CS61C

Great Ideas in Computer Architecture

(a.k.a. Machine Structures)

Assistant Teaching Professor
Lisa Yan

Course Introduction

🎉 Welcome to CS61C!!! 🎉

Live Lecture, Enrollment Updates

Slides: cs61c.org

Please move towards the **center of each row**!

If you cannot find a fixed seat to sit in,
for safety reasons we will need to ask you to leave.

- Join lectures via Zoom: https://cs61c.org/sp26/lecture-zoom
- Asynchronously: Recording posted on bCourses every evening

Ask questions via Lecture 01 on Ed! [NOT on Zoom]

Concurrent enrollment student?
We will add you so that you can submit assignments while we wait for official enrollment.

Agenda

Slides: cs61c.org/

Thinking about Machine Structures

- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- What you need to know about this class

CS61C is NOT about C Programming…!!

It is about the **hardware-software interface.**

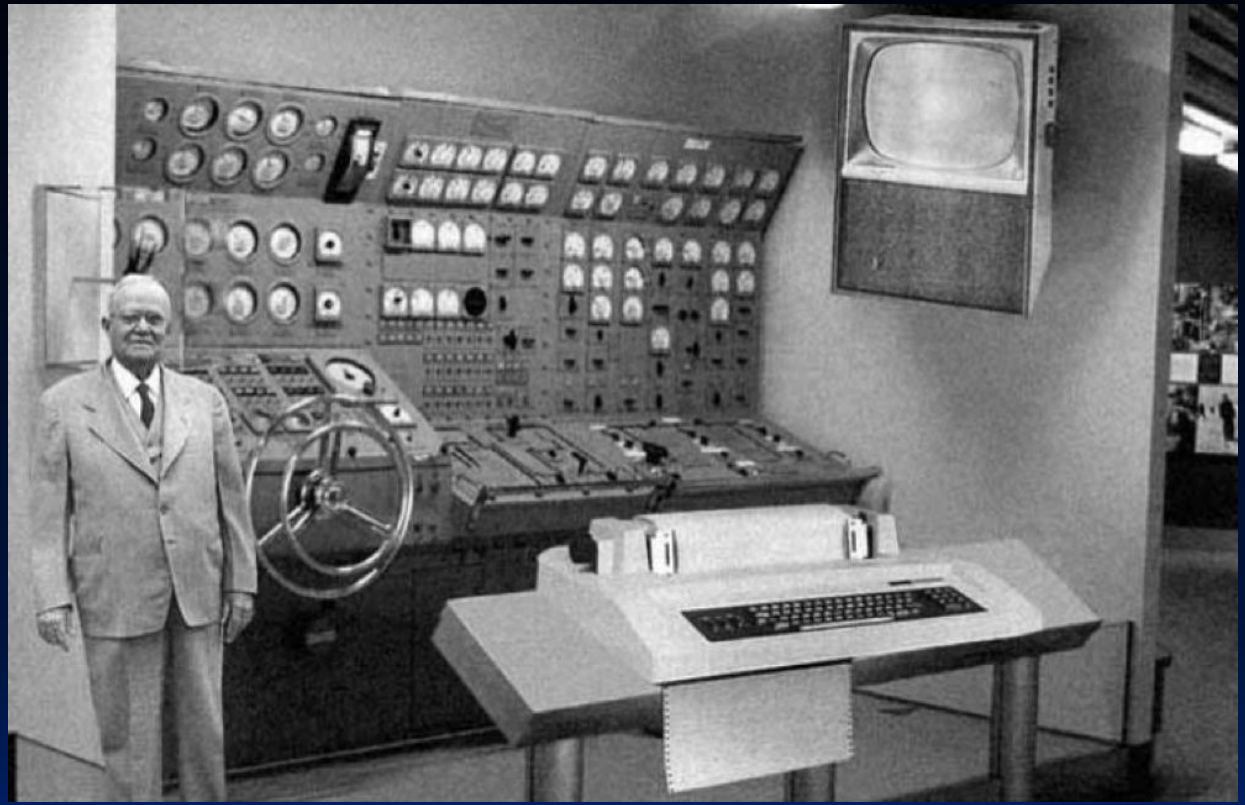
- What the programmer needs to know to achieve the highest possible performance.

Languages like C are closer to the underlying hardware.

- Unlike languages like Snap!, Python, Java
- We can talk about hardware features in higher-level terms
- C allows programmer to explicitly harness underlying hardware parallelism for high performance

Old-School CS61C

Snopes



Note:
Fake 1954 photo

New-School CS61C (1/3)

