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

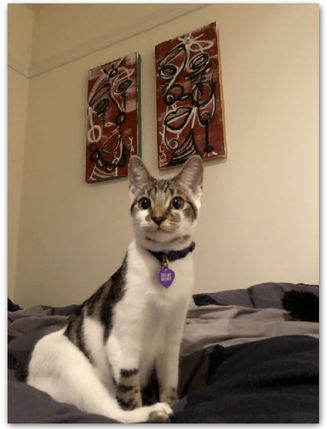
Dr. Nick Weaver
Lecturer (He/Him)

- 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…
  - After all, I hate drones, so...
- 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?

Connor McMahon (she/her)

- Graduate Student in EECS
- Undergrad at Georgia Tech
- Former 61C TA

The Head TAs



Zoe Plaxco
zapplaxco@

Hi! I love teaching and I bake a lot of bread! Come by my office hours to talk about plants, cooking, or existing in a late stage capitalist system. I am deeply passionate about breakfast foods. I am non-binary, and my pronouns are they/them. I am neurodivergent and queer

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!

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/sp22/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/***

CS 61C Spring 2022 Calendar Staff Policies Resources OH Queue Venus Semesters

Great Ideas in Computer Architecture (Machine Structures)

CS 61C at UC Berkeley with Connor McMahon, Nicholas Weaver - Spring 2022

Lecture: Tuesday/Thursday 11:00AM - 12:30PM PT, Online

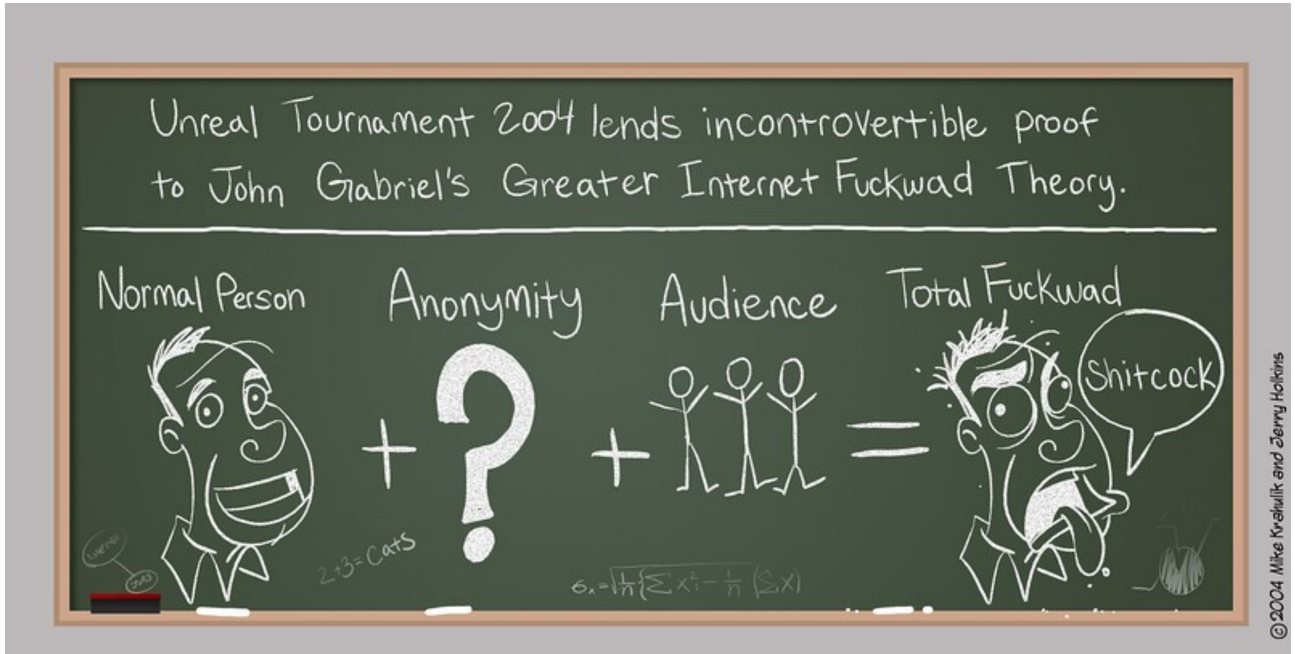
Week	Date	Lecture	Reading	Lab & Discussion	HW & Project
1	Tue 1/18	Lecture 1: Intro, Number Representation Slides Video	Course Policies	Discussion 1: Number Rep Lab 0: Intro and Setup **Due 1/24**	
	Thu 1/20	Lecture 2: C Intro - Basics Slides Video	K&R Ch. 1-5 C Reference Slides Brian Harvey's Intro to C		
	Fri 1/21				Homework 1: Number Rep **Due 1/28**
2	Mon 1/24			Discussion 2: C Basics Lab 1: C & CGDB **Due 1/31**	Project 1: TBD **Due 2/09**
	Tue 1/25	Lecture 3: C Intro - Pointers, Arrays, Strings	K&R 5-6		
	Thu 1/27	Lecture 4: C Memory (Mis)Management	K&R 7.8.5, 8.7		
	Fri 1/28				Homework 2: C Concepts **Due 2/04**
			P&H 3.5, 3.9		

Course Information

- Course Web: https://cs61c.org/
- Major tutoring support from CS-Mentors & paid tutors
- Textbooks: Average 15 pages of reading/week (can rent!)
  - ***highly recommended*** but not strictly required
  - 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 and PrairieLearn for all assignments
  - Midterm/Final with a regrade window
  - Homework on PL
  - 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

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, Zoe Plaxco, and department course managers to handle/track a lot of the logistics
  - So don't be surprised if one of them reaches out to you
  - We also maintain access control: DSP information is only available on a need-to-know basis and most TAs don't need to know.

Late Policy for Projects…
Slip Days!

- Assignments due at 11:59:59 PM PT.
- You have 10 slip day tokens that apply to projects.
  No slip days for homework assignments (problem sets).
- Every day your project is late (even by a millisecond) we deduct a token
  - Can only use up to 4 slip day token on any given project/piece (e.g. 2A and 2B are separate)
- For projects, after you've used up all tokens, it's 1/3 of your points will be deducted per day.
  - (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: 1/3rd every day

- Every day a homework is late deducts 1/3 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 extensions form!

Labs

- Labs are a key portion of the class that will help you with your projects
- We are trying a new lab format this semester
- You can either complete the lab on your own or in a lab section. There will be no lab checkoff
- If you attend lab section, the TA will guide you through the lab as a class to ensure that you can finish the lab in the allotted time

Labs

- Labs may be turned in one week late for half credit
- We will drop your lowest lab score at the end of the semester
- Lab 0 is completed on your own
  - No lab section available, but you can go to OH or ask questions on piazza if you need help
  - All other labs will have sections that you can attend, so if you have a question about them, go to a lab section or ask on piazza (no OH help)
- Lab 1 will have the option of attending a virtual section
- Hopefully the remaining labs will have both virtual and in-person sections

Beyond that:
Extension Requests (update)

- There are two forms on the web site:
  - One for normal extensions, one for if the manure is hitting the 2 MW Wind Turbine
- If you feel you have a need for an extension beyond slip days, use the appropriate form
- Our standard for granting extensions is very generous:
  Is it "reasonable"?
  - The TA responsible can say "yes" if its reasonable in their judgement (and their latitude is broad in terms of allowing extensions)
  - IF a TA might say "no", it instead becomes the instructors decision under the same "reasonable" standard

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

- Projects can either be solo or in pairs, as can labs
- All homework is to be your work ALONE.
- You are encouraged to discuss your assignments with other students, 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***”

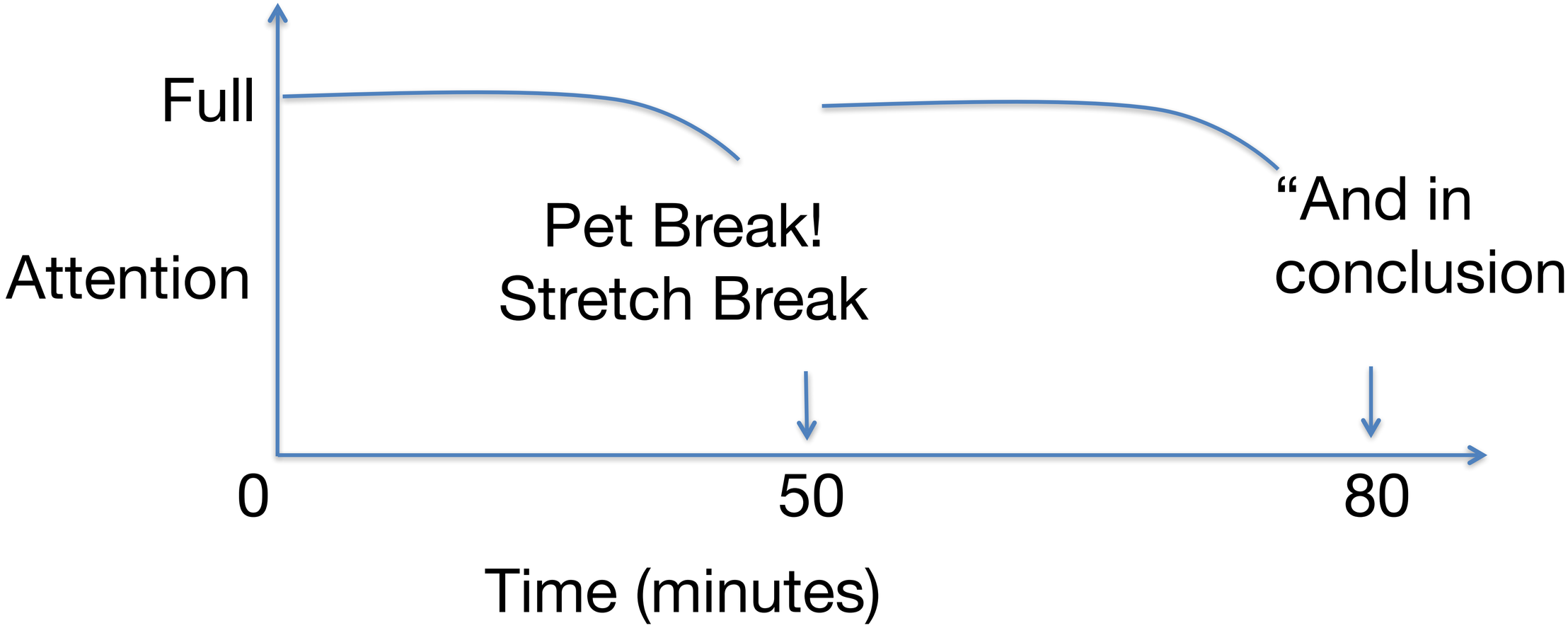
More On Academic Dishonesty...

