CISC 7310X
C01: Overview

Hui Chen

Department of Computer & Information Science

CUNY Brooklyn College

Quick Poll

- Did you take an Operating Systems class before?
- Have you written programs in the C programming language?

Outline

- Roll call
- Policy and organization of the course
- Overview of operating systems
- Assignments
  - See the class website for due dates
  - Descriptions are in Blackboard

Resources and Websites

- Class website
  - http://www.sci.brooklyn.cuny.edu/~chen/course/CISC7310X
    - Online syllabus
    - Weekly schedule, lecture notes, and assignments
    - Additional resources
- CUNY Blackboard
  - https://bbhosted.cuny.edu
    - Advisor grades
    - Description of assignments
- Assignment submission
  - Git repositories hosted at Github (https://github.com)

Course Content

- Prerequisite
  - Data structures and computer organizations
- Content
  - Systems overview; systems programing; files; access control; resources management; and system modeling

Learning Objectives

- Interaction between system components (hardware, operating systems, applications)
- Major issues and solutions in system design (problems, data structures and algorithms)
- System programming including system calls and programming tools
- System research methods and tools as well as reading, writing, presentation, and experimentation

Textbook and Major References

- Textbook
  - Andrew S. Tanenbaum and Herbert Bos. 2014. Modern Operating Systems (4th ed.). Prentice Hall Press, Upper Saddle River, NJ, USA.
- Main reference books and online resources
  - Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. 2008. *Operating System Concepts* (8th ed.). Wiley Publishing.
  - https://pdos.csail.mit.edu/6.828/2017
  - http://pages.cs.wisc.edu/~Eremzi/OSTEP/

Grading Component and Scale

- Attendance
- Practice assignments
  - Programming and laboratories
- Projects
  - Team projects
  - Research, programming, and laboratories
- Midterm exam
- Final exam

Teaming

- Draw a random ballot
- 3 – 4 students a team
- Steps
  - Draw 3-member teams
  - Students who draws "*" draw again to join a team that has 3 members or fewer.

Assignments

- Learn to accept and submit assignments using Git at Github repositories
  - Individual assignment
  - Team assignment

Tool Support for Team Work

- Version control system
- Summary of your experience

Version Control System (VCS)

- Why do we need it?
  - https://stackoverflow.com/questions/1408450/

"
Have you ever:

Made a change to code, realised it was a mistake and wanted to revert back?
Lost code or had a backup that was too old?
Had to maintain multiple versions of a product?
Wanted to see the difference between two (or more) versions of your code?
Wanted to prove that a particular change broke or fixed a piece of code?
Wanted to review the history of some code?
Wanted to submit a change to someone else's code?
Wanted to share your code, or let other people work on your code?
Wanted to see how much work is being done, and where, when and by whom?
Wanted to experiment with a new feature without interfering with working code?

Team Support with VCS

- VCS provides a "centralized" location to store project files
  - Versioned code, configuration files, build scripts …
- VCS tracks each contributors' individual changes
- VCS helps prevent concurrent work from conflicting

Benefits of VCS

- Branching & merging.
  - Example workflow: branching for each feature, branching for each release.
- Traceability
  - Example use scenarios: track changes between revisions of a project, documented history of who did what and when
- Complete history of changes
  - Example use scenarios: help in root cause analysis for bugs, fix problems in older versions of software that has been released, roll back to an older version without newly introduced bugs

