CSCI 460 Operating Systems

Computer System & Operating System Overview

Professor Travis Peters

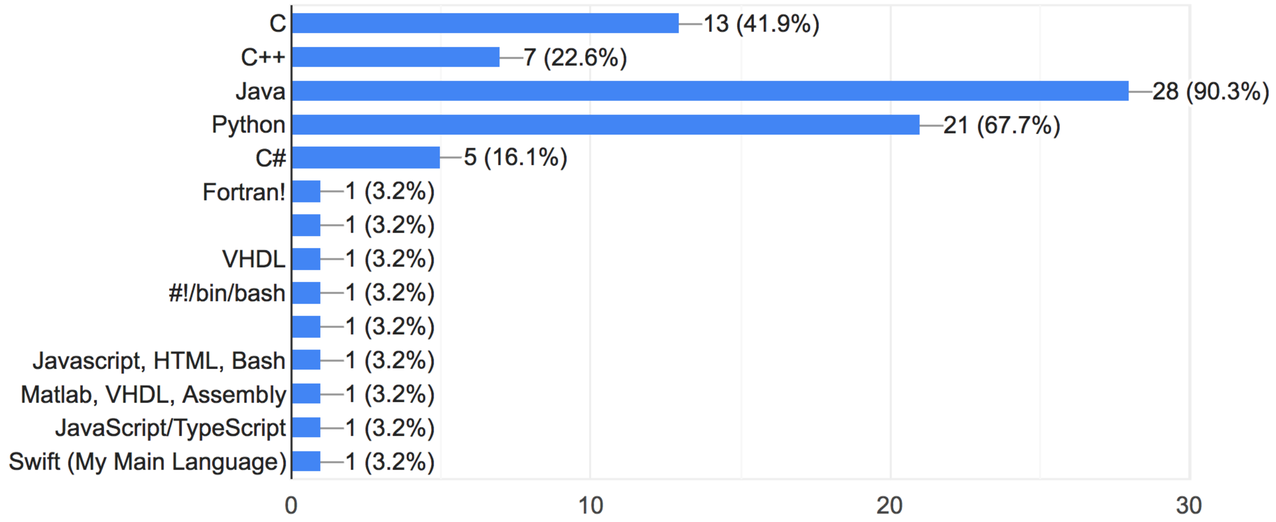
Fall 2019

Goals for Today

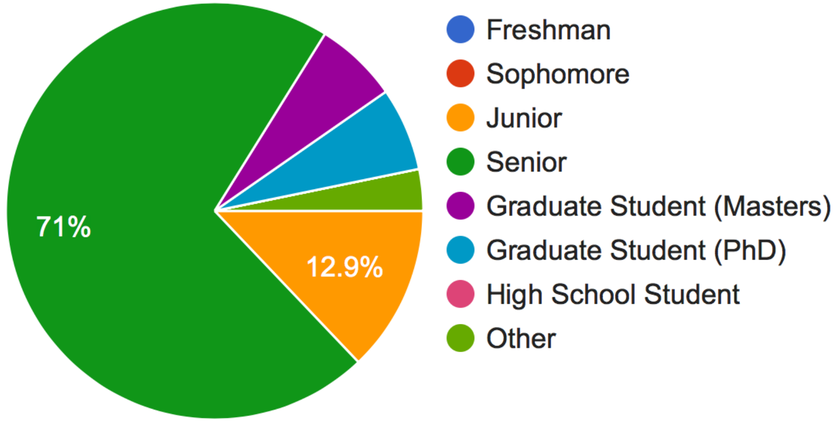
- Learning Objectives
  - Wrap up high-level computer system & OS concepts (Chapters 1 & 2)
  - Connect some dots and pave the way for the rest of the course
    - *not going deep into history, which was covered (enough) last week…*
    - *gory details in the text, of course…*
- Announcements
  - Note taker?!! ;-)
  - Grades for HW1 coming soon (restructuring things in D2L over the next week or so)
  - More HW soon, I promise ;-)