- 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
  - Oh, and university policy says we can't reduce the penalty after you meet with us
- Nick takes and handles 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 while we are on zoom, 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

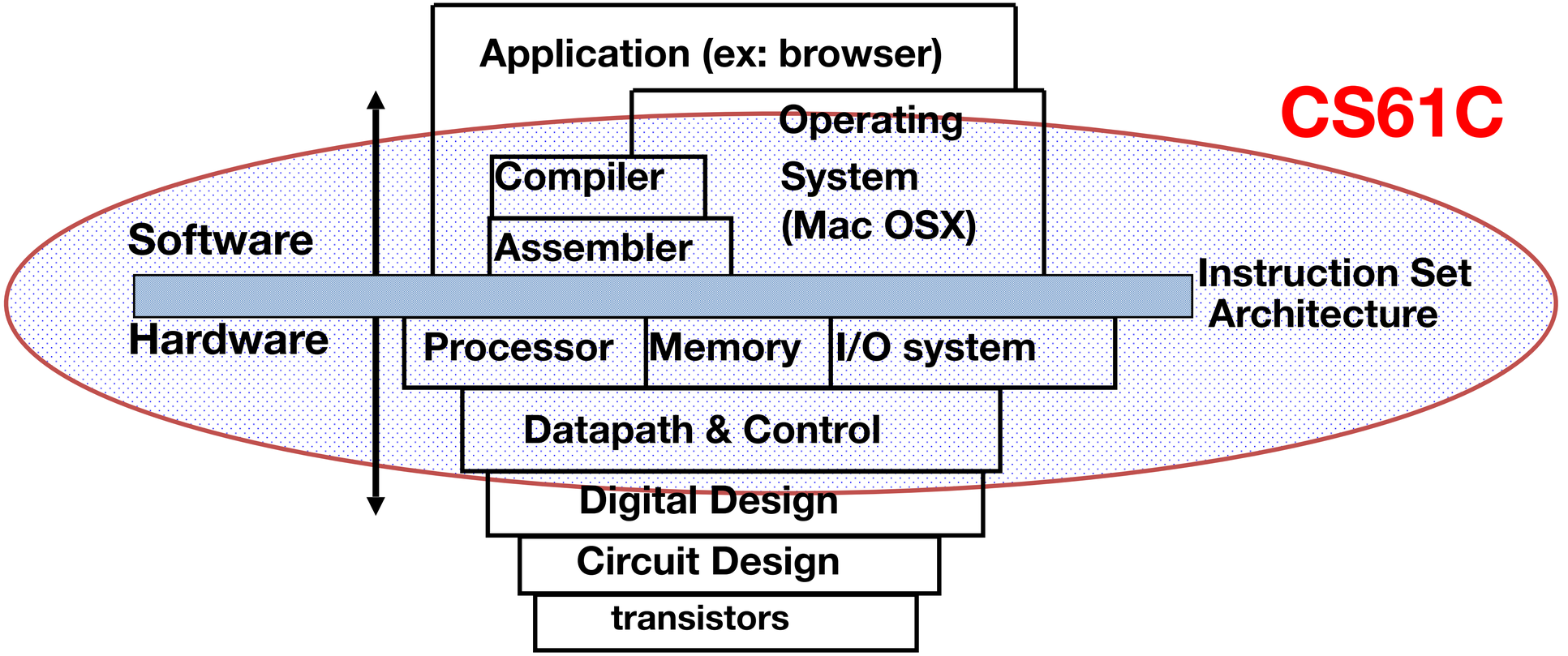
Modern 61C:
From the small...

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

Software *Hardware*

Harness Parallelism & Achieve High Performance

Warehouse-Scale Computer

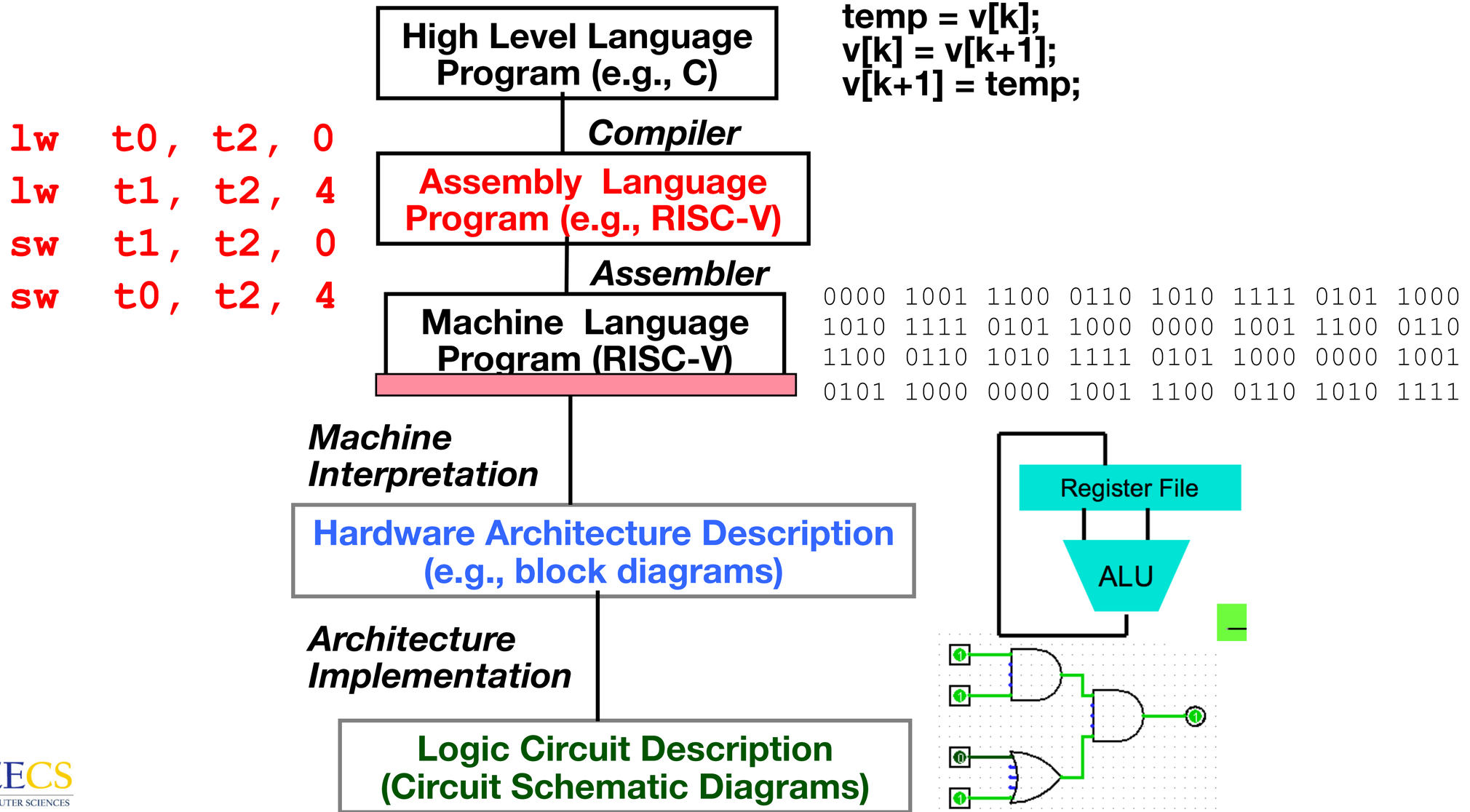
Smart Phone

IoT Device

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)

