



UC Berkeley
Teaching
Professor
Dan Garcia

CS61C

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



UC Berkeley
Lecturer
Justin Yokota

Welcome, everyone!! Course Introduction

A Janet Jackson Song Could Crash Windows XP Laptops!

...a sound frequency in Janet Jackson's song "Rhythm Nation" could crash a model 5400rpm laptop hard drive used in certain Windows XP notebooks...

"Playing the music video on one laptop caused a laptop sitting nearby to crash, even though that other laptop wasn't playing the video!" Microsoft determined the song had a frequency that matched the laptop hard drive's natural resonant frequency, which caused its moving disks to over-vibrate and

cs61c.org induce a crash..



Garcia, Yokota



Thinking about Machine Structures

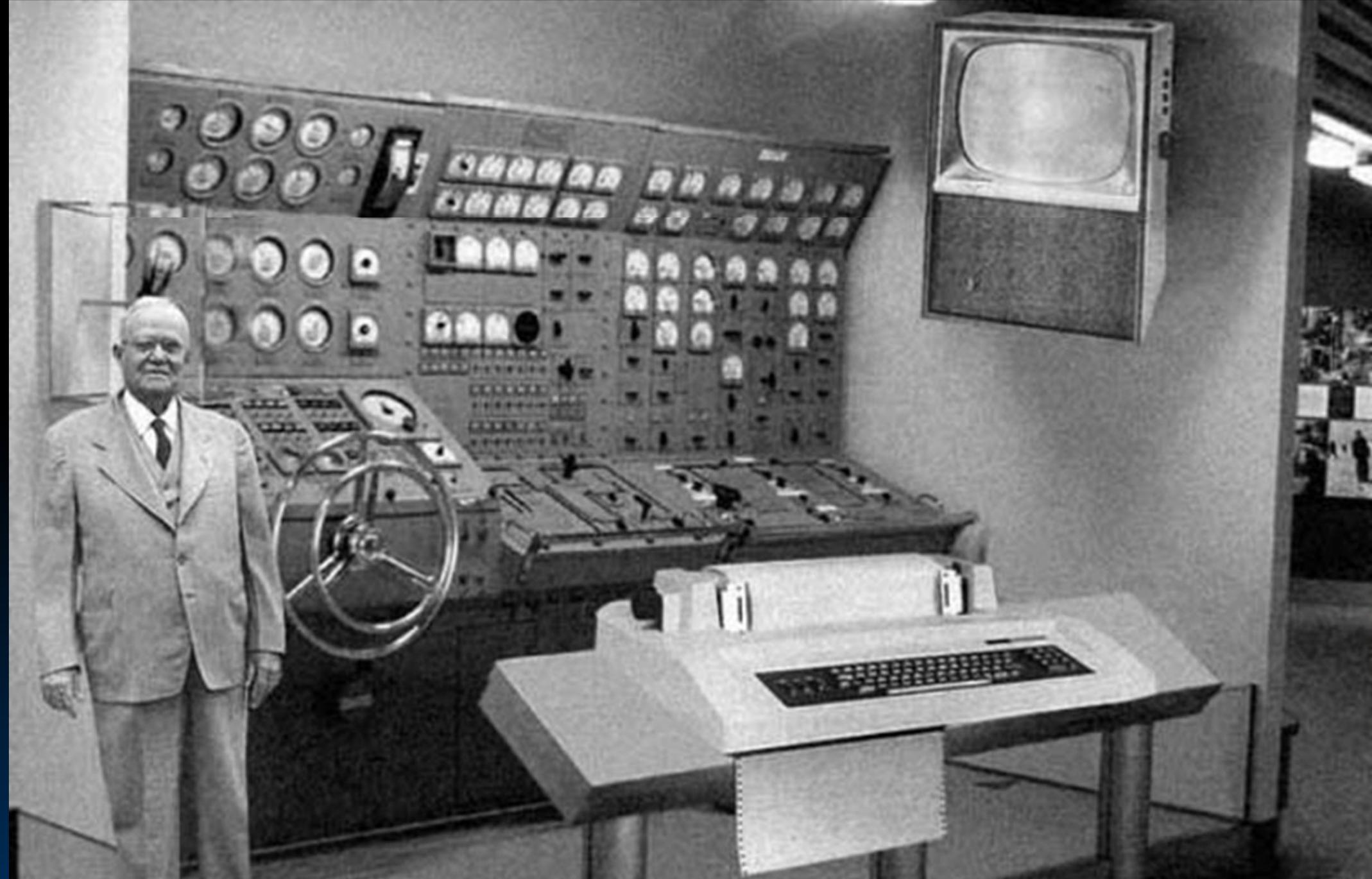
- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- What you need to know about this class

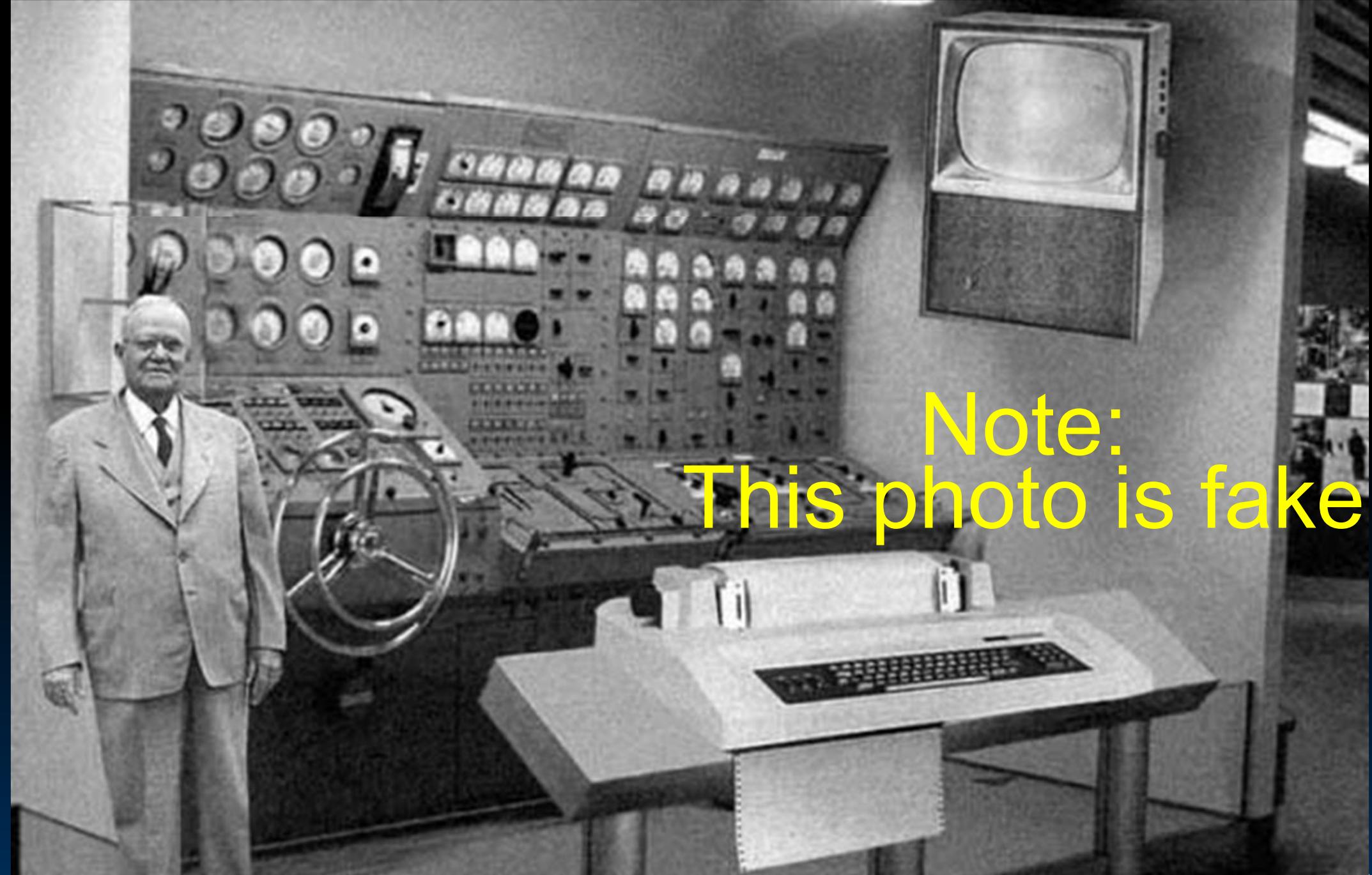


CS61C is NOT about C Programming

- It is about the hardware-software interface
 - What 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





New School CS61C (1/3)



Personal
Mobile
Devices

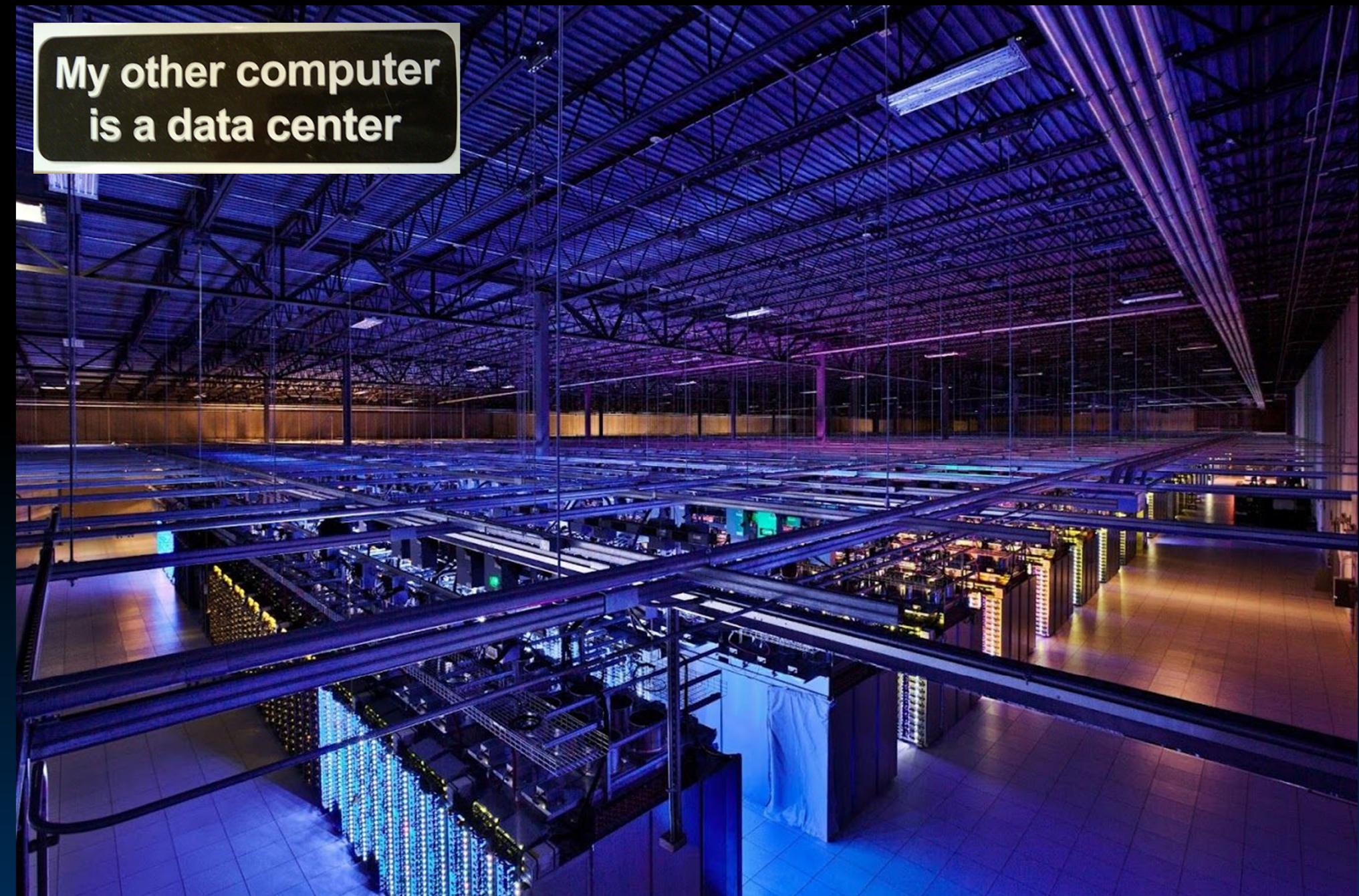
Network
Edge Devices

New School CS61C (2/3)