Questionnaire - *The results are in (mostly…)*

Programming Languages:



Who is in this class:



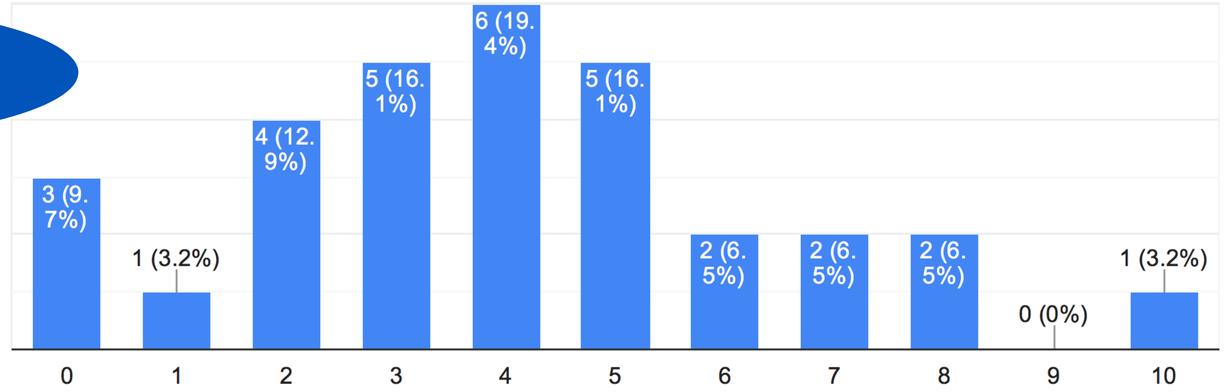
I've heard from software engineers that taking OS is a very helpful class for once you get out into the field

Why do all my profs use Macs?

I did not like the other course offerings

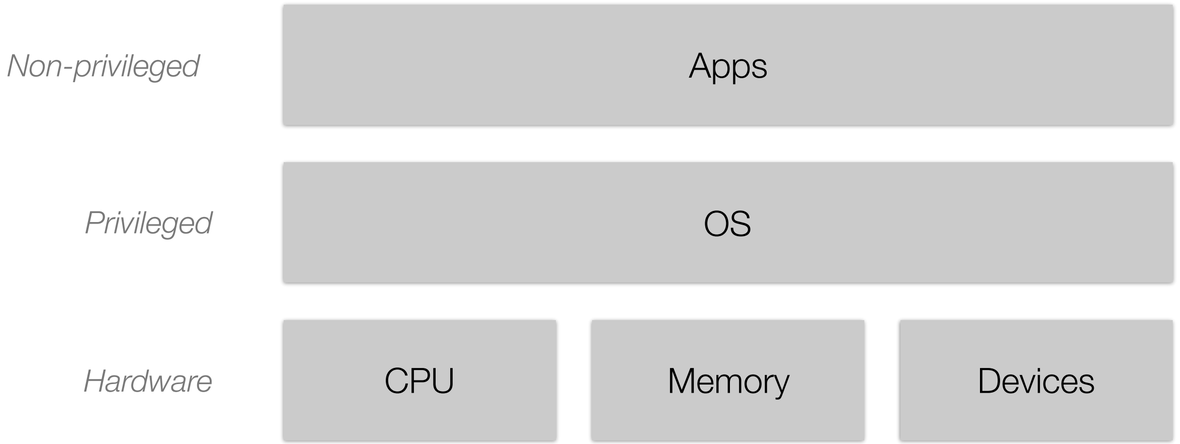
I'm just interested to learn more about the inner workings of operating systems.