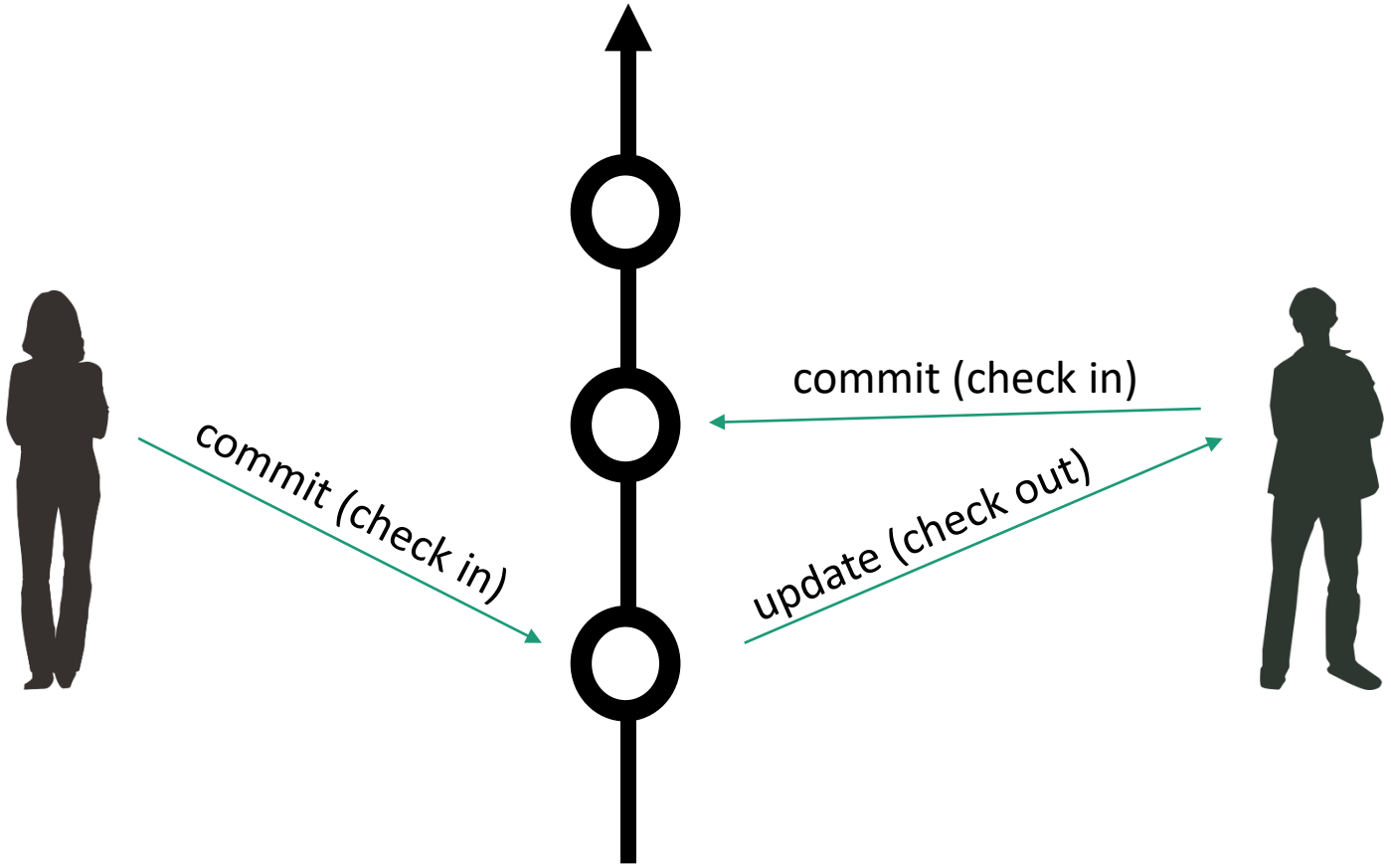
Centralized vs. Distributed

- Centralized VCS
  - Examples: Revision Control System (RCS), Concurrent versions systems (CVS), Subversion (SVN)
- Distributed VCS
  - Examples: Git, Mercurial (hg)

Basic VCS Operations

- Check out/update: copying the repository to the machine you are working at
- Check in/Commit: copying the changes you made to the repository and creating a new version
- Branch: create a new "child" development from a state of the repository

