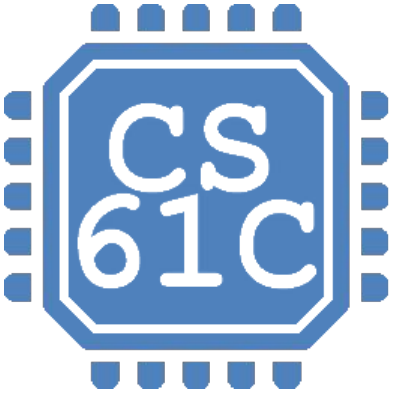
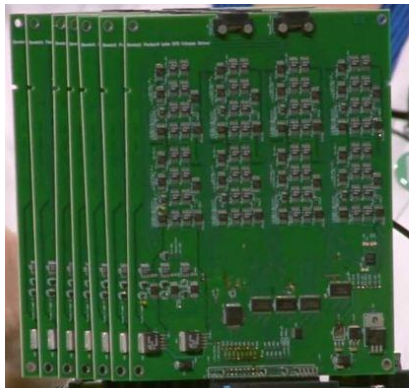


Great Ideas in Computer Architecture

Course Intro, Number Representation

Instructors: Shreyas Chand, Justin Hsia



HPE shows off a computer intended to emulate the human brain

"Brains take in a lot of data related to sights, sounds and smell, which they have to process in parallel without lagging, in terms of computation speed.

"HPE's ultimate goal is to create computer chips that can compute quickly and make decisions based on probabilities and associations, much like how the brain operates. The chips will use learning models and algorithms to deliver approximate results that can be used in decision-making."

- Online article

Introducing Shreyas

- Lecturer
- Born in India, moved around A LOT
- Graduated from Cal in May
  - BS in EECS, Education Minor
- TA for 61C for two years, also guest lecturer
- Work
  - Interned at CareZone, Yelp, Airbnb
  - Full time at Airbnb - Android Engineer
- Interests
  - Eve Online, Board Games, Homebrewing

Introducing Justin

- Lecturer – just call me Justin
- **Upbringing:** Born and raised in the Bay
- **Education:** I bleed Blue and Gold (Go Bears!)
  - B.S. ME, B.S. EECS, M.S. EECS, Ph.D. EECS
- **Research:** Synthetic Biology
- **Teaching:** EE128, CS10 (Sp16), CS61C (Su12, Su13)
- **Interests:**



Agenda

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

What is CS61C About?

- It is about the hardware-software interface
  - What does the programmer need to know about the underlying hardware
- Introduction to *systems courses*
  - **Software:** compilers, operating systems, security
  - **Hardware:** digital logic design, embedded systems, VLSI, computer architecture

- **Note:** CS61C is *not* required to declare L&S CS

CS61C For Software Development

- Knowing the tools of the trade – computers!
  - "Computers" come in all shapes and sizes
  - Computing achieved in many different ways nowadays
- When performance matters
  - Ex: taking advantage of parallelism
- 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 and create 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

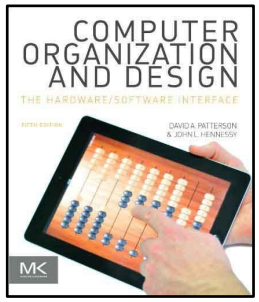
Agenda

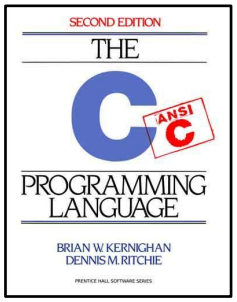
- Course Overview
- Course Policies
- Administrivia
- Number Representation
  - Number Bases
  - Overflow
  - Signed Representations
  - Sign Extension

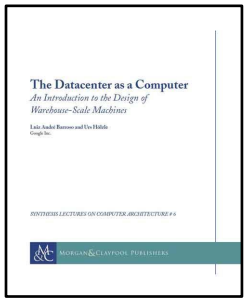
Course Information

More info found on the course policies page

- **Website:** http://inst.eecs.berkeley.edu/~cs61c/su16
  - Check for assignments, weekly schedule, contact info
- **Textbooks:**







