Great Ideas in Computer Architecture

CS 61C

View from the Top

Instructor: Jenny Song

Agenda

- **Powers-of-Ten view of CS61C**
- How much have you learned?
- What's next?
- The End

Powers of Ten-inspired View of 61C

Powers of 10 Film (youtube)

10^4 meters

The Dalles, Oregon

11,100 meters (across city)

Google's Oregon WSC

10^3 meters

1,100 meters (around facility)

Containers in WSCs

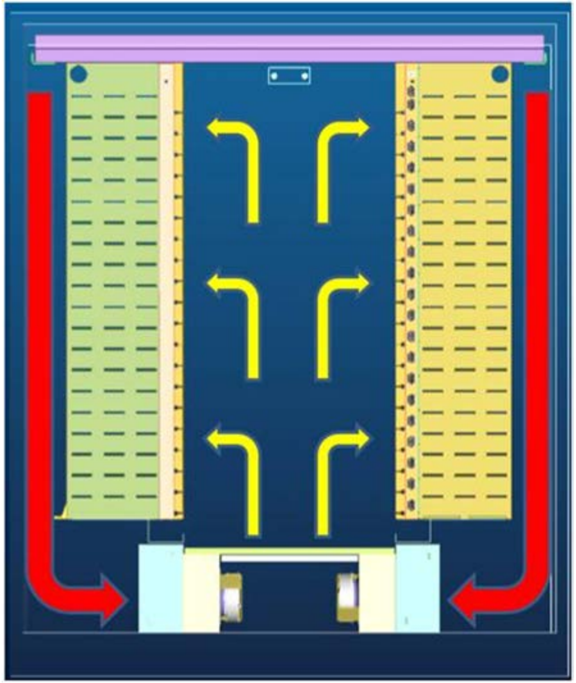
10^2 meters

100 meters (all containers)

10^1 meters

Google Server Array

10 meters (container)



- 2 long rows, each with 29 racks
- Cooling below raised floor
- Hot air returned behind racks

10^0 meters

Google Rack

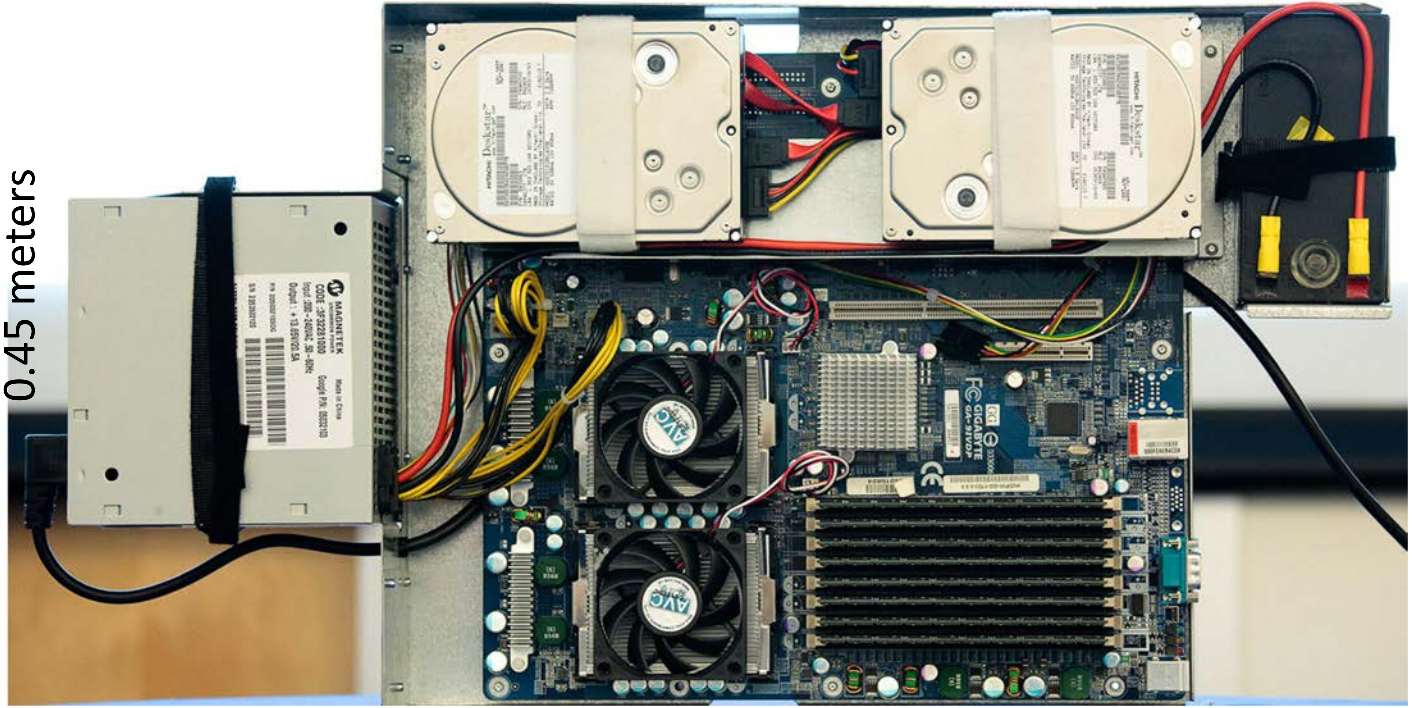
3 meters (tall)

