The Plan: Lab 1 preview

Welcome to CS 240:
Foundations of Computer Systems!

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

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

- New to Wellesley this semester!
- Research focus: programming languages & systems
- I work with undergrad research assistants!
- 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 Is, 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

---

**I just like to program. Why study the implementation?**
I just like to program. Why study the implementation?

Most system abstractions "leak."

Implementation details affect your programs:

<table>
<thead>
<tr>
<th>Their performance</th>
<th>Their correctness</th>
<th>Their security</th>
</tr>
</thead>
<tbody>
<tr>
<td>![clock]</td>
<td>![x]</td>
<td>![✓]</td>
</tr>
</tbody>
</table>

Their performance Their correctness Their security

Performance

\[
\frac{x}{973} \quad \frac{x}{1024}
\]

```
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

Correctness

\[
\text{int \neq integer} \quad \text{float \neq real}
\]

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

Security

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

Meltdown and Spectre
**Why take CS 240?**

Learn how computers execute programs.
Deepen your appreciation of abstraction.
Learn enduring system design principles.
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!

---

**CS 240**
Foundations of Computer Systems

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

3) Long but necessary!

---

**The Plan**

Welcome to **CS 240:** Foundations of Computer Systems!

Program, Application
Programming Language
Compiler/Interpreter
Operating System
Instruction Set Architecture
Microarchitecture
Digital Logic
Devices (transistors, etc.)
Solid-State Physics
Today

1. What is CS 240?
2. How does CS 240 work?
3. Foundations of computer hardware

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

Layers manage complexity.

Big Idea: Abstraction

Desired computation in a programming language

Physical implementation with circuits and electricity.

0s and 1s, electricity

compilers, assemblers, decoders

branches, procedures, operating system

0s and 1s, electricity

compilers, assemblers, decoders

branches, procedures, operating system

0s and 1s, electricity

compilers, assemblers, decoders

branches, procedures, operating system
Modern Computer Organization

Stores program code + data during execution.

Processor

Executes instructions.

Memory

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

Stores program code + data during execution.

Processor

Executes instructions.

Memory

Bus

Input/Output

Persistent Storage

Network

USB

Display

Microarchitecture (Implementation of ISA)

Instruction Set Architecture (HW/SW Interface)

Computer

Encoded Instructions

Instruction Logic

Registers

Data

Local storage
- Names, Size
- How many

Large storage
- Addresses, Locations

Computer
Machine Instructions

(adds two values and stores the result)

00000101000101100100000010000

Instruction Set Architecture specification

Assemblers and Assembly Languages

\texttt{addl \%eax, \%ecx} \rightarrow 00000101000101101000000010000

Assembly Language specification

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
• Mount Holyoke alum, class of 1948
More and more layers...

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

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

3 Long but necessary!