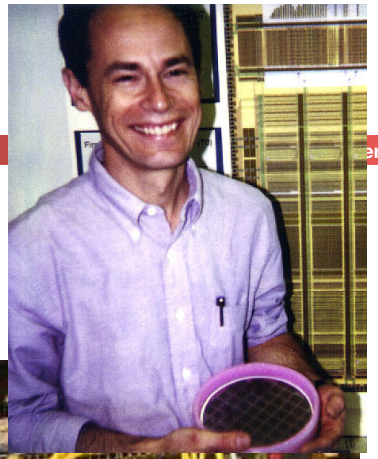
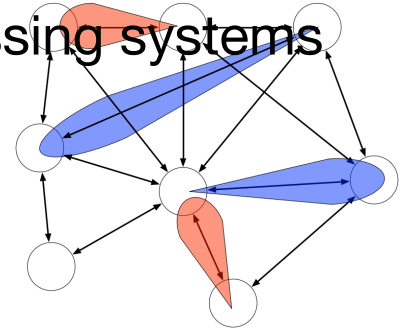
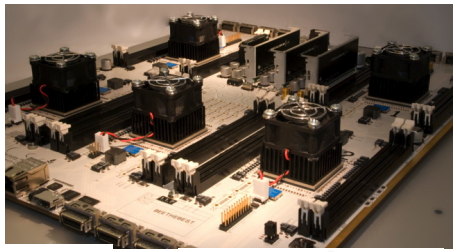
CS61C: Great Ideas in Computer Architecture (a.k.a. Machine Structures)

Dr. John Wawrzynek
Professor of EECS



- Profession Musician in New York
- JPL/NASA – space craft data systems
- PhD in CS fro Caltech – electronic music
- Berkeley faculty since 1989
  - IC design, signal processing systems
  - High performance computer design
  - Wireless system design





bina TECHNOLOGIES

ANDES NETWORKS

BEEcube

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

Connor “Cece” McMahon
mcmahon@

Hi everyone! I'm a second year CS Master's student from Atlanta, Georgia. I spent the 1st year of my program online in Atlanta, so I'm very excited to be here in person this semester! Can't wait to meet everyone!

Justin Yokota
jyokota@

Hi everyone! I’m Justin, a student in the Fifth-Year Master’s program for CS, who graduated as a Math/CS double major last semester. This semester, I’ll be the head TA primarily responsible for content (ex. homeworks, projects, exams). Outside of classes, I tend to solve puzzles and play board games. Good luck!

Avi Nandakumar
avinashnandakumar@

Hello everyone, I'm a 5th year EECS masters student from the small city of Camarillo in southern California, near Santa Barbara! This is my sixth semester teaching 61C and will be the head TA in charge of discussions, exam-reviews, and student relations, and could not be more excited. I am super social, love to be outdoors, and a huge sports fan. Come to my section and talk to me about anything!

Jerry Xu
jerryxu@

Howdy! I’m Jerry, EECS senior. I handle some of the infrastructure/software bits and bobs around here.

The TA Corps:

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

Agenda

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

Course Information: ***https://cs61c.org/***

Wawrzynek and Weaver

Note: This website is under construction. All content is tentative and subject to change.

Great Ideas in Computer Architecture (Machine Structures)

CS 61C at UC Berkeley with Nicholas Weaver, John Wawrzynek - Fall 2021

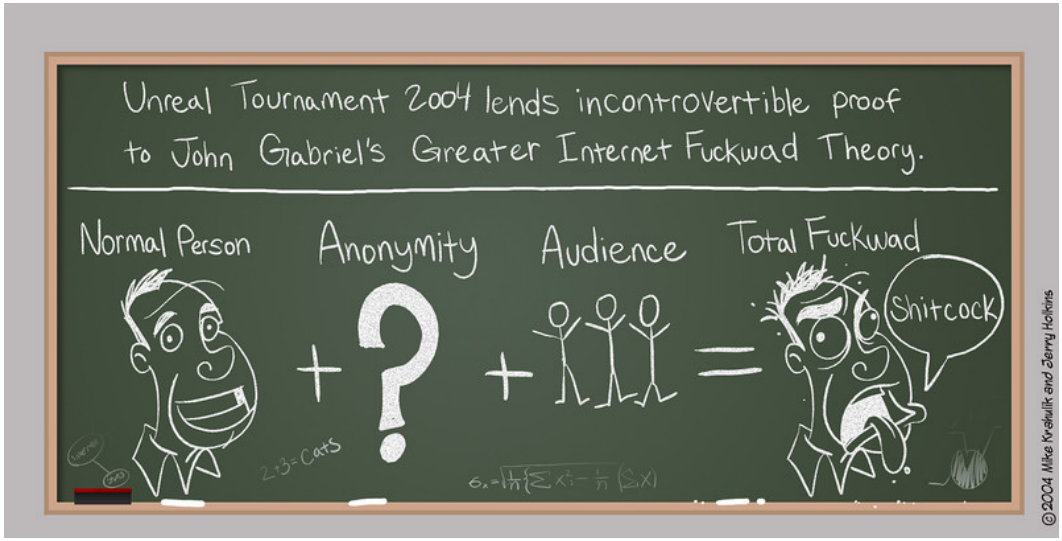
Week	Date	Lecture	Reading	Lab & Discussion	HW & Project
1	Fri 8/27	Intro, Number Representation	Course Policies		
2	Mon 8/30	C Intro	K&R Ch. 1-5 C Reference Slides Brian Harvey's Intro to C	Discussion 1: Number Rep	
	Tue 8/31			Lab 0: Intro and Setup Due 9/03	Homework 1: Number Rep Due 9/10
	Fri 9/3	C Arrays, Pointers, Strings	K&R 5-6		Project 1 Due 9/20
3	Mon 9/6	No lecture: Holiday		Discussion 2: C Basics	
	Tue 9/7			Lab 1: Number Rep, C, GDB Due 9/10	Homework 2: C Concepts Due 9/17

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 (Online so don't need to buy)

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
- Non-enrolled students will not be allowed to join our Piazza forum, as we do not have enough bandwidth to answer questions from students who are not enrolled in the course.
- You can have posts be anonymous to other students but ***not*** to course staff.



Gradescope, Class Accounts, Concurrent Enrollment…

- We will use Gradescope for all assignments
  - Midterm/Final with a regrade window
  - Homeworks
  - Labs
  - Projects
- Class accounts:
  - https://inst.eecs.berkeley.edu/webacct
  - Login here with your Calnet ID to get course computer account starting on the first day of instruction.

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
- We have a departmental target (2.8-3.3 GPA, and we 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 … 🤷‍♂️
    we have smart students and don't need to make later assignments harder
- If bins are below our target GPA ... ***then*** we 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)

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...
- We have a special GSI, Vron, and department course managers, Michael-David Sasson and Krystle Simon, to handle/track a lot of the logistics
  - So don't be surprised if one of them reaches out to you.

Late Policy for Projects…
Slip Days!

- Assignments due at 11:59:59 PM PT.
- You have 72 slip hours tokens that apply to projects.
  No slip hours for homeworks assignments (problem sets).
- Every hour your project is late (even by a millisecond) we deduct a token
- For projects, after you've used up all tokens, it's 1/72 of your points will be deducted.
  - (No credit if more than 3 days late.)
- No need for sob stories, just use a slip tokens!
- Grade in the end will use the optimal distribution of slip days between all your projects

Late Policy for Homeworks: 1/72nd every hour

- Every hour a homework is late deducts 1/72 from the score
  - So zero credit after 3 days late
- Each homework has its own late counter
- If you need an extension due to extenuating circumstances, just ask on the form!

