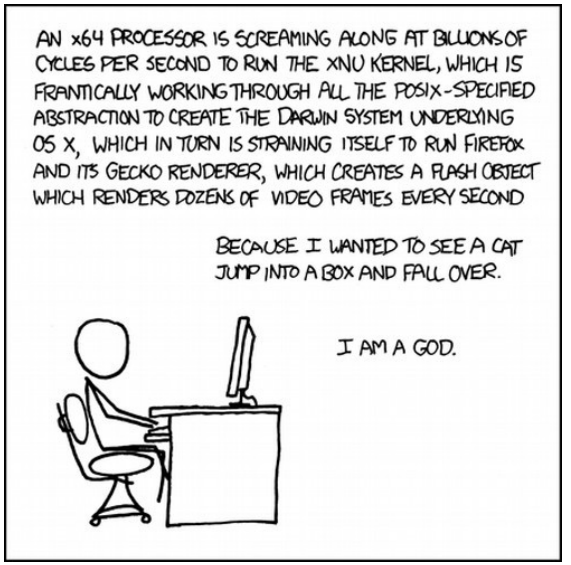
CS 261
Fall 2020

Mike Lam, Professor



Computer Systems I: Introduction

Welcome! As you enter our virtual classroom:

- Please turn on your webcam! :)
- Feel free to unmute to say hello (or type it in chat)
- Open Canvas and find today's lab ("Lab 01 - Intro")

CS 261
Fall 2020

Mike Lam, Professor

https://xkcd.com/676/

Computer Systems I: Introduction

Question

- What will be the output of this C program?

```
#include <stdio.h>
int main() {
    int x = 40000;
    int y = 50000;
    if ((x * x) < (y * y)) {
        printf("Less than\n");
    } else {
        printf("Not less than\n");
    }
    return 0;
}
```

- A) “Less than”
- B) “Not less than”
- C) Neither of the above

Question

- What will be the output of this C program?

```
#include <stdio.h>
int main() {
    double a = 1e20;
    double b = -a;
    double c = 3.14;
    if (((a+b) + c) == (a + (b+c))) {
        printf("Equal!\n");
    } else {
        printf("Not equal!\n");
    }
    return 0;
}
```

- A) “Equal!”
- B) “Not equal!”
- C) Neither of the above

Question

- Which of the following versions of a “matrix copy” routine will run the fastest?
  - A)
```
for (int i = 0; i < 2048; i++) {
    for (int j = 0; j < 2048; j++) {
        dst[i][j] = src[i][j];
    }
}
```
  - B)
```
for (int j = 0; j < 2048; j++) {
    for (int i = 0; i < 2048; i++) {
        dst[i][j] = src[i][j];
    }
}
```
  - C) Neither; they will always run at approximately the same speed.

What's happening?

- Something about our **mental model** of these programs does not match the **system** on which we're running them.

Systems

- What is a “system?”

Systems

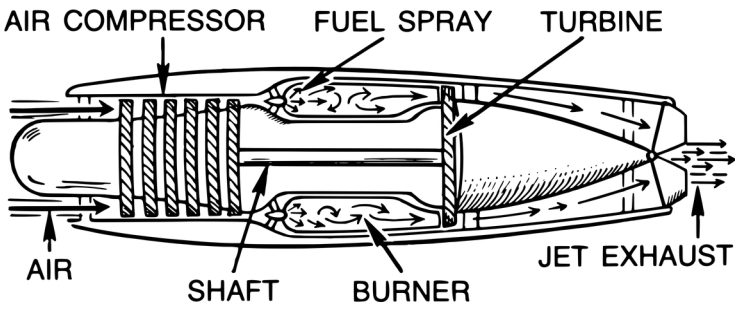
- What is a “system?”
  - Set of interacting components
  - More than the sum of its parts

Jet engine

Computer

Systems

- What is a “system?”
  - Set of interacting components
  - More than the sum of its parts

