UC Berkeley
Teaching Professor
Dan Garcia

CS61C

Great Ideas in Computer Architecture (a.k.a. Machine Structures)

UC Berkeley
Professor
Bora Nikolić

Welcome, everyone!!
Course Introduction

Agenda

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

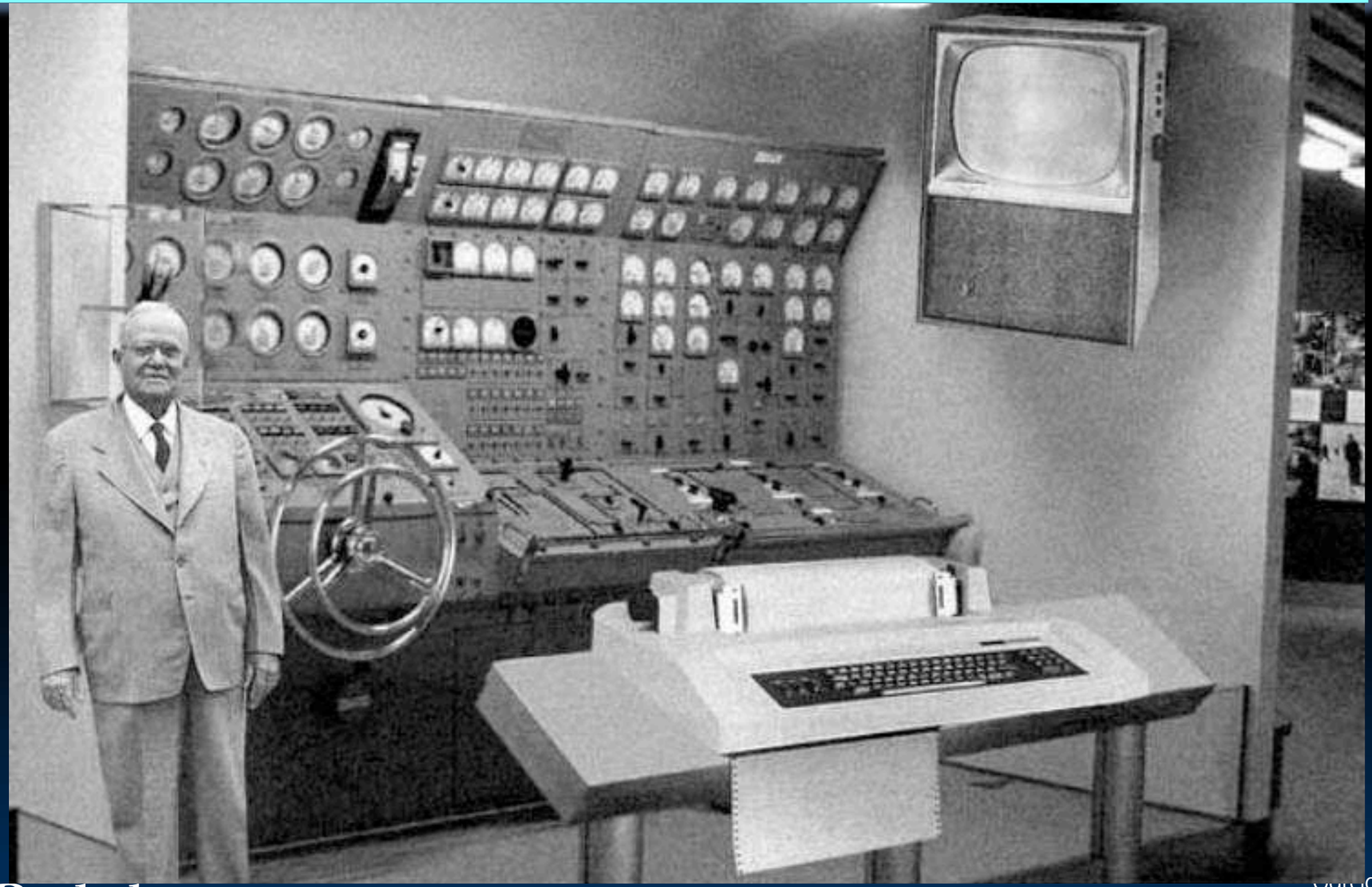
Agenda

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

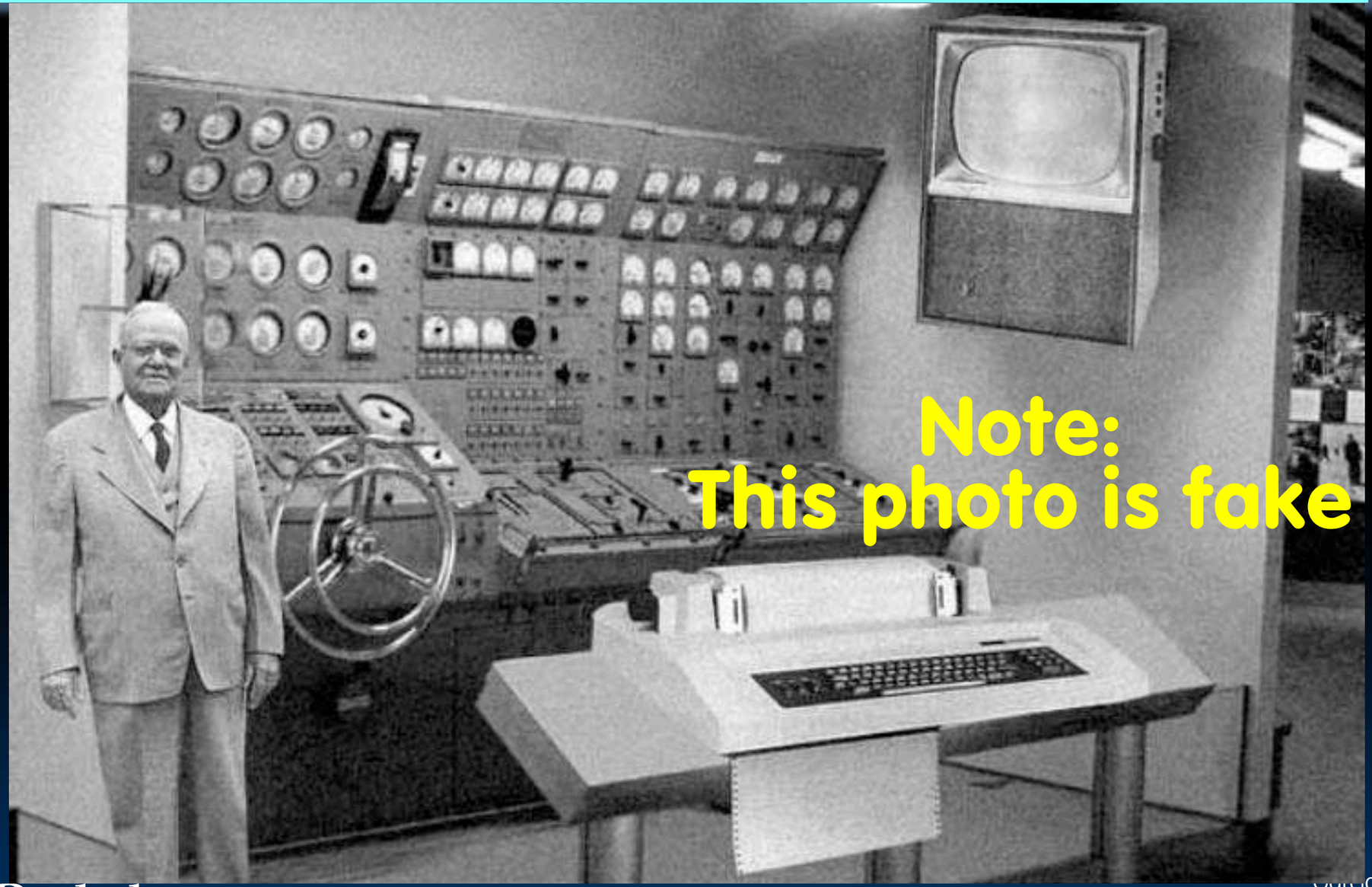
CS61C is NOT about C Programming

- **It is about the hardware-software interface**
  - What does the programmer need 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
  - Allows programmer to explicitly harness underlying hardware parallelism for high performance

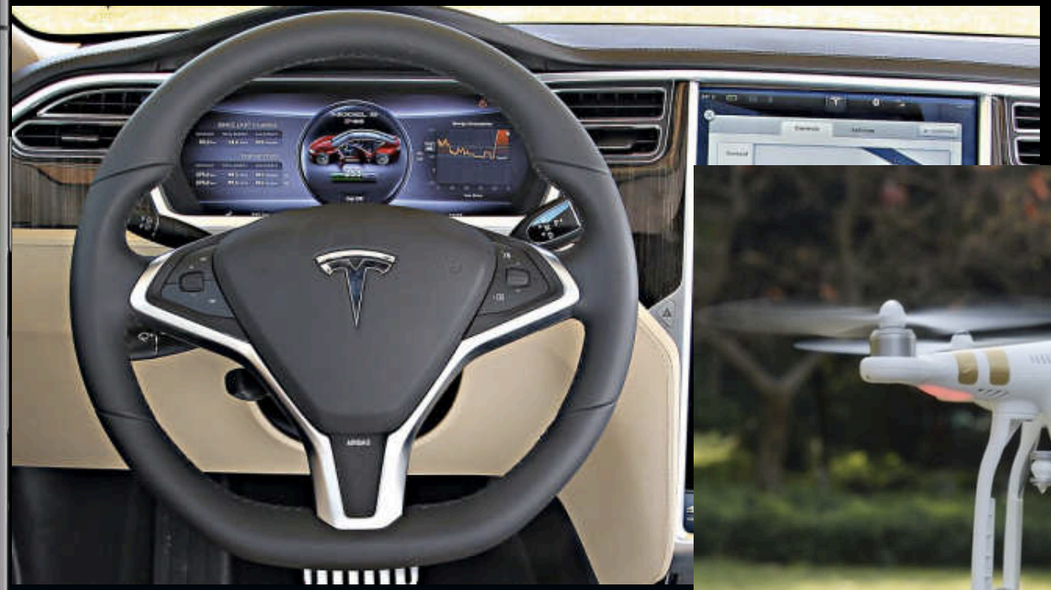
Old School CS61C



Old School CS61C



New School CS61C (1/3)