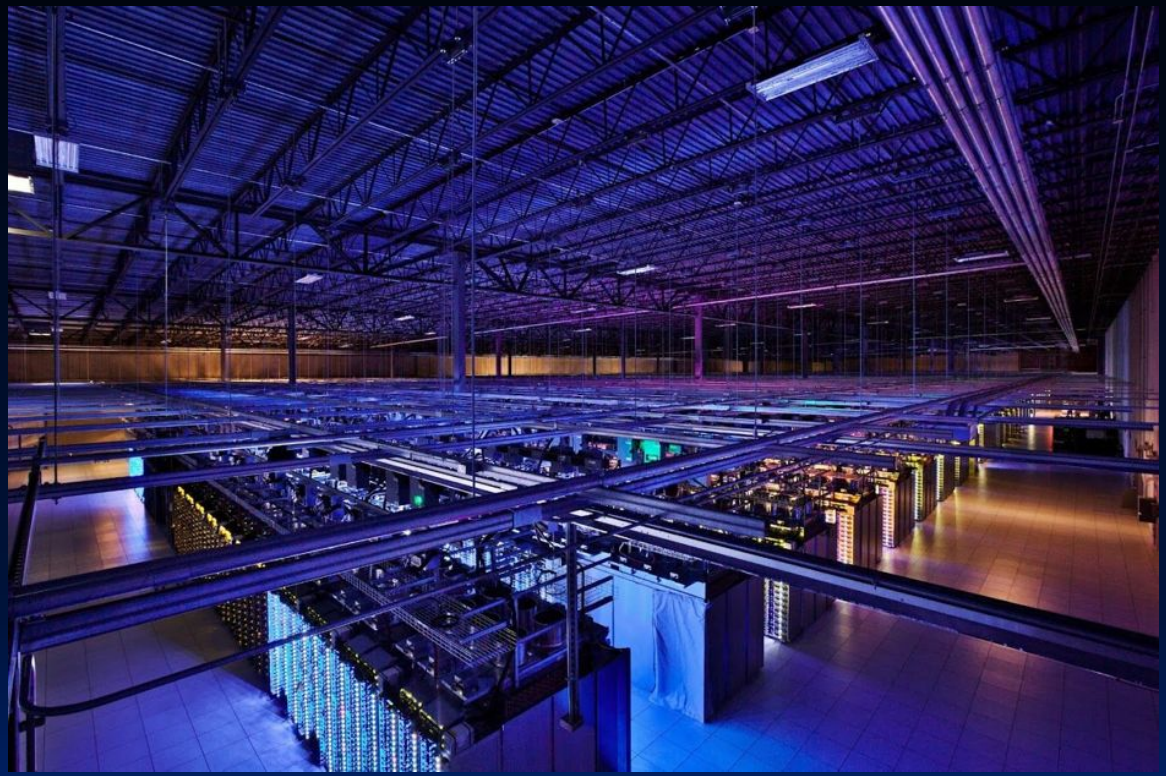
Personal Mobile Devices

Network Edge Devices

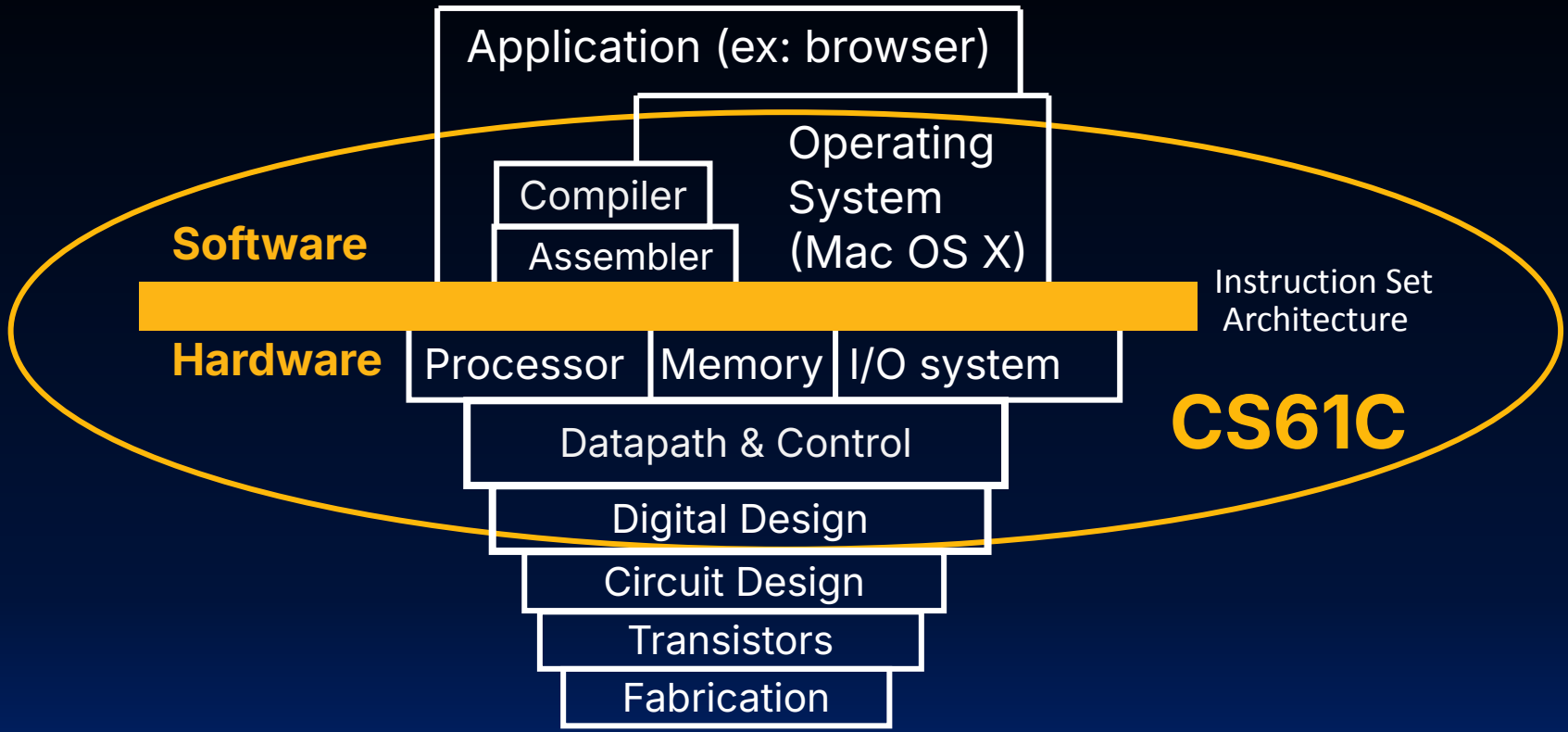
New-School CS61C (2/3)

New-School CS61C (3/3)

A popular laptop sticker



We learned Old School "Machine Structures"...



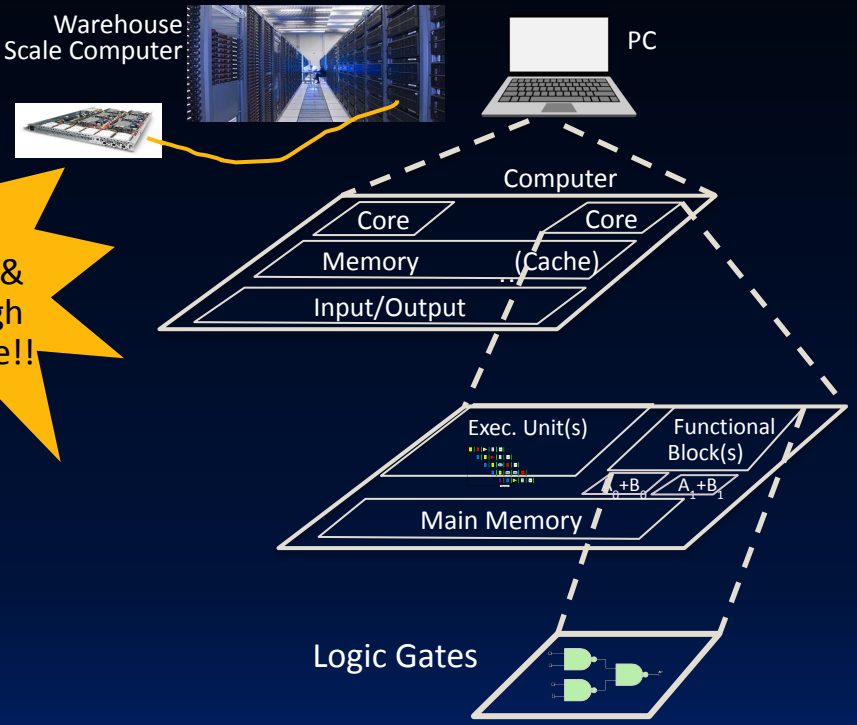
New school Machine Structures is much more complicated!

Software

- Parallel Requests
  - Assigned to computer
  - e.g., Search “Cats”
- 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 of 4 pairs of words
- Hardware descriptions
  - All gates work in parallel, at same time

Harness Parallelism & Achieve High Performance!!

Hardware



Agenda

Slides: cs61c.org/

Great Ideas in Computer Architecture

- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- What you need to know about this class

6 Great Ideas in Computer Architecture

1. Abstraction (Layers of Representation/Interpretation)
2. Moore's Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement & Improvement
6. Dependability via Redundancy

Great Idea #1: Abstraction (Levels of Representation/Interpretation)

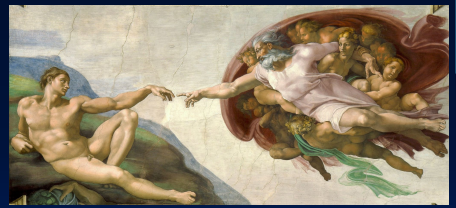
Structure and Interpretation of Computing Programs, my good friend...

```
import numpy

x = 3
x = "hello, world"
x = [3, "cs61a", True]
x = len
x = numpy
x = lambda x: x + 2
```

In CS61A: Anything can be a Python name.

We'll make HW/SW Contact!!!



The Creation of Adam, Michelangelo, c. 1512

High Level Language Program (e.g., C)

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

Compiler

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

```
lw    x3 0(x10)
lw    x4 4(x10)
sw    x4 0(x10)
sw    x3 4(x10)
```

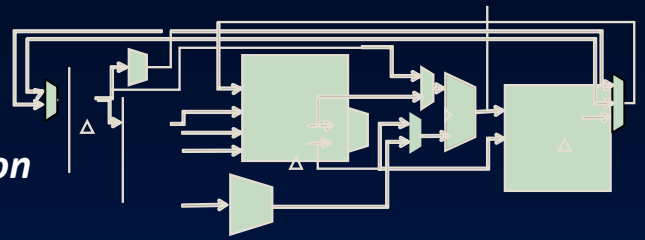
Assembler

Machine Language Program (RISC-V)

```
1000 1101 1110 0010 0000 0000 0000 0000
1000 1110 0001 0000 0000 0000 0000 0100
1010 1110 0001 0010 0000 0000 0000 0000
1010 1101 1110 0010 0000 0000 0000 0100
```

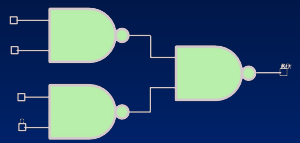
Anything can be a ***number***—data, instructions, etc.