Sys Programming Skillzzz:



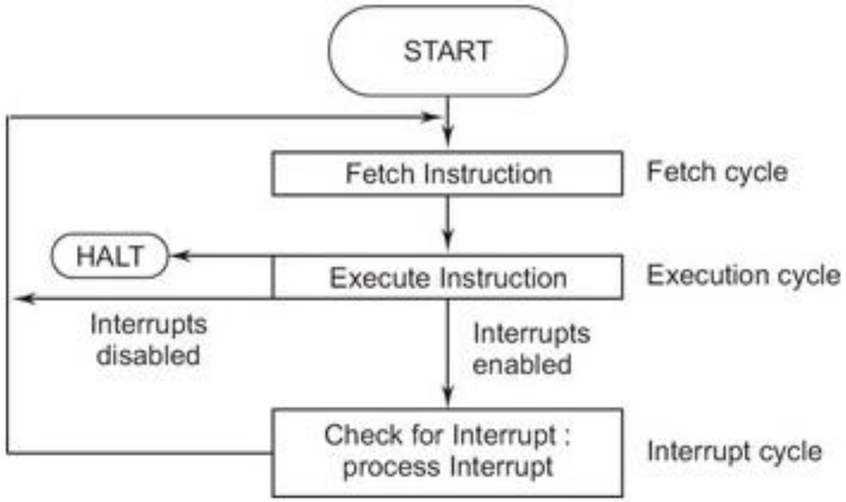
Computer System Overview

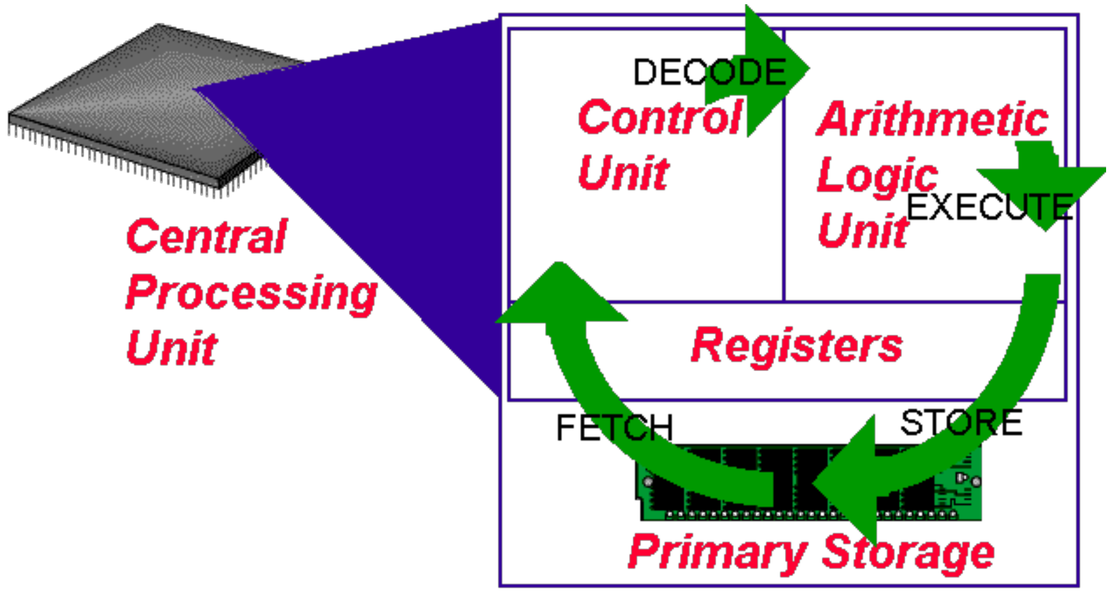
- Basic Model



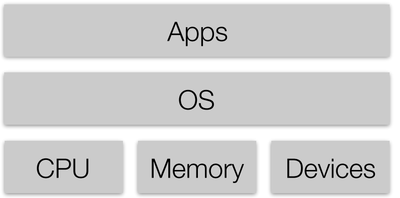
Computer System Overview

- Instruction Exectuion / Instruction Cycle
  - **FETCH** the next instruction, **DECODE** it, **EXECUTE** it, and **STORE** the result.
- Interrupts *(interrupt the processor!)*
  - E.g., program int. (e.g., illegal instruction, out-of-bounds access), timer, I/O, hardware failure
  - Run corresponding “handler”





