The Hardware/Software Interface

Binary and Numerical Representation

Instructors:

Justin Hsia, Amber Hu

Teaching Assistants:

Anthony Mangus, Divya Ramu, Grace Zhou, Jessie Sun, Jiuyang Lyu, Kanishka Singh, Kurt Gu, Liander Rainbolt, Mendel Carroll, Ming Yan, Naama Amiel, Pollux Chen, Rose Maresh, Soham Bhosale, Violet Monserate

http://xkcd.com/676/

Lecture Outline (1/3)

- **Course Introduction**
- Course Policies
  - https://courses.cs.washington.edu/courses/cse351/25au/syllabus.html
- Binary and Numerical Representation

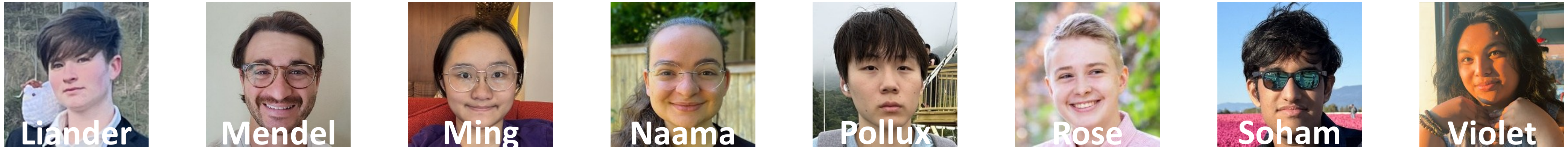
Course Staff: Instructors

- Justin Hsia (he/him)
  - CSE Associate Teaching Professor
  - You can just call me "Justin"
  - Important: expecting Baby #2 in the middle of this quarter (!)

- Amber Hu (they/them)
  - CSE Lecturer (part-time)
  - You can call me "Amber," or for fun "Doctor Hu?"
  - I'll be taking on a bigger role as Justin welcomes a new family member

Course Staff: Teaching Assistants

- TAs:



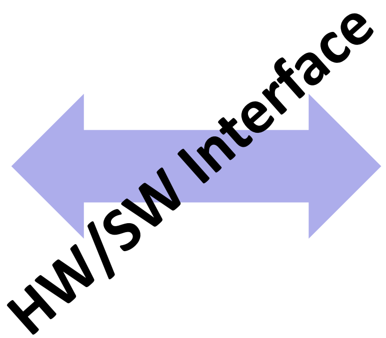
  - Learn more about us on the course website!

- More than anything, we want you to feel…
  - ✓ Comfortable and welcome in this space
  - ✓ Able to learn and succeed in this course
  - ✓ Comfortable reaching out if you need help or want change

Welcome to CSE351!

10000001101111110000100100000011100000000000
0111010000011000
100010110100010000100100000010100

