CS61C: Great Ideas in Computer Architecture (a.k.a. Machine Structures)

Dr. Nick Weaver
Lecturer

- My primary specialty is Network Security and Network Measurement
  - Although a reformed hardware person, originally specializing in FPGAs, doing some of my own circuit board design these days…
- 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
- What does lecturer mean?

The Head TAs

Stephan Kaminsky

skaminsky115@ • https://skaminsky115.github.io/

Office Hours: M 10-11am Soda 283E&H

Hi! I'm a fourth year EECS Major and this is my fifth semester on staff for 61C. In my spare time, I like to ride motorcycles, go to movies with friends, eat avocado toast at acme bread company, and pet dogs and cats. Lets have fun this semester :)

Sayan Paul

sayanpaul@

Office Hours: W 1-3 Soda 283E&H

Hello! I'm a fourth year CS major from near Washington, DC. I spend my free time watching as much TV as possible, following the NBA (go Wizards!), and losing to my friends in Smash Ultimate. Looking forward to a great semester!

The TA Corps:

- Too many to list on slides:
  - See https://cs61c.org/staff/

Course Information

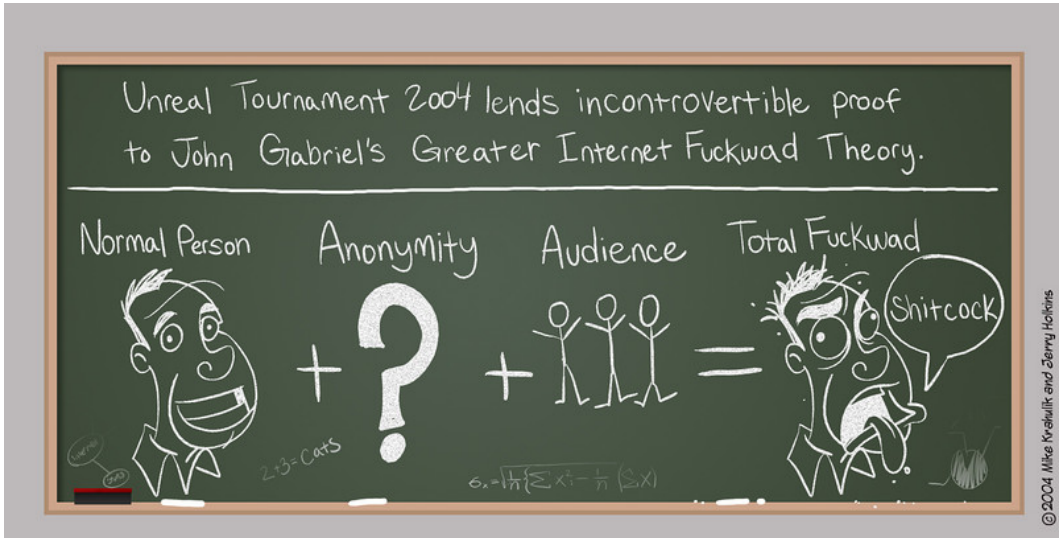
- Course Web: https://cs61c.org/
- Major tutoring support from CS-Mentors & paid tutors
- Textbooks: Average 15 pages of reading/week (can rent!)
  - Patterson & Hennessey, Computer Organization and Design, 5/e (RISC-V)
  - Kernighan & Ritchie, The C Programming Language, 2nd Edition
    - "ANSI" (old-school) C…
  - Barroso & Holzle, The Datacenter as a Computer, 2nd Edition
- Sign up for discussion & lab using Signup Genius
  - You can check off in any lab, but those signed up have priority

Reducing The Workload...

- I've apparently gotten a reputation for running this class a bit higher on the workload than others...
  - We are fixing that this semester:
    Workload is designed to be equivalent to Fall 2019
- Just some of the changes from previous semesters
  - Drop concurrency project altogether
    - Same # of actual projects as Fall 2019, just we call project 4 a project rather than homework so you ***can*** use slip days...
  - Go back to tested & easier projects elsewhere:
    No more "Lets turn 164 into a class project...  "
  - And the head TAs will keep me in line!

Piazza

- 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
- Use Piazza, not email, to make requests of course staff
  - Piazza scales much better and allows us to keep track of things easier
- If you aren't enrolled but want to join the Piazza, see the policy on the web site for how to be added



Gradescope, Class Accounts, Concurrent Enrollment…

- We will use Gradescope for all assignments
  - Midterms/Final with a regrade window
  - Homeworks
  - Lab checkoffs
- Class accounts:
  - https://acropolis.cs.berkeley.edu/~account/webacct/
- Concurrent enrollment:
  - You need to fill out the form at https://forms.gle/1gXPJMbdnaodz19w8 to get class accounts/Piazza access
  - You should be let in, but the department doesn't process the applications until the third week

Scaling the class...

- I may be House Slytherin... But there is still a lot of Hufflepuff:
  - "I will teach them all and treat them all the same"
- We want to scale to support every student who wants to take this class
  - Lectures will be webcast ***for later review***
    - Why the iClicker is one dimension of EPA:
      We find that in-person attendance is useful
  - Attend ***any*** discussion & lab section (but sign up for a section for priority)
    - Lab checkoffs designed to be super-fast
  - Lots of TA & Tutoring Staff
  - Use Piazza for questions! DO NOT USE EMAIL!

Tried-and-True Technique: Peer Instruction/iclickers

- 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
  - The WiFi is generally not good enough for REEF soft-clickers: I can probably give a whole lecture on why it sucks…

Class Grading Policy:
Fixed Bins or Curves, Whichever Is Better!

- We have a set of fixed bins already defined
  - These bins are ***guaranteed***: If you are in a bin, you will get ***at least*** that grade
- I have a departmental target (2.8-3.3 GPA, and I bias towards the high end)
  - http://www.eecs.berkeley.edu/Policies/ugrad.grading.shtml
- If bins result in above the target gpa... 🤷‍♂️
  - If students look to be doing better than expected… 🤷‍♂️
    I have smart students and don't need to make later assignments harder
- If bins are below my target gpa... ***then*** I will curve the class

EPA!

- **E**ffort
  - Attending prof and TA office hours, completing all assignments, turning in HW, doing reading quizzes
- **P**articipation
  - Attending lecture and voting using the clickers
  - Asking great questions in discussion and lecture and making it more interactive
- **A**ltruism
  - 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, and not used when setting a curve if needed)

Late Policy Projects…
Slip Days on projects!

- Assignments due at 11:59:59 PM PT
- You have 3 slip day tokens (NOT hour or min) that apply to projects
- Every day your project is late (even by a ***millisecond***) we deduct a token
- After you've used up all tokens, it's 33.33333334% 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***!
- Grade in the end will use the optimal distribution of slip days between all your projects

Labs & Homework...

- Labs are a key portion of the class
  - We will [attempt to] release the lab the Wednesday before
    - So you can get stuff done before lab itself
  - Labs are due by the last ***lab section*** that lab day
    - If you check off the next week you will get 1/2 credit
  - ***Highly recommended*** to have a partner
  - The lab autograder doesn't give you course points!
    - It is designed to speed up the checkoff process
  - You have 1 "lab drop"
- Homeworks on Gradescope
  - No late credit, unlimited attempts with feedback
- Lab and discussions start next week

DSP...

