CSE 390B, Spring 2022

Building Academic Success Through Bottom-Up Computing

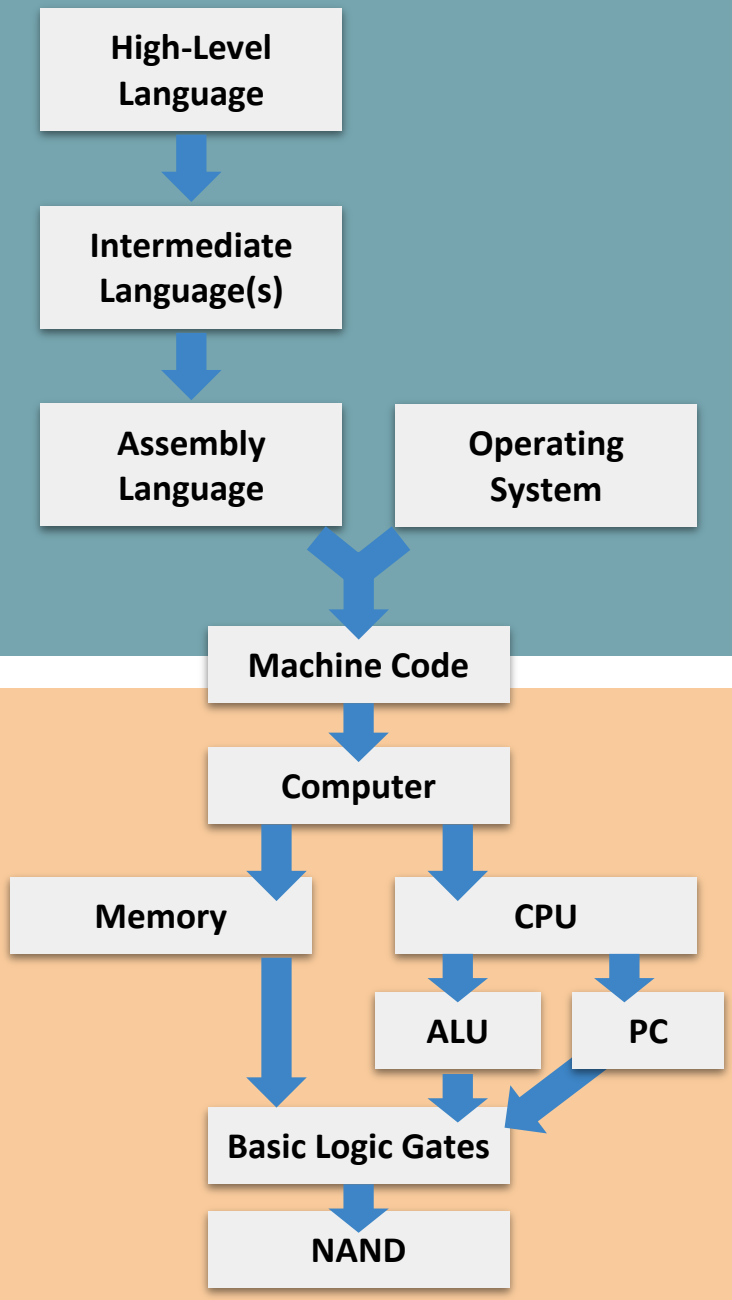
Operating Systems & Final Project Overview

