CS 261
Fall 2018

Mike Lam, Professor

AN x64 PROCESSOR IS SCREAMING ALONG AT BILLIONS OF CYCLES PER SECOND TO RUN THE XNU KERNEL, WHICH IS FRANTICALLY WORKING THROUGH ALL THE POSIX-SPECIFIED ABSTRACTION TO CREATE THE DARWIN SYSTEM UNDERLYING OS X, WHICH IN TURN IS STRAINING ITSELF TO RUN FIREFOX AND ITS GECKO RENDERER, WHICH CREATES A FLASH OBJECT WHICH RENDERS DOZENS OF VIDEO FRAMES EVERY SECOND
BECAUSE I WANTED TO SEE A CAT JUMP INTO A BOX AND FALL OVER.
I AM A GOD.

Computer Systems I: Introduction

Welcome to CS 261!

Please go to **`socrative.com`** on your phone or laptop, choose “student login” and join room “LAMJMU”

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

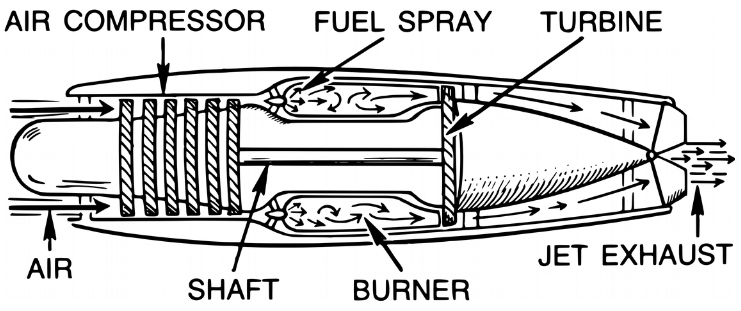
- What is a “system?”
  - Set of interacting components
  - More than the sum of its parts

