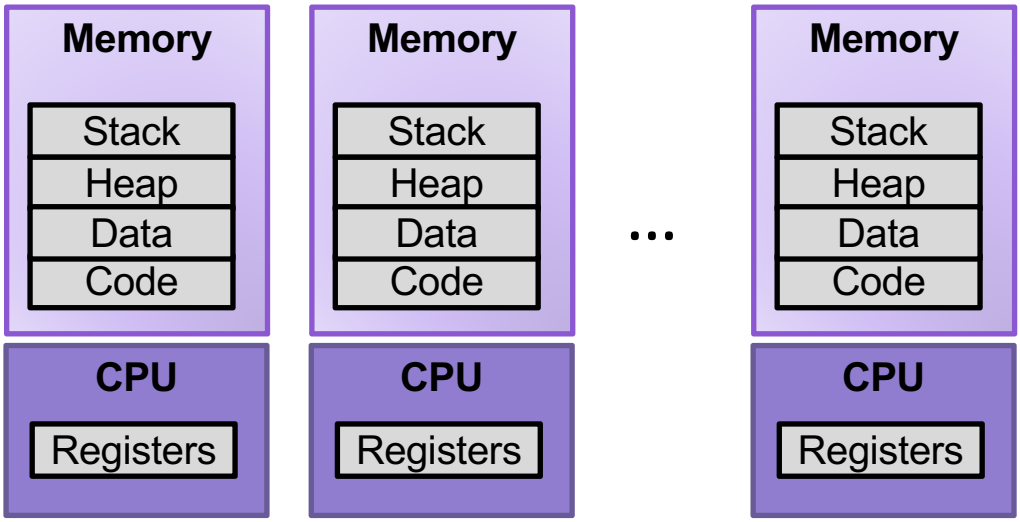
October 29, 2019

Multiprocessing

- Computer runs many processes simultaneously
- Running program “top” on Mac
  - System has 123 processes, 5 of which are active
  - Identified by Process ID (PID)

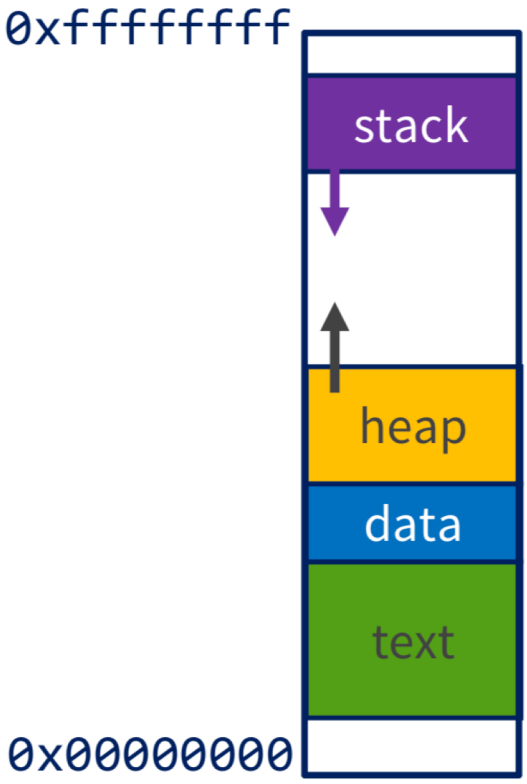


Multiprocessing: The Illusion



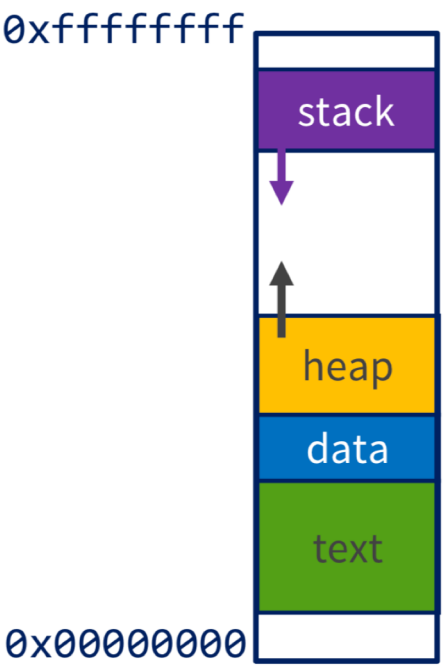
- Process provides each program with two key abstractions:
  - ***Logical control flow***
    - Each program seems to have exclusive use of the CPU
    - Provided by kernel mechanism called ***context switching***
  - ***Private address space***
    - Each program seems to have exclusive use of main memory.
    - Provided by kernel mechanism called ***virtual memory***