The Software Stack, Operating Systems Overview, Final Project Overview

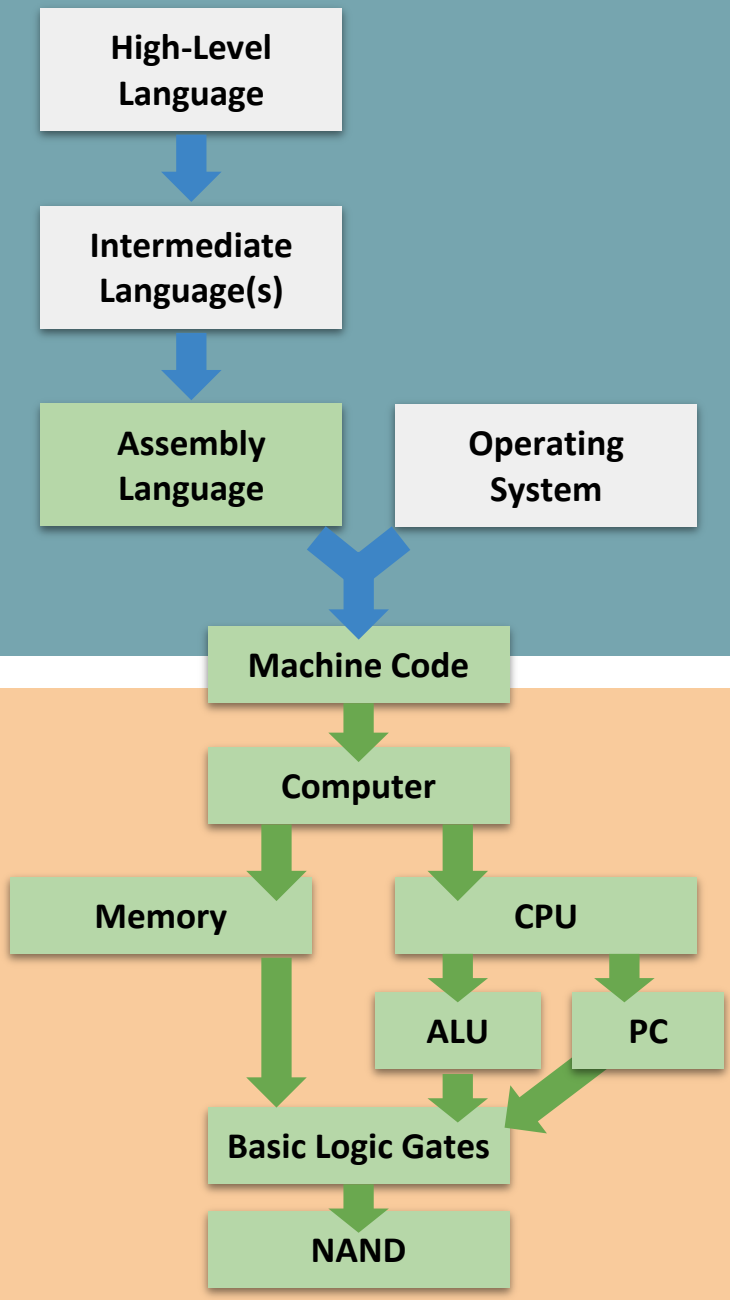
Lecture Outline

- **The Software Stack**
  - **Roadmap of Hardware and Software Components**
- Operating Systems Overview
  - Abstraction, Protection, Processes, Virtual Memory
- Final Project Overview
  - E-Portfolio Details and Presentation Logistics

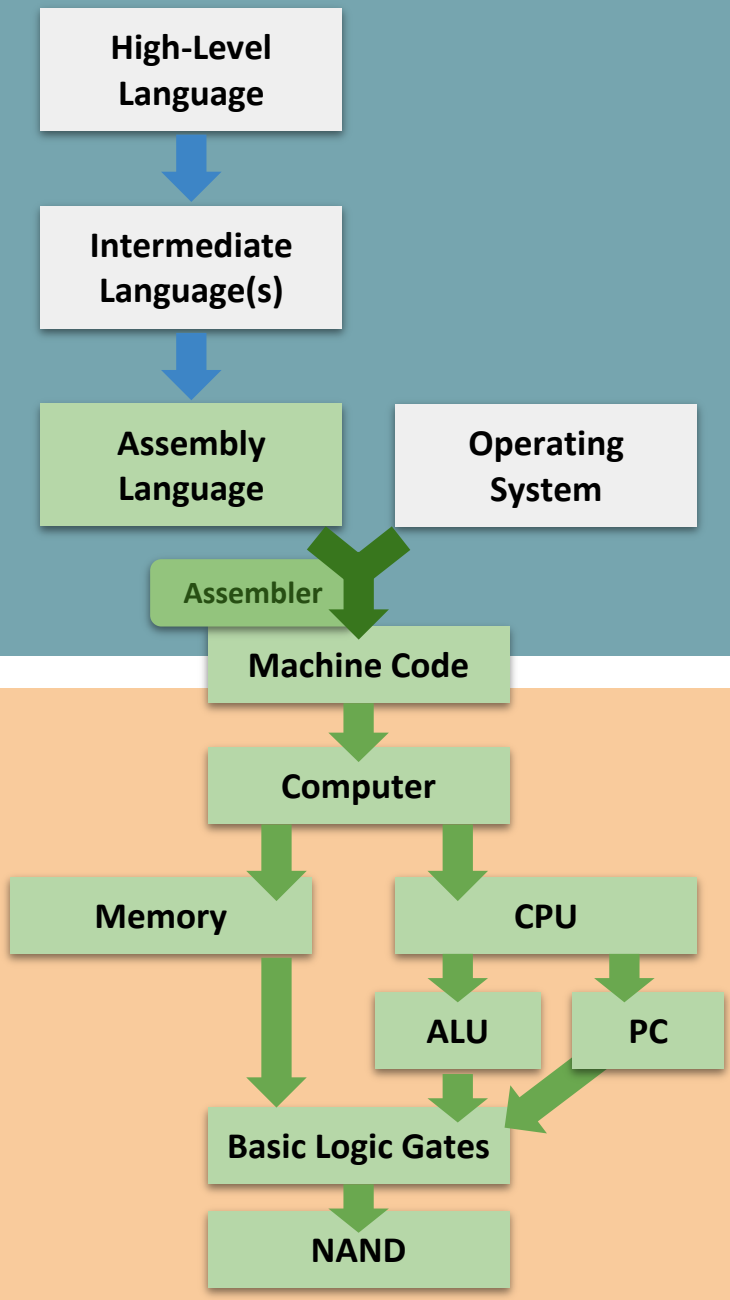
Roadmap


High-Level Language
Intermediate Language(s)
Assembly Language
Operating System
SOFTWARE
Machine Code
HARDWARE
Computer
Memory
CPU
ALU
PC
Basic Logic Gates
NAND

Roadmap


High-Level Language
Intermediate Language(s)
Assembly Language
Operating System
SOFTWARE
Machine Code
HARDWARE
Computer
Memory
CPU
ALU
PC
Basic Logic Gates
NAND