Labs ...

- 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
  - ***Highly recommended*** to have a partner
  - After completing each lab, you will meet with a course staff member for checkoff, where you will answer a few checkoff questions to demonstrate that you have completed the assignment. Graded out of 2 points.
  - For full credit, labs must be checked off in the week they are assigned. You can get up to half credit for checking off the week after a lab is assigned. We will give 25% extra credit per lab if you submit the checkoff request within the first hour of your assigned section.
  - You have 1 "lab drop"
- Lab and discussions start next week - both in-person and remote labs.

Beyond that:
Extension Requests

- There are two forms on the web site:
  - One for DSP students, one for non-DSP students
- If you feel you have a need for an extension beyond slip days, use the appropriate form.
- Again, we will ask the course managers process these requests.

Exams

- We will have two exams, a midterm and a final
  - There will be an alternate time slot ***immediately*** after the scheduled slot for those with a time conflict
  - There will be a second alternate time-slot skewed greatly for those on the other side of the planet
- Exams will be full hybrid
  - You can take an exam in person in a room like the Before Times at the scheduled time
  - You can take an exam remotely with remote proctoring
- You will default to in-person, but we will have a form a couple weeks before the exam to request alternates/remote exams
  - You need to give us a reason but it is pretty much any reason:
    E.g. Nick takes exams better if his cat is on his lap.

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 mistakes:
    "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 and Office Hours...

- We are all here to help 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
- We also unfortunately have to enforce office hour time limits

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 we (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"
  - We will go to DEFCON 1 (aka "launch the nukes") ***immediately***
    - “Nick doesn’t make threats. ***He keeps promises***”

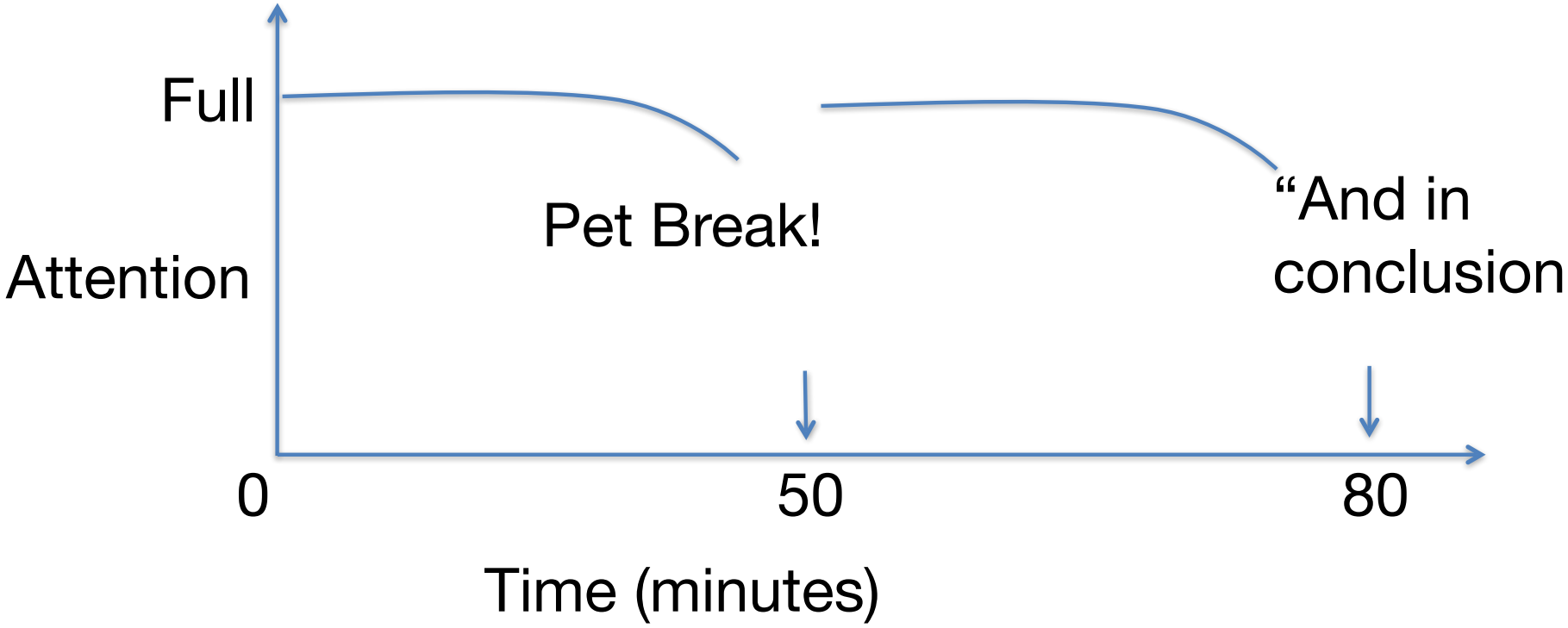
More On Academic Dishonesty...

- We take cheating ***personally***
  - We 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 we have just a 50% chance of catching you, you are provably better off just not doing the work. It is not rational to cheat
  - And penalties can go up from there:
    We can and have given Fs
- We 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(*log(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



Zoom And "Pet Time"

- Doing everything over Zoom ***sucks*** except for two things!
  - Chat and PETS!
- ***Use the chat to ask questions***
  - We get more interactivity with the chat channel than we did with just "raise hand" in the lecture hall..
    So we are going to use it!
- Lectures start at Berkeley Standard Time...
- But the 20 minutes before is "Pet Time"
  - We will bring our pets (when they are behaving) if we are operating remotely
  - You are encouraged to bring yours and share

Agenda

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

CS61C is not really about C Programming

- It is about the hardware-software interface
  - What does the programmer need to know to make best use of the hardware.
- C is close to the underlying hardware, unlike other common languages like Python, Java, JavaScript, Go, R, Perl, Ruby, Haskell ...
  - Allows us to talk about key hardware features in higher level terms
  - Allows high performance
  - Only language which comes close is Rust…
- Also allows programmer to shoot themself in the foot in ***amazingly*** spectacular ways
  - A goal in this class is for you to understand how C can be dangerous - leading to security loopholes and difficult bugs

Other Pedagogical Choices In This Class...