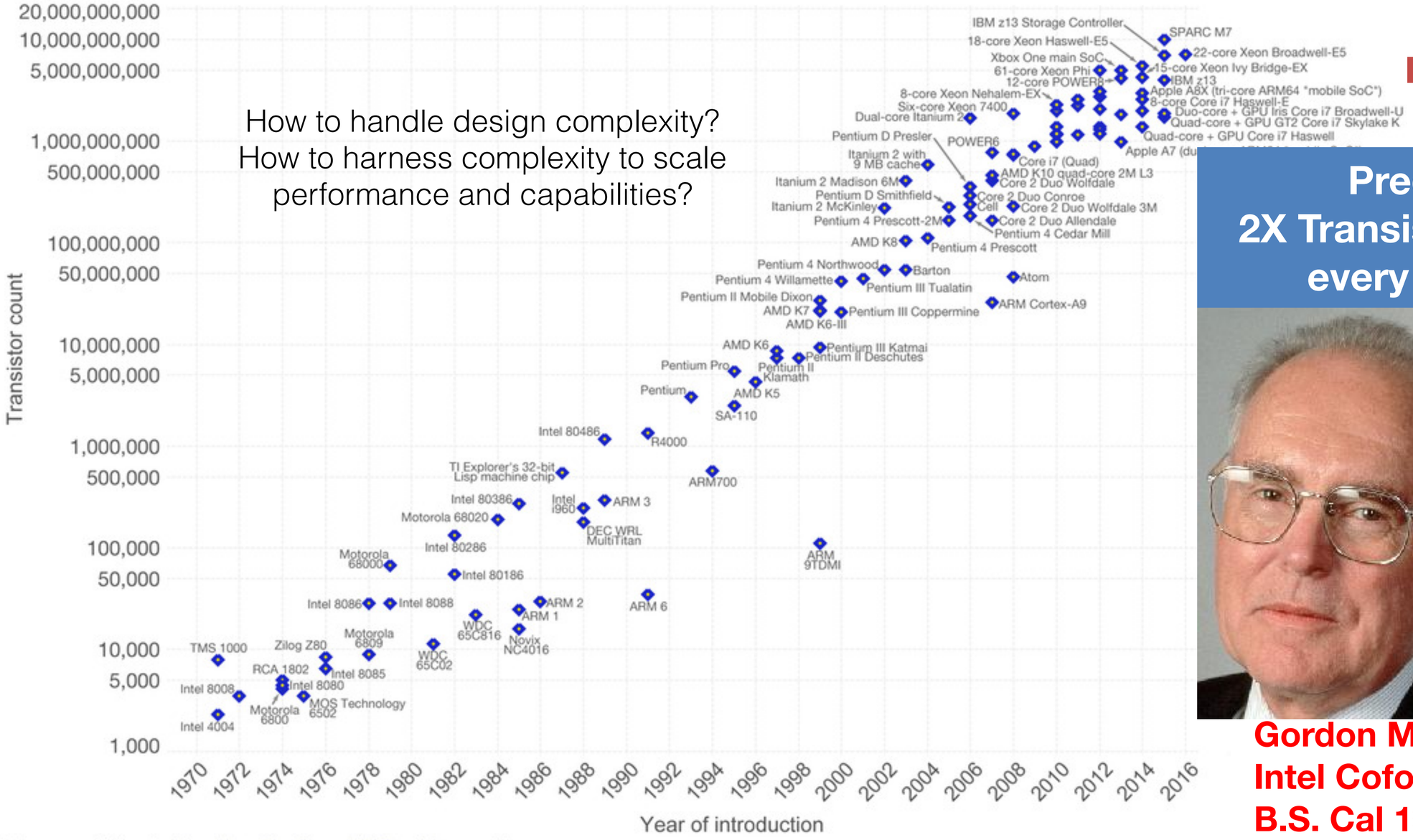


#2: Moore's Law – The number of transistors on integrated circuit chips (1971-2016)

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!

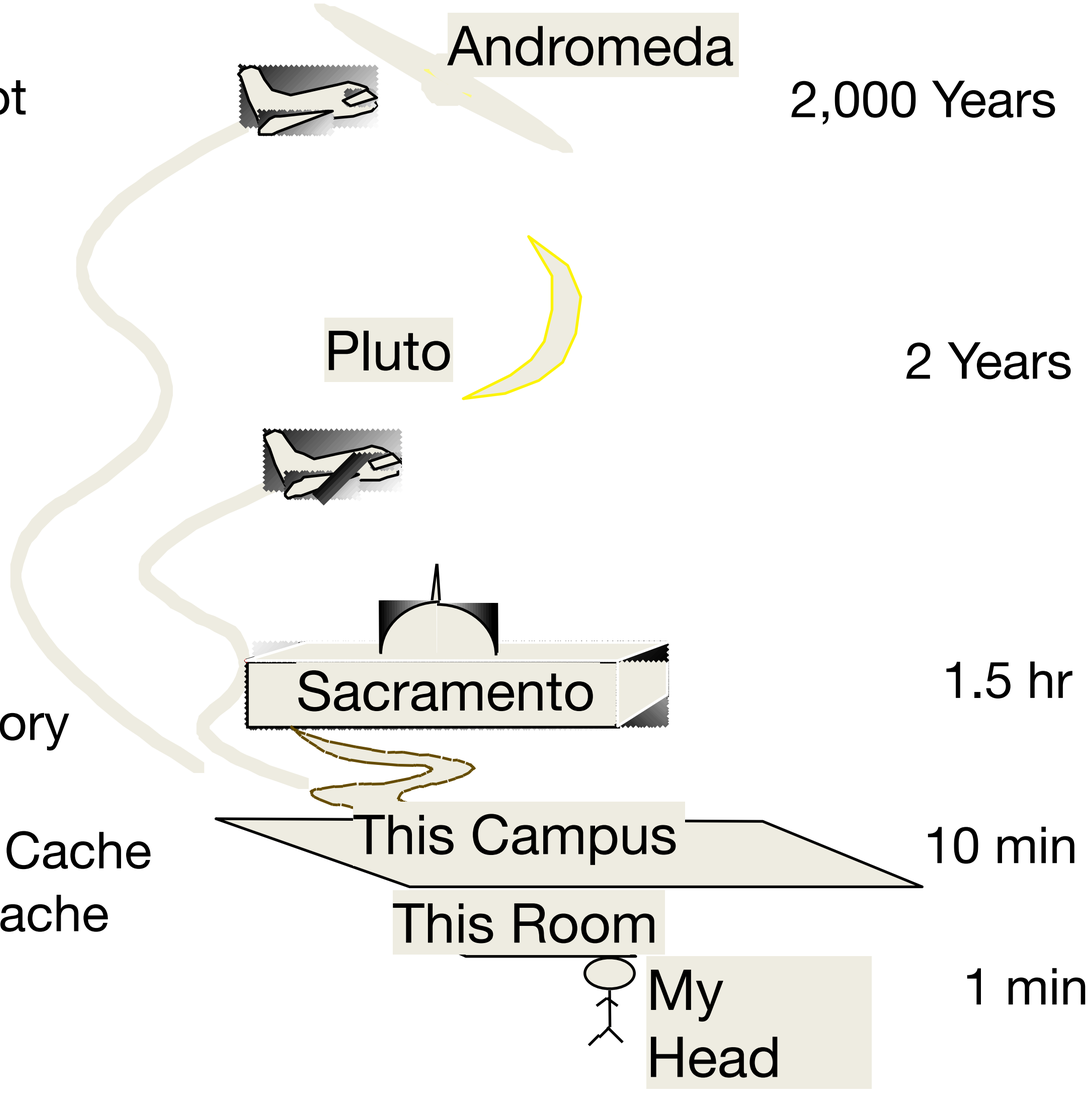
Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

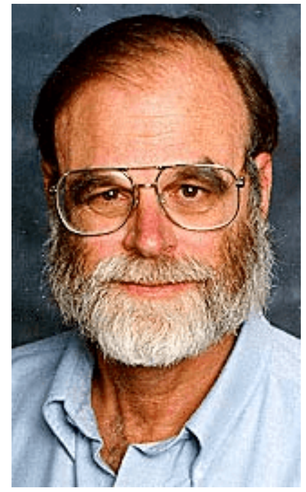
Interesting Times

- 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?

10^9	Tape 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)		