Hardware Architecture Description (e.g., block diagrams)

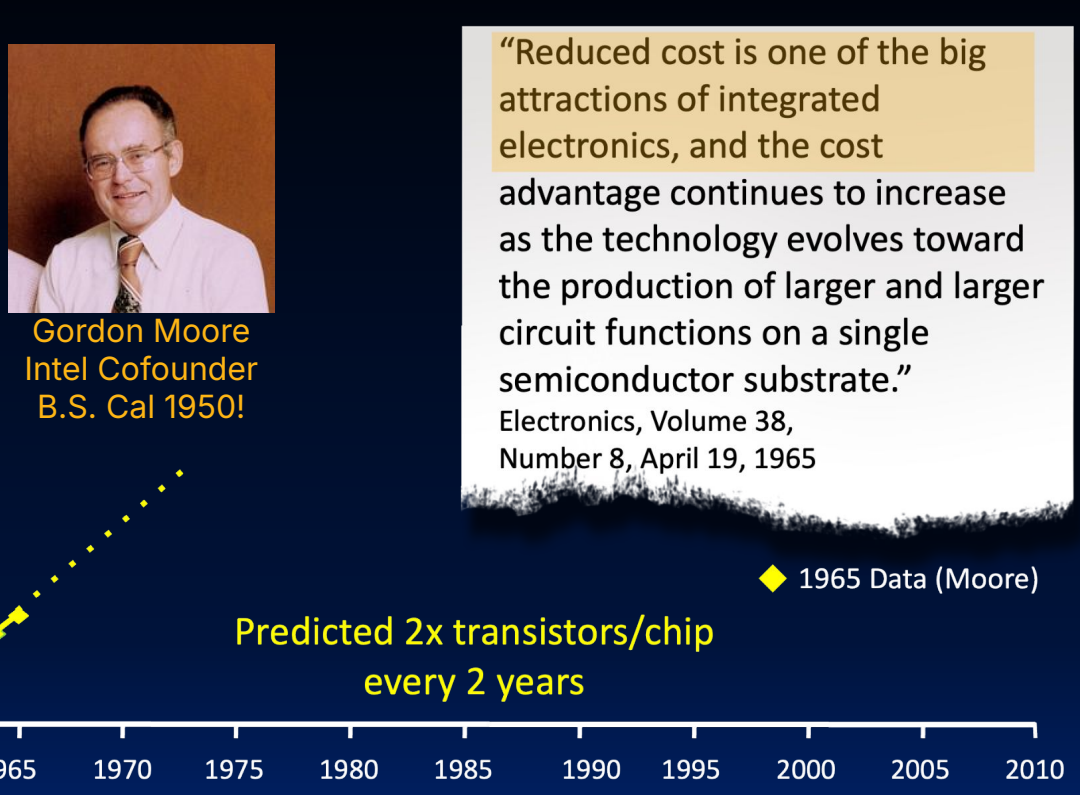


Architecture Implementation

Logic Circuit Description (Circuit Schematic Diagrams)



Great Idea #2: Moore's Law - 1965

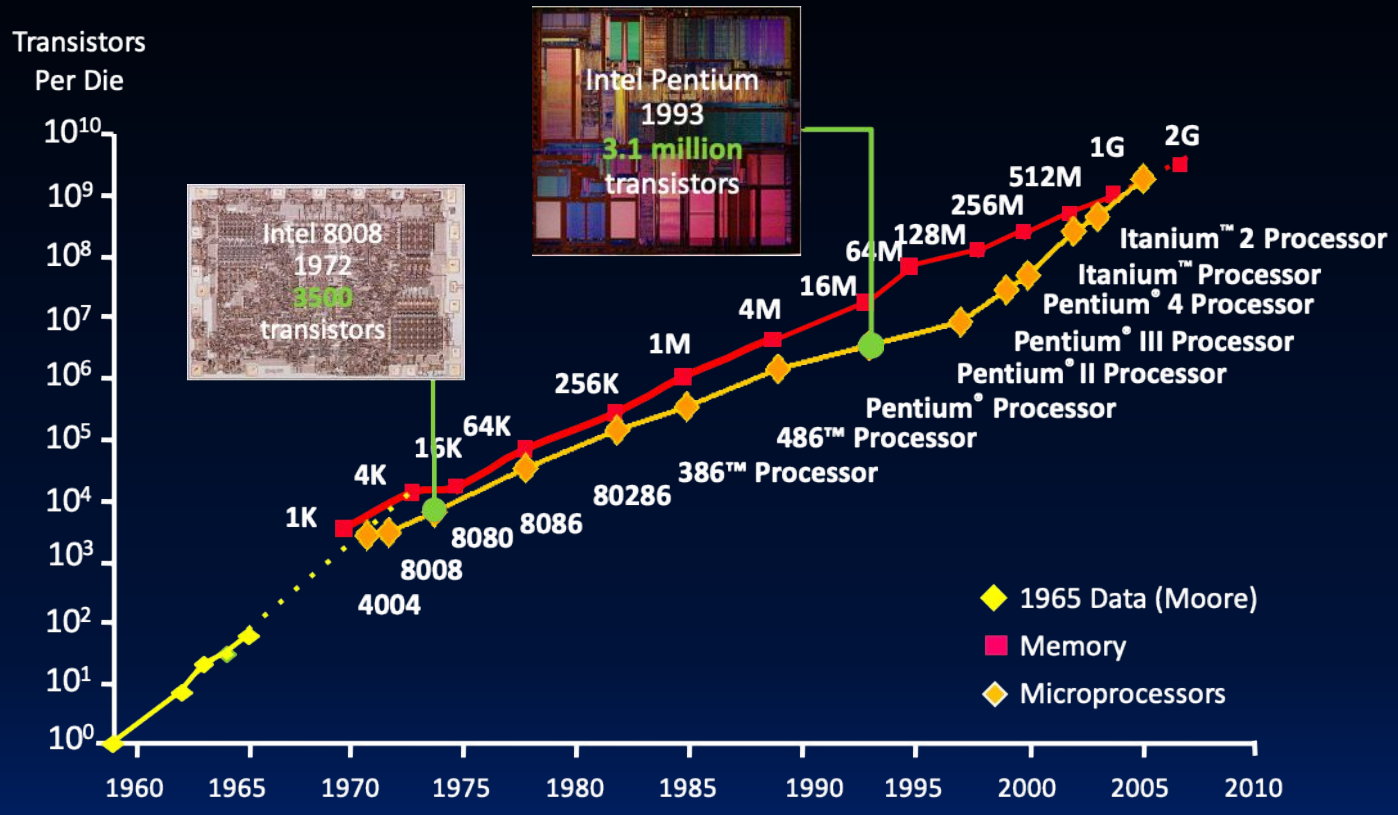


Gordon Moore
Intel Cofounder
B.S. Cal 1950!

