Welcome to

CS 240: Foundations of Computer Systems
Today

1. What is CS 240?
2. Why take CS 240?
3. How does CS 240 work?
4. Dive into foundations of computer hardware.
CS 111, 230, 231, 235, 251:

• 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...
CS 240: How do computers work?
Devices (transistors, etc.)
Solid-State Physics

Hardware

Software

CS 111, 230, 231, 235, 251

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

CS 240
Big Idea: Abstraction

Layers of virtual machines manage complexity.
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
<table>
<thead>
<tr>
<th>Decade</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1800s</td>
<td>Ada Lovelace writes the first computer program</td>
</tr>
<tr>
<td>1810s</td>
<td></td>
</tr>
<tr>
<td>1820s</td>
<td></td>
</tr>
<tr>
<td>1830s</td>
<td>Charles Babbage designs Analytical Engine</td>
</tr>
<tr>
<td>1840s</td>
<td>George Boole describes formal logic for computers (Boolean Algebra)</td>
</tr>
<tr>
<td>1850s</td>
<td></td>
</tr>
<tr>
<td>1860s</td>
<td>Countess Ava Lovelace, 1840s</td>
</tr>
<tr>
<td>1870s</td>
<td>George Boole, 1860s</td>
</tr>
<tr>
<td>1880s</td>
<td></td>
</tr>
</tbody>
</table>

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

*Image: public domain*
1890s 1900s 1910s 1920s 1930s 1940s 1950s 1960s 1970s

Human computers

Computing machines

Alan Turing, 1940s
Imitation Game, 2014

NASA computers, 1953
Hidden Figures, 2016

Image: Flikr mark_am_kramer, Imitation Game poster
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
Physical control flow

Programming 1940s-style *with switches and cables.*

Image: public domain
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-Share Alike 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:SSEM_Manchester_museum_close_up.jpg
PDP-11 "minicomputers"
<table>
<thead>
<tr>
<th></th>
<th>1940s</th>
<th>1950s</th>
<th>1960s</th>
<th>1970s</th>
<th>1980s</th>
<th>1990s</th>
<th>2000s</th>
<th>2010s</th>
<th>2020s</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ENIAC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>1946</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weight</td>
<td>30 tons</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Volume</td>
<td>2,400 ft³</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost (USD, 2014)</td>
<td>$6,000,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>few 1000 ops/sec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>~100 bytes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power</td>
<td>150,000 W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input/Output</td>
<td>Switches, lights, later punchcards</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Production</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>iPhone 5</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>2012</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weight</td>
<td>4 oz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Volume</td>
<td>3.4 in³</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost (USD, 2014)</td>
<td>$600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td>2,500,000,000 ops/sec</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory</td>
<td>1,073,741,824 bytes (1 GB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power</td>
<td>&lt;5W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input/Output</td>
<td>Touchscreen, audio, camera, wifi, cell, ...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Production</td>
<td>5,000,000 sold in first 3 days</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Modern Computer Organization

Stores program code + data during execution.

Processor

Executes instructions.

Memory

Bus

Input/Output

Persistent Storage

Network

USB

Display

...
Modern Computer Organization

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

Processor

Executes instructions.

Memory

Stores program code + data during execution.
Physical implementation of instructions and resources.

Desired computation represented as instructions.

Hardware/Software Interface

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)

000000010100010101011001000000010000

Instruction Set Architecture specification
Assemblers and Assembly Languages

1940s | 1950s | 1960s | 1970s | 1980s | 1990s | 2000s | 2010s | 2020s

Addl %eax, %ecx 00000010100010101100100000010000

Assembly Language specification

Assembly program → Assembler → Machine code program → Hardware
Higher-Level Programming Languages

\[ x = x + y; \]

Machine code program

```
addl %eax, %ecx 00000010100010101100100000010000
```

Programming Language specification
A-0: 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
- Mount Holyoke alum, class of 1948
More and more layers...

- Operating systems
- Virtual machines
- Hypervisors
- Web browsers
- ...

<table>
<thead>
<tr>
<th>1940s</th>
<th>1950s</th>
<th>1960s</th>
<th>1970s</th>
<th>1980s</th>
<th>1990s</th>
<th>2000s</th>
<th>2010s</th>
<th>2020s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CS 240 in 3 acts
(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
I just like to program.

Why study the implementation?

It's fascinating, great for critical thinking.

System design principles apply to software too.

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
Reliability?

Ariane 5 Rocket, 1996
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
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 `GetHOS` function involved in the exploit.

**Cyber-Safe**

**All computers are flawed -- and the fix will take years**

by Selena Larson  @seilena larson

January 26, 2018, 12:07 PM ET

**Meltdown and Spectre**

---

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

By Barnaby J. Feder

Published: March 12, 2018

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.
Why take CS 240?

• 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!
https://cs.wellesley.edu/~cs240/

Everything is here. Please read it.