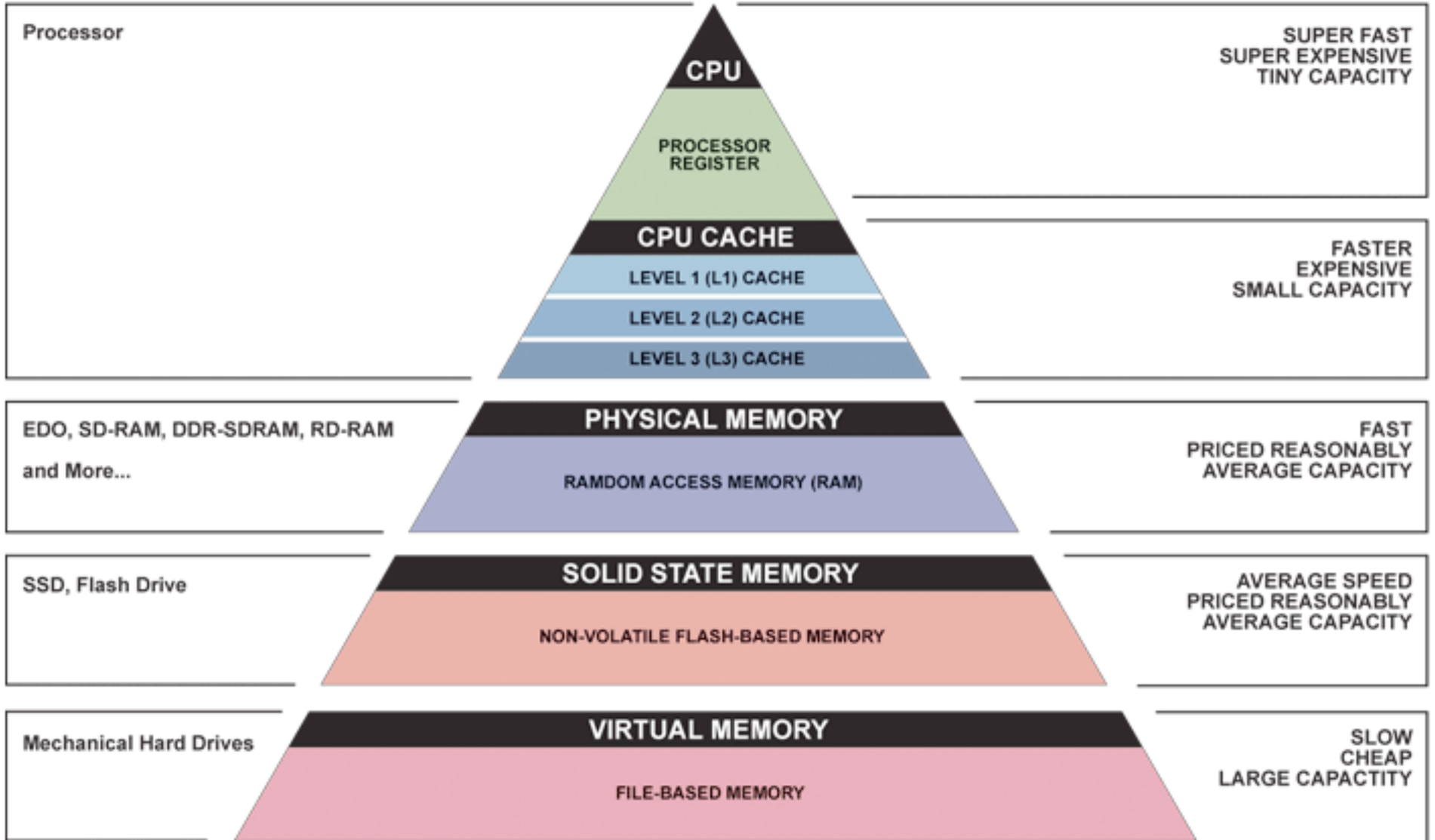




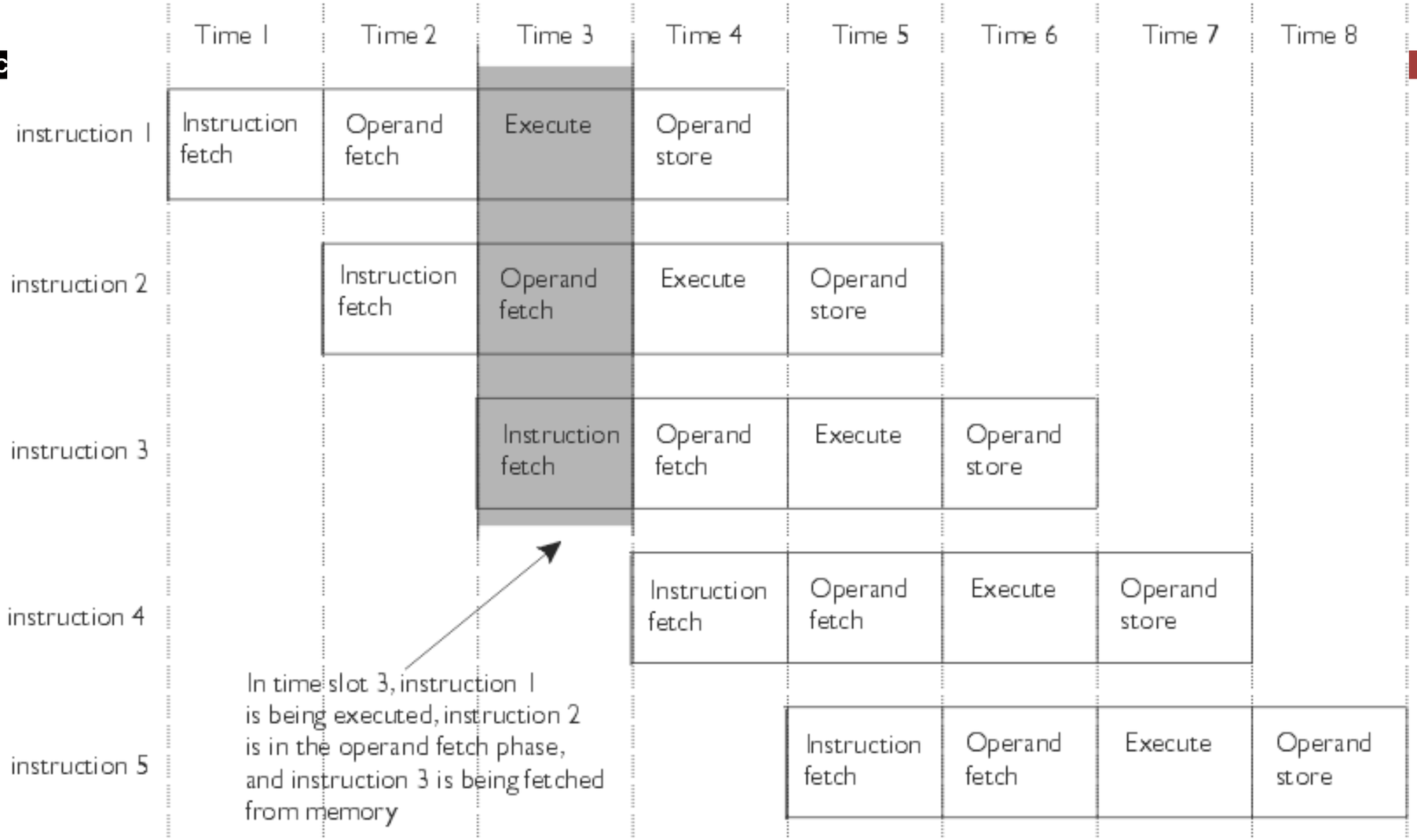
Jim Gray
Turing Award
B.S. Cal 1966
Ph.D. Cal 1969!

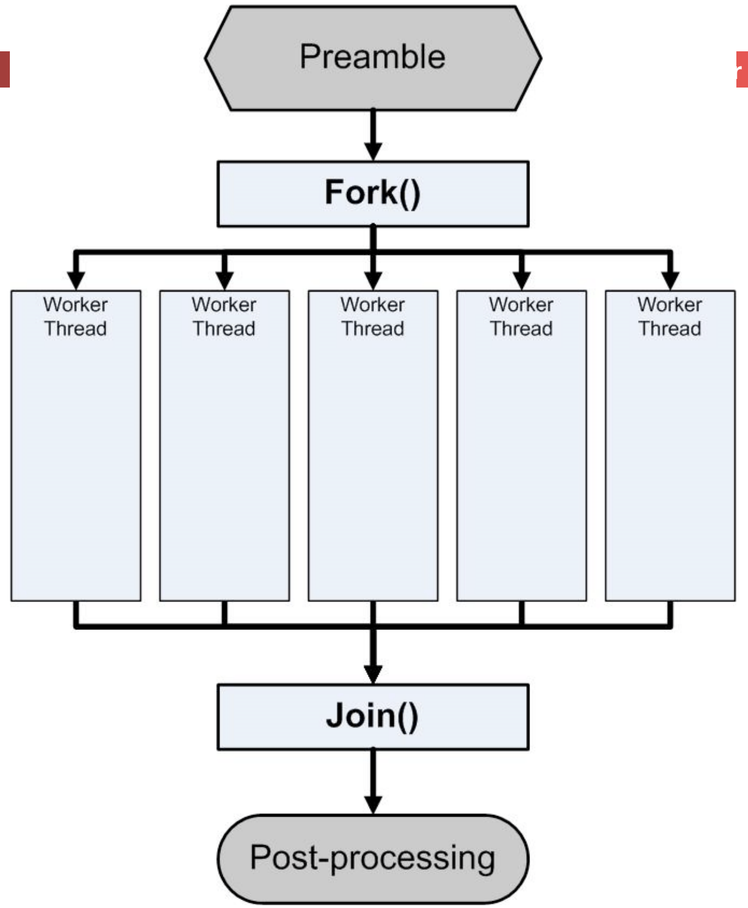
Great Idea #3: Principle of Locality/ Memory Hierarchy