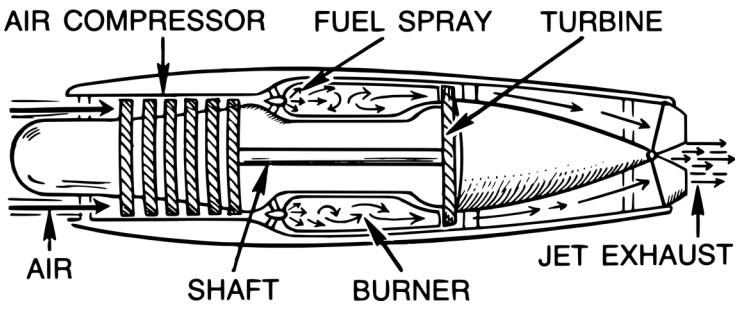


Jet engine

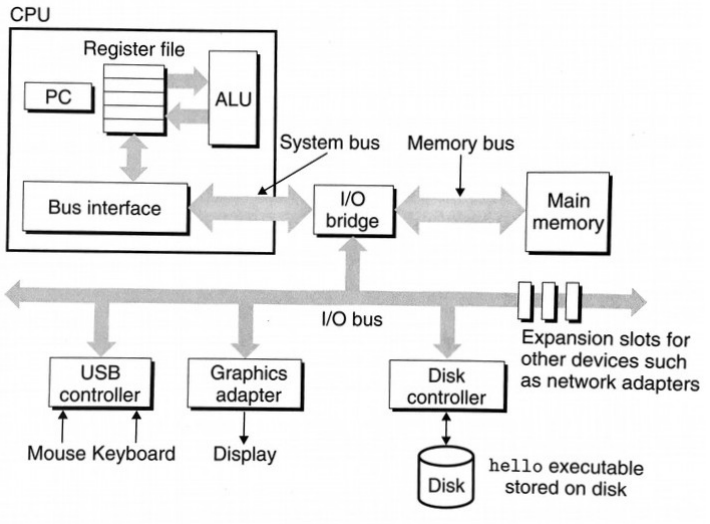
Computer

Systems

- What is a “system?”
  - Set of interacting components
  - More than the sum of its parts



Jet engine



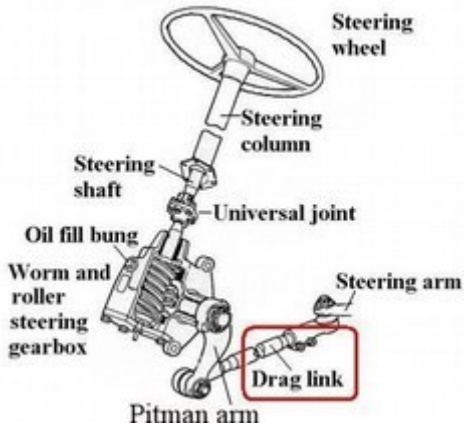
Computer

Systems

- A **computer system** consists of multiple hardware and software components that work together to run user applications.
  - We use complex computer systems every day
  - Our goal: peel back some of the complexity
    - See (some of) what's "under the hood"

Systems

- What is a *process*? What is a *file*?
  - These are examples of abstraction; "fake" views of reality that reduce complexity for users
  - Key ideas: **ignore details** and **focus on interfaces**
  - Especially important in large, complicated systems
  - Understanding abstractions can improve your ability to use them effectively



abstraction

Caveat

- Software system vs systems software
  - Former: interconnected software components
  - Latter: software providing services to other software
  - We are concerned with both!
    - Examples: multiprocessing, networking, operating systems, compilers, distributed systems

