Welcome to CS240

Foundations of Computer Systems

- Algorithm, Data Structure, Application
- Programming Language
- Compiler/Interpreter
- Operating System
- Instruction Set Architecture
- Microarchitecture
- Digital Logic
- Devices (transistors, etc.)
- Solid-State Physics
The Plan

1) What is CS240?
2) Why take CS240?
3) How will CS240 work?
4) Jump into the foundations of computer hardware
Other areas of CS...

CS111, CS230, CS231, CS235, CS251

- What can a program do?
- How can a program solve a problem?
- How do you structure a program?
- How do you know it is correct or efficient?
- How hard is it to solve a problem?
- How is computation expressed?
- What does a program mean?
- ...

A BIG question is missing...
1) What is CS240?

How do computers work?

```java
public class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello, world!");
    }
}
```

Welcome to DrJava. Working directory is /Users/bpw/courses/cs240/cs240f14
> run HelloWorld
Hello, world!
>

Running main method of Current Document

6:0
CS 111, 230, 231, 235, 251

CS 240

Software

Programming Language

Compiler/Interpreter

Operating System

Instruction Set Architecture

Microarchitecture

Digital Logic

Devices (transistors, etc.)

Solid-State Physics

Hardware

Algorithm, Data Structure, Application
Big Idea: Abstraction

Layers of virtual machines manage complexity.

interface
implementation
Big Idea: Abstraction
with a few recurring subplots

Simple, general interfaces:
- Hide complexity of efficient implementation.
- Make higher-level systems easy to build.
- But they are not perfect.

Representation of data and programs

Translation of data and programs

Control flow within/across programs

0s and 1s, electricity
compilers, assemblers, decoders
branches, procedures, OS
Charles Babbage designs Analytical Engine

Prototype of Analytical Engine, (was never actually built), Science Museum, London

Image: public domain

Ada Lovelace writes the first computer program

George Boole describes formal logic for computers — Boolean Algebra

Countess Ava Lovelace, 1840s
George Boole, 1860s
University College Cork, Ireland

Image: public domain
Human computers

Computing machines

Alan Turing, 1940s
Imitation Game, 2014
Image: Flikr mark_am_kramer, Imitation Game poster

NASA computers, 1953
Hidden Figures, 2016
Image: NASA/JPL/Caltech, Hidden Figures
ENIAC (Electronic Numerical Integrator and Computer),
First Turing-complete all-electronic programmable digital computer.
University of Pennsylvania, 1940s

Image: public domain
Jean Jennings Bartik and Frances Bilas Spence with part of ENIAC.

*The programmers of ENIAC were six women.*

http://eniacprogrammers.org/, http://sites.temple.edu/topsecretrosies/

Image: public domain
Programming 1940s-style with switches and cables.
1930s
1940s
1950s
1960s
1970s
1980s
1990s
2000s
2010s

programs are data

machine translates instructions to control flow

Manchester “Baby” SSEM (Small-Scale Experimental Machine), replica first stored-program computer -- University of Manchester (UK), 1948

Image: “SSEM Manchester museum close up” by Parrot of Doom - Own work. Licensed under Creative Commons Attribution-ShareAlike 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:SSEM_Manchester_museum_close_up.jpg
PDP-11
"minicomputers"

http://www.pcworld.com/article/249951/if_it_aint_broke_dont_fix_it_ancient_computers_in_use_today.html?page=2

http://simh.trailing-edge.com/
<table>
<thead>
<tr>
<th>ENIAC</th>
<th>iPhone 7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Year</strong></td>
<td>1946</td>
</tr>
<tr>
<td><strong>Weight</strong></td>
<td>30 tons</td>
</tr>
<tr>
<td><strong>Volume</strong></td>
<td>2,400 ft³</td>
</tr>
<tr>
<td><strong>Cost</strong> (USD, 2014)</td>
<td>$6,000,000</td>
</tr>
<tr>
<td><strong>Speed</strong></td>
<td>few 1000 ops/sec</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>~100 bytes</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>150,000 W</td>
</tr>
<tr>
<td><strong>Input/Output</strong></td>
<td>Switches, lights, later punchcards</td>
</tr>
<tr>
<td><strong>Production</strong></td>
<td>1</td>
</tr>
</tbody>
</table>
Modern Computer Organization

**Processor**
- Executes instructions.

**Memory**
- Stores program code + data during execution.

**Bus**
- Connects processor and memory.

**Input/Output**
- Persistent Storage
- Network
- USB
- Display
- ...
Modern Computer Organization

Processor

Executes instructions.