- **Piazza:** http://piazza.com/class#summer2016/cs61c
  - Will have every announcement, discussion, clarification
- **Inst account:** http://inst.eecs.berkeley.edu/webacct
  - Submit assignments, check grades, access lab files

Course Assignments and Grading

- **Homework** (9%) – 7 total on edX, drop lowest
- **Labs** (8%) – 14 total, drop lowest 2
- **Projects** (30%) – 4 total, weighted by difficulty
- **Exams**
  - **Midterm 1** (13%): Thursday, July 7 in class
  - **Midterm 2** (13%): Monday, July 25 in class
  - **Final** (22%): Thursday, August 11 @ 9am
- **EPA** (5%) – Effort, Participation, and Altruism

Late Policy – Slip Days

- Project submissions due at 11:59:59pm
  - Count lateness in *days* (even if just by a second)
- Late penalty is 33% deduction of score per day
- You are given 3 slip day tokens
  - Will be distributed amongst your late submissions at the end of the semester to maximize your grade
  - No benefit to having leftover tokens
- Use at own risk – don't want to fall too far behind

Exam Clobber Policy

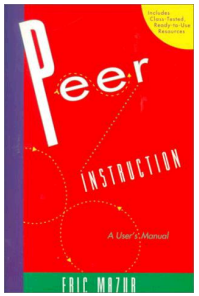
- MT1 and MT2 are roughly independent; Final is cumulative
- Can replace MT1 and MT2 scores with MT1 and MT2 portion scores of Final
  - Make up for a bad testing day, illness, other unforeseen circumstance
  - In practice, final is harder than the midterms
- See course policies for exact formulas

EPA

- Encourage class-wide learning!
- Effort
  - Attending office hours, completing all assignments
  - Keeping up with Piazza activity
- Participation
  - Making the class more interactive by asking questions in lecture, in discussion, and on Piazza
  - Peer instruction voting
- Altruism
  - Helping others in lab, discussion, and Piazza

Peer Instruction

- Increase real-time learning in lecture, test your understanding, increase student interactions
  - Lot of research supports its effectiveness
- Multiple choice question at end of lecture "segment"
  - 1 minute to decide on your own
  - 2 minutes in pairs to reach consensus
  - Learn through discussion
- Vote using i>clicker remotes
  - Can borrow for summer with $40 check as collateral



Question: Which statement is FALSE about the CS61C policies this summer?

(A) Both homework and labs are being graded on completion, not correctness

(B) Activities in person (lab, disc) and online (Piazza) count towards your class grade

(C) Every student needs to go out and buy an i>clicker remote if he/she does not own one already

(D) The exams account for a non-majority of the overall grade

Working with Others

- We are encouraging partner/group work
- Beneficial for the class
  - Someone to bounce ideas off of and work with and keep you on task
  - Don't have to go it alone – meet new people!
- Valuable life skill: working well with others
  - Distribution of work, communication skills, time management, teaching skills

Policy on Assignments and Independent Work

- All submissions are expected to be yours and yours alone (or group's when applicable)
- 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)