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



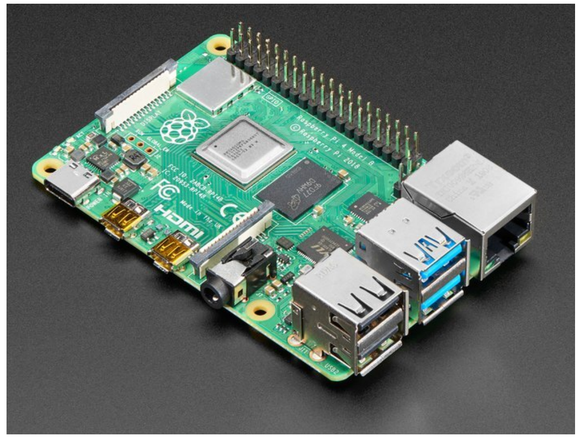
Great Idea #4: Parallelism





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

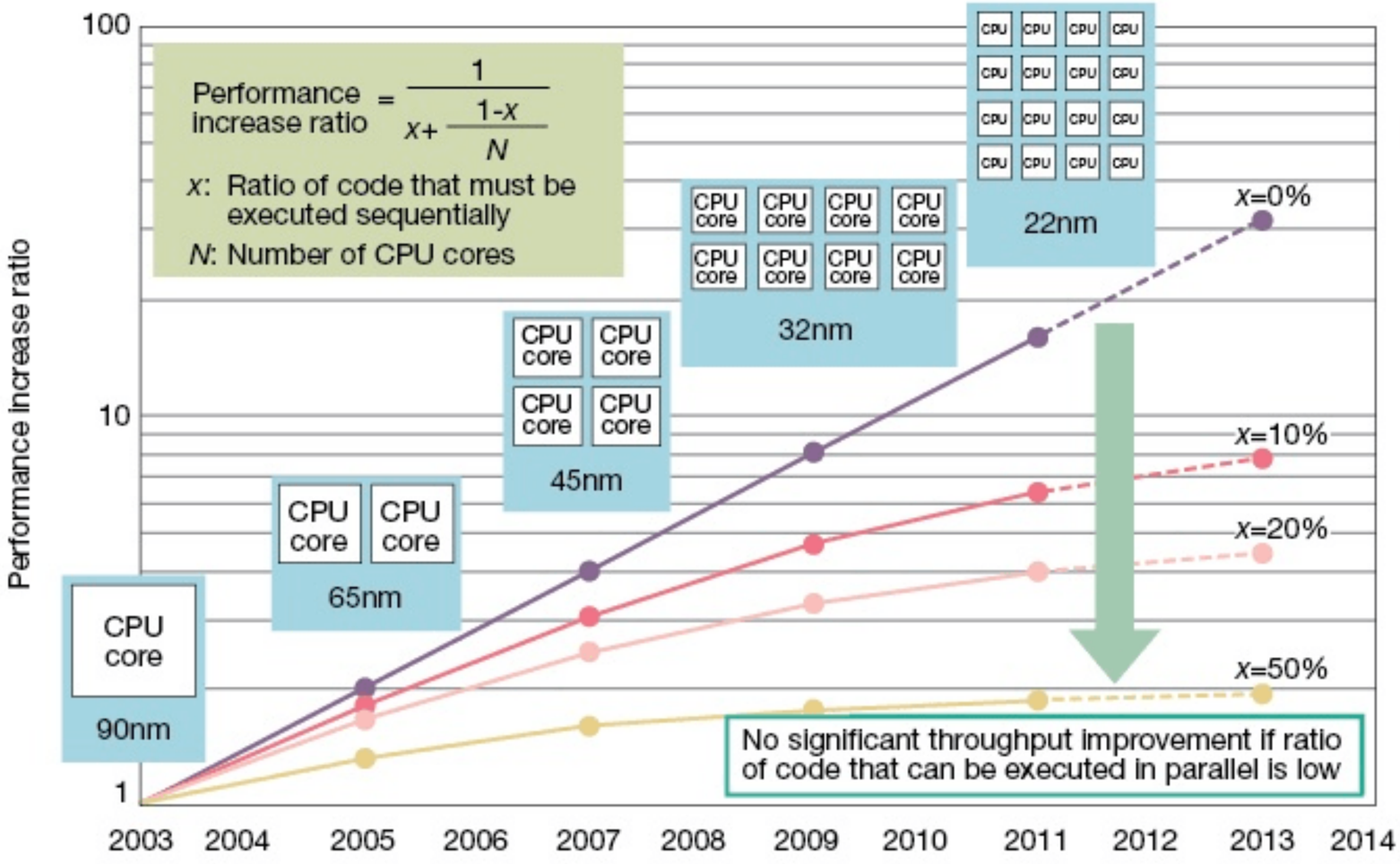


Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor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

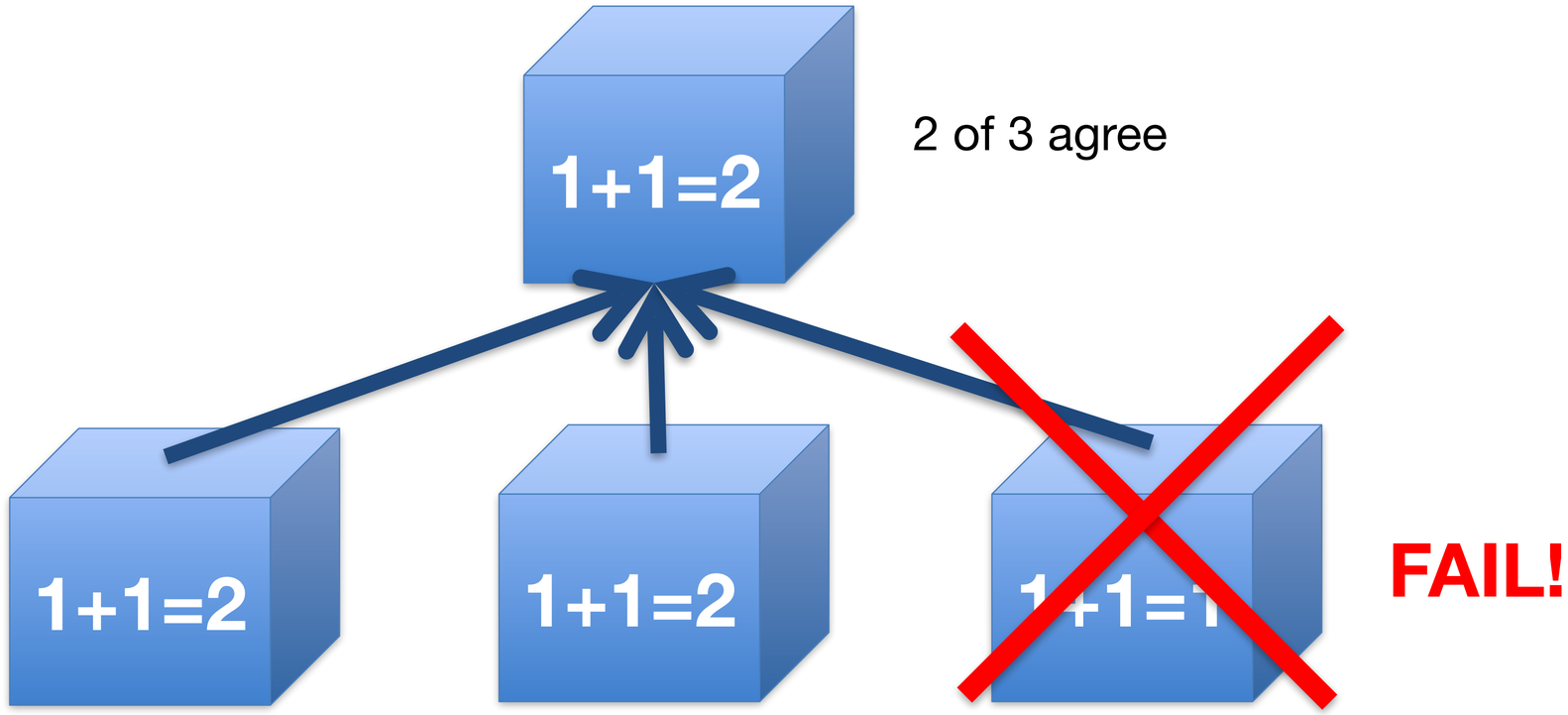
- 4 disks/server, 50,000 servers
- Failure rate of disks: 2% to 10% / year
  - Assume 4% annual failure rate
- On average, how often does a disk fail?

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

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

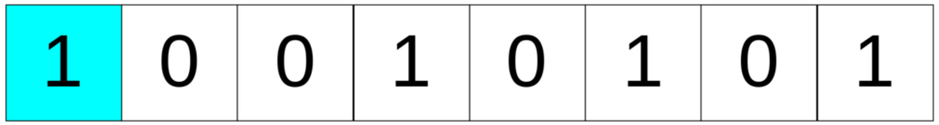
- What you need to know about this class
- Thinking about Machine Structures
- Great Ideas in Computer Architecture
- Number Representation

What is Binary?

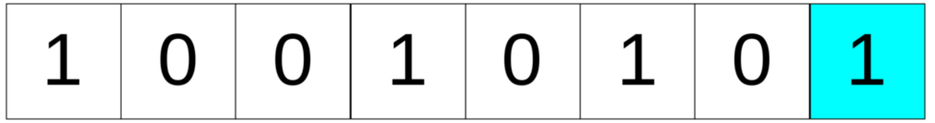
- A method of representing numbers using a string of 0s and 1s
- Why do computers use binary?
  - The fundamental building block of a computer is a transistor which can only represent two values. We decided to label these two values as 0 and 1
- Denoted by prepending a 0b or appending a subscripted 2 to the string

Terms

- Bit
  - 1 binary digit
- Nibble
  - 4 bits
- Byte
  - 8 bits
- Base
  - The number of different digits that a system has to represent numbers
- Most significant bit (MSB)
  - The bit in the highest position



- Least significant bit (LSB)
  - The bit in the lowest position



Binary Table

Leading zeros don't change the value!

