CS 61C
Great Ideas in Computer Architecture
(a.k.a. Machine Structures)
Lecture 1: *Course Introduction*

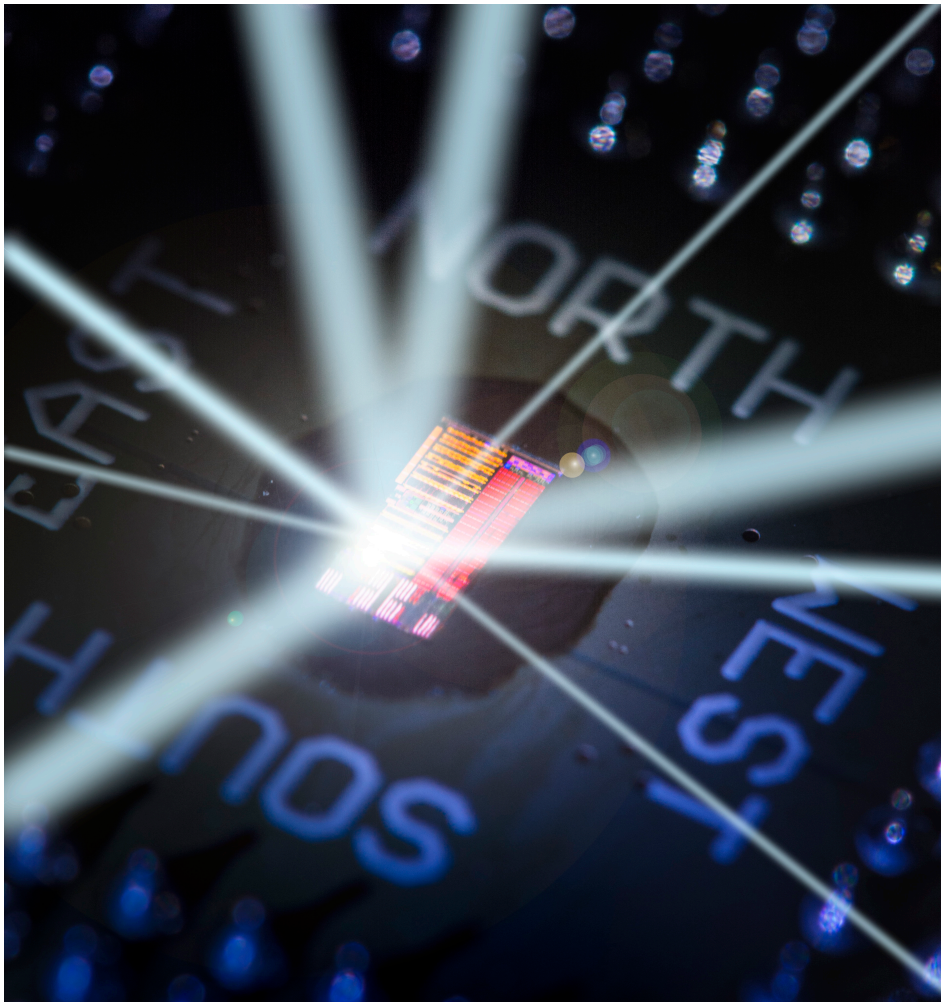
Instructors:

Nicholas Weaver (call me "Nick", and I'm no prof)
Professor Vladimir Stojanovic (call me "Vladimir")
(lots of help from **TAs,** esp. **Head TAs Fred and William**)

`http://inst.eecs.berkeley.edu/~cs61c/`

V. Stojanovic and K. Asanovic:

First Processor that Communicates with Light!



Nature, Dec. 2015

3mm X 6mm Chip Fabricated in 45nm SOI 75m+ transistors

Monolithically-Integrated Silicon Photonic Links

1MB SRAM Memory Structure for Testing

Dual-Core RISC-V Processor with Vector Accelerators

About me:
PhD Stanford – High-speed I/O
At Berkeley since 2013
At MIT since 2005

IC design, Sig. processing, Chip design with new devices

http://news.berkeley.edu/2015/12/23/electronic-photonic-microprocessor-chip/

Nicholas Weaver - Researcher

- Network Security & Network Measurement
  - Worms, malcode, things that go bump on the net…
  - Netalyzr
  - "Bang on it with a stick" security guy
- Former hardware person
  - Ph.D. In 2003 from Berkeley: Dissertation topic, FPGAs
- Hobbies:
  - Video games
  - Building NSA surveillance tools