"Reduced cost is one of the big attractions of integrated electronics, and the cost advantage continues to increase as the technology evolves toward the production of larger and larger circuit functions on a single semiconductor substrate."
Electronics, Volume 38,
Number 8, April 19, 1965

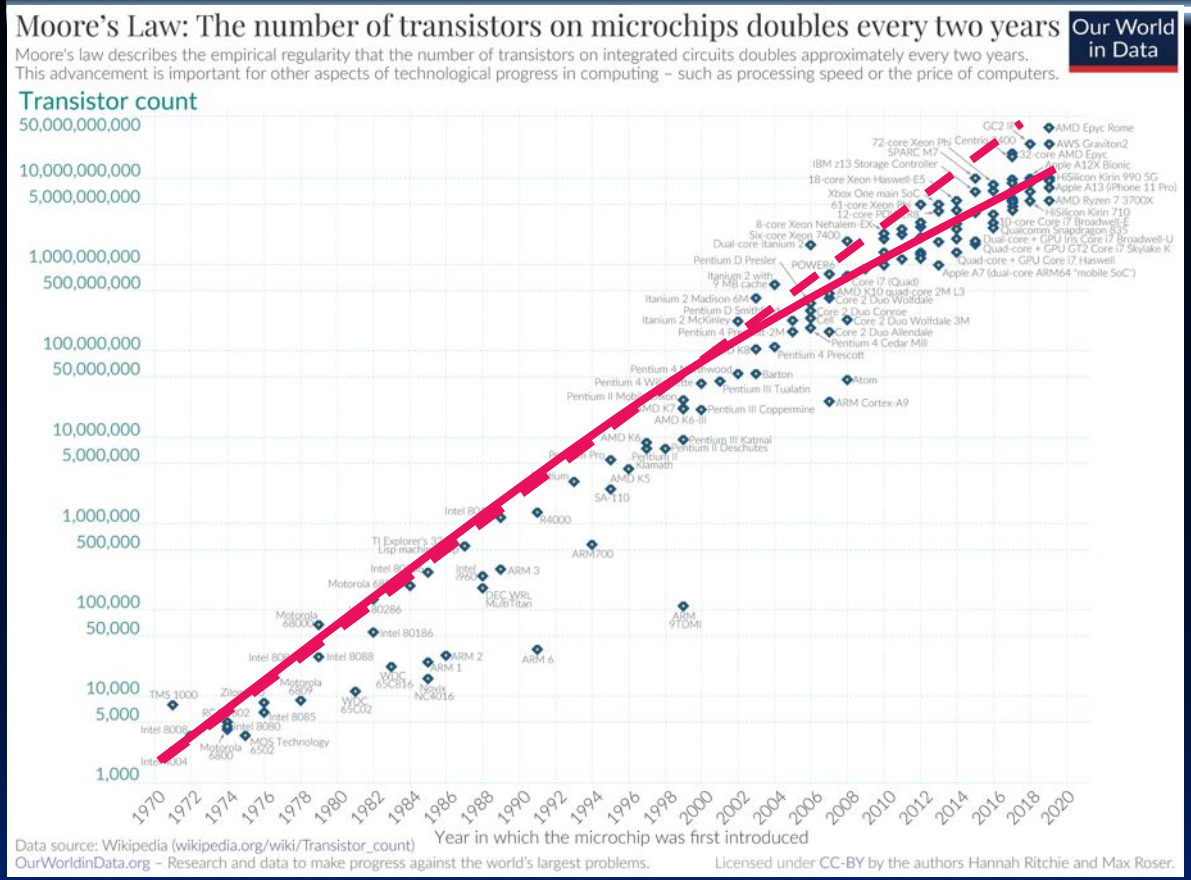
Source: Intel

Great Idea #2: Moore's Law - 2005



Source: Intel

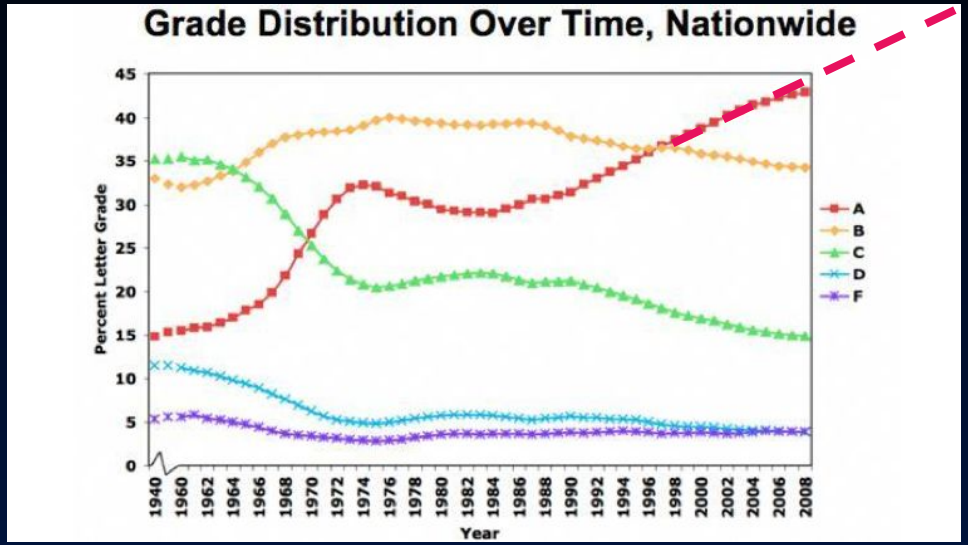
Moore's Law...?



Seems to be tapering...?
(more later)

A's Law...?

Great news:
Your grandchildren WILL all get As!