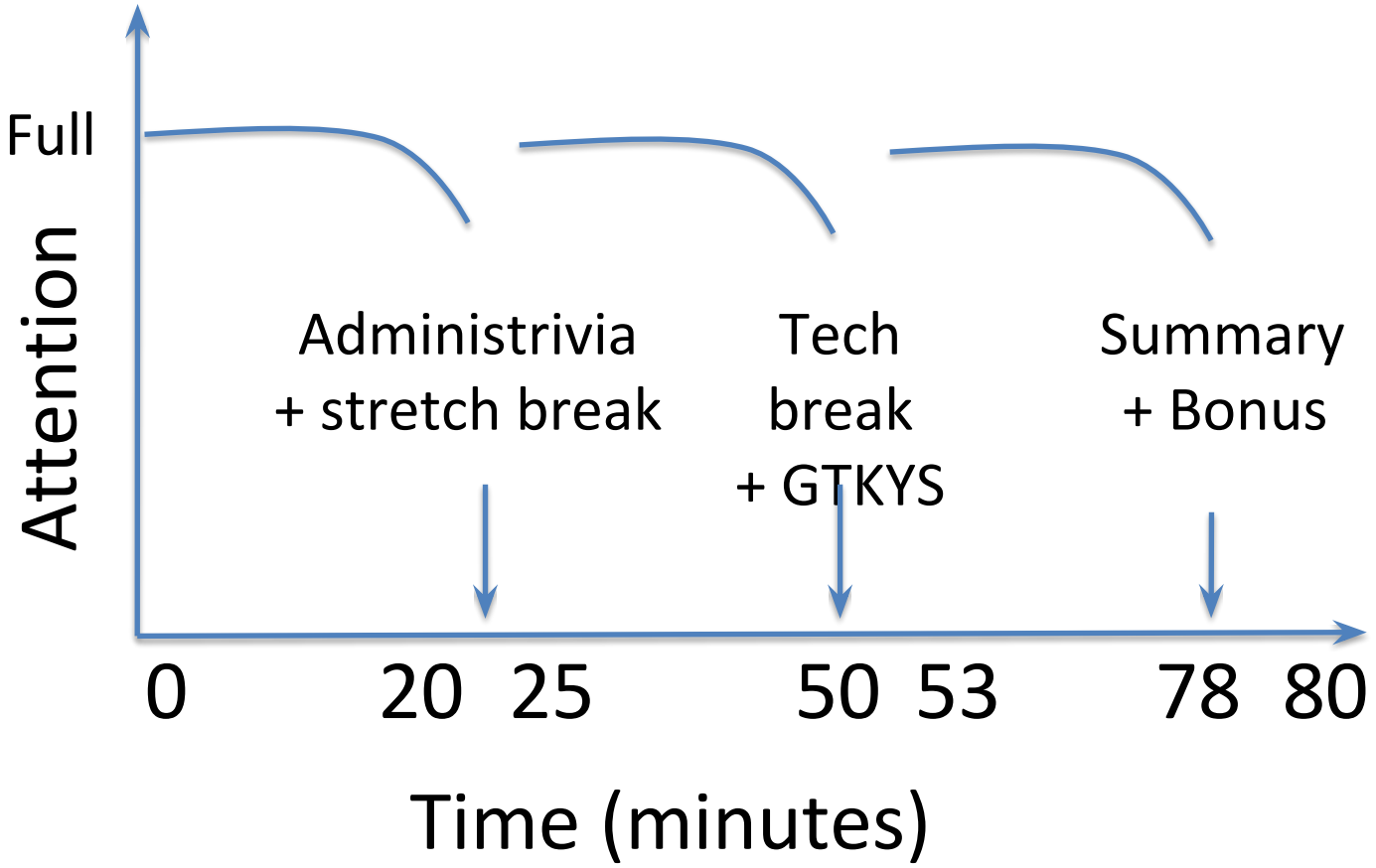
Policy on Assignments and Independent Work

- 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.
  - Possible automatic F in the course
- People are caught every semester of 61C

Successful Behaviors

- Practice, practice, practice
  - Learn by doing: deep learning doesn't happen in lec
  - Growth mindset: success through effort
- Find a learning community
  - Learning is much more fun with friends
  - Learn via teaching or different perspectives
- Engage frequently with the course material
  - You don't know what you don't know
  - Take advantage of available resources
- Avoid electronic screens during lecture
- You learn best from your mistakes
  - Don't be afraid to be wrong; you lose if you remain silent

Architecture of a Lecture



Administrivia

- Discussions, labs, and OHs start immediately
  - Yes, that means today!
  - Switching sections: if you find another 61C student willing to swap lab, talk to your TAs
  - Attend whichever discussion you want, as long as there is enough physical room
- HW0 and mini-bio due this Sunday, June 26th
  - Find a *small* digital image of yourself
  - Both worth points, neither can be dropped

Agenda

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

Number Representation

- Numbers are an abstract concept!
  - Recognize that these are *arbitrary* symbols
- Inside a computer, everything stored as a sequence of 0's and 1's (bits)
  - Even this is an abstraction!
- How do we represent numbers in this format?
  - Let's start with integers

Number Bases

- **Key terminology:** digit (*d*) and base (*B*)
- In base B, each digit is one of B possible symbols
- Value of *i*-th digit is $d \times B^i$ where *i* starts at 0 and increases from right to left
  - *n* digit number $d_{n-1} d_{n-2} \ldots d_1 d_0$
  - value = $d_{n-1} \times B^{n-1} + d_{n-2} \times B^{n-2} + \ldots + d_1 \times B^1 + d_0 \times B^0$
- Notation: Base is indicated using either a prefix or a subscript

