Lecture 1

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

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

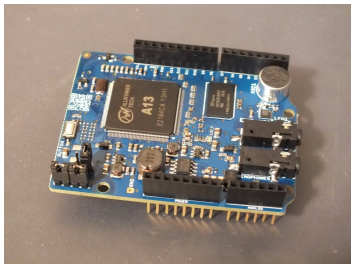
Dr. Gerald Friedland
Adjunct Assistant Professor

LAWRENCE LIVERMORE NATIONAL LABORATORY

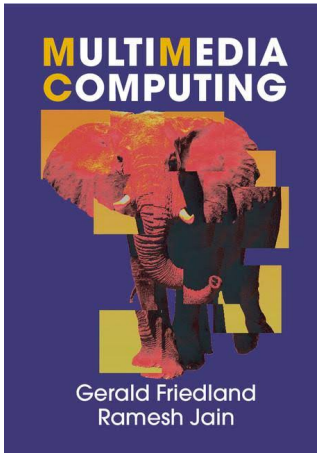
ICSI INTERNATIONAL COMPUTER SCIENCE INSTITUTE

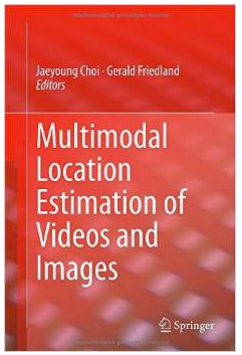
- My primary specialty is Multimedia Computing
- I also work on Privacy and Privacy Education
- The combination led me to build hardware!
- Previously taught (undergraduate):
  - CS10 (The Beauty and Joy of Computing)
  - CS88 (Computational Structures for Data Science)

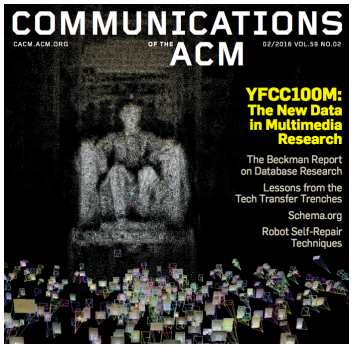
<Teaching Privacy>



AUDEME







Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

Dr. Nick Weaver
Lecturer

- My primary specialty is Network Security and Network Measurement
  - Although a reformed hardware person, originally specializing in FPGAs
- I will sprinkle a fair bit of security stuff throughout the lectures
  - Security is not an afterthought, but needs to be engineered in from the start: Since this class covers everything from the transistor to the cloud, I'll make security notes along the way

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

Course Information

- Course Web: http://www-inst.eecs.bekeley.edu/cs61c/
- Instructors:
  - Gerald Friedland & 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*

Course Grading

- EPA: Effort, Participation and Altruism (5%)
- Homework (5%)
- Labs (5%)
- Projects (25%) (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): 6th & 12th week evening the day of the class, can be clobbered!
- Final (30%)
- Performance Competition for honor (and EPA)

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

Piazza & Slack

- Piazza is an official channel
  - We will post announcements in it and we expect that, by posting announcements, you will read them
  - You can use this as a discussion forum to ask open questions
  - If you have private questions of the instructors and staff: ***do not use email, use a private question in Piazza***
- The Slack channel is an ***unofficial channel***
  - But some may drop in anyway.

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 don't know if the WiFi is good enough for REEF soft-clickers

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

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

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

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! Why are we not 7.0 for you? **We will listen!**

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

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
    - If you have a lecture conflict, please note it for us on Piazza
      If you just miss one or two lectures, don't worry about it....
  - Asking great questions in discussion and lecture and making it more interactive
- Altruism
  - Helping others in lab or on Piazza: Be Excellent to Each Other
- EPA! points have the potential to bump students up to the next grade level! (but actual EPA! scores are internal)

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

Late Policy…
Slip Days!

- Assignments due at 11:59:59 PM PT
- 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 (or even "start with") solutions from other students or 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.
- Both Giver and Receiver are equally culpable and suffer equal penalties
  - If it is from a previous semester, the previous semester's students will ***also be reported to the student conduct office***

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
- Also, for any C coding, use valgrind
  - C is ***not memory safe***,
    valgrind will catch most of these errors when you make them

Intellectual Honesty Policy: Detection and ***Retribution***

- We view those who would cheat as “attackers”
  - This includes sharing code on homework or projects, midterms, finals, etc…
  - But we (mostly) assume rational attackers: Benefit of attack > ***Expected*** cost
    - Cost of launching attack + cost of getting caught * probability of getting caught
- We take a detection and response approach
  - We use many tools to detect violations
    - "Obscurity is not security", but obscurity can help. Just let it be known that "We Have Ways"
  - We will go to DEFCON 1 (aka "launch the nukes") ***immediately***
    - “Nick doesn’t make threats. ***He keeps promises***”
    - Punishment can be up to an F in the class but, at minimum, ***negative points***:
      - You will do better if you don't do your work at all than if you cheat
      - ***All*** incidents will be reported to the office of student conduct

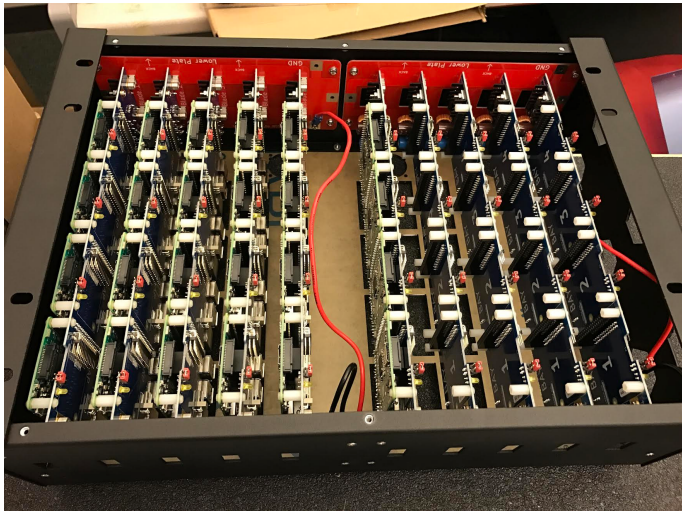
Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

Stress Management & Mental Health...