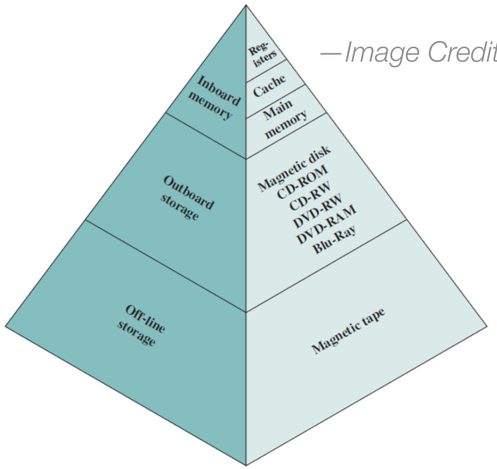
Computer System Overview



- Memory
  - The Memory Hierarchy

    want ***fast access*** to a ***large amount of memory***

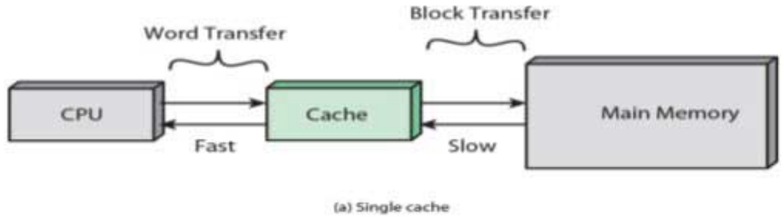
    locality of reference*:* memory access tends to cluster (e.g., loops, subroutines)



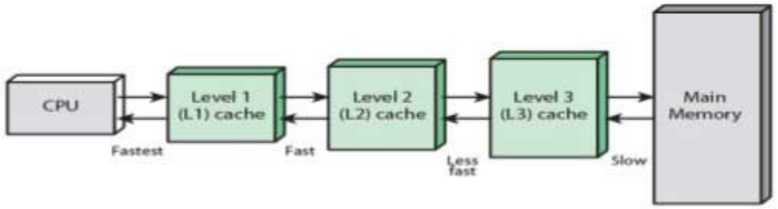
—Image Credit: Stallings

  - Cache Memory

    keep as much as possible, as close as possible…
    …but still support large amounts of memory.
    Multiple levels of cache… need ***cache coherency****.*



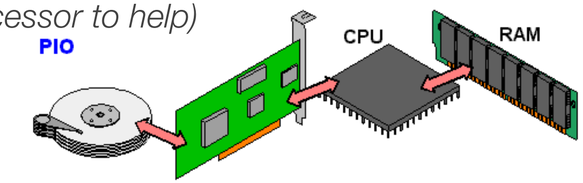
(a) Single cache

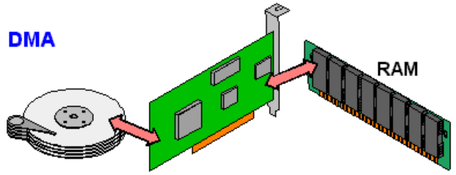


—Image Credit: Stallings

  - Direct Memory Access (DMA)

