CSE 390B, Spring 2023

Building Academic Success Through Bottom-Up Computing

Finals Preparation & Operating Systems

Gearing up for Finals Week, The Software Stack, Overview of Operating Systems, Final Project Overview

Lecture Outline

- **Gearing up for Finals Week**
  - **Study Plan Outline and Tips for Success**
- The Software Stack
  - Roadmap of Hardware and Software Components
- Overview of Operating Systems
  - Abstraction, Protection, Processes, Virtual Memory
- Final Project Overview
  - E-Portfolio Details and Topics Brainstorming

Gearing up for Finals Week

- Revisit and reassess your goals each day
  - Break-up into different levels—safe, solid, reach
- Have an accountability buddy
  - Study groups or working sessions—having someone who can help you stay motivated, accountable, and avoid procrastination
- Recall Bloom's Taxonomy
  - How is your preparation involving higher level thinking skills?
- Stick to a routine
  - Provides normalcy & structure for maintaining sleep and wellness

Developing a Plan for Finals Week

- First, list the commitments that you have for finals week (final exams, final projects, presentations, etc.)

- Then, outline the steps that you'll need to follow to complete those tasks
  - Be specific with these steps — instead of "review derivatives," add detail about the **how**: "Review lecture slides and examples on derivatives and redo five derivative problems on WebAssign"

- Lastly, add dates to when you will work on and complete each of the steps (be realistic here!)
  - Add these to your calendar

Planning for Success on Finals Week

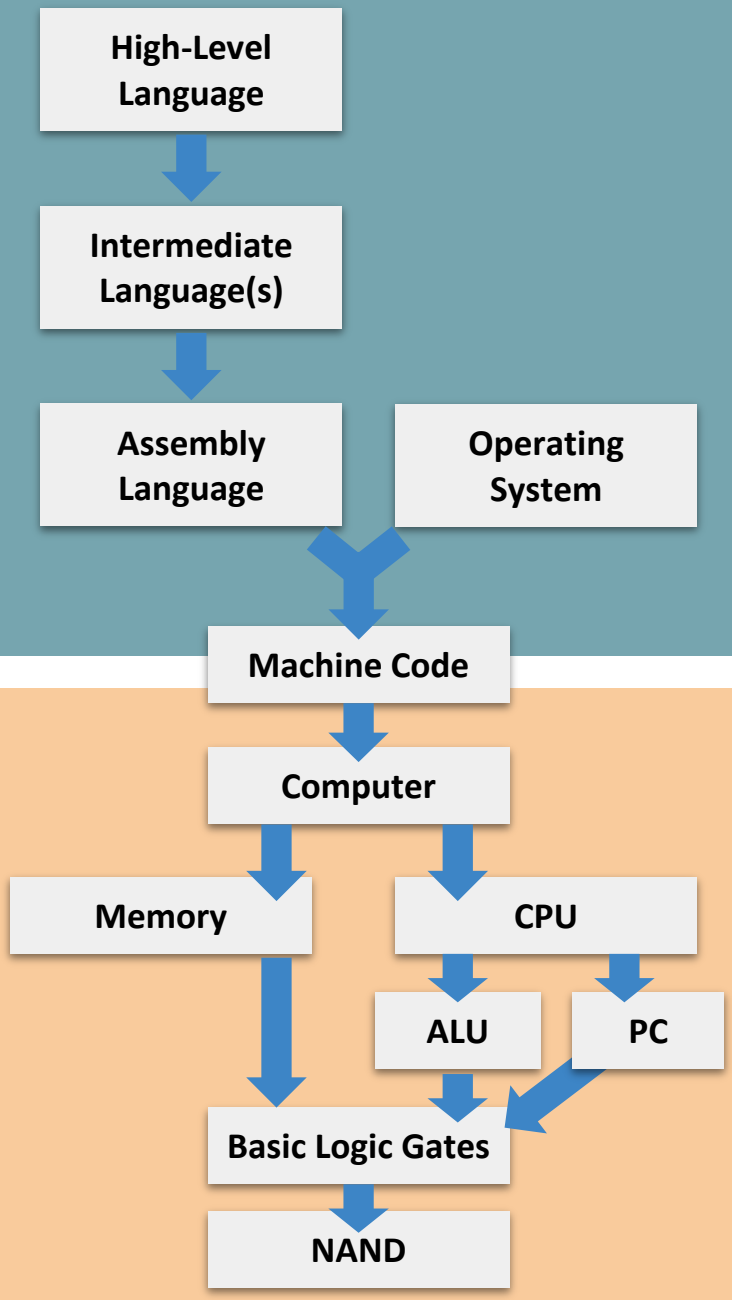
In groups, discuss the following questions:

- What are some metacognitive strategies that you plan on using to succeed in finals week?
  - How can you stay disciplined or keep yourself accountable in applying these metacognitive skills?
- In terms of academic and metacognitive subjects, what are your strengths and weaknesses going into finals?
  - How can you cultivate your strengths and improve the areas you are weaker in?