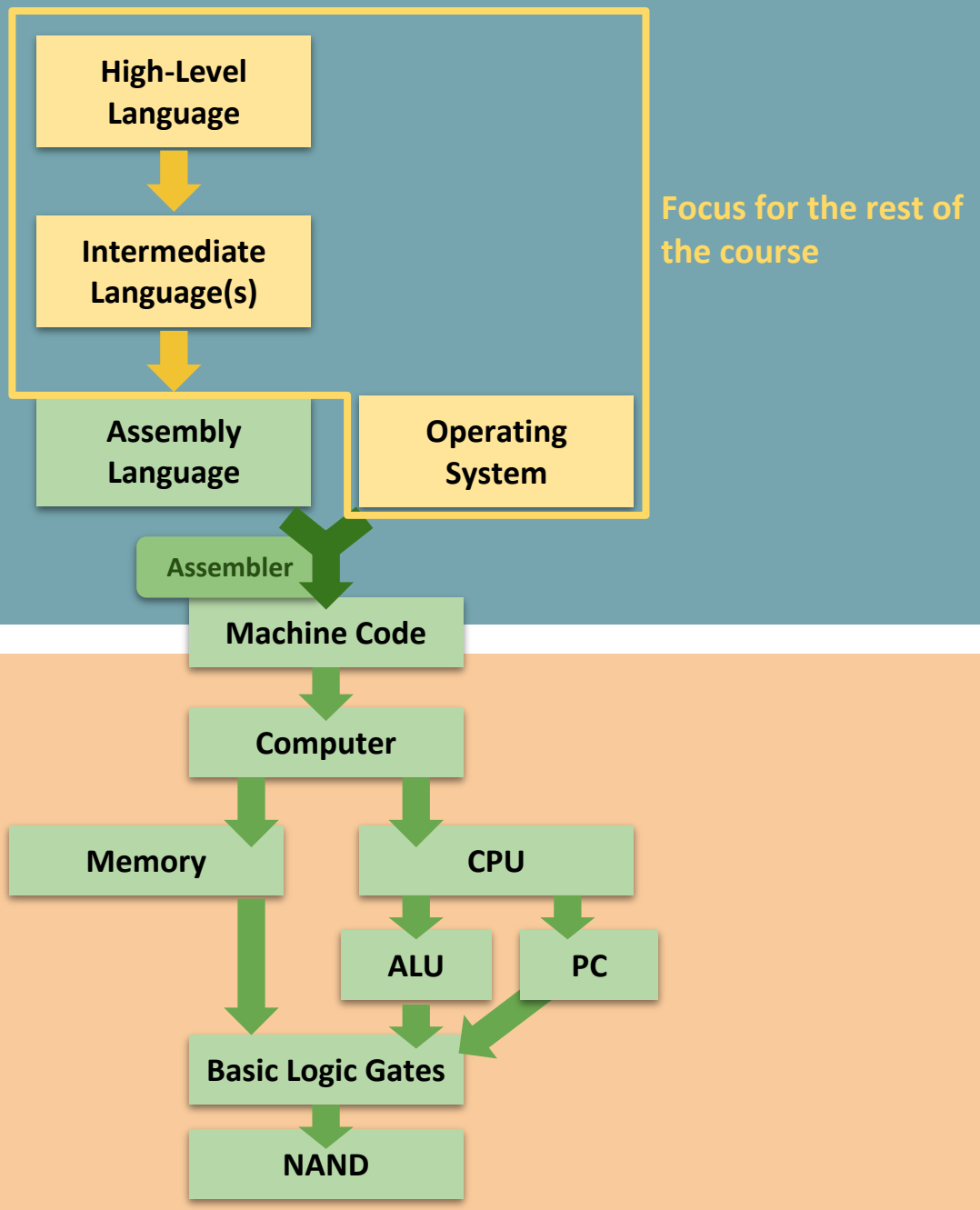
Roadmap



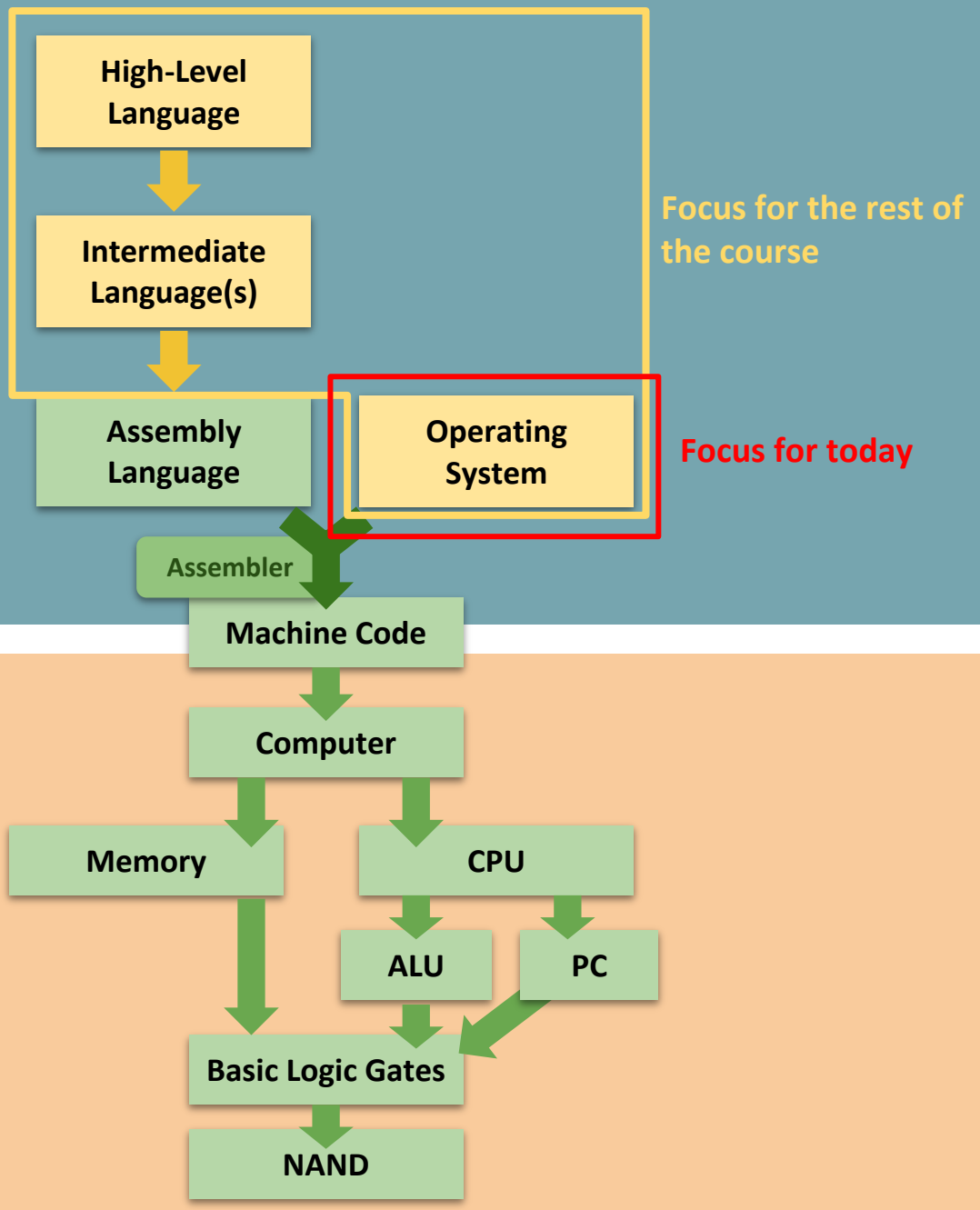
SOFTWARE

HARDWARE

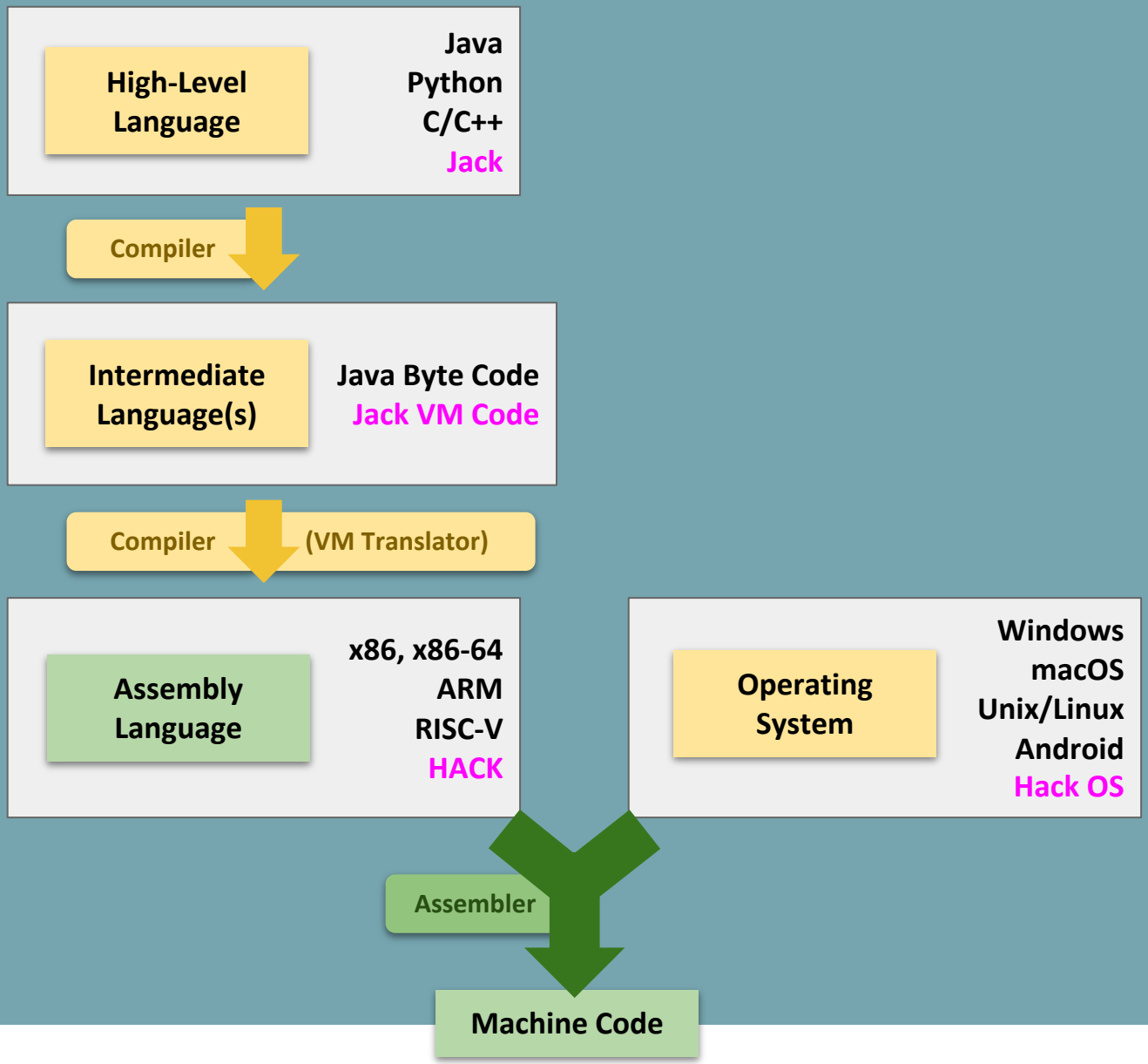
Roadmap


High-Level Language
Intermediate Language(s)
Assembly Language
Operating System
Focus for the rest of the course
SOFTWARE
Assembler
Machine Code
HARDWARE
Computer
Memory
CPU
ALU
PC
Basic Logic Gates
NAND

Roadmap


High-Level Language
Intermediate Language(s)
Assembly Language
Operating System
Focus for the rest of the course
Focus for today
SOFTWARE
Assembler
Machine Code
HARDWARE
Computer
Memory
CPU
ALU
PC
Basic Logic Gates
NAND