- Google rack with 20 servers + Network Switch in the middle
- 48-port 1 Gigabit/sec Ethernet switch every other rack
- Array switches connect to racks via multiple 1 Gbit/s links
- 2 datacenter routers connect to array switches over 10 Gbit/s links

Google Server Internals

10^{-1} meters

0.45 meters



Central Processing Unit (CPU)

10^{-2} meters

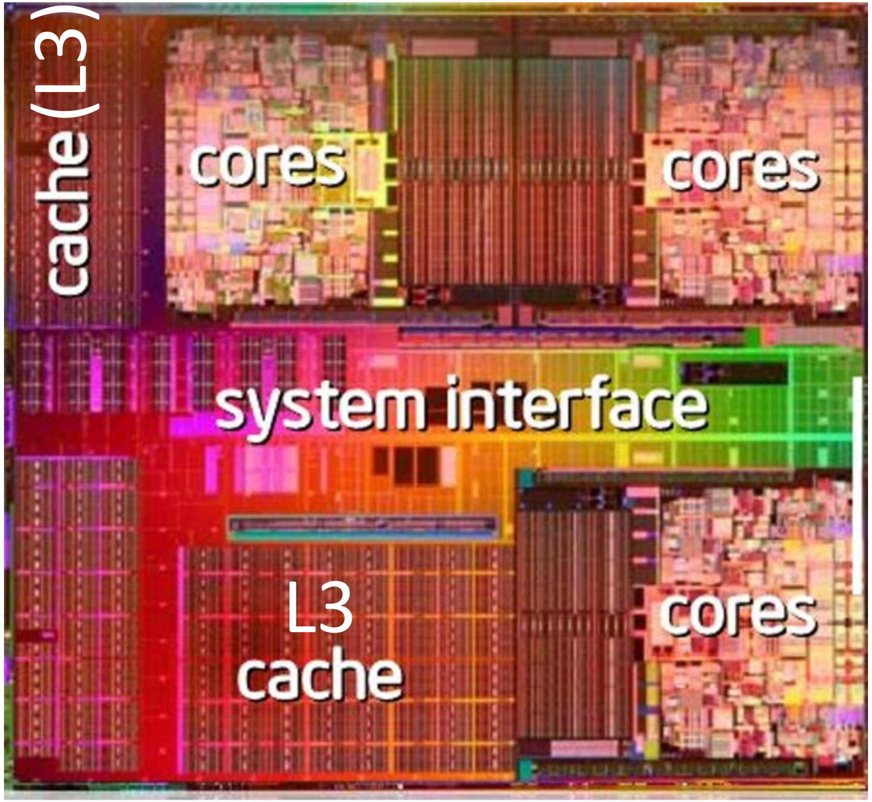
3 centimeters

2020

Processor

10^{-2} meters

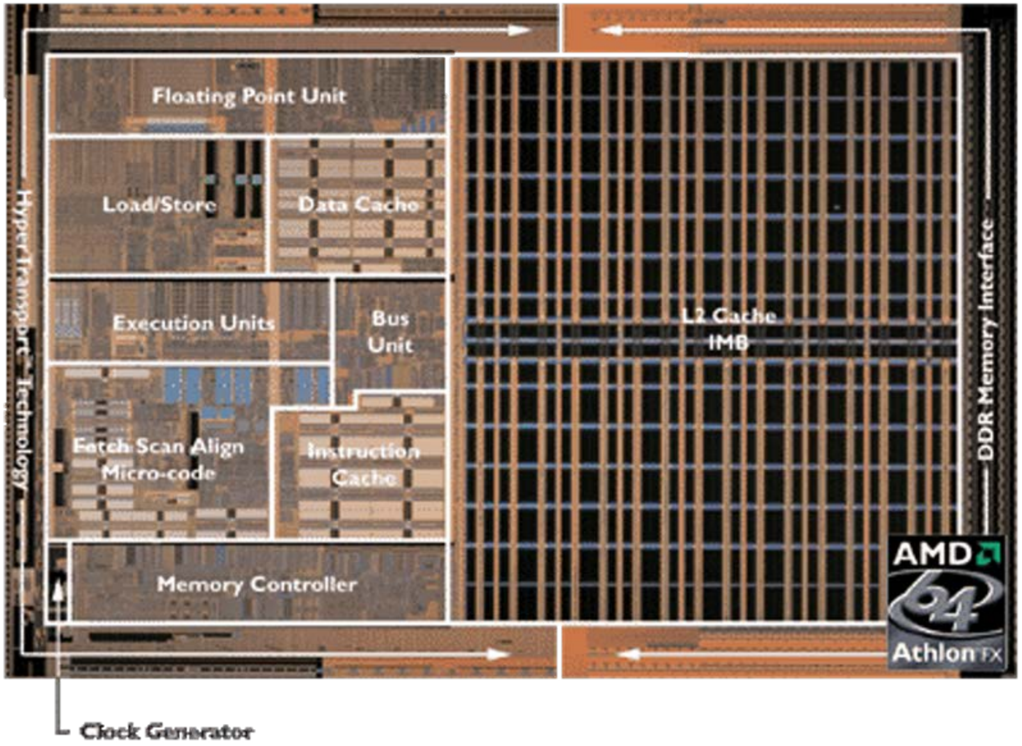
1 centimeter



10^{-3}
meters

CPU Core