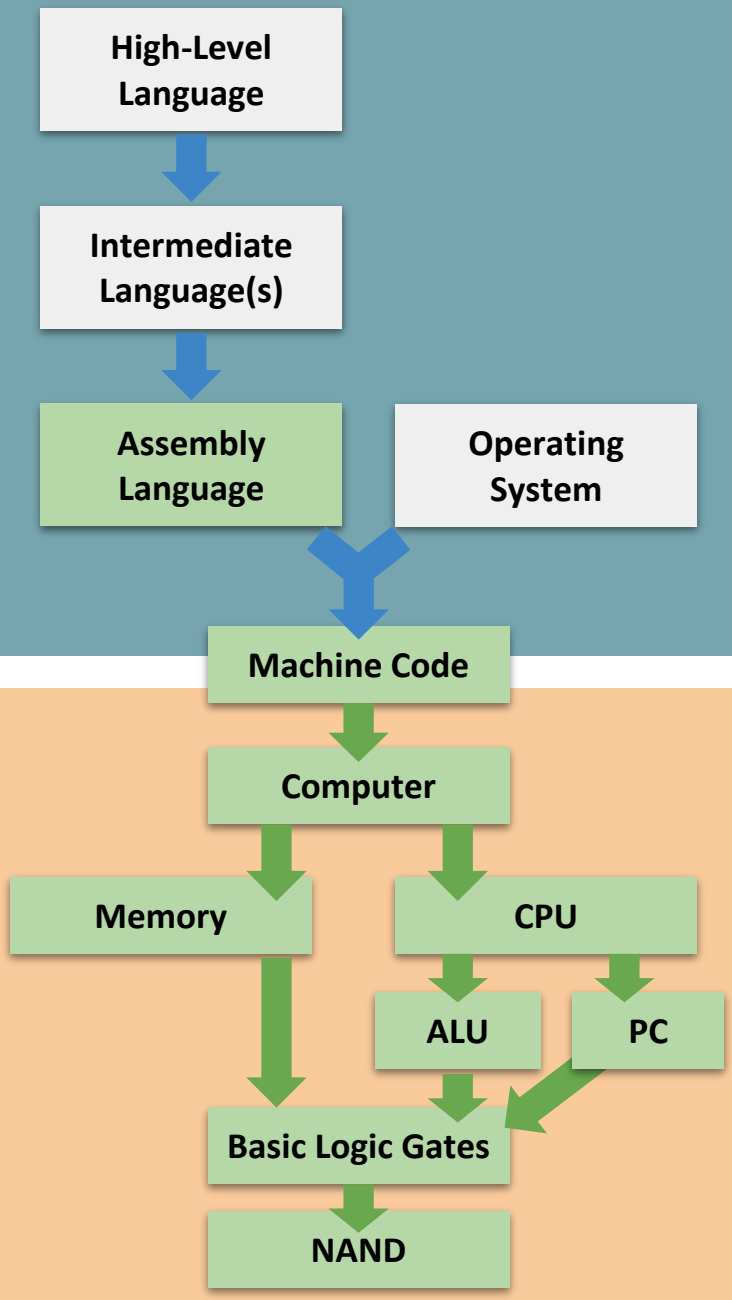
Lecture Outline

- Gearing up for Finals Week
  - Study Plan Outline and Tips for Success
- **The Software Stack**
  - **Roadmap of Hardware and Software Components**
- Overview of Operating Systems
  - Abstraction, Protection, Processes, Virtual Memory
- Final Project Overview
  - E-Portfolio Details and Topics Brainstorming