Example: Centralized Workflow

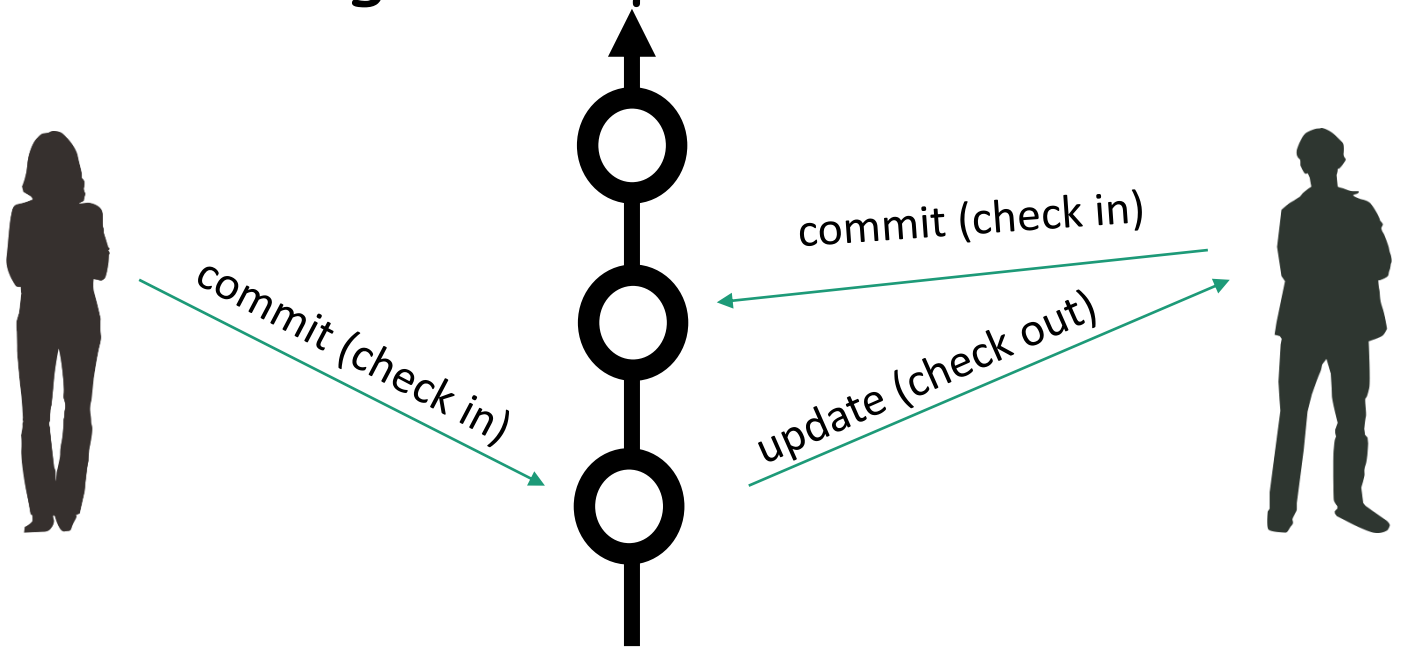


Merge Conflicts

- A conflict may occur when two developers edit the same file
- Merge
  - The developer that tries to commit the file ***last*** will have to combine her changes with those of the prior developer
  - Many VCS's (e.g., git) may automatically combine the changes
  - Developers may have to merge the changes by hand

Question: A Merge?