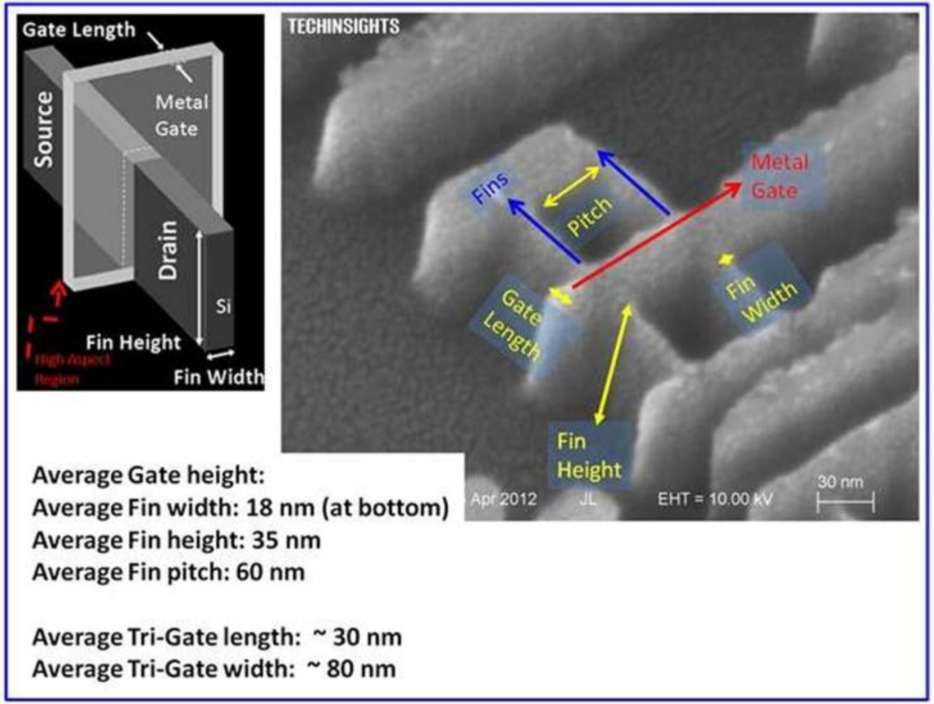
3 millimeters



10^{-6} - 10^{-9}
meters

Transistors

100-10 nanometers



Agenda

- Powers-of-Ten view of CS61C
- **How much have you learned?**
- What’s next?
- The End (with Q&A)

Six Great Ideas in Computer Architecture

1) Abstraction
2) Moore’s Law
3) Principle of Locality/Memory Hierarchy
4) Parallelism
5) Performance Measurement & Improvement
6) Dependability via Redundancy

Great Idea #1:Layers of representation/interpretation

Higher-Level Language Program (e.g. C)