- Our processor of choice is the "RISC-V"
  - Invented at Berkeley as a open-source hardware alternative to commercial processors, eg. x86, ARM
  - Now gaining wide acceptance worldwide in academia and industry
  - RISCs (Reduced Instruction Set Computers) by design are much simpler than tradition CISC architectures (x86 from Intel/AMD best examples) => simpler to learn, program, write compilers for, etc.
  - RISC-V is our model for learning about processor hardware design
  - We learn how to program at the processor instruction set level with RISC-V assembly language
- Our hardware design is in Logisim (schematics)
  - Later if you 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

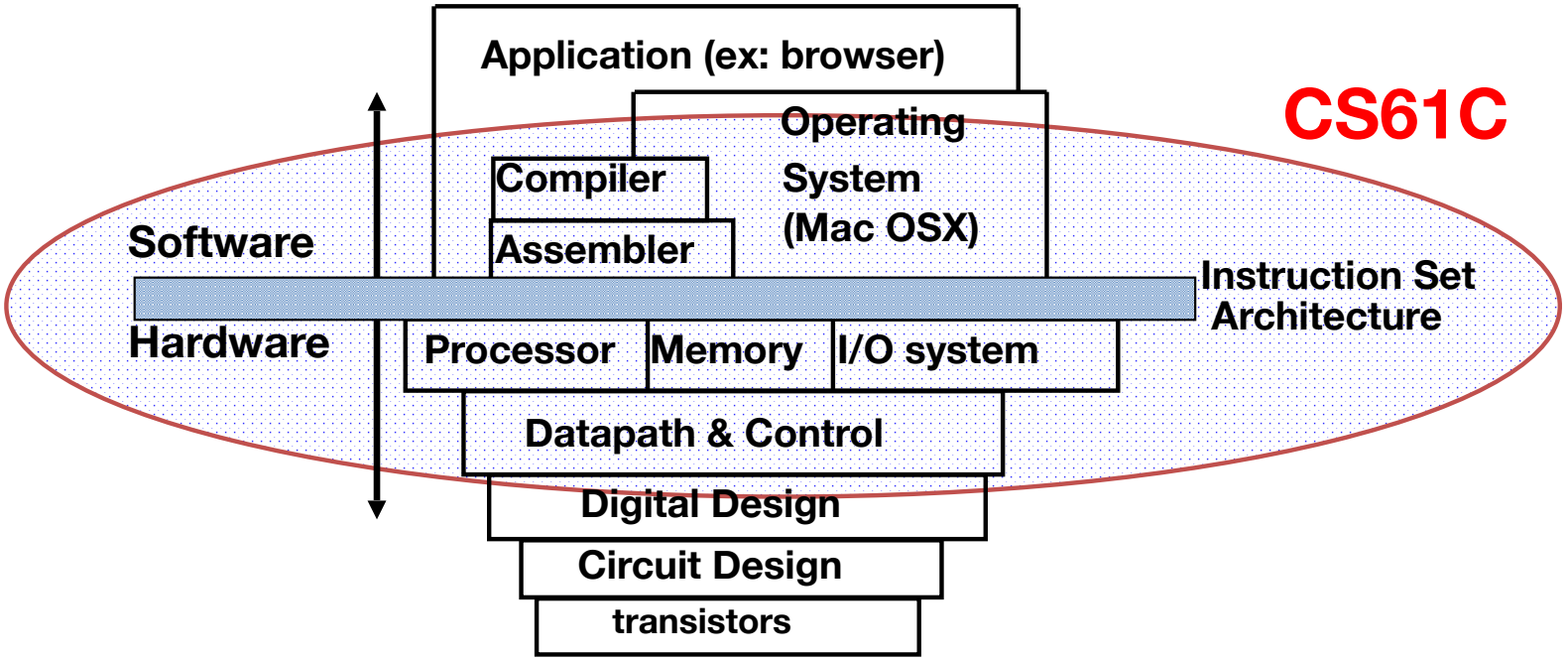
Modern 61C:
From the small...

To the very small...

