CS 240 in context
How Computers Work

1

Solid-State Physics

Devices (transistors, etc.)

Digital Logic

Microarchitecture

Instruction Set Architecture

Operating System

Compiler/Interpreter

Programming Language

Program, Application, Algorithm

Software

Hardware

How Computers Work

1

Solid-State Physics

Devices (transistors, etc.)

Digital Logic

Microarchitecture

Instruction Set Architecture

Operating System

Compiler/Interpreter

Programming Language

Program, Application, Algorithm

Software

Hardware
3 Skills for Thinking and Programming

Few of you will build new HW, OS, compiler, but...

1. Effective programmers and computer scientists 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*.

Remember low-level implications of high-level choices.
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.