Personal Mobile Devices

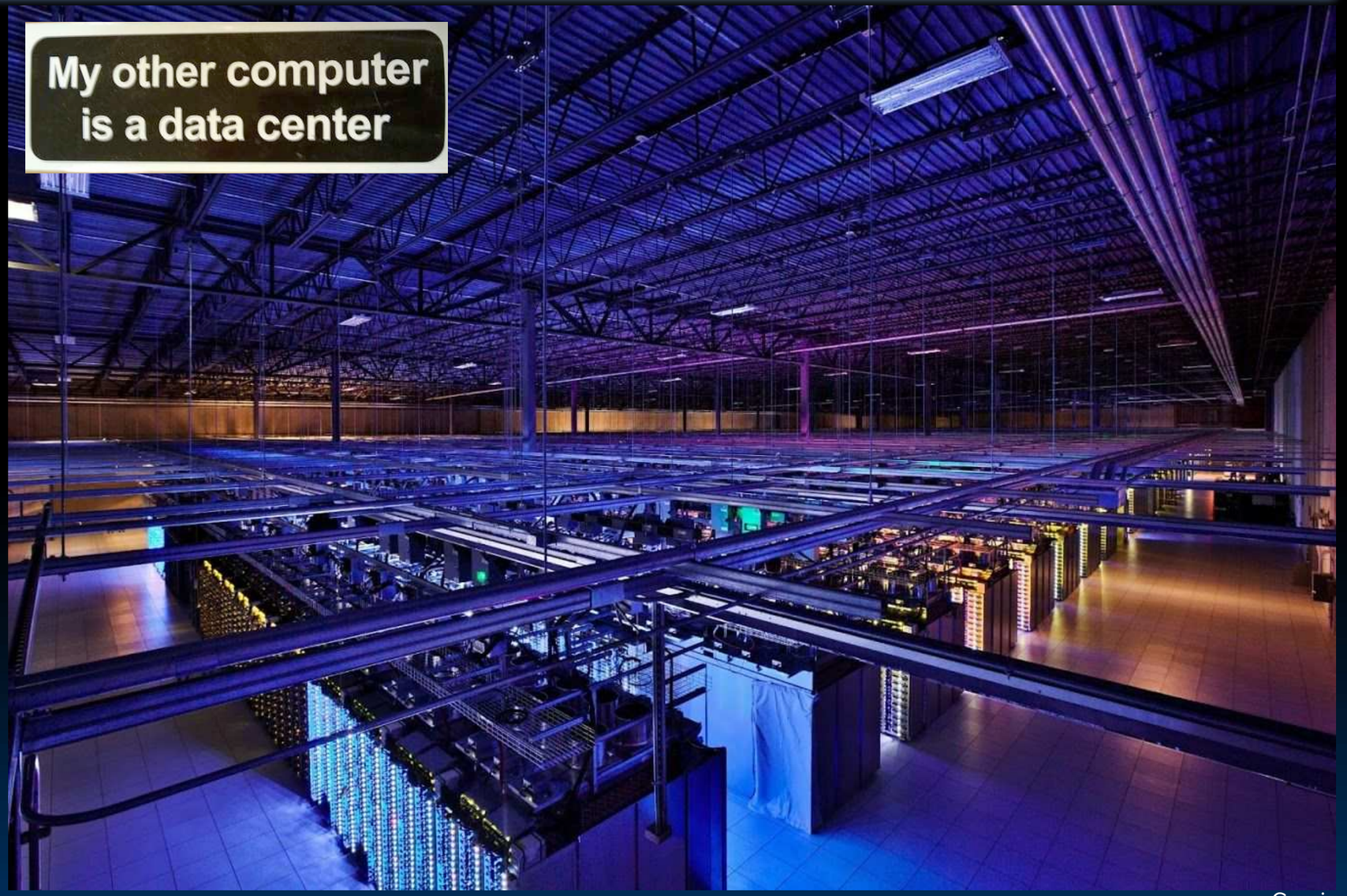
Network Edge Devices

Berkeley
UNIVERSITY OF CALIFORNIA

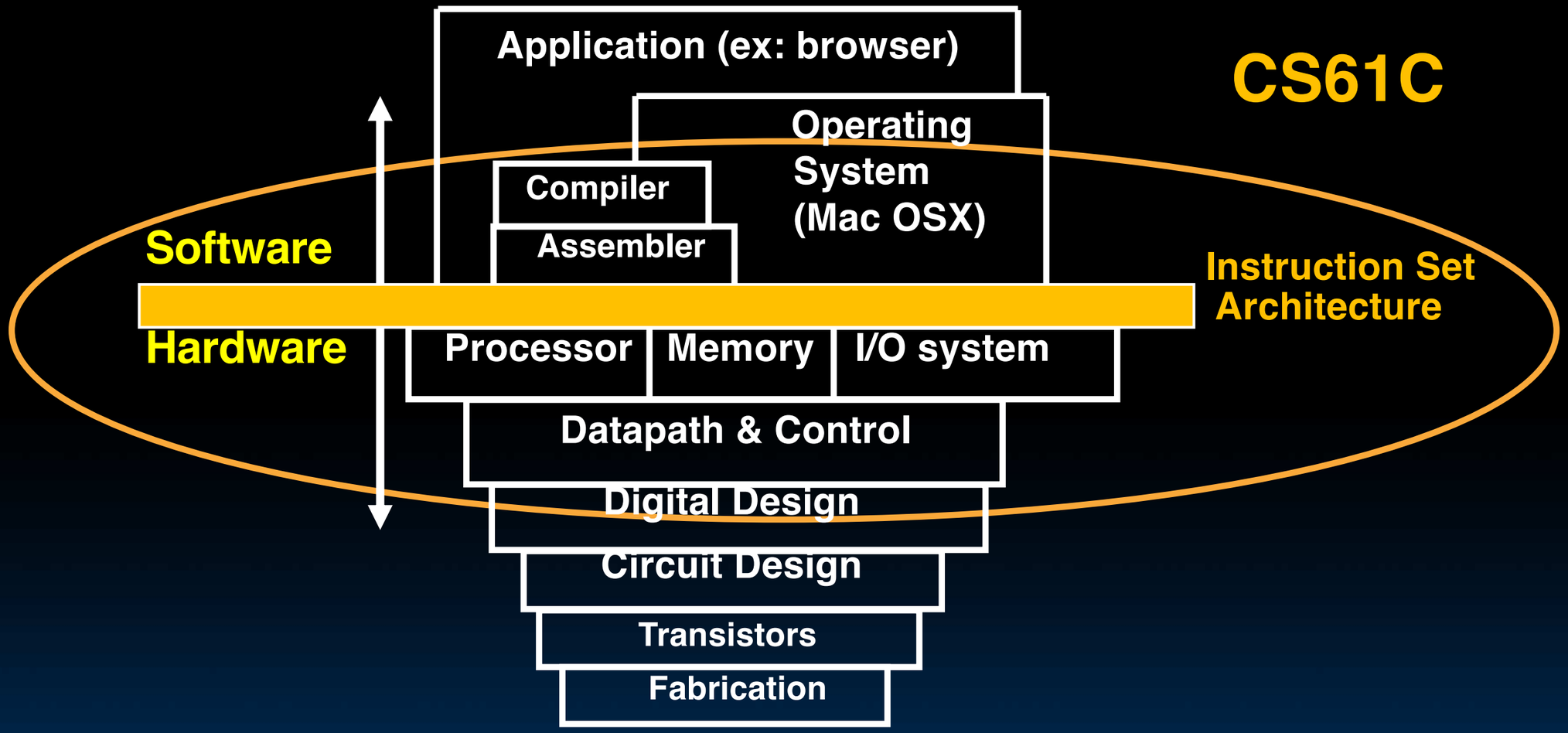
Garcia, Nikolić

New School CS61C (2/3)

New School CS61C (3/3)



Old School Machine Structures



New-School Machine Structures (It's a bit 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

Warehouse Scale Computer

Smart Phone

Agenda

- 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)

