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?

```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!
>
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

- Os and 1s, electricity
- Compilers, assemblers, decoders
- Branches, procedures, OS
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.*

Image: public domain
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 of the 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"

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>1930s</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>
</tr>
</thead>
</table>

Images:
- "Ibm pc 5150" by Ruben de Rijcke - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0 via Wikimedia Commons - [http://commons.wikimedia.org/wiki/File:Ibm_pc_5150.jpg](http://commons.wikimedia.org/wiki/File:Ibm_pc_5150.jpg)
- "Macintosh-motherboard" by Shieldforyoureyes Dave Fischer - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0 via Wikimedia Commons - [http://commons.wikimedia.org/wiki/File:Macintosh-mo](http://commons.wikimedia.org/wiki/File:Macintosh-mo)
<table>
<thead>
<tr>
<th></th>
<th>ENIAC</th>
<th>iPhone 5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Year</strong></td>
<td>1946</td>
<td>2012</td>
</tr>
<tr>
<td><strong>Weight</strong></td>
<td>30 tons</td>
<td>4 oz</td>
</tr>
<tr>
<td><strong>Volume</strong></td>
<td>2,400 ft³</td>
<td>3.4 in³</td>
</tr>
<tr>
<td><strong>Cost (USD, 2014)</strong></td>
<td>$6,000,000</td>
<td>$600</td>
</tr>
<tr>
<td><strong>Speed</strong></td>
<td>few 1000 ops/sec</td>
<td>2,500,000,000 ops/sec</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>~100 bytes</td>
<td>1,073,741,824 bytes (1 GB)</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>150,000 W</td>
<td>&lt;5W</td>
</tr>
<tr>
<td><strong>Input/Output</strong></td>
<td>Switches, lights, later punchcards</td>
<td>Touchscreen, audio, camera, wifi, cell, ...</td>
</tr>
<tr>
<td><strong>Production</strong></td>
<td>1</td>
<td>5,000,000 sold in first 3 days</td>
</tr>
</tbody>
</table>
Modern Computer Organization

**Processor**
- *Executes instructions.*

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

**Bus**

**Input/Output**
- Persistent Storage
- Network
- USB
- Display
- ...
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.

Hardware/Software Interface

Abstraction!
Microarchitecture (Implementation of ISA)

- Instruction Fetch and Decode
- Registers
- ALU
- Memory
Instruction Set Architecture (HW/SW Interface)

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

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

**Large storage**
- Addresses, Locations

**Computer**

**Instruction Logic**

**Registers**

**Encoded Instructions**

**Data**
Machine Instructions

<table>
<thead>
<tr>
<th>1930s</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>
</tr>
</thead>
</table>

(adds two values and stores the result)

000000101000101100100000010000

Instruction Set Architecture specification

Hardware

machine code program
Assemblers and Assembly Languages

Assemblers transform assembly language programs into machine code programs. For example:

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

becomes the binary code:

```
00000010100010101100100000010000
```
Higher-Level Programming Languages

x = x + y;

addl %eax, %ecx

Programming Language specification

Compile time

high-level language program → Compiler → assembly program → Assembler → machine code program → Run time

Hardware

x = 00000010100010101100100000010000

23
A-0: first compiler, by Grace Hopper

Early 1950s
Maybe closer to assembler/linker/loader

Later:
B-0 $\rightarrow$ FLOW-MATIC
$\rightarrow$ 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
- ...

<table>
<thead>
<tr>
<th>1930s</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>
</tr>
</thead>
<tbody>
<tr>
<td>1930s</td>
<td>1940s</td>
<td>1950s</td>
<td>1960s</td>
<td>1970s</td>
<td>1980s</td>
<td>1990s</td>
<td>2000s</td>
<td>2010s</td>
</tr>
</tbody>
</table>
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
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
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
Arithmetic Performance

\[
x / 973 \quad x / 1024
\]

Memory Performance

```c
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];
}
```

```c
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, 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.
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!
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

Everything is here.
Please read it.

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