To the big…

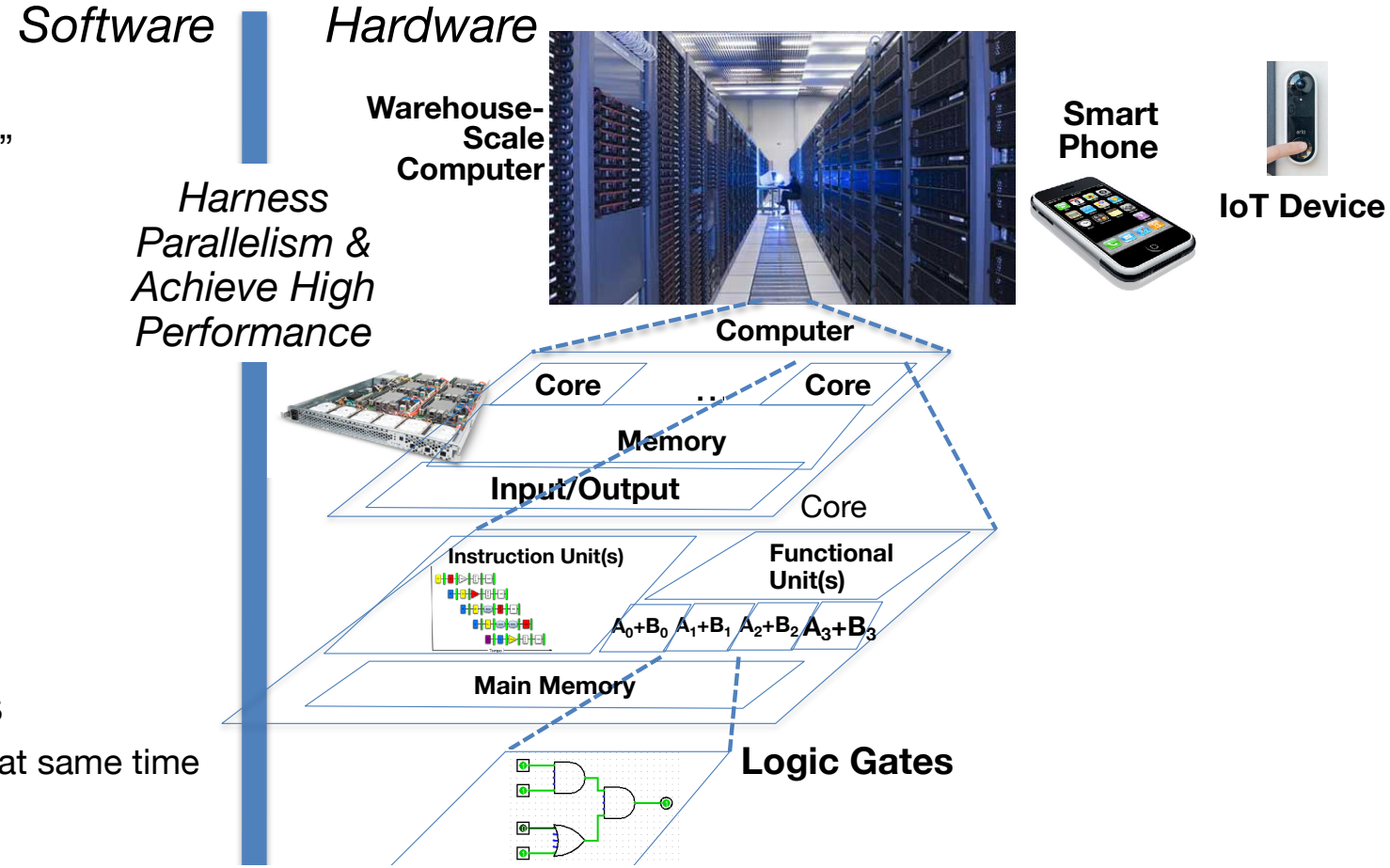

warehouse-scale
computer
My other computer
is a data center

Old School Machine Structures



New School 61C:
From the Data Center to the Hardware Logic Gates

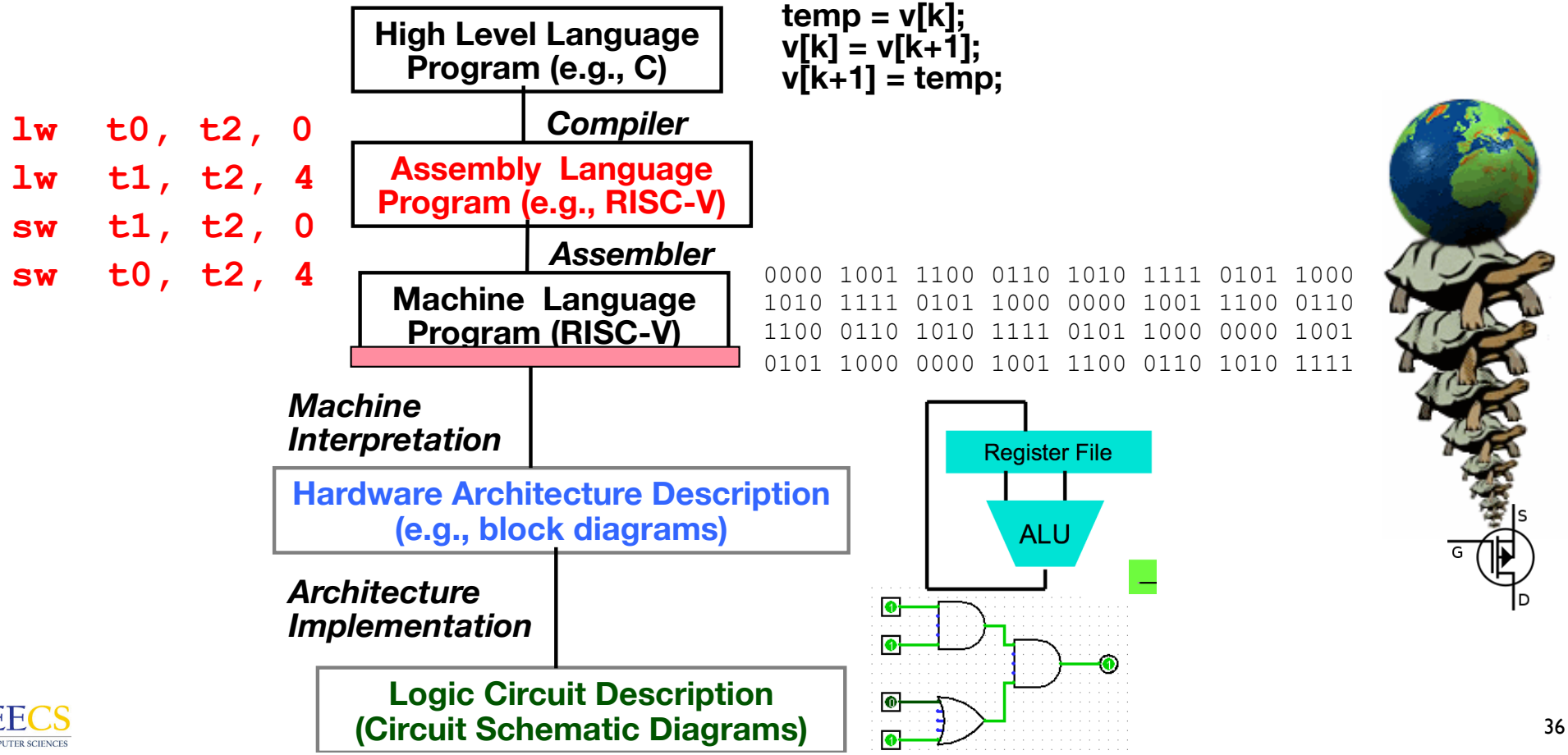
- Parallel Requests
  Assigned to computer
  e.g., Search “giant military cats”
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware 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)



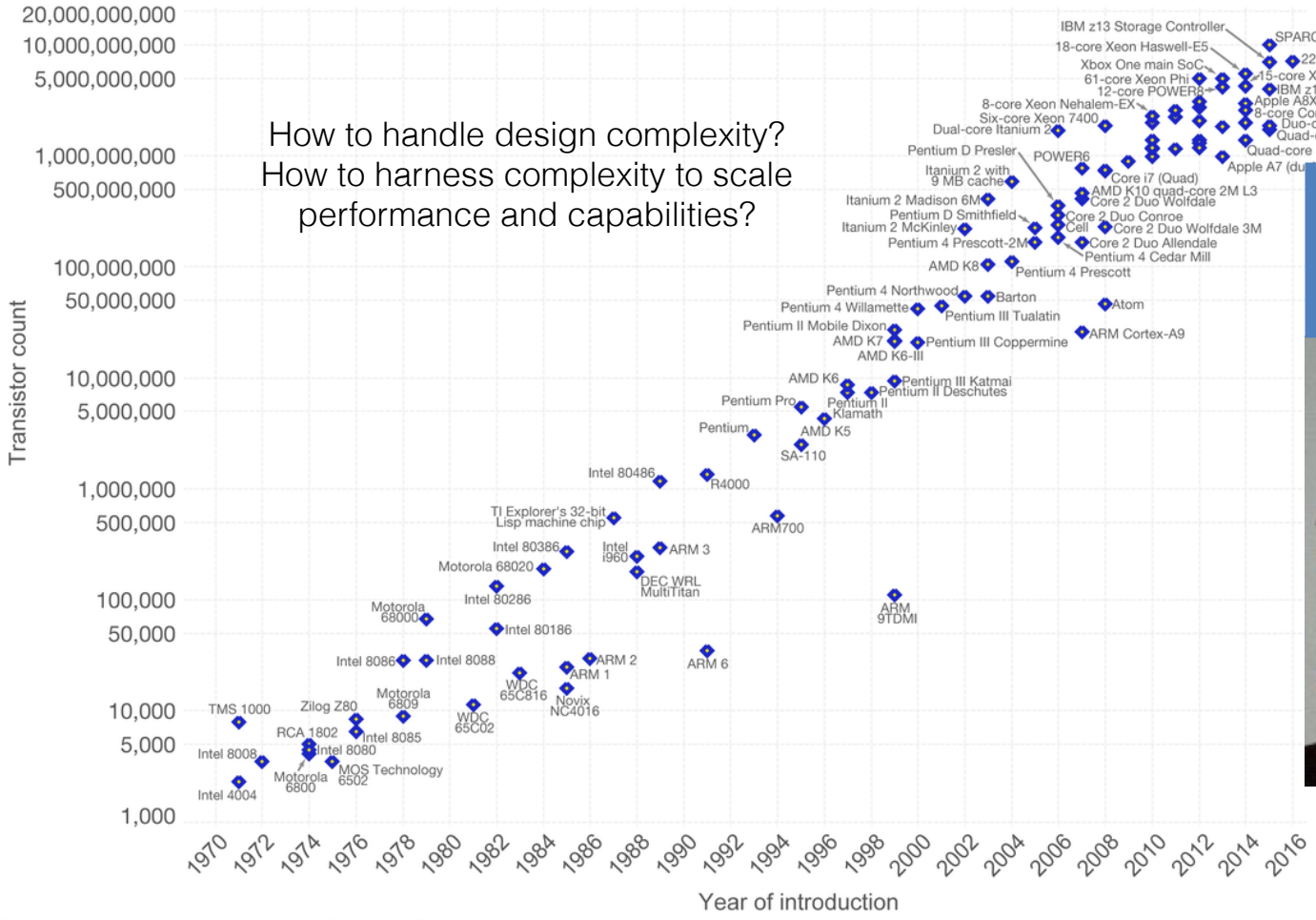
#2:

Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

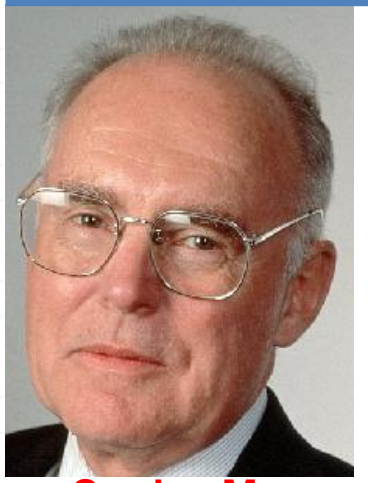
Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are strongly linked to Moore's law.