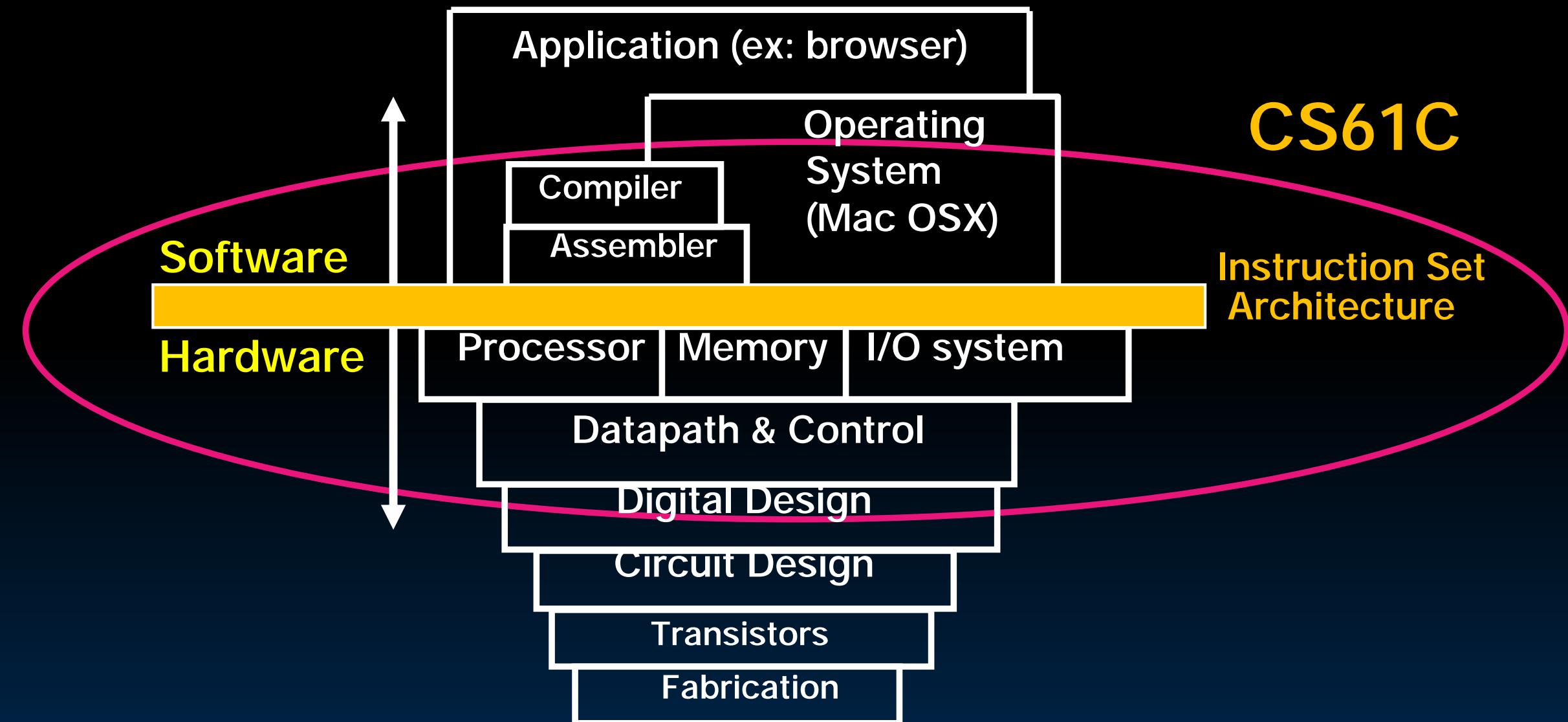


Garcia, Yokota

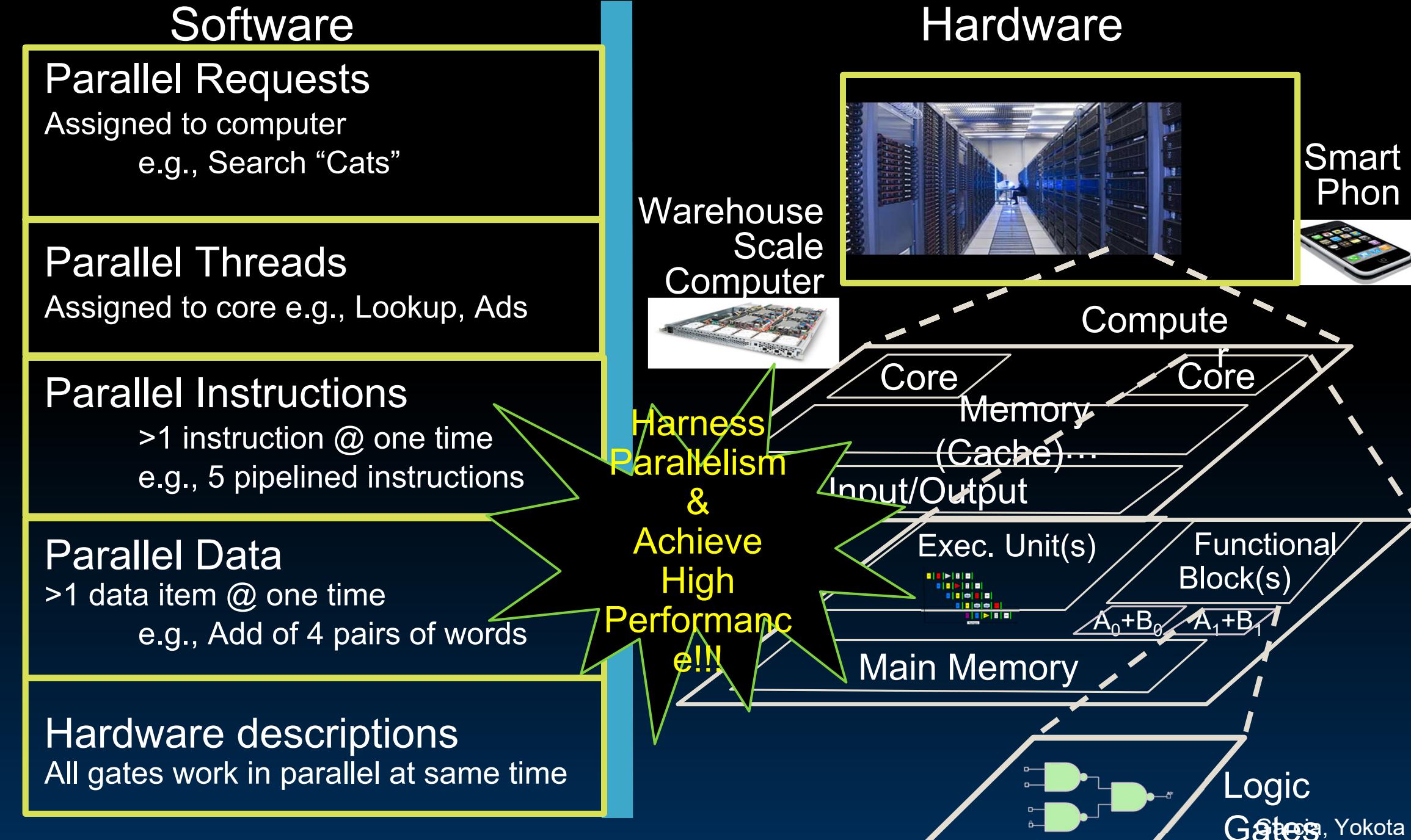
New School CS61C (3/3)



Old School Machine Structures



New-School Machine Structures (It's a bit more complicated!)