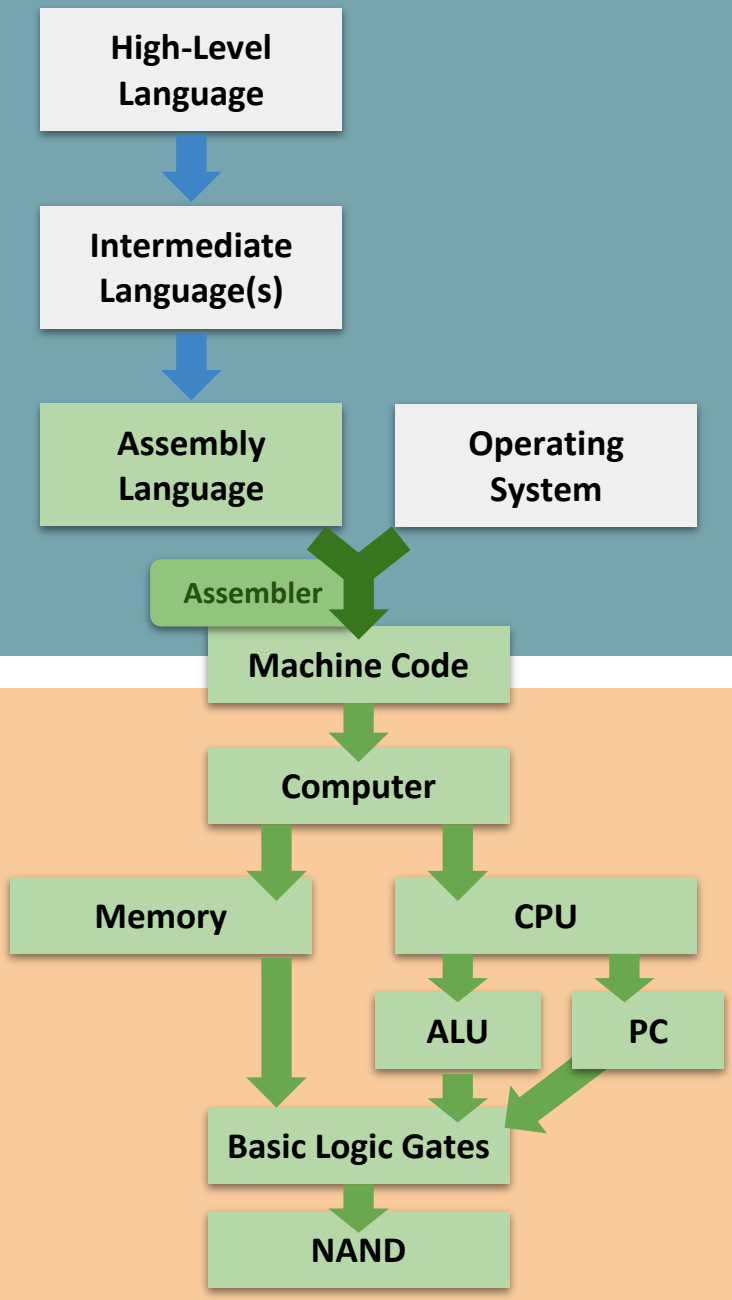
Roadmap


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

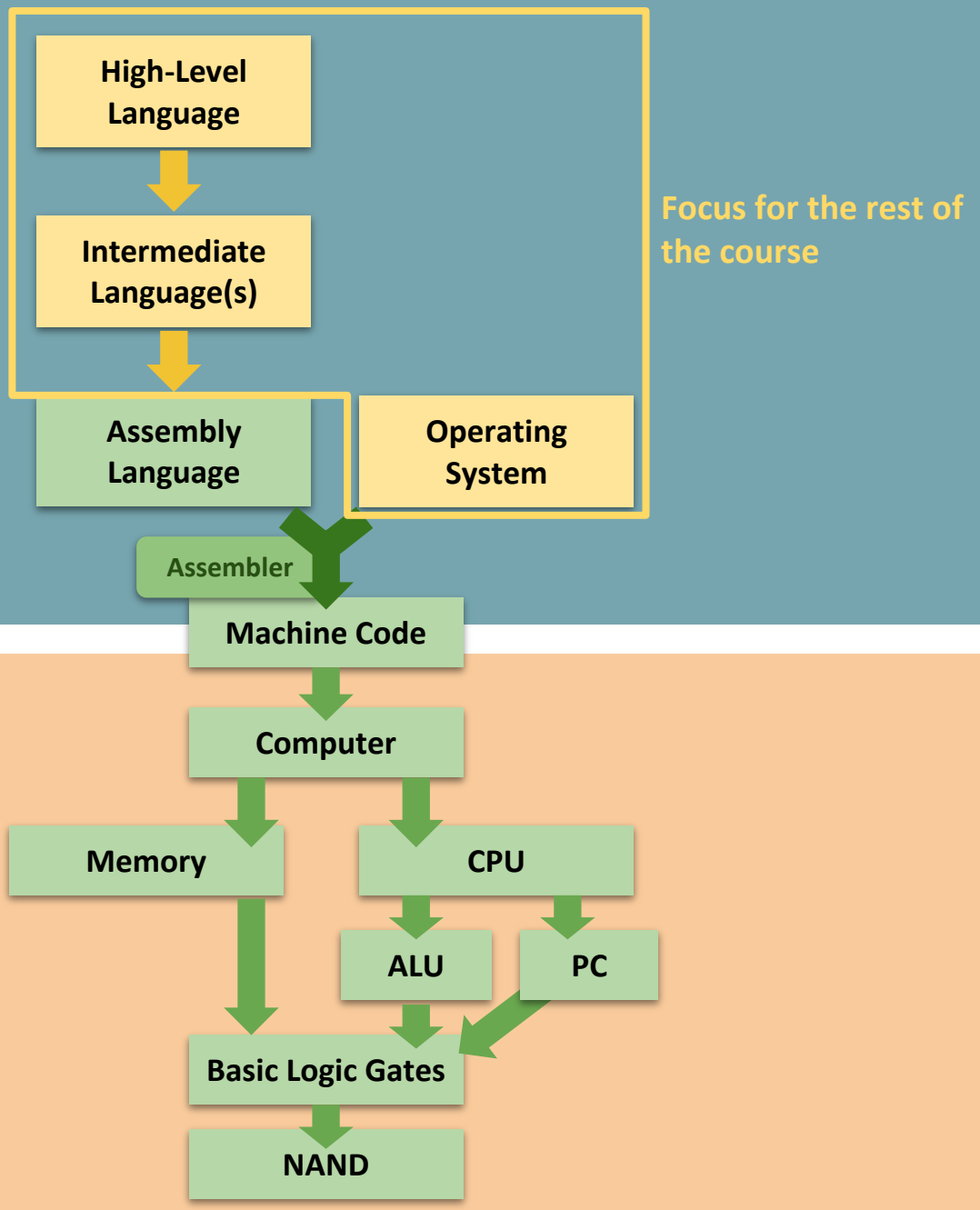
Roadmap


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

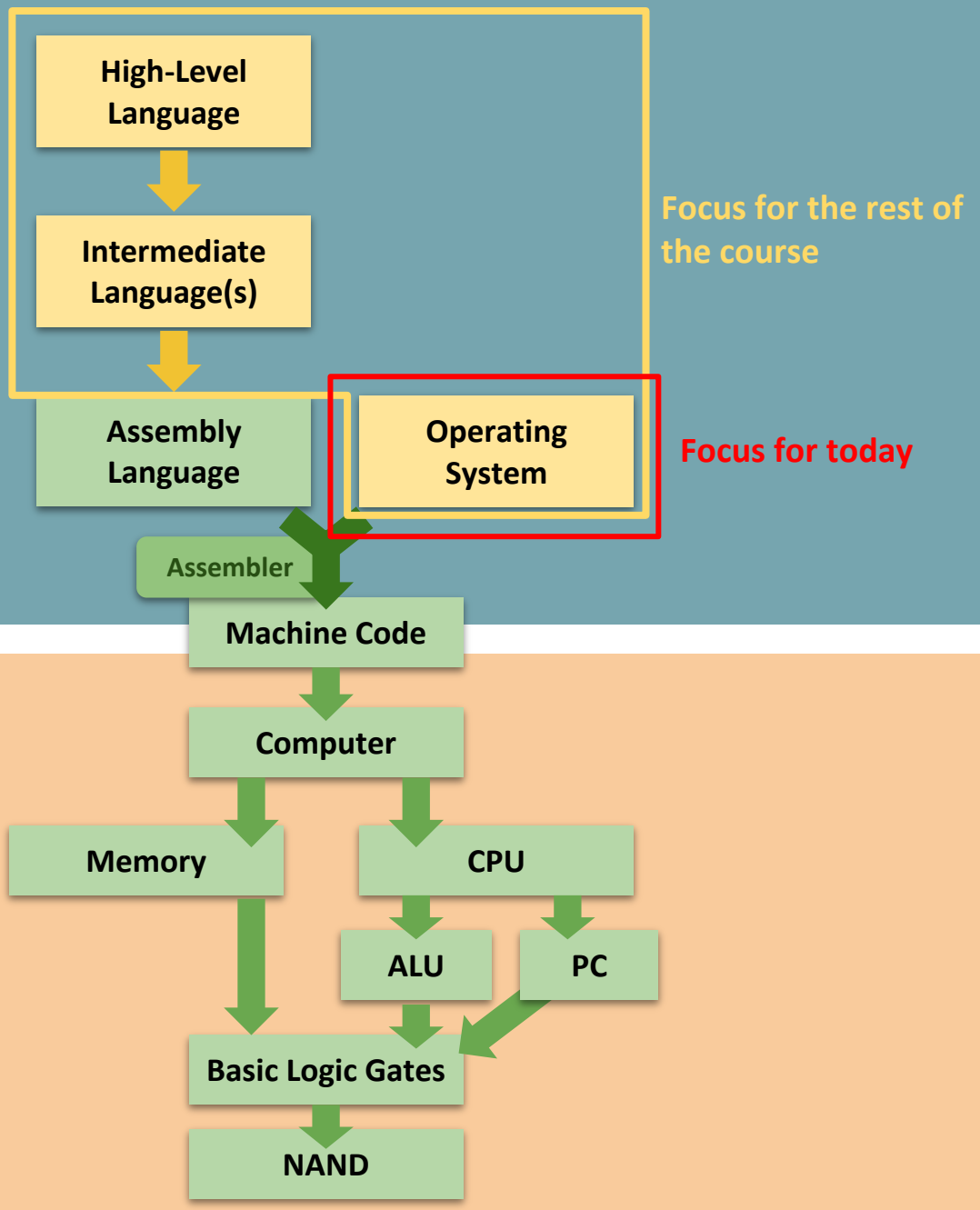
Roadmap


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

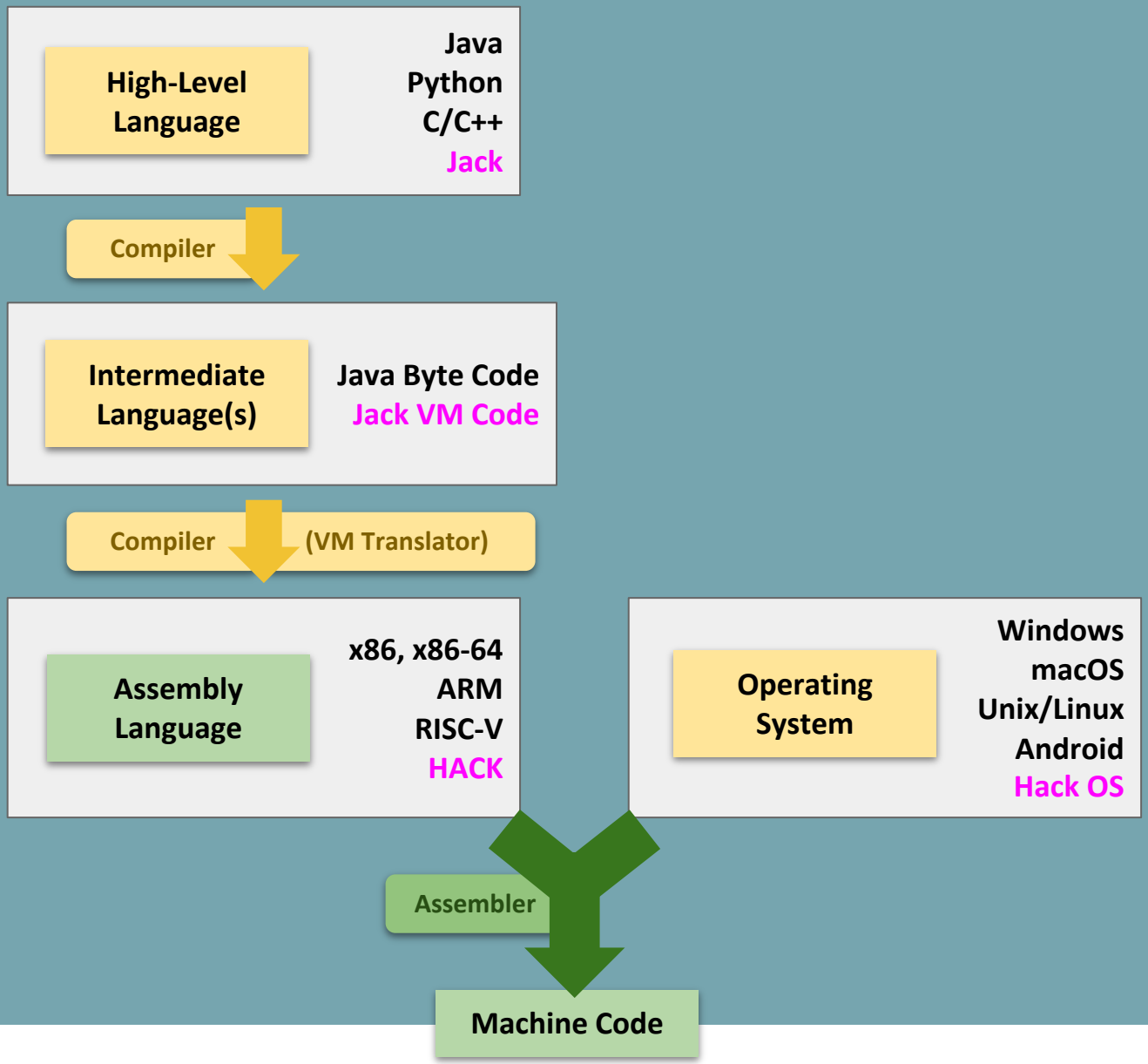
Roadmap


High-Level Language
Intermediate Language(s)
Assembly Language
Operating System
Focus for the rest of the course
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

- Gearing up for Finals Week
  - Study Plan Outline and Tips for Success
- The Software Stack
  - Roadmap of Hardware and Software Components
- **Overview of Operating Systems**
  - **Abstraction, Protection, Processes, Virtual Memory**
- Final Project Overview
  - E-Portfolio Details and Topics Brainstorming

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
  - Prevents user-level programs from causing errors in the hardware
  - Maintains security between programs and user accounts

Operating Systems: 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 Systems: 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
  - Operating systems can manage resource sharing
  - OS switches very quickly, illusion of running both “at once”

Operating Systems: 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 higher 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?