Agenda

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

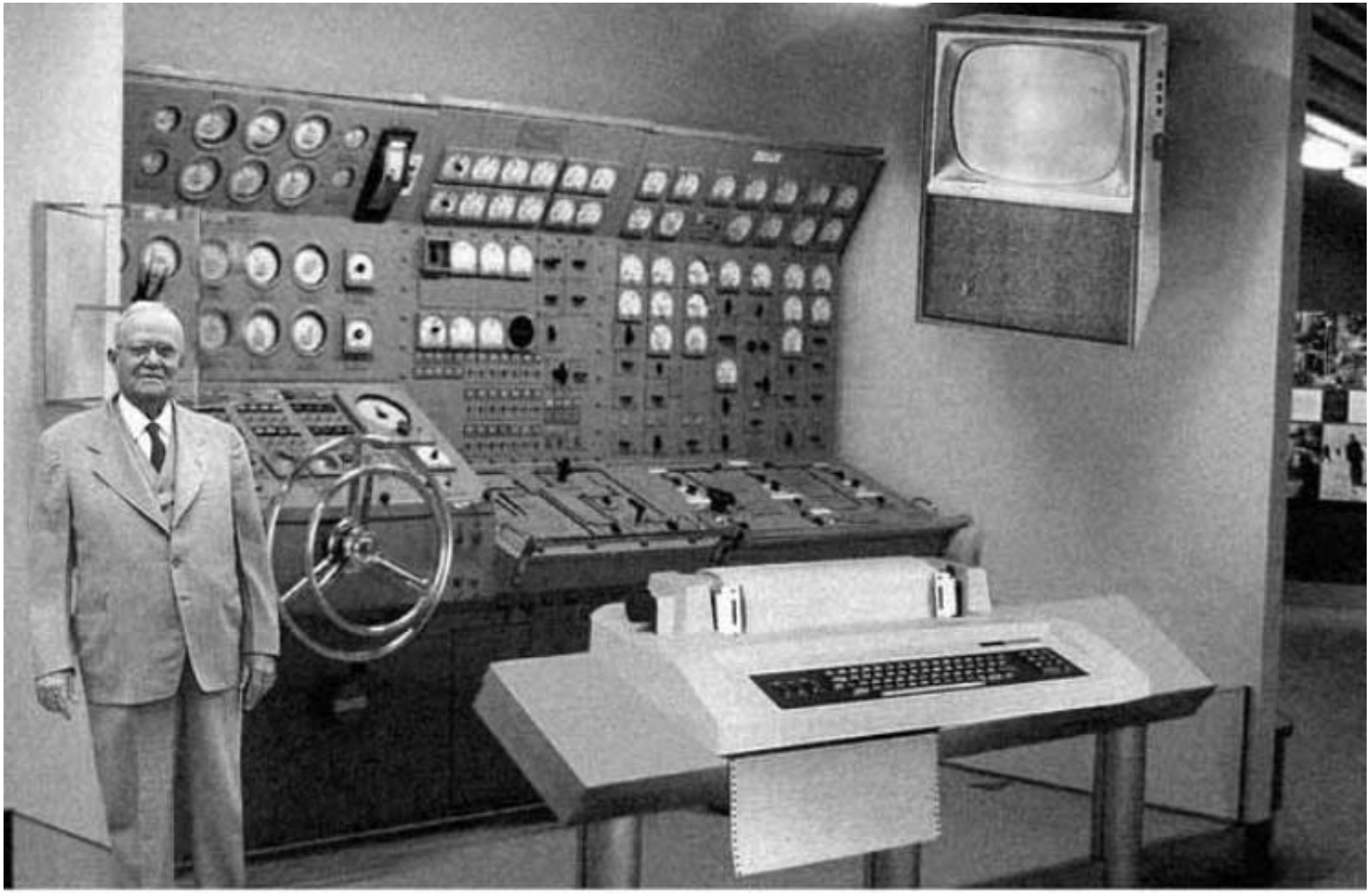
Agenda

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

CS61C is NOT really about C Programming

- It is about the *hardware-software interface*
  - What does the programmer need to know to achieve the highest possible performance
- C is close to the underlying hardware, unlike languages like Scheme, Python, Java!
  - Allows us to talk about key hardware features in higher level terms
  - Allows programmer to explicitly harness underlying hardware parallelism for high performance
  - Also allows programmer to shoot oneself in the foot in amazingly spectacular ways

Old School CS61C



Scientists from the RAND Corporation have created this model to illustrate how a "home computer" could look like in the year 2004. However the needed technology will not be economically feasible for the average home. Also the scientists readily admit that the computer will require not yet invented technology to actually work, but 50 years from now scientific progress is expected to solve these problems. With teletype interface and the Fortran language, the computer will be easy to use.

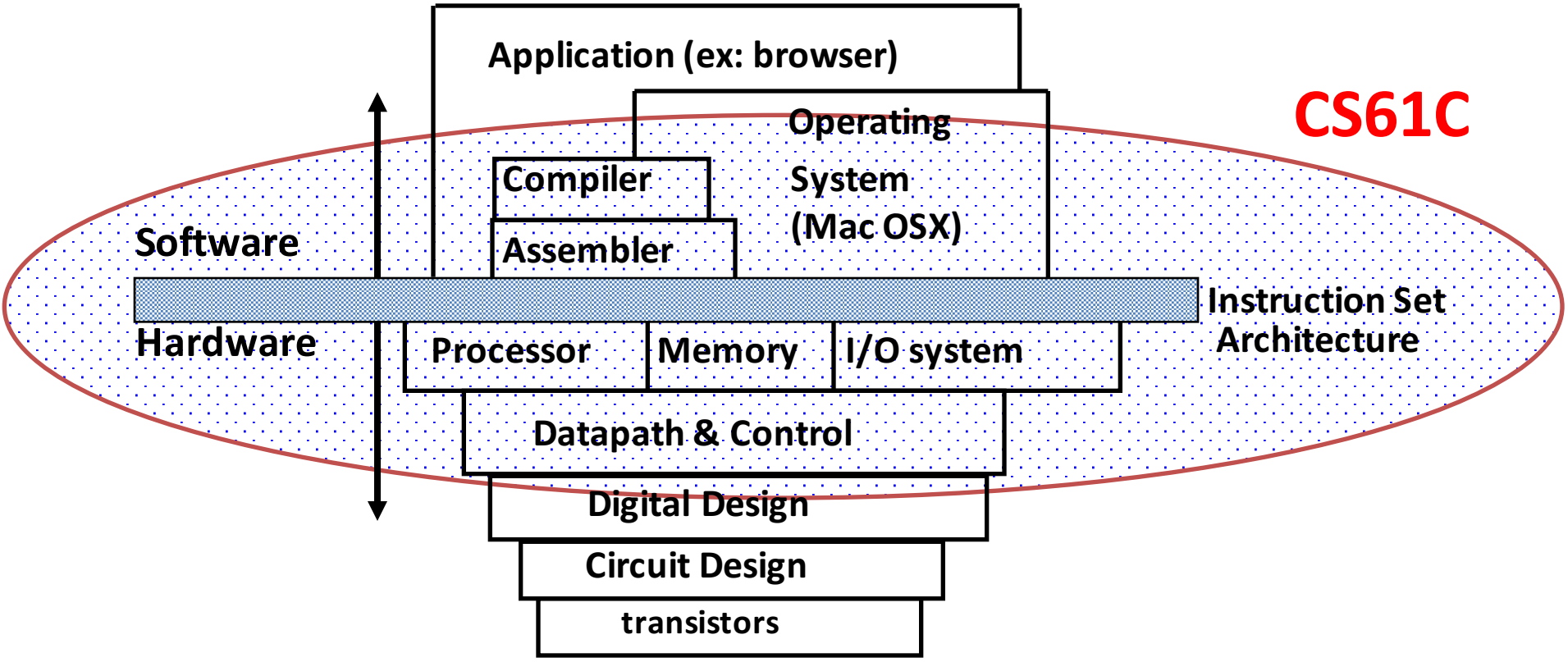
New School CS61C (1/2)

Personal Mobile Devices

New School CS61C (2/3)

New School CS61C (3/3)

Old School Machine Structures



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

Project 5

Software | *Hardware*

- 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 functioning in parallel at same time

Harness Parallelism & Achieve High Performance

Warehouse -Scale Computer

Smart Phone

Project 1,2

Computer

Core … Core

Memory (Cache)

Input/Output

Project 4

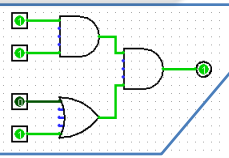
Core

Instruction Unit(s)

Functional Unit(s)