- We are happy to accommodate you, but we need to know
- So DSP students, please get your accommodations in now so we can schedule exams etc...

Use Git and Push Often…

- You will be using GitHub to host your projects for submission…
  - So use it for your normal workflow too
- Push your work back on a regular basis
  - It really prevents screwups:
    "Ooops, go back" is the reason for version control
    - Your computer should be able to ***blow up***, and you should only lose a couple hours work!
  - It gives a timestamp we can trust 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

Debugging...

- We are all very helpful during office hours...
  - But we will ***not*** simply debug your project for you
- In order to receive project assistance you ***must***:
  - Have a test case which shows the problem
  - Have the debugger running and at a breakpoint that shows the problem
  - If it is a memory problem (segfault etc) you must also have the project running in valgrind to indicate where the problem is

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***

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 I (mostly) assume rational attackers:
    Benefit of attack > ***Expected*** cost
    - Cost of 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"
  - I will go to DEFCON 1 (aka "launch the nukes") ***immediately***
    - “Nick doesn’t make threats. ***He keeps promises***”

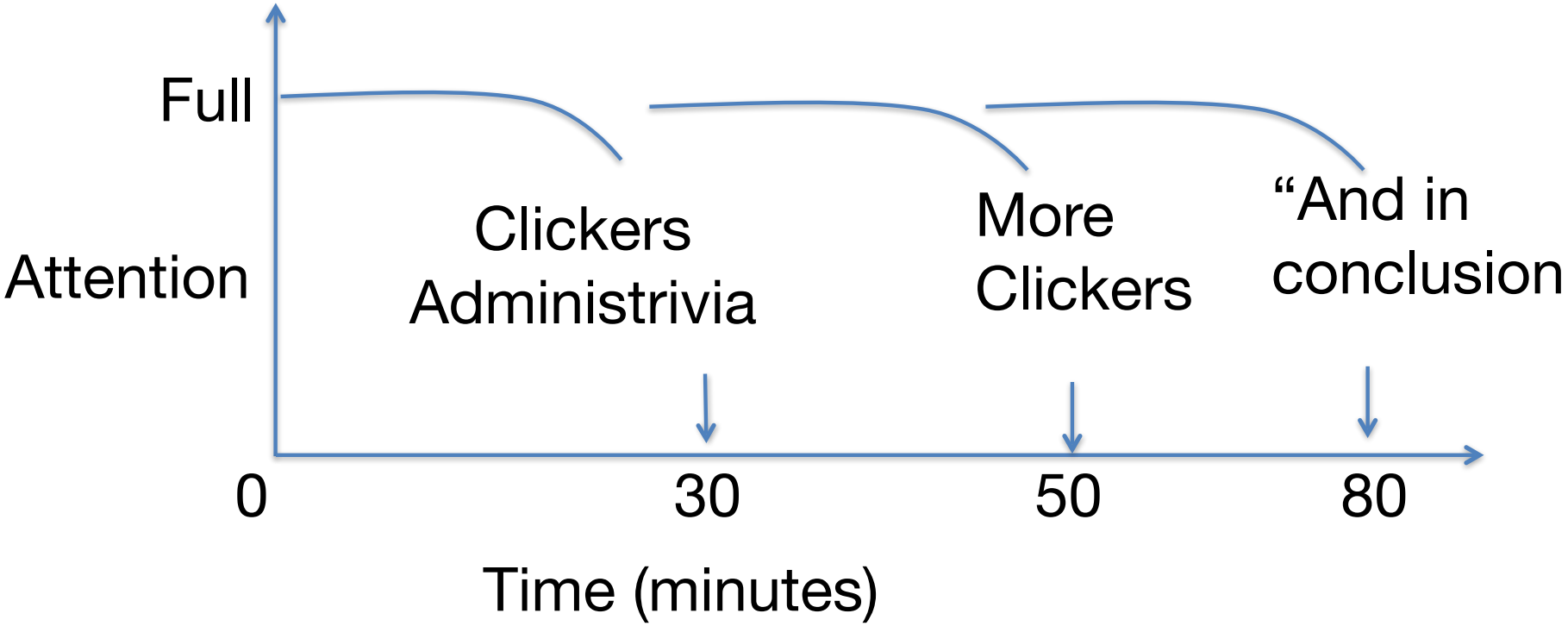
More On Academic Dishonesty...

- I take cheating ***personally***
  - I have an obligation to protect the value of this University for honest students: A critical threshold of cheating hurts everybody else
- Minimum penalty is ***negative*** points:
  - If I have just a 50% chance of catching you, you are provably better off just not doing the work. It is not rational to cheat in my classes
  - And penalties can go up from there: I can and have given Fs
- I also handle cheating cases personally
  - "I ought to of shot that dog myself, George. I shouldn't ought to of let no stranger shoot my dog." -John Steinbeck, ***Of Mice and Men***

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
- If you feel overwhelmed, please use the resources available
  - Academically: Ask on Piazza, Tutoring, Office hours, Guerrilla sections, etc...
    - We have lots and lots and lots of different ways for you to get help: Find the one that works for you!
    - Partially how we have scaled is ***not*** turning **O(*n*)** to **O(*lg(n)*)** but allowing us to scale up the TA/tutor staff!
  - Non-Academic: Take advantage of University Health Services if you need to
    - ***Nick did!*** Zoloft (an antidepressant) and therapy saved my life, twice.

Architecture of a typical Lecture



Agenda

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

Agenda

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

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...
  - Allows us to talk about key hardware features in higher level terms
  - Allows programmer to explicitly harness underlying hardware parallelism for high performance
  - Only language which comes close is Rust…
- Also allows programmer to shoot oneself in the foot in ***amazingly*** spectacular ways
  - One of my personal goals in this class is for you to develop a ***rational hatred*** of C

Other Pedagogical Choices In This Class...