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Powers of 2

Decimal	Binary(0b)	Powers of 2
0	0000	
1	0001	2^0
2	0010	2^1
3	0011	
4	0100	2^2
5	0101	
6	0110	
7	0111	
8	1000	2^3
9	1001	
10	1010	
11	1011	
12	1100	
13	1101	
14	1110	
15	1111	

Powers of 2

The exponents correspond to the bits that are 1!

Decimal	Binary(0b)	Powers of 2
0	0000	
1	0001	2^0
2	0010	2^1
3	0011	2^1+2^0
4	0100	2^2
5	0101	2^2+2^0
6	0110	2^2+2^1
7	0111	$2^2+2^1+2^0$
8	1000	2^3
9	1001	2^3+2^0
10	1010	2^3+2^1
11	1011	$2^3+2^1+2^0$
12	1100	2^3+2^2
13	1101	$2^3+2^2+2^0$
14	1110	$2^3+2^2+2^1$
15	1111	$2^3+2^2+2^1+2^0$

Powers of 2

Power	Result
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096

Decimal Notation

$$5072_{10}$$

$$(5 \times 10^3) + (0 \times 10^2) + (7 \times 10^1) + (2 \times 10^0)$$

$$5000 + 0 + 70 + 2$$

$$5072_{10}$$

Binary to Decimal

$$1011_2$$

$$(1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$$

$$8 + 0 + 2 + 1$$

$$11_{10}$$

Decimal to Binary

1. Determine the largest power of 2 that is less than or equal to the current number
2. Put a 1 in that position in the final answer
3. Subtract that power of 2 from the current number
4. Repeat until the current number is 0

```
  114
- 64
-----
   50
-  32
-----
   18
-  16
-----
    2
-   2
-----
    0
```

0b	1	1	1	0	0	1	0
	64	32	16	8	4	2	1

Binary Addition

$$\begin{array}{r} 11 \\ +\ 3 \\ \hline 14 \end{array} \qquad \begin{array}{r} {\color{blue}11} \\ 1011 \\ +\ 0011 \\ \hline {\color{blue}1110} \end{array}$$

Binary Addition - Overflow

- Overflow occurs when you cannot represent the result of the operation in the given number of bits
- When you overflow, you end up with an incorrect result

$$\begin{array}{r} 11 \\ +\ 5 \\ \hline 16 \end{array} \qquad \begin{array}{r} 1111 \\ 1011 \\ +\ 0101 \\ \hline 0000 \end{array}$$

How many numbers can we represent given n bits?

- Each bit can either be a 0 or 1
- With n bits, we can represent 2^n values
- Ex:
  - How many numbers can we represent with 5 bits?
  - XXXXX
  - 2x2x2x2x2
  - 2^5
  - 32
- If our range starts at 0, then we can represent [0, 2^n-1] values

Hexadecimal

- Method of representing binary that's easier for humans to read
- Base 16
  - One hex digit can represent 16 numbers
  - One hex digit = 1 nibble
- Denoted by prepending "0x" or appending a subscripted 16

Decimal	Binary(0b)	Hex(0x)
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Binary to Hex

1. Group the digits into groups of 4 from right to left
2. Prepend any leading zeros needed to make the leftmost group have four binary digits
3. Convert each group to the corresponding hex character

0b111010

0011 1010

3 A

0x3A

Hex to Binary

1. Convert each hex digit to their corresponding binary digits
2. Group the terms together

0x61C

0110 0001 1100

0b011000011100

Decimal to Hex
Hex to Decimal

- Convert to binary then convert to the desired base

Hex to Decimal

$$A90B_{16}$$

$$(Ax16^3) + (9x16^2) + (0x16^1) + (Bx16^0)$$

$$40960 + 2304 + 0 + 11$$

$$43275_{10}$$

Decimal to Hex

1. Determine the largest power of 16 that is less than or equal to the current number (n)
2. Determine how many of this number will fit into the current number (n*b)
3. Subtract (n*b) from the current number
4. Repeat until the current number is 0

$$
\begin{array}{r}
11082 \\
- \quad 2(4096) \\
\hline
2890 \\
- \quad 11(256) \\
\hline
74 \\
- \quad 4(16) \\
\hline
10 \\
- \quad 10 \\
\hline
0
\end{array}
$$

0x

2	B	4	A
4096	256	16	1

Sign and Magnitude

- We need a way to represent negative values
- The MSB represents the sign of the value
  - 0 stands for positive
  - 1 stands for negative
- The remaining bits represent the magnitude of the number
- Ex: Convert -5 to sign and magnitude using 5 bits:
  - Sign = 1
  - Magnitude = 0101
  - Answer = 10101

Range of Sign and Magnitude

- Recall that the range of unsigned representation is [0, 2^n-1]
- What is the range of negative values that we can represent?
  - XXXXX
  - [-(2^{n-1}-1), -0]
- What is the range of positive values that we can represent?
  - XXXXX
  - [+0, 2^{n-1}-1]
- Putting it together
  - [-(2^{n-1}-1), 2^{n-1}-1]

Problems with Sign and Magnitude

- Two zeros
  - Makes hardware and programming implementations more difficult
- It's difficult to implement in hardware because of the sign bit

One’s Complement

- Another way to represent negative numbers
- To form the negative numbers, flip the bits of the positive form of the number
- The MSB represents the sign of the value
  - 0 means its positive
  - 1 means its negative

One's Complement Examples

- Ex1: What is the one's complement representation of -5 using 6 bits?
  - Unsigned: 000101
  - Flip bits: 111010

- Ex2: What's the one's complement representation of 6 using 4 bits?
  - Unsigned 0110
  - No need to flip bits
  - Answer: 0110

- Ex3: Convert the following one's complement number to decimal: 0b11001
  - MSB is 1, so it's negative
  - Flip bits: 0b00110
  - Answer: -6

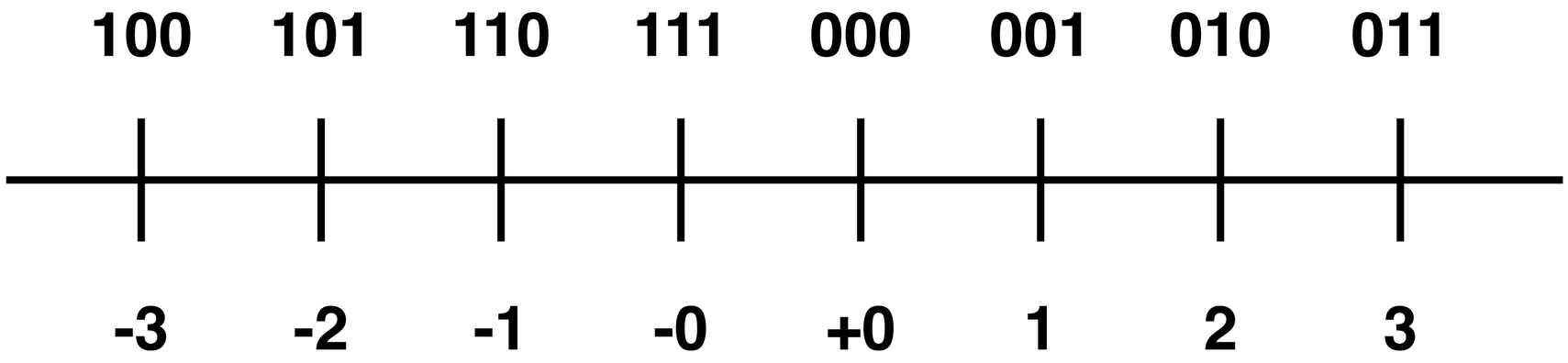
- Ex4: Convert the following one's complement number to decimal: 0b01001
  - MSB is 0, so its positive
  - No need to flip bits
  - Answer: 9

Range of One's Complement

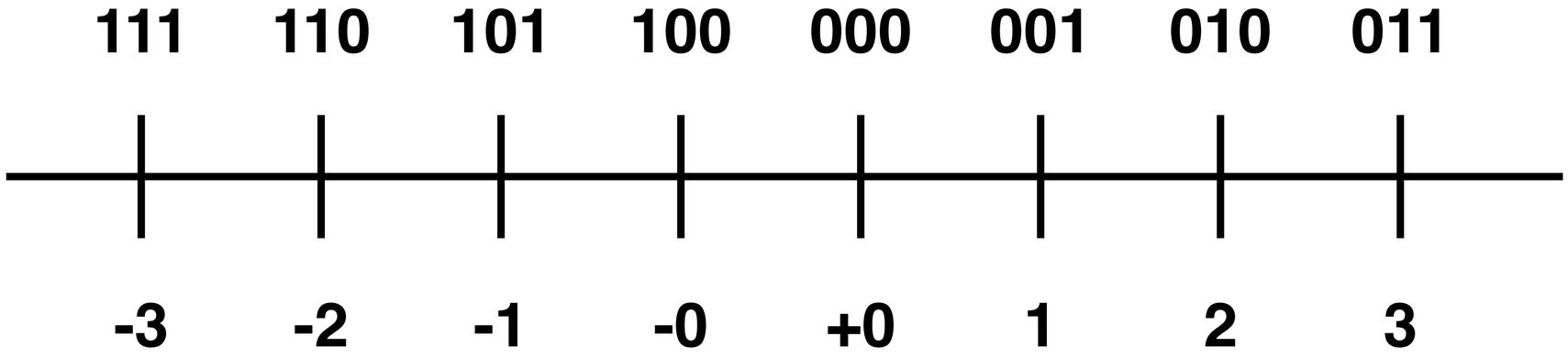
- Recall that the range of sign and magnitude is $[-(2^{n-1}-1), 2^{n-1}-1]$
- What is the range of negative values that we can represent?
  - XXXXX
  - Same as sign and magnitude
  - $[-(2^{n-1}-1), -0]$
- What is the range of positive values that we can represent?
  - XXXXX
  - Same as sign and magnitude
  - $[+0, 2^{n-1}-1]$
- Putting it together
  - $[-(2^{n-1}-1), 2^{n-1}-1]$