Your Code

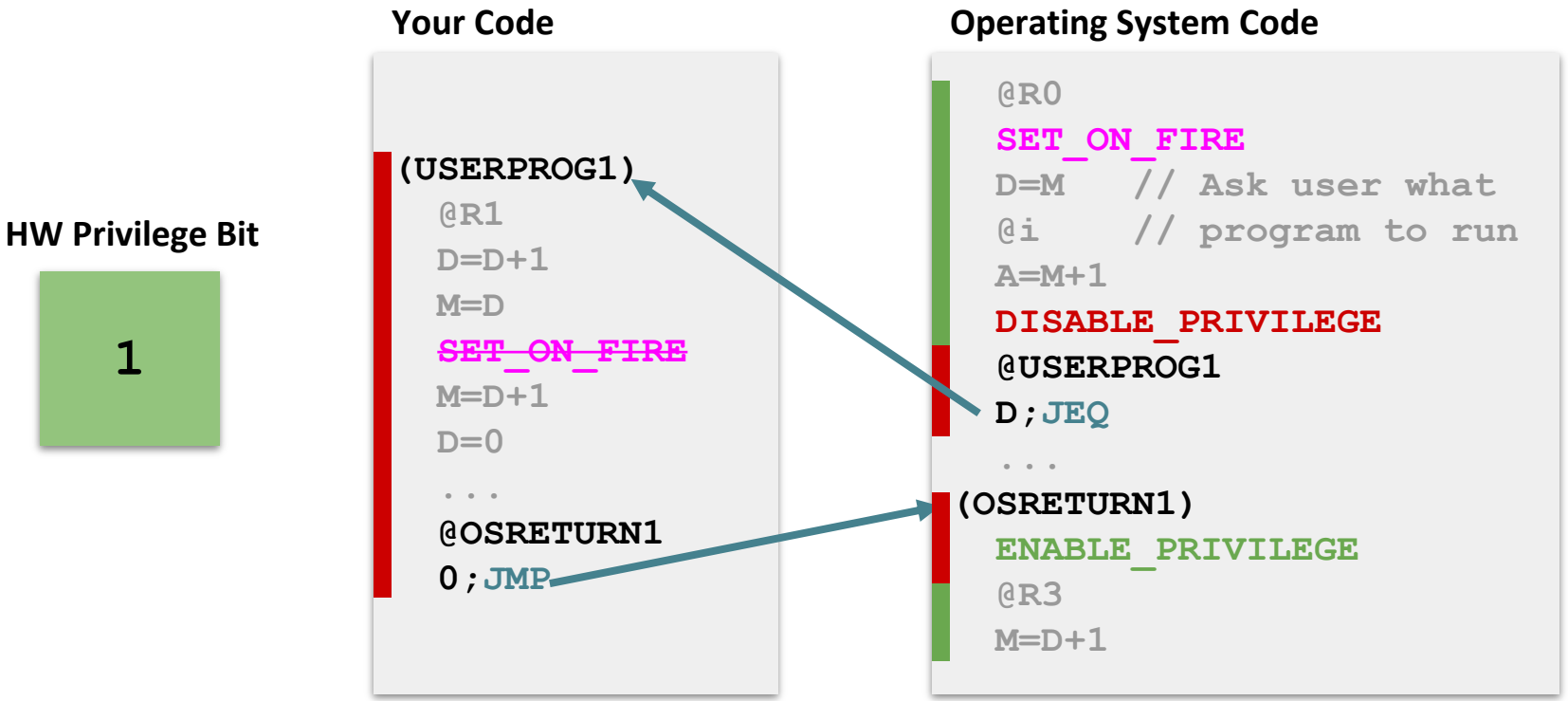
```
(USERPROG1)
  @R1
  D=D+1
  M=D
  SET_ON_FIRE
  M=D+1
  D=0
  ...
  @OSRETURN1
  0;JMP
```

Operating System Code

```
  @R0
  D=M    // Ask user what
  @i     // program to
run
  A=M+1
  M=-D
  @USERPROG1
  D;JEQ
  ...
(OSRETURN1)
  @R3
  M=D+1
```