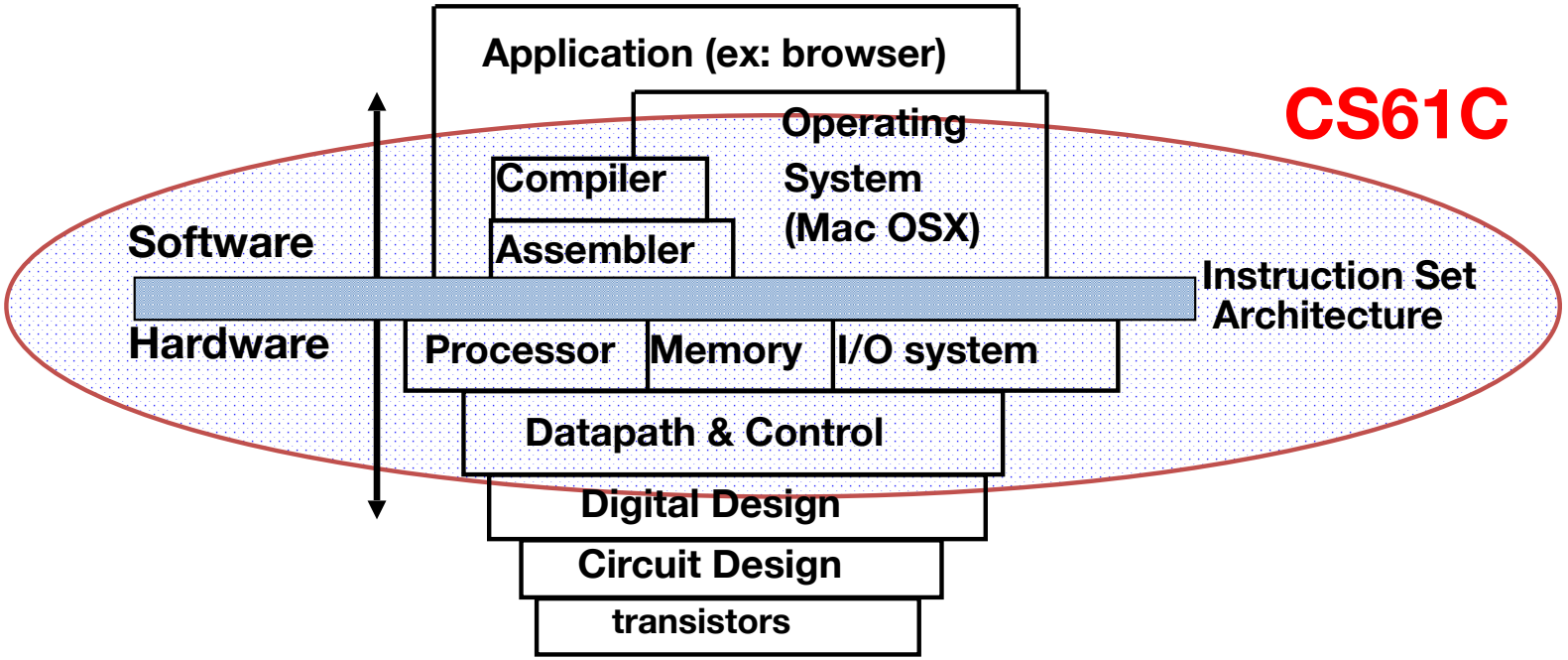
- Our assembly language is RISC-V...
  - If you ever have to program in assembly, it will probably be x86 or 32/64b ARM
    - Those have actually “won” in the marketplace:
      RISC is more efficient but strap on enough rockets and a pig will fly…
      And Intel strapped that pig to a Saturn V…
  - But RISC processors are much simpler, so we use a RISC (Reduced Instruction Set Computer)
    - And why RISC-V? "All RISC processors are effectively the same except for one or two bad design decisions that 'seemed like a good idea at the time'", and RISC-V is new enough that we don't know of any that only ***seemed*** like a good idea...
- Our hardware design is in Logisim (schematics)
  - If you ever do hardware design, you'll only use schematics for circuit boards, everything else is Verilog or VHDL...
    - But its harder to get up to speed on those

Modern 61C:
From the small...

To the Big...

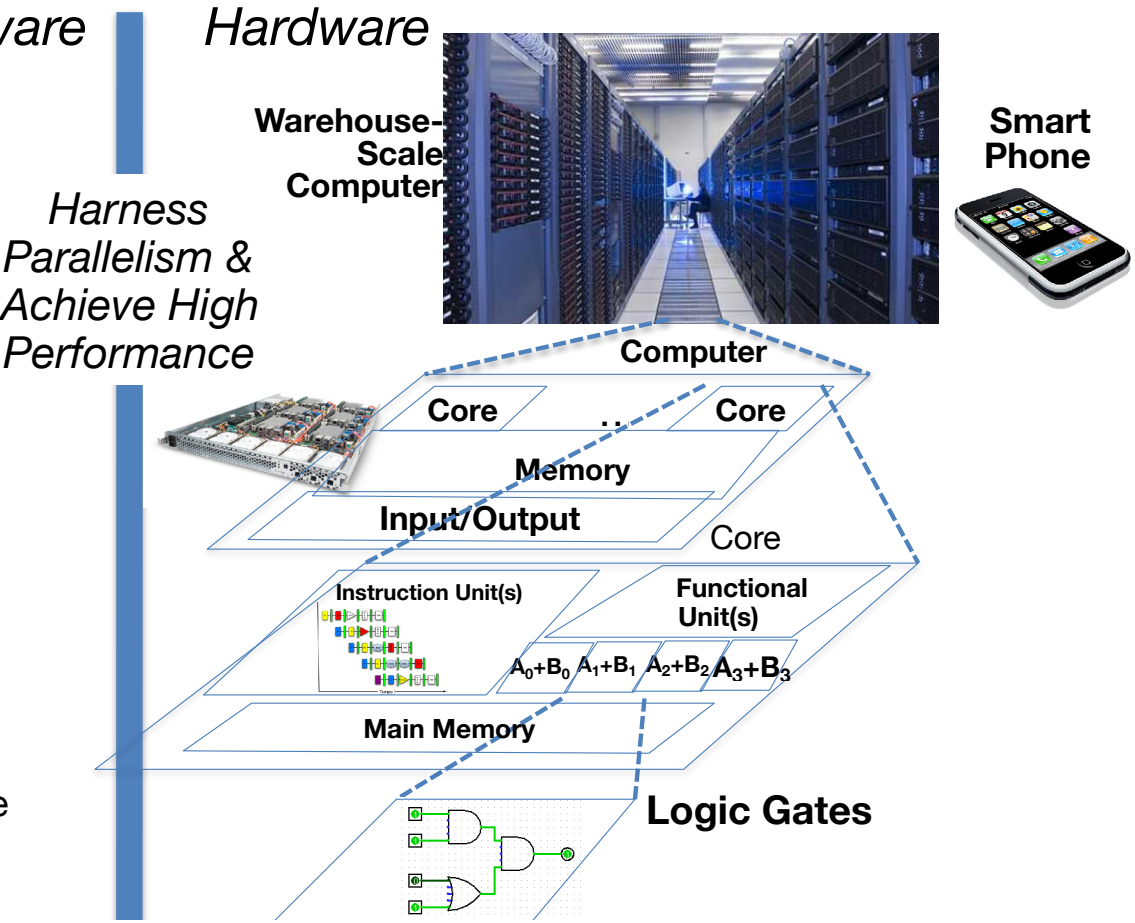

warehouse-scale
computer
My other computer
is x data center
YOUR

Old School Machine Structures



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

- Parallel Requests
  Assigned to computer
  e.g., Search “giant military 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