A_0+B_0 A_1+B_1 A_2+B_2 A_3+B_3

Main Memory



Logic Gates

Project 3

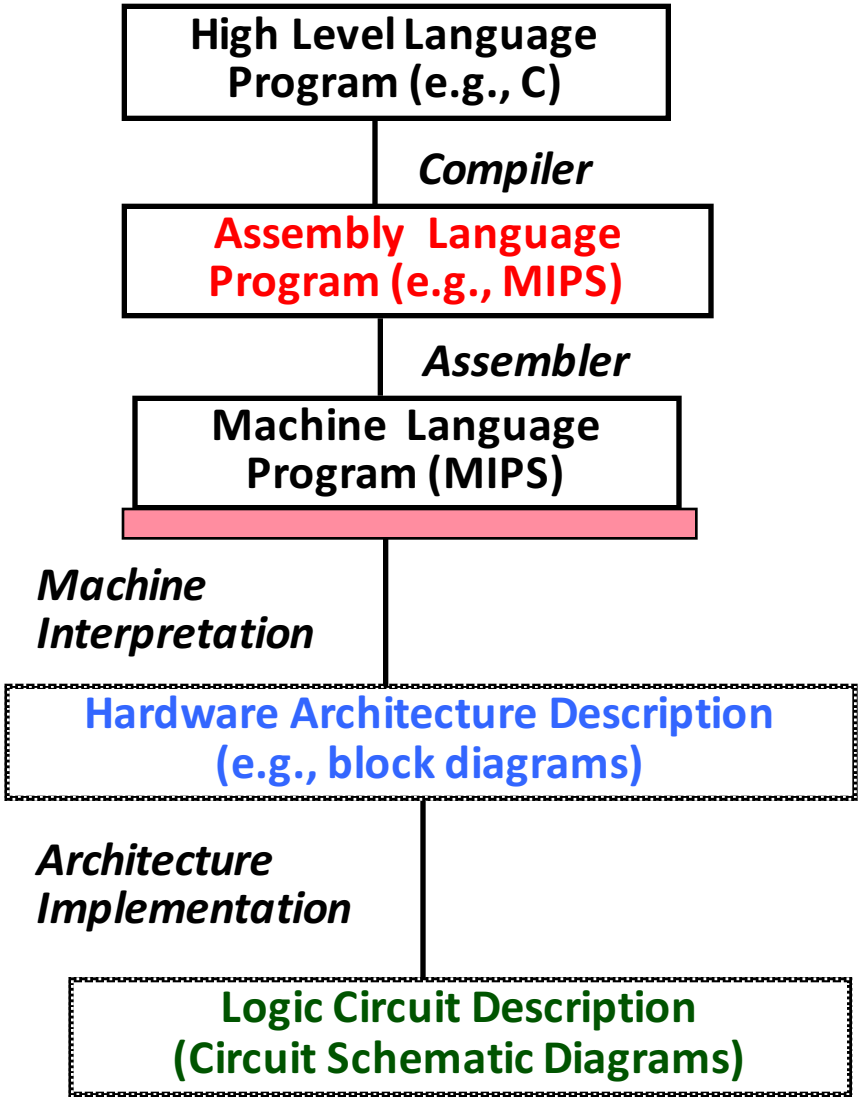
Agenda

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

5 Great Ideas in Computer Architecture

1. Abstraction
   (Layers of Representation/Interpretation)
2. Moore's Law (Designing through trends)
3. Principle of Locality (Memory Hierarchy)
4. Parallelism
5. Dependability via Redundancy

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

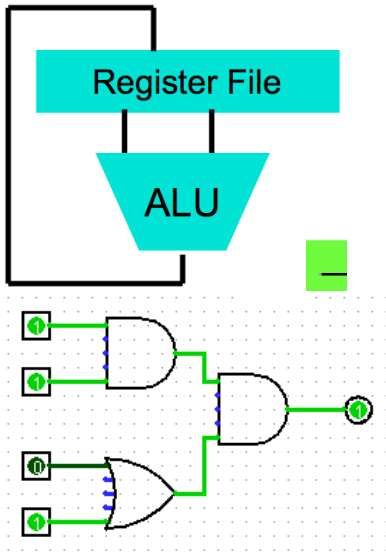



```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

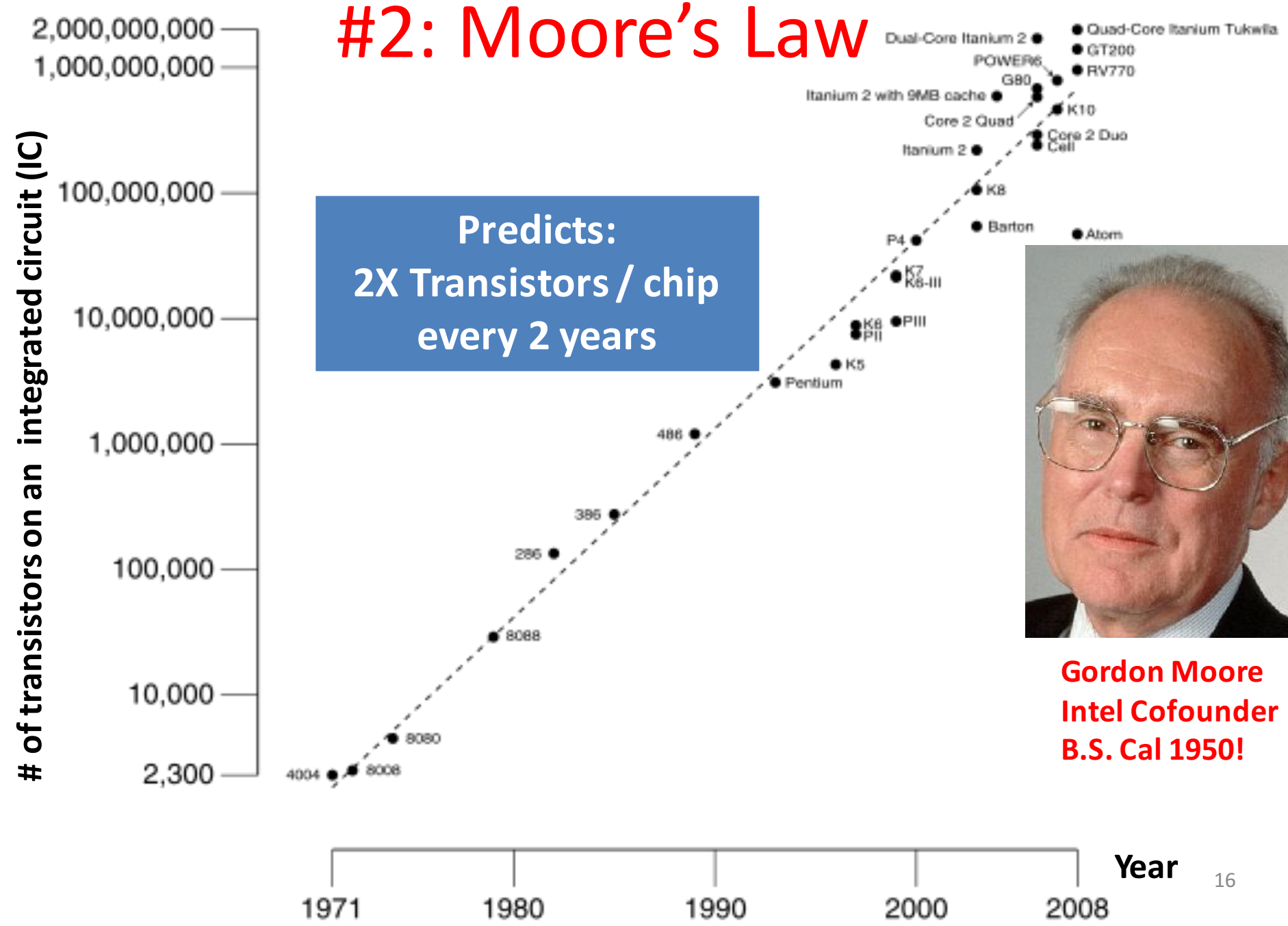
```
lw   $t0, 0($2)
lw   $t1, 4($2)
sw   $t1, 0($2)
sw   $t0, 4($2)
```

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

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000                .0
1100 0110 1010 1111 0101                )1
0101 1000 0000 1001 1100                .1
```