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)
```

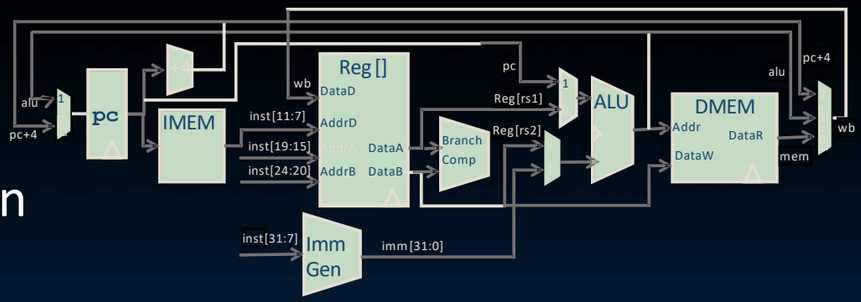
Anything can be represented as a number, i.e., data or instructions

Assembler

Machine Language Program (RISC-V)

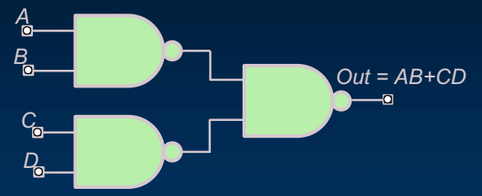
```
1000 1101 1110 0010 0000 0000 0000 0000
1000 1110 0001 0000 0000 0000 0000 0100
1010 1110 0001 0010 0000 0000 0000 0000
1010 1101 1110 0010 0000 0000 0000 0100
```

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

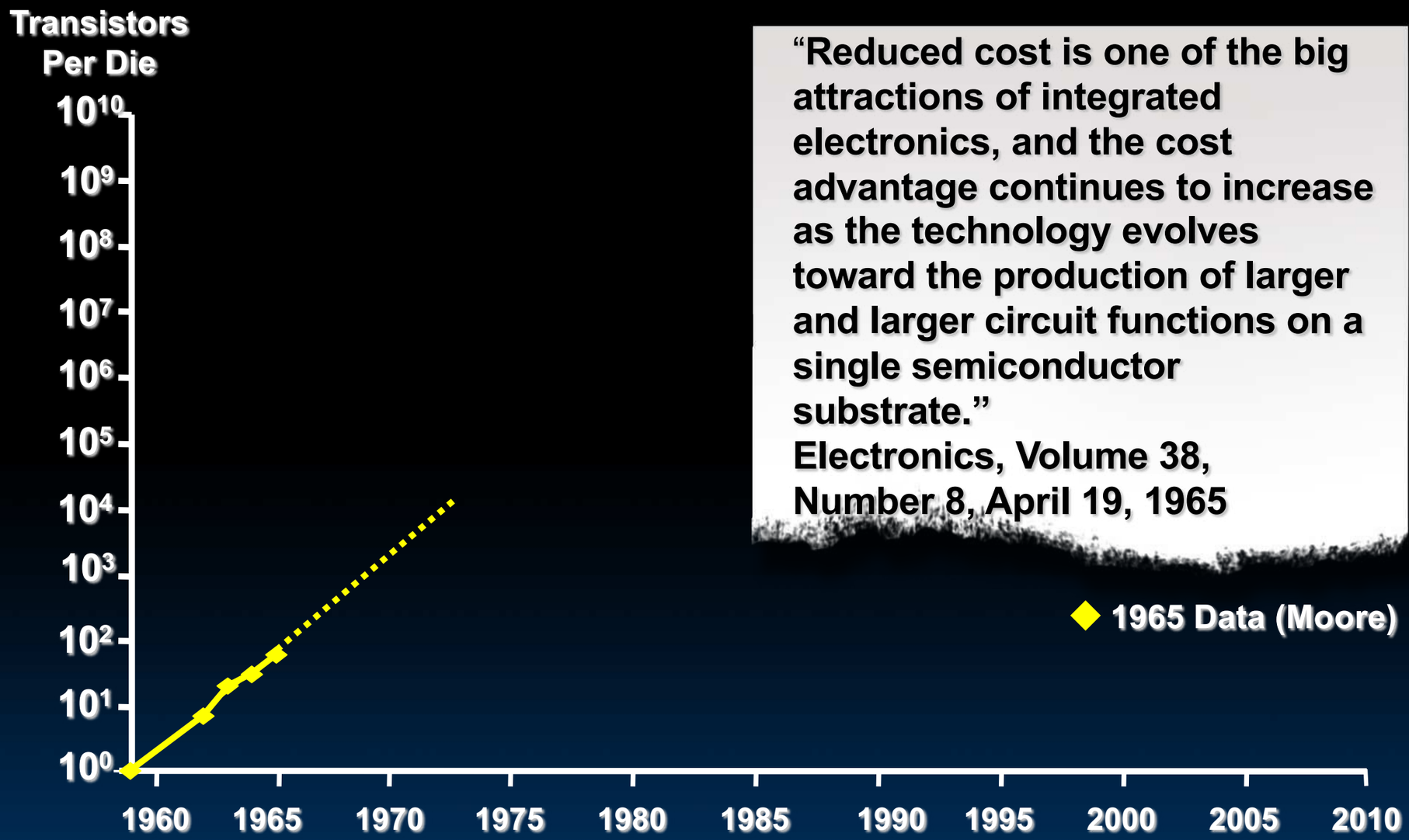


Architecture Implementation

Logic Circuit Description (Circuit Schematic Diagrams)



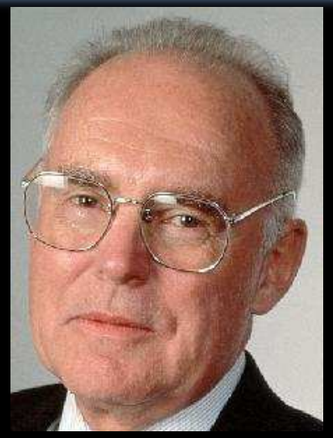
Moore's Law - 1965



"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

Moore's Law - 1965



Gordon Moore
Intel Cofounder
B.S. Cal 1950!

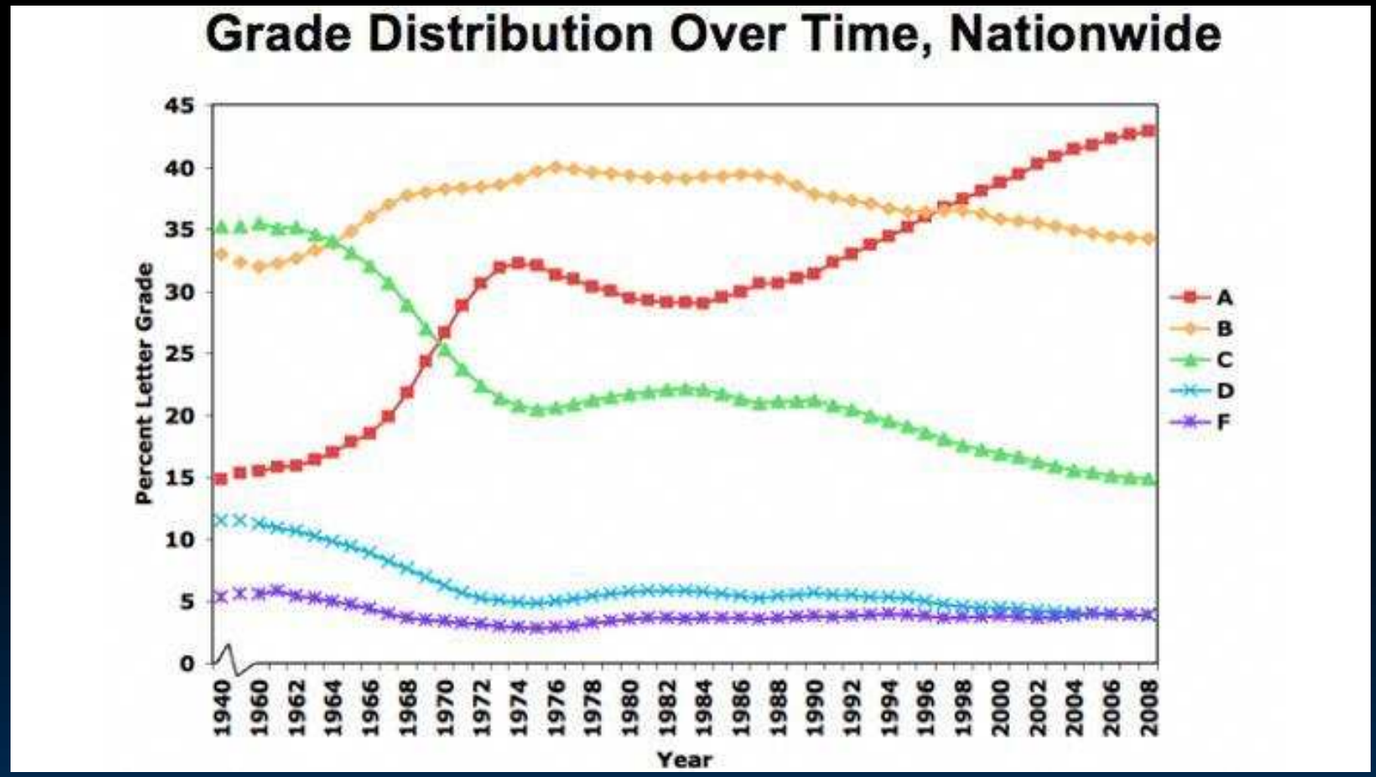
"Reduced cost is one of the big attractions of integrated electronics, and the cost advantage continues to increase as the technology evolves toward the production of larger and larger circuit functions on a single semiconductor substrate."
Electronics, Volume 38, Number 8, April 19, 1965

Source: Intel

Moore's Law - 2005

Source: Intel

A's Law



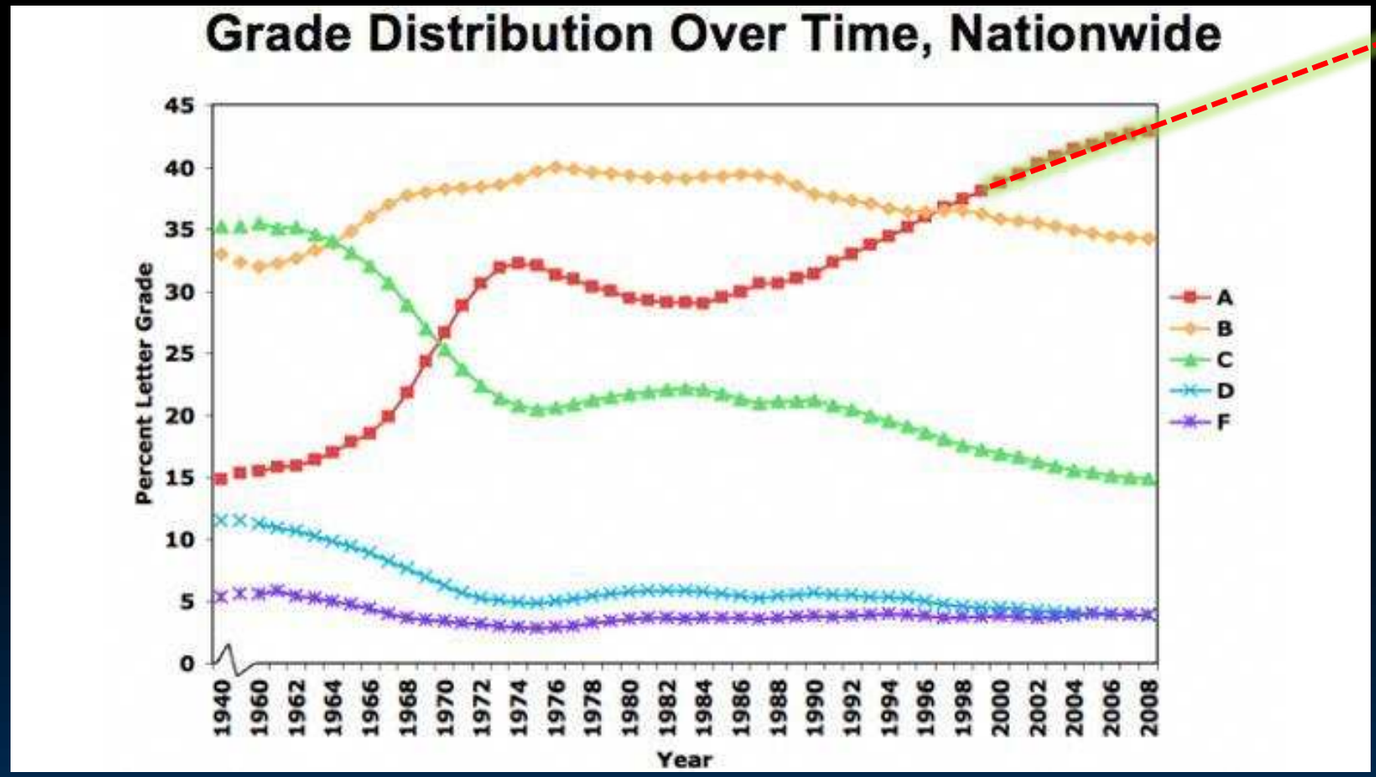
Teachers College Record Volume 114 Number 7, 2012, p. 1-23

Garcia, Nikolić

A's Law

100%

Great news:
Your grandchildren WILL get As!

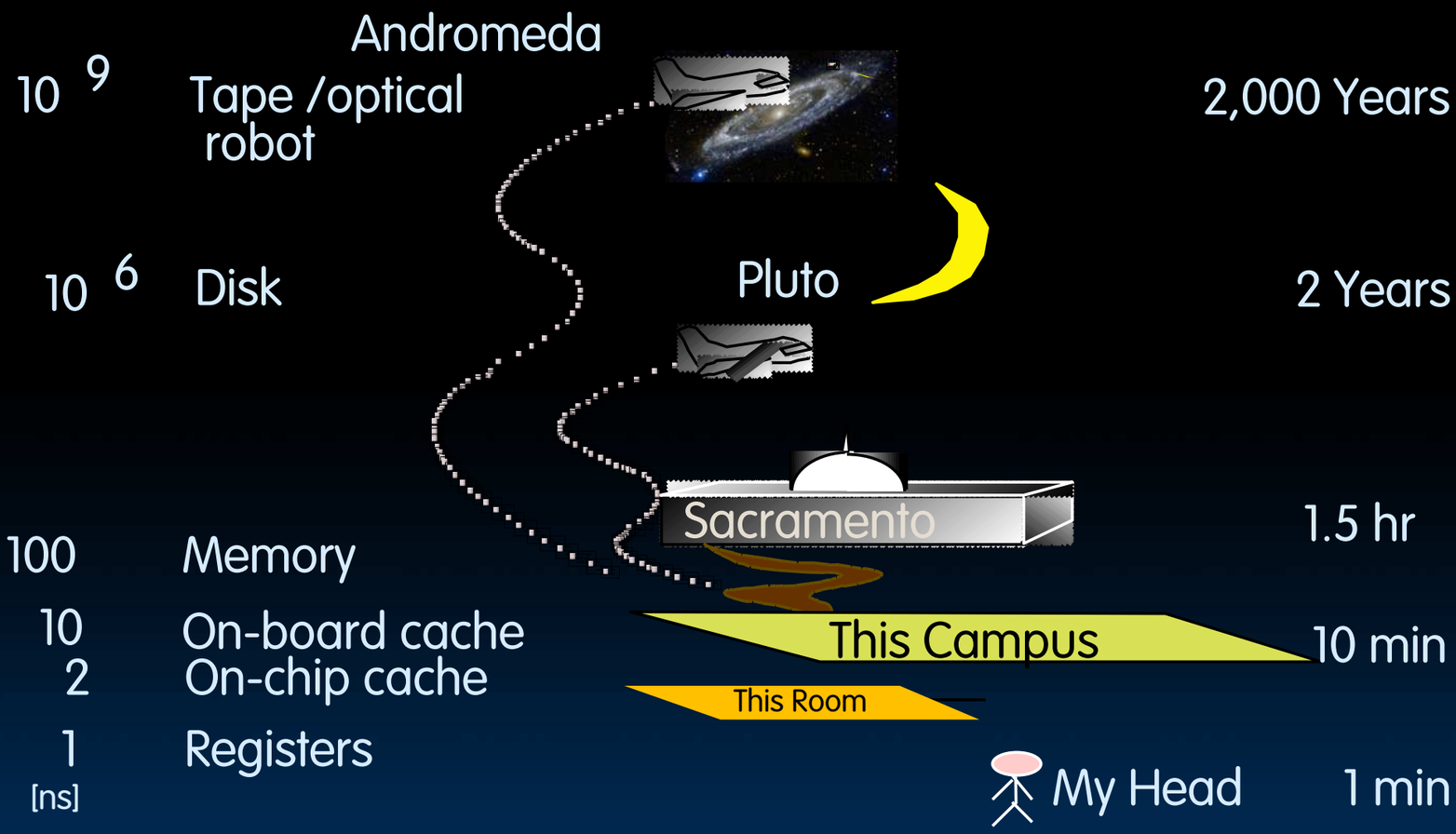


Teachers College Record Volume 114 Number 7, 2012, p. 1-23

2070

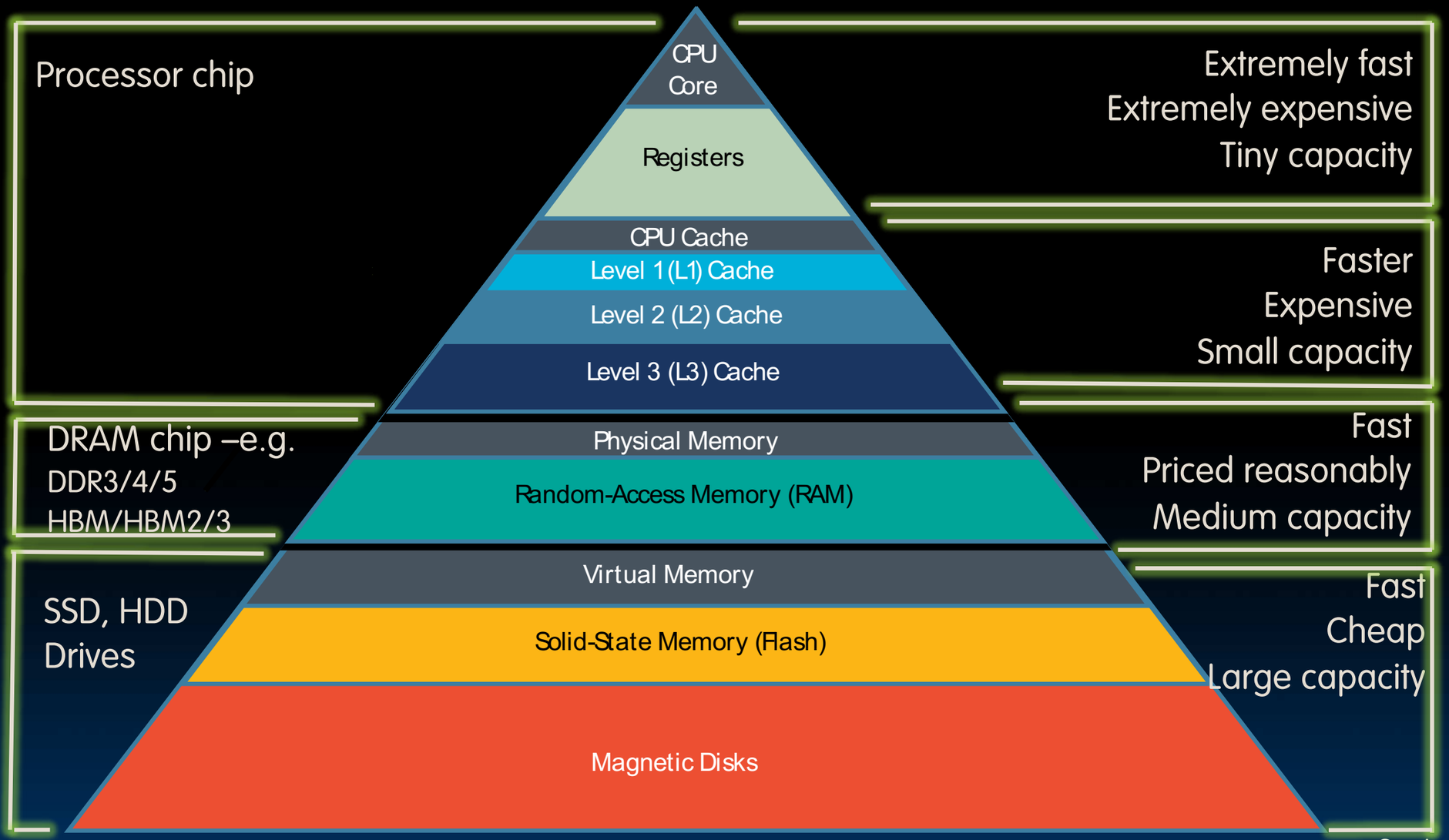
Jim Gray's Storage Latency Analogy: How Far Away is the Data?

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

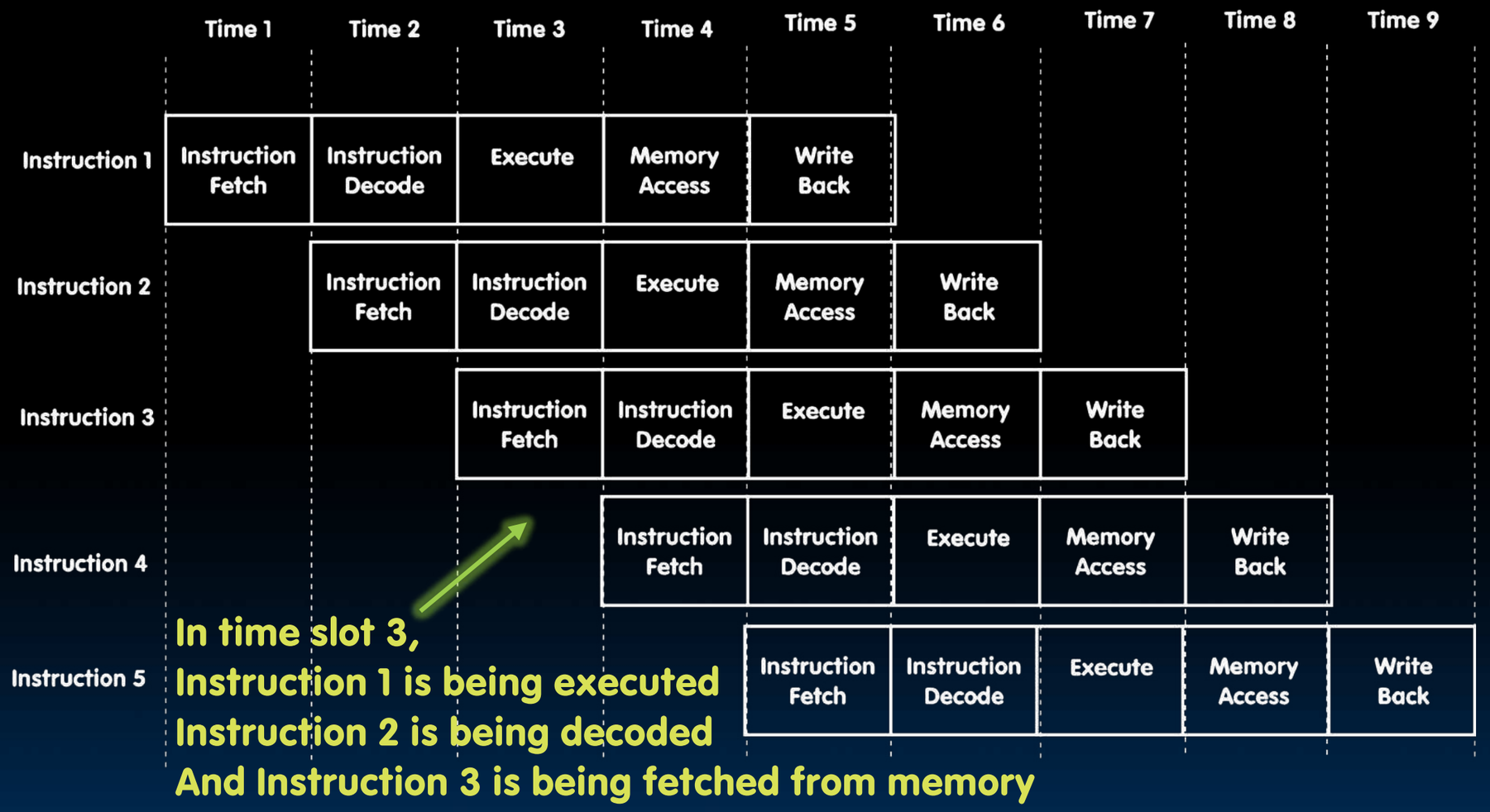


10^9	Tape /optical robot	2,000 Years	
10^6	Disk	2 Years	
100	Memory	1.5 hr	
10	On-board cache	10 min	
2	On-chip cache		
1	Registers	1 min	
[ns]			

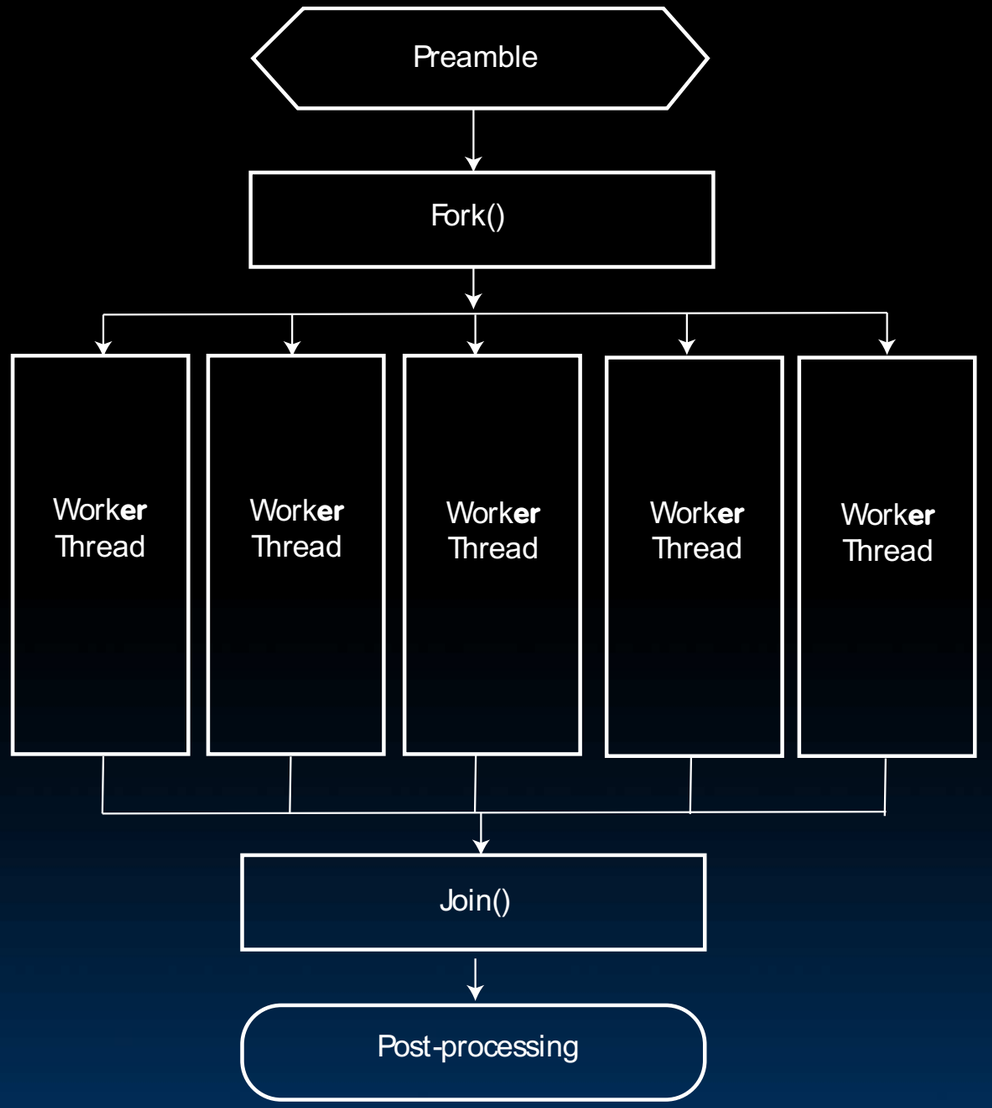
