Welcome to
CS 240:
Foundations of
Computer Systems!

The Plan

1. Devices (transistors, etc.)
2. Solid-State Physics
3. Digital Logic
4. Microarchitecture
5. Instruction Set Architecture
6. Operating System
7. Compiler/Interpreter
8. Programming Language
9. Program, Application
10. Programming Language

Welcome to
CS 240:
Foundations of
Computer Systems!

Your lecture instructor: Alexa VanHattum
Note: you can call me “Alexa”, “Prof. Alexa”, or “Prof. VanHattum”

- New to Wellesley this year!
- Research focus: programming languages & systems
- Before Wellesley:
  - PhD in Computer Science at Cornell
  - Software engineer for Apple health (heart monitoring)
  - THIS CLASS one of the most helpful across industry and research

Today’s preview

1. What is CS 240?
2. Why take CS 240? (in brief)
3. How does CS 240 work? (in brief)
CS 111, 230, 231, 235, 251:
- How do you use programming to 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...
Big Idea: Abstraction
with a few recurring subplots

Simple, general interfaces:
- Hide complexity of efficient implementation.
- Make higher-level systems easy to build.

**Representation** of data and programs:
- Ds and 1s, electricity

**Translation** of data and programs:
- Compilers, assemblers, decoders

**Control flow** within/across programs:
- Branches, procedures, operating system

---

**CS 240 in 3 acts** (4-5 weeks each)

1. **Hardware implementation**
   - From transistors to a simple computer

2. **Hardware-software interface**
   - From instruction set architecture to programming in C

3. **Abstraction for practical systems**
   - Memory hierarchy
   - Operating system basics
   - Higher-level languages and tools

---

**Machine Instructions**

Machine code
(Program added values, stores the result)

Instruction Set Architecture specification

---

**Desired computation**
in a programming language

**Hardware/Software Interface**

Physical implementation
with circuits and electricity.
Assemblers and Assembly Languages

Assemblers and Assembly Languages

Higher-Level Programming Languages

More and more layers...

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

Computer

Microarchitecture (Implementation of ISA)
2. *I just like to program. Why study the implementation?*

Most system abstractions "leak."

Implementation details affect your programs:

- Their performance
- Their correctness
- Their security

**Performance**

- \(x / 973\)
- \(x / 1024\)

Which of these code snippets is faster? `\(x / 973\)` or `\(x / 1024\)`

- `\(x / 973\)` is faster
- `\(x / 1024\)` is faster
- The same
Of these code snippets is faster? `x / 973` or `x / 1024`

<table>
<thead>
<tr>
<th>Code snippet</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x / 973</code> is faster</td>
<td>0%</td>
</tr>
<tr>
<td><code>x / 1024</code> is faster</td>
<td>0%</td>
</tr>
<tr>
<td>The same</td>
<td>0%</td>
</tr>
</tbody>
</table>

```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

---

**Performance** 🏢

`x / 973` vs `x / 1024`

- `x / 973` is faster
- `x / 1024` is faster
- The same

---

**Correctness** ✗✔

```
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];
}
```

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
Why take CS 240?

Learn how computers execute programs.
Deepen your appreciation of abstraction.
Improve your critical thinking skills.

Become a better programmer:
Think rigorously about execution models.
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/

Long but necessary!