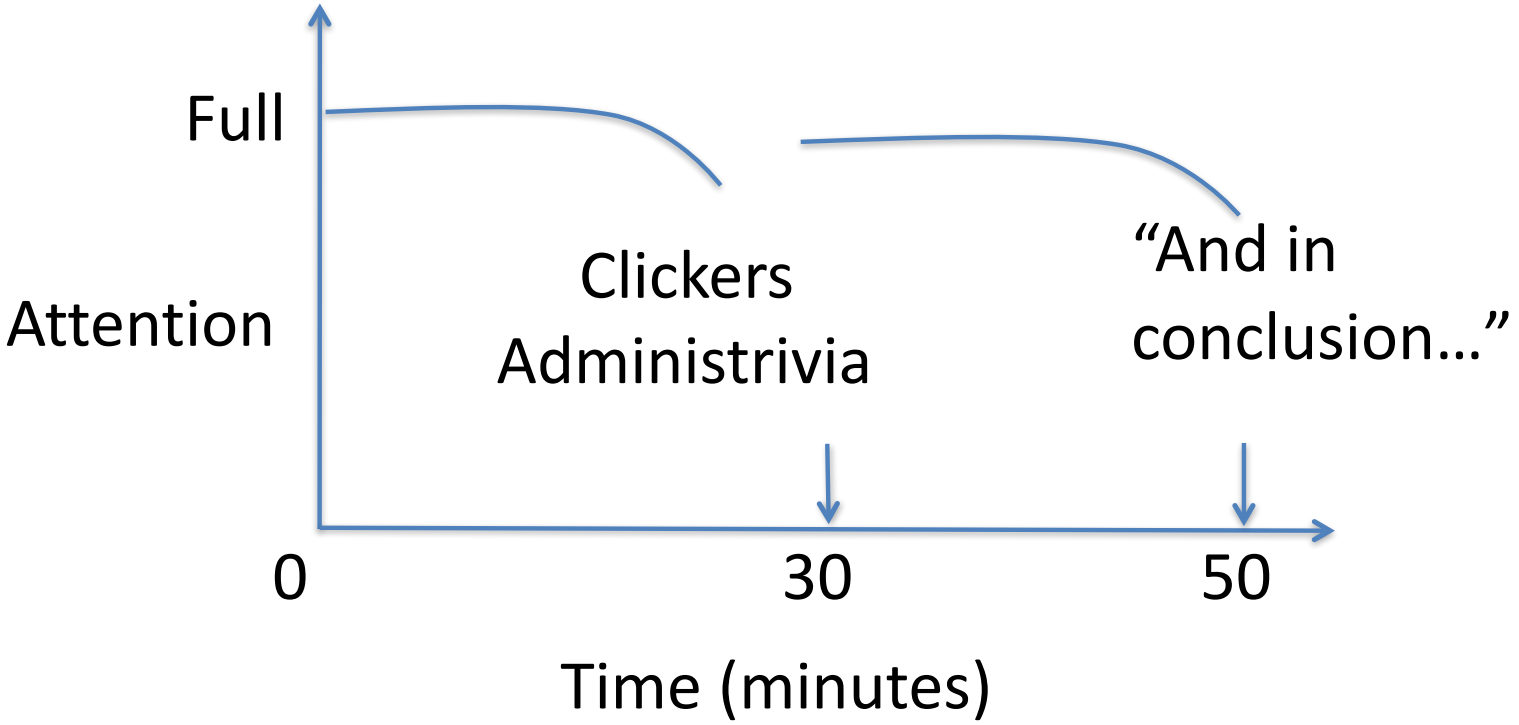
- We'll try to not over-stress you too much
  - But there really is a lot to cover and this really is a demanding major
  - There are 5 projects!
- If you feel overwhelmed, please use the resources available
  - Academically: Ask on Piazza, Tutoring, Office hours, the Slack channel, Guerrilla sections, etc...
  - Non-Academic: Take advantage of University Health Services if you need to
    - ***Nick did!*** Zoloft (an antidepressant) and therapy saved his life, twice.

Oh, and get a Raspberry Pi 3...

- You don't have to (we have a cluster for remote login), but you really want to get one...
  - Will work for all projects
  - Project 4 and 5 will specifically target the Raspberry Pi 3
- Its also a beast:
  - GHz processor, quad core, 64b ARM processor
    - 1 GB of RAM, 100 Mbps Ethernet, WiFi, Bluetooth, HDMI-out, 4x USB 2.0
  - SD card (I prefer 128GB but 8GB will do)
    - Complete systems for <$70
- Compare with Nick's computer when he took CS60b...
  - NeXTStation: 25 MHz 68040 processor, 20 MB of RAM, 200 MB hard drive and it cost nearly 100x as much!



Architecture of a typical Lecture



Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

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
- C is close to the underlying hardware, unlike languages like Scheme, Python, Java, Go, Perl, R, Prolog...
  - C is portable PDP8 Assembly Language.
    C++ is portable PDP8 Assembly Language with delusions of grandeur
  - 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
  - One of the goals in this class is for you to develop a ***rational hatred*** of C