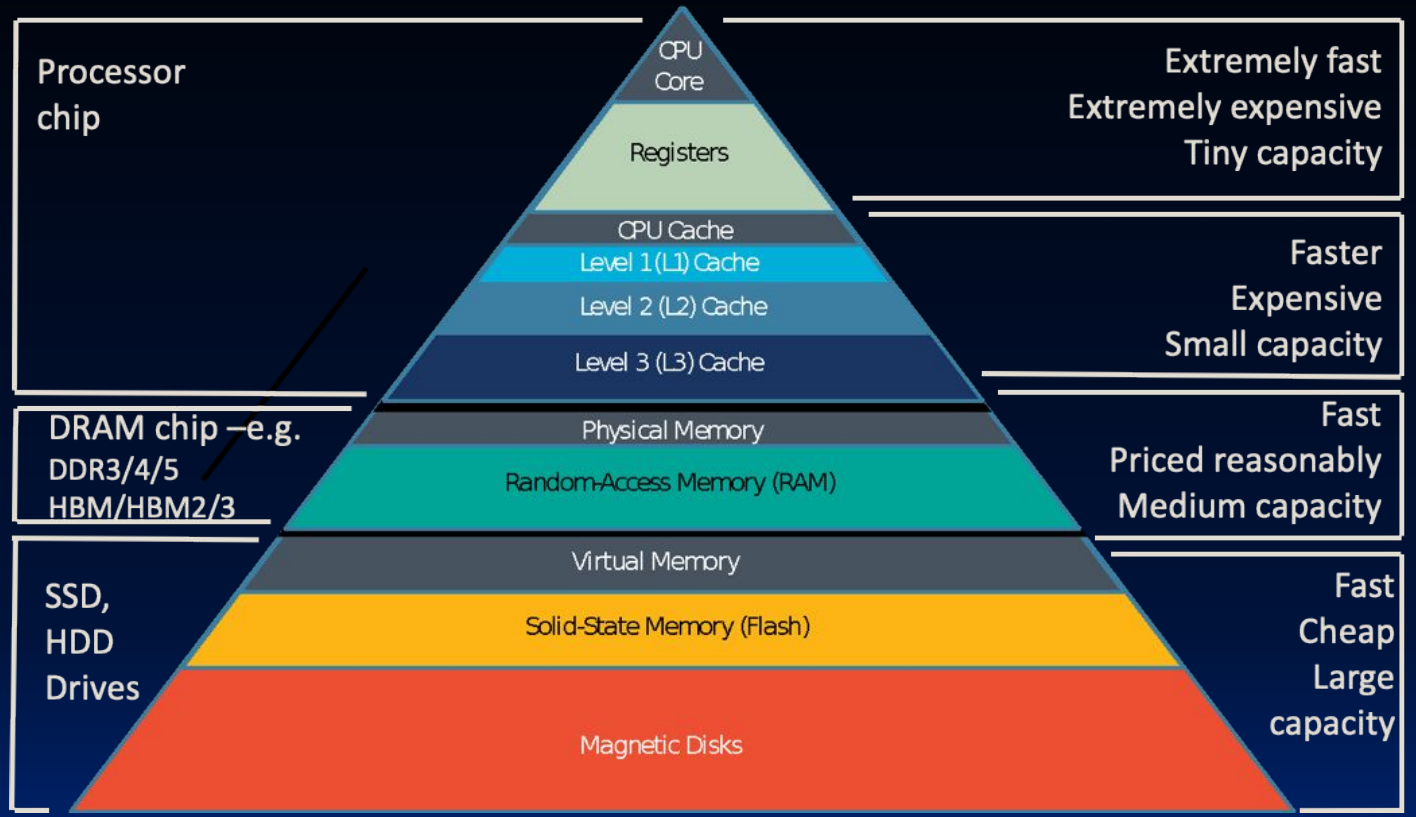


100%

2070

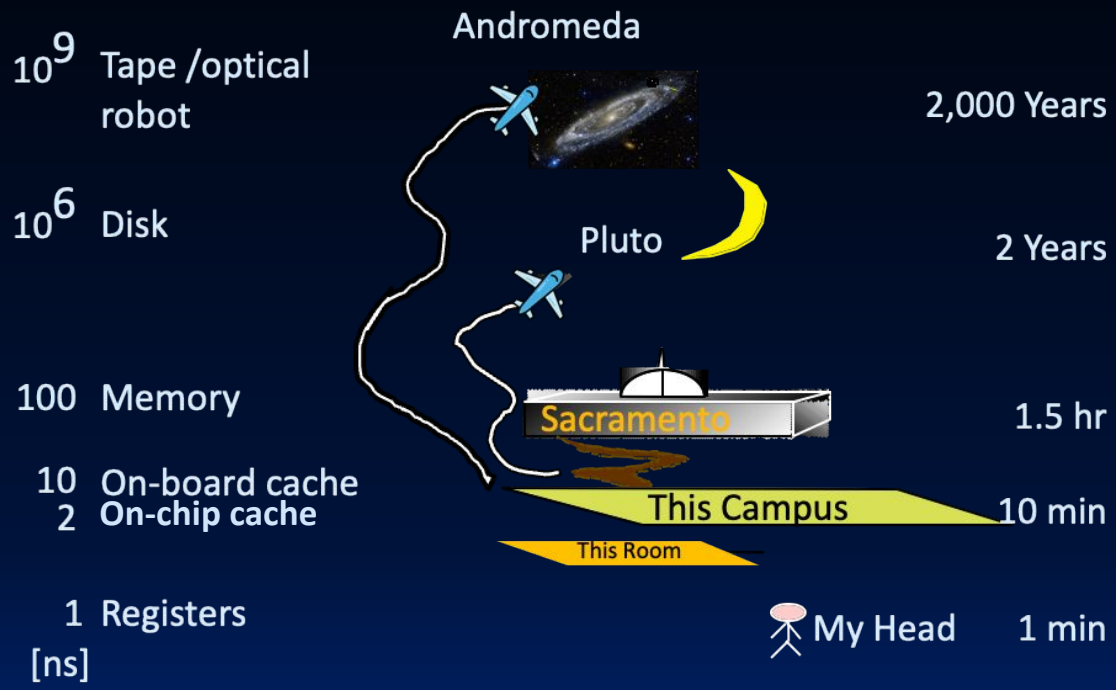
Teachers College Record Volume 114 Number 7, 2012, p. 1-23

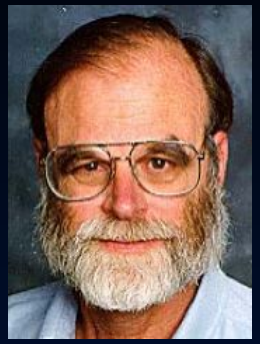
Great Idea #3: Principle of Locality / Memory Hierarchy (1/2)



Great Idea #3: Principle of Locality / Memory Hierarchy (2/2)

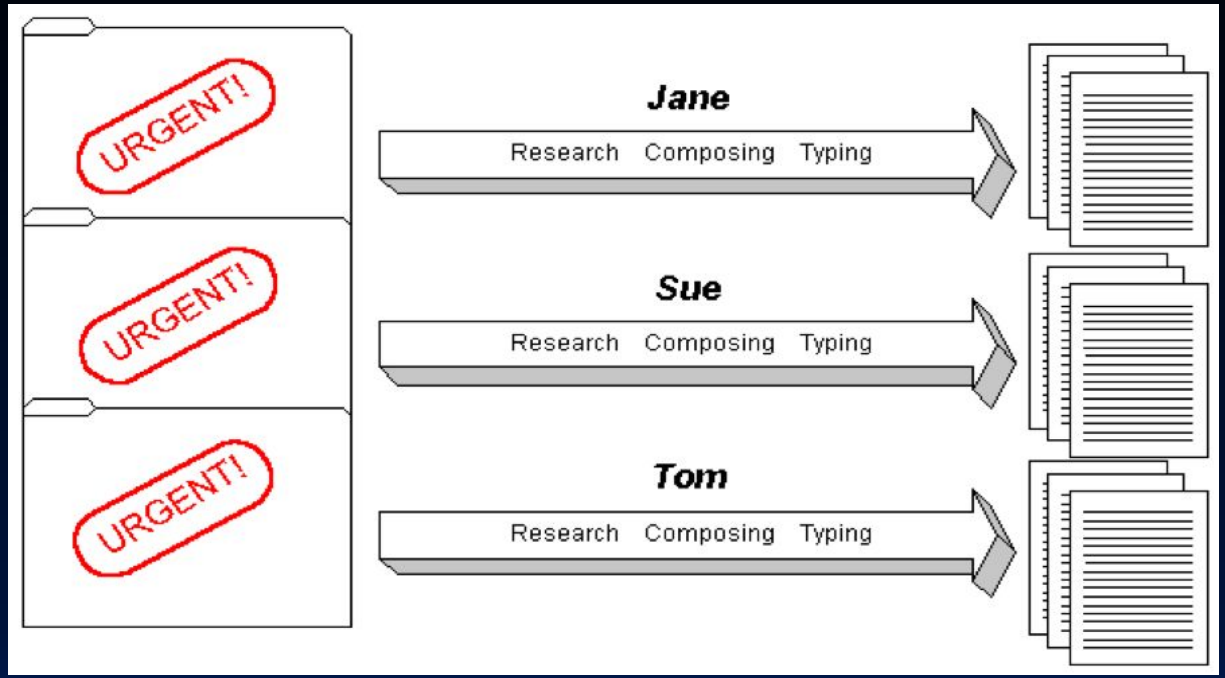
Storage Latency Analogy: How Far Away is the Data?