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

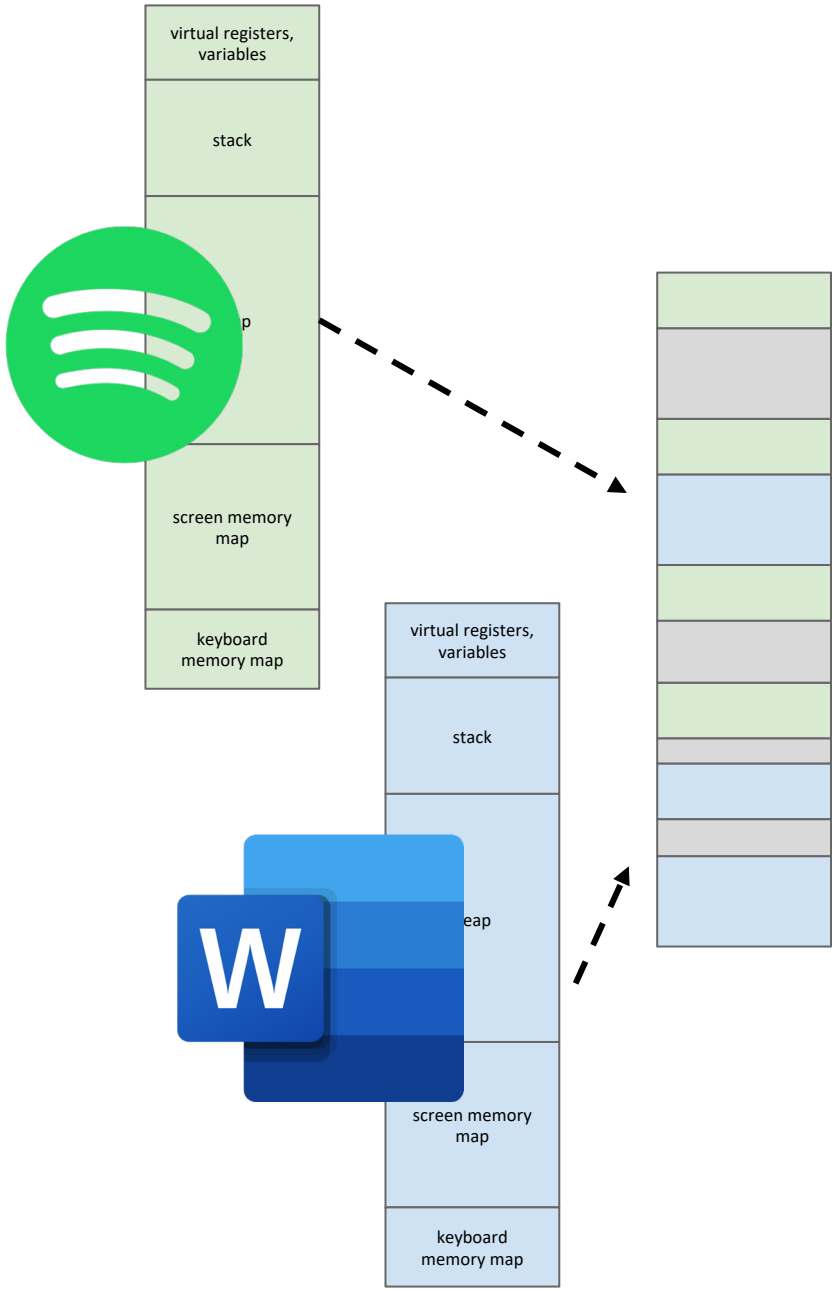
Why *Not* an Operating System?

- 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
  - Pro: developer gets complete control over the device
  - Con: re-implement OS features, no protection