#2: Moore's Law



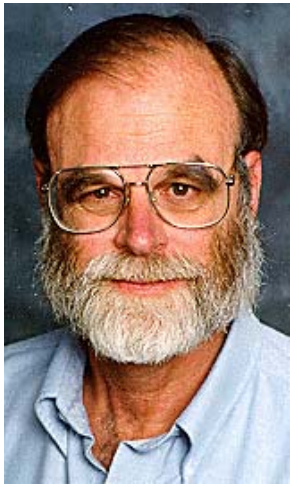
Gordon Moore
Intel Cofounder
B.S. Cal 1950!

Interesting Times

Moore's Law relied on the cost of transistors scaling down as technology scaled to smaller and smaller feature sizes.

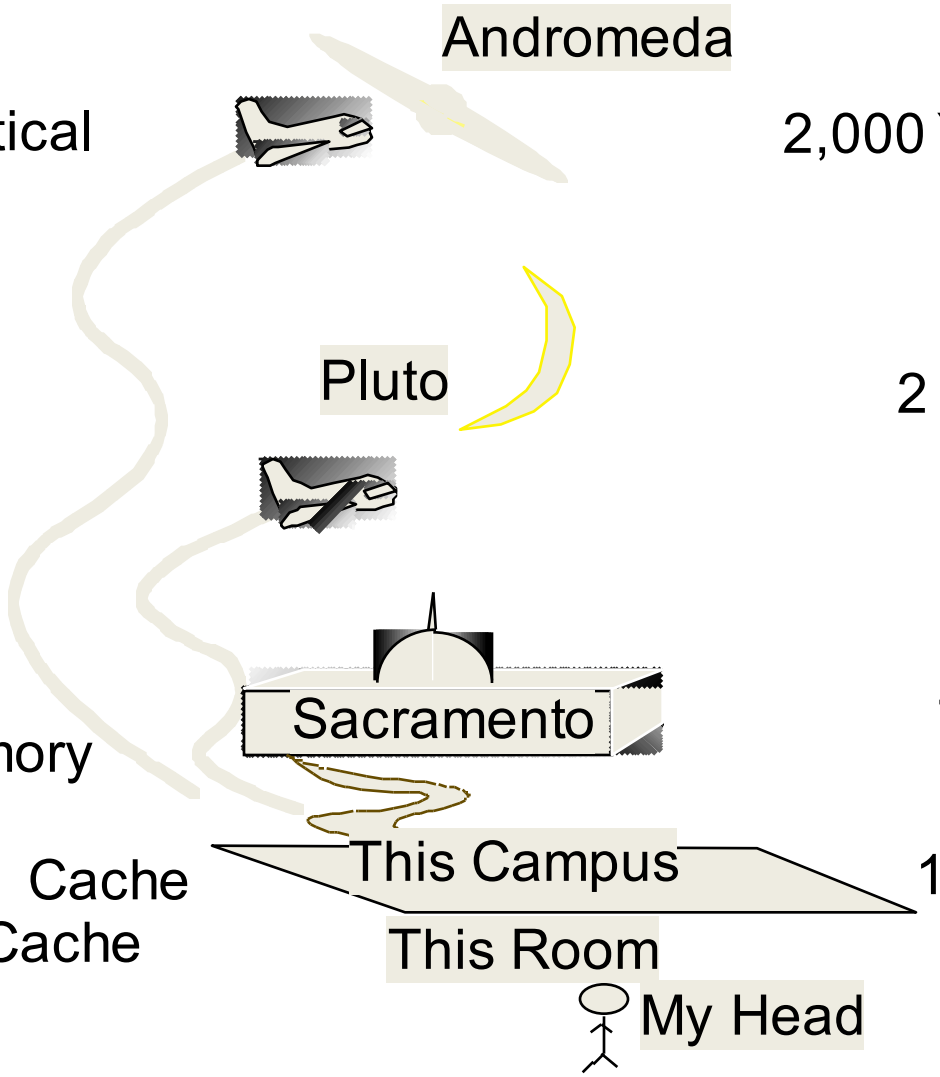
BUT newest, smallest fabrication processes <14nm, might have greater cost/transistor !!!!
So, why shrink????

