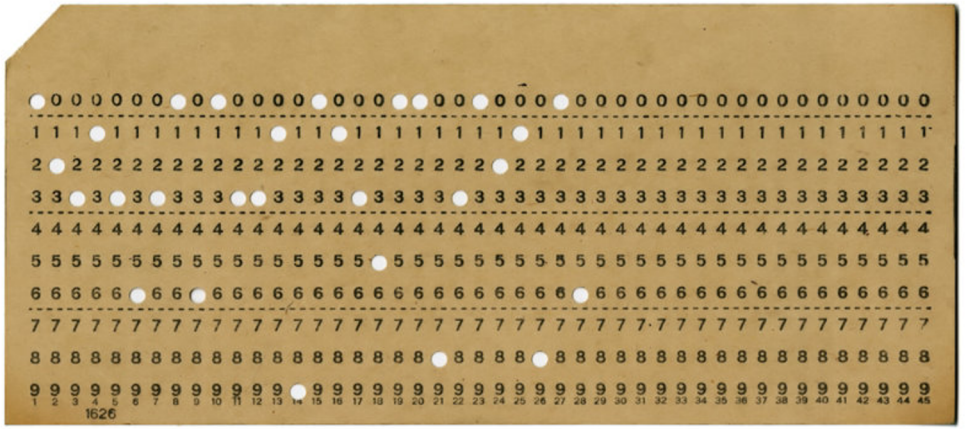
Great Ideas in Computer Architecture

Course Intro, Number Representation

Instructors: Stephan Kaminsky,
Sean Farhat, Jenny Song



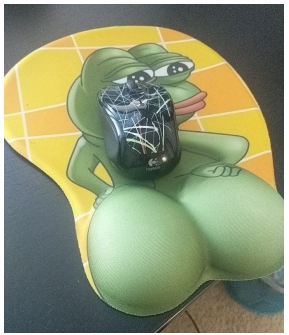
Introducing Stephan (he/him)

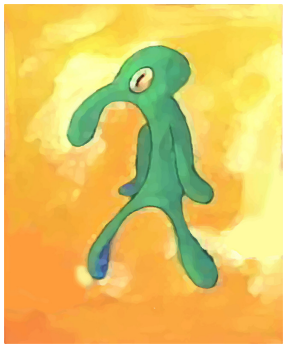
- Instructor, “the one who goes zoom zoom”
- **Upbringing:** Born and raised in Long Beach
- **Education:** New grad ;=; but doing a 5th Year Masters in EECS @ UCB!
- **Teaching:** 61C course staff for 6 semesters, head TA last spring and fall
- **Research:** Secure enclaves with RISC-V (Shameless plug https://keystone-enclave.org/)
- **Interests:** Riding motorcycles, petting dogs, video games

Keystone

Introducing Sean (he/him)

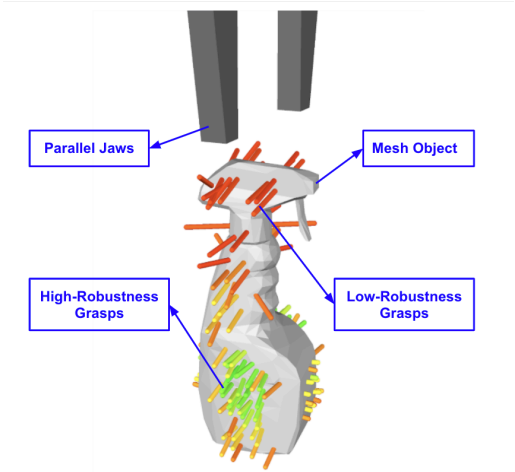
- Instructor, the one who can never write a normal bio
- **Upbringing:** Newport Beach, home of the 👍 "sah dude"
- **Education**: UC Blockeley graduate. Going to UIUC for 2 years to become a Master… (then Ph.D? o.O who knows)
- **Research**: Optimization, computational creativity
- **Teaching**: 61C (this class!), 127 (Optimization)
- **Interests**: mémés, art, music, r/hydrohomies





Introducing Jenny (she/her)

- **Upbringing:** Born in Hangzhou, China, High school in New Jersey
- **Education:** Rising Senior
- **Teaching:**
  - TA for 61C for 3 semesters
- **Research:** @Autolab Task-Based Grasping
- **Interests**: Binge Watching Shows, Bake