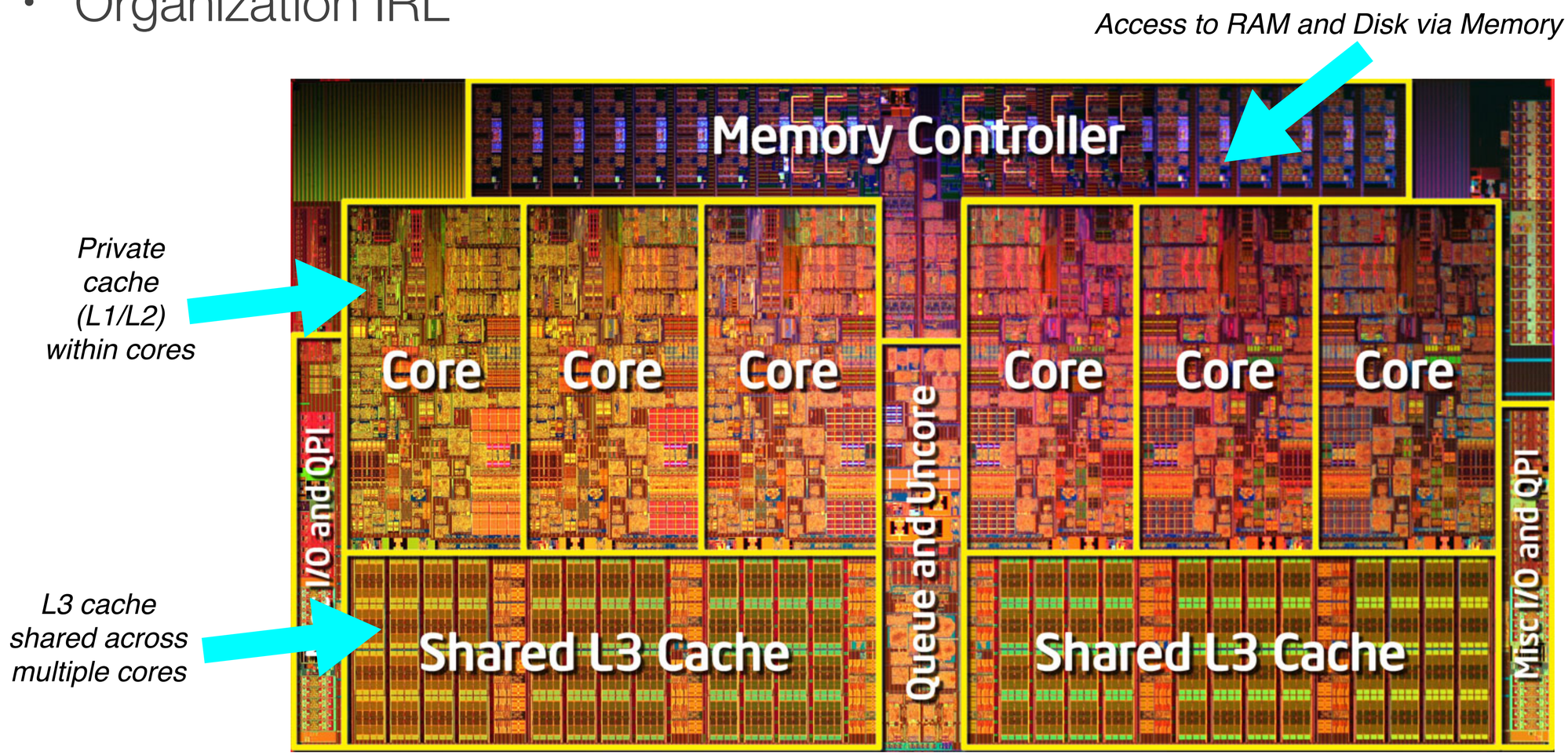
    programmed I/O (active; processor polls I/O device)
    vs.
    interrupt-driven I/O (assist; I/O device does work; interrupt processor to help)
    vs.
    DMA I/O (delegate; DMA module is given
    OP, DEV, ADDR, #WORDS; sent INT when complete)