One’s Complement vs Sign and Magnitude Number Lines

One’s Complement Number Line



Sign and Magnitude Number Line

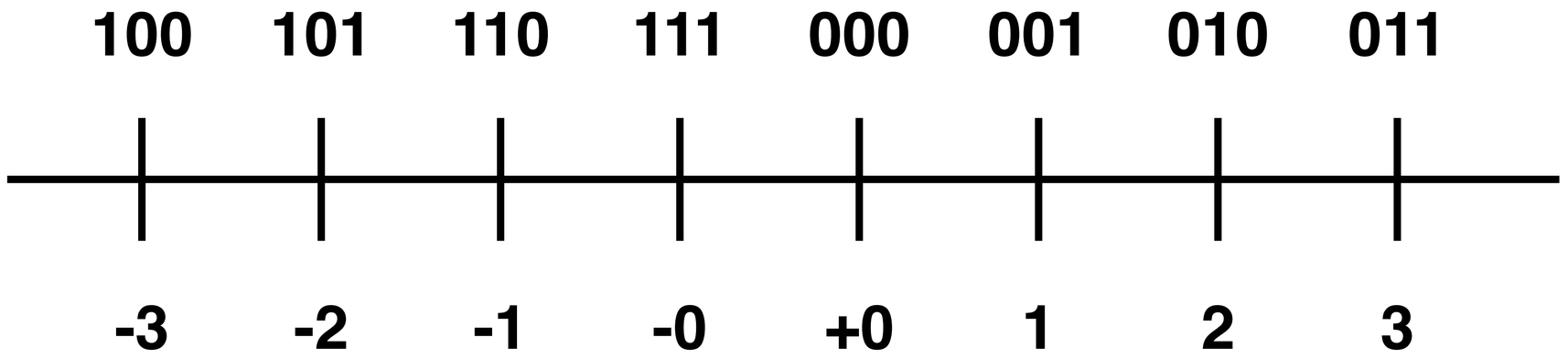


One’s Complement Shortcomings

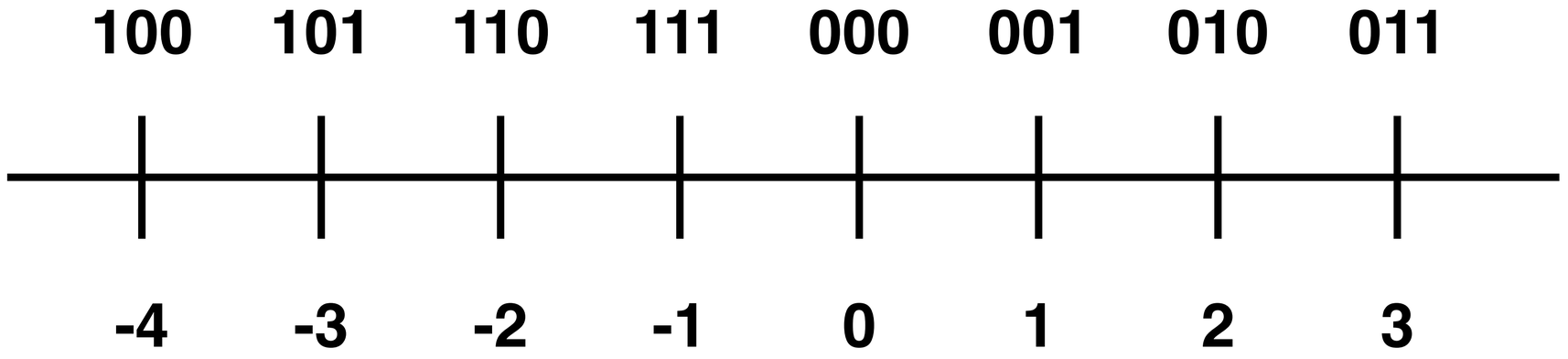
- Still have two zeros

Two's Complement

One's Complement Number Line



Two's Complement Number Line



Two's Complement

- How do we do this arithmetically?
  - Flip the bits and add one
- The MSB represents the sign of the value
  - 0 means its positive
  - 1 means its negative
- We only have one zero now!
- Easier to implement in hardware

Two's Complement Examples

- Ex1: Represent -7 in two's complement with 5 bits
  - Unsigned: 0b00111
  - Flip bits: 0b11000
  - Add one: 0b11001

- Ex2: Represent 10 in two's complement with 6 bits
  - Unsigned: 0b001010
  - Positive, so no need to flip bits
  - Answer: 0b001010

- Ex3: Convert the following two's complement number to decimal: 0b11010
  - MSB is 1, so it's negative
  - Flip bits: 0b00101
  - Add one: 0b00110
  - Answer: -6

- Ex4: Convert the following two's complement number to decimal: 0b01111
  - MSB is 0, so it's positive
  - No need to flip bits
  - Answer: 15

Two's Complement Range

- Recall that the range for one's complement is $[-(2^{n-1}-1), 2^{n-1}-1]$
- What is the range of negative values we can represent?
  - Remember that we shifted the negative values over by one, meaning that we can represent one more negative value
  - $[-2^{n-1}, 0]$
- What is the range of positive values we can represent?
  - Same as one's complement
  - $[0, 2^{n-1}-1]$
- Putting it together
  - $[-2^{n-1}, 2^{n-1}-1]$

Two’s Complement Addition

- Arithmetic in two’s complement works!

Two's Complement Addition

$$\begin{array}{rr} & -2 \\ + & -4 \\ \hline & -6 \end{array}$$

$$\begin{array}{rr} & {\scriptstyle 11} \\ & 1110 \\ + & 1100 \\ \hline & 1010 \end{array}$$

0101

0110

Two's Complement Overflow

- Overflow: when the result of an operation cannot be represented in the given number of bits
- When adding two positive numbers, overflow occurs when the result is negative
- When adding two negative numbers, overflow occurs when the result is positive
- Overflow will never occur when adding two numbers of opposite signs

Two's Complement Overflow Examples


```
  111
  0111
+ 0101
------
  1100
```

Overflow

```
  111
  1111
+ 0110
------
  0101
```

No Overflow

```
 1
  1000
+ 1001
------
  0001
```

Overflow

```
 1111
  1101
+ 1011
------
  1000
```

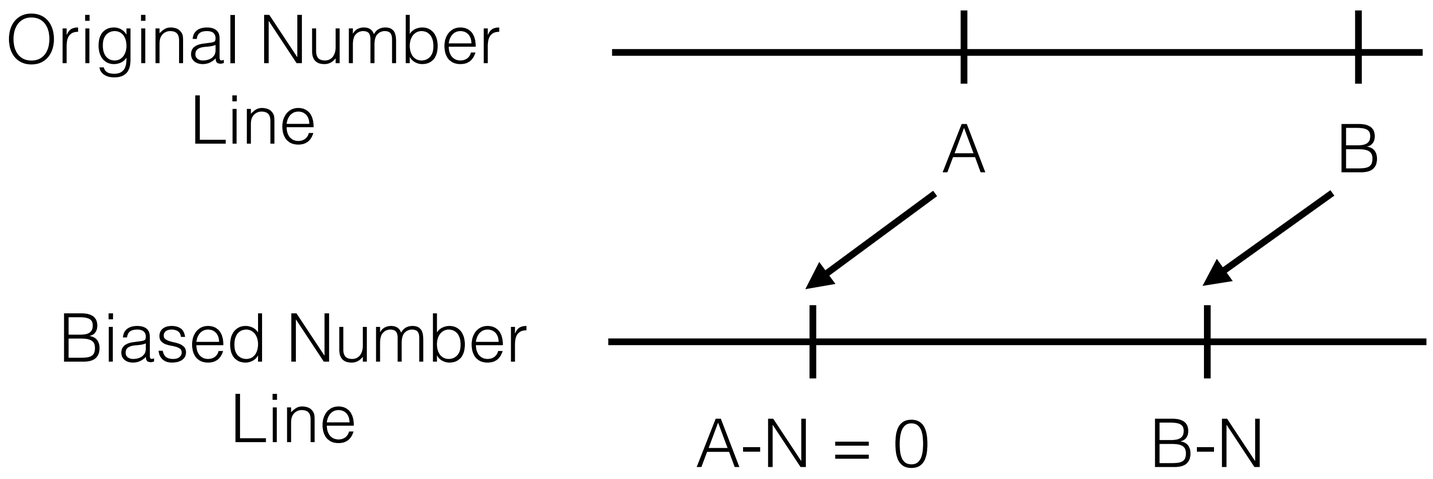
No Overflow

Bias Encoding

- A method for storing a range of values where the lowest value is encoded as all zeros
- We'll see an application of this in the floating point lecture