1000100111000010
110000011111101000011111
111101110111110000100100000011100

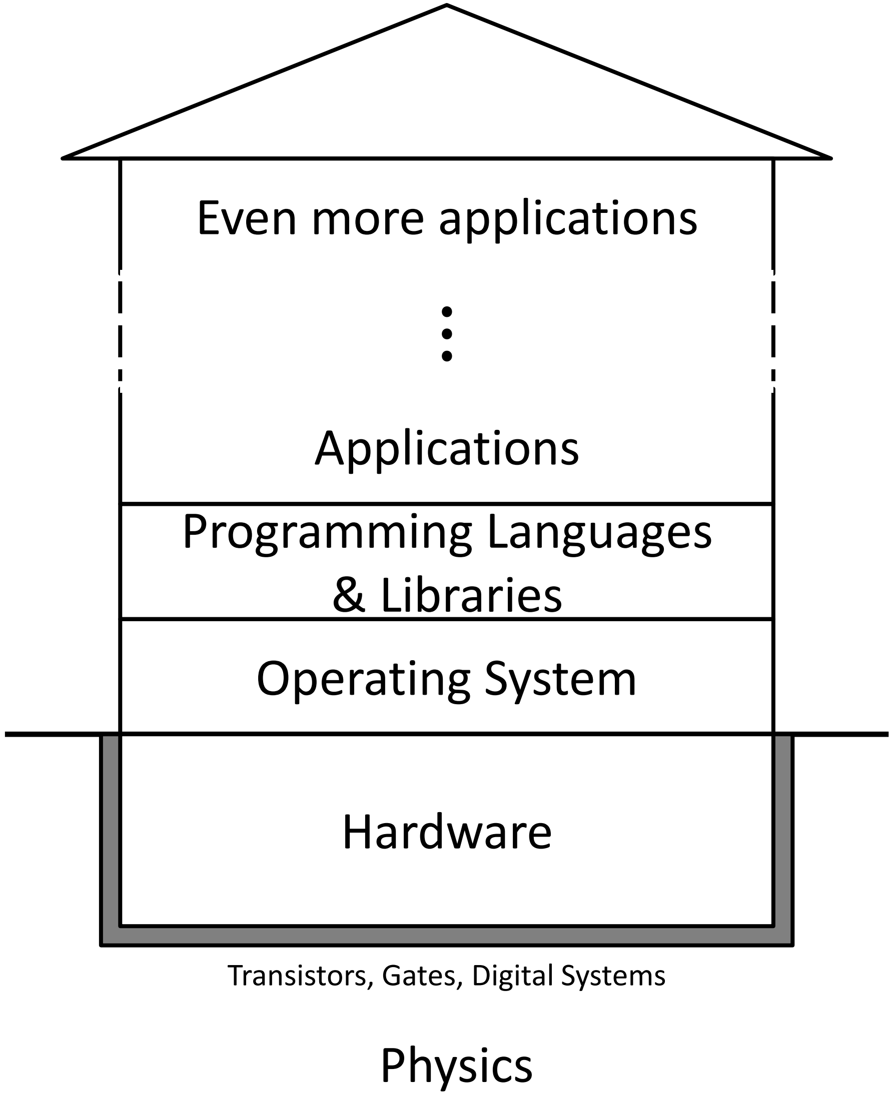


- Our goal is to teach you the key abstractions "under the hood:"
  - Better understand computing hardware as it has evolved
  - Better understand program translation and execution

"House" of Computing Metaphor

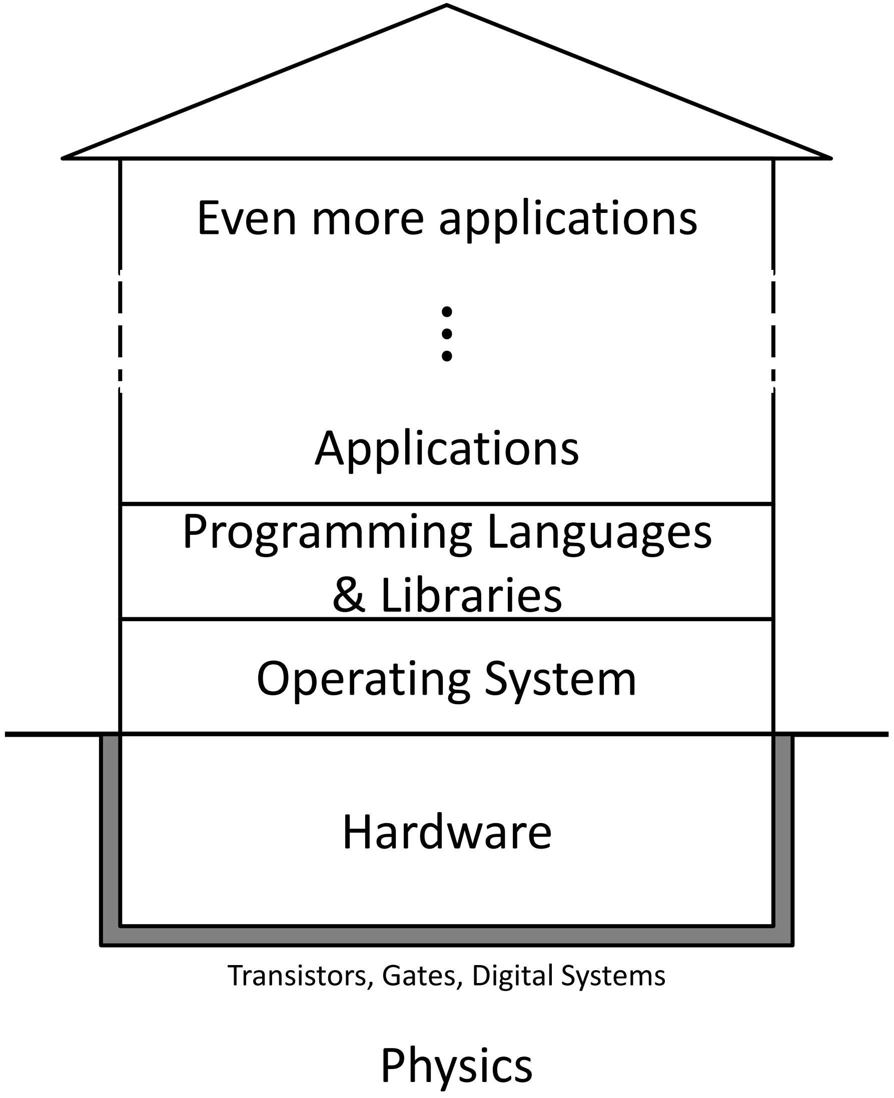
- We continue to build upward but everything relies on the base & foundation
  - We'll explore parts of Hardware, OS, and PL