Computer System Overview

- Organization IRL

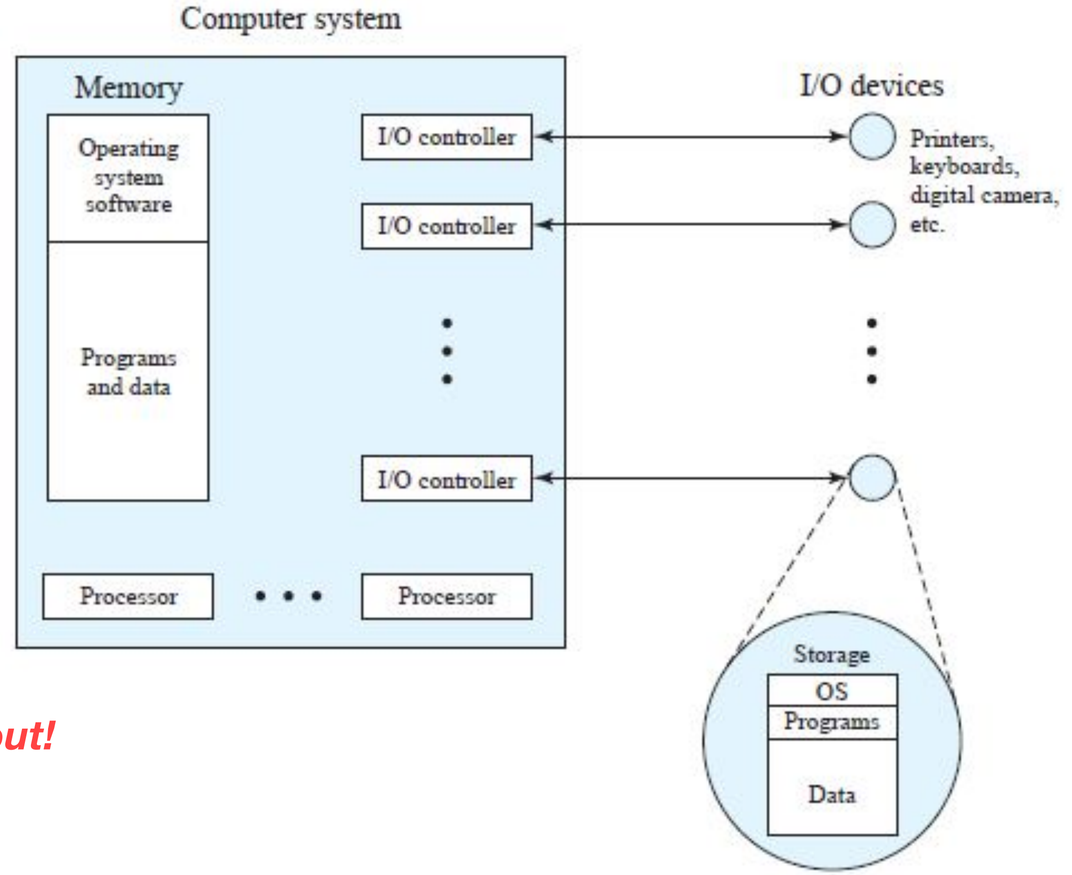


—*Image Credit: Stallings*

Operating System Overview

- Objectives of an OS
  - Covenience, Efficient, Modular
    vs.
    Referee, Illusionist, Glue
- Major Achievements
  - The Process
  - Memory Management
  - Security
  - Scheduling & Resource Management

These topics are what most of the rest of the class is about!



—Image Credit: Stallings

OS Objectives/Roles

The textbook offers one way of looking at the role of the OS. Here is another:

- Referee
  - manage resources between apps
  - isolate apps and users from one another
  - facilitate communications between apps/users