KILLING EVE

WESTWORLD

Introducing Your TAs

Sunay Poole

Caroline Liu

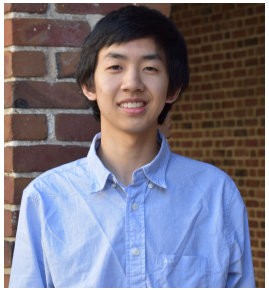
Cynthia Zhong

Daniel Fan

Ivy Li

Jerry Xu

Jie Qiu



Justin Cheng

Ryan Searcy

Justin Yokota

Kimberly Zhu

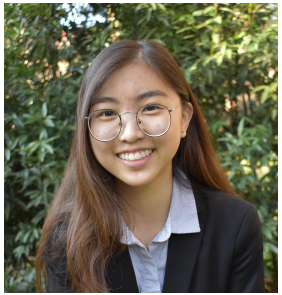
Max Lister

Robin Chu

Introducing Your Tutors

Edward Zeng

Emily Wang



Kevin Chang

Kevin Zhu

Nareg Megan

Troy Sheldon

Vincent Chiang

Yijie Huang

Agenda

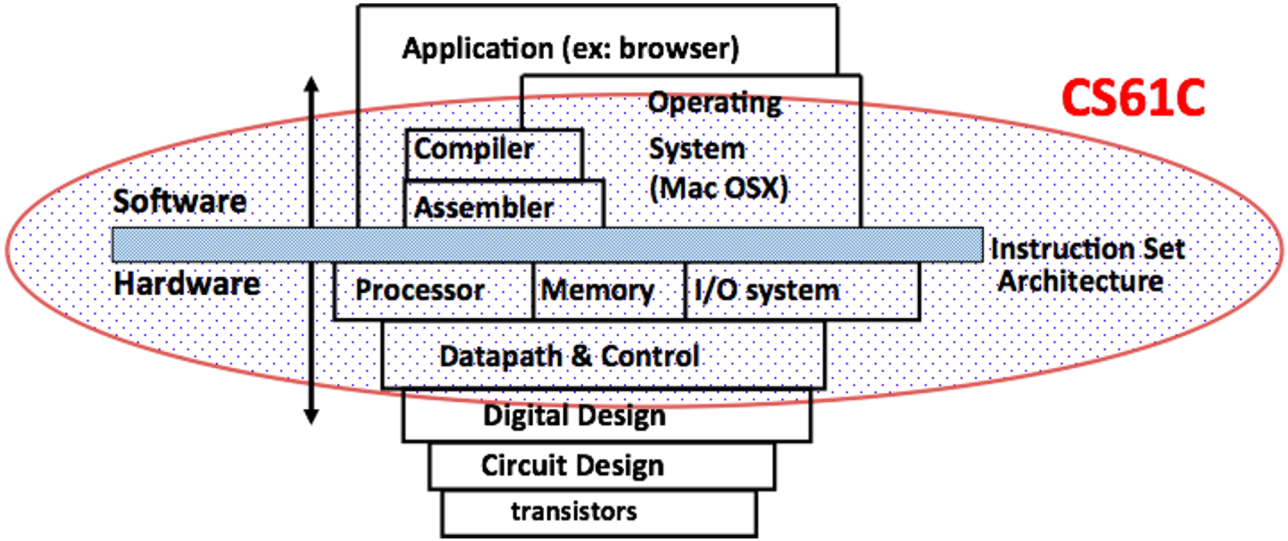
- Course Overview
- Course Policies
- Number Representation
  - Number Bases
  - Signed Representations
  - Overflow
  - Sign Extension

Course Goals

1. How do computer processors and memories work, and how do they affect software design and performance?

2. Introduction to “computer systems” areas: architecture, compilers, security, embedded, operating systems, large-scale computing

Hardware-Software Interface



CS61C For Software Development

- Know the tools of the trade – computers!
  - "Computers" come in all shapes and sizes
  - Computing achieved in many different ways nowadays
- Know when performance matters
  - Ex: taking advantage of parallelism
- Understand the differences between programming languages under the hood
- Design large systems – abstraction in hardware
- Security
- Design methodology – limitations and tradeoffs