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

Number Base Examples

- Examples:

3 2 1 0

$\mathbf{9472}_{ten}$ = 9000 + 400 + 70 + 2

9x1000 + 4x100 + 7x10 + 2x1

$9x10^3 + 4x10^2 + 7x10^1 + 2x10^0$

$\mathbf{9472}_{ten} = 2x16^3 + 5x16^2 + 0x16^1 + 0x16^0$

$= \mathbf{2500}_{hex}$

0xA15 = 0b 1010 0001 0101

Bits Can Represent Anything

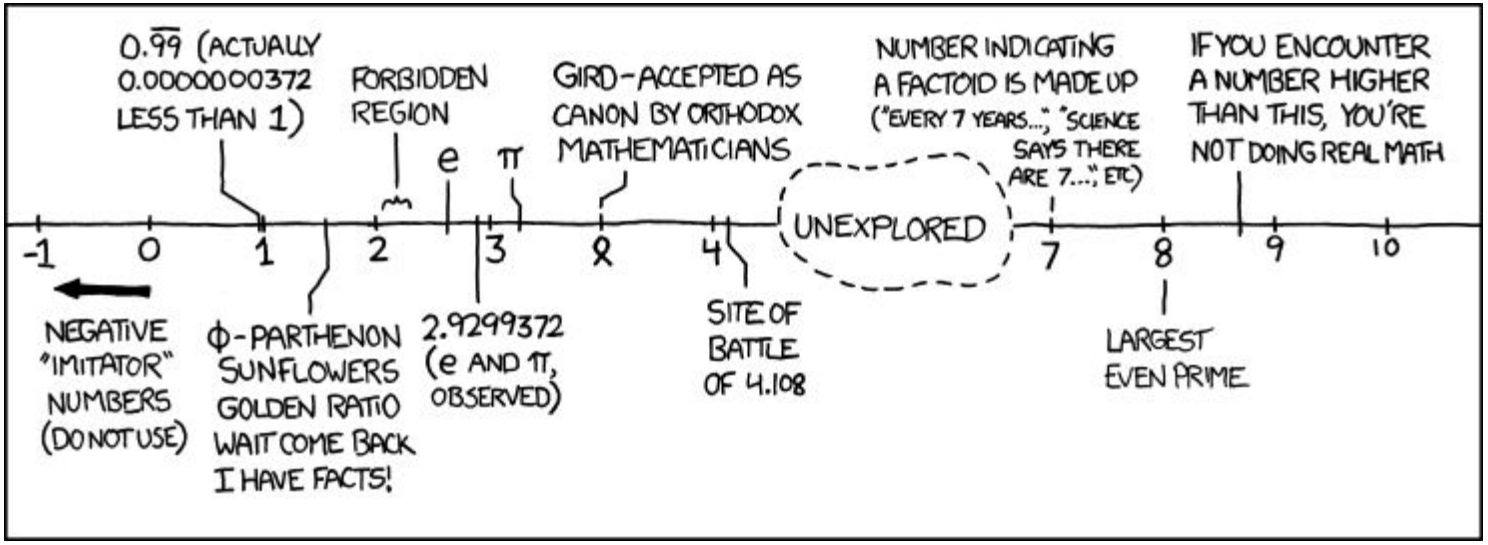
- *n* digits in base *B* can represent at most B^n things!
  - Each of the *n* digits is one of *B* possible symbols
  - Have more things? Add more digits!
- Example: Logical values (1 bit) – 0 is False, 1 is True
- Example: Characters
  - 26 letters require 5 bits ($2^5 = 32 > 26$)
- Example: Students in this class (8 bits)
- For convenience, can group into *nibbles* (4 bits) and *bytes* (8 bits)

So What Number Is It Anyway?