Software Overview



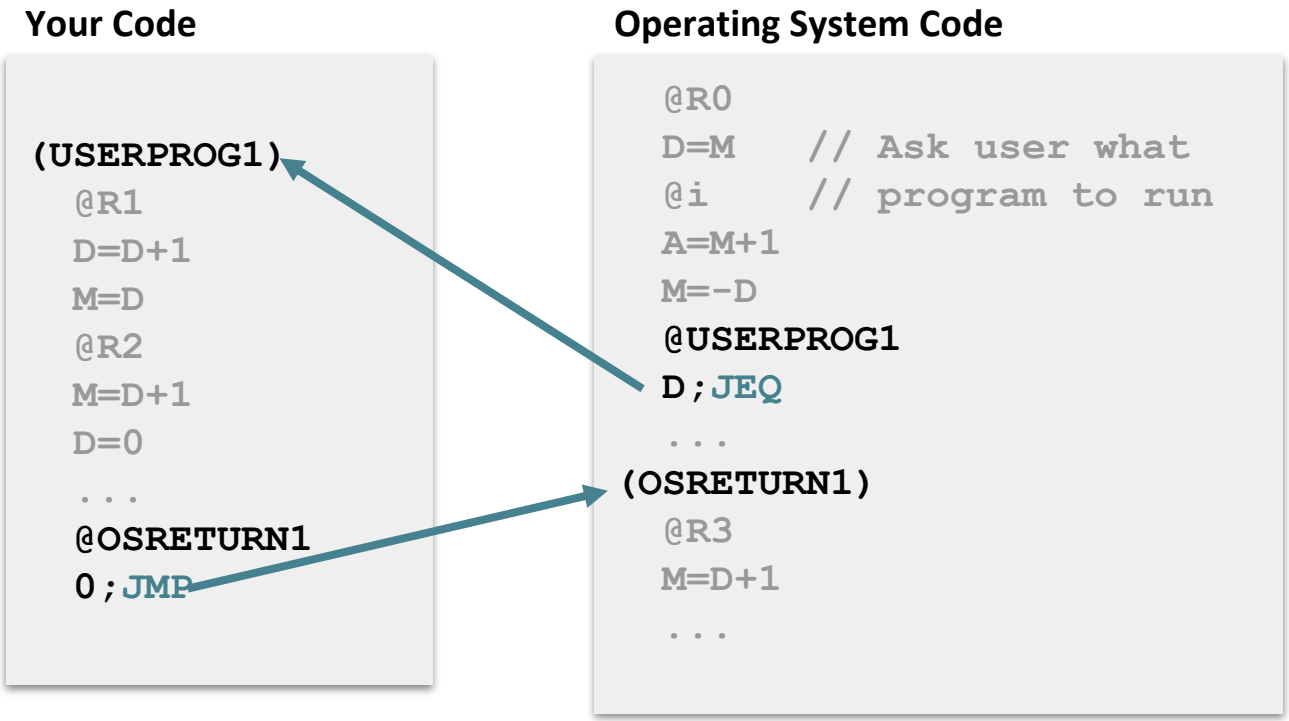
SOFTWARE

Lecture Outline

- The Software Stack
  - Roadmap of Hardware and Software Components
- **Operating Systems Overview**
  - **Abstraction, Protection, Processes, Virtual Memory**
- Final Project Overview
  - E-Portfolio Details and Presentation Logistics

The Operating System

- The operating system (OS) is just another piece of software
  - A massive, complex piece of software
  - In the end, uses the same machine language your code does



The Operating System

- The operating system (OS) is just another piece of software
  - A massive, complex piece of software
  - In the end, uses the same machine language your code does
- OS is more trusted than the rest of the software that runs on your computer
- User programs and applications invoke (ask) the OS to perform operations they are not trusted or allowed to
  - Means the OS needs to be secure

Why an Operating System?

- Directly interacts with the hardware

- Benefit: **Abstraction**
  - Provides high-level functionality for messy hardware devices
  - OS must be ported to new hardware, but user-level programs can then be portable

- Benefit: **Protection**
  - OS is trusted to touch hardware; user-level programs are not
  - User-level programs cannot “break things”
  - Maintains security between programs and user accounts

Operating System: Abstraction

- Many abstractions provided by real-world operating systems

- File System
  - File contents = just bits in the "giant array" that is the hard drive ("permanent" storage, as opposed to temporary storage in RAM that disappears when computer is turned off)
  - OS keeps a record of which ones fall into which "files"

Operating System: Abstraction

- Many abstractions provided by real-world operating systems

- Network Stack
  - Communicating with network devices ≈ communicating with screen/keyboard memory map
  - OS handles messy, time-sensitive protocols

- Processes
  - Only one process can run at once on a CPU
  - OS switches very quickly, illusion of running both “at once”

Operating System: Protection

- The CPU has different "privilege" levels when it is executing (controlled by a register on the CPU)

- OS code and memory can only be executed by an OS privilege level
  - Your applications run at a lower level and cannot access OS code and memory