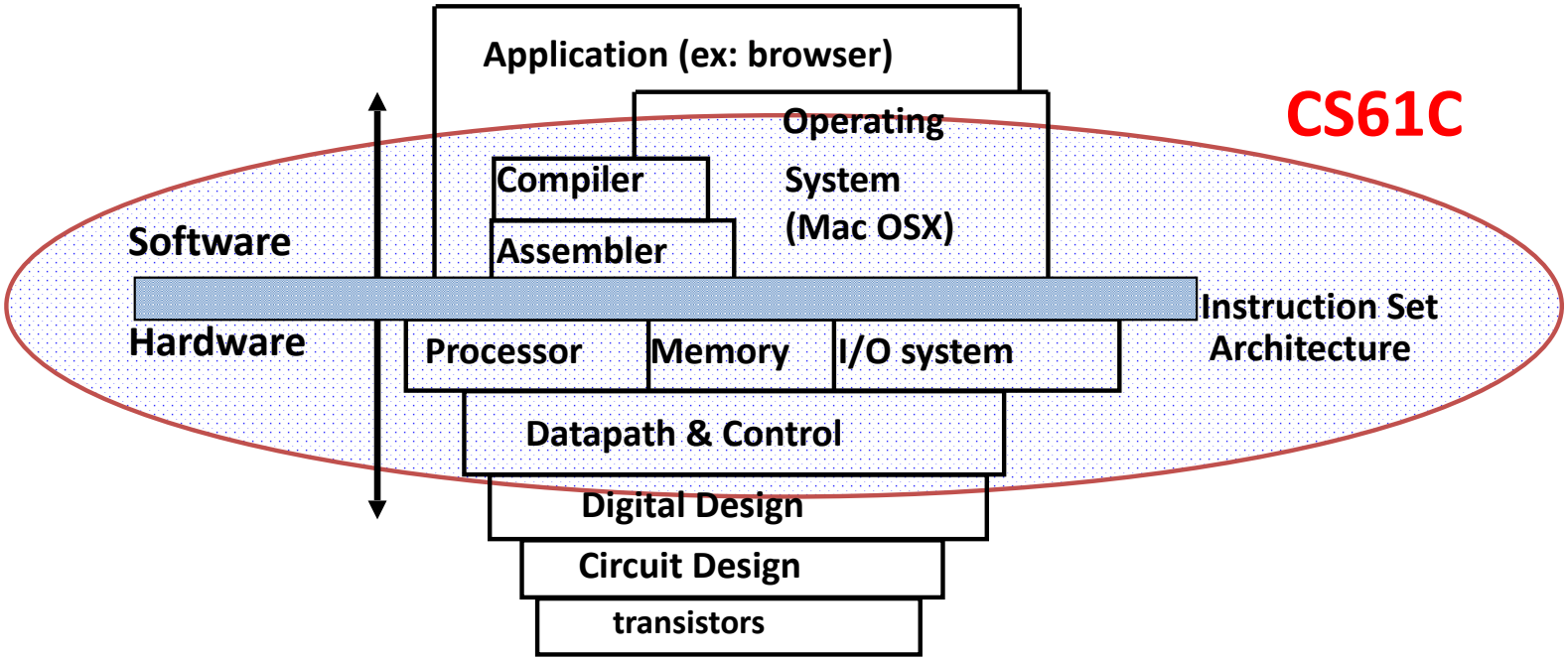
Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

Modern 61C:
From the small...

To the Big...