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

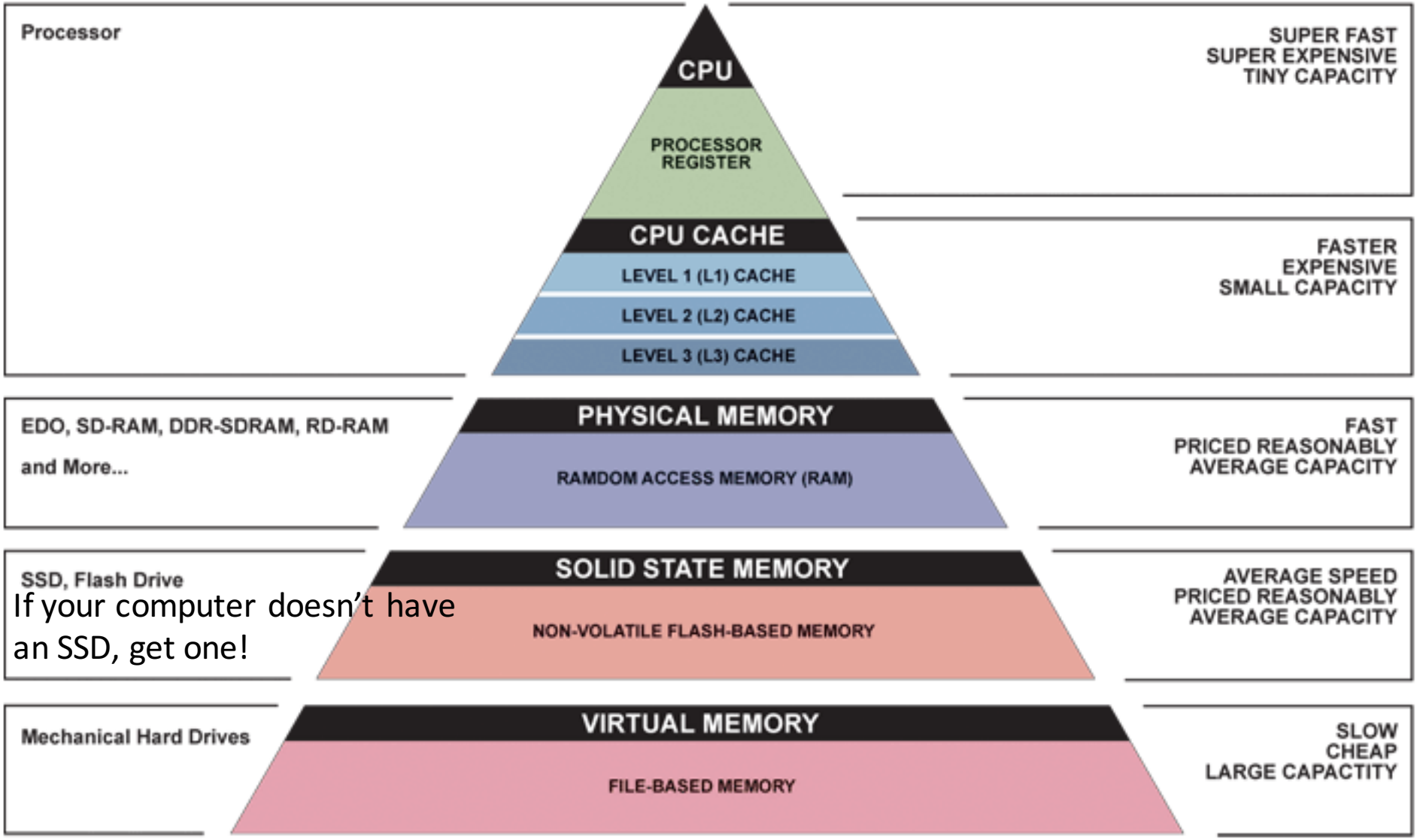


Jim Gray
Turing Award
B.S. Cal 1966
Ph.D. Cal 1969!

10^9	Tape /Optical Robot	2,000 Years
10^6	Disk	2 Years
100	Main Memory	1.5 hr
10	On Board Cache	10 min
2	On Chip Cache	
1	Registers	1 min
(ns)		