↓ *Compiler*

Software

Assembly Language Program (e.g. RISC-V)

↓ *Assembler*

Machine Language Program (RISC-V)

Hardware

↓ *Machine Interpretation*

Hardware Architecture Description (e.g. block diagrams)

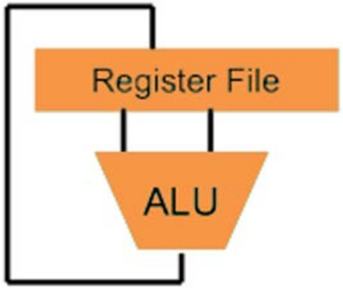
↓ *Architecture Implementation*

Logic Circuit Description (Circuit Schematic Diagrams)

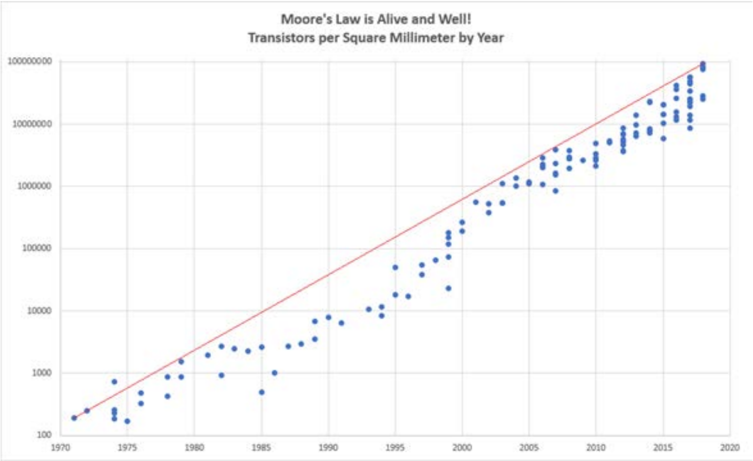
```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw   t0, 0(S2)
lw   t1, 4(S2)
sw   t1, 0(S2)
sw   t0, 4(S2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

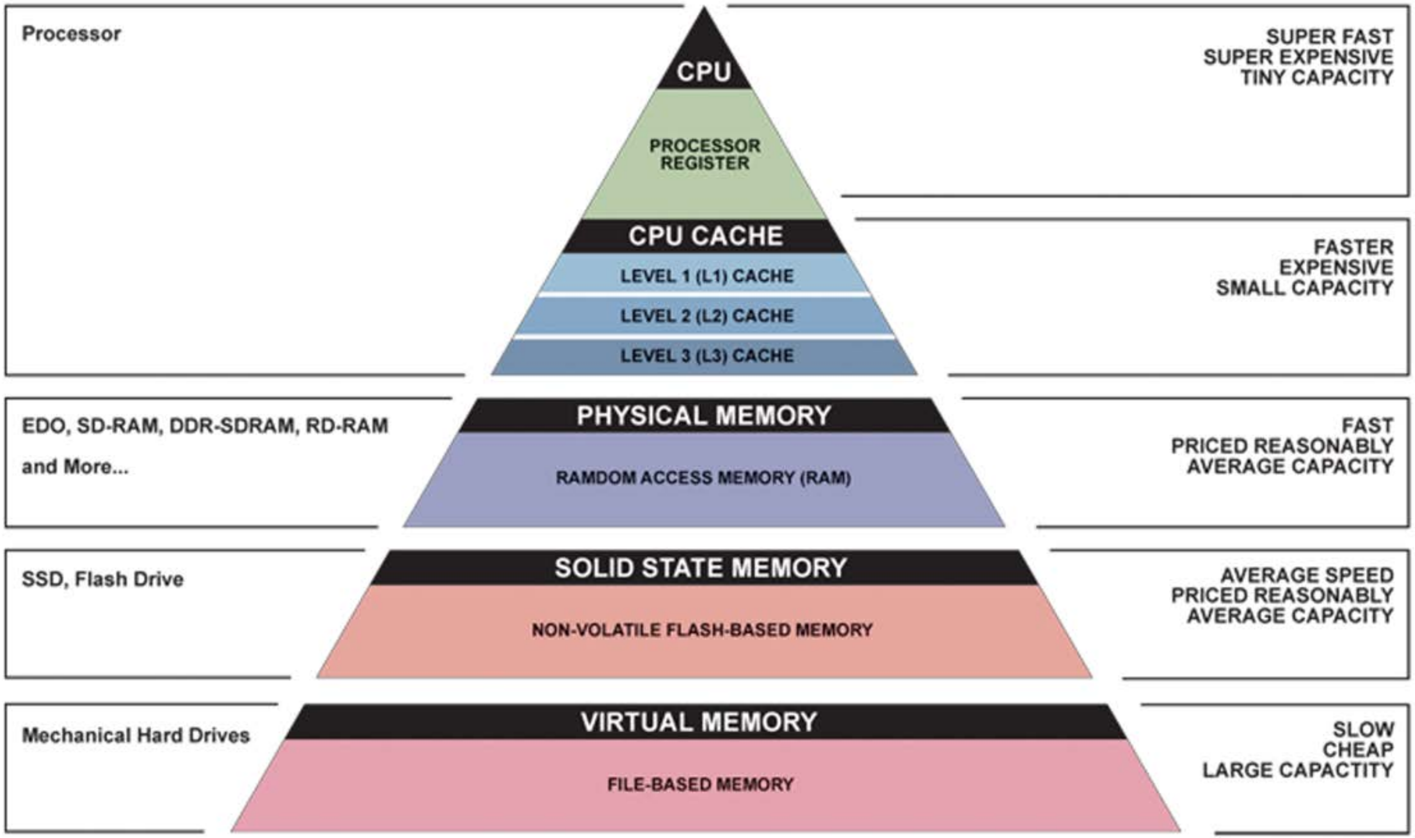


Great Idea #2: Moore's Law



"Every two years, the number of transistors on a chip of a fixed size doubles"

Great Idea #3: Principle of Locality/ Memory Hierarchy

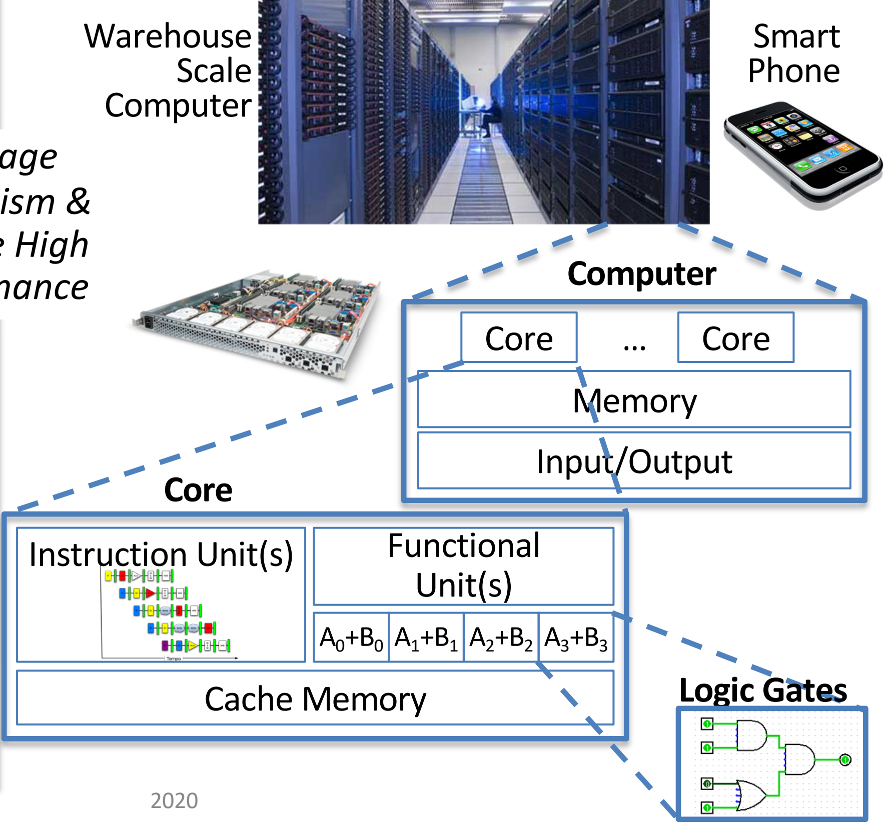


Great Idea #4: Parallelism

Software | *Hardware*

- Parallel Requests
  Assigned to computer
  e.g. search “cs 61c”
- Parallel Threads
  Assigned to core
  e.g. lookup, ads
- Parallel Instructions
  > 1 instruction @ one time
  e.g. 5 pipelined instructions
- Parallel Data
  > 1 data item @ one time
  e.g. add a pair of 6 words
- Hardware descriptions
  All gates functioning in
  parallel at same time

Leverage Parallelism & Achieve High Performance