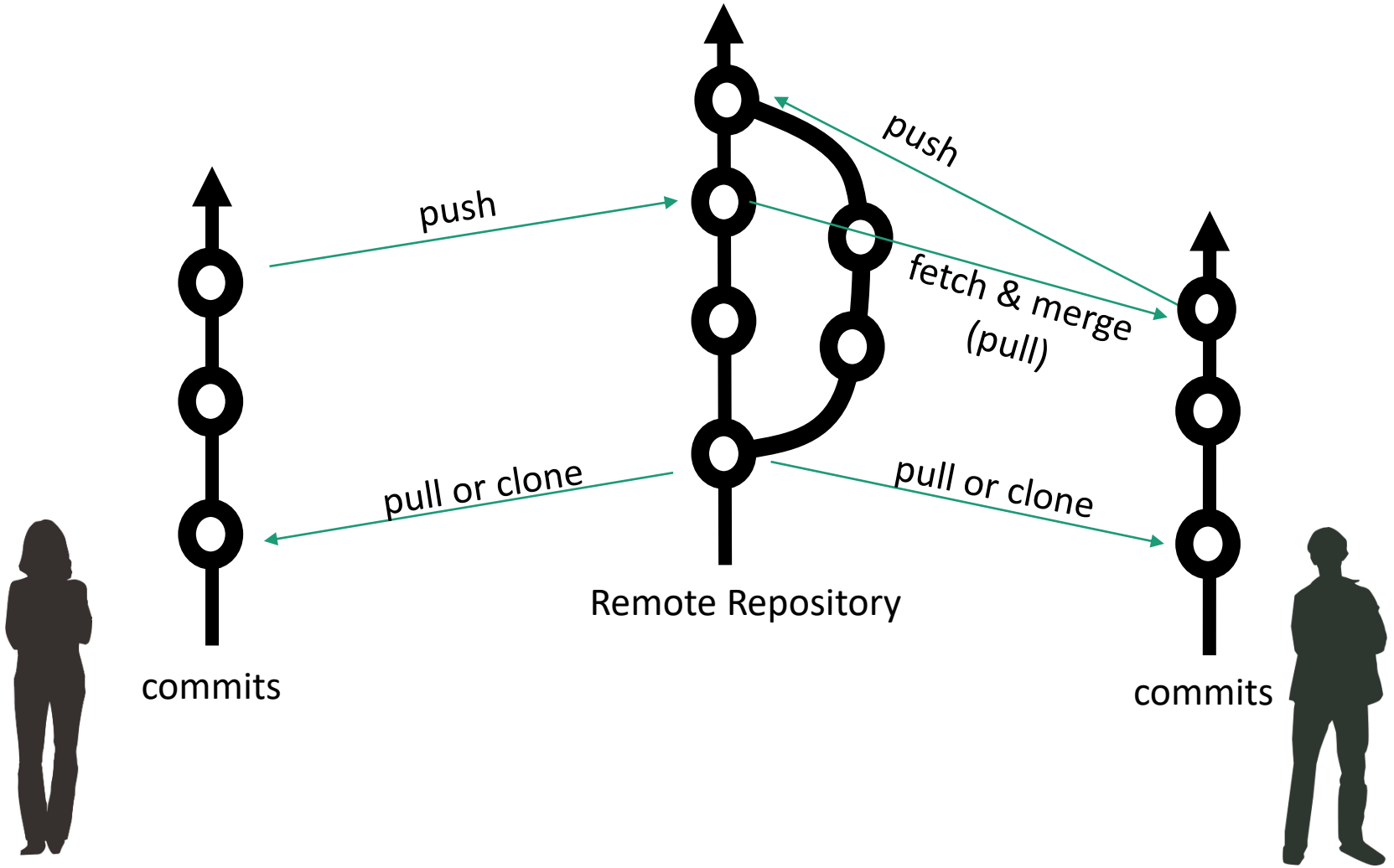
- How would you revise this graph to illustrate when a merge is required?