Course Learning Objectives

- After taking this class students should be able to:
  - ✓ Identify and explain the various layers of abstraction that allow computer users to perform complex software tasks without understanding what the computer hardware is actually doing
  - ✓ Judge the effect of changing computer components (e.g. processor, RAM, HDD, cache) on the performance of a computer program
  - ✓ Explain how the memory hierarchy creates the illusion of being almost as fast as the fastest type of memory and almost as large as the biggest memory
  - ✓ Construct a working CPU from logic gates for a specified instruction set architecture
  - ✓ Identify the different types of parallelism and predict their effects on different types of applications

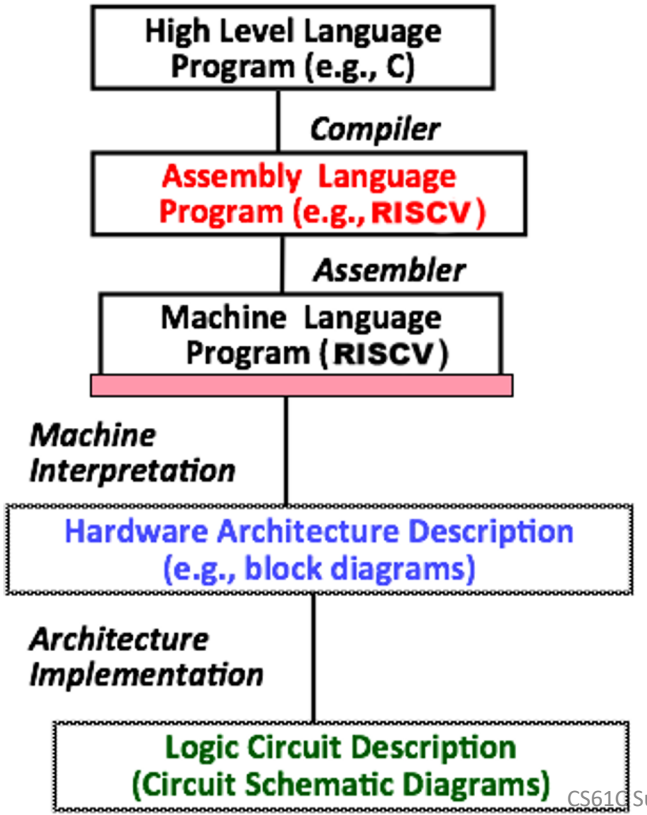
Course Learning Objectives

- In addition, this class will requires students to work on the following skills:
  - Creating and modifying designs to meet a given set of *specifications*
  - Identifying unexpected or problematic situations using debugging tools, and creating test cases to ensure proper behavior
  - Defending design choices based on tradeoffs and limitations

Six Great Ideas in Computer Architecture

1) Abstraction
2) Technology Trends
3) Principle of Locality/Memory Hierarchy
4) Parallelism
5) Performance Measurement & Improvement
6) Dependability via Redundancy

