CS 240 in context

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.

How Computers Work

1

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

How to Detect Exploits of the GHOST Buffer Overflow

Vulnerability

DETECTING GHOST VULNERABILITY

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 GHOST 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?!?!?)*

http://www.safetyresearch.net/blog/articles/toyota-unintended-acceleration-and-big-bowl-%E2%80%9Cspaghetti%E2%80%9D-code

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/](https://opendatakit.org/about/deployments/)

Energy, materials

Reliability

...
3 Skills for Thinking and Programming

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
- Research
- CS 349: Distributed Computing
- CS 39?: Operating Systems
- CS 249: Scientific and Parallel Computing
- CS 39?: Computer Architecture
- CS 304: Databases with Web Interfaces