Distributed Version Control

- Possible to commit locally without upsetting the others
- Allow more flexibility and support different kinds of workflow

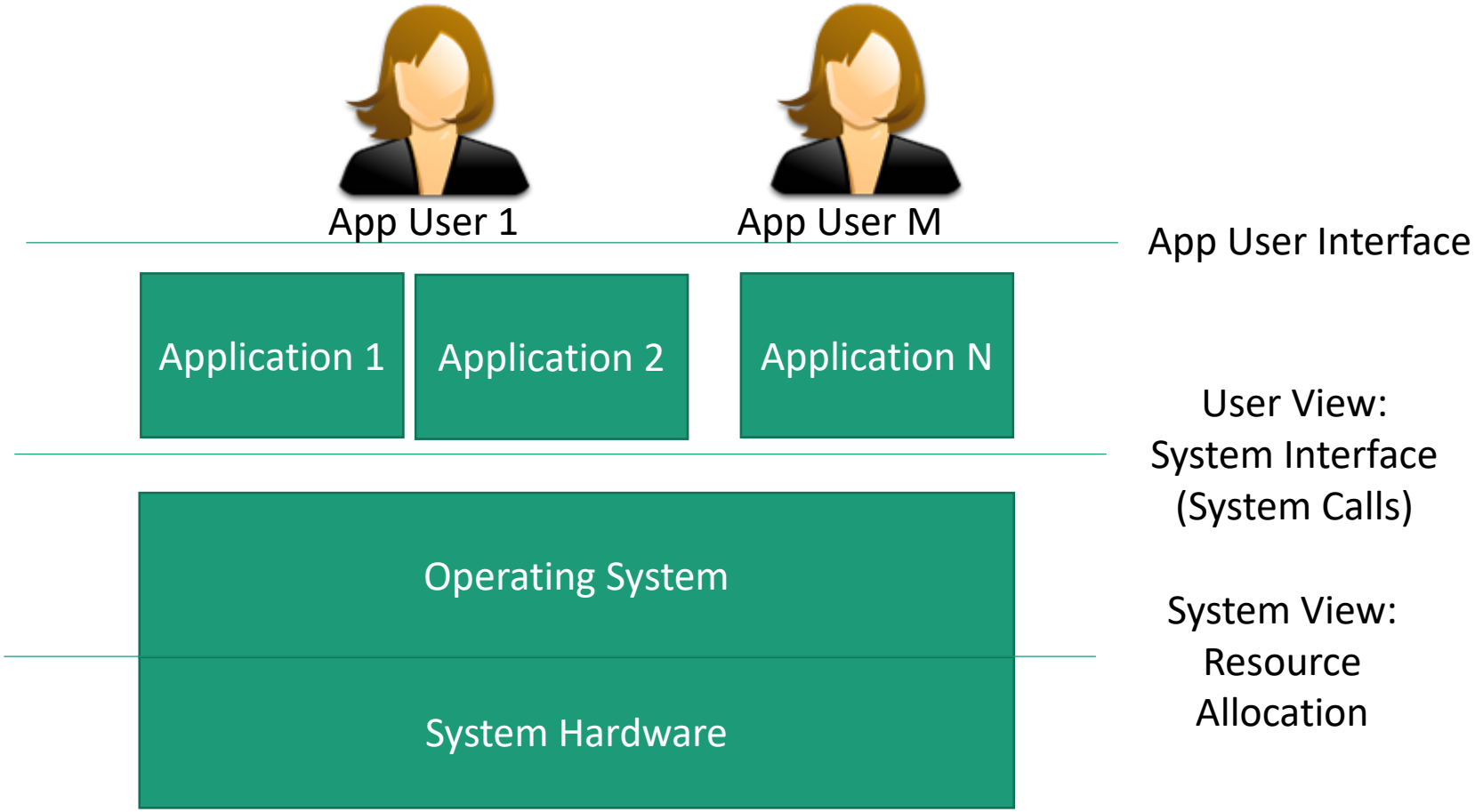
Example: Distributed Workflow



Questions

- Course policy and organization
- Assignments
- Assignments submission via Git and Github