Great Idea #3: Principle of Locality / Memory Hierarchy



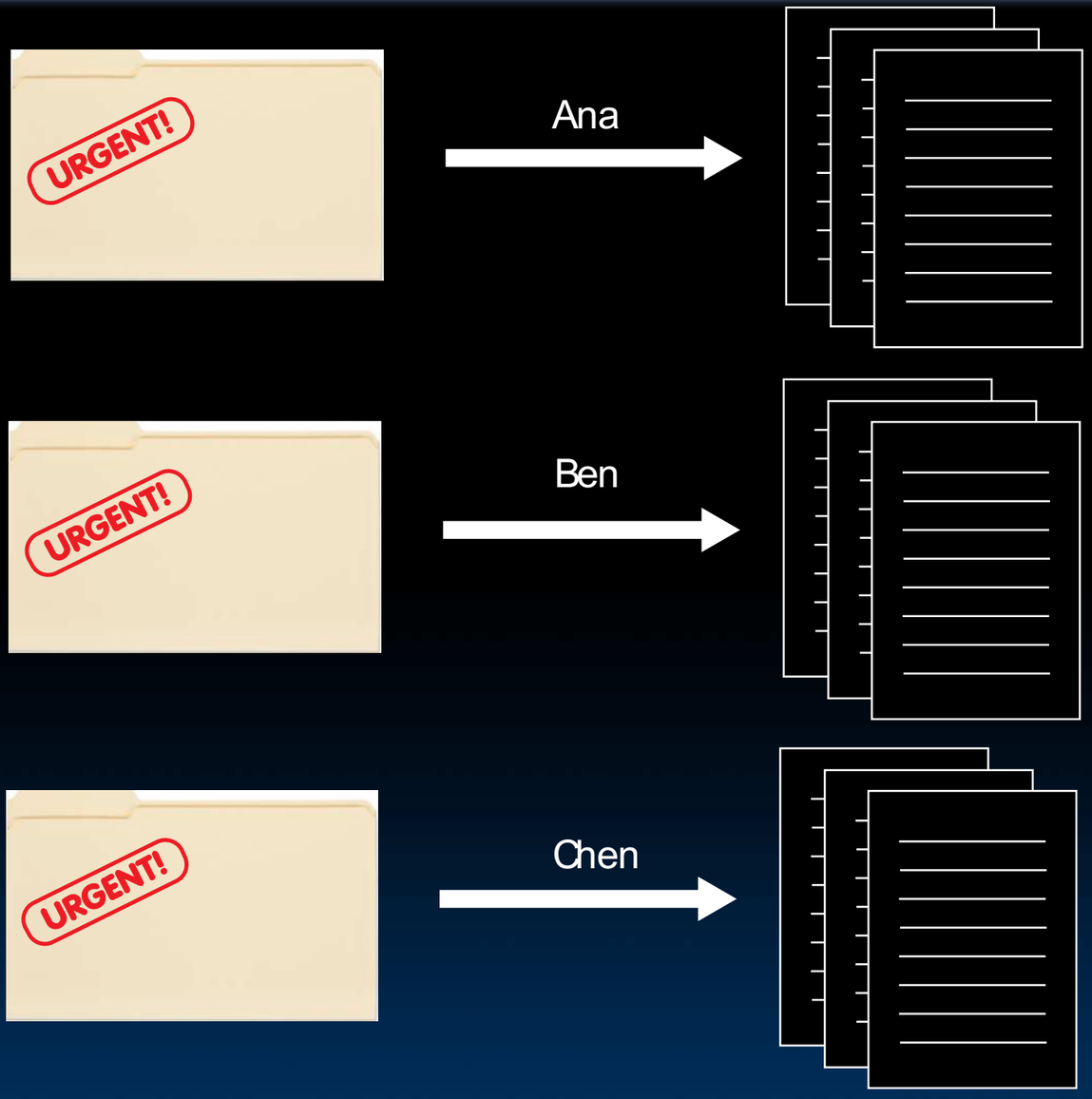
Great Idea #4: Parallelism (1/3)



Great Idea #4: Parallelism (2/3)



Great Idea #4: Parallelism (3/3)



Caveat! Amdahl's Law

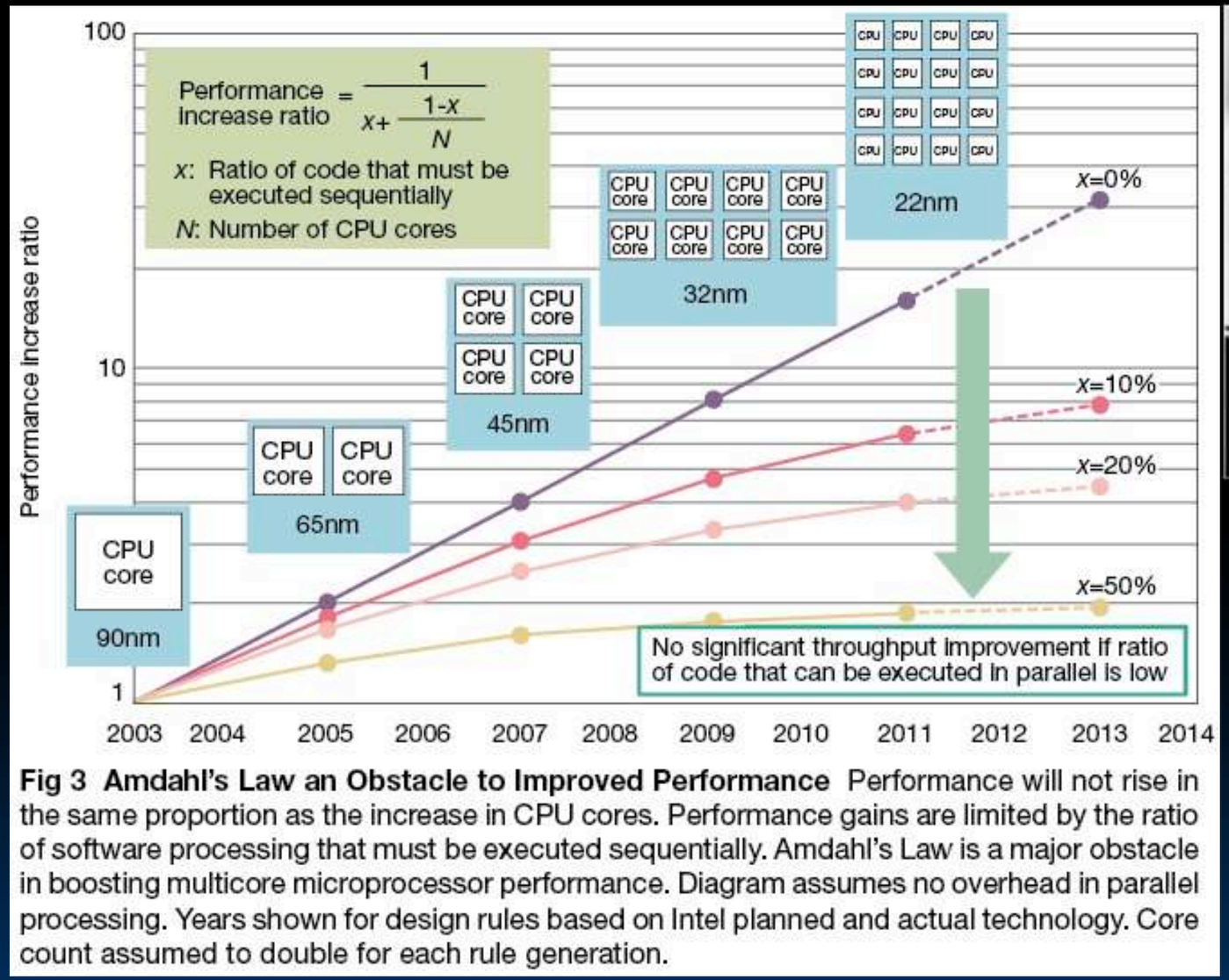


Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor 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

Great Idea #5: Performance Measurement and Improvement

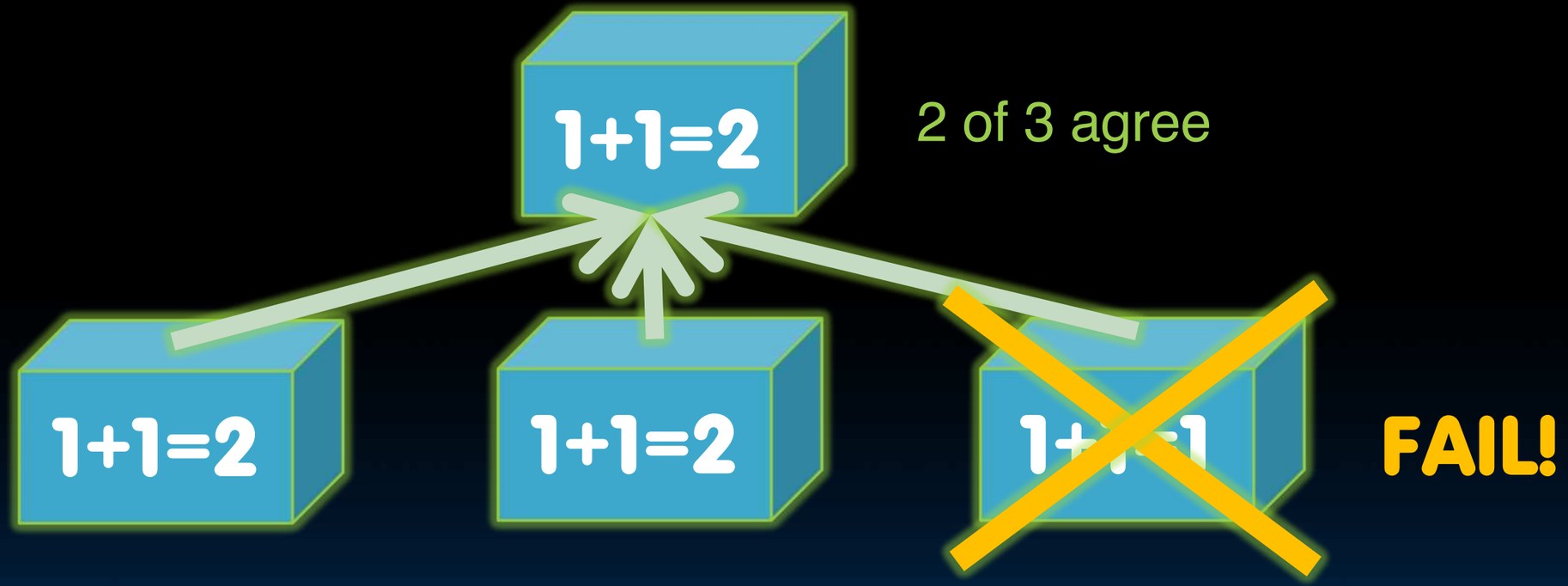
- **Matching application to underlying hardware to exploit:**
  - Locality
  - Parallelism
  - Special hardware features, like specialized instructions (e.g., matrix manipulation)
- **Latency/Throughput**
  - How long to set the problem up and complete it (or how many tasks can be completed in given time)
  - How much faster does it execute once it gets going
  - Latency is all about time to finish

Great Idea #6: Dependability via Redundancy

Image source: Hackaday.com

Great Idea #6: Dependability via Redundancy

- Redundancy so that a failing piece doesn't make the whole system fail



Increasing transistor density reduces the cost of redundancy

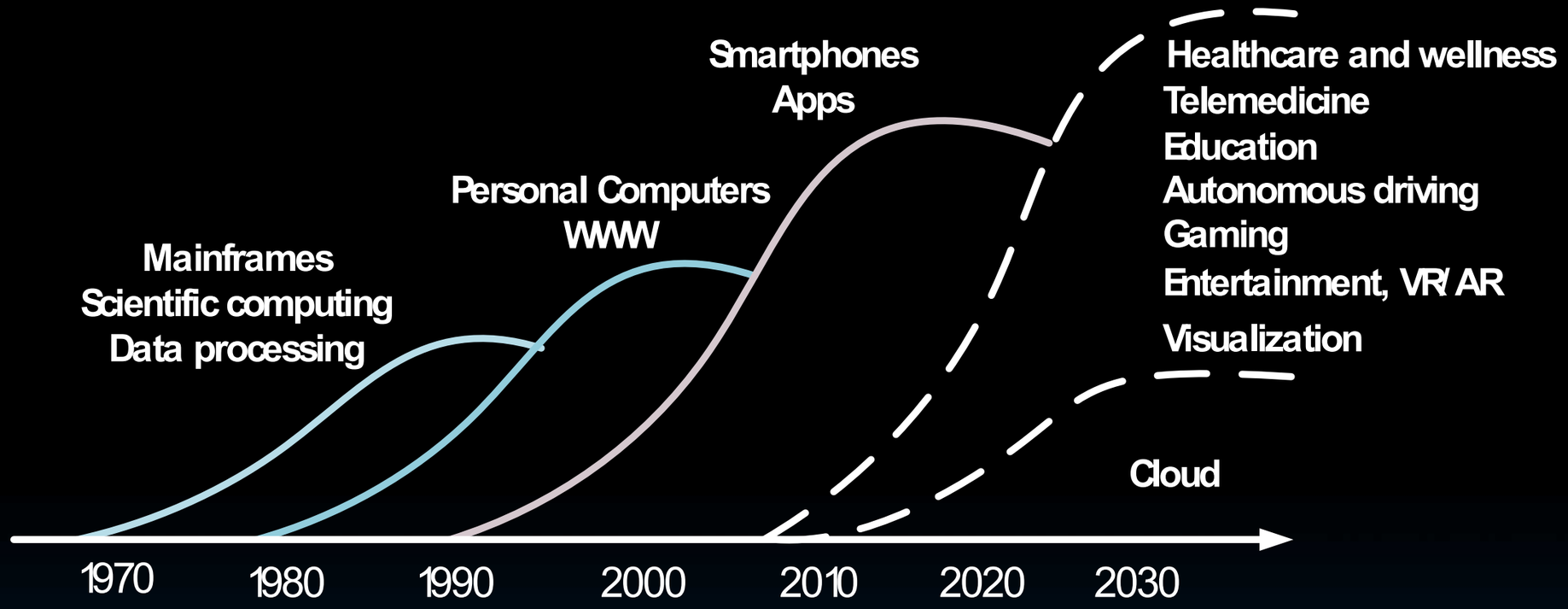
Great Idea #6: Dependability via Redundancy

- **Applies to everything from datacenters to storage to memory to instructors**
  - Redundant datacenters so that can lose 1 datacenter but Internet service stays online

  - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)

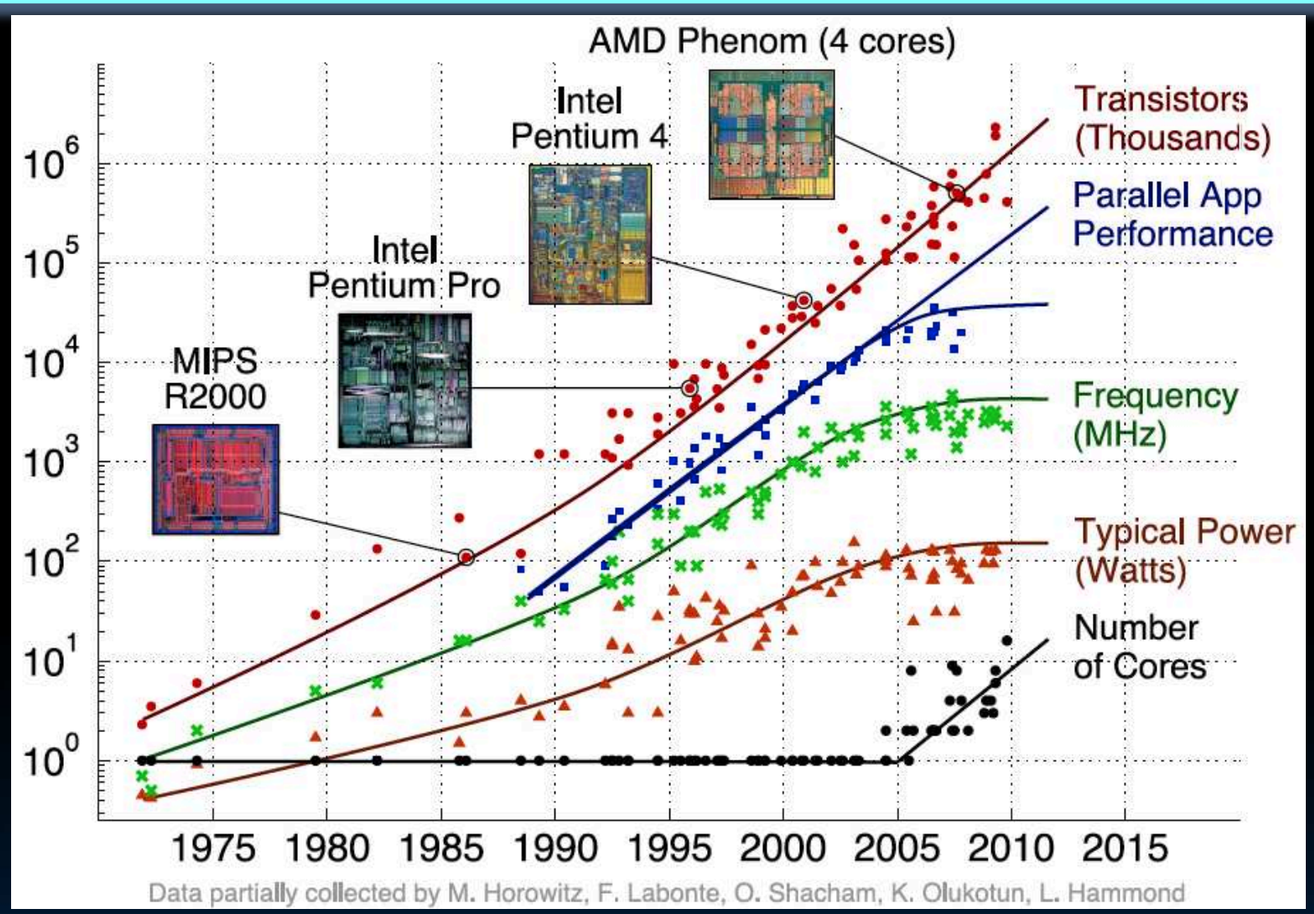
  - Redundant memory bits of so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)

Why is Computer Architecture Exciting Today?



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

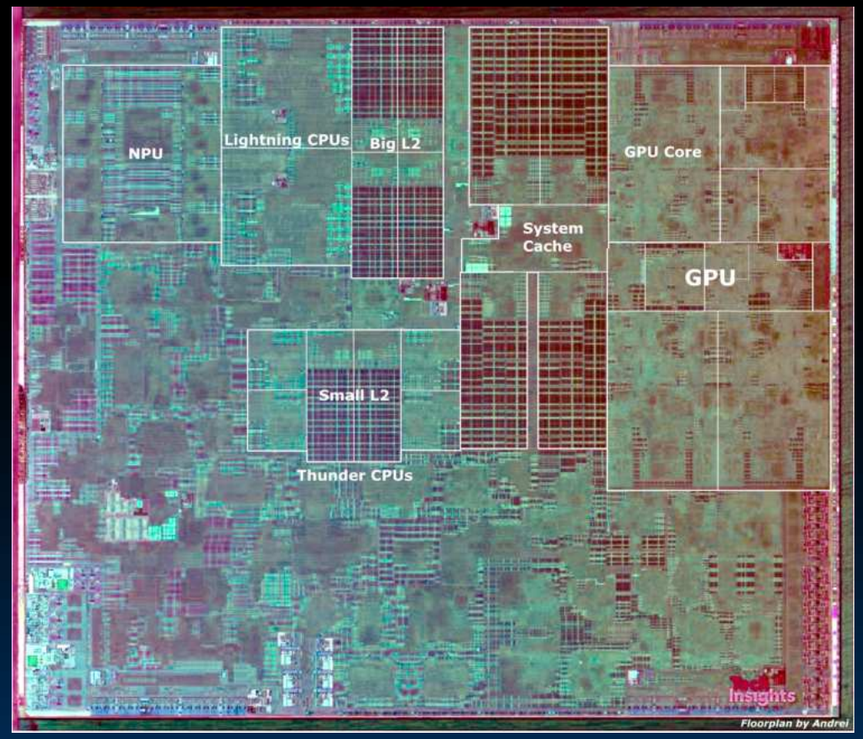
Changing Constraints



- Power limitations
- Moore's Law ending

Era of Domain-Specific Computing

- Each domain requires heterogeneous systems: Multiple processor cores, GPUs, accelerators, interfaces, memory