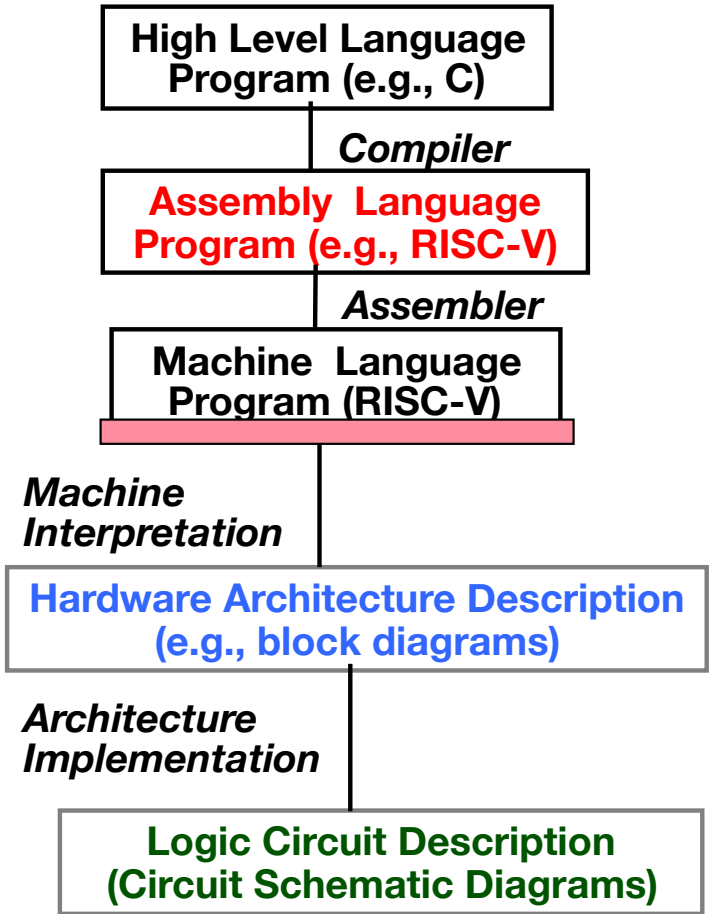


5 Great Ideas in Computer Architecture

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)

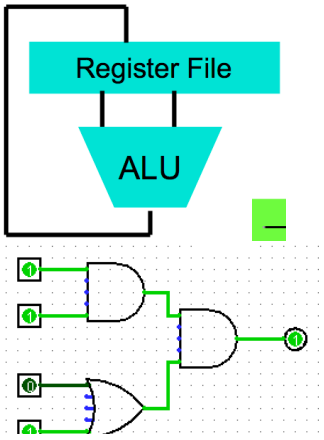

```
lw   t0, t2, 0
lw   t1, t2, 4
sw   t1, t2, 0
sw   t0, t2, 4
```