Great Ideas in Computer Architecture

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



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

# anything can be a Python name!
```

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



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

$\text{temp} = \text{v}[k];$
 $\text{v}[k] = \text{v}[k+1];$
 $\text{v}[k+1] = \text{temp};$

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

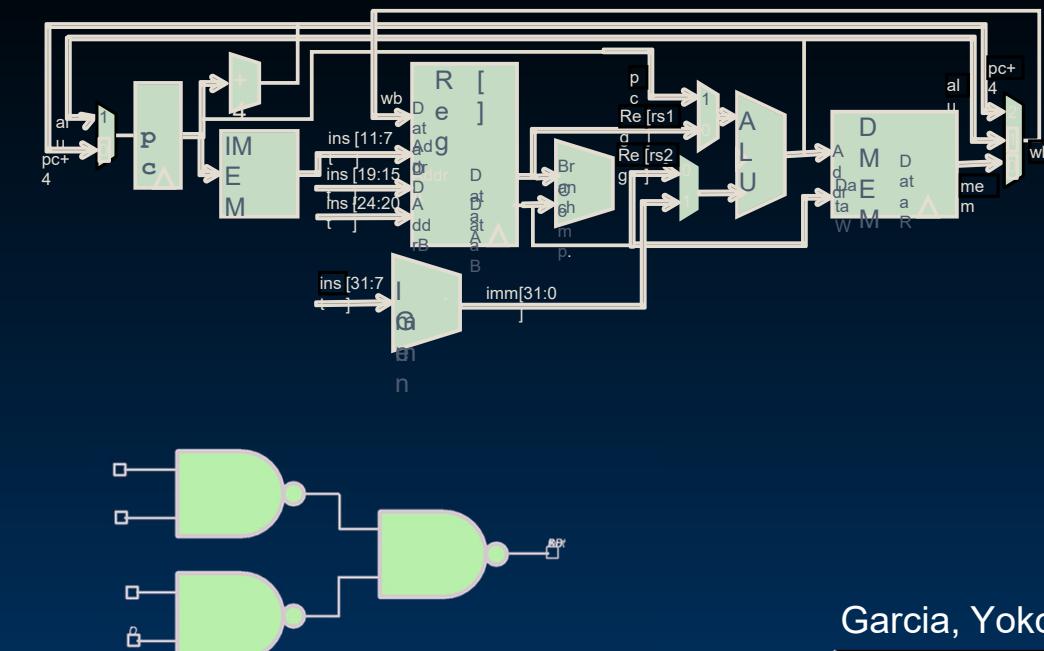
lw x3, 0(x10)
lw x4, 4(x10)
sw x4, 0(x10)
sw x3, 4(x10)

Assembler
Machine Language
Program (RISC-V)

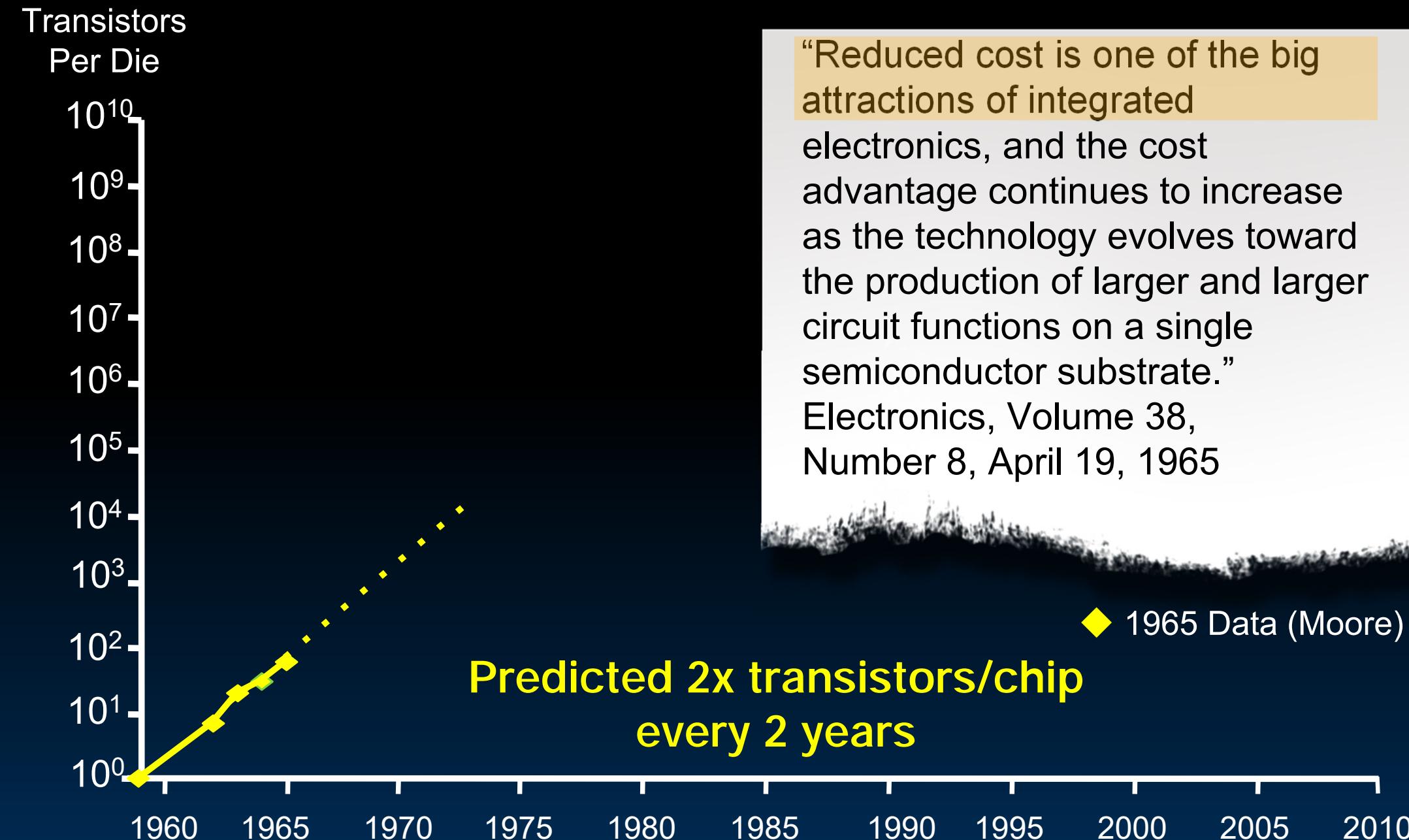
1000 1101 1110 0010 0000 0000 0000 0000
1000 1110 0001 0000 0000 0000 0000 0100
1010 1110 0001 0010 0000 0000 0000 0000
1010 1101 1110 0010 0000 0000 0000 0100

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

Hardware Architecture
Description
(e.g., block Architecture)
Logic Circuit Implementation
(Circuit Schematic Diagrams)



Great Idea #2: Moore's Law - 2005

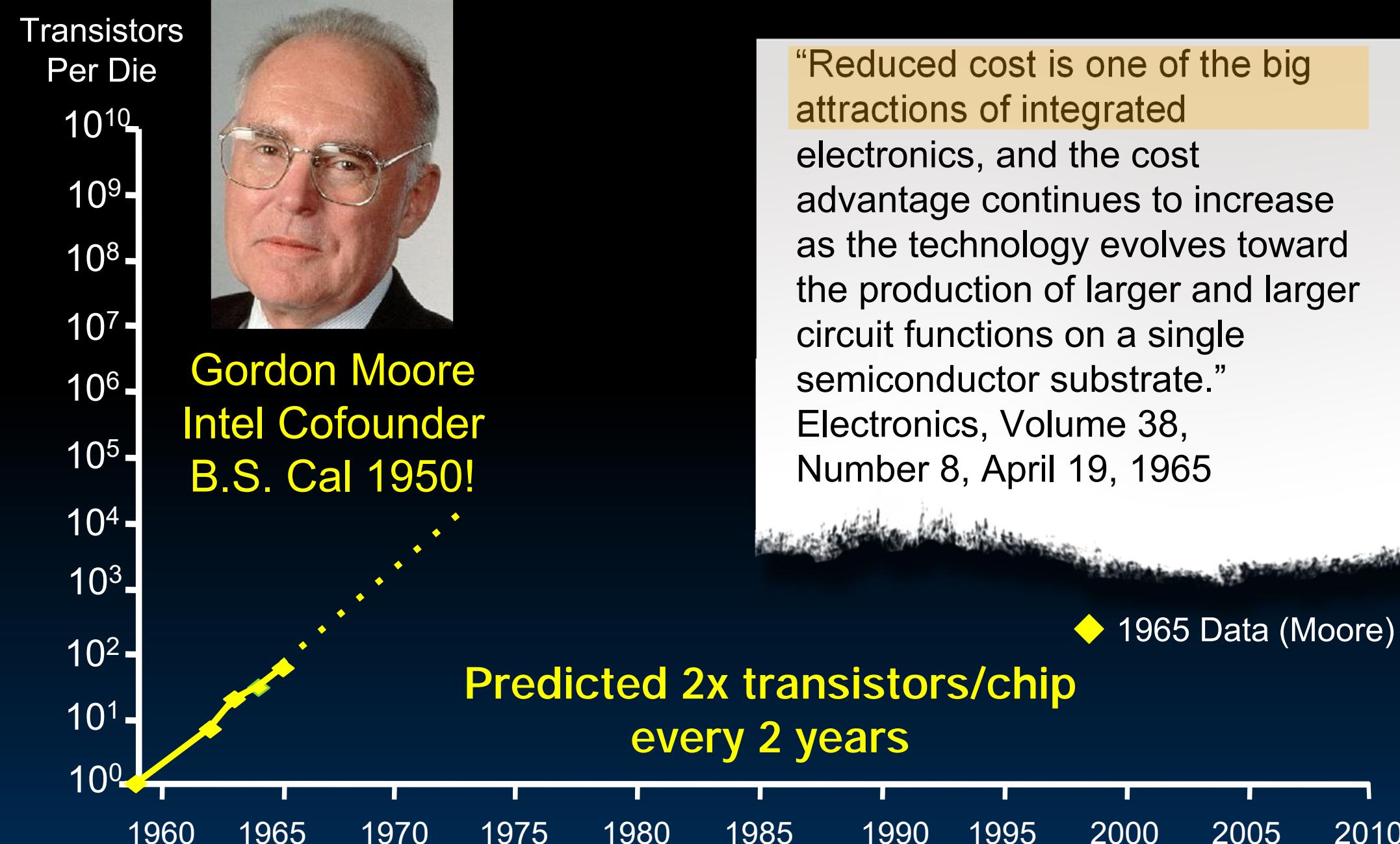


Source: Intel

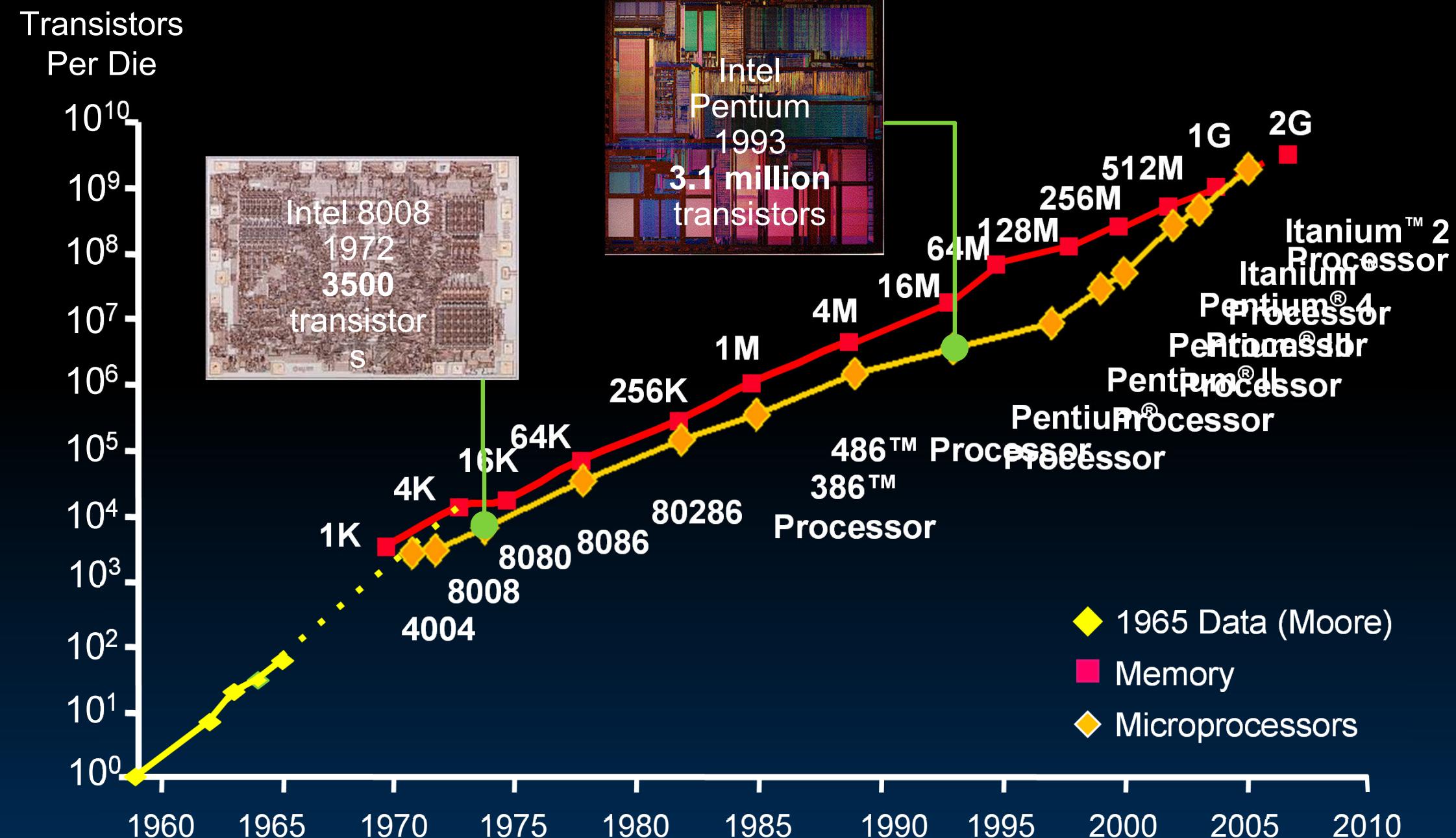
Garcia, Yokota

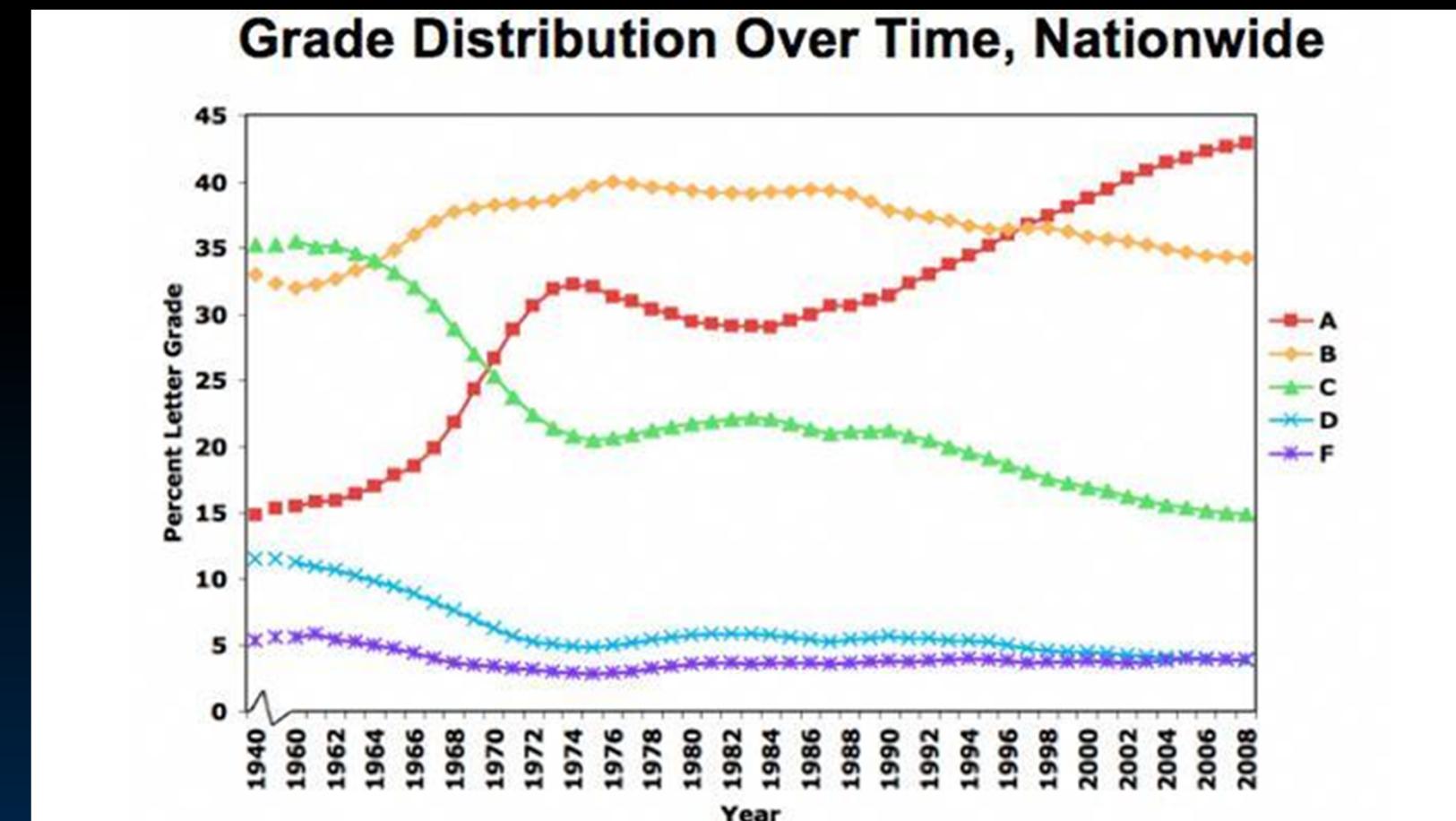


Great Idea #2: Moore's Law - 2005



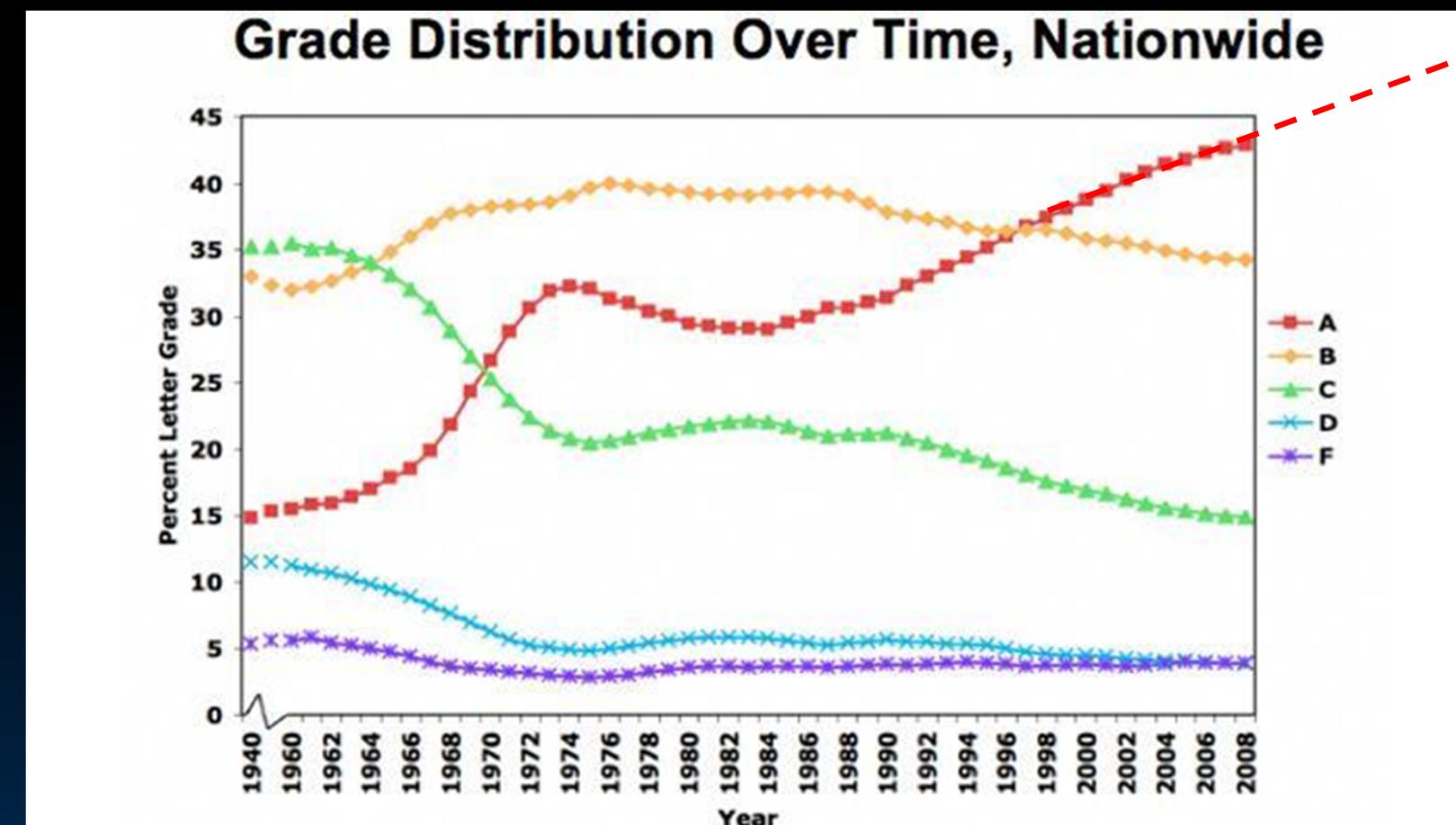