How to handle design complexity?
How to harness complexity to scale performance and capabilities?



Predicts:
2X Transistors / chip every 2 years



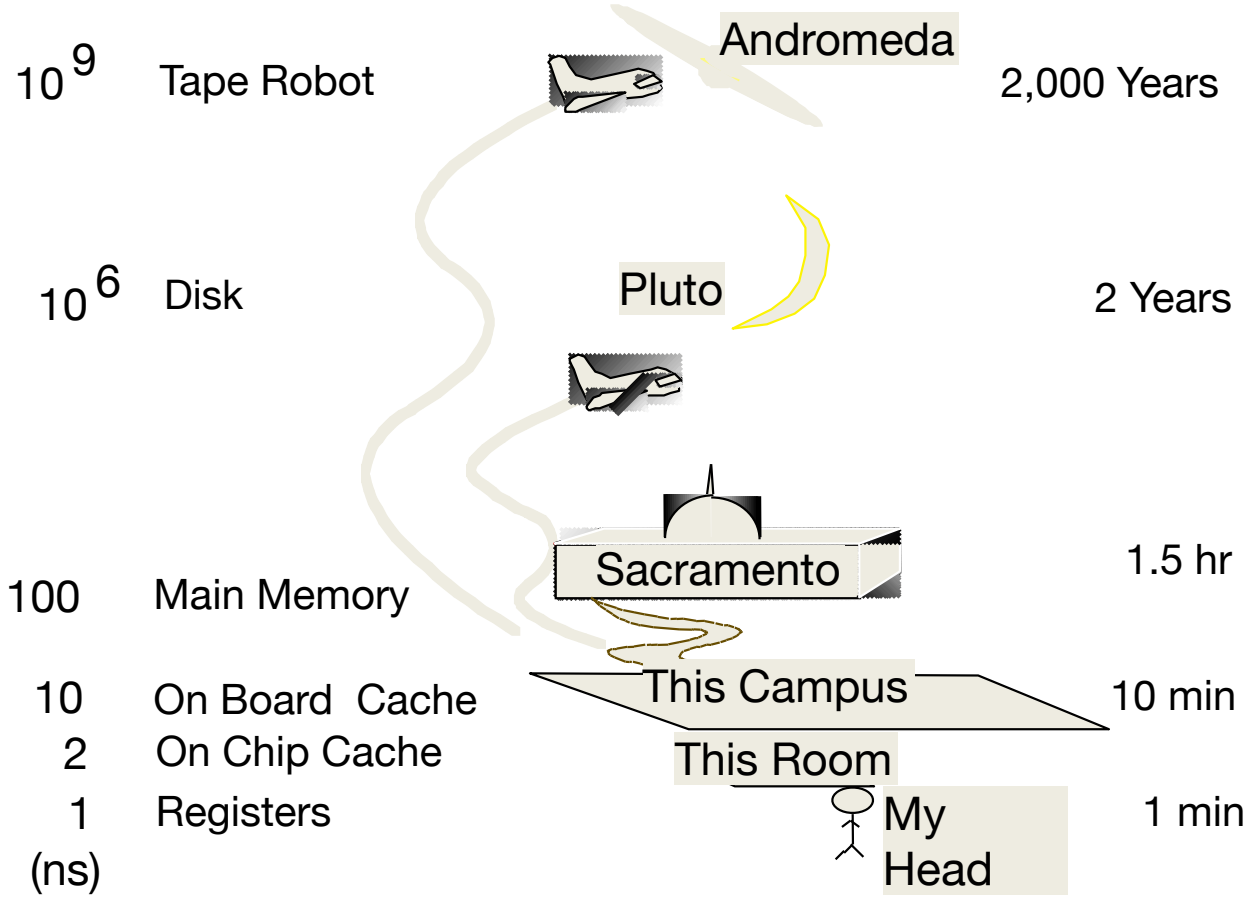
Gordon Moore
Intel Cofounder
B.S. Cal 1950!

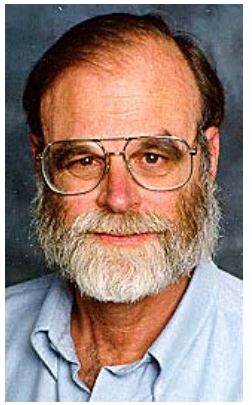
Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

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...
- State-of-the art commercial devices now 5nm
  - Latest TSMC/Apple silicon...
    This is only ***part*** of the reason the M1 processors are so fast!