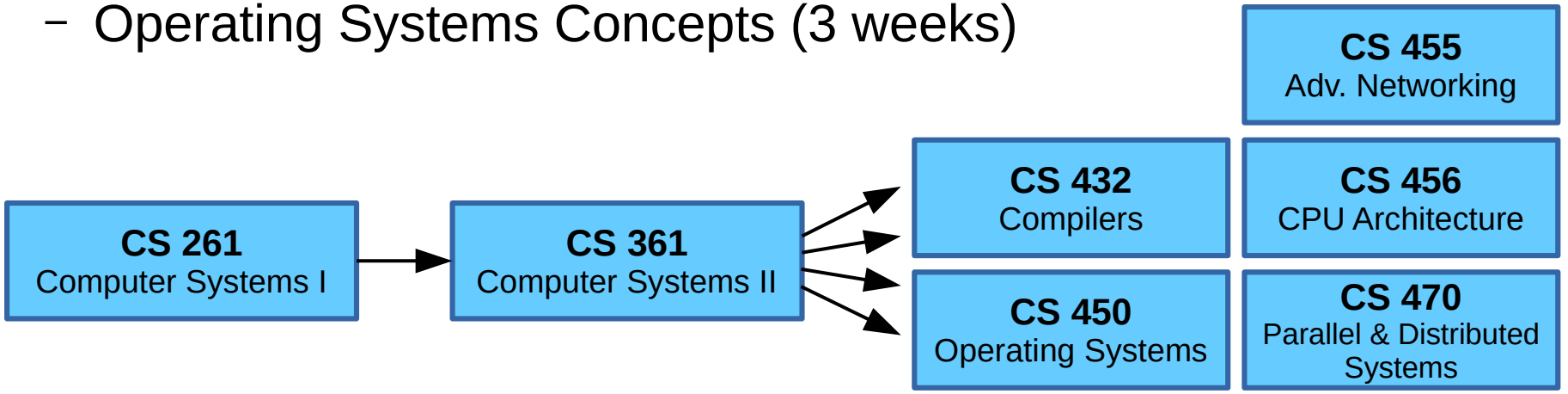
Course Objectives

- Explain machine-level representation of data and code
- Summarize the architecture of a computer
- Explain how complex systems are built from simple components
- Translate high-level code into assembly and machine language
- Write code to emulate the functionality of a computer

- Cultivate a sense of control over computer systems
- Gain an appreciation for software development tools
- Develop a sense of play when writing code
- Appreciate the complexity of systems-level software

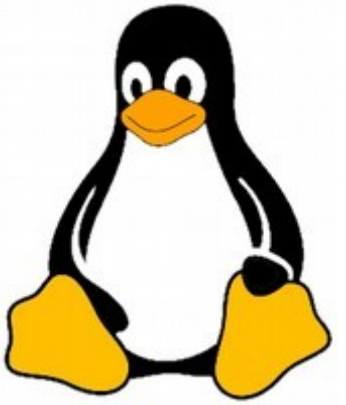
Systems courses

- CS 261 units:
  - C and Linux (3 weeks)
  - Binary Representations (2-3 weeks)
  - Assembly and Machine Code (2-3 weeks)
  - Computer Architecture (3 weeks)
  - Operating Systems Concepts (3 weeks)



CS 261

- What this course is NOT:
  - Programming 101 – I will assume you can program
    - However, we will spend a few weeks learning C
  - Electronics 101 – we won't be going THAT deep
    - If you're interested, check out PHYS 240/250
  - Linux 101 – but you have the Unix Users Group
    - InstallFests and weekly meetings
    - https://www.jmunixusers.org



CS 261

- This is not an "**easy**" course
  - *But you **can** handle it!*
  - Be prepared to **read** and **work** a lot
  - Don't be afraid to experiment
  - Growth vs. fixed mindset
  - Some stuff is worth memorizing
    - (e.g., powers of two and hex characters)
  - For other stuff, **Google** is your friend
  - **Piazza** is also your friend (literally)
  - Start assignments **early** and ask questions

Semester-specific info

- The remaining slides are specific to Fall 2020
- This is a challenging semester
  - I pledge to extend extra grace and patience
  - I ask that you do the same
  - This course is fully online due to physical distancing requirements; it is impossible to do effective small-group lab work in a “hybrid” class
- This is a **flipped class** (except for today)
  - Ahead of time: watch lecture, do reading, take quiz
  - During class: work on labs in small groups