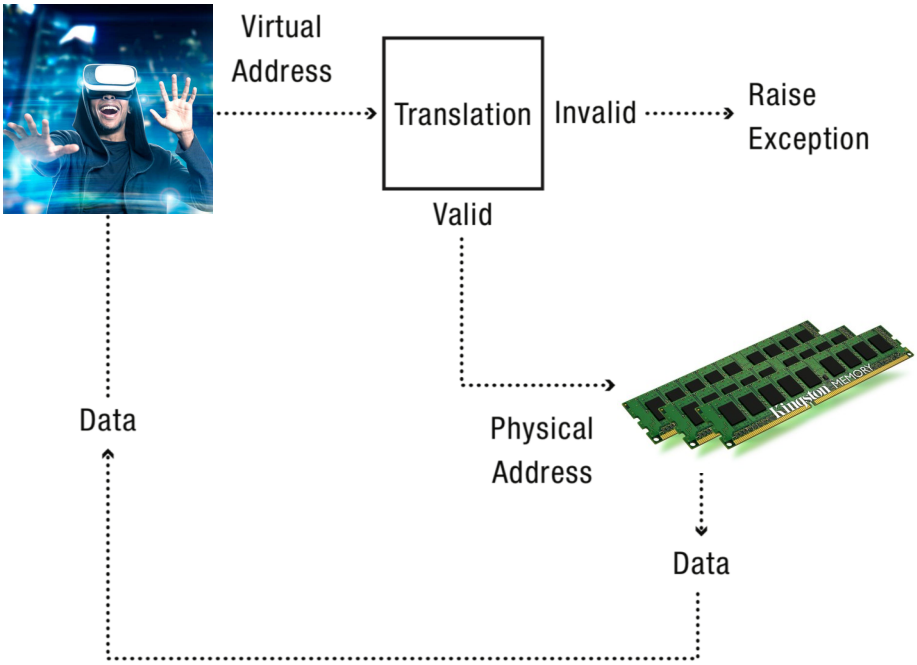
Virtual Memory Goals



- **Isolation:** don't want different process states collided in physical memory
- **Utilization:** want best use of limited resource
- **Virtualization:** want to create illusion of more resources
- **Efficiency:** want fast reads/writes to memory
- **Sharing:** want option to overlap for communication