2020

Great Idea #5:Performance

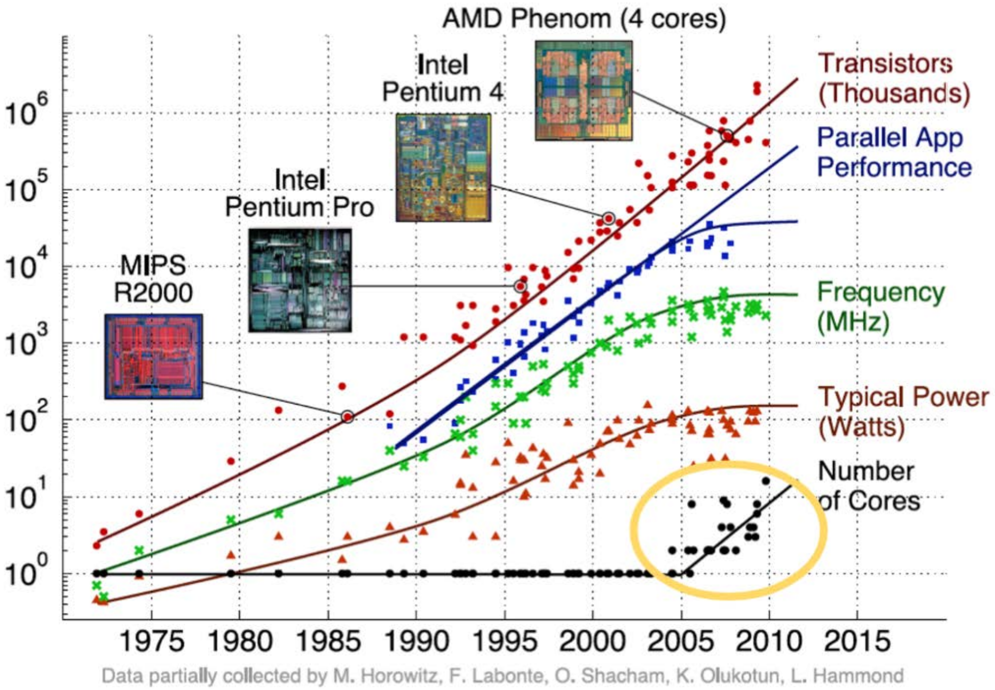
$$\frac{\text{Time}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} * \frac{\text{Cycles}}{\text{Instruction}} * \frac{\text{Time}}{\text{Cycle}}$$

Latency   Throughput

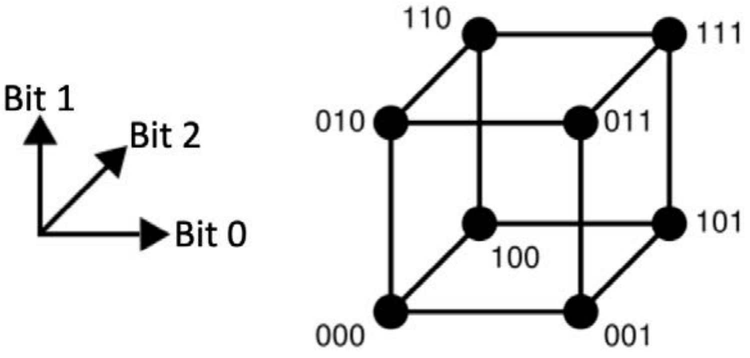
AMAT = Hit time + Miss rate × Miss penalty

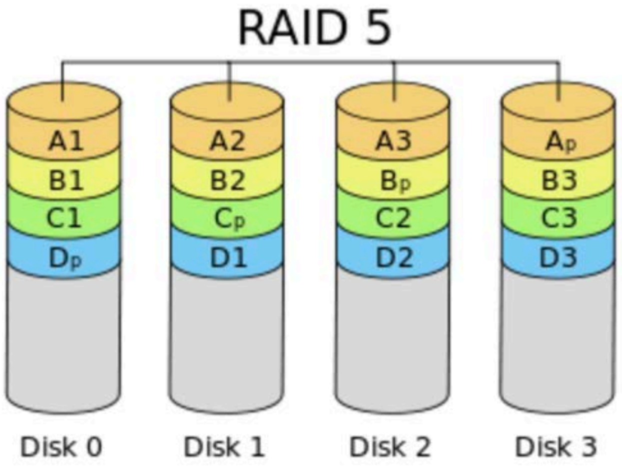
—*TLB Miss Rate:* Fraction of TLB accesses that result in a TLB Miss

—*Page Table Miss Rate:* Fraction of PT accesses that result in a page fault



Great Idea #6: Dependability via Redundancy



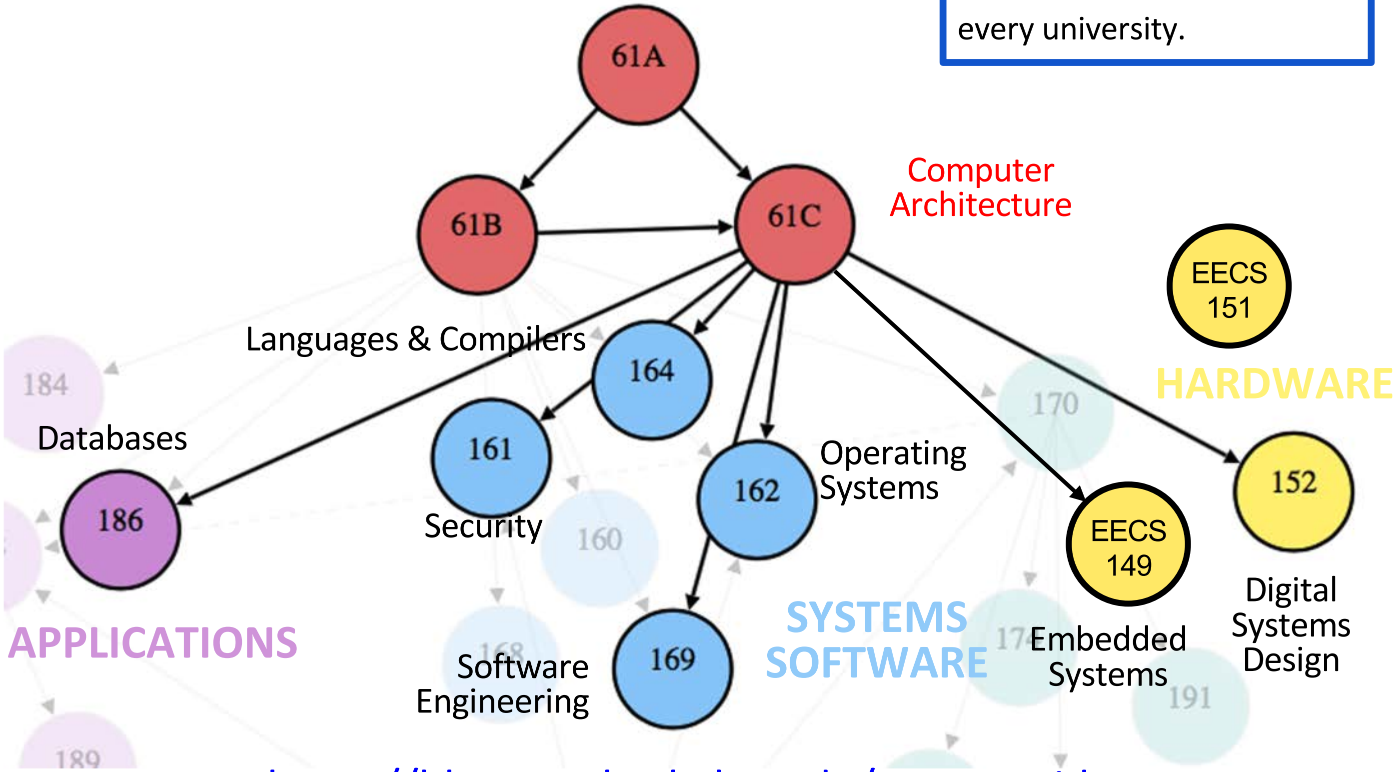


Agenda

- Powers-of-Ten view of CS61C
- How much have you learned?
- **What's next?**
- The End

What's Next? Classes

Non-Berkeley Students: Similar classes are offered at every university.



https://hkn.eecs.berkeley.edu/courseguides

CS161: Security

Bad people, bad programs are out there! How do we make systems secure? How do we write secure code?

- Lots of neat research in this area, esp at Cal
- Cool, gratifying projects
- Many, many tie-ins to other courses (CS168, CS162, etc.)
  - Security is everywhere!
- Web, network (CS168), software, system, etc.

CS162: Operating Systems

- Focused on the OS level of a computer