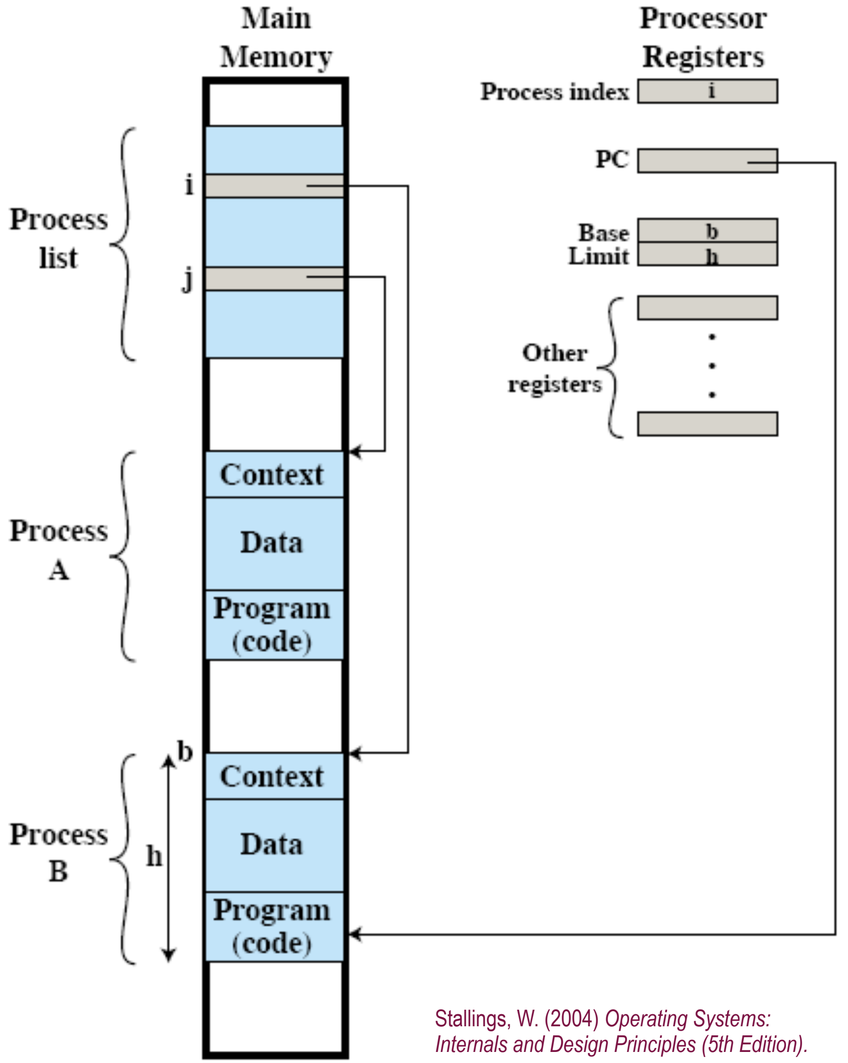
- Illusionist
  - make apps believe they have the whole machine to themselves
  - create appearance of infinite processors and memory
  - abstract away complexity of storage, network communications, etc.

- Glue
  - manage hardware so apps can be machine-agnostic; provide a set of common services

—Thanks to Adam Bates for the nice analogies: https://courses.engr.illinois.edu/cs423/sp2018/

The Process

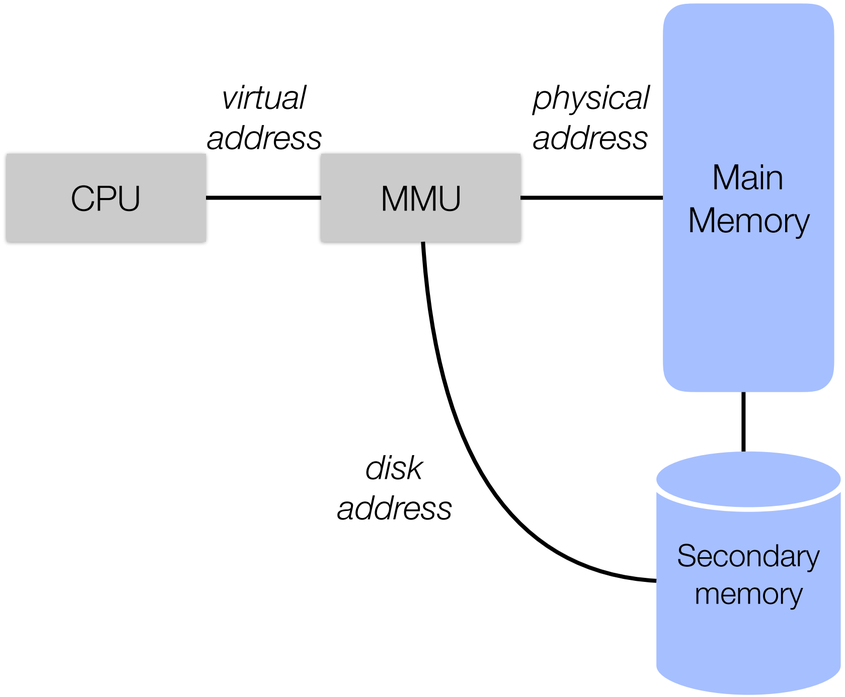
- The notion of a ***Process***…
  - arose from multiprogramming, time sharing, real-time transactions;
  - is more general than a "job"; many definitions;
    e.g., ***an instance of a program running on a computer****;*
  - consists of
    - (1) an executable program (**code**),
    - (2) assocated **data**,
    - (3) **execution context** (info the OS needs to manage the process)
  - is realized as nothing more than a data structure!
  - A ***thread*** = cooperative execution *within* a process; use shared context.
- Process Switching
  - Interrupt = save context (e.g., PC and other registers)
    ➡ execute interrupt handler
    ➡ resume processing (same or different process)
  - states (simply put) = ***executing*** or ***awaiting*** *execution*