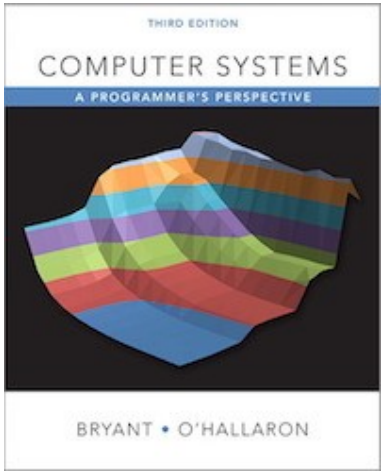
Course Components

- **Canvas course**
  - Quizzes, unit tests, and exams
  - Grades and private files (e.g., lab solutions)
  - Links to everything else
- **Public website** (`w3.cs.jmu.edu/lam2mo/cs261`)
  - Syllabus, **calendar**, project descriptions, and resources (links)
- **YouTube**
  - Lecture videos (watch these **well ahead of time** before doing the reading and quiz)
- **Zoom**
  - Class periods (**flipped classroom** – primarily for working on labs)
- **Google Drive**
  - Whiteboards (breakout room sign-ups) and lab documents
- **Student server** (`stu.cs.jmu.edu`)
  - Project development and submission
- **Piazza**
  - Q&A (especially re: projects)

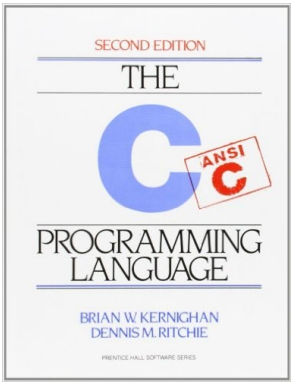
Make sure you can access ALL of these!

Textbook(s)

- Required textbook: “Computer Systems”
  - “CS:APP” textbook from Carnegie-Mellon
  - A practical, example-filled introduction to systems
  - Reserve copy at the Rose library



- Recommended book: "The C Programming Language"
  - **Brian Kernighan** and **Dennis Ritchie** (creator of C)
  - This is “the book” about C
  - Available on Safari Books through the library



Course Grades

Quizzes and Labs	25%
Programming Projects	30%
Unit Tests	25%
Exams	20%

- Quizzes and labs are **formative**
  - Designed to help you learn
- Tests/exams are **summative**
  - Designed to assess what you have learned
- Projects are **both**
  - Designed to help you learn C and reinforce other course concepts
  - Also designed to assess whether you are ready for CS 361

Class Policies

- Check Canvas daily for quizzes
- Class attendance is necessary and expected
  - We will be “learning by doing” most of the time
  - Find a group (2-3 people) to work with consistently, or switch it up
- Slides and lectures are/will be posted
  - Slides linked from website and lectures are on YouTube
- Please exercise good virtual meeting etiquette
  - Leave webcam on, but mute yourself in the large room when not speaking

Course Policies

- The projects in this course are VERY important!
  - One purpose of this course is to ensure you are ready to tackle harder projects in CS 361 and the system electives

- Projects are **individual** and **mandatory**
  - A “good faith” submission shows evidence of significant work and investment in writing a solution
  - A “good faith” submission gets you an “F” (50 or 60 points) instead of a zero!

Course Policies

- The JMU Honor Code applies on ALL assignments
  - Violations may be sent to the honor council
  - See relevant section in the syllabus
  - All quizzes, tests, and exams must be done by yourself with no assistance aside from what is allowed in the assignment description in Canvas
- All submitted labs must represent YOUR work
  - You will work in groups to discuss the answers
  - You will be able to see other student's work
  - By submitting a PDF on Canvas, you are asserting that these answers are YOUR answers and that you understand WHY you have answered the way you have

Course Policies

- All submitted project code must be YOUR work entirely
  - You may work in groups to discuss general approaches (in fact, I encourage this; use *pseudocode* if necessary)
  - However, one goal of the projects in this course is to develop individual competency, so **you may NOT share code** with anyone who is not a TA or CS 261 instructor
  - This includes letting someone examine or take a screenshot of your code, or “talking it through” with them line-by-line
  - If you have questions about this, please ask!

Action items

- If you haven't already:
  - **Take the intro survey on Canvas**
  - **Take syllabus quiz on Canvas**
  - **Read CS:APP 1.1-1.4 and 1.8 and take quiz (due tomorrow)**
  - **Make sure you can log into `stu`**

- Over the weekend (before class on Monday):
  - **Watch Monday's lecture videos (no reading/quiz this time)**
  - Make sure you can access Piazza
  - Review these slides
  - Read project guide on website
    - For a real head start, read the Project 0 description