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

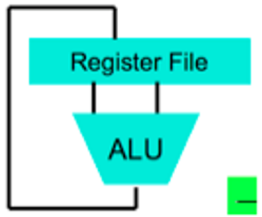



```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

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

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

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



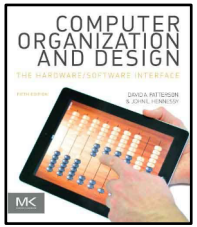
Agenda

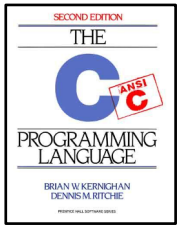
- Course Overview
- Course Policies
- Number Representation
  - Number Bases
  - Overflow
  - Signed Representations
  - Sign Extension

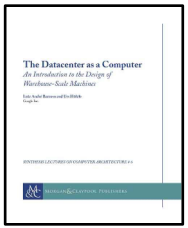
Course Information

More info found on the course policies page

- **Website:** http://cs61c.org/su20
  - Check for assignments, weekly schedule, contact info
- **Textbooks:**







- **Piazza:** http://piazza.com/class#summer2019/cs61c
  - Will have every announcement, discussion, clarification
- **Gradescope**
  - Submit assignments, check grades
- **Inst account:** http://inst.eecs.berkeley.edu/webacct
  - Access instructional ("hive") machines for project and lab work

Course Assignments and Grading

- **Lecture** (5%) – On Gradescope/YouTube
- **Labs** (5%) – 12 total
- **Homework** (10%) – On Gradescope
- **Projects** (40%) – 4 total, weighted equally
  - Proj1 (10%) – Proj2 (10%)
  - Proj3 (10%) – Proj4 (10%)
- **Exams** (40%)
  - **Midterm 1** (10%): Thursday, July 9th @ 9 AM PDT
  - **Midterm 2** (10%): Wednesday, July 29th @ 9 AM PDT
  - **Final** (20%): Thursday, August 13th @ 9 AM PDT

Exam Clobbering & Conflicts

- People add this class late!
  - Midterm 1 is clobber-able with your score on the final exam
  - MT2 is clobber-able with your score on the final exam
  - NOTE: Clobbering is z-score clobbering, meaning if you get a 80% on the final, it doesn't mean your mt1 score is also 80%.
  - Involves standard deviations, the mean, and your performance wrt to them
- Exam conflict/timezone issue? Follow the directions on Piazza!

Lab Grading Policy

- Out of 2 points. A new lab is assigned every Tue/Thu
  - In order to receive credit for a lab, you must get it checked off by a staff member (TA or AI). **You MUST have a partner!**
  - For full credit, Tu labs must be checked off before Th and Th labs before the next Tu
    - You will receive 0 points if you fail a checkoff
  - Lab checkoffs will be by 13 min appointment (Released on Monday @ 9AM for the week).
    - You may ONLY sign up for one appointment slot per lab—even if you fail to get checked off!
    - If you miss your appointment or need to do a late lab checkoff, you may do it during lab OH though there are no guarantees we will get to you as we will focus on lab questions first. You will have the lowest priority on the queue. In other words, if you get checked off in lab office hours, it is not possible to get full credit on the lab (unless you were granted an extension due to DSP or other accommodations).