Jet engine

Computer

Systems

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

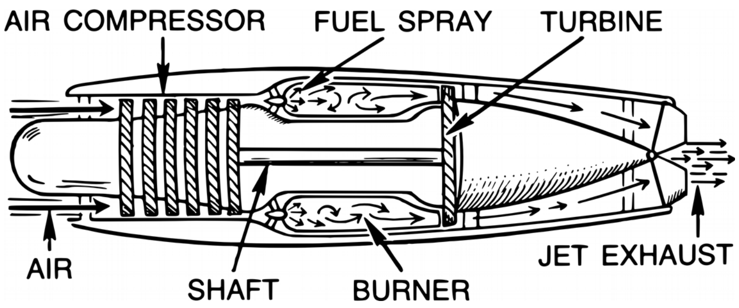


Jet engine

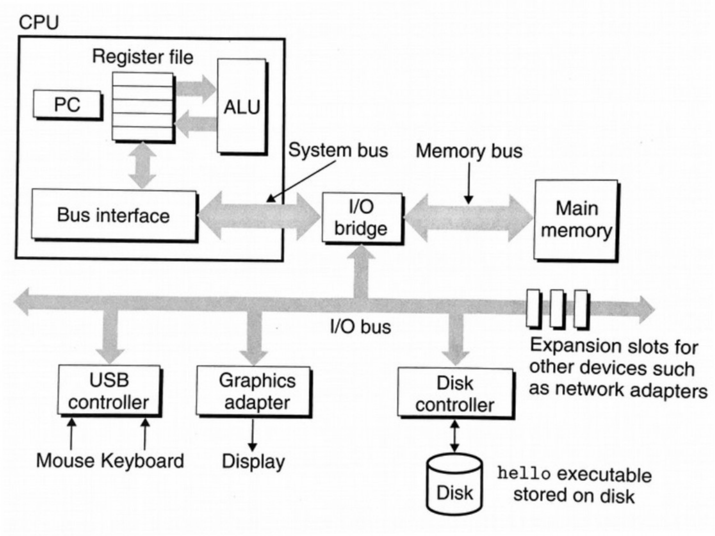
Computer

Systems

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



Jet engine



Computer

Systems

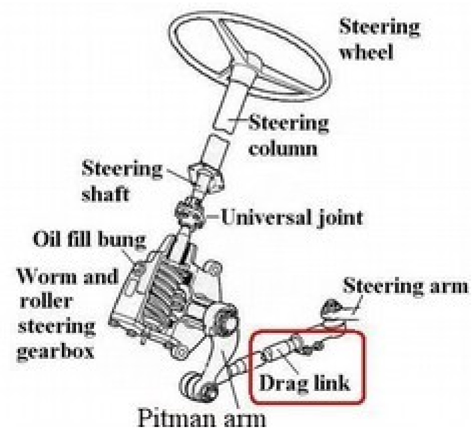
- A **computer system** consists of multiple hardware and software components that work together to run user applications.
  - We use complex computer systems every day
  - Our goal: peel back some of the complexity
    - See (some of) what’s “under the hood”

Systems

- What is a *process*? What is a *file*?

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

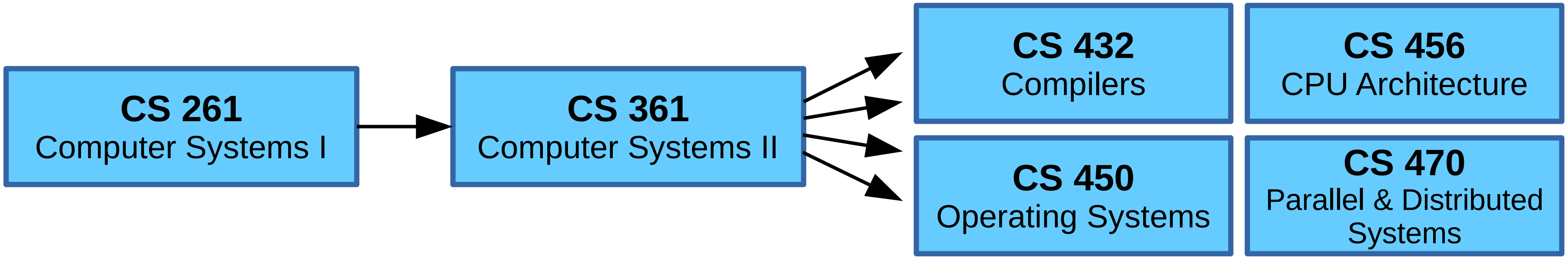
Course Objectives

- 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 weeks)
  - Computer Architecture (3 weeks)
  - Operating Systems Concepts (3 weeks)



CS 261
Computer Systems I

CS 361
Computer Systems II

CS 432
Compilers

CS 456
CPU Architecture

CS 450
Operating Systems

CS 470
Parallel & Distributed Systems

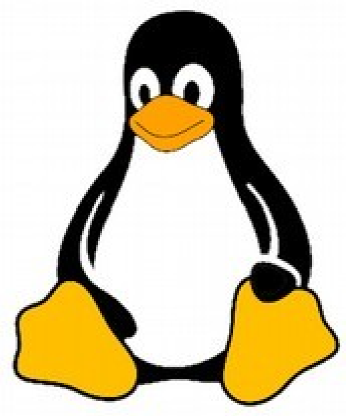
Fundamentals of digital,
single-process systems

Multi-process systems
and networking

In-depth study of a particular
kind of complex system

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
    - InstallFest on Wed, Sep 5 at 6:30 in ISAT/CS 246
    - Weekly meetings thereafter (same time and place)



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
  - Learn the **why** and not just the **what**
  - 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 <u>ask questions</u>

Course Components

- **Public website** (`w3.cs.jmu.edu/lam2mo/cs261`)
  - Syllabus, **calendar**, assignments, and resources (links)
- **Canvas course**
  - Quizzes and unit tests
  - Grades and private files (e.g., lab solutions)
  - Piazza Q&A and discussion forum
- **Student server** (`stu.cs.jmu.edu`)
  - Project development and submission

- Make sure you can access all of these!