Apple A13 Bionic
Source: Anandtech

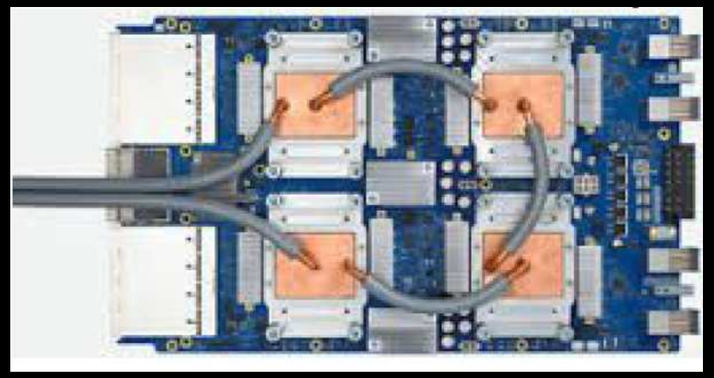
Check out Apple event tomorrow!

Old Conventional Wisdom

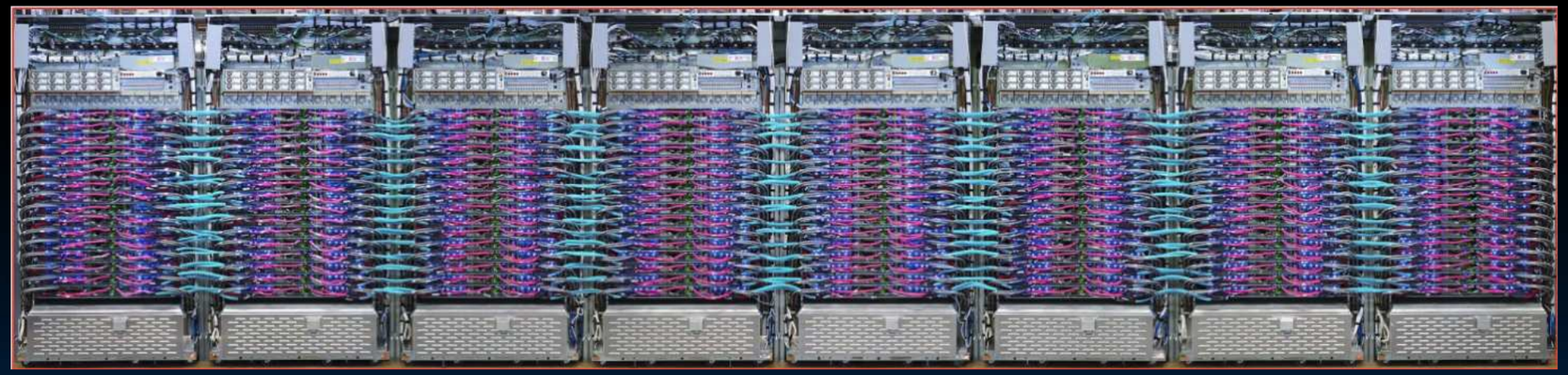
- Moore's Law + Dennard Scaling = faster, cheaper, lower-power general-purpose computers each year
- In glory days, 1%/week performance improvement!

- Dumb to compete by designing parallel or specialized computers
- By time you've finished design, next generation of general-purpose will beat you

New Conventional Wisdom



Google TPU3
Specialized Engine for NN training
Deployed in cloud



1024 chips, > 100PetaFLOPs

Patterson and Hennessy win Turing!

Agenda

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

Yoda says…

"Always in motion, the future is…"

Our schedule may change slightly depending on some factors.
This includes lectures, assignments & labs…

Course Information

- **Course Website: `cs61c.org`**
- **Instructors:**
  - Dan Garcia & Bora Nikolić
- **Teaching Assistants: (see webpage)**
- **Textbooks: Average 15 pages of reading/week**
  - Patterson & Hennessy, *Computer Organization and Design*, RISC-V ed
  - Kernighan & Ritchie, *The C Programming Language*, 2nd Edition
  - Barroso & Holzle, *The Datacenter as a Computer, 3rd*
- **Piazza:**
  - Every announcement, discussion, clarification happens there

Summary of Course Elements (1/2)

	Fall 2019 F2F	Fall 2020 Online
Lecture	MWF for an hour. If you miss it, you can watch the video. No attendance taken.	Monday live (and recorded for those who miss it), everyone strongly encouraged to attend, no attendance taken. Content (what used to be MWF lectures) has been pre-recorded with high production value. Clicker questions after every lecture, which earn credit. These are due the day after they would be normally given.
