Week 7

Announcements

- Shell assignments due February 23
- Regex assignments due March 9

Lecture 7: Potpourri

Overview

- How do computers understand us?
- Fields and jobs in EECS
  - Internships

How do computers understand us?

Quick rundown on comp arch

Parts

- CPU
  - The star of the show: handles general workloads
- Memory (RAM)
- Storage (HDD, SSD)
- Accelerators
  - GPU
- Motherboard
  - Connects everything together

How do computers work?

- The CPU executes a sequence of instructions in the form of binary
  - This is known as **machine code**
- The **instruction set architecture** defines what instructions there are and how they are encoded in binary
  - Yes, even CPUs have their own languages
  - Example ISAs: x86-64, ARMv8, RISC-V, MIPS
- CPUs perform relatively rudimentary operations
  - Arithmetic: `add`, `sub`, `mul`, `div`
  - Control: `br`, `jmp`
  - Memory: `ld`, `st`
  - (These are generic instruction names)
- Example operation: "add 5 to variable"
  - x86-64: `[0x48,0x83,0xc0,0x05]`
  - AArch64 (ARMv8): `0xe2800005`

How do computers understand us?

- Writing a program in machine code with a hex editor is not most people's idea of fun
- **Assembly languages** are human readable representations of machine instructions
  - x86-64: `add rax, #5` -> `[0x48,0x83,0xc0,0x05]`
  - AArch64 (ARMv8): `add r0, r0, #5` -> `0xe2800005`
- An **assembler** takes assembly code and turns it into machine code
- Assembly is still rather cumbersome to use
  - You're still working with machine instructions
  - You just now you have a tool that remembers how instructions are encoded
  - This isn't portable: assembly for one ISA won't work for another

How do computers understand us?

- Higher level languages were developed
  - Raises abstraction level: can think in arithmetic expressions and higher level constructs
  - Leads to portability since you don't have to worry as much about hardware/ISAs
- "Classic" classes
  - Compiled
  - Interpreted
- Things get a bit hazy
  - Byte-code
  - Just-in-time (JIT) compilation
- Keep in mind languages don't necessarily need to fit any one classification

Compiled languages

- "Compilation" in general refers to taking the language and producing a lower level form
- Compiled languages *usually* have the source code compiled down to the machine code level
- The CPU can directly run this machine code
  - This is fast
- Examples: C, C++, Fortran, Go, Objective-C, Rust, (Java*)

Interpreted languages

- These languages *usually* are read by another program known as an **interpreter**
  - This program will parse statements and do something based off of them
  - This is understandably slow and there are other strategies to improve performance
- Examples: Shell, Bash, Python, Ruby

Lines get hazy

- Some languages compile down to **byte-code** that is meant to be efficiently interpreted
  - Java is a poster child
- Some languages have interpreters that do **just-in-time (JIT)** compilation
  - Source code is compiled to machine code at runtime
  - Traditional compilation is known as **ahead-of-time (AOT)** compilation in contrast
- Ultimately these aren't really hard classifications
  - You can compile Python if you want
  - You can write a C interpreter if you want

Running a program

- On *nix systems you can execute both machine code and interpreted programs
  - Machine code programs are put into file formatted with data pertaining to execution known as an "executable"
  - The execute permission bit must be set for you to execute a program file
- Executable file formats have information that can identify their file type
- *nix systems will recognize the shebang (`#!`) on the first line to figure that it's an interpreted program, then call the appropriate interpreter
  - If there's no shebang, the OS will generally assume it's a shell script
  - `#!/bin/env python3`: invoke the `/bin/env` executable to find the `python3` executable
  - `#!/bin/bash`: invoke Bash as the interpreter

Questions?

Fields and jobs in EECS

EECS is a **very** broad subject

- No wonder when you have EE, CE, and CS all in one acronym
- There's a lot of different ways to divy up the space: this is just my conception of it
- I'll keep it *computer*-related
  - Sorry, analog and power EEs
- Not an all-encompassing list

Domains

Computer hardware

- Physical design engineer
- RTL/ASIC design engineer
- Design verification engineer
- Computer architect

Hardware-software twilight zone

- Embedded systems engineer
- Firmware engineer

Domains

Computer systems

- Systems software engineer
- Compiler engineer
- Embedded software engineer

Domains

Applications

- Desktop software developer 🦕
- Mobile app developer 📱
- Front-end web developer
- Back-end web developer
- Full-stack developer
- Software engineer*
- DevOps engineer
- Testing/QA engineer

Domains

Data science

- Data analyst
- Data engineer
- Machine learning engineer

Other areas

- Cybersecurity
- Robotics
- Scientific computing/simulation
- Research

Industries

- You can find these jobs in various industries
  - Tech
  - Gaming
  - Finance
  - Telecommunications
  - Defense
  - Government
  - Health care
  - and more (you'd be surprised at the opportunities in the restaurant industry)

Internships

- Great way to get experience and build up your resume
  - I learned a lot of my practical skills on the job
- Job postings will go up during the Fall and early Winter: try to nab them!
  - That doesn't mean all hope is lost if you're applying in late February/March
  - I got my internships as a "late" applicant
  - **It doesn't cost anything to apply, unlike with college applications**
- Career fairs are not *the* only way to apply
  - You can still find success in online applications
- Interviews often have a behavioral and technical portion
  - Behavioral interview will involve questions about you as a person
  - Technical interview will test your knowledge
  - Getting an answer wrong on a technical interview isn't the end of the world: often the interviewer is just as interested in your problem-solving process

Internships

- If your summer is open, try looking for an internship!
- If you have companies in mind, check the job postings on their website
- You can use websites like Indeed, Glassdoor, and LinkedIn to browse for jobs and see what's around
- You might find something you like based on the description!
- r/cscareerquestions can be a useful place to get some opinions on how the job market is doing

Questions?