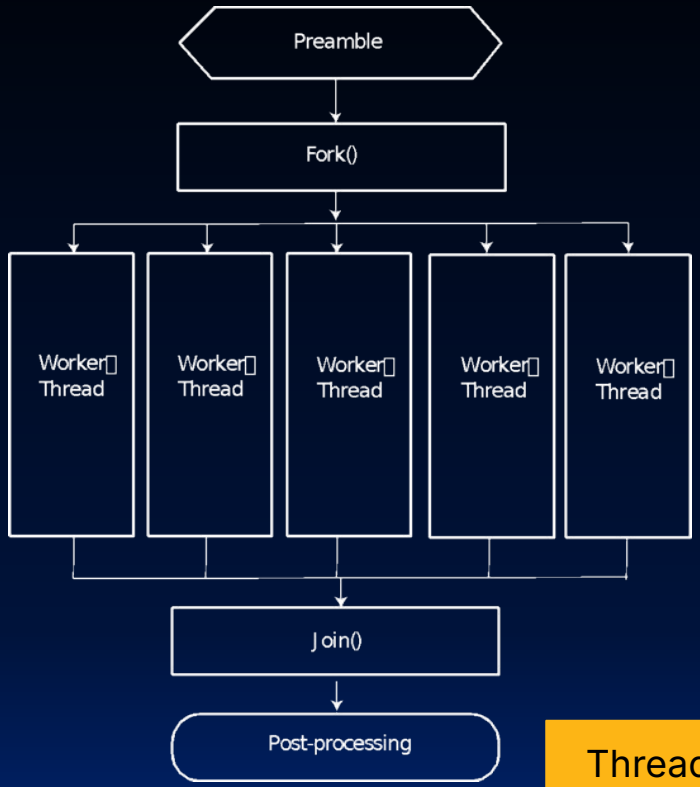


Jim Gray
Turing Award
B.S. Cal 1966
Ph.D. Cal 1969

Great Idea #4: Parallelism



Great Idea #4: Parallelism



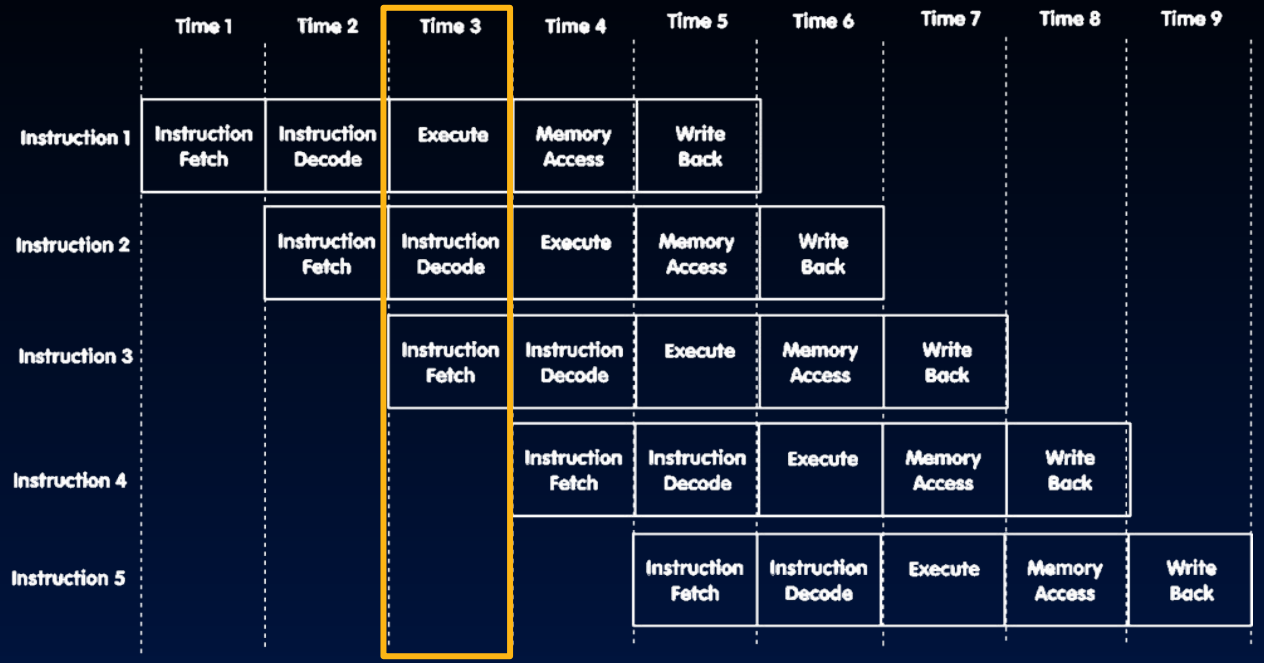
Thread-Level Parallelism

Great Idea #4: Parallelism

Instruction-Level Parallelism

In time slot 3:

- Instruction 1 is being executed
- Instruction 2 is being decoded
- Instruction 3 is being fetched from memory



...Amdahl's Law

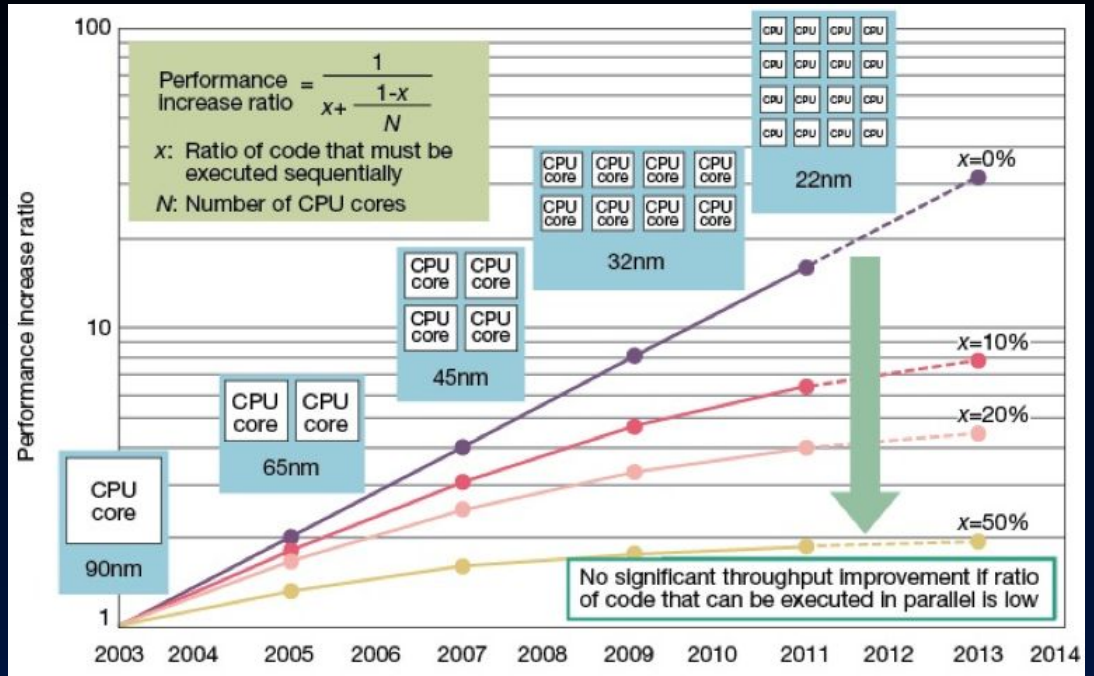
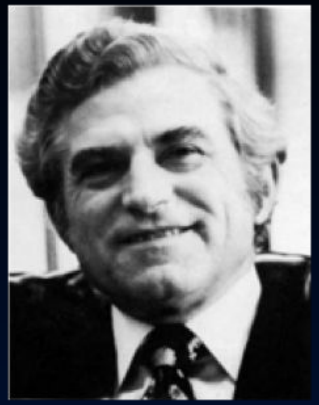


Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor performance. Diagram assumes no overhead in parallel processing. Years shown for design rules based on Intel planned and actual technology. Core count assumed to double for each rule generation.



Gene Amdahl
Computer
Pioneer

Caveat!
(more later)

Great Idea #5: Performance Measurement & Improvement

Match application to underlying hardware to exploit:

- Locality;
- Parallelism;
- Special hardware features, like specialized instructions (e.g., matrix manipulation).