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

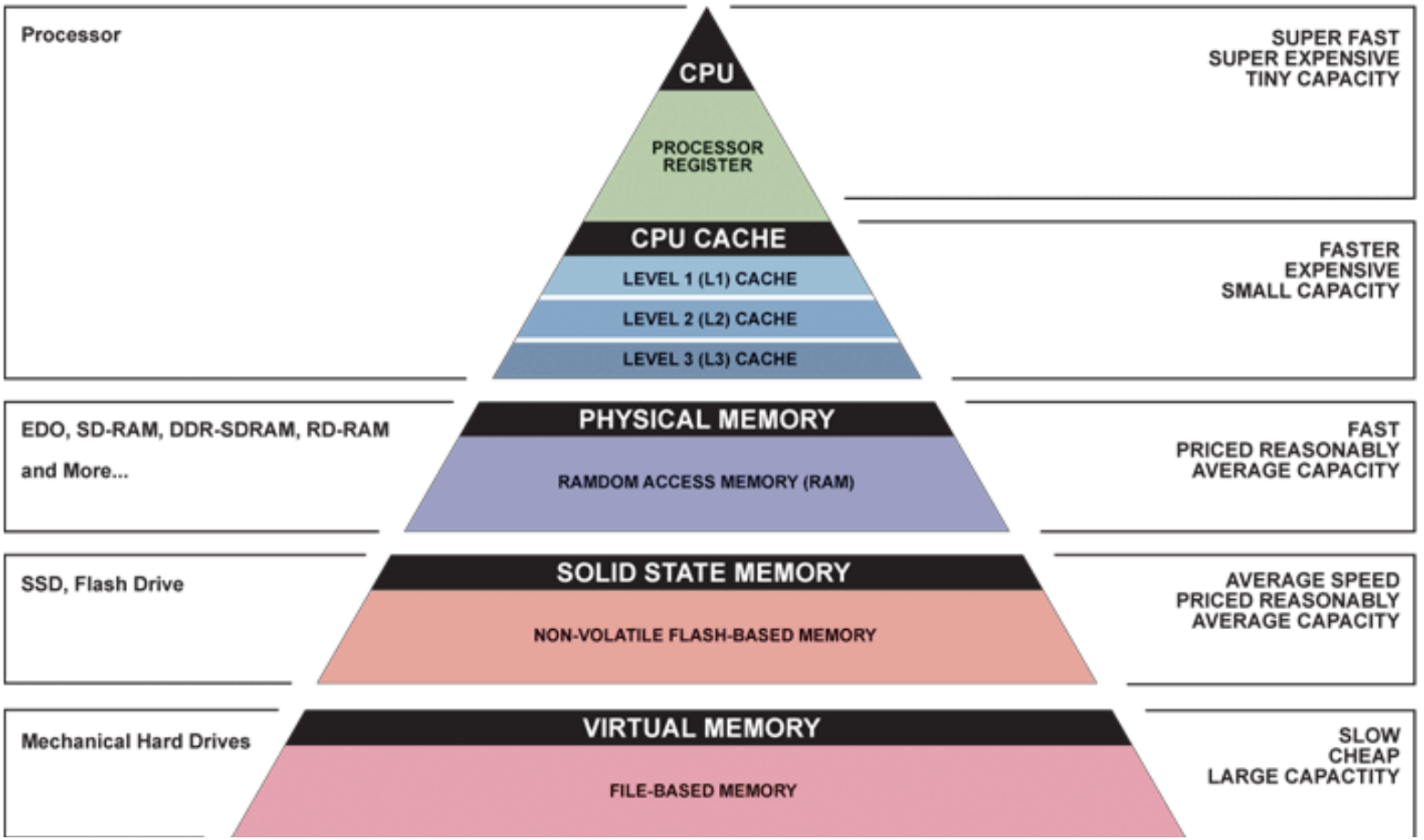




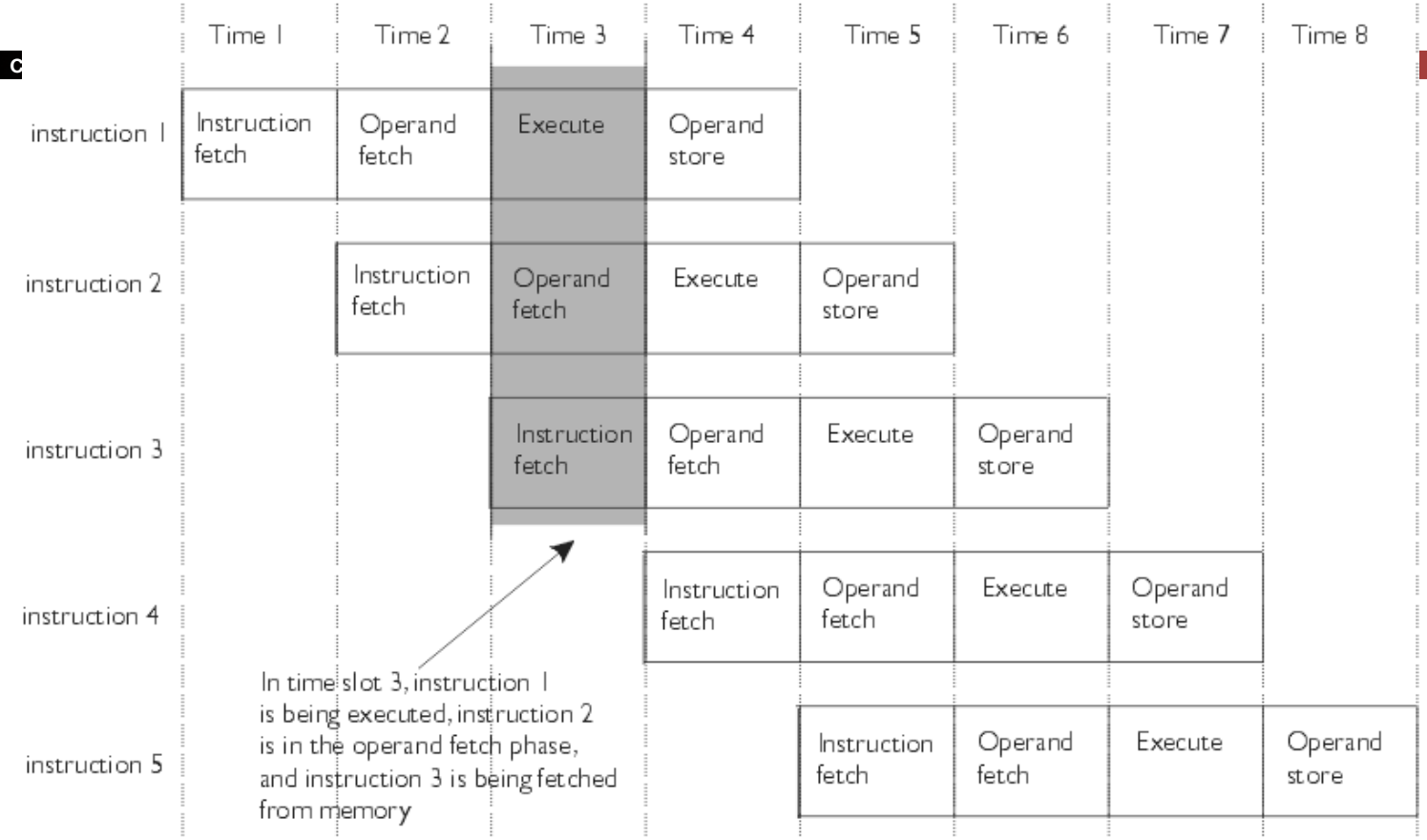
Jim Gray
Turing Award
B.S. Cal 1966
Ph.D. Cal 1969!

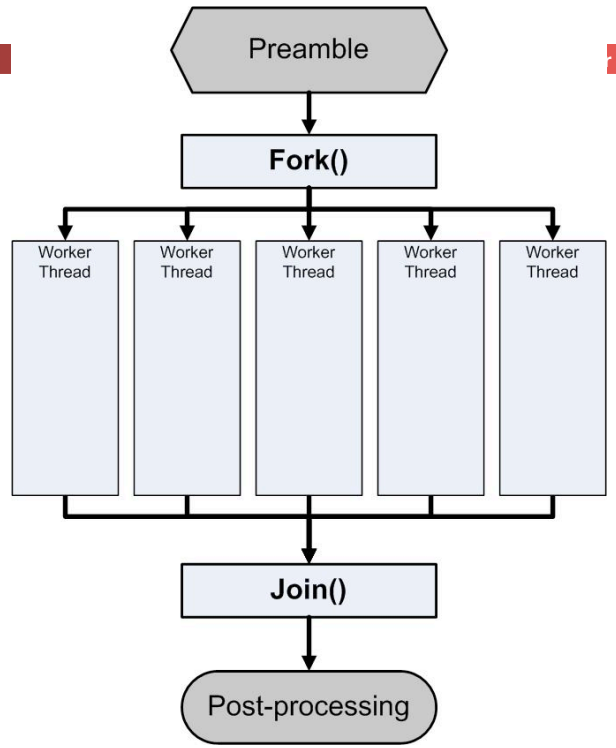
Great Idea #3: Principle of Locality/ Memory Hierarchy