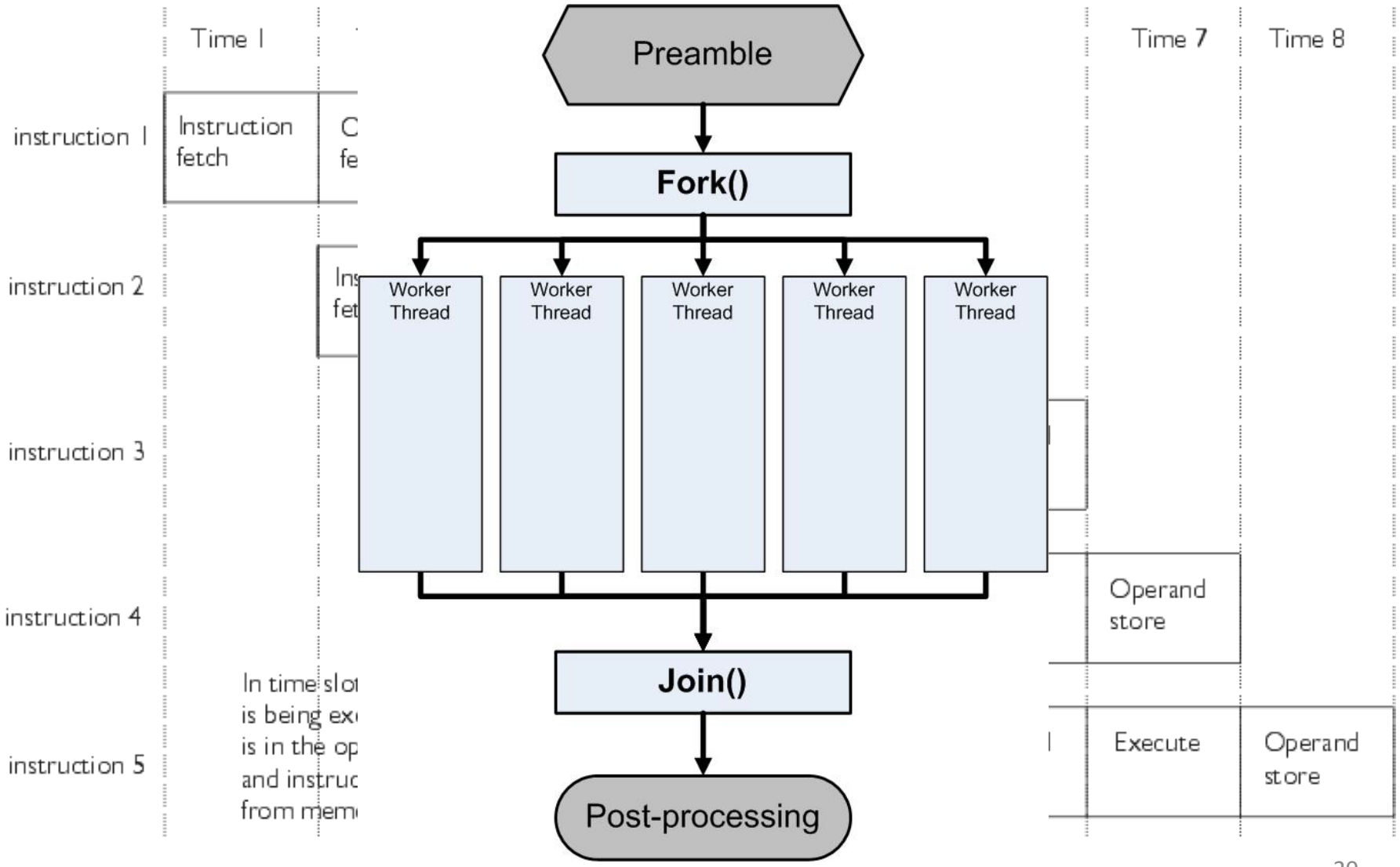


Great Idea #3: Principle of Locality/ Memory Hierarchy

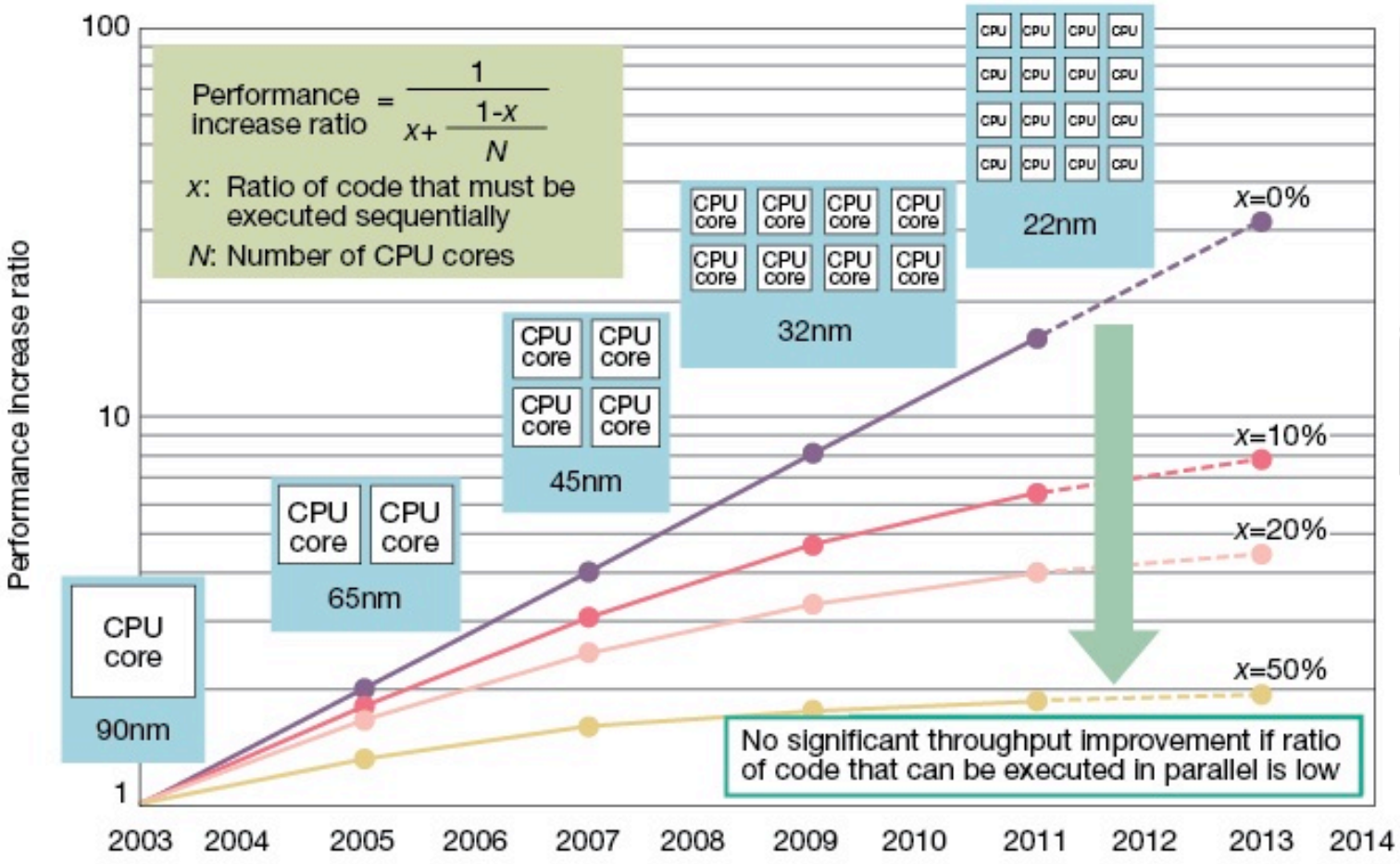


If your computer doesn't have an SSD, get one!

Great Idea #4: Parallelism



Caveat: Amdahl's Law



Gene Amdahl
Computer Pioneer

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.

Coping with Failures

- 4 disks/server, 50,000 servers
- Failure rate of disks: 2% to 10% / year
  - Assume 4% annual failure rate
- On average, how often does a disk fail?
  a) 1 / month
  b) 1 / week
  c) 1 / day
  d) 1 / hour

Coping with Failures

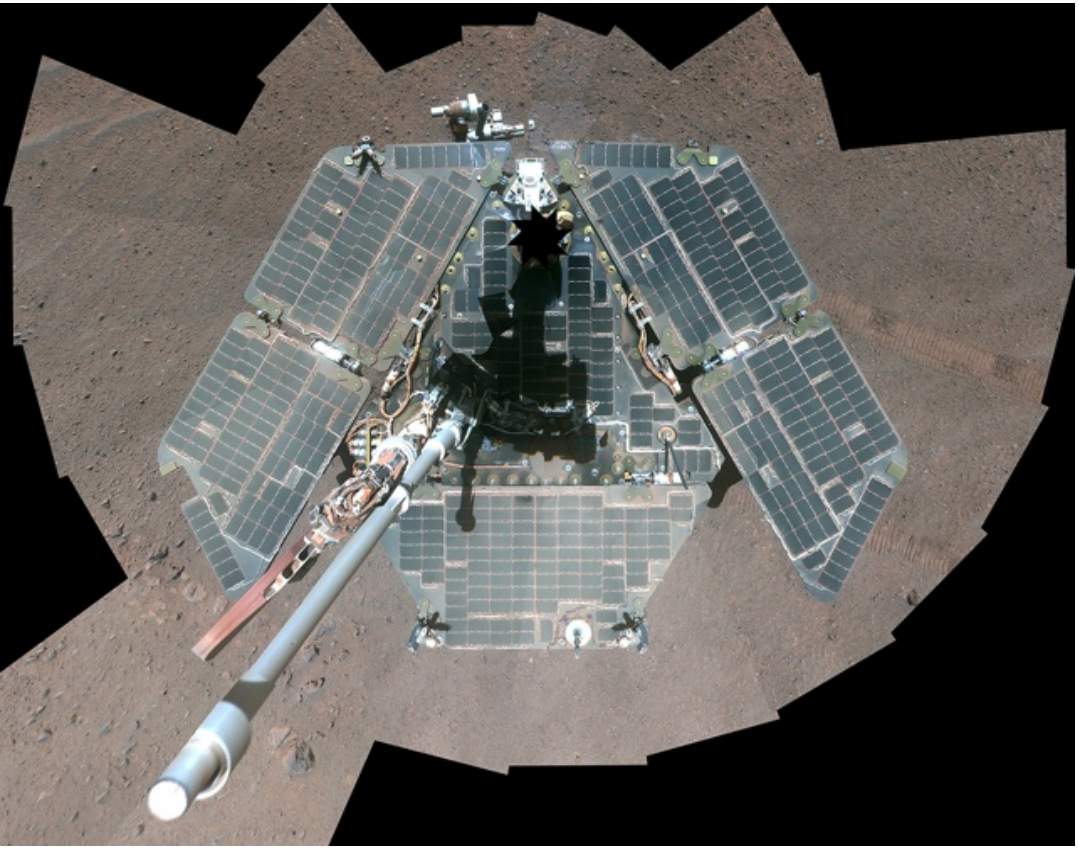
- 4 disks/server, 50,000 servers
- Failure rate of disks: 2% to 10% / year
  - Assume 4% annual failure rate
- On average, how often does a disk fail?
  a) 1 / month
  b) 1 / week
  c) 1 / day
  d) 1 / hour

