Welcome/what is architecture?

If the room is full, you can watch the recording!
Waitlist is being handled through HW0 (more info during lecture)

Staff introductions!

? ? ?

How does a computer actually run a program?
Be as detailed as possible!

Instructions get loaded into memory
Instruction: a representation that gets interpreted as an operation in machinery
Memory: a region that can store data, it can persist or be volatile
Memory looks like a series of charges and not charges
CPU has actual hardware that does operations
Each operation has several inputs that the CPU can react to them

    For example: a bit in an instruction can decide whether something is an add, multiply, etc

    Bits are encoded in zeroes and ones (in hardware: electrical charges)

CPU has hardware that can react to the electrical charges

Questions we’ll answer

How is a program represented on a computer?

How does a CPU actually translate stored bits in memory into tasks?

How do we load and store data efficiently?

Can we speed it up by parallelizing instructions? Data?

What is the interplay between architecture and OS? Compilers? Security?

What is the history of the field? How do/did market forces shape engineering decisions?

What’s fresh and what’s next?

? ? ?

How do we measure whether one computer is “better” than another? Get creative – go beyond “speed”

Speed – how quickly can one computer execute a given program?
Can I run concurrent processes?
How much power does it use?
How large its online community is?
Size of its memory?
Instructions or dataflow per second?
Security?
Resilience?
Production costs?
Does it have accelerators/specialized hardware?
Manufacturing guarantees?
Backdoors?
Learning curve, adoptability
Footprint?
Correctness of software?
Environmental impact
Longevity?
Customizability?

Fine, but what is architecture?

Like many terms in CS, "architecture" is overloaded

"Computer architecture" describes the structure/organization of a computer; specifically, how HW and SW interact

Three general parts:

- ISA (Instruction Set Architecture): what instructions can the computer execute and how are they defined?
- Microarchitecture: how does the CPU actually implement the ISA?
- Hardware system: what physically makes up the computer?

Computer architecture gets interesting when we consider the *interplay* between these parts!

How will we study these things?

Try to “invent” them ourselves as much as possible

- What we want a computer to do → Definition of ISA→ Microarchitecture implementation

Simulate!

- Ripes for low-level design, gem5 for more complicated design
- Allows us to evaluate design choices
- *Caveat*: need to understand our simulators and their drawbacks

Explore emerging applications

? ? ?

Why should software people care about how a computer works?

It’s cool to look under the hood!
Can write faster, more efficient programs
Can gain a deeper understanding of low-level code
Diagnose performance issues
Prepares you to work on limited hardware
Learn how to use dedicated hardware

Course structure and policies

Course website: browncs1952y.github.io

Waitlist

Handled through HW0 (also submit an override request on cab!!)

Randomized by day

Need to be caught up with HW1 to be admitted

Ways you can give me feedback

E-mail
In person (after class, in office hours)
Anonymous form
Via TAs (anonymous or not)
DE&I, accessibility, culture issues: department and university-wide resources
→ **Feedback only works if I follow up on it**

Course culture discussion