Stallings, W. (2004) *Operating Systems: Internals and Design Principles (5th Edition).*

Memory Management

- OS responsibities w.r.t. MM
  - Process Isolation + Automatic Allocation/Mgmt. + Modular + Protection & Access Control + Long-Term Storage
- Virtual Memory
  - address memory ***logically***, without regard to the actual, ***physical*** memory.
  - (recall “paging”) ***virtual address*** = page # + offset within the page.
  - MM maps between virtual address and real (physical) addresses.
  - pages can be in memory (or not!) — *how is that possible?!*
- File Systems
  - good for persisting information for extended periods of time
  - **files** = named objects

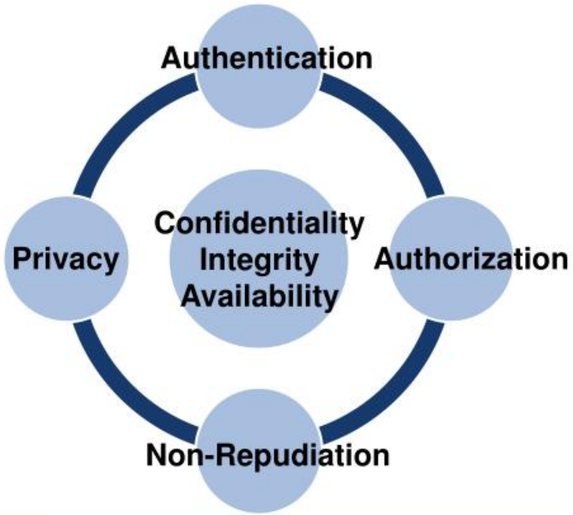


Information Protection & Security

- CIA
  - **C**onfidentiality — protection from unauthorized access *(e.g., snooping)*
  - **I**ntegrity — protection from unauthorized modification *(e.g., tampering)*
  - **A**vailability — protection from interruption *(e.g., denial of service)*

- Also…
  - Authenticity — verification of the origin/identity
  - +…

traditional infosec model…



…increasingly common to include other goals

Scheduling & Resource Management

- Manage Resources!
  - ***Manage*** system resources (Main Memory, I/O devices, Processors)
  - Keep these resources ***utilized*** (i.e., schedule processes to utilize them)
  - …all while being ***fair***, ***responsive***, and ***efficient***
- Operations-Research Problems & Lots of Maths

  —How do we do things good?
  - **Data Structures:** A bunch of queues, and lists, and (mostly) other simple data structures
  - **Algorithms:** A ***scheduler*** or ***dispatcher*** to pick which process runs next
    - ***round-robin*** = everyone gets some time in turn. *Other approaches?*