  - You will receive half credit if you get it checked off the section AFTER it is due.
    - If your lab is late, you have one checkoff attempt
  - Anything submitted after the ½ credit deadline is ZERO credit, but we still recommend completing them!

Project Late Policy – Slip Days

- Project submissions due at 11:59:00pm
- Late penalty is 33% deduction of score per day
- You are given 3 slip day tokens (auto optimally applied)
  - Will be distributed amongst your late submissions at the end of the semester to maximize your grade
  - No benefit to having leftover tokens
  - Tokens are in "1-day" units (not hours, minutes…)
- Use at own risk – don't want to fall too far behind
  - These are meant to be a method for you to get extra time without prior approval. If you request an extension and have not used up your slip days, we will not give you an extension!

EPA

- Encourage class-wide learning!
- Effort
  - Attendance and active engagement in additional course events (eg: office hours, guerilla sessions, project parties, tutoring, etc)
- Participation
  - Attendance and active engagement in required course events (eg: discussion, lab…)
- Altruism
  - Helping others anywhere (eg: lab, discussion, OH and Piazza)

EECS Grading Policy

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

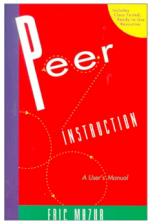
  "A typical GPA for a lower division course will fall in the range 2.8 - 3.3, depending on the course and the students who enroll. For example, a GPA of 3.0 would result from 35% A's, 45% B's, 13% C's, and 7% D's and F's."

- Spring 2020: GPA 3.33
- 41% A's, 52% B's, 1.9% C's, 1.3% D's, 1.9% F's
- Graded using bins, but they often move!

Raw Score	290+	[270,290)	[260,270)	[250,260)	[230,250)	[220,230)	[210,220)	[190,210)	[180,190)	[140,180)	[0,140)
Grade	A+	A	A-	B+	B	B-	C+	C	C-	D	F

- If the grade bins result in an average GPA that is too low, the course will be curved to match department guidelines BEFORE adding EPA.

Lectures/Peer Instruction



- Increase real-time learning in lecture, test your understanding, increase student interactions
  - Lots of research supports its effectiveness
- Different type questions sprinkled throughout lecture
  - There will be a few questions per lecture video
- You will be graded on **CORRECTNESS.**
  - You have unlimited attempts and instant feedback! ☺

Policy on Assignments and Independent Work

- All submissions are expected to be yours and yours alone
- You are encouraged to discuss your assignments with other students (*ideas*), but we expect that what you turn in is yours
- It is NOT acceptable to copy solutions from other students
- It is NOT acceptable to copy (or start your) solutions from the Web (including Github)

Policy on Assignments and Independent Work