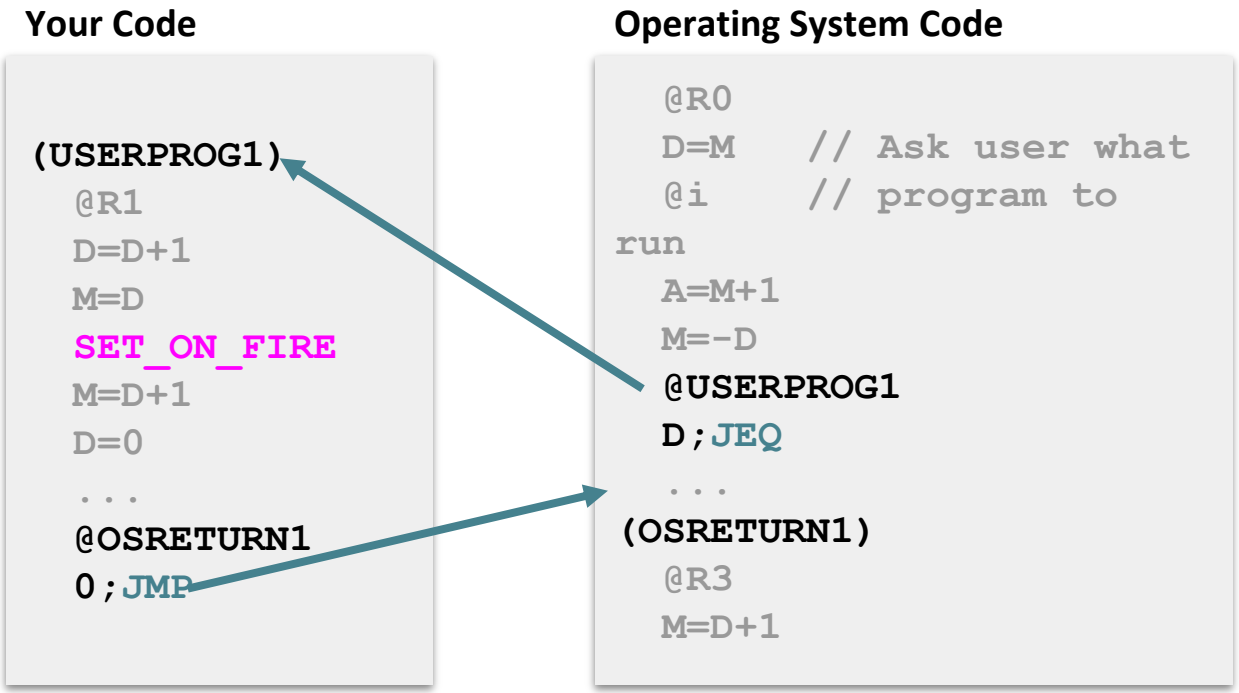
- This prevents applications from crashing entire system
  - For example, if your web browser crashes, usually it doesn't crash your entire computer
  - Also helpful for security purposes

Operating Systems: Protection

- Example: Suppose we want only the OS to be allowed to run instruction **`SET_ON_FIRE`**
  - But if the OS is just a machine code program like any other… what's the security hole?

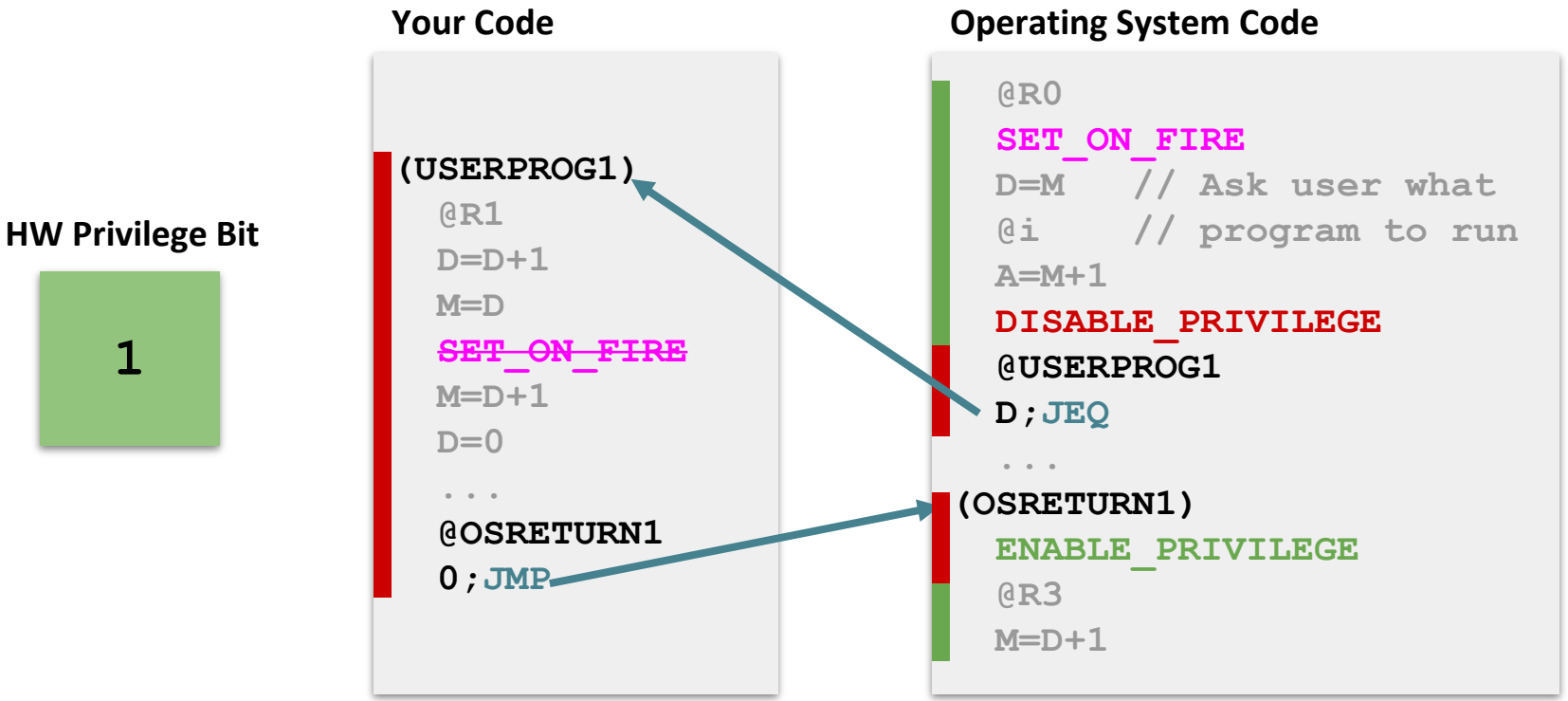
Operating Systems: Protection

- Example: Suppose we want only the OS to be allowed to run instruction **`SET_ON_FIRE`**
  - But if the OS is just a machine code program like any other… what's the security hole?