Lab	2 hours, typically done in pairs, with questions asked of an AI or TA in the room. Checkoffs at the end.	You and your partner do them together (in your own zoom rooms) or in Lab OH, and sign up for a 13m checkoff spot. This is the only required F2F interaction in class, and is a time for our staff to check-in with you about how you are doing in class, as well.
Discussion	1 hour, mostly work on worksheets in groups, some clarification mini-lectures.	Three live 1-hour discussions spread out through the day, and these videos are saved and shared. A pre-recorded "archival-quality" discussion of TAs going over the worksheet is also available. All four will cover the same material and the same worksheets. Attendance is not taken.

Summary of Course Elements (2/2)

	Fall 2019 F2F	Fall 2020 Online
Office Hours	Students drop in to 1-hour (conceptual and project) TA or faculty group office hours	TAs and faculty still hold the same office hours, just on Zoom. We will have 3 types of office hours: Lab OH for lab checkoffs and questions, Project OH for project questions, and Regular OH for anything else. Please note that a TA can help you at anything at any of the office hours though unrelated questions will not have priority. Faculty add 1-on-1 appointment slots.
Project Parties	These happen close to projects, in a big room	We are no longer having project parties and instead having project OH which are spread out closer to the due dates of the projects. They will be held via Zoom.
Exams	Quest: 1 hour (1 study sheet) Midterm: 2 hours (2 study sheets) Final: 3 hours (3 study sheets) All exams occur in the evening spread all across campus.	Quest: 1 hour worth of work Midterm: 2 hours worth of work Final: 3 hours worth of work All exams occur over a 24-hour period. Open book, closed internet. Everyone given a randomized exam and 24 total hours to complete it.

Course Grading

- **Lecture (5%)**
- **Labs (5%)**
- **Homework (10%)**