Great Idea #2: Moore's Law - 2005





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

Great news:
Your grandchildren WILL all get As!



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

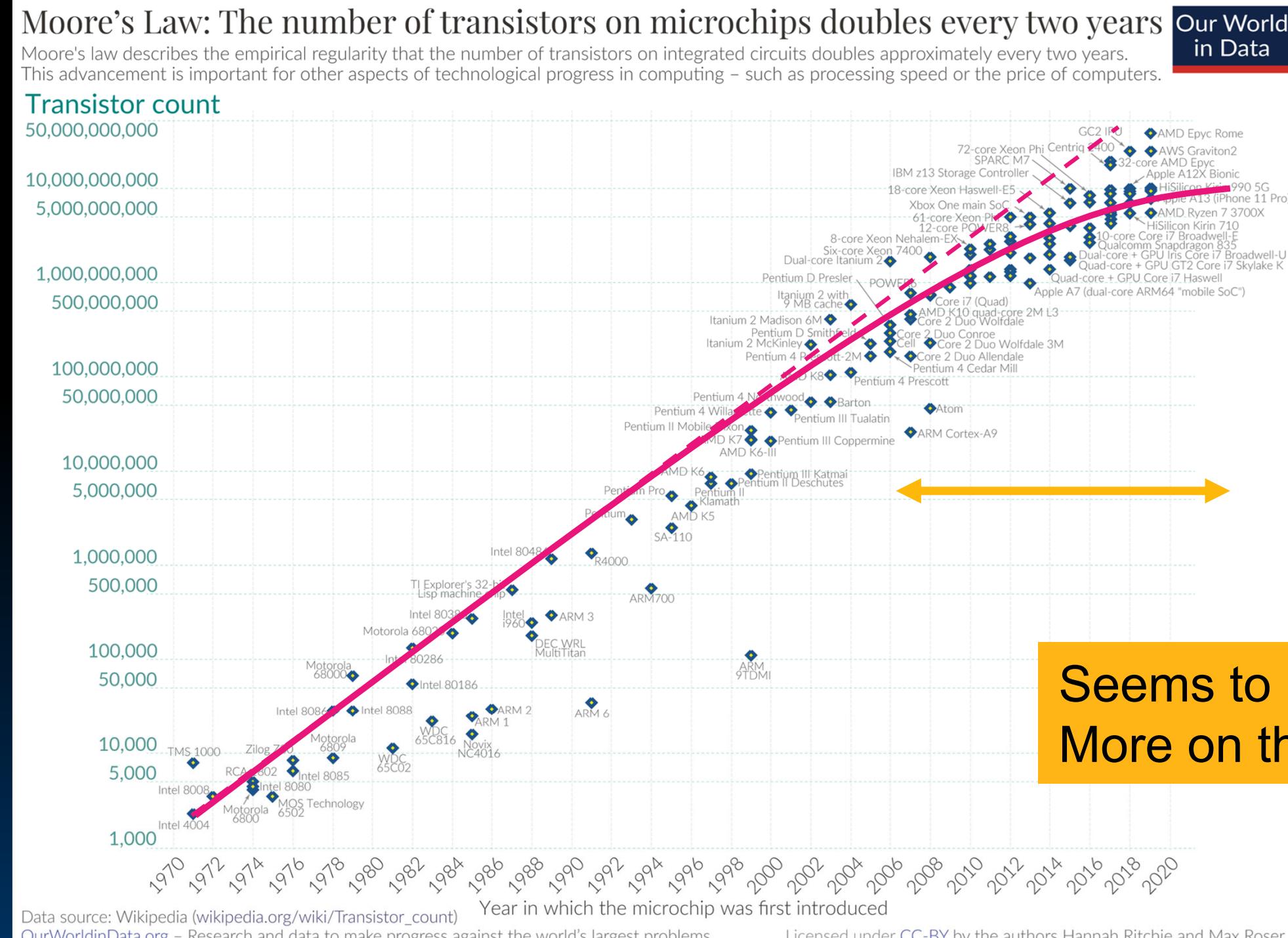
100

%

2070

...

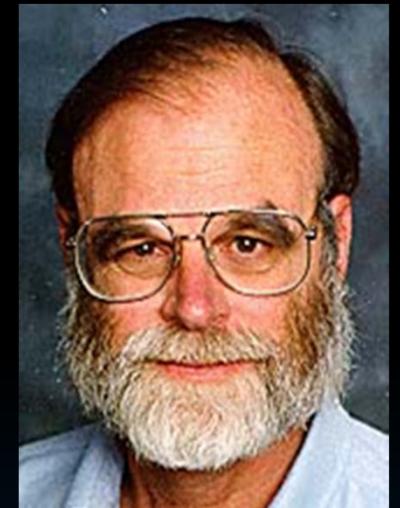
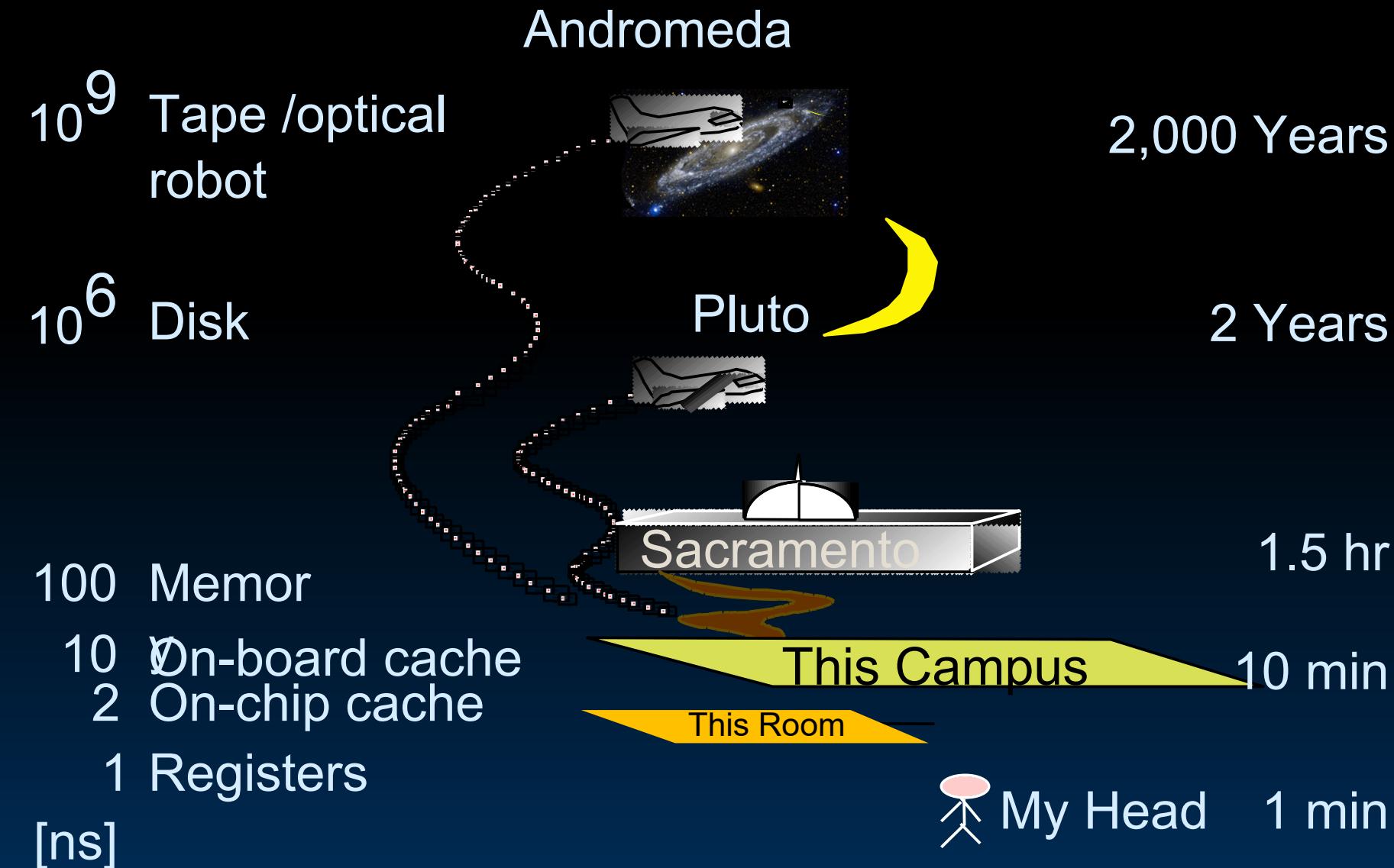
Moore's Law...?



Great Idea #3: Principle of Locality / Memory

Hierarchy

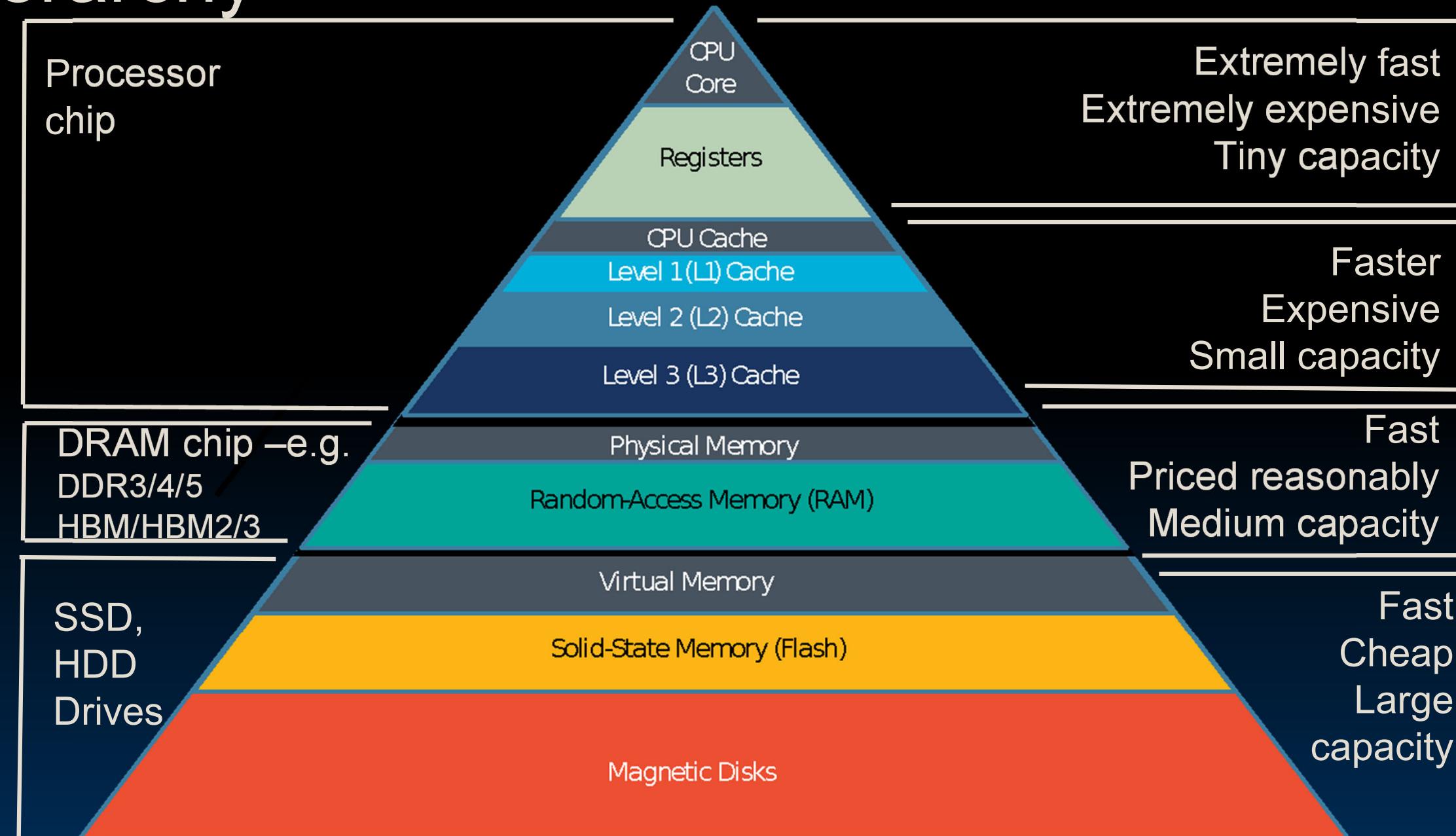
Storage Latency Analogy: How Far Away is the Data?



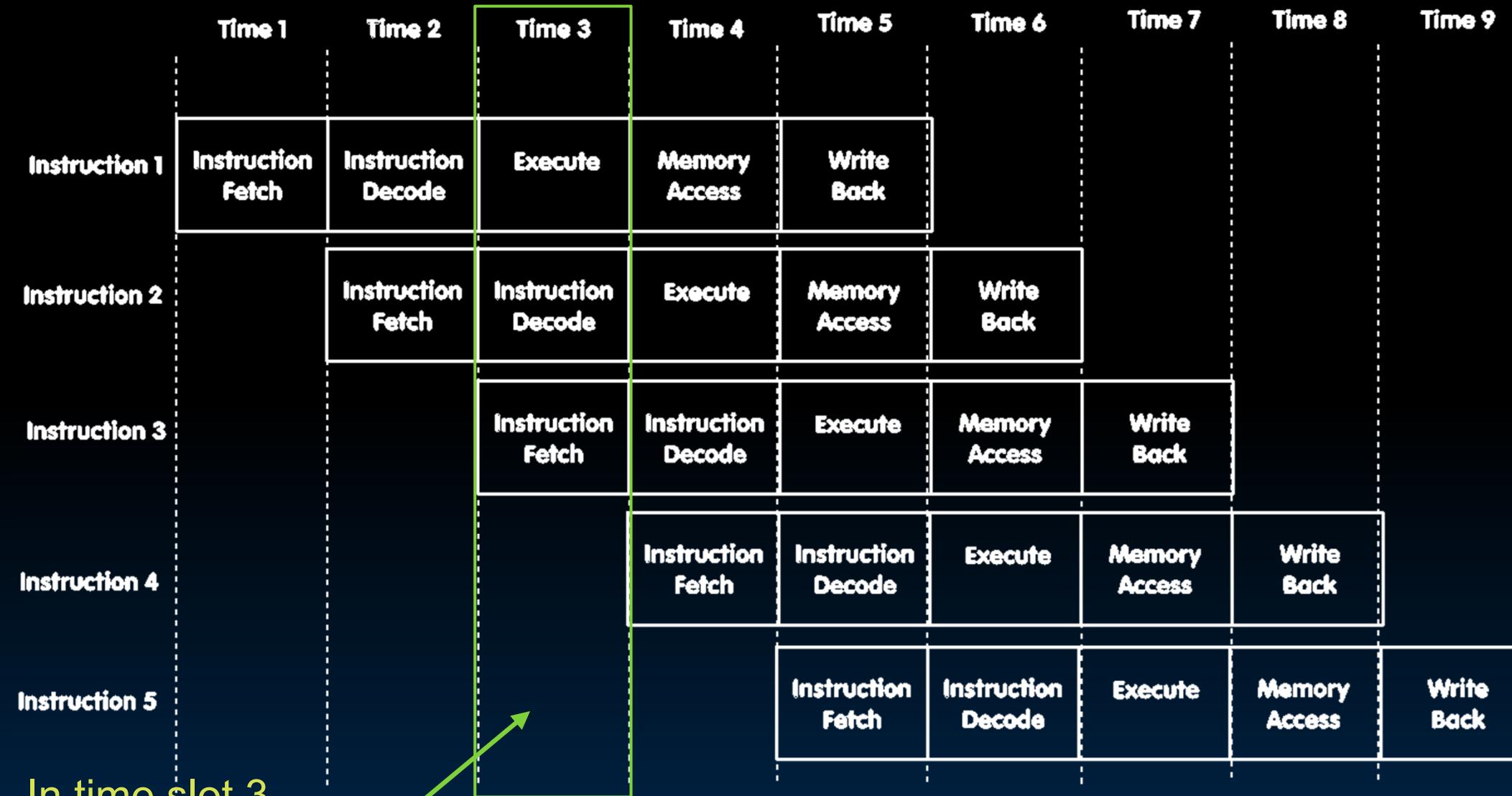
Jim Gray
Turing Award
B.S. Cal
1966
Ph.D. Cal
1969
Garcia, Yokota



Great Idea #3: Principle of Locality / Memory Hierarchy



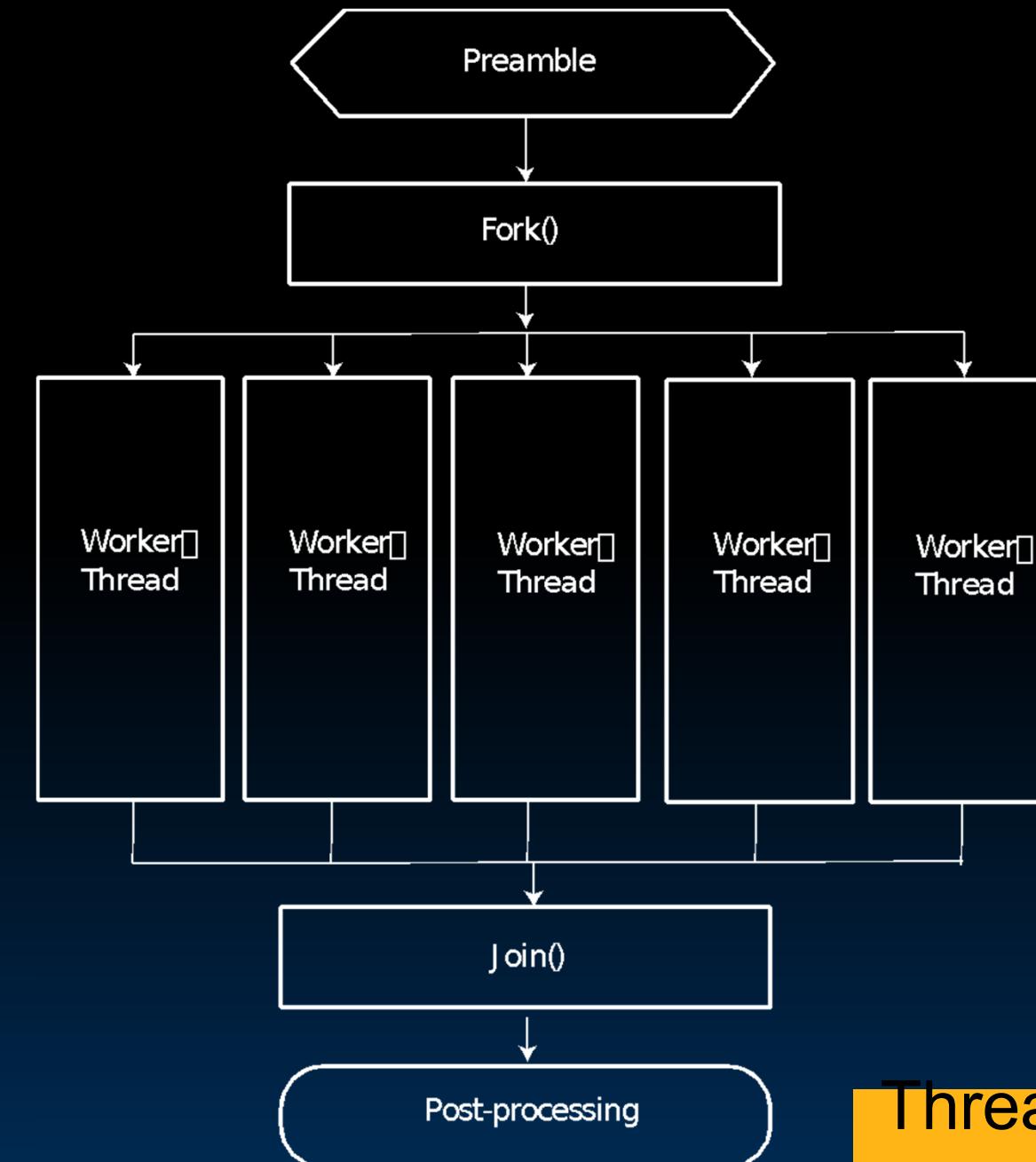
Great Idea #4: Parallelism (1/3)



In time slot 3,
Instruction 1 is being executed
Instruction 2 is being decoded
And Instruction 3 is being fetched from memory

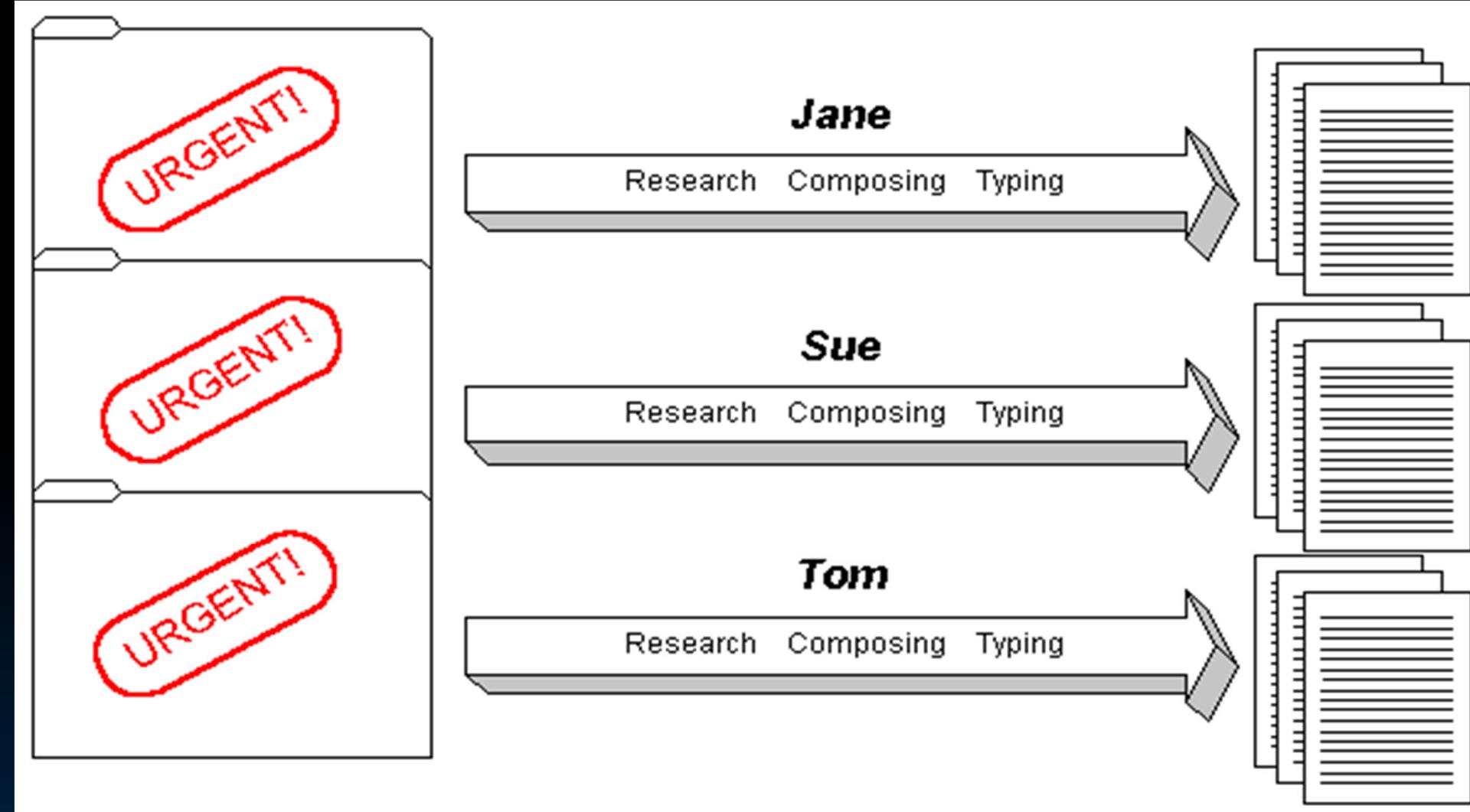
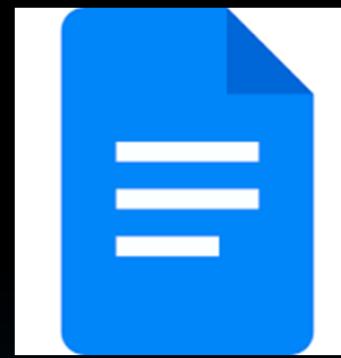
Instruction-Level
Parallelism

Great Idea #4: Parallelism (2/3)



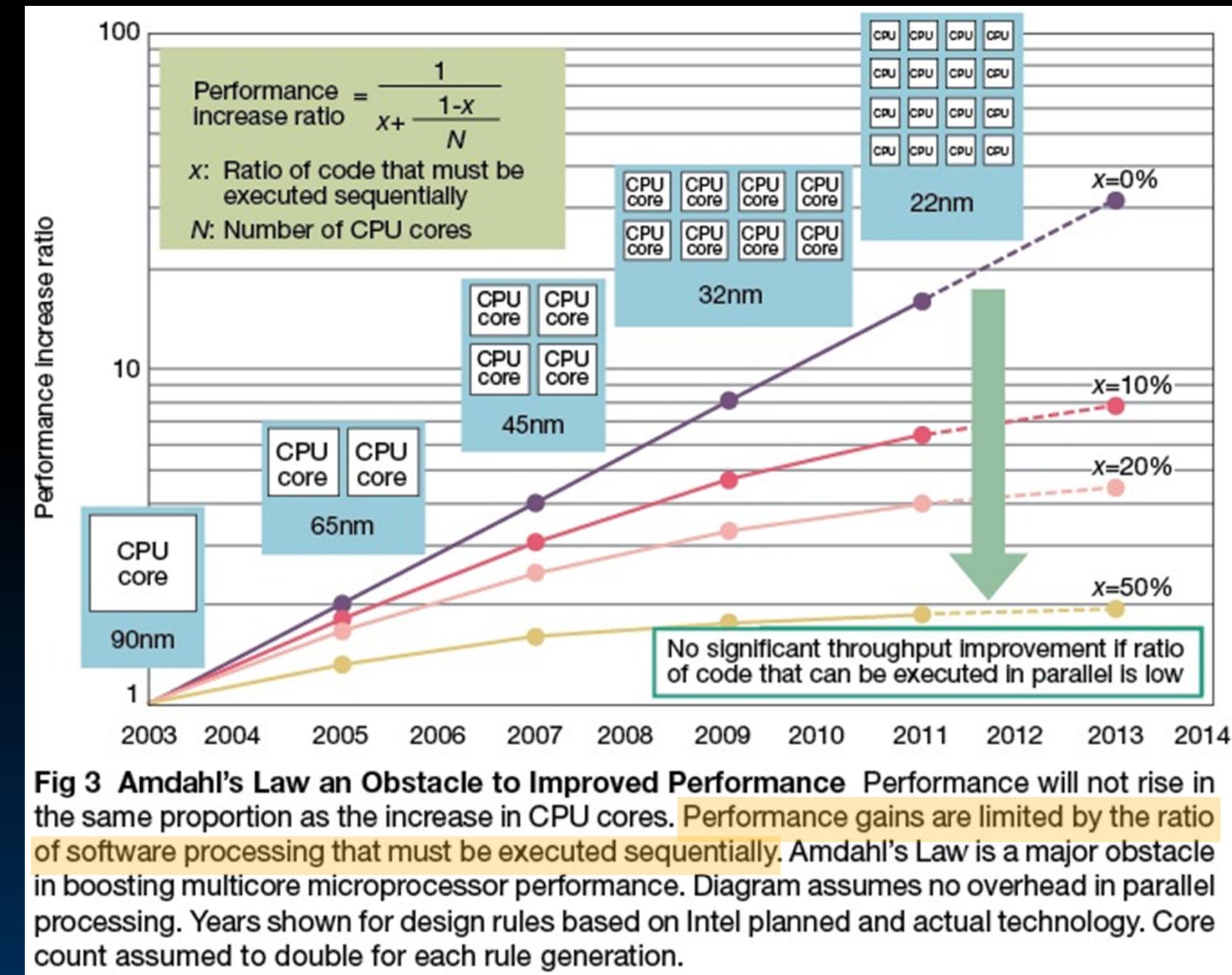
Thread-Level
Parallelism

Great Idea #4: Parallelism (3/3)



Data-Level Parallelism

Caveat! Amdahl's Law



Gene Amdahl
Computer Pioneer

- Match 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