Operating Systems: Protection

- The fix: hardware bit for "privileged mode"
  - Processor checks before running SET_ON_FIRE
  - OS disables before jumping to user code, re-enables on return
    - (Processor also must check that user code can't enable privilege)



Operating System: Processes

- A "process" is an application running on your computer
  - E.g., your web browser, terminal, Microsoft Word, etc.
- Each app instance contained in one or more processes
  - The OS manages these processes
- Multiple processes are "running" at the same time, but it's just the OS quickly switching between them
- A process only has access to its memory, and cannot access the memory of other processes
  - This is helpful because if one process crashes or is malicious, it makes it more difficult to crash or corrupt other processes too

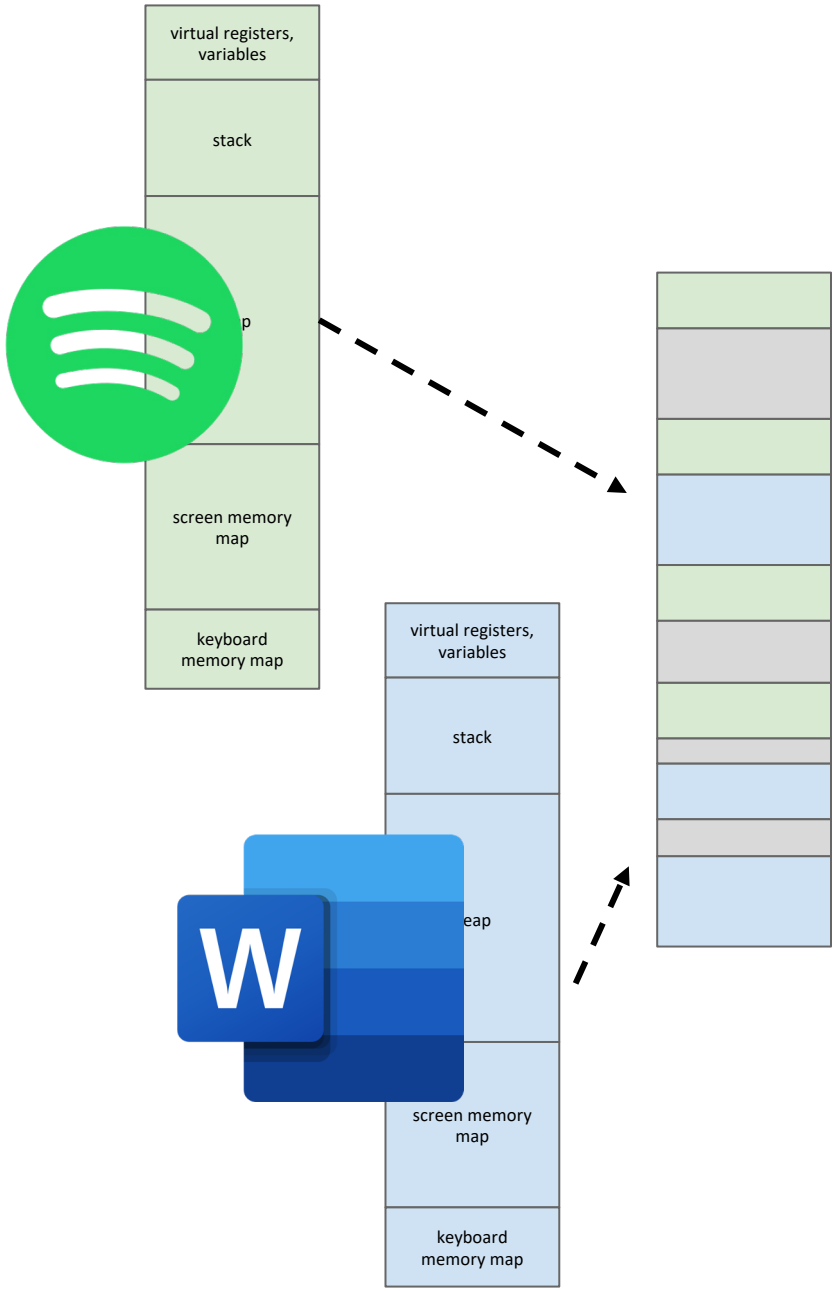
Why *Not* an Operating System?

- The Hack computer we've built is… small
  - Uses the same principles as your laptop CPU
  - But in terms of scale, closer to a microprocessor or small embedded chip

- For embedded systems, often an OS is overkill—instead, designed to be programmed with/run a single program at a time
  - Benefit: developer gets complete control over the device
  - Drawback: re-implement OS features, no protection