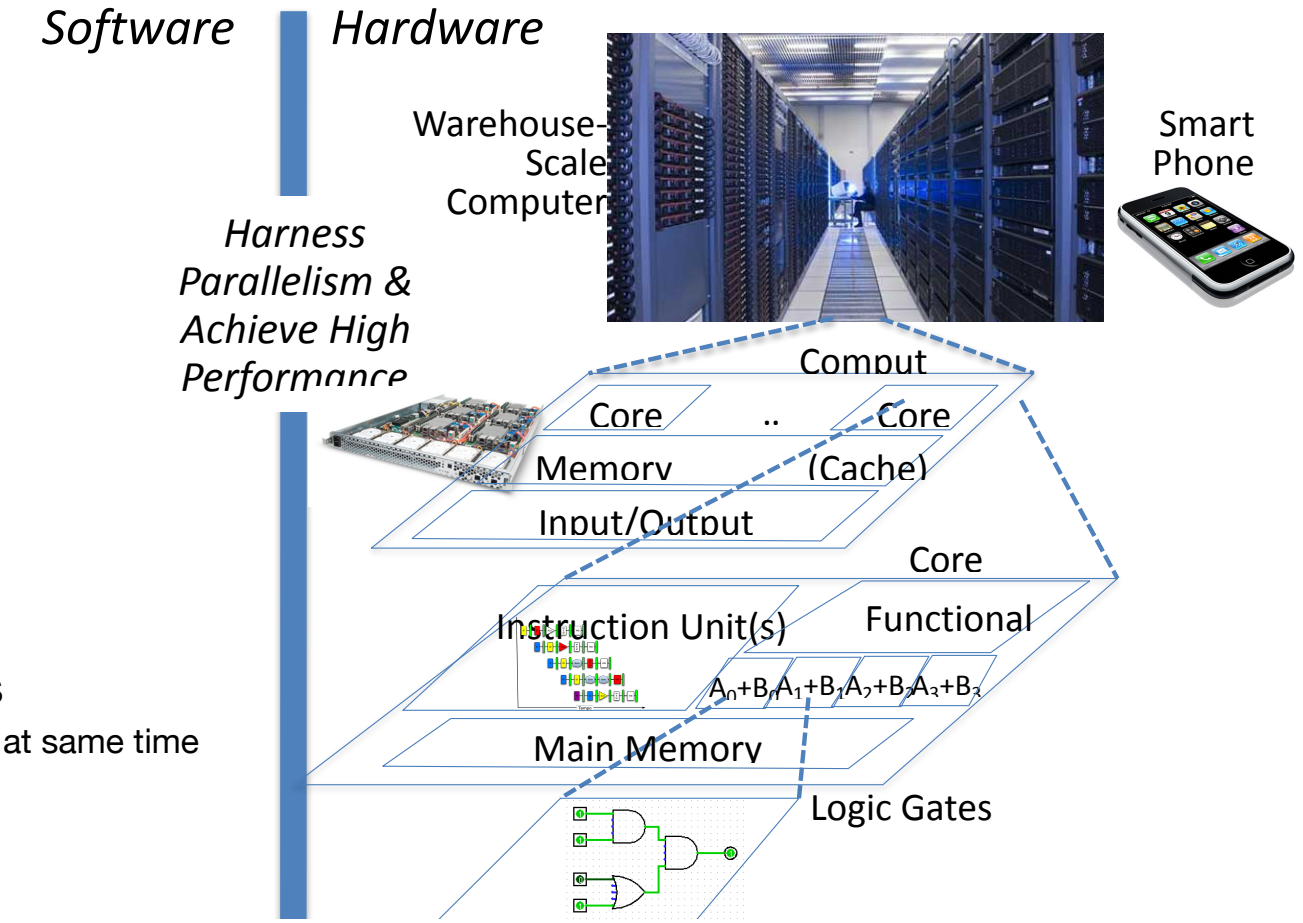

warehouse-scale
computer
My other computer
is a data center

Old School Machine Structures



New School 61C: From the Data Center to the Gate

- 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



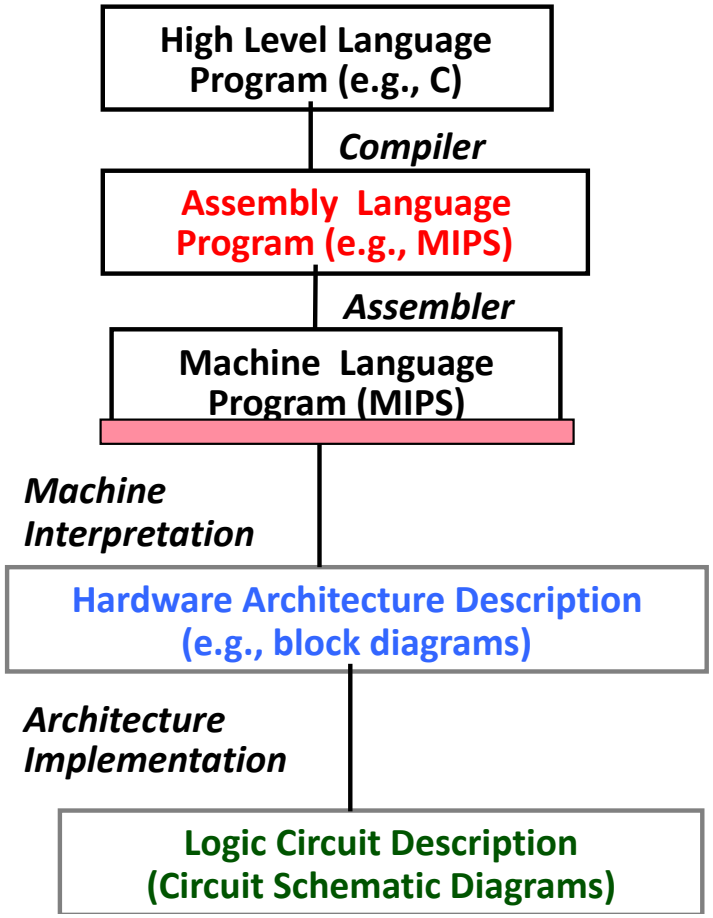
Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

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 & Amdahl's law (which limits it)
5. Dependability via Redundancy

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

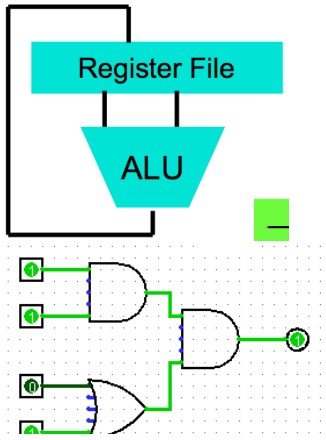

```
lw    $t0, 0($2)
lw    $t1, 4($2)
sw    $t1, 0($2)
sw    $t0, 4($2)
```




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

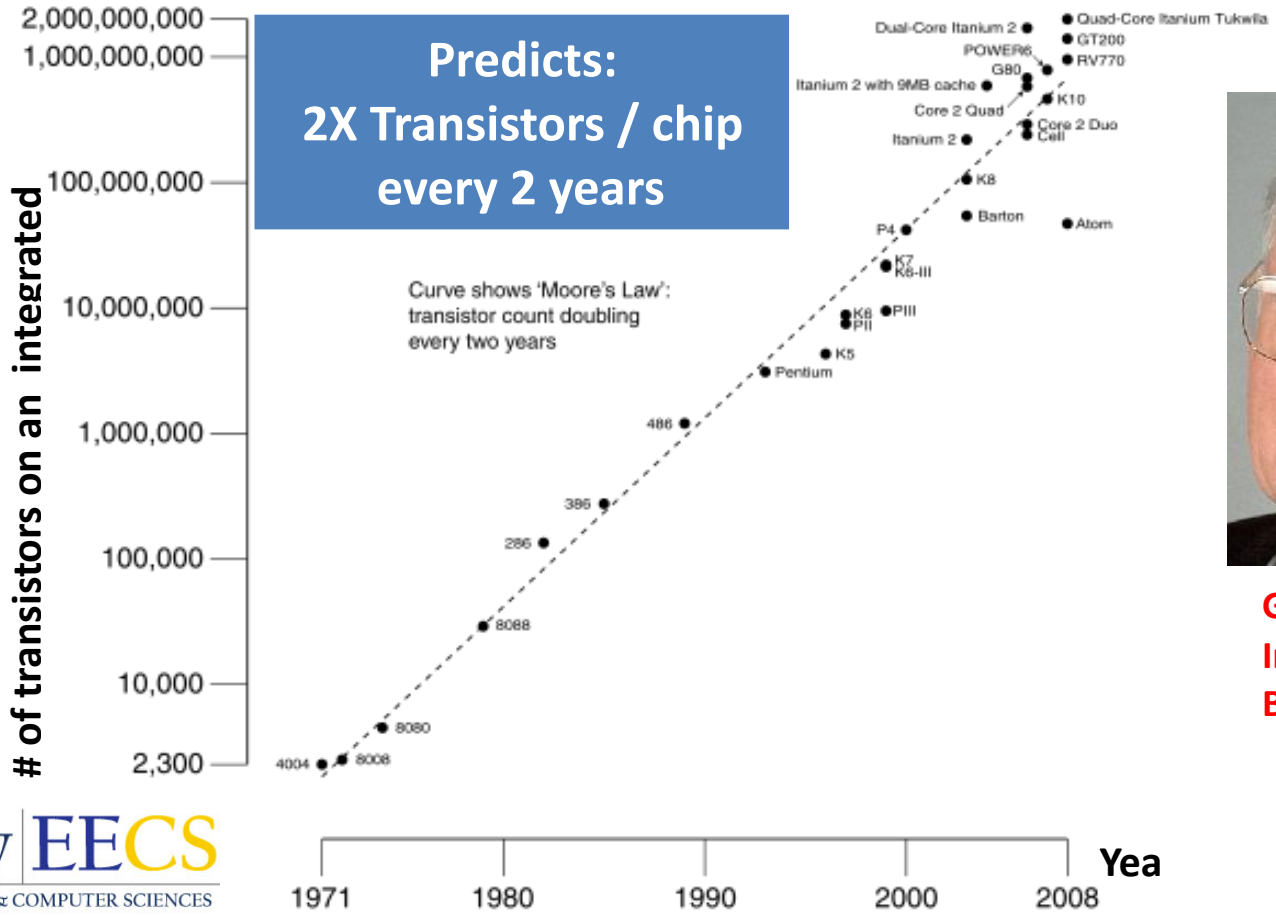
Anything can be represented as a *number*, i.e., data or instructions