Operating System



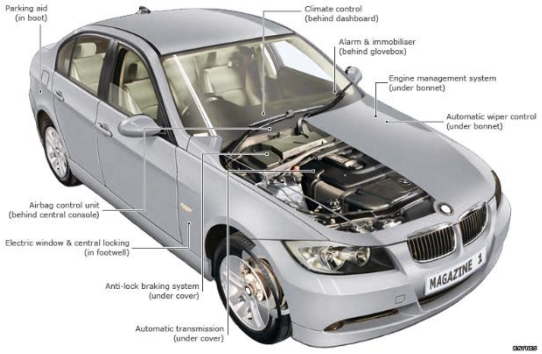
Concept of Operating Systems

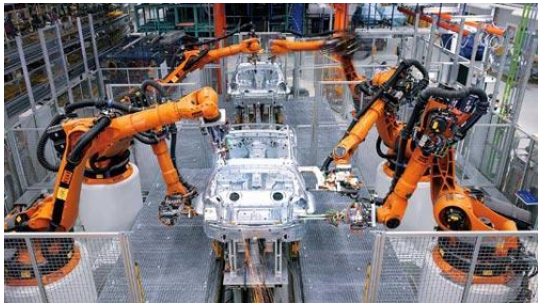
- A large piece of software function as
  - an extended machine (user view)
    - to provide an "beautiful" interface for application programs via the application developers
  - a resource manager (system view)
    - to provide a "beautiful" allocation scheme to share the processors, memories, and I/O devices in a "computer system"

“beautiful”

- Question & Discussion: what is “beautiful”?