  - What tasks should be privileged/protected?
  - How do you enforce protection?
  - How do you write code that runs other code?

Big project class!

- choose partners wisely ;)
- very gratifying/fulfilling, but also lots of details!

CS164: Compilers & Programming Languages

Interested in optimisation, CALL, performance, or project 2?

- How are languages invented and used?
- What does it mean for languages to be compiled?
- How are language features developed?
- What classifies a particular statement as valid or invalid for a particular language?
- Build your own python-esque compiler! RISC-V :)

CS 168:Networking

- How do we turn the network, I/O into something usable?
  - We have a unrelialble “best effort” system
  - Le’s make something useful
  - And build don top fp

CS186: Databases

How do data centers and databases work? What happens on disk?

- Cool real-world immediate application: work at Amazon!
- Lots of industry connection in this course
- Coding in Java (good step from 61B, historically easier upper-div)
- Often webcast-only
- Good for webdev/backend stuff too!

CS152: Computer Architecture & Design

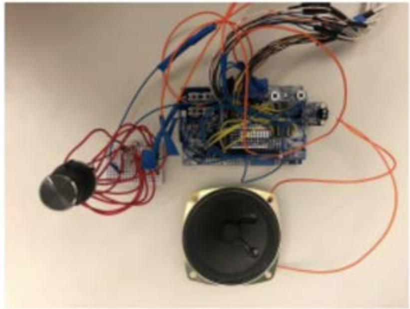
- Focused on the “system” of a computer:
  - Different kinds of datapaths