- Here we are using binary bit patterns to represent numbers
  - Strictly speaking they are called *numerals* and have no meaning until you interpret them
  - Is CAB a word (taxi) or a number (3243_{ten})?
  - Is 0x999999 a number or a color (RGB)?
- Keep in mind that the same bit pattern will mean different things depending on how you choose to interpret it

Numbers in a Computer (1/2)

- The number line:



Alt text: The Wikipedia page "List of Numbers" opens with "This list is incomplete; you can help by expanding it.“

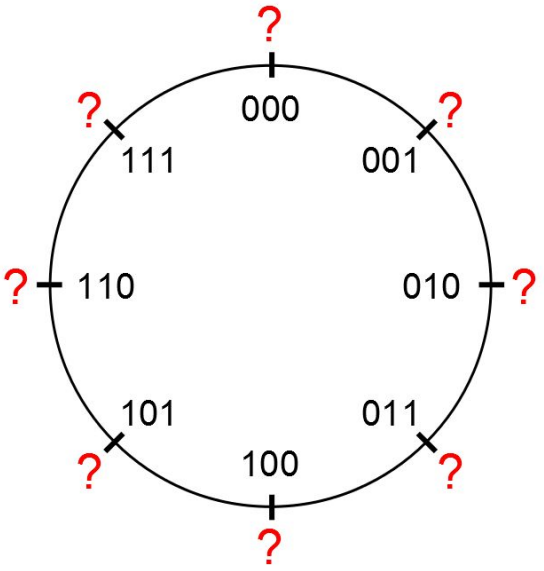
(Source: http://xkcd.com/899)

- What do we choose to represent?

Numbers in a Computer (2/2)

- 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)

- "Circle" of binary numerals:
  (3-bit example)



Unsigned Integers

Represent only non-negative (unsigned) integers:

$0000_{two} = 0_{ten}$

$0001_{two} = 1_{ten}$

. . .

$0111_{two} = 7_{ten}$

$1000_{two} = 8_{ten}$

. . .

$1110_{two} = 14_{ten}$

$1111_{two} = 15_{ten}$

Zero?

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

Most neg number?

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

Most pos number?

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

Increment?

Signed Integers

- *n* bits can represent 2^n different things
  - Ideally, want the range evenly split between positive and negative
- How do we want to encode zero?
- Can we encode them in such a way that we can use the same hardware regardless of whether the numbers are signed or unsigned?
  - e.g. incrementing pos/neg numbers

Sign and Magnitude

MSB gives sign, rest treated as unsigned (magnitude):

$0000_{two} = +0_{ten}$
$0001_{two} = +1_{ten}$
. . .
$0110_{two} = +6_{ten}$
$0111_{two} = +7_{ten}$
$1000_{two} = -0_{ten}$
$1001_{two} = -1_{ten}$
. . .
$1110_{two} = -6_{ten}$
$1111_{two} = -7_{ten}$

Zero?

$0\ldots0_{two}$ and $10\ldots0_{two} = \pm0_{ten}$

Most neg number?

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

Most pos number?

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

Increment?

Biased Notation

Like unsigned, but "shifted" so zero is in the middle:

$0000_{two} = -7_{ten}$
$0001_{two} = -6_{ten}$
. . .
$0110_{two} = -1_{ten}$
$0111_{two} = 0_{ten}$
$1000_{two} = +1_{ten}$
$1001_{two} = +2_{ten}$
. . .
$1110_{two} = +7_{ten}$
$1111_{two} = +8_{ten}$

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?

One's Complement

New negation procedure – complement the bits:

$0000_{two} = +0_{ten}$
$0001_{two} = +1_{ten}$
. . .
$0110_{two} = +6_{ten}$
$0111_{two} = +7_{ten}$
$1000_{two} = -7_{ten}$
$1001_{two} = -6_{ten}$
. . .
$1110_{two} = -1_{ten}$
$1111_{two} = -0_{ten}$

Zero?

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

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:

$0000_{two} = 0_{ten}$
$0001_{two} = +1_{ten}$
. . .
$0110_{two} = +6_{ten}$
$0111_{two} = +7_{ten}$
$1000_{two} = -8_{ten}$
$1001_{two} = -7_{ten}$
. . .
$1110_{two} = -2_{ten}$
$1111_{two} = -1_{ten}$

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?

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 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

Overflow

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

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

- Number Representation: How to represent positive and negative integers using binary
  - Unsigned: Interpret numeral in base 2
  - Signed: Two's Complement
  - Biased: Subtract bias
  - Sign extension must preserve signed number