Various Computer Systems



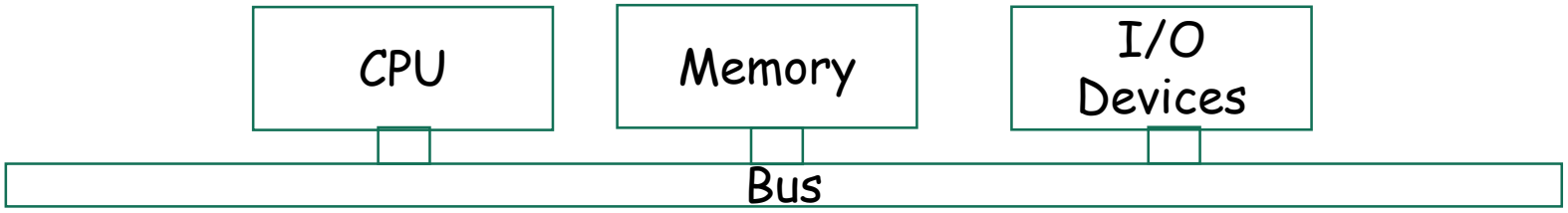


So, lots of computers …

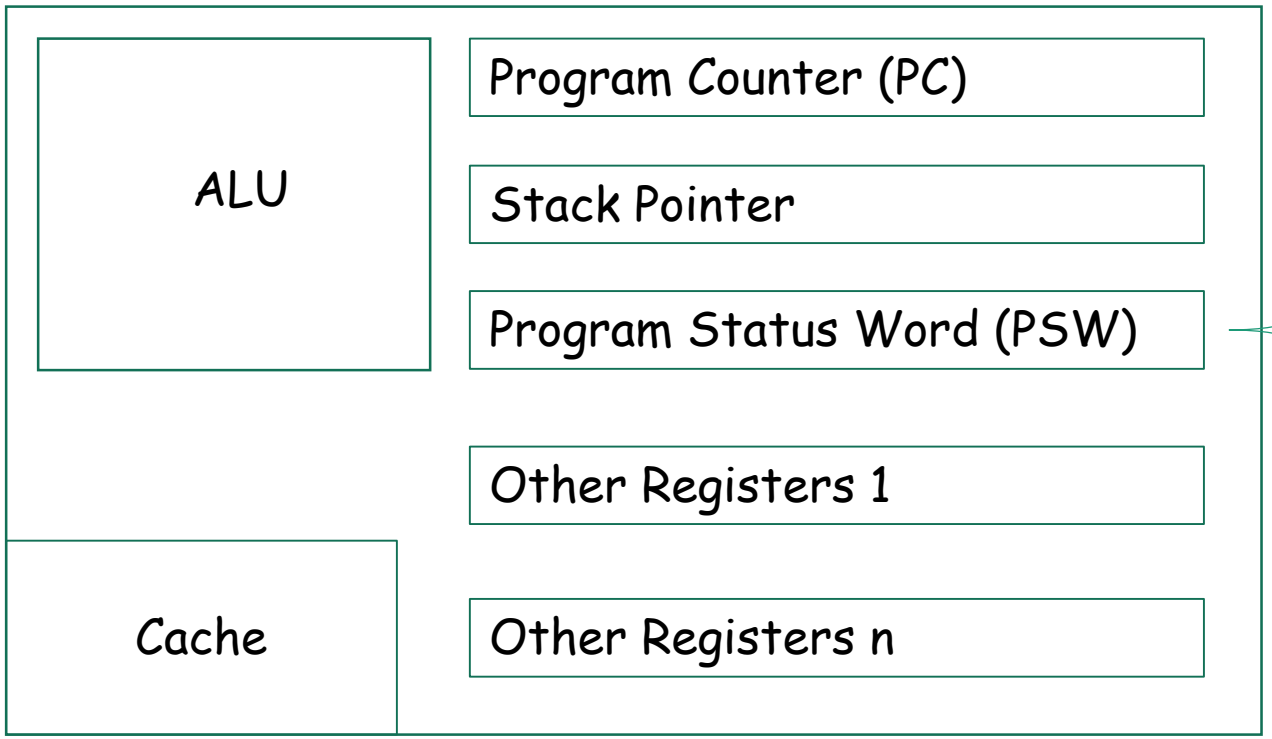
- Question and discussion: what are in common and what are different?

Major Hardware Components

- Processors (CPU)
  - Multithreaded and multicore processors
- Main Memory (Memory)
- Secondary Memory (Disks)
- I/O Devices
- Buses



Processors

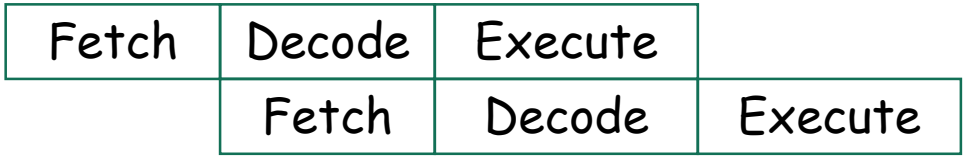


Kernel mode: can execute all instructions and access all hardware features

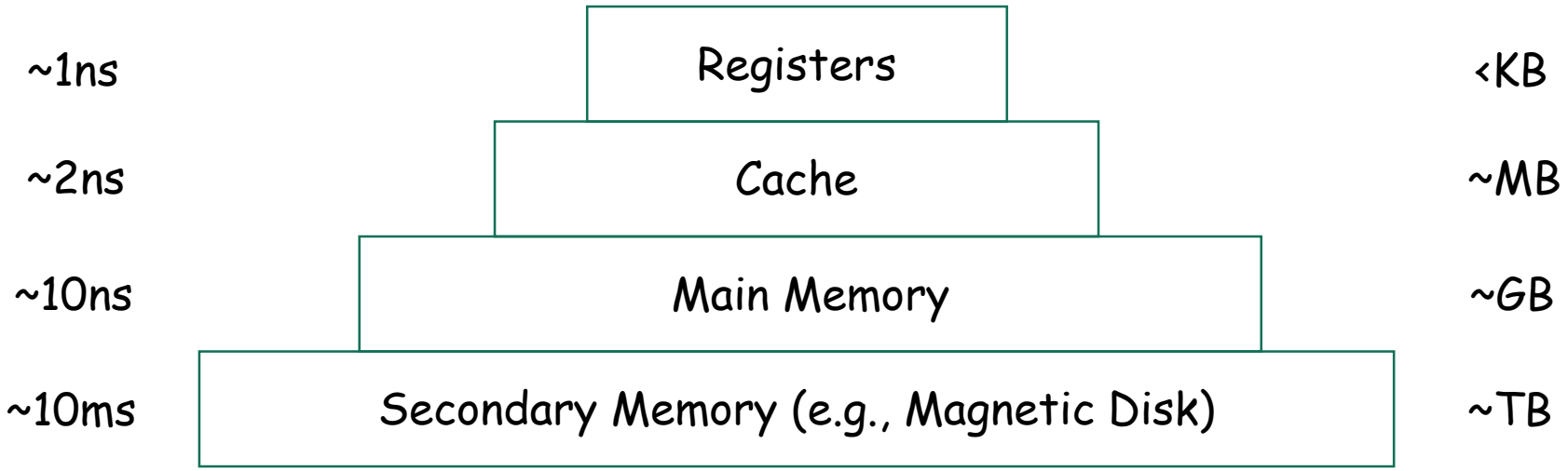
User mode: can execute subset of instructions and access subset of hardware features