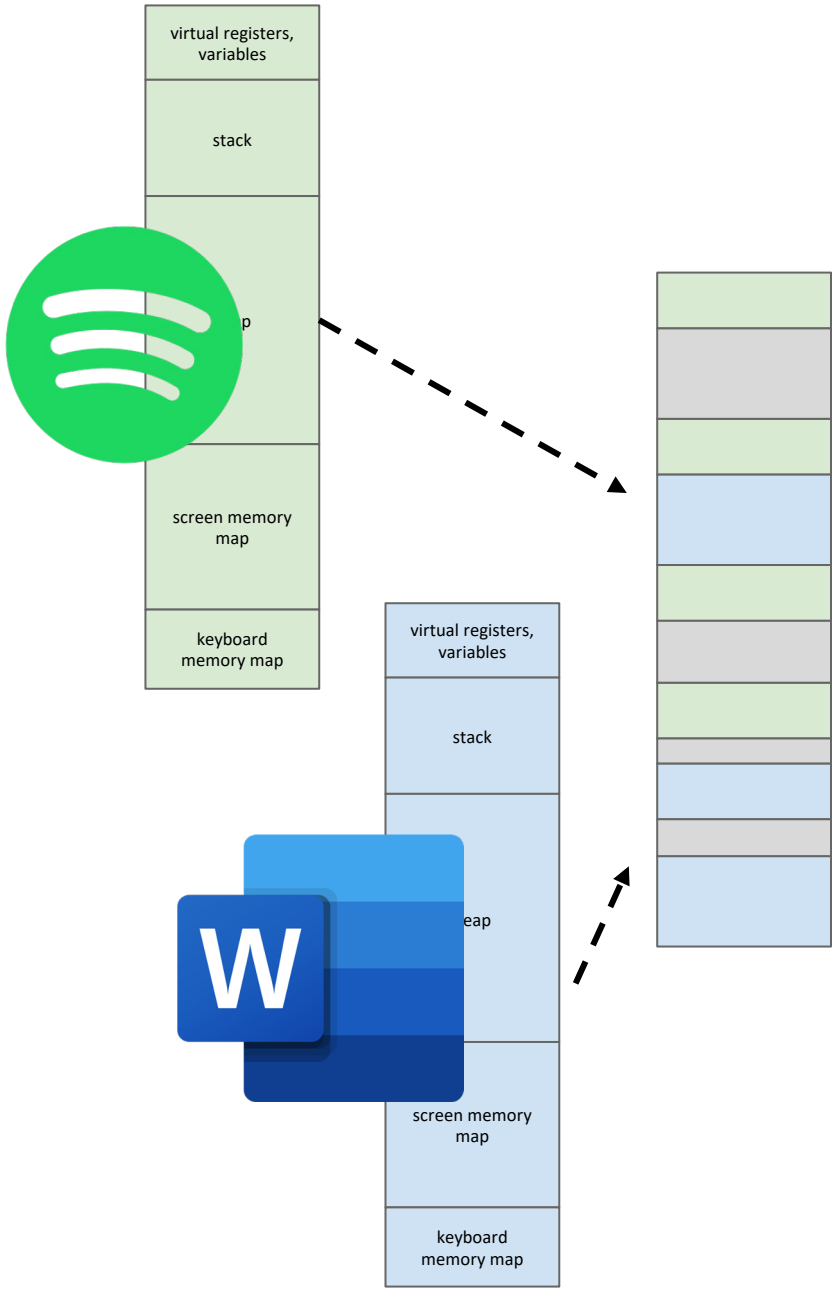
Virtual Memory

- Most OS's allow multiple processes, but shouldn't be able to modify values in another's address space
- OS provides illusion of separate address spaces via virtual memory
  - Really all one physical memory
  - OS & hardware map pieces of virtual memory to pieces of physical memory



Virtual Memory

- Benefit:
  - Security: programs only know about their own address space
    - Don't even have a way to describe address of other application's data
- Drawback:
  - Efficiency: virtual address translation is fast nowadays but still slower than directly accessing memory (what microprocessors do)



Comparison of Operating Systems

- Three different ways to do essentially the same thing
  - Everyone has their own preference
- Each have their own benefits and tradeoffs
  - Work on varying types of hardware, provide different levels of customization, different features, work better with different software, open source vs. proprietary, etc.
- You could choose to do some research next time you are deciding on a laptop, computer, or OS

Five-minute Break!

- Feel free to stand up, stretch, use the restroom, drink some water, review your notes, or ask questions
- We'll be back at:
- Research shows mid-lecture breaks reduce the decline of attention in the middle of lecture (Olmsted, 1999)

Olmsted III, John. "The Mid-Lecture Break: When Less Is More." *Journal of Chemical Education* (1999). https://pubs.acs.org/doi/abs/10.1021/ed076p525.

Lecture Outline

- The Software Stack
  - Roadmap of Hardware and Software Components
- Operating Systems Overview
  - Abstraction, Protection, Processes, Virtual Memory
- **Final Project Overview**
  - **E-Portfolio Details and Presentation Logistics**

Final Project E-Portfolio Overview

- You will create an E-Portfolio that is geared toward a new Allen School student
- Your E-Portfolio is a culminating project in having you reflect on the **metacognitive skills** you've learned and **providing advice** for entering the program
- During our final class, you will give an 8–10-minute presentation on your E-Portfolio

Final Project E-Portfolio Components

- Part I: Outline of E-Portfolio
  - Final project check-in and feedback during next Tuesday's class
  - Due next Thursday (6/2) at 11:59pm PDT
- Part II: Final E-Portfolio Submission
  - Due Tuesday of finals week (6/7) at 4:00pm PDT
- Part III: E-Portfolio Presentations
  - During the scheduled CSE 390B final (Tuesday, 6/7 at 4:30pm PDT)
  - We'll split up into small groups with your TA

Final Project Ideas Brainstorming

- Brainstorm individually for five minutes:
  - Looking back on this quarter, what metacognitive skills impacted you the most?
  - What were some concrete examples of yourself demonstrating these metacognitive skills?
  - What was one technical topic from CSE 390B that helped you connect the dots?
- Discuss in groups of 3-4 for five minutes

Lecture 17 Wrap-up

- Thursday's Lecture: Procrastination, Stress Response Cycle, & Computer Networks
  - Reading: Feminist Survival Project Episode 2

- Upcoming Project Reminders
  - Project 7, Part II (Professor Meeting Report) due this Thursday (5/26) at 11:59pm PDT
  - Project 8 (Debugging & Implementing a Compiler) due next Tuesday (5/31) at 11:59pm PDT

- Check Canvas for late updated late days through Project 5