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



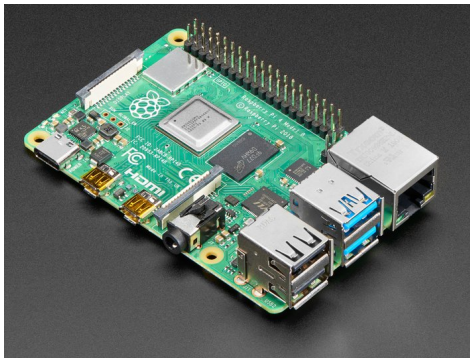
Great Idea #4: Parallelism





And Parallelism is ***Everywhere***

- Most evident at the processor level:
- 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 the Compute Module 4 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:
  - 2u server which supports 4 processors - each processor can have 20+ cores, so 80 processor cores!
- But, as we will see, present at every level of hardware design
  - multiple memory units, arithmetic units, logic gates all operate in parallel, etc.
- *Parallelism is what makes hardware semantics different from single-thread software semantics.*

The Caveat: Amdahl's Law

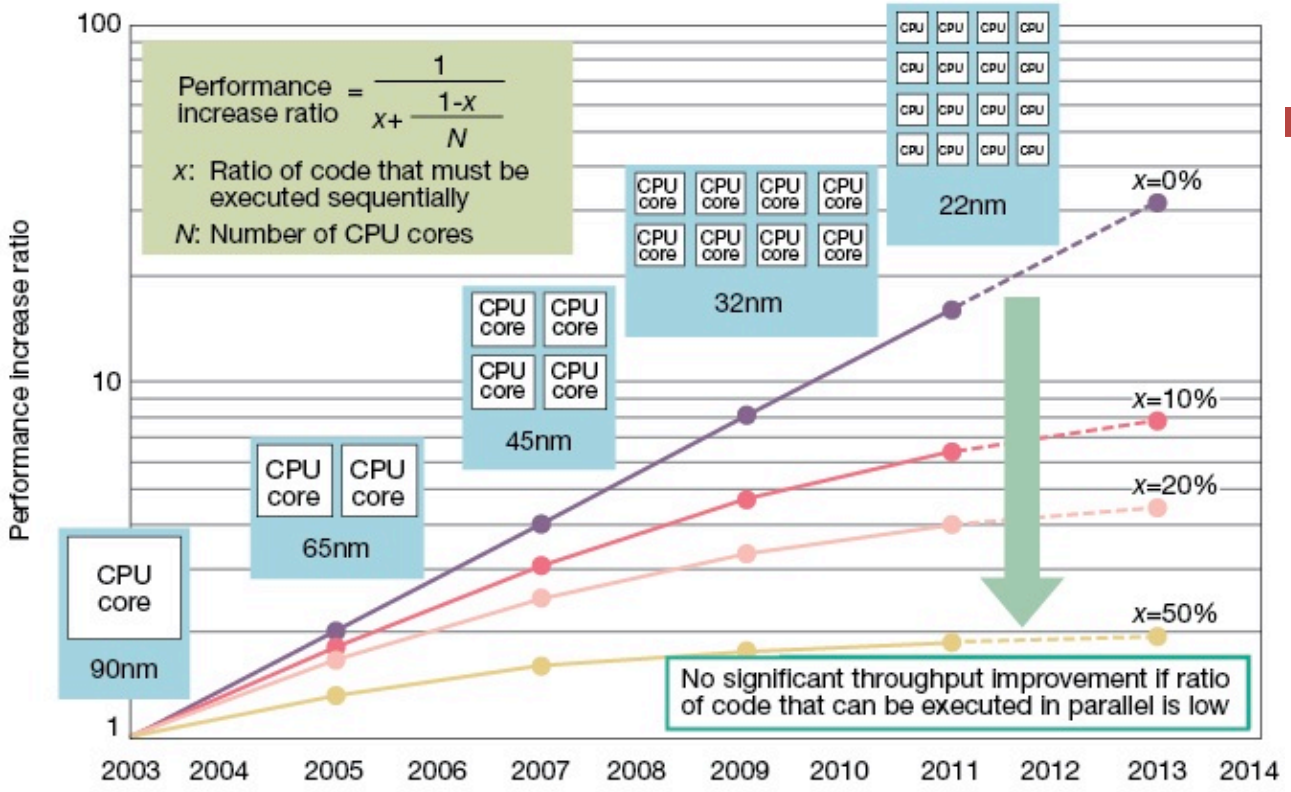


Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor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

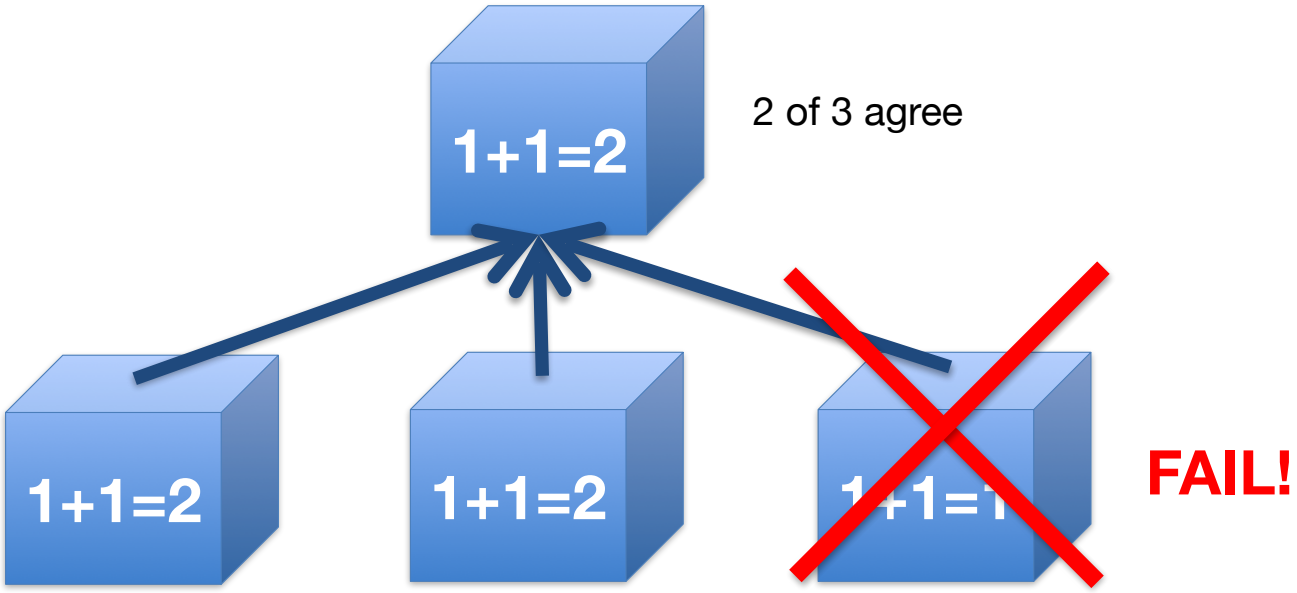
NASA Fixing Rover's Flash Memory

- "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!**

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

Agenda

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

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
- Everything in a computer is represented with bits...
  - Not just numerical values, but text, pictures, sound, …
- Arithmetic operations within hardware can lead to results too big/small to store within their representations: *overflow/ underflow*

Why Bits?