- We have tools and methods, developed over many years, for detecting this. You WILL be caught, and the penalties WILL be severe.
- Both the *cheater* and the *enabler* receive –100% for the assignment. Letter to your university record documenting the incidence of cheating.
  - IT IS BETTER TO NOT DO THE ASSIGNMENT THEN TO CHEAT

People are caught every semester of 61C

Course Resources

- Weekly Guerrilla Sessions (synchronous but recorded)
  - Led by tutors
  - Review sessions coming exam problems on specific topics
- Small-group tutoring (synchronous NOT recorded)
  - Sign up for a weekly tutoring session w/ a tutor
  - Groups are as small as 5-10 people; personalized help
  - Priority for DSP, EOP students, open enrollment after; private post on piazza under ‘tutoring’ with EOP/DSP proof for early enrollment link
- Project Parties
  - (Near) weekly events staffed by TAs, and possibly lab assistants to help you with your projects
- Office hours
  - Please come chat with us about any questions or concerns!

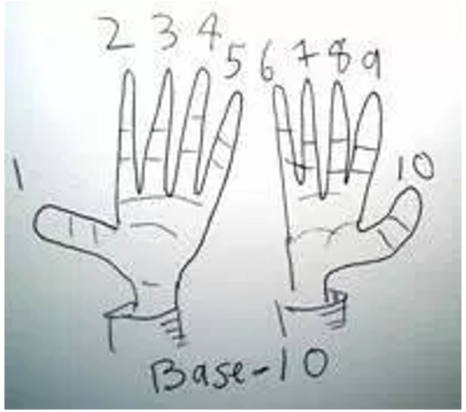
Successful Behaviors

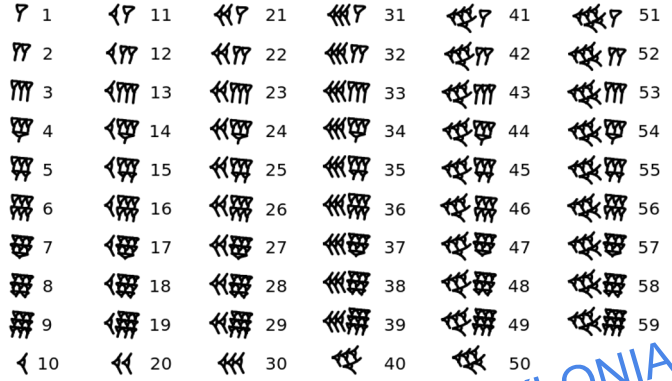
- Practice, practice, practice
  - Learn by doing: deep learning doesn't happen in lecture (and it shouldn't!)
  - Growth mindset: success through effort and repetition
- Find a learning community
  - Learning is much more fun with friends who have similar skill levels
  - Learn via teaching or different perspectives
- Avoid comparison: do your best, and judge yourself on your progress alone.
  - Remember, we are a bucketed course and will NOT purposely make assignments harder if everyone is doing well!
- You learn best from your mistakes
  - Don't be afraid to be wrong; you lose if you remain silent

Agenda

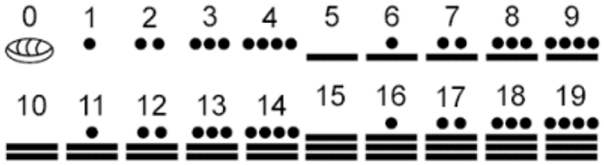
- Course Overview
- Course Policies
- Number Representation
  - **Number Bases**
  - Signed Representations
  - Overflow
  - Sign Extension

Number Representation





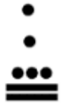
BABYLONIANS



Example:

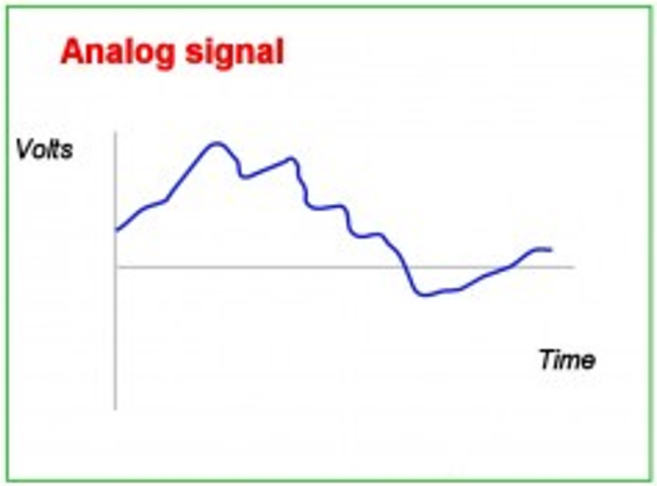
28 = (1 x 20) + 8 =

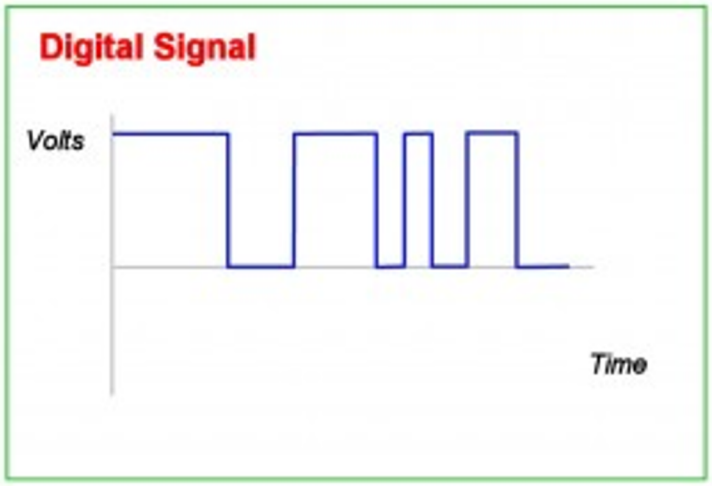
433 = (1 x 400) + (1 x 20) + 13 =



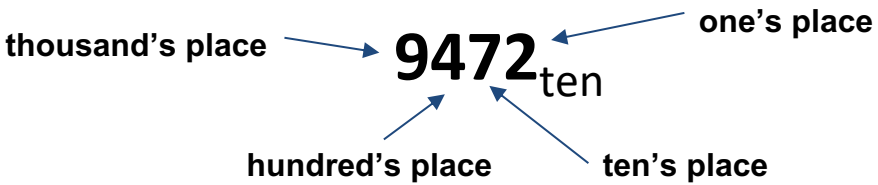
MAYANS

Computers: Base 2





Number Base Examples



$$\mathbf{9472}_{\text{ten}} = 9000 + 400 + 70 + 2$$
$$= 9\text{x}1000 + 4\text{x}100 + 7\text{x}10 + 2\text{x}1$$
$$= 9\text{x}10^3 + 4\text{x}10^2 + 7\text{x}10^1 + 2\text{x}10^0$$

General Formula:

$d_{n-1}d_{n-2} \ldots d_1d_0$ (n-digit number in base B)

$$= d_{n-1}\times B^{n-1} + d_{n-2}\times B^{n-2} + \ldots + d_1\times B^1 + d_0\times B^0$$

a digit d can take on values from 0 to B - 1

Commonly Used Number Bases

- **Decimal** (base 10)
  - Symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Notation: 9472_{ten} = 9472
- **Binary** (base 2)
  - Symbols: 0, 1
  - Notation: 101011_{two} = 0b101011
- **Hexadecimal** (base 16)
  - Symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - Notation: $2A5D_{hex}$ = 0x2A5D

Decimal	Binary	Hex
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

Question

Convert **0b11110** to **decimal.**

(A) 11

(B) 30

1*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 0*2^0

(C) -2

(D) I'm so lost

(E) Who even cares?


CAUTION:
BIG IDEA AHEAD
WHAT DOES IT ALL MEAN?!?!
quickmeme.com

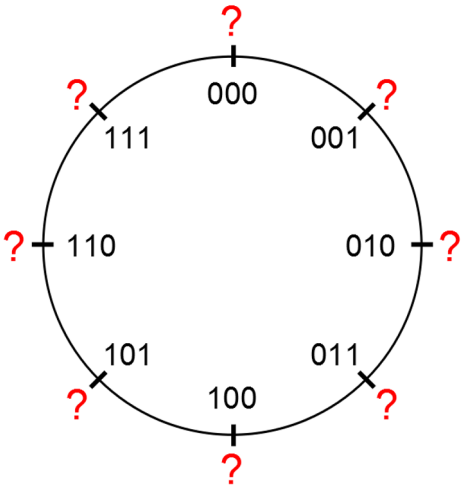
BIG Idea:

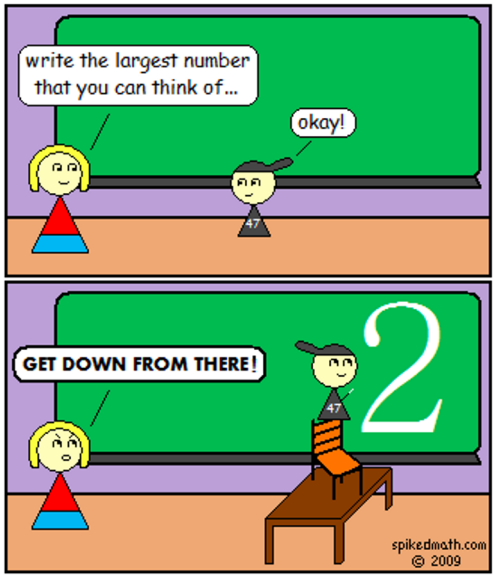
Bits Can Represent ANYTHING

- ...entirely depending on our *interpretation*
- **Characters?!**
  - 26 letters require 5 bits ($2^5 = 32 > 26$)
  - upper/lower case + punctuation → 7 bits ("ASCII")
- **Logical values?!** 0 → False, 1 → True
- **Students in this class?!** 2^9 = 512 > 330 → 9 bits
- **Colors?!** **Red (00)** **Green (01)** **Blue (11)**
- ONLY ONE constraint:

  n digits (base B) ⇒ <= B^n things
  - Each of the n digits is one of B possible symbols
  - Have more things? Add more digits!

How do we assign values to the Binary numerals available to us?





Unsigned Integers

Represent only non-negative (unsigned) integers:

$000_{two} = 0_{ten}$
$001_{two} = 1_{ten}$
$010_{two} = 2_{ten}$
$011_{two} = 3_{ten}$
$100_{two} = 4_{ten}$
$101_{two} = 5_{ten}$
$110_{two} = 6_{ten}$
$111_{two} = 7_{ten}$

Zero?

$0…0_{two} = 0_{ten}$

Most neg number?

$0…0_{two} = 0_{ten}$

Most pos number?

$1…1_{two} = (2^n-1)_{ten}$

Increment?

Agenda

- Course Overview
- Course Policies
- Number Representation
  - Number Bases
  - **Signed Representations**
  - Overflow
  - Sign Extension

Signed Integers

- n bits ⇒ 2^n things
  - —half positive, half negative?
- First logical step:

  +534 -256
- Let the first bit be interpreted as a sign!

Sign and Magnitude

"first" bit gives sign, rest treated as unsigned (magnitude):

$000_{two} = +0_{ten}$
$001_{two} = +1_{ten}$
$010_{two} = +2_{ten}$
$011_{two} = +3_{ten}$

$100_{two} = -0_{ten}$
$101_{two} = -1_{ten}$
$110_{two} = -2_{ten}$
$111_{two} = -3_{ten}$

Zero?

$0\ldots0_{two}$ and $10\ldots0_{two} = \pm 0_{ten}$

TWO ZEROS?!

Most pos number?

$01\ldots1_{two} = (2^{(n-1)}-1)_{ten}$

Most neg number?

$1\ldots1_{two} = -(2^{(n-1)}-1)_{ten}$

Increment?

Biased Notation

Like unsigned, but "shifted" so zero is (roughly) in the middle: (value = "unsigned value" - bias)

PRO

$000_{two} = -3_{ten}$

$001_{two} = -2_{ten}$

$010_{two} = -1_{ten}$

CON $011_{two} = 0_{ten}$ ← Conventional Bias: $(2^{n-1}-1)$

$100_{two} = +1_{ten}$

$101_{two} = +2_{ten}$

$110_{two} = +3_{ten}$

$111_{two} = +4_{ten}$

PRO

Zero?

$01\ldots1_{two} = 0_{ten}$

Most neg number?

$0\ldots0_{two} = -(2^{(n-1)}-1)_{ten}$

Most pos number?

$1\ldots1_{two} = 2^{(n-1)}{}_{ten}$

Increment?

PRO just like unsigned (sorta)

One's Complement

New negation procedure – complement the bits:

$000_{two} = +0_{ten}$
$001_{two} = +1_{ten}$