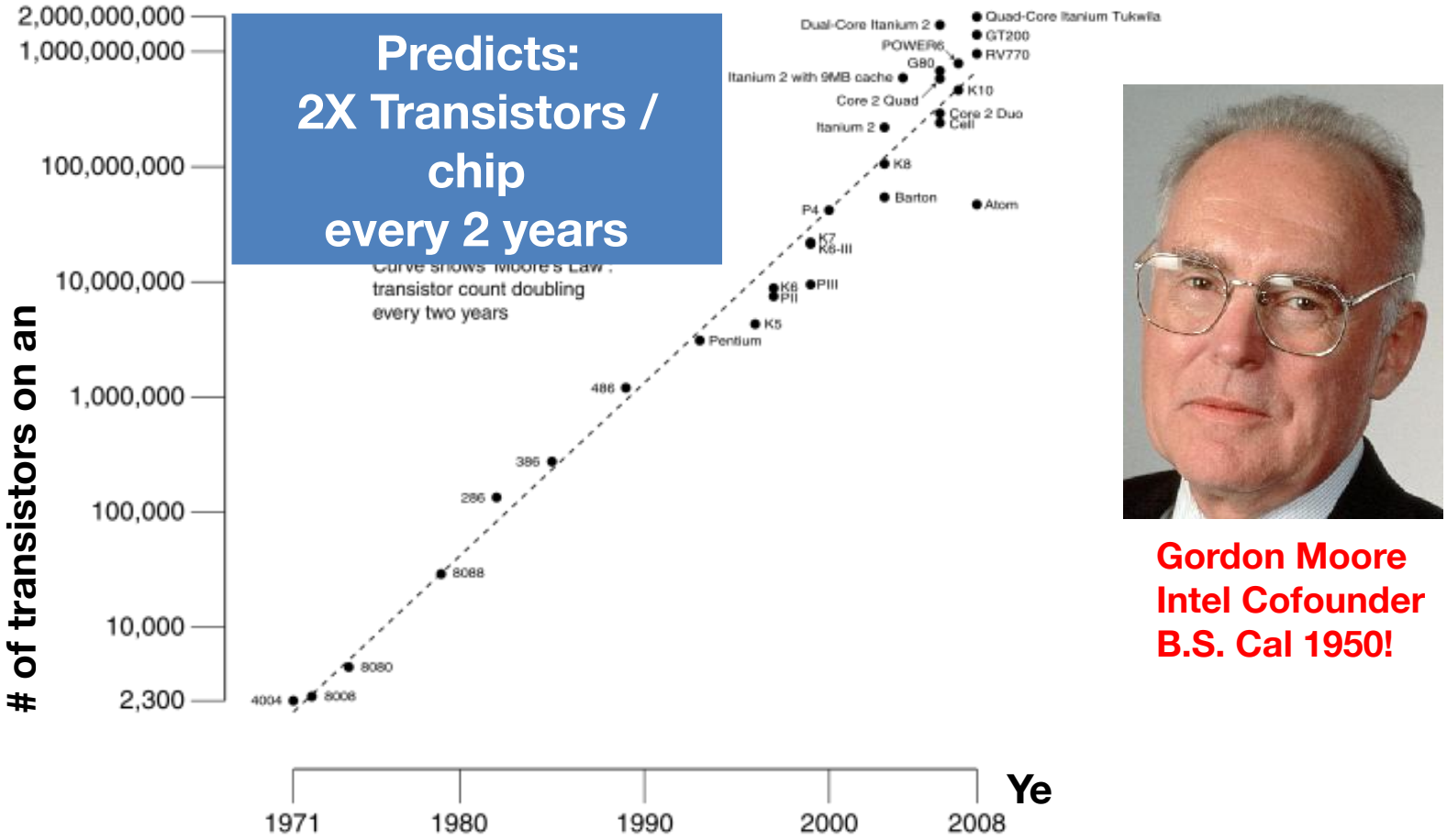

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

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
```



#2: Moore's Law

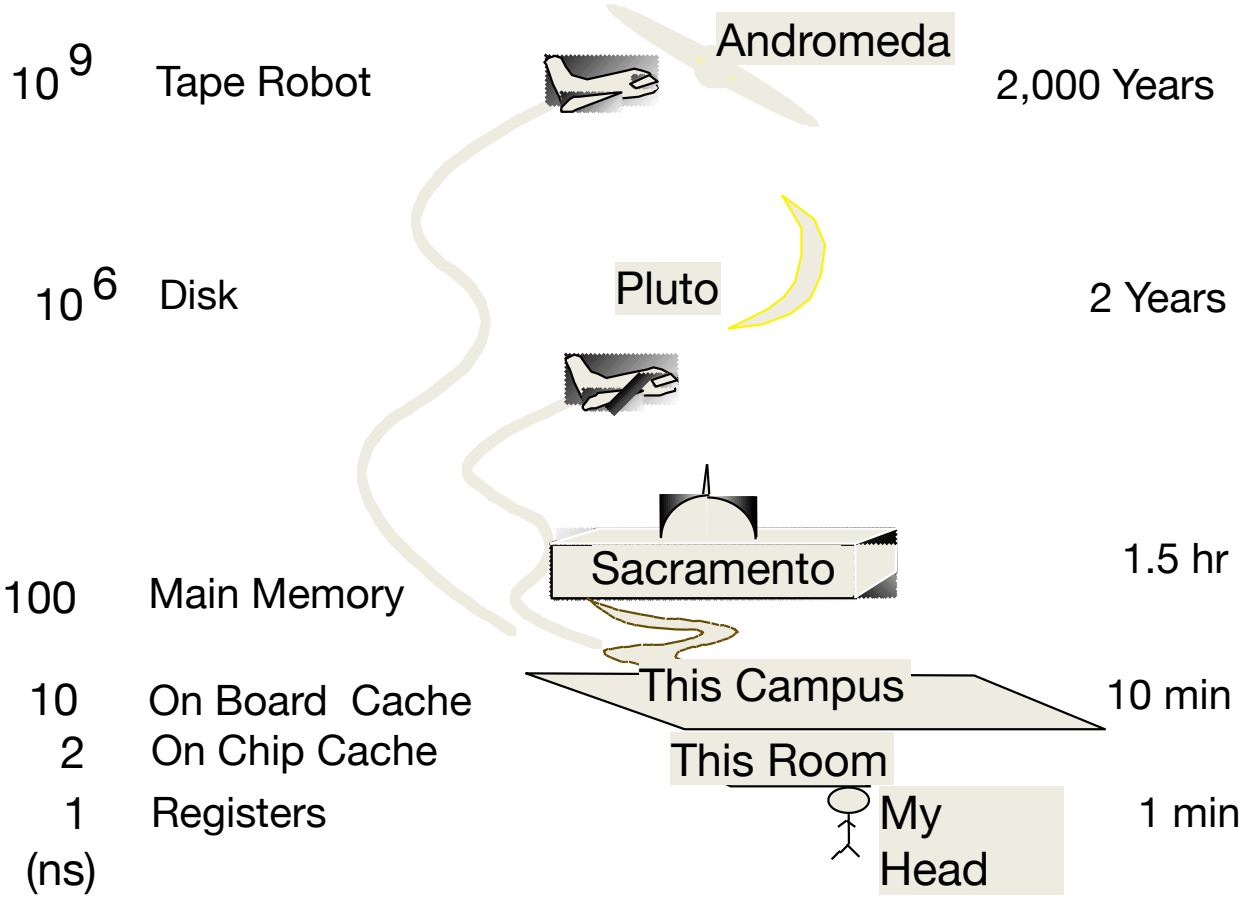


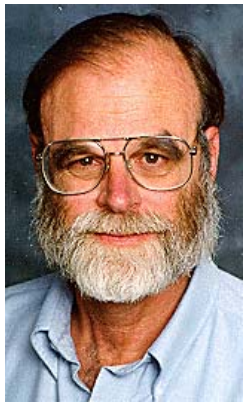
Gordon Moore
Intel Cofounder
B.S. Cal 1950!

Interesting Times

- Moore's Law meant that the cost of transistors scaled 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????

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

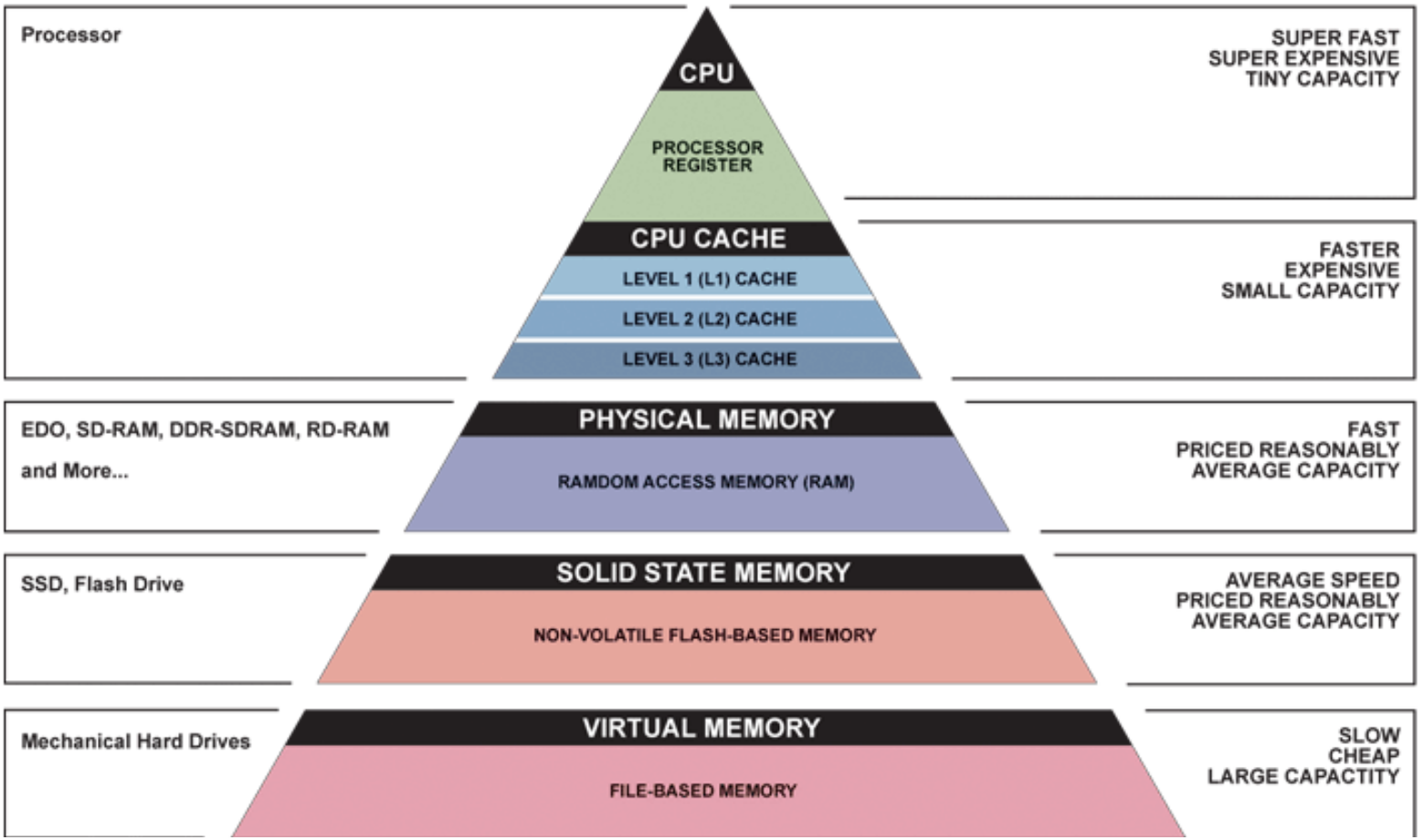




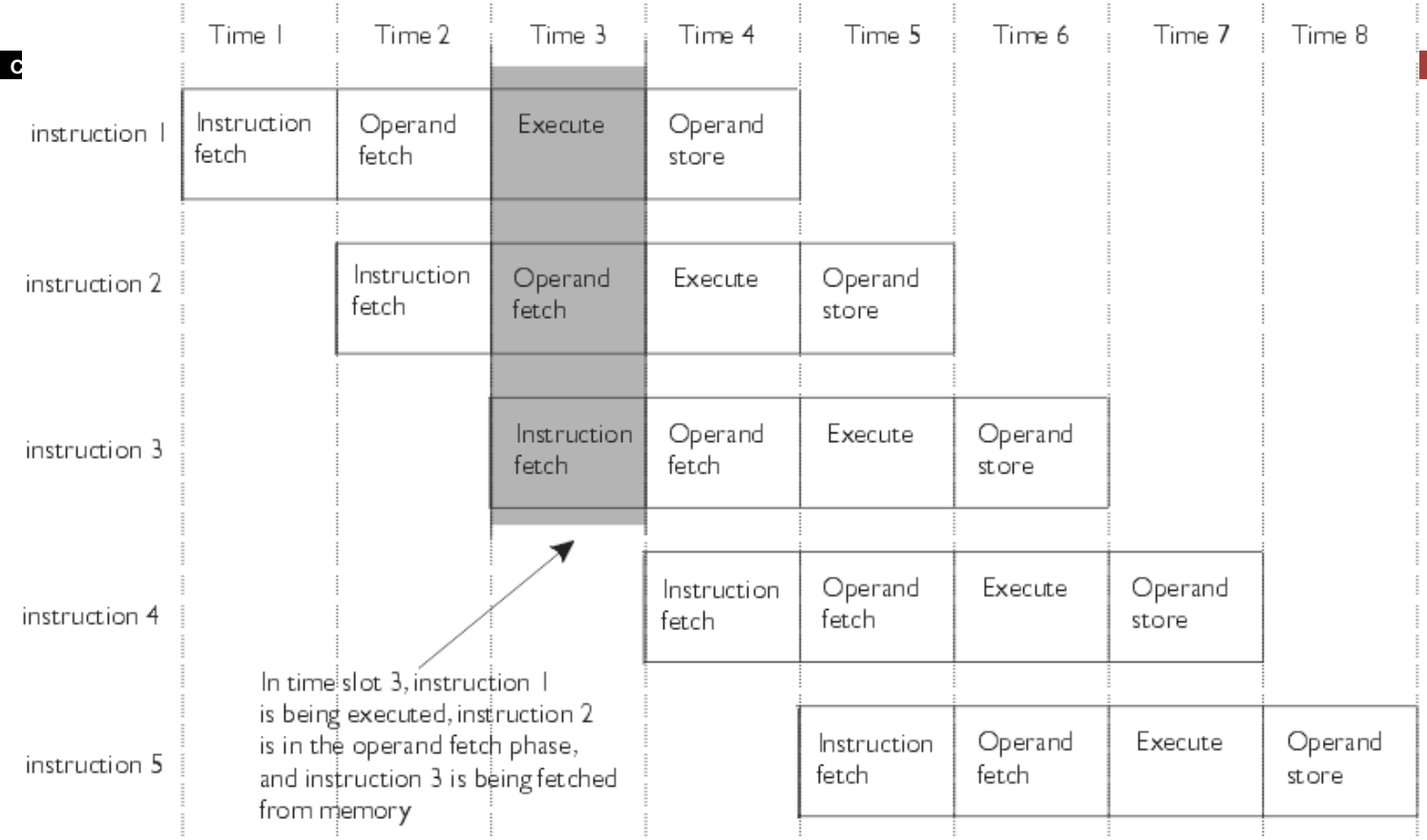
Jim Gray
Turing Award
B.S. Cal 1966
Ph.D. Cal 1969!

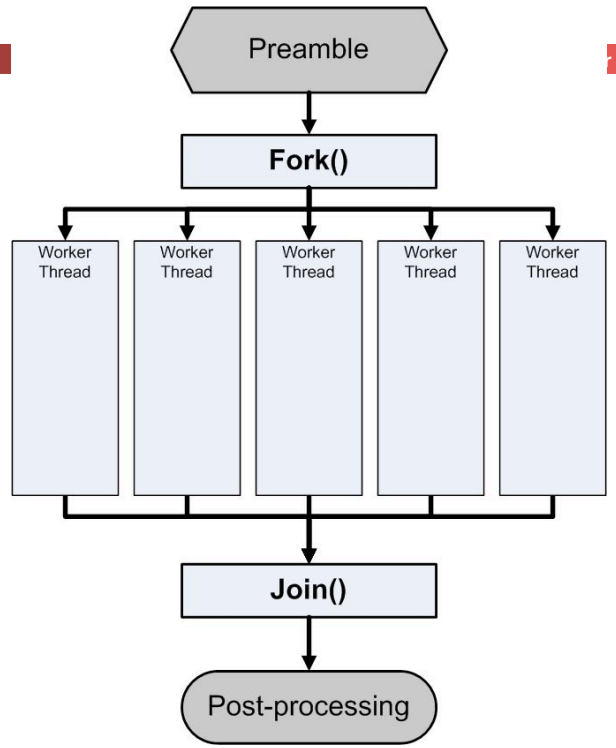
Great Idea #3: Principle of Locality/ Memory Hierarchy

*Memory Hierarchy effectively creates a **large fast cheap** memory.*



Great Idea #4: Parallelism





And Parallelism is ***Everywhere***

- Raspberry Pi 4 B+
  - Quad core processor at 1.5 GHz
    - Each core is 3-issue, out-of-order superscalar
  - Plus GPU, 1-4 GB RAM, 2x USB3, 2x USB2, Gigabit Ethernet, 2x HDMI...
  - $35-55
    - Nick is working on a board to turn one of these to power a fully autonomous, vision-guided drone
- Compare with a Cray-1 from 1975:
  - 8 MB RAM, 80 MHz processor, 300MB storage, $5M+
- Or modern high end servers:
  - You can get a 2u server which supports 4 processors