- A single bit is either 0 or 1...
- At the hardware circuit level, bits are very easy to ***robustly*** represent - store, manipulate
  - We will see how later
- But it is also insanely powerful: two entire branches of mathematics
  - Boolean algebra: logic for working with only two values: true and false
  - Information theory: how to encode data
- The latter is critical:
  With ***N*** bits, we can represent one of up to 2^N things
  - EG, we need 2 bits to represent the suite of a card (Hearts, Spades, Clubs, Diamonds)
  - We need 4 bits to represent the value (Ace-King: 13 possibilities)
  - Or we can use 6 bits to represent both the suite and value of a card

Other bases: In general with ***N*** digits we can represent unsigned integers from ***0*** to ***base^{N}-1***

- Binary (Base 2), Octal (Base 8), Hexadecimal (Base 16), Decimal (Base 10) all different ways to represent integers
- Value of i^{th} digit is $d \times Base^{i}$ where i starts at 0 and increases from right to left:
- $123_{10} = 1_{10} \times (10_{10})^2 + 2_{10} \times (10_{10})^1 + 3_{10} \times (10_{10})^0$

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

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

  $= 123_{10}$

  - Same idea works for all bases
  - And in song: https://www.youtube.com/watch?v=UIKGV2cTgqA

Favorites: binary and hexadecimal (plus decimal)

Hex is a convenient way to represent binary

- Hexadecimal digits: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- For instance, the binary number: $1101\ 1110\ 1100\ 1010\ 1111_2 = DECAF_{16}$,
- $DAD_{16} = 1101\ 1010\ 1101_2$
- Proof of equivalence:
- $DAD_{hex} = 13_{10} \times 16_{10}^2 + 10_{10} \times 16_{10}^1 + 13_{10} \times 16_{10}^0$
  $= 3328_{10} + 160_{10} + 13_{10}$
  $= 3501_{10}$
- $1101\ 1010\ 1101_2 = 1x2^{11} + 1x2^{10} + 1x2^8 + 1x2^7 + 1x2^5 + 1x2^3 + 1x2^2 + 1x2^0$
  $= 2048_{10} + 1024_{10} + 256_{10} + 128_{10} + 32_{10} + 8_{10} + 4_{10} + 1_{10}$
  $= 3501_{10}$

(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 usually used for memory 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 integers 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:

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

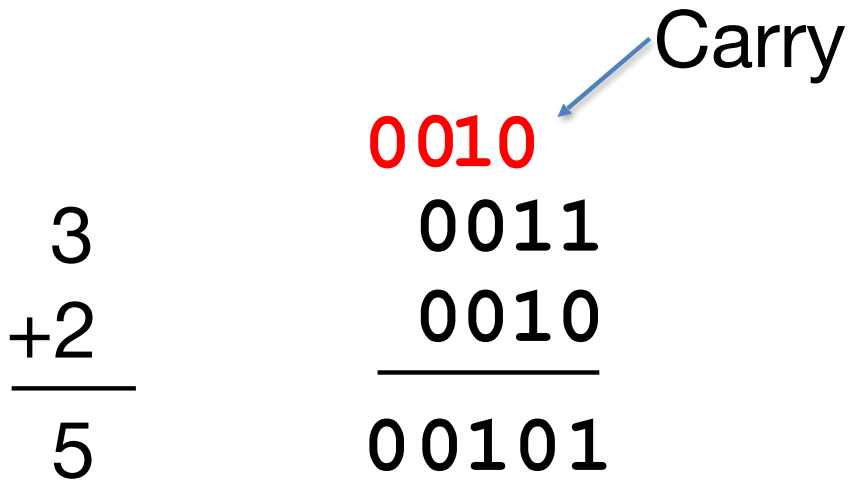
- For a 4-bit number, $3_{10}=0011_2$, its two's complement $-3_{10}=1101_2$ $(-1000_2 + 0101_2 = -8_{10} + 5_{10})$

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

3_{10}	0011_2
Bitwise Invert	1100_2
+	1_2
-3_{10}	1101_2

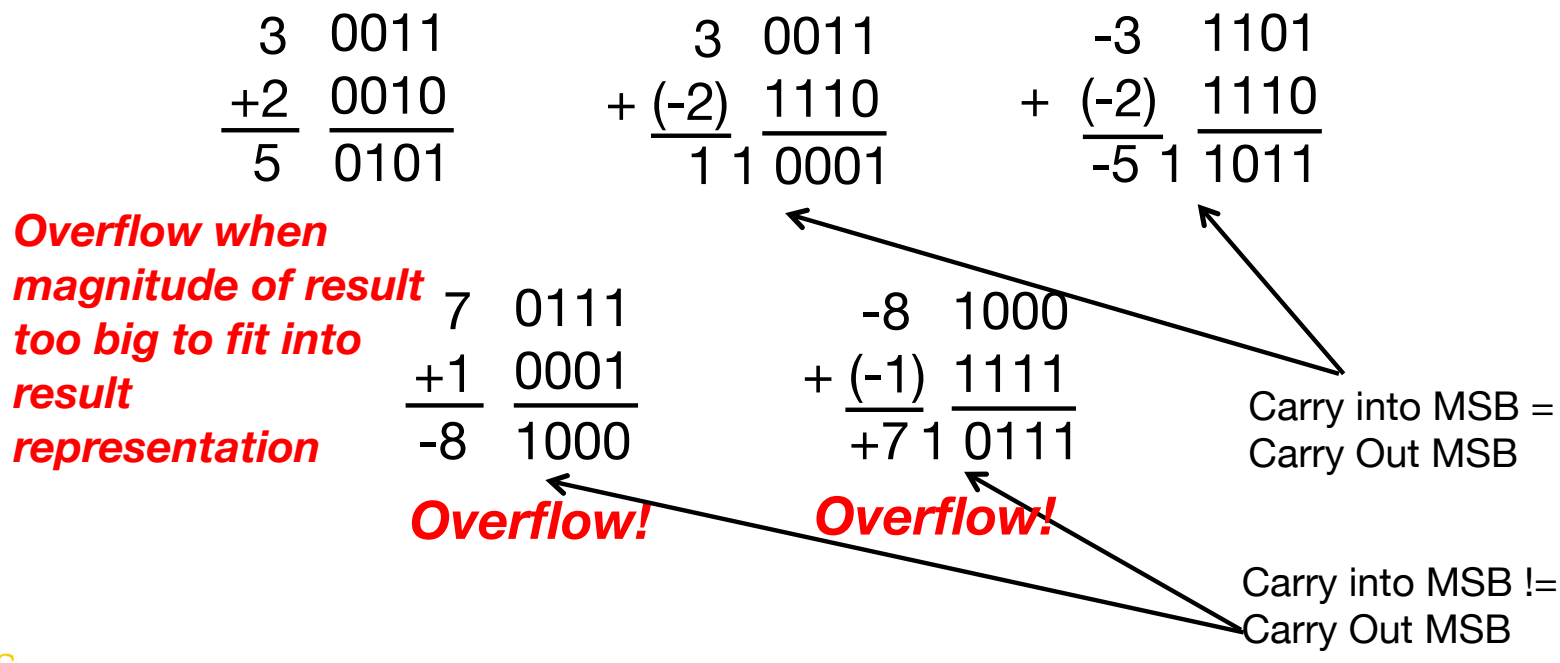
works both ways!

Binary Addition Example



Two's-Complement Examples

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



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
- Signed 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 used in floating point
  - Not going to talk about 1's complement