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

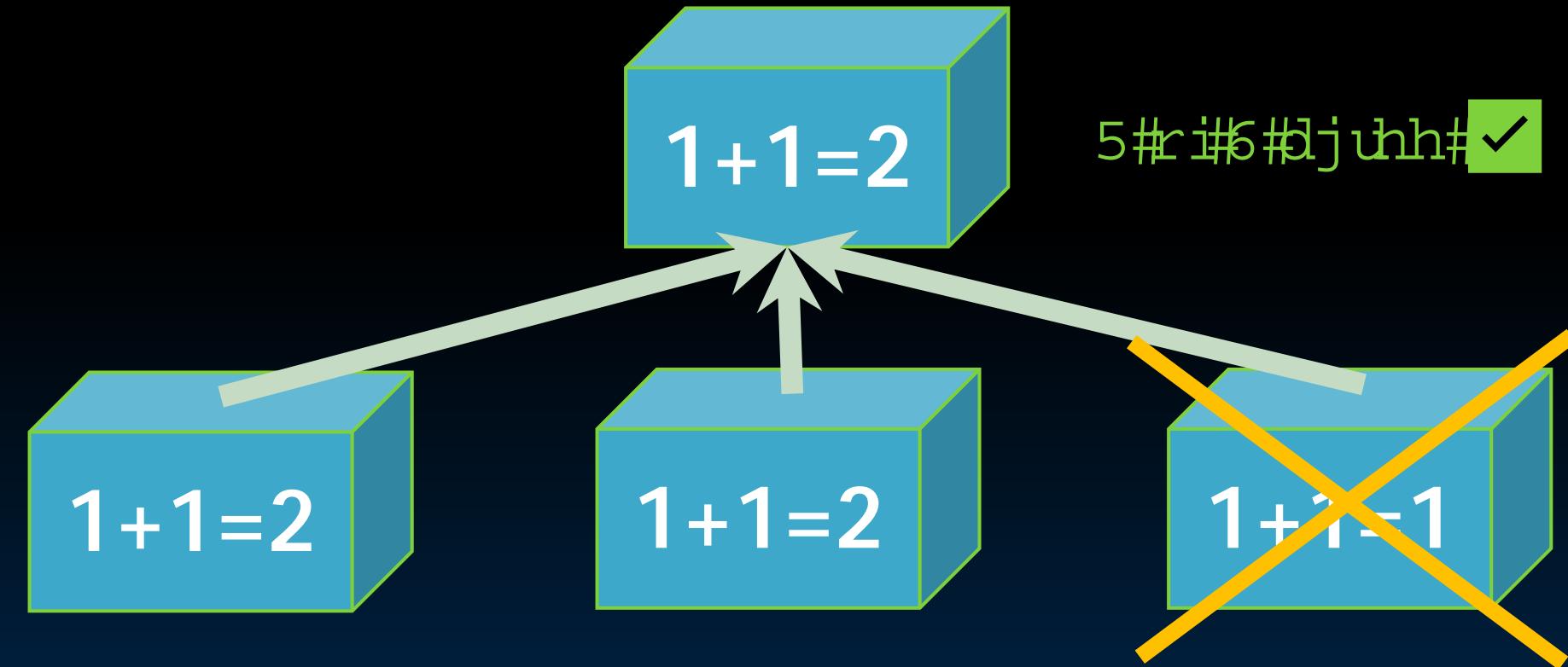


<https://www.exploratorium.edu/exhibits/cloud-chamber>

Garcia, Yokota

Great Idea #6: Dependability via Redundancy

- Design with 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 so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)

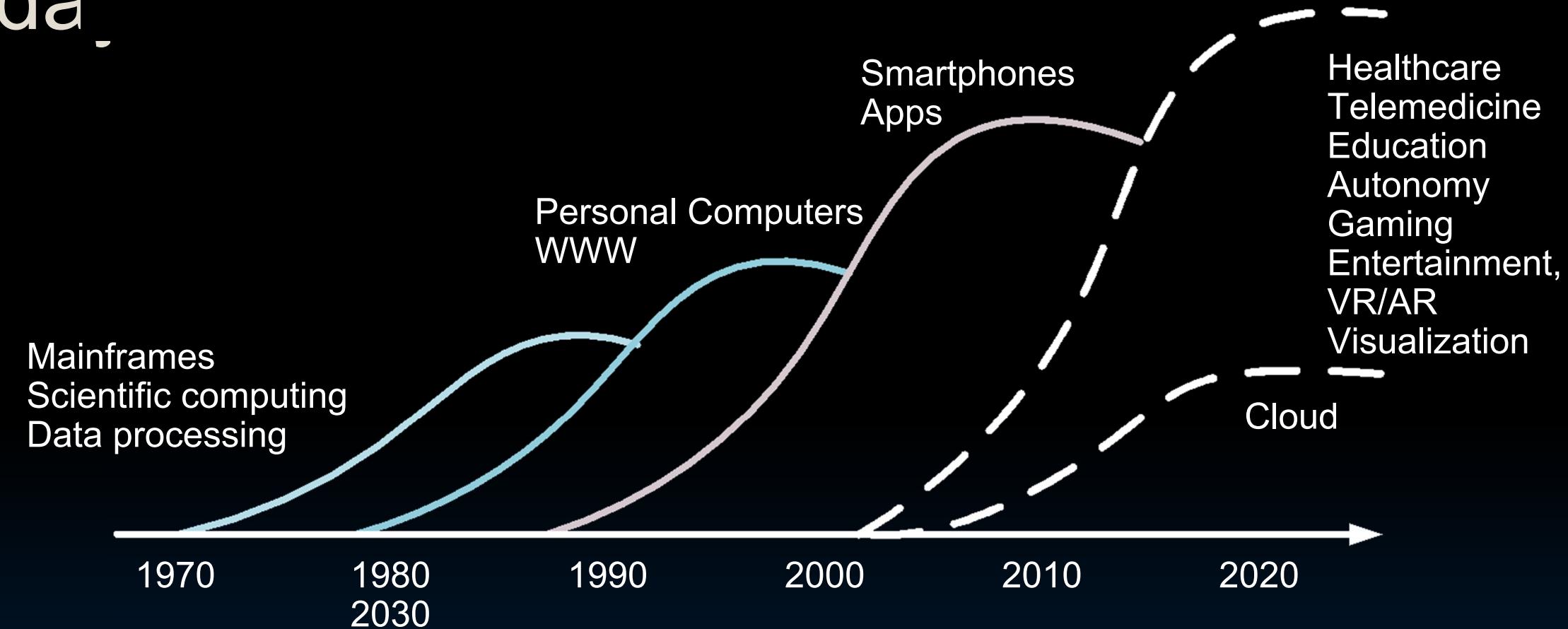


(dramatic
pause)

Why is computer
architecture exciting
today?

Why Is Computer Architecture Exciting

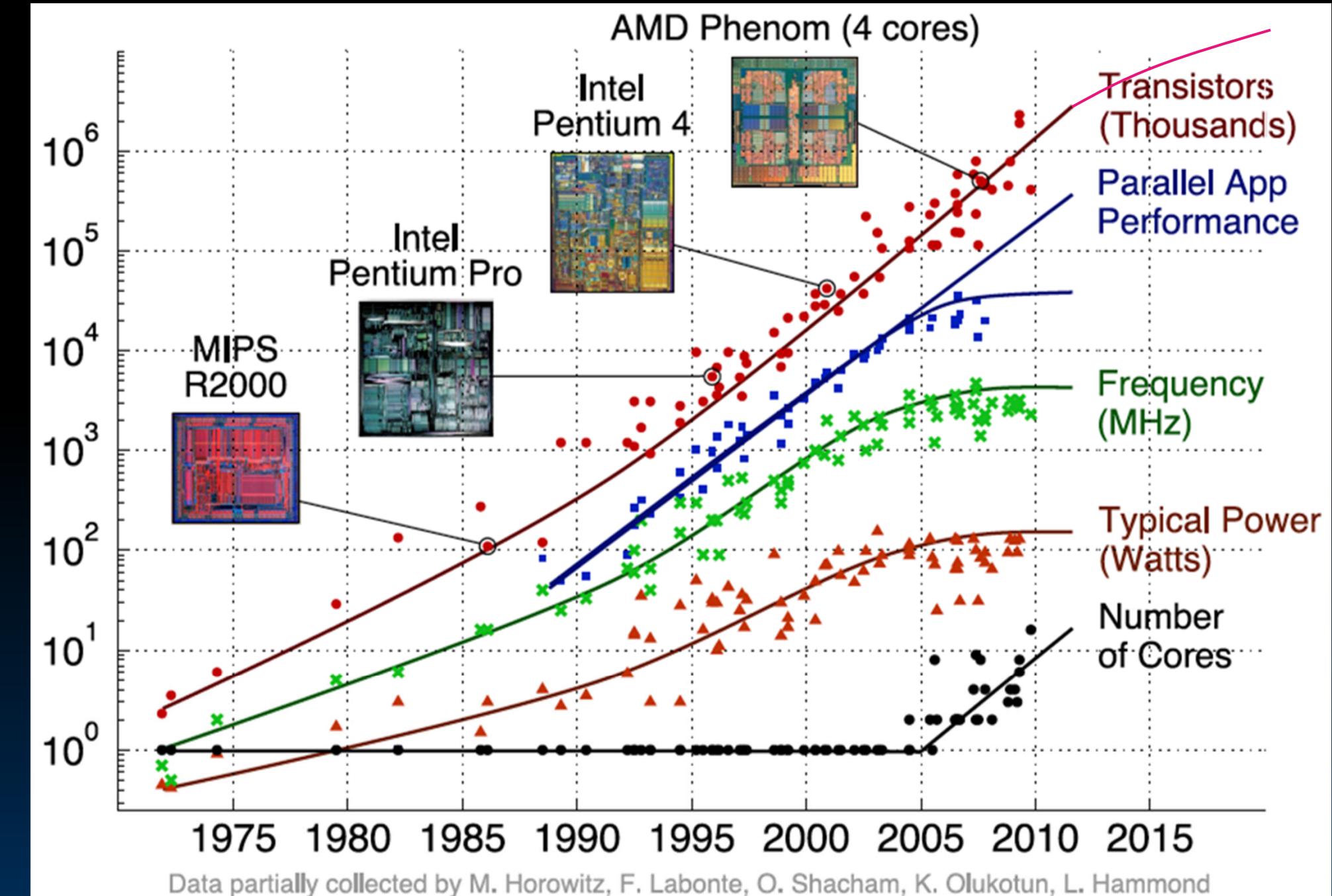
Today?



- 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

Reason 1: Changing Constraints

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



Reason 2: Era of Domain-Specific Computing

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

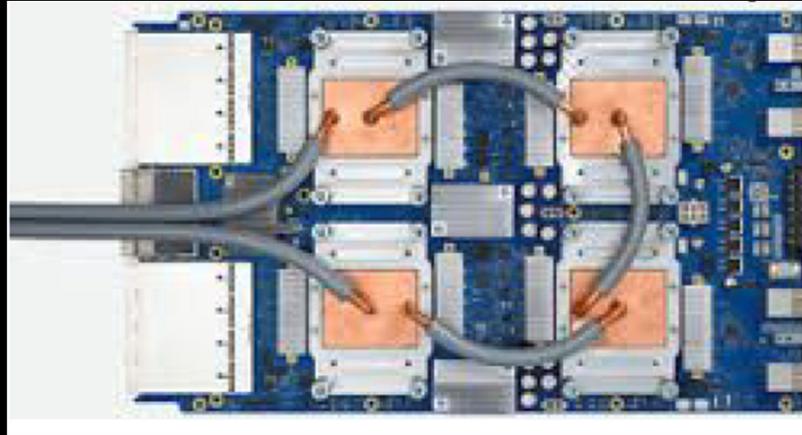


Apple A15 Bionic
Source: SemiAnalysis Garcia, Yokota

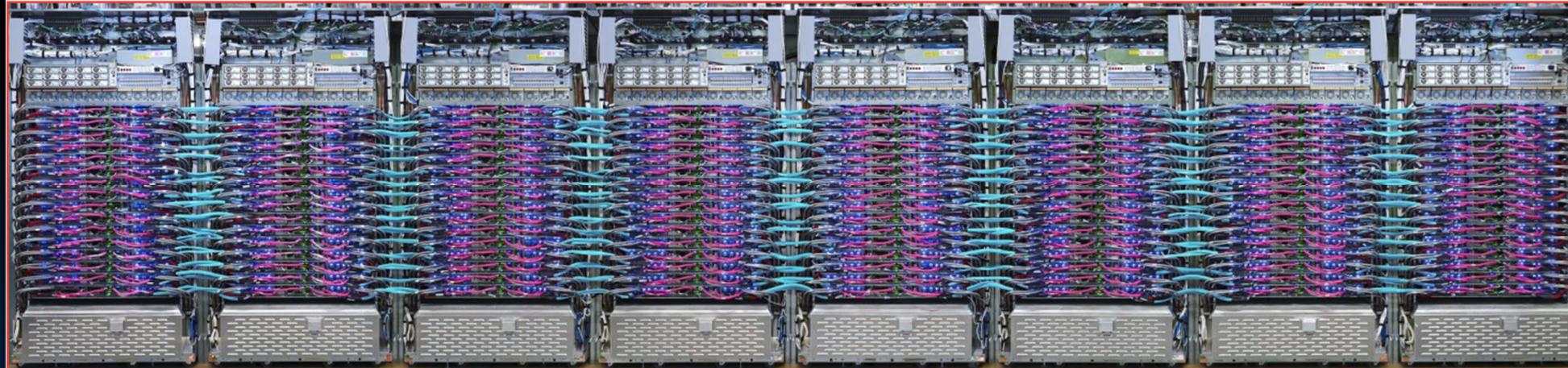
Old Conventional Wisdom

- 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



Google TPU3
Specialized Engine for training
Neural Networks
Deployed in cloud



1024 chips, >
100PetaFLOPs

Patterson and Hennessy win Turing!



Innovations in computer architecture have a convention of ***defying conventional wisdom.***