Metrics of Latency/Throughput:

- **How long** to set the problem up and complete it (or how many tasks can be completed in a given time)
- How much faster does it execute **once it gets going**

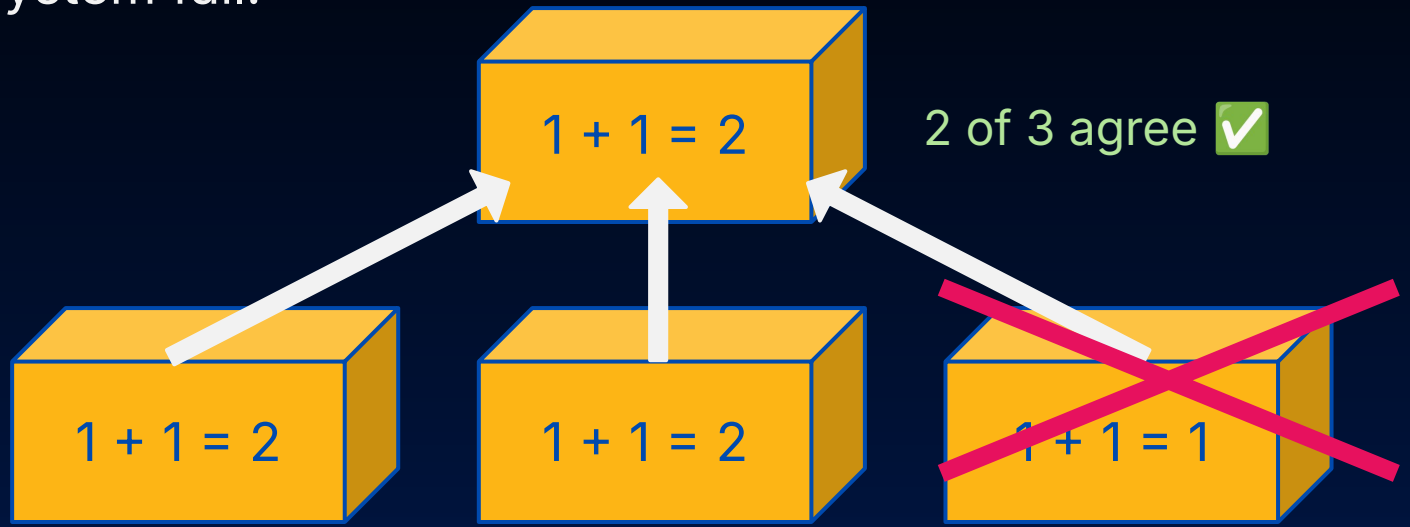
Great Idea #6: Dependability via Redundancy

Unintended transistor behavior can be caused by unintended electron flow from cosmic rays (among other reasons)!

https://www.exploratorium.edu/exhibits/cloud-chamber

Great Idea #6: Dependability via Redundancy

Design with redundancy so that a failing piece doesn't make the whole system fail.



Great Idea #6: Dependability via Redundancy (3/3)

Applies to everything from datacenters to storage to memory...to instructors!

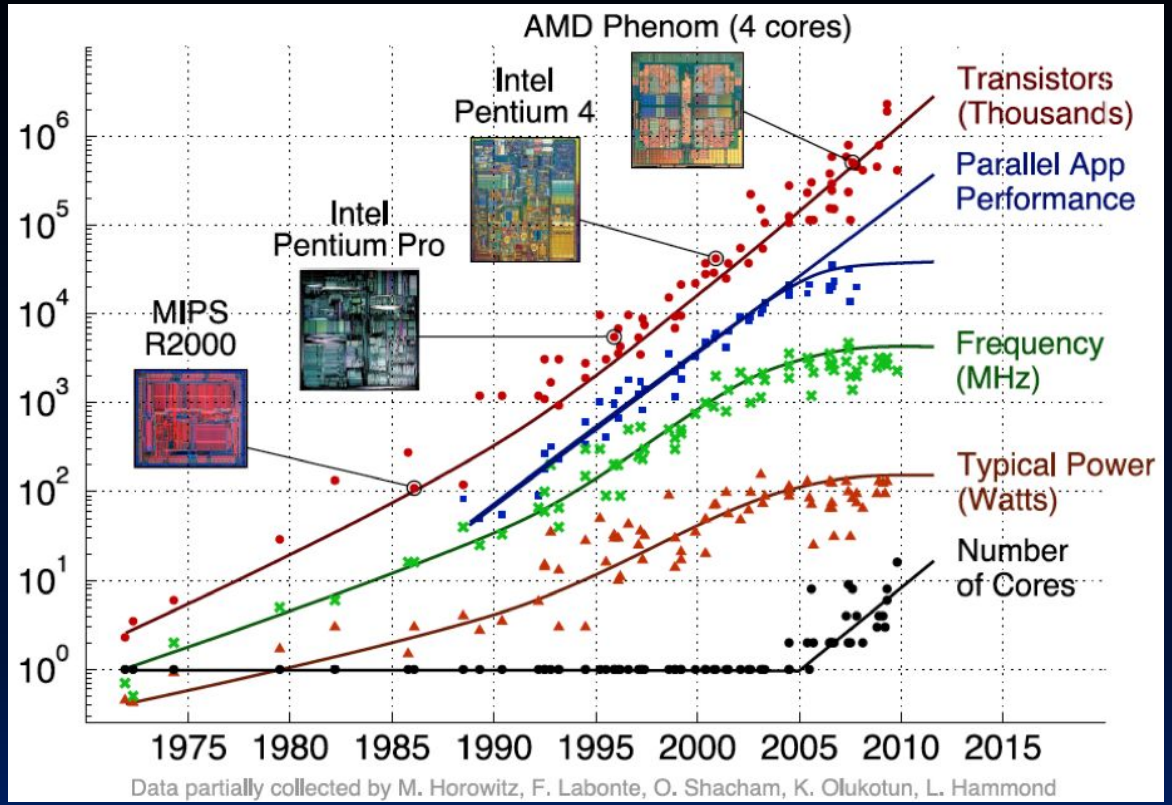
- Increasing **transistor density** reduces the cost of redundancy
- Error-correcting codes: Redundant **memory bits** so that can lose 1 bit but no data
- RAID: Redundant **disks** so that can lose 1 disk but not lose data
- Redundant **datacenters** so that can lose 1 datacenter but Internet service stays online

(dramatic pause)

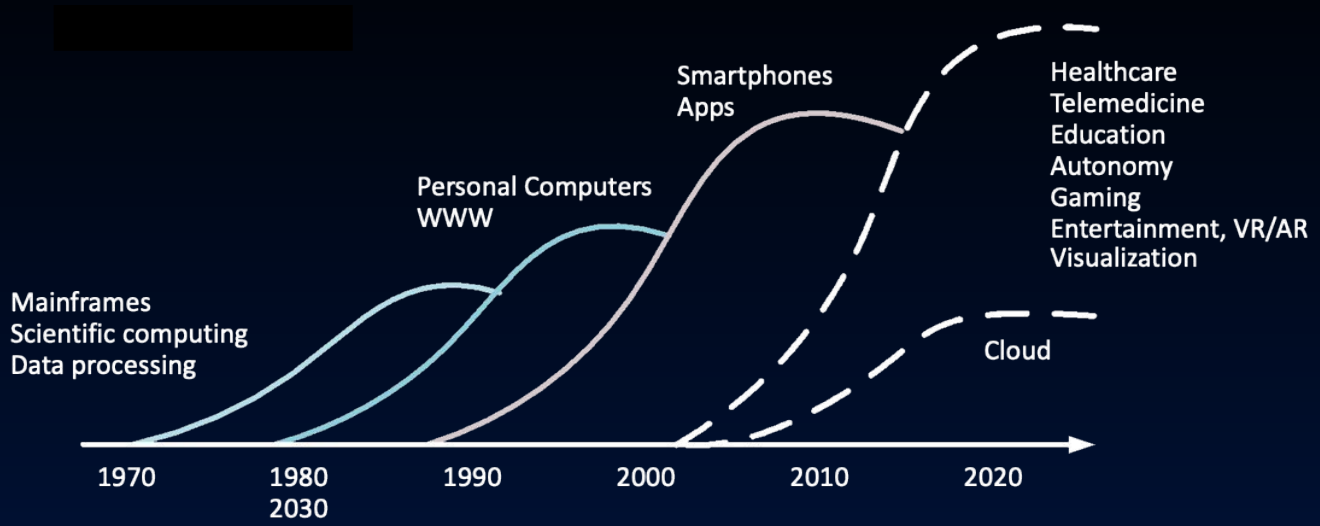
Why is computer architecture exciting ***today***?

Reason 1: Changing Constraints

- Moore's Law ending
- Power limitations
- Amdahl's Law



Reason 2: Era of Domain-Specific Computing



- Number of deployed devices continues to grow, but there is no single killer application.
  - Diversification of needs, architectures
  - Machine learning is common for most domains

Old Conventional Wisdom

- Moore's Law + Dennard Scaling = faster, cheaper, lower-power **general-purpose computers** each year
  - In glory days, 1%/week performance improvement!
- Dumb to compete by designing parallel or specialized computers
  - By time you've finished design, the next generation of general-purpose will beat you!