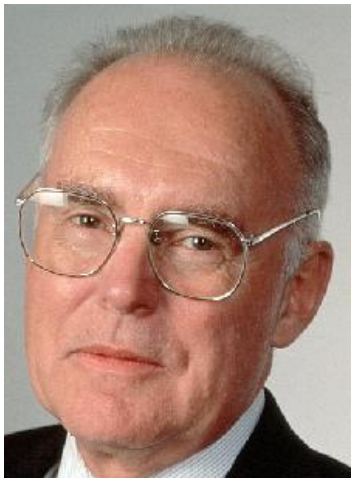
```
0000 1001 1100 0110 1010
1010 1111 0101 1000 0000
1100 0110 1010 1111 0101
0101 1000 0000 1001 1100
```



Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

#2: Moore's Law





Gordon Moore
Intel Cofounder
B.S. Cal 1950!

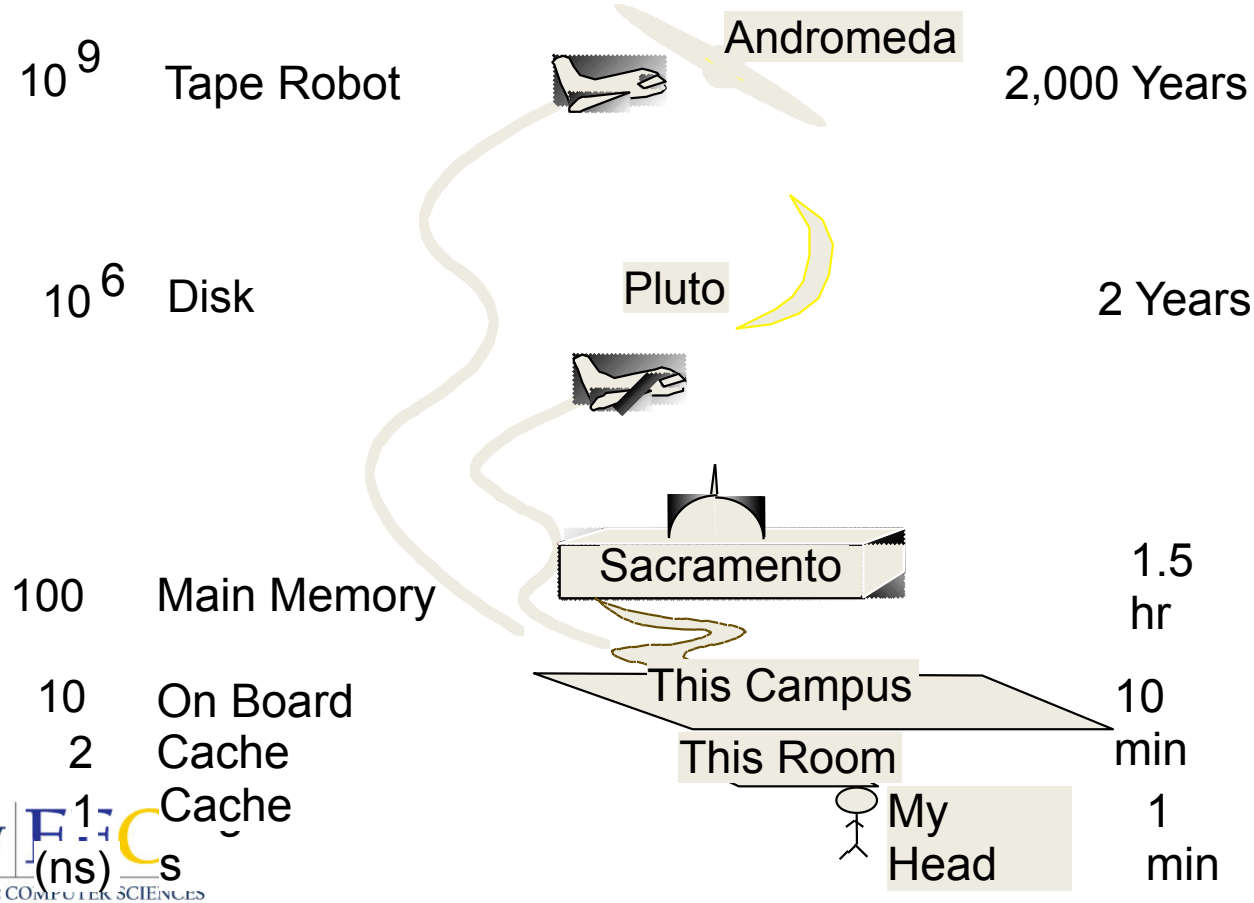
Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

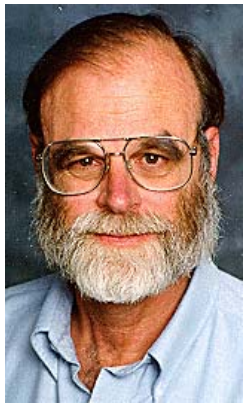
Interesting Times

- Moore's Law relied on the cost of transistors scaling down as technology scaled to smaller and smaller feature sizes.
  - And the resulting transistors resulted in increased single-task performance
- But single-task performance improvements hit a brick wall years ago...
- And now the newest, smallest fabrication processes <14nm, might have greater cost/transistor!!!! So, why shrink????