  - Different kinds of VM, caching
  - Some performance-based content
- Upper-div CS61C with ~really cool~ labs!
  - Build a branch predictor!
  - Learn how to write vectorised assembly!
  - Simulate different design decisions and benchmarks

EECS149: Introduction to Embedded Systems

How do we make computers interact with the real world?

- Cyber-Physical Systems (CPS) and the Internet of Things (IoT)
- Microcontrollers
- Low-level hardware and software
- Wireless communications

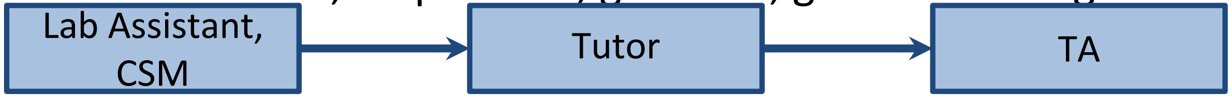
Project class: 2 months, 3-4 student teams



Becoming a Part of CS61C

- Four levels of the CS61C staff:
  - Lab assistant: help students in labs (course credit)
  - Tutor: teach guerrilla sessions and assist students ($$)
  - TA: teach labs, sections, and help run the course ($$$)
    - Head TA: coordinate hiring, “big picture” course stuff
  - Lecturer…?!?!?!? o:

- Lab Assisting:
  - Learn more, help others, get units, get on teaching track



Take advantage of educational opportunities

- Why are we one of the top university in the world?
  - Research, research, research !
  - Whether you want to go to grad school or industry, you need someone to vouch for you!
- Techniques