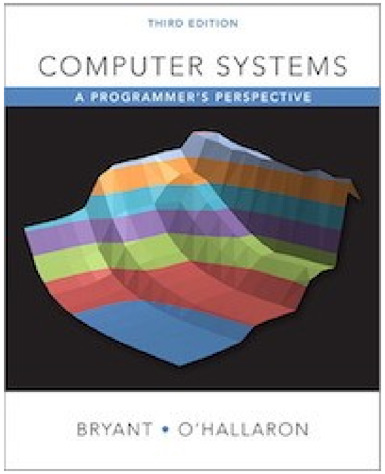
Course Grades

Quizzes and Labs	20%
Programming Projects	30%
Online Unit Tests	20%
Written Exams	30%

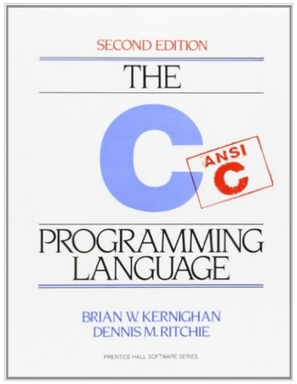
- Quizzes and labs are **formative**
  - Designed to help you learn
- Projects and tests/exams are **summative**
  - Designed to assess what you have learned

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


SECOND EDITION
THE
C
ANSI C
PROGRAMMING
LANGUAGE
BRIAN W. KERNIGHAN
DENNIS M. RITCHIE
PRENTICE HALL SOFTWARE SERIES

Class Policies

- Check Canvas daily for quizzes
- Class attendance is necessary
  - We will be “learning by doing” much of the time
  - Find a group (2-3 people) to work with consistently, or switch it up
- Slides will be posted on the website
  - No need to copy them to your notes
- Please silence your cell phones during class
  - Be respectful with laptop and tablet usage

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 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, the primary goal of the projects in this course is to develop individual competency, so **you may NOT share code**
  - This includes letting someone examine or take a photo of your code, or "talking it through" with them line-by-line
  - If you have questions about this, please ask!

Question

- Which of the following are honor code violations in this course when done in the presence of non-instructors? (Select all that apply.)
  - A) Writing English psuedocode of project solutions on a whiteboard
  - B) Storing project solutions in a public Github repo
  - C) Screen-sharing with project code visible on Skype
  - D) Writing C code of project solutions on a whiteboard
  - E) Discussing code design choices (e.g., “did you write a helper method for this part?”)
  - F) Storing project solutions in a private Github repo
  - G) Taking a photo of project code on a computer screen

Course Policies

- There are a total of three sections of CS 261
    - Two Lam sections and one Weikle section (all T-Th)
    - Projects, unit tests, and exams are common
    - **Quizzes and labs may differ**
    - You are welcome to study with students from other sections, but you must attend and submit assignments to the section you are registered for

Intro lab

- Material from Chapter 1
- Front page: **Computer Organization**
- Back page: **C Compilation**
- Work in groups of 2-3 (no computer required)
- Submit at end of class

Office hours

- My office hours TBD (just drop in this week)
- General TAs
  - ISAT/CS 248 and 250
  - 5pm-11pm on Mon-Thurs and Sunday 1-11 pm
- 261-specific TA: Becky Wild
  - 7-9pm Tue and 7-11pm Thur

Have a great semester!

- Before Thursday:
  - **Take the intro and email disclosure surveys on Canvas**
  - **Read sections 1.1-1.4 and 1.8 in CS:APP and take quiz**
  - **Make sure you can log into `stu`**
  - Make sure you can access Piazza
  - Review these slides
  - Read project guide on website
    - For a real head start, read the Project 0 description