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

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



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
- Course Email: cs61c@berkeley.edu
 - Please DO NOT email/DM individual TAs/instructors with course-related questions
- Instructors: Dan Garcia & Justin Yokota
- Teaching Assistants: (see webpage)
- EdStem
 - Every announcement, discussion, clarification happens here
- Optional Readings: avg 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 Edition

Today's
required
reading:

**Our Course
Policies
page!**

Summary of Course Elements (1/2)

- **Lecture**
MWF for an hour, in-person or on Zoom.
If you miss it, watch the video. No attendance taken.
- **Discussion and Homework**
Design, Analyze, Illustrate
- **Lab and Project**
Implement, Simulate, Debug
Lab sections all on Monday (the same day that the corresponding lab is due); first one is 1/30, focusing on Lab 1
- **Midterm and Final Exam**
Rock on!!! Show your stuff!!!

Lab 0 (Setup)
due **next week (1/27)**

Lab 01,
Discussion 01
sections start
next week

Summary of Course Elements (2/2)

- Office Hours
 - TA: In-person or hybrid; queue-based
 - Instructor: In-person, group-based; “Concept OH” for exam prep
 - Full schedule currently in construction, will be up on the website soon
- Project Parties
 - None explicitly, but we may have extra OH around project deadlines
- Extension form
 - Late policy for all assignments (see website)
 - If you can’t meet deadline, **ask!**

- We're full with ~55 on the waitlist
 - Don't email us about the waitlist (we were overwhelmed by declared emails)
 - We are trying to expand, please keep up with the class so you're not behind if added
- Concurrent enrollment students allow this to happen, so thank you CE students!!
- If you don't turn in anything by week 3
 - we'll assume you signed up by accident and we will drop you

Extension Request Form

- All assignments due at 11:59:59 PM PT
- Extension form is available on the website under the "extensions" tab
- Please don't hesitate to ask for one!
 - We know life happens, we're here to help...
 - Note that OH support will be limited after an assignment's deadline

Exams (worth 40%, 8% per hour for 5 hrs)

- Midterm Exam
 - In-person, evening, standard paper exam, 2 hrs
 - ONE alternate slot immediately after if exam conflict
 - No remote exams outside DSP/special circumstances.
 - Prob 8th week, still waiting on room
- Final Exam
 - In-person, time TBD, standard paper exam, 3 hrs
 - ONE alternate slot immediately after if exam conflict
 - No remote exams outside DSP/special circumstances.
 - No clobber
 - ~2/3 for post-midterm material, ~1/3 for cumulative Qs

Course Grading

- Labs (10%)
 - Homework (10%)
 - Projects (40%)
 1. Non-Parallel Application (C)
 2. RISC-V Application
 3. Computer Processor Design (Logisim)
 4. Parallelization C Program
- ALL projects can be done individually or with one partner
- Midterm (16%)
 - Final (24%)

- **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 know and can do, 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 (extensions)
- 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!!



Policy on Assignments and Independent Work

(1/2)

- Projects (and labs) can be done either solo or in pairs.
- All homework is to be 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 Ed that help them debug (don't share code, even snippets there). We expect that what you hand in is yours.

Policy on Assignments and Independent Work

(2/2)

- 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 negative points in the assignment, 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.

- 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