$010_{two} = +2_{ten}$
$011_{two} = +3_{ten}$

$100_{two} = -3_{ten}$
$101_{two} = -2_{ten}$
$110_{two} = -1_{ten}$
$111_{two} = -0_{ten}$

Zero?

$0\ldots0_{two}$ and $1\ldots1_{two} = \pm0_{ten}$

TWO ZEROS AGAIN……

Most neg number?

$10\ldots0_{two} = -(2^{(n-1)}-1)_{ten}$

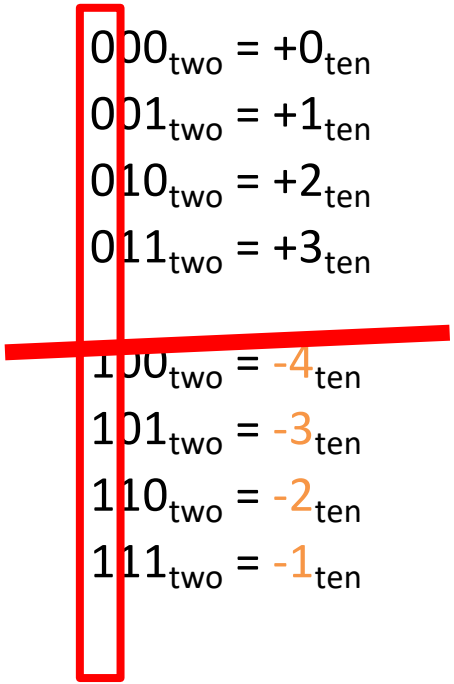
Most pos number?

$01\ldots1_{two} = (2^{(n-1)}-1)_{ten}$

Increment?

Two's Complement

Like One's Complement, but "shift" negative #s by 1:



Zero?

$0\ldots0_{two} = 0_{ten}$

Most neg number?

$10\ldots0_{two} = -2^{(n-1)}{}_{ten}$

Most pos number?

$01\ldots1_{two} = (2^{(n-1)}-1)_{ten}$

Increment?

just like unsigned!

Two's Complement Summary

- Used by all modern hardware
- Roughly evenly split between positive and negative
  - One more negative # because positive side has 0
- Can still use MSB as sign bit
- To negate: Flip the bits and add one
  - Example: +7 = 0b 0000 0111, -7 = 0b 1111 1001

Two's Complement Review

- Suppose we had 5 bits. What range of integers can be represented in two's complement?

(A) -31 to +31 ← need 6 bits

(B) -15 to +15 ← one's complement

(C) 0 to +31 ← unsigned

(D) -16 to +15 ← two's complement

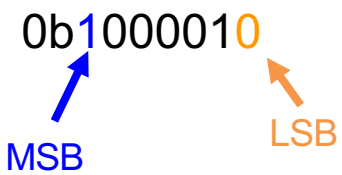
(E) -32 to +31 ← need 6 bits

Agenda

- Course Overview
- Course Policies
- Number Representation
  - Number Bases
  - Signed Representations
  - **Overflow**
  - Sign Extension

Numbers in a Computer

- Numbers really have ∞ digits, but hardware can only store a finite number of them (fixed)
  - Usually ignore *leading zeros*
  - Leftmost is *most significant bit* (MSB)
  - Rightmost is *least significant bit* (LSB)



Overflow

- Overflow is when the result of an arithmetic operation can't be represented by the (FINITE) hardware bits
  - i.e. the result is mathematically incorrect
- Examples:
  - Unsigned: $0b1\ldots1 + 1_{10} = 0b0\ldots0 = 0?$
  - Two's: $0b01\ldots1 + 1_{10} = 0b10\ldots0 = -2^{(n-1)}{}_{ten}?$

- Solution: add more bits

Agenda

- Course Overview
- Course Policies
- Number Representation
  - Number Bases
  - Signed Representations
  - Overflow
  - **Sign Extension**

Sign Extension

- Want to represent the same number using more bits than before
  - Easy for positive #s (add leading 0's), more complicated for negative #s
  - Sign and magnitude: add 0's *after* the sign bit
  - One's complement: copy MSB
  - Two's complement: copy MSB
- Example:
  - Sign and magnitude: 0b 11 = 0b 1001
  - One's/Two's complement: 0b 11 = 0b 1111

Summary

- Review course policies and deadlines!
  - Attend discussion and lab this week
  - Get to know your fellow students and also TA's, Tutors, AI's and instructors :)
- Number Representation: How to represent positive and negative integers using binary
  - Unsigned: Interpret numeral in base 2
  - Signed: Two's Complement
  - Biased: Add bias
  - Sign extension must preserve signed number