  - Find out what you like, do a lot of web research(read published papers), email professor and grad students, hit lab meetings
  - http://research.Berkeley.edu/
  - http://researchmatch.Heroku.com/

Making as much of college as is humanly and healthily reasonable

- Seek out experiences that lead to new experiences
  Build skills, interests, relationships
  - Meet new people, join interesting clubs, go on adventures
- Don't go it alone – find a friend group for classes
- Take advantage of educational opportunities
  - Research: research.berkeley.edu, beehive.berkeley.edu, Architecture Research
  - Student groups: UCBUGG, Mobile Developers of Berkeley, GamesCrafters, Hackers@Berkeley, CS Mentors, etc.
  - Classes: Non-major courses, DeCal
- Take care of yourself!

Agenda

- Powers-of-Ten view of CS61C
- How much have you learned?
- What's next?
- **The End**

Thanks all the staff!

Sunay Poole


Caroline Liu

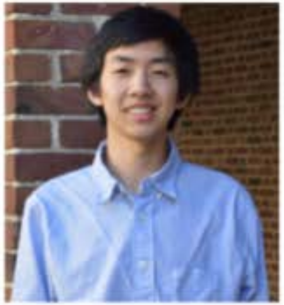

Cynthia Zhong


Daniel Fan


Ivy Li


Jerry Xu


Jie Qiu



Justin Cheng


Ryan Searcy


Justin Yokota


Kimberly Zhu


Max Lister


Robin Chu

Thanks all the staff!

Edward Zeng | Emily Wang | Kevin Chang | Kevin Zhu








Nareg Megan | Troy Sheldon | Vincent Chiang | Yijie Huang


GOOD
LUCK!


That's all Folks!