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

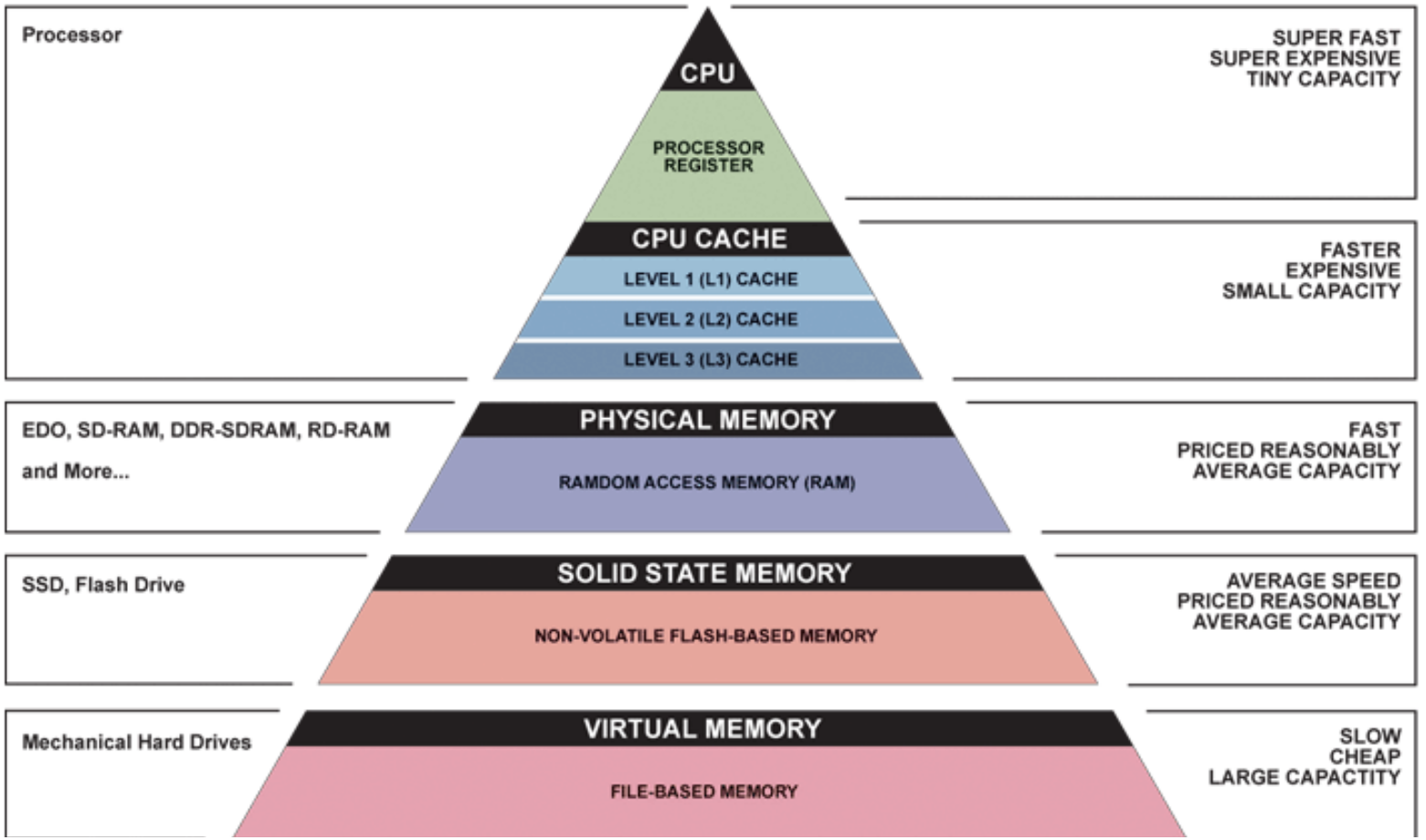
Jim Gray's Storage Latency Analogy: How Far Away is the Data?