Memory

Stores program code + data during execution.

Processor repeats:
1. fetch instruction
2. fetch data used by instruction
3. execute instruction on data
4. store result or choose next instruction
Desired computation represented as instructions.

Physical implementation of instructions and resources.

Abstraction!
Instruction Set Architecture (HW/SW Interface)

**Instructions**
- Names, Encodings
- Effects
- Arguments, Results

**Local storage**
- Names, Size
- How many

**Large storage**
- Addresses, Locations

**Computer**

**Processor**
- Instruction Logic
- Registers

**Memory**
- Encoded Instructions
- Data
Machine Instructions

(adds two values and stores the result)

00000010100010101100100000010000

Instruction Set Architecture specification
Assemblers and Assembly Languages

```
addl %eax, %ecx
```

00000010100010101100100000010000

Assembly Language specification
Higher-Level Programming Languages

\[ x = x + y; \]

addl %eax, %ecx  

Programming Language specification

Compile time

Run time

Machine code program

Hardware
A-o: first compiler, by Grace Hopper

Early 1950s

Maybe closer to assembler/linker/loader

Later:

B-0 → FLOW-MATIC
→ COBOL, late 50s

Jean Sammet also involved

headed first sci comp
group at Sperry in the '50s

Later first female president of ACM.
More and more layers...

- Operating systems
- Virtual machines
- Hypervisors
- Web browsers
- ...
CS 240: a 3-stage sprint
(4-5 weeks each)

Hardware implementation
From transistors to a simple computer

Hardware-software interface
From instruction set architecture to C

Abstraction for practical systems
Memory hierarchy
Operating systems
Higher-level languages

Sometimes system abstractions "leak."
Implementation details affect your programs.
int ≠ integer
float ≠ real

int x = ...;

x*x >= 0 ?
   40000 * 40000 == 1600000000
   50000 * 50000 == -1794967296

float a = ..., b = ..., c = ...;

(a + b) + c == a + (b + c) ?
   (-2.7e23 + 2.7e23) + 1.0 == 1.0
   -2.7e23 + (2.7e23 + 1.0) == 0.0
Exploded due to cast of 64-bit floating-point number to 16-bit signed number. Overflow.

Boeing 787, 2015

"... a Model 787 airplane ... can lose all alternating current (AC) electrical power ... caused by a software counter internal to the GCUs that will overflow after 248 days of continuous power. We are issuing this AD to prevent loss of all AC electrical power, which could result in loss of control of the airplane." --FAA, April 2015
Arithmetic Performance

\[ x / 973 \]

Memory Performance

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (j = 0; j < 2048; j++)
        for (i = 0; i < 2048; i++)
            dst[i][j] = src[i][j];
}
```

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (i = 0; i < 2048; i++)
        for (j = 0; j < 2048; j++)
            dst[i][j] = src[i][j];
}
```

several times faster
due to hardware caches
The **GHOST vulnerability** is a buffer overflow condition that can be easily exploited locally or remotely, which makes it extremely dangerous. This vulnerability is named after the `GetHOST` function involved in the exploit.

**A Heart Device Is Found Vulnerable to Hacker Attacks**

By BARNABY J. FEDER

Published: March 12, 2008

To the long list of objects vulnerable to attack by computer hackers, add the human heart.

The threat seems largely theoretical. But a team of computer security researchers plans to report Wednesday that it had been able to gain wireless access to a combination heart defibrillator and pacemaker.
2) Why take CS240?

- Learn how computers execute programs.
- **Build software tools** and appreciate the value of those you use.
- Deepen your appreciation of **abstraction**.
- Learn enduring **system design principles**.
- Improve your **critical thinking** skills.
- Become a **better programmer**:
  - Think rigorously about execution models.
  - Program carefully, defensively.
  - Debug and reason about programs effectively.
  - Identify limits and impacts of abstractions and representations.
  - Learn to use software development tools.

- **Foundations** for:
  - Compilers, security, computer architecture, operating systems, ...

- Have fun and feel accomplished!
Also: C programming language

- Invented to build UNIX operating system, 1970s
  - OS manages hardware, C close to machine model

- Simple pieces look like Java:
  - if, while, for, local variables, assignment, etc.

- Other pieces do not:
  - structs vs. objects, functions vs. methods
  - addresses, pointers
  - no array bounds checks
  - weak type system

- Important language, still widely used, but many better PL ideas have come along since.
3) How will CS240 work?

https://cs.wellesley.edu/~cs240/