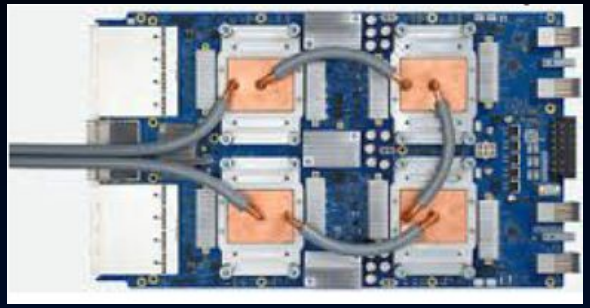
New Conventional Wisdom

Each domain requires heterogeneous systems.

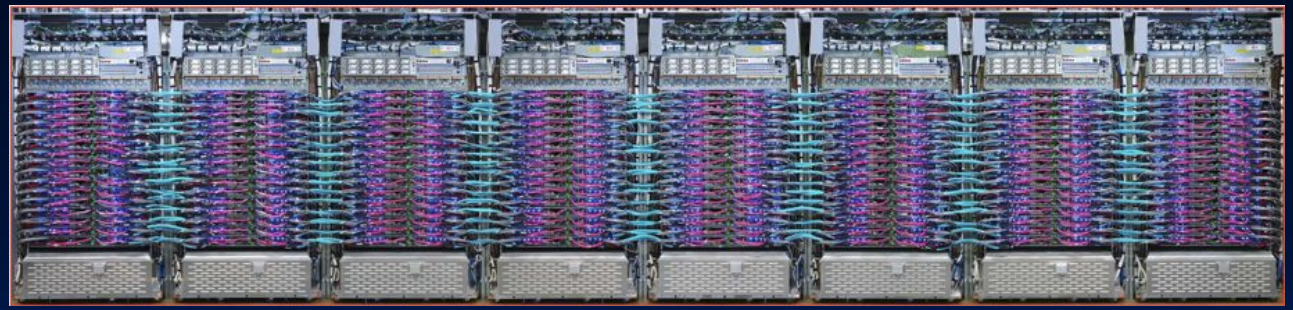
- Multiple processor cores
- GPUs,
- NPUs,
- accelerators,
- interfaces,
- memory, ...

Apple M14 (link)

New Conventional Wisdom



Google TPU3
Specialized Engine for training
Neural Networks
Deployed in cloud



1024 chips, > 100PetaFLOPs

Top Companies in the World

Job/Internship Interviews: They grill you with technical questions, so it's what you know and can do, not your GPA.

- CS61C gives good stuff to say!

Name	M. Cap
NVIDIA 1 NVDA	$4.380 T
Alphabet (Google) 2 GOOG	$3.955 T
Apple 3 AAPL	$3.633 T
Microsoft 4 MSFT	$3.271 T
Amazon 5 AMZN	$2.428 T
TSMC 6 TSM	$1.696 T

Largest Companies by Market Cap

Patterson and Hennessy win Turing Award! (2017)

Innovations in computer architecture often defy conventional wisdom.

https://engineering.berkeley.edu/news/2018/06/patterson-wins-turing-award/

Agenda

Slides: cs61c.org/

What you need to know about this class

- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- What you need to know about this class

Our Spring 2026 Course Staff













Course Information

Course Website: https://cs61c.org/
Instructor: Lisa Yan

- Website: Course resources (lectures, handouts, assignments)
- EdStem
  - Every announcement, discussion, clarification happens here

Today's required reading

Read our Course Policies in full:

https://cs61c.org/sp26/policies

Typical Week

Monday	Tuesday	Wednesday	Thursday	Friday
Lecture		Lecture		Lecture
Discussion (in-person) [or recordings]		Lab (Gradescope) [with drop-in, in-person help]		
	Assignment due (11:59pm)		Assignment due (11:59pm)	

Usually: Homework Lab OR project

Next week (Week 2)

Monday	Tuesday	Wednesday	Thursday	Friday
Lecture		Lecture		Lecture
Discussion 1 (audit, no attendance*)		Lab 1 (Gradescope) [with drop-in, in-person help]		
	Lab 0 (Set up) **Welcome Form**		Lab 1 AND HW 1	

*Welcome Form: Submit discussion preference.

- Regular (1-hr) vs. Bridge (2-hr) vs. Video discussion
- Discussion attendance taken starting **Week 3**

Course policy highlights

Lateness and Accommodations Policy:
flexible assignment extensions

- If you need just a little time, pre-approved automated extension
- If you need more time, one-time custom extension
- If your life is falling apart, or you need else, talk to us!

Exams: Quest (CBTF), paper Midterm, paper Final
Absolute Grading! (not curved!)

DSP Students: Submit your letter of accommodation ASAP!

Yoda says…

"Always in motion, the future is…"

Our precise schedule (lectures, assignments, labs) may change slightly over the course of the semester.

(somber pause)

We expect that what you hand in for this course is **yours**.

Collaboration and Independent Work Policy

It is **NOT** acceptable:

- To copy solutions from other students.
- To copy (or start your) solutions from the Web.
- **To be irresponsible** and create opportunities for unscrupulous students (e.g., public GitHubs, walking away while logged on, leaving notes around, etc.).

You are encouraged to help teach others to **debug**.

- Beyond that, we don't want you sharing approaches or ideas or code or whiteboarding with other students. **The "pseudocode/algorithm" is sometimes the entire point!**
  - HKN/tutoring working through pseudocode are not allowed.
- On Ed: Answer questions that help classmates **debug**. Don't share code, nor code snippets.

Collaboration and Independent Work Policy

Assignment Collaboration:

- All homework and labs are to be your work ALONE.
- Projects can be done either solo or in pairs.
  - **Partners commit to responsibility to working synchronously with each other, not other partner teams.**

Excessive Collaboration and Academic Misconduct

We have tools, developed over many years, for detecting this. **You WILL be caught, and the penalties WILL be severe.** **If you have questions whether a behavior is crossing the line, ask!**

- At minimum: negative points in the assignment AND a report to the Center of Student Conduct.
- Both Giver and Receiver are equally culpable and suffer equal penalties.

50-100 students are caught every semester.

- We also check **intermediate student work**—accounted for 60% of Fall 2023 cases.
- **Plagiarism at any point in your work is unacceptable.**

What about working with AI?

- **Acceptable uses of AI**
  - "Teach me <CS61C topic>"
    - Try: "teach it to me as if I were in grade school" (then middle school → high school → college) if lost to get big ideas first.
  - "Do you have any mnemonics to help me remember X?"
  - "Generate some exam problems for me to practice"
  - "Can you teach me debugging techniques for C or RISC-V?"

- **Unacceptable uses of AI (and Googling and other people)**
  - "Here's the project prompt, please do it for me"
  - "Write this function for me"
  - "Here's my broken code, debug it for me"

Finally: Our goals as instructors

To make your experience in CS61C as enjoyable & informative as possible

- Humor, enthusiasm, guests, technology-in-the-news
- Fun, challenging projects & HW
- Pro-student accommodations policies
- Peer discussion opportunities in section, learn how to learn

To maintain Cal standards of excellence

- Projects & exams as rigorous as every year.

With all of that said—you will have plenty of opportunities to demonstrate success.

- Our ultimate goal is for you to **learn the material**, not to stress you out.
- **If you are feeling lost**, **please reach out**. It is much better to do so than to violate our trust. We are in this together!

Welcome to CS61C!!!

CS61C: Learn 6 **great ideas** in computer architecture to enable high performance programming via parallelism, not just learn C.

1. Abstraction (Layers of Representation / Interpretation)
2. Moore's Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement and Improvement
6. Dependability via Redundancy