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

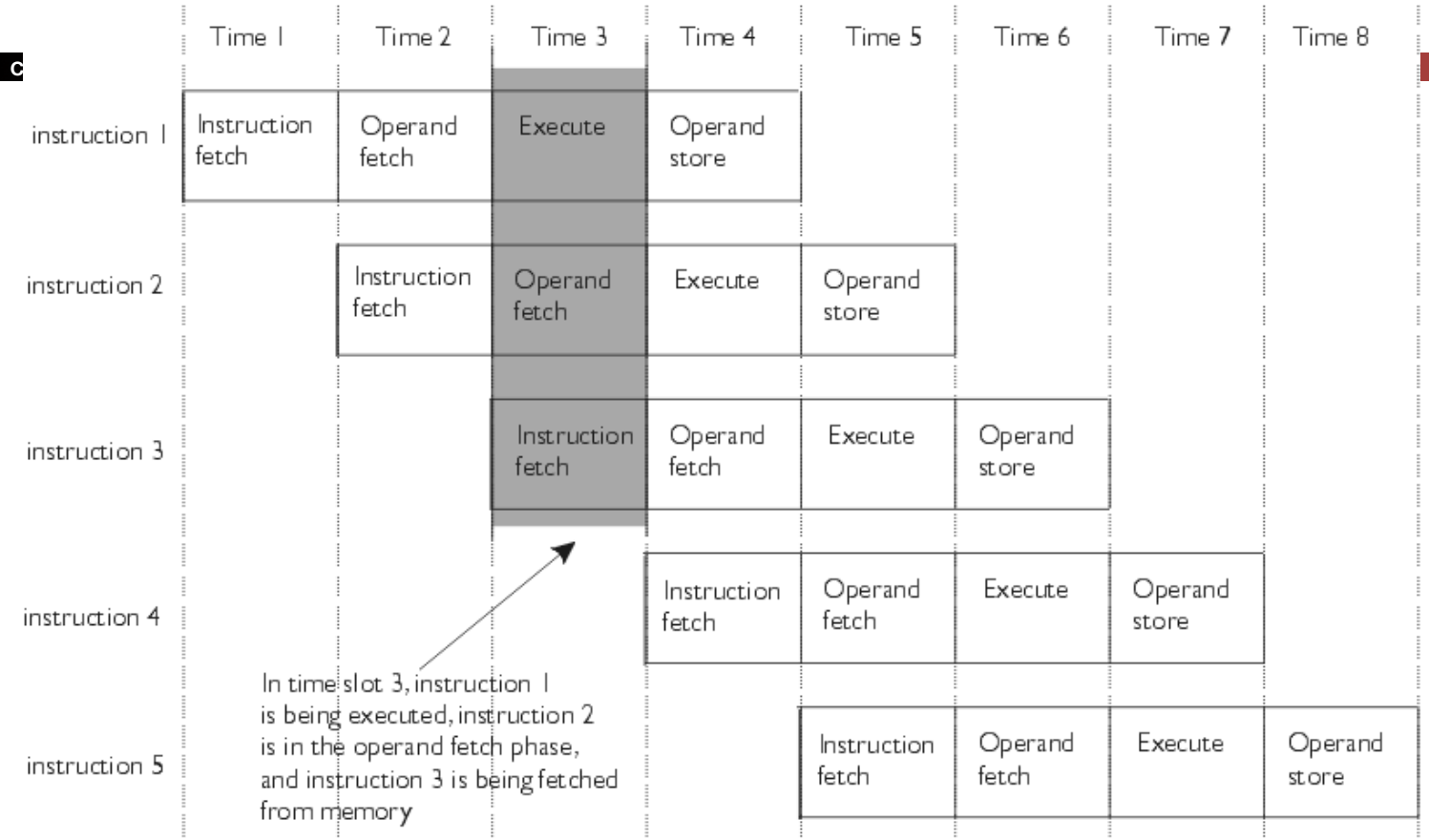
Great Idea #3: Principle of Locality/ Memory Hierarchy

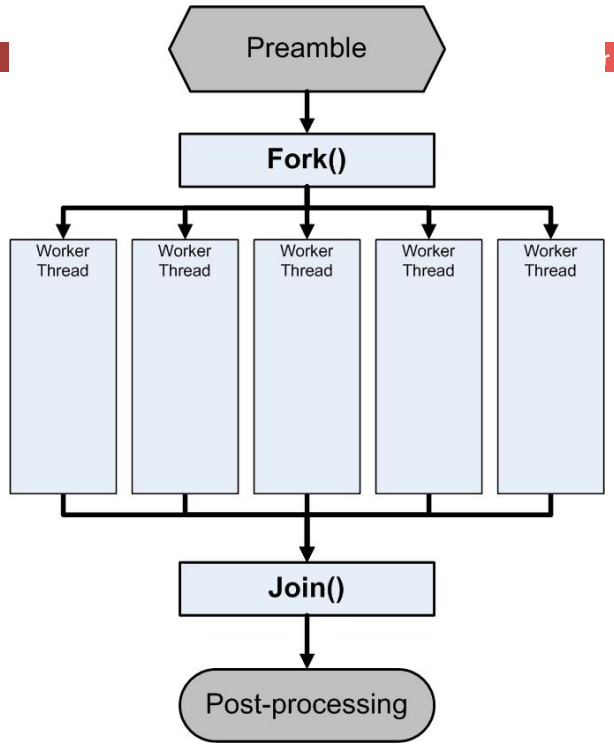


If your computer doesn't have an SSD, get one!

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

Great Idea #4: Parallelism





Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

The Caveat: Amdahl's Law

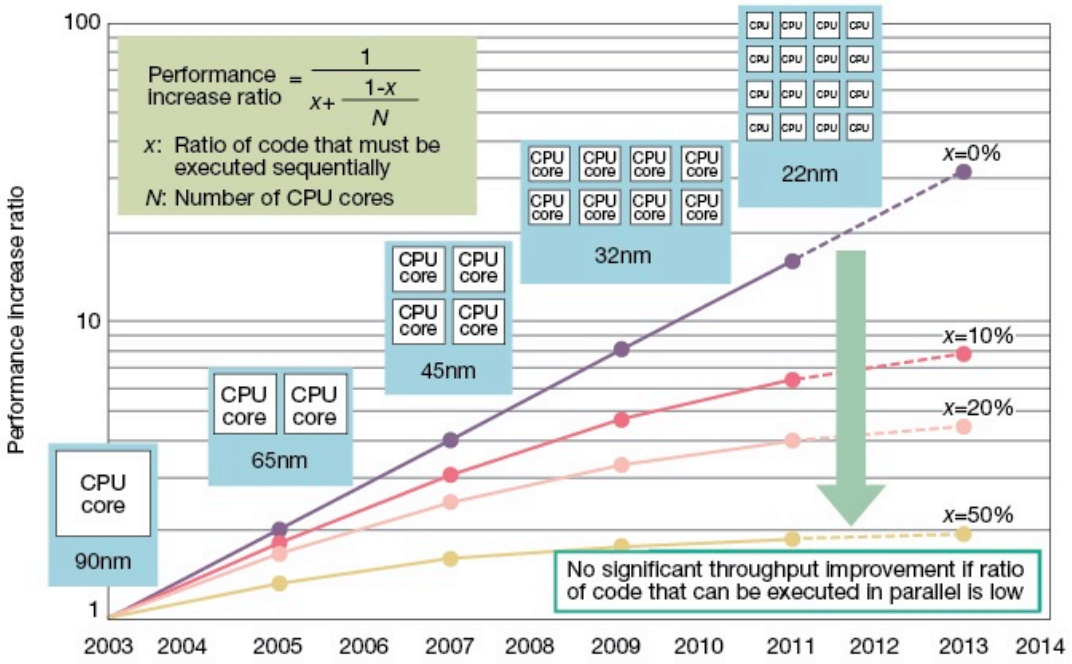


Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor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

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

Great Idea #5: Failures Happen, so...