Instructions

- Basic cycle
  - Fetch, decode, execute
  - Enhance: e.g., pipelining
- Instruction set
  - Examples
    - x86 (i386 for 32-bit; amd64 for 64-bit)
    - ARM



Memory Hierarchy



I/O

- Busy waiting
- Interrupted I/O
- Direct memory access (DMA)

Operating Systems

- Mainframe Operating Systems
- Server Operating Systems
- Multiprocessor Operating Systems
- Personal Computer Operating Systems
- Handheld Computer Operating Systems
- Embedded Operating Systems
- Sensor-Node Operating Systems
- Real-Time Operating Systems
- Smart Card Operating Systems

So, a zoo of operating systems?

- Question: why? How are they differ?

Design Goals

- Resource utilization
- Timeliness
- Throughput
- Robustness
- Energy efficiency

Operating Systems Concepts

- Processes
- Address spaces
- Files
- I/O
- Protection
- The Shell
- The Kernel
- System Calls

Process

- A program in execution
  - Address space
    - Divided into a few parts: e.g., stack, heap, program code, program data
  - Resources
    - List of open files
    - List of related processes
  - Current working directory

Systems Research Literature

- Digital Libraries
  - ACM, IEEE, and USENIX
- Google Scholar
  - The Computer Systems subcategory
- List of researches in Section 1.9

Computing Research

- Computation is synthetic
  - Different from natural sciences, such as, biology and physics
  - We create and study artifacts – must show the artifacts are "better"
- Two paradigms
  - Theory and experimentation
    - Theory: Similar to mathematics of an abstract phenomena
    - Experimentation: Property of artifacts
      - System research are largely experimental.

"Better" Property

- Examples
  - "solves a problem in less time"
  - "solves a larger class of problems"
  - "is more efficient of resources"
  - "is more expressive by some criterion"
  - "is more visually appealing in the case of graphics"
  - "presents a totally new capability"

What Makes it Better?

- The "better" property is not simply an observation
- More about postulating that a new idea that something fundamental leads to the "better" result
- Examples
  - Data structure, algorithm, language, mechanism, process, representation, protocol, methodology, optimization or simplification, and model

Research and Practice

- "Research" is broadly defined.
- In practice, the same principle applies
  - When you design a system solution, is it because this is the first design that comes to your mind or it is a better design?

Questions

- Overview of operating systems
  - Concept
  - Hardware
  - Operating systems concepts
  - Systems research

Assignments

- Gain hands-on experience and familiarity with operating systems concepts
  - Submission required
- Familiar yourself with digital libraries and Google Scholar

Quick Poll

- Do you have a laptop that you can bring to the class (on some days)?

Questions

- Policy and organization of the course
- Overview of operating systems
- Assignments
  - See the class website for due dates
  - Descriptions are in Blackboard