- **Projects (40%)**
  1. Non-Parallel Application (C) - Individual
  2. RISC-V Application - Invidual
  3. Computer Processor Design (Logisim) – Partners
  4. Parallization C Program – Partners
- **Quest (5%): 24-Hour, Sunday before drop, can be clobbered**
- **Midterm (15%): 24-Hour Sunday, can be clobbered!**
- **Final (20%): 24-Hour Sunday before exams**
- **EPA: Effort, Participation and Altruism**
  - (sprinkle points added on top)
- **Performance Competition for honor (and EPA)**

EECS Grading Policy

- **Absolute Grading! (not curved!)**
  - http://www.eecs.berkeley.edu/Policies/ugrad.grading.shtml
  - "A typical GPA for courses in the lower division is 2.8-3.3. This GPA would result, for example, from 35% A's, 45% B's, 13% C's, 7% D's,F's.
- **Job/Intern Interviews: They grill you with technical questions, so it's what you say, not your GPA**
  - CS61C gives good stuff to say

Our goal as instructors

- **To make your experience in CS61C as enjoyable & informative as possible**
  - Humor, enthusiasm, guests, technology-in-the-news in lecture
  - Fun, challenging projects & HW
  - Pro-student policies (exam clobbering)
- **To maintain Cal standards of excellence**
  - Projects & exams as rigorous as every year.
- **To be HKN "7.0" instructors**
  - Please give feedback so we can improve! Why are we not 7.0 for you? We will listen!!

Extra Credit: EPA!

- **Effort**
  - Attending Prof and TA office hours, completing all assignments
  - turning in HW0
- **Participation**
  - Asking great Qs in zoom & making it interactive
  - Doing the performance competition
- **Altruism**
  - Helping others in lab or on Piazza
  - Writing software, creating art, tutorials that help others learn
- **EPA! can bump students up to the next grade level**
  - (but EPA! #s are internal)

Late Policy … Slip Days!

- **Assignments due at 11:59:59 PM**