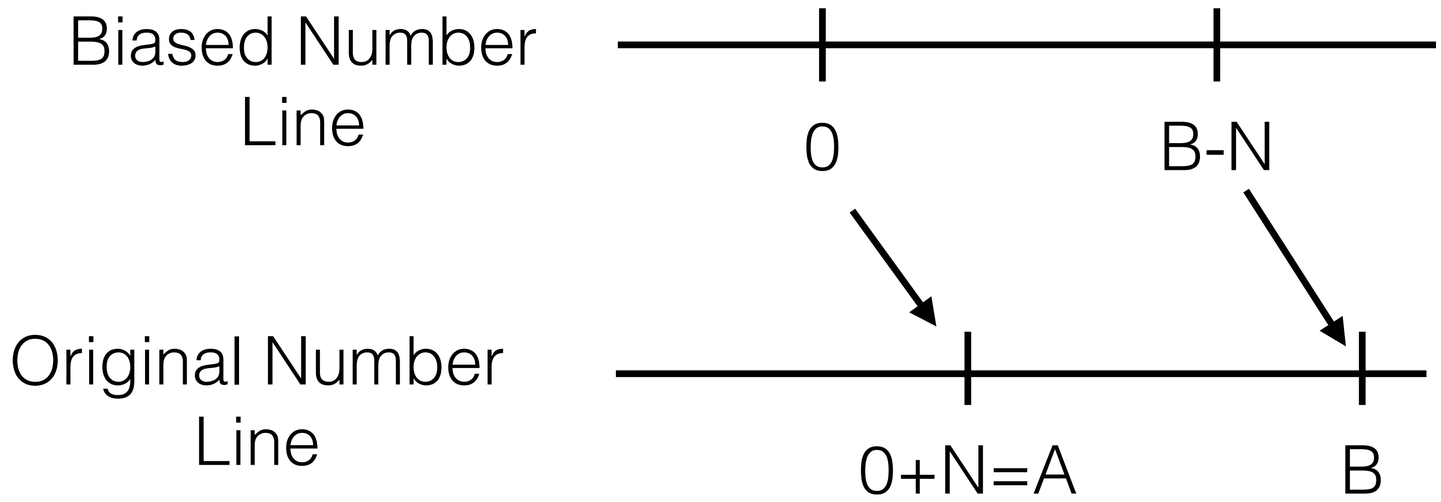
Bias Encoding (unbiased -> biased steps)

1. Translate the number so that the lowest value is at zero
   - The amount that we translate by is called the bias (N)
2. Convert the number to binary



Bias Encoding (biased -> unbiased steps)

1. Convert the number to decimal
2. Add the bias

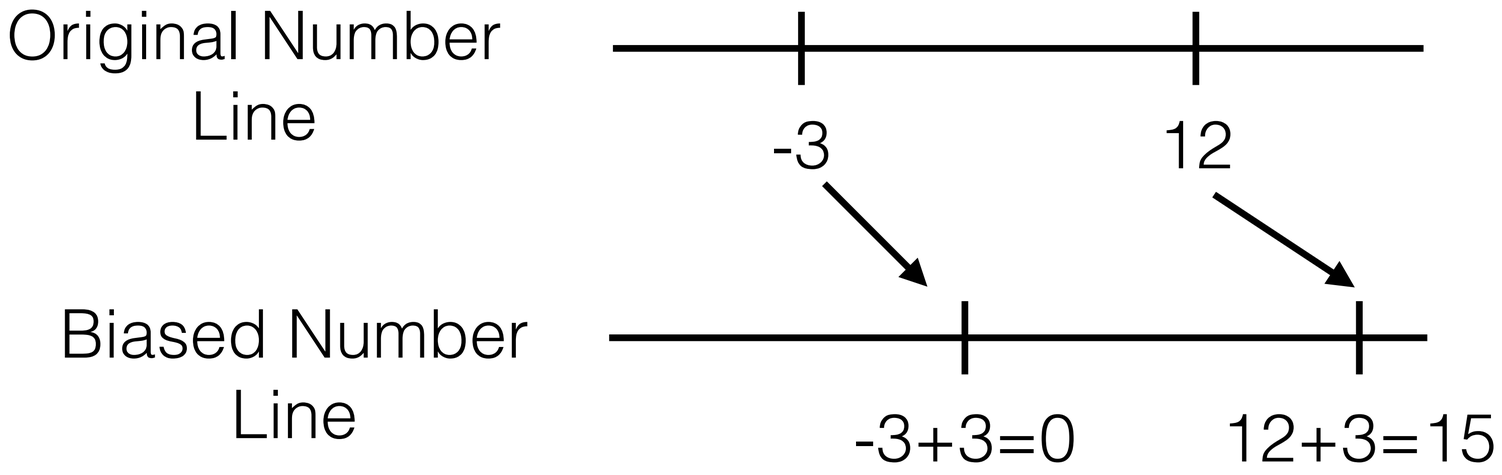


Bias Encoding Convention

- Non-biased encoding -> bias encoding
  - Subtract the bias
- Biased encoding -> non-biased encoding
  - Add the bias

Bias Encoding

- You can set the bias to be anything you want
- If you want to represent the [-3, 12], you can set the bias to -3 in order to make the range be [0, 15]
  - This enables you to represent the range [-3, 12] with only 4 bits!



Calculating the Bias (for 2's complement)

- Recall that the range of a 2's complement number with n bits is $[-2^{n-1}, 2^{n-1}-1]$
- Bias Formula for a 2's complement number with n bits
  - $N = -(2^{n-1}-1)$
- Hold on, $-2^{n-1} - -(2^{n-1}-1) = -1$. That's not zero
  - In floating point representation, all ones is used for a special case

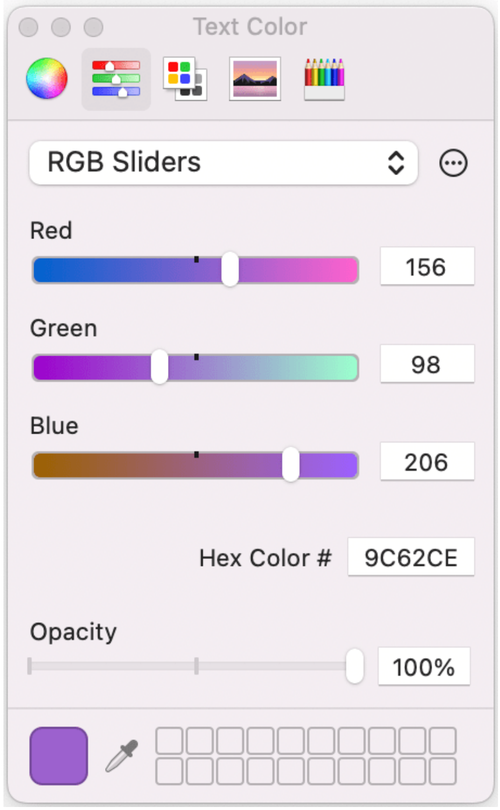
Bias Examples (2's Complement)

- Covert -9 to bias notation with 5 bits
  - Bias = $-(2^{n-1}-1)$
  - Bias = $-(2^{5-1}-1)$
  - Bias = -15
  - -9 + 15 = 6
  - Bias Encoding = 00110

- What number is being represented in this biased encoding? 0b1010
  - Bias = $-(2^{n-1}-1)$
  - Bias = $-(2^{4-1}-1)$
  - Bias = -7
  - 10 - 7 = 3

Big Idea

- Bits can represent anything, you can make them mean anything you want



Powers of 2

Power	Result
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096

SI Prefixes

Name	Abbreviation	Power of 2	Power of 10
Kilo	K	2^{10}	10^{3}
Mega	M	2^{20}	10^{6}
Giga	G	2^{30}	10^{9}
Tera	T	2^{40}	10^{12}
Peta	P	2^{50}	10^{15}
Exa	E	2^{60}	10^{18}
Zetta	Z	2^{70}	10^{21}
Yotta	Y	2^{80}	10^{24}

IEC Standard Prefixes

Name	Abbreviation	Power of 2
kibi	Ki	2^{10}
mebi	Mi	2^{20}
gibi	Gi	2^{30}
tebi	Ti	2^{40}
pebi	Pi	2^{50}
exbi	Ei	2^{60}
zebi	Zi	2^{70}
yobi	Yi	2^{80}

Converting powers of 2 to IEC notation

- 2^{XY}
  - X tells us the prefix
  - Y tells us the number that precedes the prefix (ex: 2 Ti)

X = 0 => no prefix	Y = 0 => 1
X = 1 => kibi	Y = 1 => 2
X = 2 => mebi	Y = 2 => 4
X = 3 => gibi	Y = 3 => 8
X = 4 => tebi	Y = 4 => 16
X = 5 => pebi	Y = 5 => 32
X = 6 => exbi	Y = 6 => 64
X = 7 => zebi	Y = 7 => 128
X = 8 => yobi	Y = 8 => 256

IEC Examples

- Write the following in IEC notation: 2^{75}
  - X = 7 => zebi
  - Y = 5 => 32
  - Answer: 32 Zi

- Write the following as a power of 2: 64 Ti
  - Ti => X = 4
  - Y = $\log_2(64) = 6$
  - Answer: 2^{46}

Additional Problems

- Unfortunately, we cannot go over all of the examples needed to fully cover this topic due to the large amount of material that we are required to teach in the course
- Please attend discussion or review the discussion worksheet for more examples including
  - Two's complement subtraction
  - Hex arithmetic

Summary

- Binary, Decimal, Hex
- Sign and Magnitude
- One’s Complement
- Two’s Complement
- Bias
- IEC Prefixes