50,000 x 4 = 200,000 disks

200,000 x 4% = 8000 disks fail

365 days x 24 hours = 8760 hours

NASA Fixing Rover's Flash Memory



Opportunity still active on Mars after >10 years

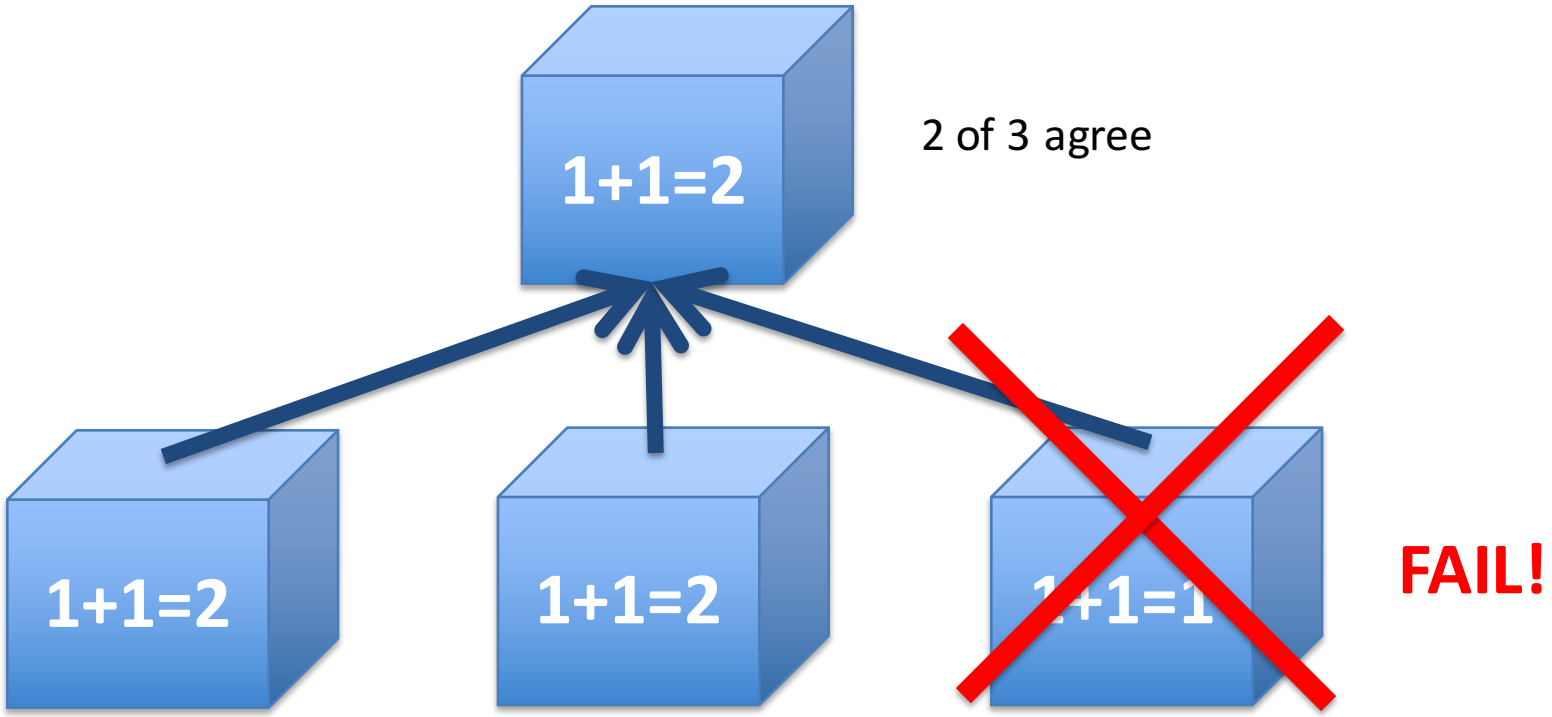
But flash memory worn out

New software update to avoid using worn out memory banks

http://www.engadget.com/2014/12/30/nasa-opportunity-rover-flash-fix/

Great Idea #5: 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 #5: 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 computers was Google's original internal innovation
  - 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)

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…

Weekly Schedule

	Monday	Tuesday	Wednesday	Thursday	Friday
9:00-9:30	Lecture 2050 VLSB	LAB 016 330 Soda	Lecture 2050 VLSB		Lecture 2050 VLSB
9:30-10:00					
10:00-10:30				DIS 116 3119 Etcheverry	
10:30-11:00					
11:00-11:30	LAB 011 330 Soda / LAB 023 271 Soda	LAB 017 330 Soda		DIS 117 3105 Etcheverry	
11:30-12:00					
12:00-12:30			DIS 111 3119 Etcheverry / DIS 123 320 Soda		
12:30-1:00					
1:00-1:30	LAB 012 330 Soda / LAB 022 271 Soda	LAB 018 330 Soda	DIS 112 3119 Etcheverry / DIS 122 320 Soda		
1:30-2:00					
2:00-2:30				DIS 118 102 Latimer	
2:30-3:00					
3:00-3:30	LAB 013 330 Soda	LAB 019 330 Soda		DIS 119 121 Wheeler	
3:30-4:00					
4:00-4:30			DIS 113 3119 Etcheverry	DIS 120 B56 Hildebrand	
4:30-5:00					
5:00-5:30	LAB 014 330 Soda	LAB 020 330 Soda	DIS 114 3119 Etcheverry / DIS 115 136 Barrows	DIS 121 3119 Etcheverry	
5:30-6:00					

Course Information

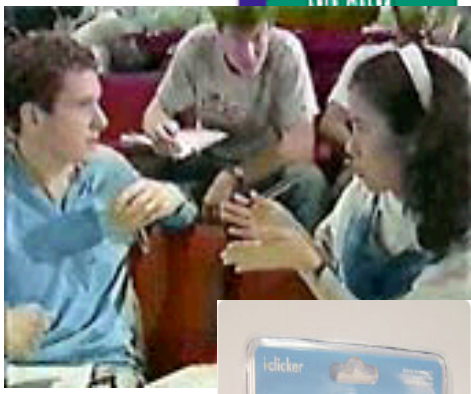
- Course Web: http://inst.eecs.Berkeley.edu/~cs61c/sp16
- Instructors:
  - Vladimir Stojanovic & Nicholas Weaver
- Teaching Assistants: (see webpage)
- Textbooks: Average 15 pages of reading/week (can rent!)
  - Patterson & Hennessey, *Computer Organization and Design*, 5/e (we'll try to provide 4th Ed pages, not Asian version 4th edition)
  - Kernighan & Ritchie, *The C Programming Language*, 2nd Edition
  - Barroso & Holzle, *The Datacenter as a Computer, 2nd Edition*
- Piazza:
  - Every announcement, discussion, clarification happens there

Course Grading

- EPA: Effort, Participation and Altruism (5%)
- Homework (5%)
- Labs (5%)
- Projects (25%) (New – Projects done and submitted individually)
  1. Build a regular expressions matcher (C)
  2. Assembler and Linker (MIPS & C)
  3. Computer Processor Design (Logisim)
  4. Parallelize for Performance, SIMD, MIMD
  5. Massive Data Parallelism (Spark on Amazon EC2)
- Two midterms (15% each): 6^{th} & 12^{th} week evening the day of the class, can be clobbered!
- Final (30%): 2016/5/09 @ 7-10pm
  - Will announce make-up final time later
- Performance Competition for honor (and EPA)

Tried-and-True Technique: Peer Instruction

- Increase real-time learning in lecture, test understanding of concepts vs. details
- As complete a "segment" ask multiple-choice question
  - 1-2 minutes to decide yourself
  - 2 minutes in pairs/triples to reach consensus.
  - Teach others!
  - 2 minute discussion of answers, questions, clarifications
- You can get transmitters from the ASUC bookstore
  - We'll start this next week
  - No web-based clickers, sorry!



