CS 240 in context
How Computers Work

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, Algorithm

Hardware

Software
Big Ideas in CS, Systems, and beyond

Abstraction
Do not start every project with transistors. Abstraction is beautiful and empowering, but real abstractions have leaks and wrinkles.

Translation
Between layers of abstraction. Structured computation.

Representation
No representation without taxation. Representations have costs.

Performance
Memory: clever, imperfect abstraction. Tiny code changes, huge impact.

Security + Reliability
Trickiest exploits & errors involve multiple layers, even hardware!

These things matter more every day.
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.
Ariane 5 Rocket, 1996
Exploded due to cast of 64-bit floating-point number to 16-bit signed number. Overflow.

1998
Mars Climate Orbiter
Disintegrated due to mismatched units in Lockheed-Martin / NASA software components.
Toyota "Unintended Acceleration Events"

Oklahoma jury:  
"Spaghetti Code" = "reckless disregard"

>10,000 global variables  
81,514 violations of MISRA-C coding rules  
Expect 3 minor bugs + 1 major bug per 30 violations

Task/process monitoring failed to monitor tasks/processes  
Memory corruption

(Wait, it was written in C?!?!?!)
"... a Model 787 airplane that has been powered continuously for 248 days can lose all alternating current (AC) electrical power due to the generator control units (GCUs) simultaneously going into failsafe mode ... This condition is 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

https://xkcd.com/571/
How could we improve computer systems?

Security

Efficiency

Speed
Space
Programmer
Cost, availability

What a simple phone can do for people: https://opendatakit.org/about/deployments/

Energy, materials

A few of the impacts we usually don't see: http://www.nytimes.com/2015/06/07/magazine/making-and-unmaking-the-digital-world.html?_r=0

Reliability

...
one thing to make the a better place...
Few of you will build new HW, OS, compiler, but…

1. Effective programmers understand their tools and systems.
2. The skills and ideas you learn here apply everywhere.

Reason about computational models, translation.

Debug for correctness and performance (with tools to help).

Assess costs and limits of representations.

"Figure it out" via documentation, experiments, *critical thinking*. 
4 Foundations

CS 301: Compilers and Runtime Systems
CS 342: Computer Security
CS 242: Computer Networks
CS 251: Programming Languages
CS 240: Computer Systems/Organization
Research
CS 349: Distributed Computing
CS 3??: Operating Systems
CS 249: Scientific and Parallel Computing
CS 3??: Computer Architecture
CS 304: Databases with Web Interfaces