    - And each processor can have 20+ cores, so 80 processor cores!

The Caveat: Amdahl's Law

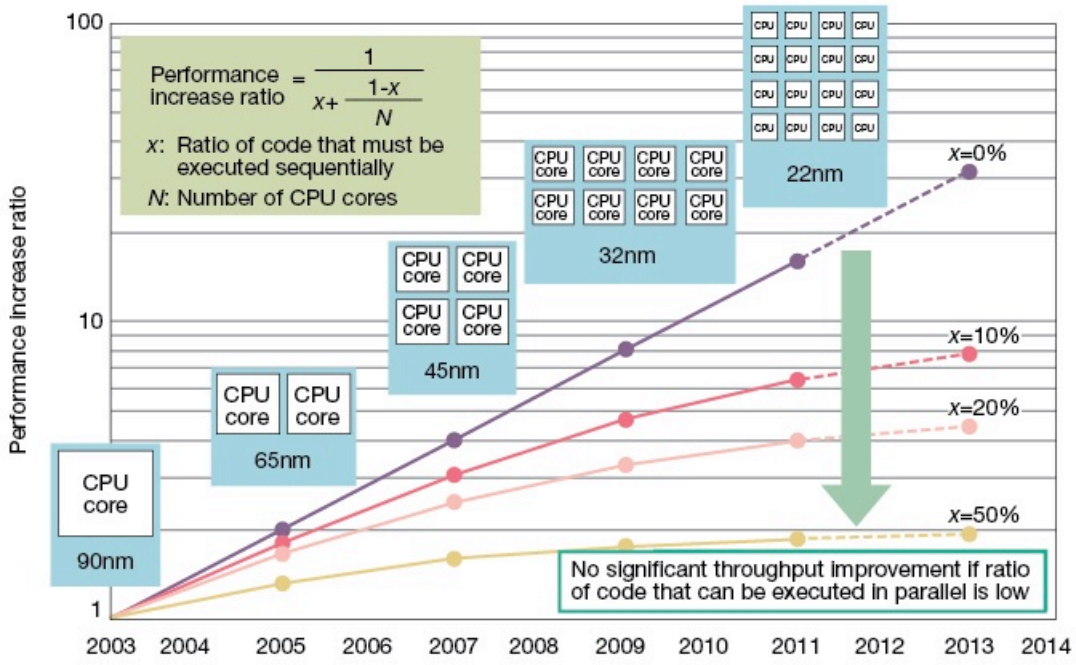


Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor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

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

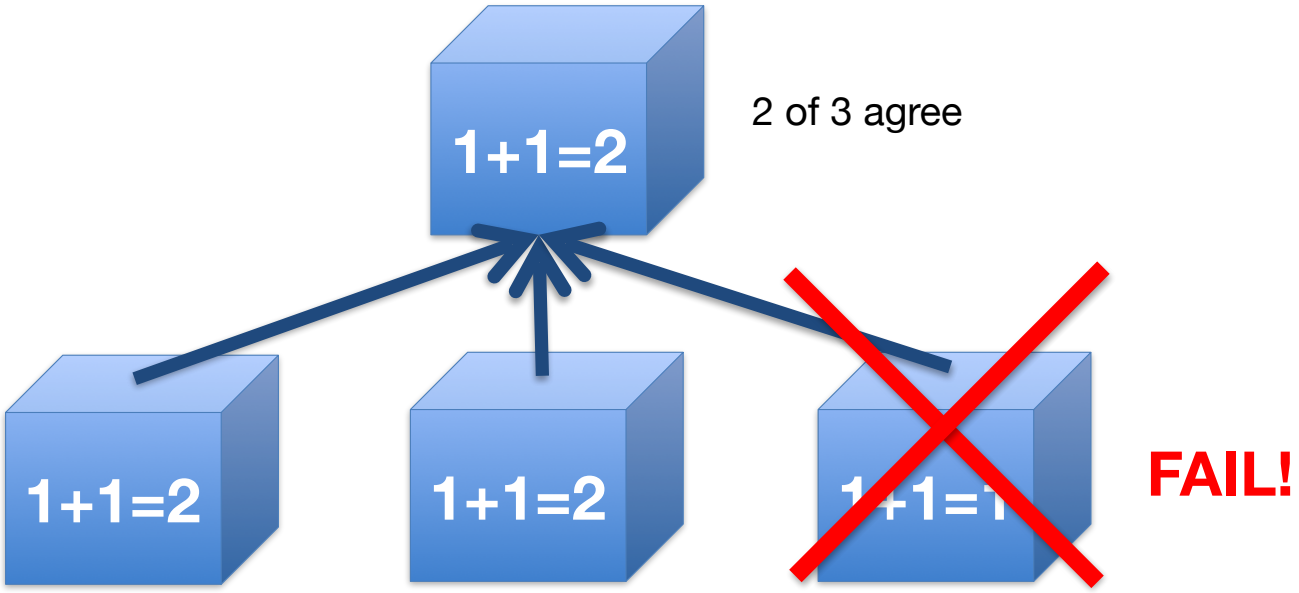
NASA Fixing Rover's Flash Memory

- "Opportunity" (JPL/NASA mars rover) was active on Mars for 14 years
  - Exceeded target lifespan by 5500% !
- But flash memory worn out
- Deployed a software update to avoid using worn out memory banks
  - And then kept on driving across the Martian landscape…

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** so that can lose 1 bit but no data (Error Correcting Code/ECC Memory/"Chipkill" memory)
  - **Redundant instructors** (I wish... I'd want to call in a "hot spare" if I have to travel!)

Summary

- CS61C: Learn 5 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. Dependability via Redundancy

Within the Computer: Everything is a Number.

- But numbers usually stored with a fixed size
  - 8-bit bytes, 16-bit half words, 32-bit words, 64-bit double words, …
  - And there are really only two primitive "numbers": 0 and 1 is a "bit" (BInary digiT). A byte is 8 bits
- Integer and floating-point operations can lead to results too big/small to store within their representations: *overflow/ underflow*