- **You have 3 slip day tokens (NOT hour or min)**
- **Every day your project or homework is late (even by a minute) we deduct a token**
- **After you've used up all tokens, it's 33% off per day.**
  - No credit if more than 3 days late
  - Save your tokens for projects, worth more!!
- **No need for sob stories, just use a slip day!**

Policy on Assignments and Independent Work

- ALL PROJECTS (but the first) WILL BE DONE WITH A PARTNER
- With the exception of laboratories and assignments (projects and HW) that explicitly permit you to work in groups, all homework and projects are to be YOUR work and your work ALONE.
- PARTNER TEAMS MAY NOT WORK WITH OTHER PARTNER TEAMS
- 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, since sometimes the point of the assignment WAS the "algorithm" and if you share that, they won't learn what we want them to learn). HKN and tutoring sessions that work you through the pseudocode are not allowed. The pseudocode is sometimes the entire point! Feel free to answer questions on Piazza that help them debug (don't share code, even snippets there). We expect that what you hand in is yours.
- It is NOT acceptable to copy solutions from other students.
- It is NOT acceptable to copy (or start your) solutions from the Web.
- It is NOT acceptable to leave your code anywhere where an unscrupulous student could find and steal it (e.g., public GITHUBs, walking away while leaving yourself logged on, leaving printouts lying around,etc)
- We have tools and methods, 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 the minimum F in the course, and a letter in your Cal record documenting the incidence of cheating.
- (We've caught people in recent semesters!) – ~50-100 students are caught every semester!
- Both Giver and Receiver are equally culpable and suffer equal penalties

Summary

- **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