EECS Grading Policy

- http://www.eecs.berkeley.edu/Policies/ugrad.grading.shtml

  "A typical GPA for courses in the lower division is 2.7. This GPA would result, for example, from 17% A's, 50% B's, 20% C's, 10% D's, and 3% F's. A class whose GPA falls outside the range 2.5 - 2.9 should be considered atypical."

- Fall 2010: GPA 2.81
  26% A's, 47% B's, 17% C's,
  3% D's, 6% F's
- Job/Intern Interviews: They grill you with technical questions, so it's what you say, not your GPA

  (New 61C gives good stuff to say)

	Fall	Spring
2015	2.82	
2010	2.81	2.81
2009	2.71	2.81
2008	2.95	2.74
2007	2.67	2.76

Our goal as instructors

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

EPA!

- Effort
  - Attending prof and TA office hours, completing all assignments, turning in HW, doing reading quizzes
- Participation
  - Attending lecture and voting using the clickers
  - Asking great questions in discussion and lecture and making it more interactive
- Altruism
  - Helping others in lab or on Piazza
- EPA! points have the potential to bump students up to the next grade level! (but actual EPA! scores 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 is late (even by a millisecond) we deduct a token
- After you've used up all tokens, it's 33% deducted per day.
  - No credit if more than 3 days late
  - Cannot be used on homeworks!
- No need for sob stories, just use a slip day!

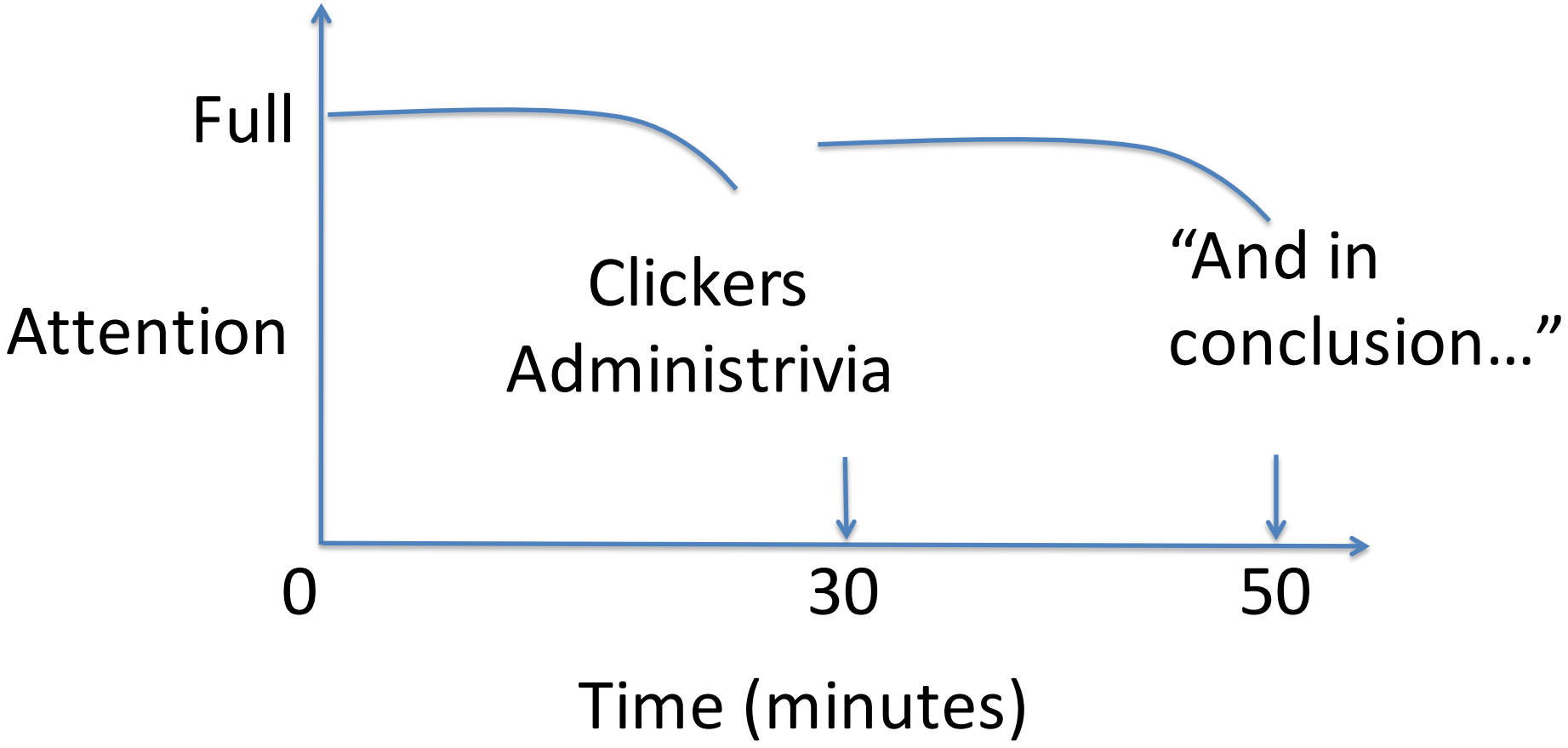
Policy on Assignments and Independent Work

- **ALL PROJECTS WILL BE DONE AND SUBMITTED INDIVIDUALLY**
- With the exception of laboratories and assignments that explicitly permit you to work in groups, all homework and projects are to be YOUR work and your work ALONE.
- You are encouraged to discuss your assignments with other students, and extra credit will be assigned to students who help others, particularly by answering questions on Piazza, but 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 use PUBLIC GitHub archives (giving your answers away)
- We have tools and methods, developed over many years, for detecting this. You WILL be caught, and the penalties WILL be severe.
- **At the minimum F in the course**, and a letter to your university record documenting the incidence of cheating.
- (We've caught people in recent semesters!)
- **Both Giver and Receiver are equally culpable and suffer equal penalties**

Use Git and Push Often…

- You will be using BitBucket to host your projects for submission…
  - So use it for your normal workflow too
- Push your work back to BitBucket on a regular basis
  - It really prevents screwups: “Ooops, go back” is the reason for version control
  - It gives a timestamp of when you wrote your code
    - Very useful if flagged for cheating

Architecture of a typical Lecture



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