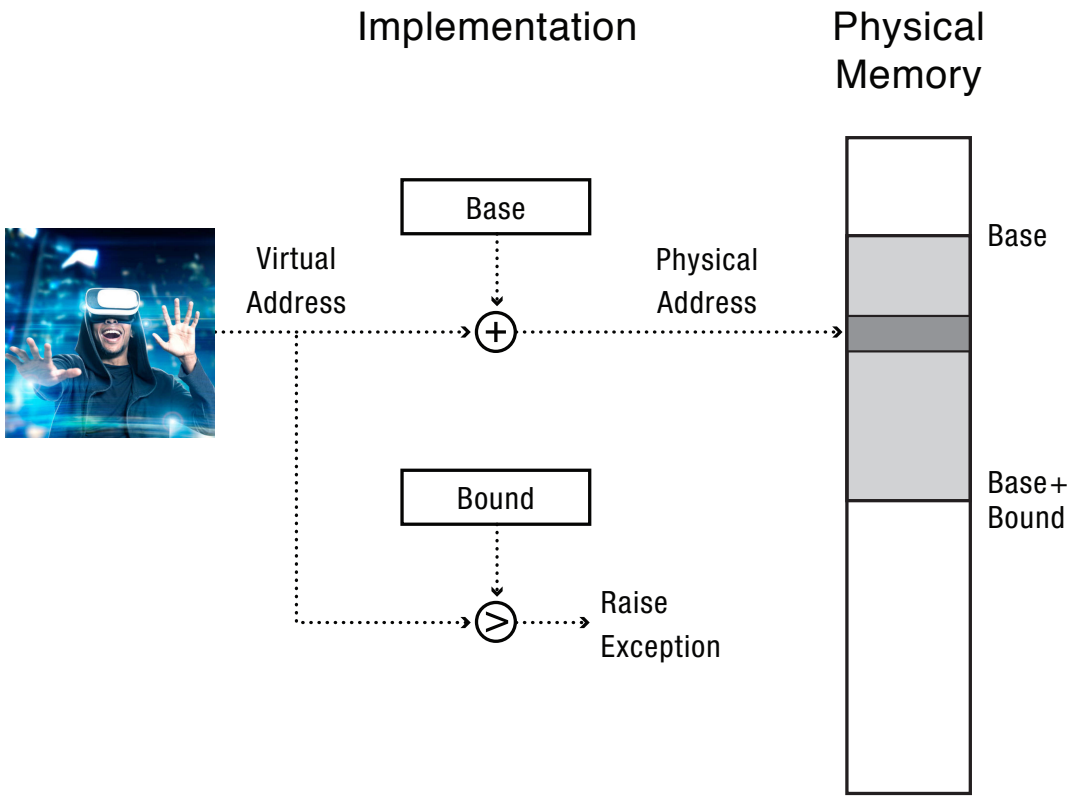
Address Translation





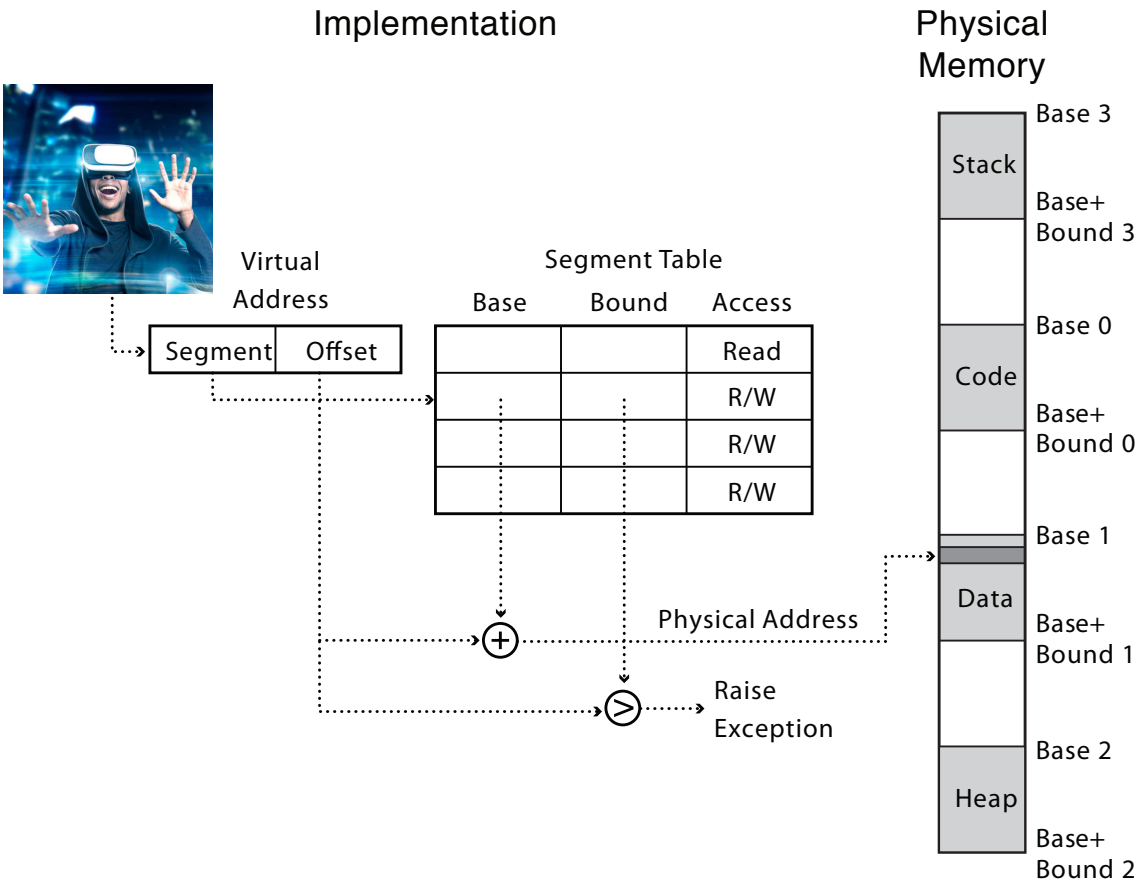
Base-and-Bound

Base-and-Bound



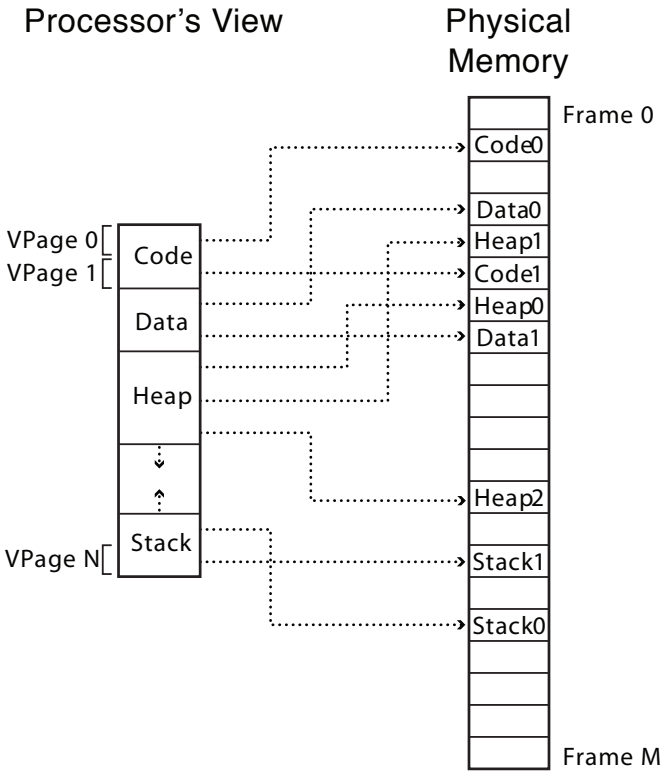
Segmentation

Segmentation



Paging

Paging


Processor's View
Physical Memory
VPage 0
VPage 1
VPage N
Code
Data
Heap
Stack
Code0
Data0
Heap1
Code1
Heap0
Data1
Heap2
Stack1
Stack0
Frame 0
Frame M

Paging

Physical Memory
Physical Address
Frame
Offset
Frame 0
Frame 1
Virtual Address
Page #
Offset
Page Table
Frame
Access
Virtual Address
Page #
Offset
Physical Address
Frame
Offset
Frame M

Memory as a Cache

- each page table entry has a valid bit
- for valid entries, frame indicates physical address of page in memory
- a **page fault** occurs when a program requests a page that is not currently in memory
  - takes time to handle, so context switch
  - evict another page in memory to make space (which one?)

Page Replacement Algorithms

- **Random:** Pick any page to eject at random
  - Used mainly for comparison
- **FIFO:** The page brought in earliest is evicted
  - Ignores usage
- **OPT:** Belady's algorithm
  - Select page not used for longest time
- **LRU:** Evict page that hasn't been used for the longest
  - Past could be a good predictor of the future
- **MRU:** Evict the most recently used page
- **LFU:** Evict least frequently used page