- 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?

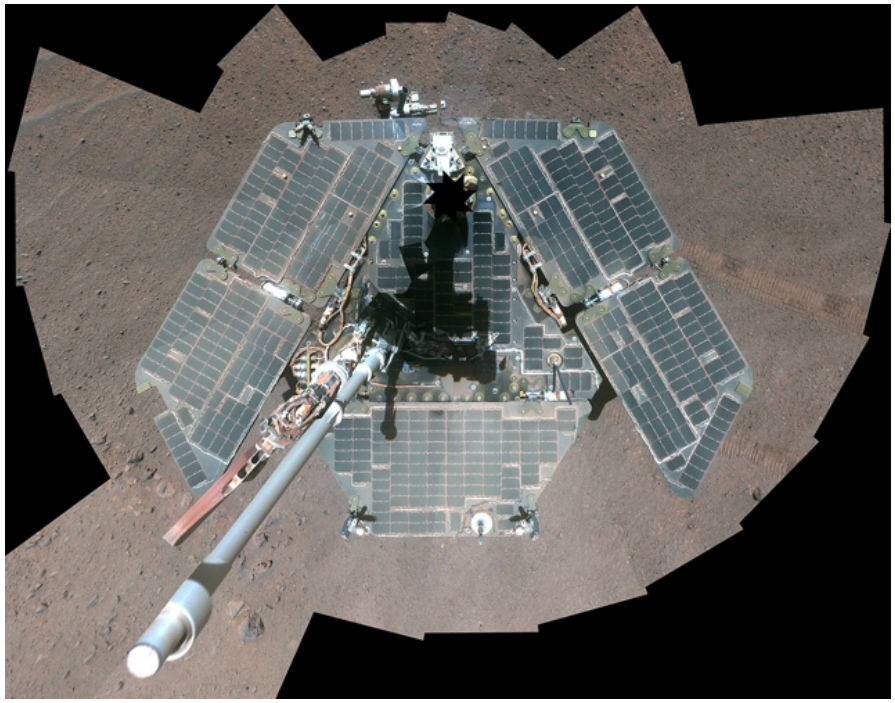
a) 1 / month
b) 1 / week
c) 1 / day
d) 1 / hour

50,000 x 4 = 200,000 disks
200,000 x 4% = 8000 disks fail
365 days x 24 hours = 8760 hours

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

NASA Fixing Rover's Flash Memory

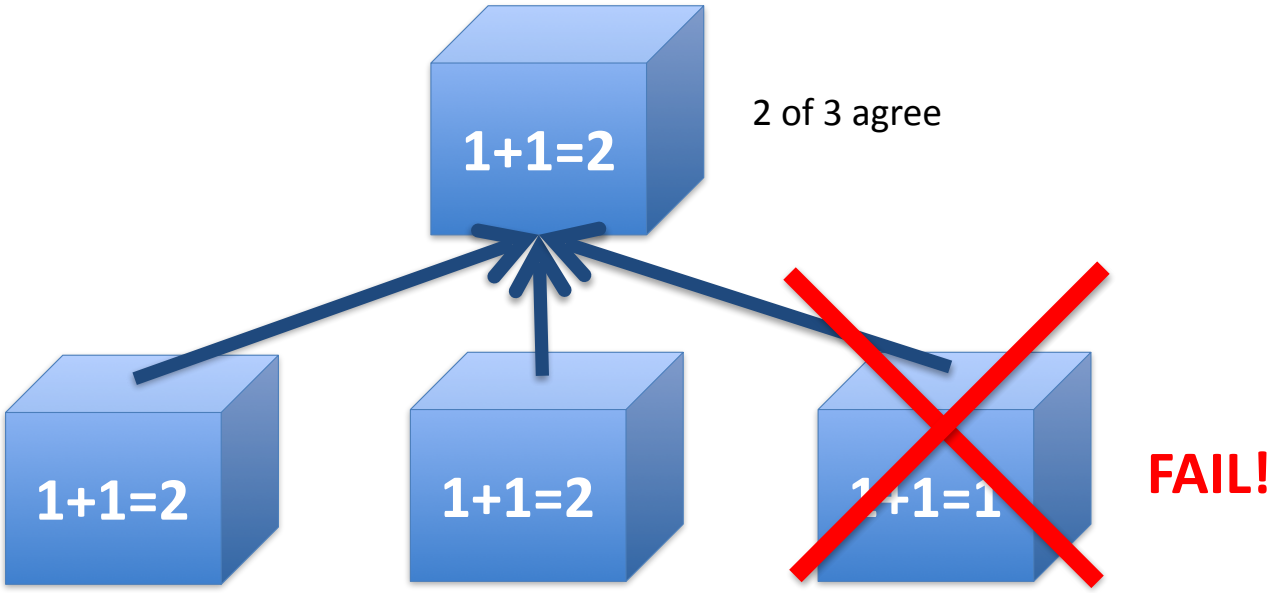
- Opportunity still active on Mars after >10 years
- But flash memory worn out
- New software update to avoid using worn out memory banks



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

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

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/"Chipkill" memory)
  - Redundant instructors so one of us can travel while the other teaches ;-)

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

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