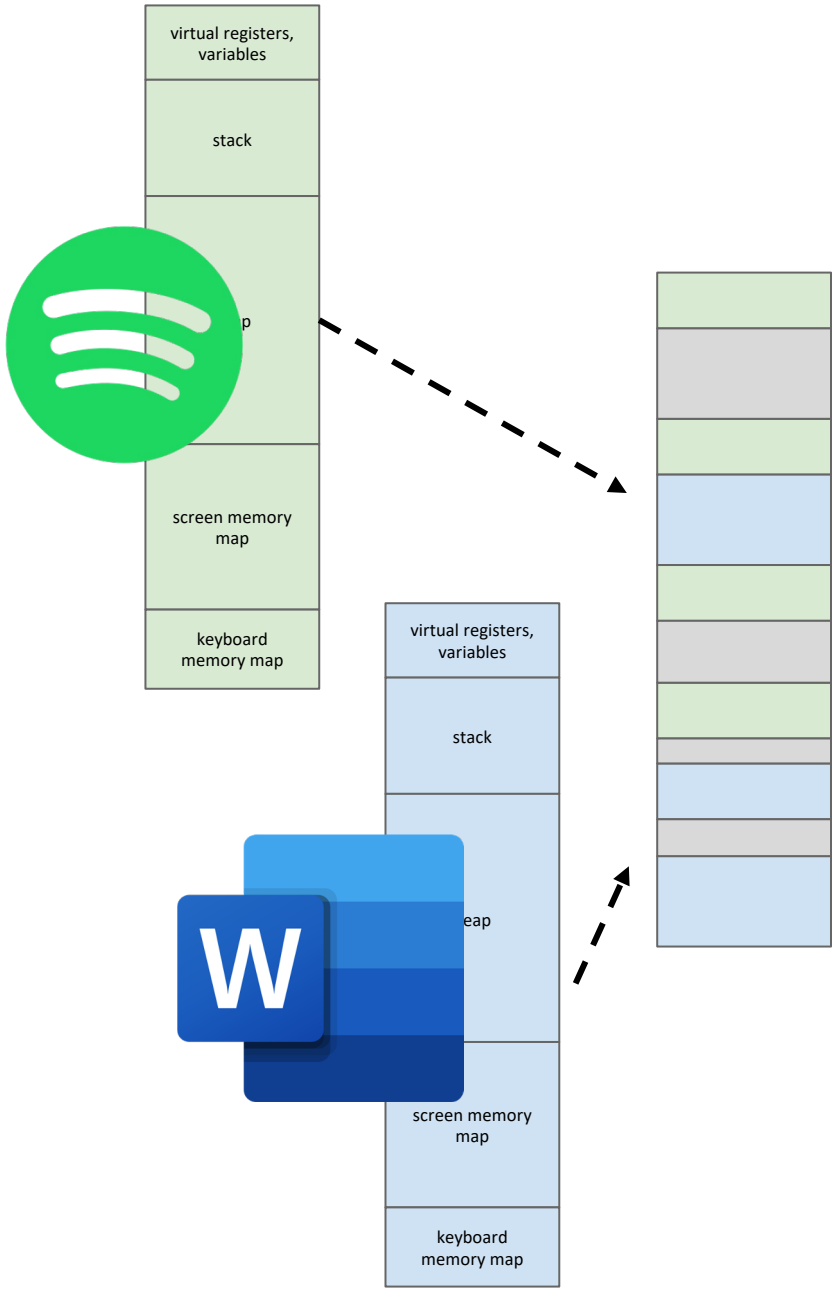
Virtual Memory

- 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 of security**: Programs only know about their own address space
  - Don't even have a way to describe address of other application's data
- **Drawback is efficiency**
  - Virtual address translation is fast nowadays, but still slower than directly accessing memory



Comparison of Operating Systems

- Three different ways to do essentially the same thing
  - Everyone has their own preference
- Each has 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

Lecture Outline

- Gearing up for Finals Week
  - Study Plan Outline and Tips for Success
- The Software Stack
  - Roadmap of Hardware and Software Components
- Overview of Operating Systems
  - Abstraction, Protection, Processes, Virtual Memory
- **Final Project Overview**
  - **E-Portfolio Details and Topics Brainstorming**

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 a short presentation on your E-Portfolio

Final Project Due Dates

- Part I: E-Portfolio Outline
  - Due next Tuesday (5/30) at 11:59pm
- Part II: Final E-Portfolio
  - Due Tuesday of finals week (6/6) at 4:00pm
- Part III: E-Portfolio Presentations
  - During the scheduled CSE 390B final
  - CSE 390B Final Time: Tuesday, 6/6 from 4:30-6:20pm
  - CSE 390B Final Location: CSE2 G01 (same as usual classroom)

Reflection on Metacognitive Skills

Individually first, take some time to reflect on the following questions, and then discuss in groups:

- Which two metacognitive topics would you consider including in your E-Portfolio and why?
  - Reflect on which ones you've grown the most in, have impacted you the most, were most challenging to grow in, etc.
- What are some examples of yourself demonstrating those two metacognitive skills?
  - Please be specific here! Aim to share these skills as if you are telling a story and showing concrete applications of these skills

Reflection on a Technical Skill

Individually first, take some time to reflect on the following questions, and then discuss in groups:

- What technical topic from CSE 390B would you consider including in your E-Portfolio and why?
  - Reflect on technical skills that helped connect the dots, were most interesting to you, most challenging for you to grasp, etc.
- What is the impact of having knowledge of that technical skill? In other words, why is that technical skill useful?
  - Please be specific here as well — think about how this technical skill would be useful in an academic or personal setting

Post-Lecture 17 Reminders

- Project Reminders
  - **Project 7, Part II: Professor Meeting Report due this Thursday (5/25) at 11:59pm**
  - Project 8: Debugging & Implementing a Compiler due next Tuesday (5/30) at 11:59pm
  - Final Project, Part I: E-Portfolio Outline due next Tuesday (5/30) at 11:59pm

- Preston has office hours after class in CSE2 152
  - Feel free to post your questions on the Ed board as well