Thrashing

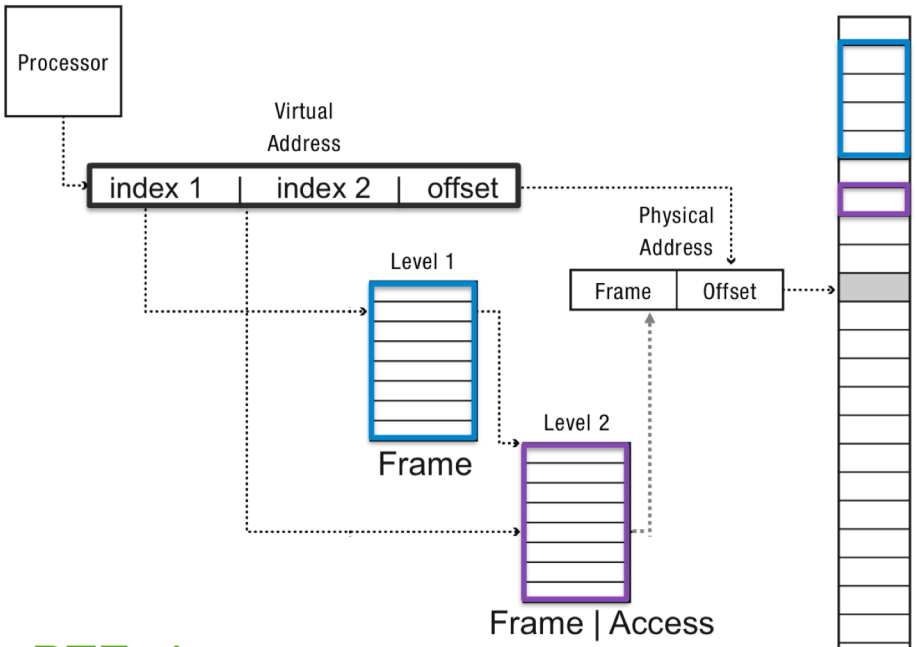
- working set is the collection of a pages a process requires in a given time interval
- if it doesn't fit in memory, program with thrash

Efficient Paging

- How big should our pages be?
  - How much internal fragmentation will there be?
  - How big is the page table?
    - Example: consider 64-bit address space, 4KB (2^12) page size, assume each page table entry is 8 bytes.

- **Performance:** every data/instruction access requires *two* memory accesses:
  - One for the page table
  - One for the data/instruction

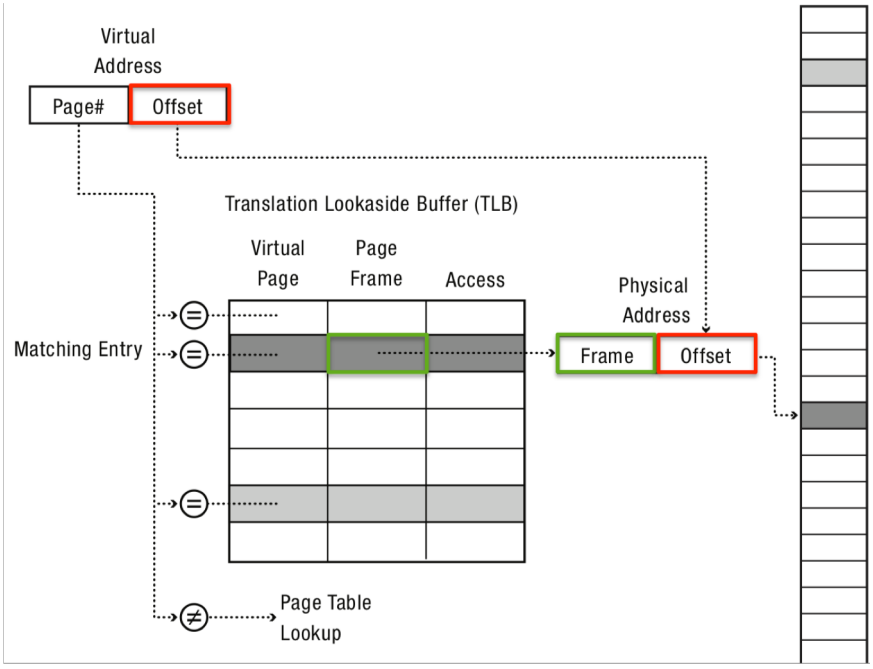
Multi-level page tables



+ Allocate only PTEs in use
+ Simple memory allocation
− ***more*** lookups per memory reference

Translation Look-aside Buffer (TLB)

- **Translation lookaside buffer (TLB):** special cache for page table entries



Example: Core i7 Address Translation