Number Representation

- Value of i-th digit is $d \times Base^i$ where i starts at 0 and increases from right to left:
- $123_{10} = 1_{10} \times 10_{10}^2 + 2_{10} \times 10_{10}^1 + 3_{10} \times 10_{10}^0$

  $= 1\text{x}100_{10} + 2\text{x}10_{10} + 3\text{x}1_{10}$

  $= 100_{10} + 20_{10} + 3_{10}$

  $= 123_{10}$
- Binary (Base 2), Octal (Base 8), Hexadecimal (Base 16), Decimal (Base 10) different ways to represent an integer
  - We'll use 1_{two}, 4_{eight}, 5_{ten}, 10_{hex} to be clearer

    (vs. 1_2, 4_8, 5_{10}, 10_{16})
  - And in song: https://www.youtube.com/watch?v=UIKGV2cTgqA

Number Representation

- Hexadecimal digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- $FFF_{hex} = 15_{ten} \times 16_{ten}^2 + 15_{ten} \times 16_{ten}^1 + 15_{ten} \times 16_{ten}^0$
  $= 3840_{ten} + 240_{ten} + 15_{ten}$
  $= 4095_{ten}$
- $1111\ 1111\ 1111_{two} = FFF_{hex} = 4095_{ten}$

$= 1x2^{11} + 1x2^{10} + 1x2^8 + 1x2^7 + 1x2^6 + 1x2^5 + 1x2^4 + 1x2^3 + 1x2^2 + 1x2^1 + 1x2^0$

$= 2048_{ten} + 1024_{ten} + 512_{ten} + 256_{ten} + 128_{ten} + 64_{ten} + 32_{ten} + 16_{ten} + 8_{ten} + 4_{ten} + 2_{ten} + 1_{ten}$
$= 4095_{ten}$

(May put blanks every group of binary or hexadecimal digits to make it easier to parse, like commas in decimal)

Signed and Unsigned Integers

- C, C++, and Java have *signed integers*, e.g., 7, -255:

```
int x, y, z;
```

- C, C++ also have *unsigned integers*, which are used for addresses
- 32-bit word can represent 2^{32} binary numbers
- Unsigned integers in 32 bit word represent 0 to 2^{32}-1 (4,294,967,295)

Unsigned Integers

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 1_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2_{ten}$

... ...

$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 2{,}147{,}483{,}645_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 2{,}147{,}483{,}646_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 2{,}147{,}483{,}647_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 2{,}147{,}483{,}648_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 2{,}147{,}483{,}649_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2{,}147{,}483{,}650_{ten}$

... ...

$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 4{,}294{,}967{,}293_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 4{,}294{,}967{,}294_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 4{,}294{,}967{,}295_{ten}$

Signed Integers and Two's-Complement Representation

- Signed integers in C; want ½ numbers <0, want ½ numbers >0, and want just a single 0
- Two's complement treats 0 as positive, so 32-bit word represents 2^{32} integers from
  -2^{31} (–2,147,483,648) to 2^{31-1} (2,147,483,647)
  - Note: one negative number with no positive version
  - Book lists some other options:
    - All of which are worse except in very limited circumstances
  - Every computer uses two's complement today
- Most-significant bit (leftmost) is the sign bit, since 0 means positive (including 0), 1 means negative
  - Bit 31 is most significant, bit 0 is least significant

Two's-Complement Integers

Sign Bit

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = 1_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = 2_{ten}$
... ...
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = 2{,}147{,}483{,}645_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = 2{,}147{,}483{,}646_{ten}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = 2{,}147{,}483{,}647_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = -2{,}147{,}483{,}648_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = -2{,}147{,}483{,}647_{ten}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = -2{,}147{,}483{,}646_{ten}$
... ...
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = -3_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = -2_{ten}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = -1_{ten}$

Ways to Make Two's Complement

- *In two's complement the sign-bit has negative weight:*
- So the value of an N-bit word $[b_{N-1}\ b_{N-2}\ \ldots\ b_1\ b_0]$ is:

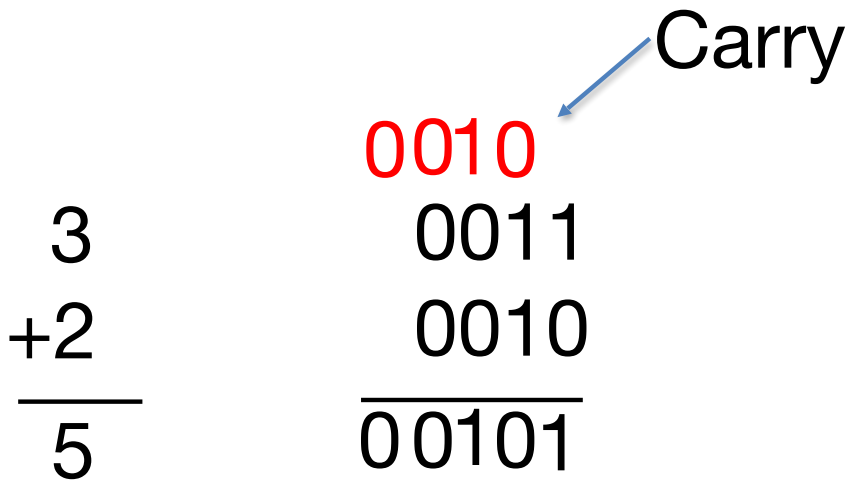
$$-2^{N-1} \times b^{N-1} + 2^{N-2} \times b^{N-2} + \ldots + 2^1 \times b^1 + 2^0 \times b^0$$

- For a 4-bit number, $3_{ten}=0011_{two}$, its two's complement $-3_{ten} = 1101_{two}$ $(-1000_{two} + 0101_{two} = -8_{ten} + 5_{ten})$

- Here is an easier way:
  - Invert all bits and add 1
  - Computers circuits do it like this, too

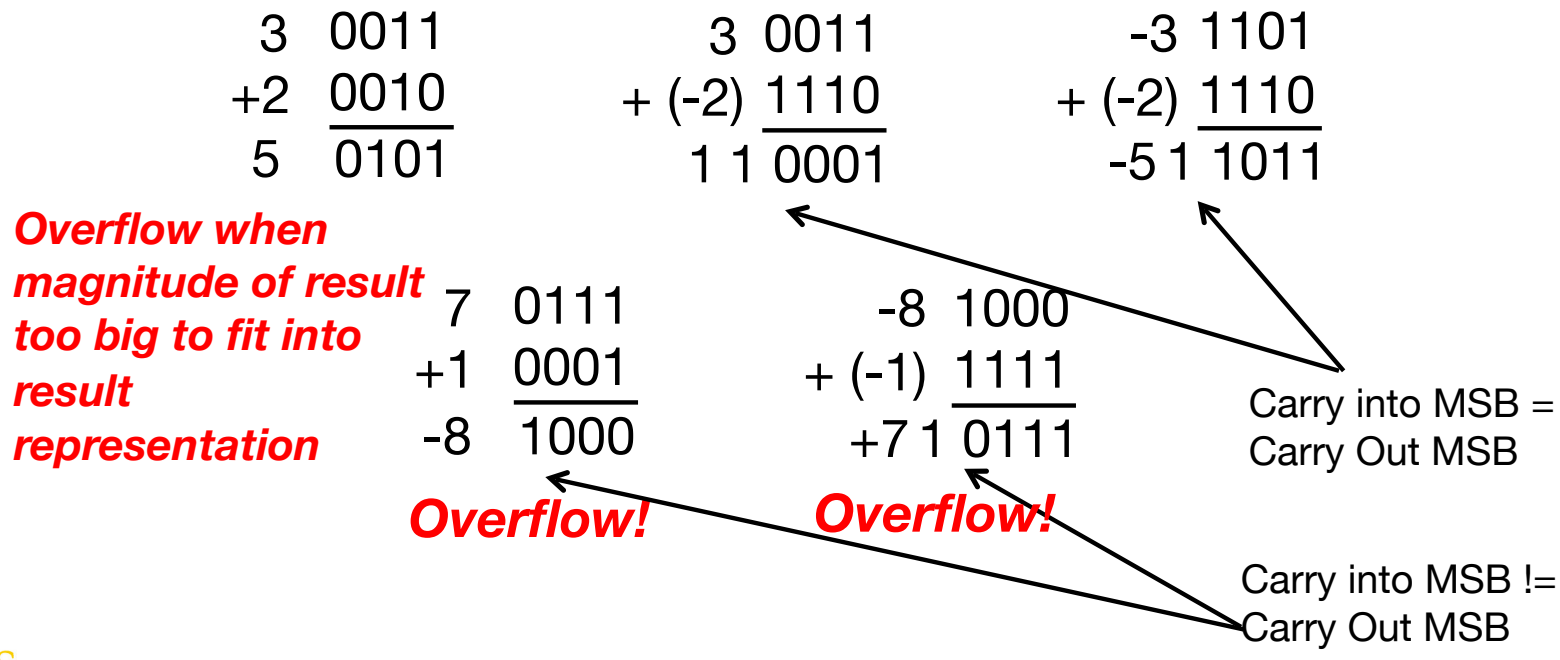
$$\begin{array}{rr} 3_{ten} & 0011_{two} \\ \text{Bitwise Invert} & 1100_{two} \\ + & 1_{two} \\ \hline -3_{ten} & 1101_{two} \end{array}$$

Binary Addition Example



Two's-Complement Examples

- Assume for simplicity 4 bit width, -8 to +7 represented



Suppose we had a 5-bit word.
What integers can be represented in two's complement?

- -32 to +31
- 0 to +31
- -16 to +15
- -15 to +16

Suppose we had a 5-bit word.
What integers can be represented in two's complement?

- ☐ -32 to +31
- ☐ 0 to +31
- ☐ -16 to +15
- ☐ -15 to +16

Summary: Number Representations

- Everything in a computer is a number, in fact only 0 and 1.
- Integers are interpreted by adhering to fixed length
- Negative numbers are represented with Two's complement
  - Overflows can be detected utilizing the carry bit
- We will get into some more representations later when we talk about floating point
  - Sign Magnitude & Biased representations are needed in floating point for specific uses
  - Not going to talk about '1s complement', its a joke that nobody uses