- Built a long time ago
  - Some parts have been updated over the years, some have not
  - More remodeling necessary, but should understand *how* and *why* things are this way before demolishing anything



The Hardware/Software Interface Topic Groups

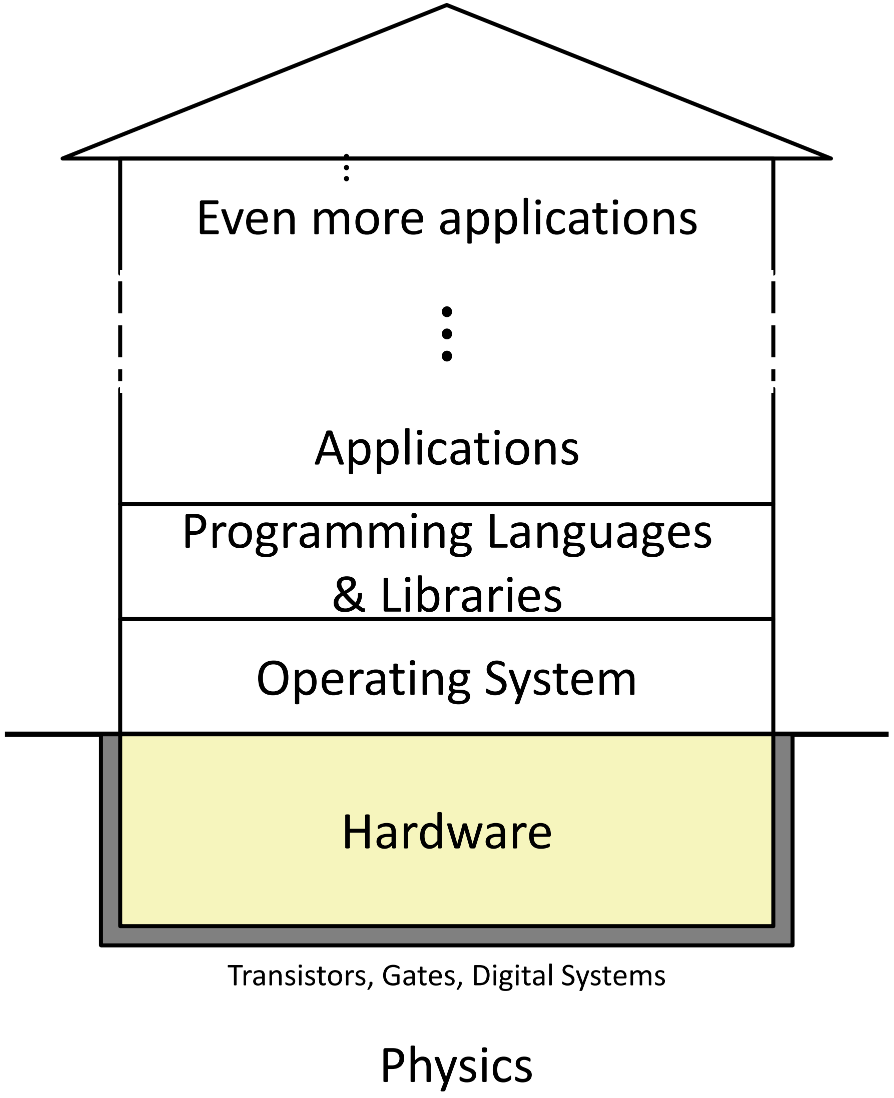
- Topic Group 1: **Data**
  - Memory, Data, Integers, Floating Point, Arrays, Structs
- Topic Group 2: **Programs**
  - x86-64 Assembly, Procedures, Stacks, Executables
- Topic Group 3: **Scale & Coherence**
  - Caches, Memory Allocation, Processes, Virtual Memory



The Hardware/Software Interface Topic Group 1

- Topic Group 1: **Data**
  - Memory, Data, Integers, Floating Point, Arrays, Structs

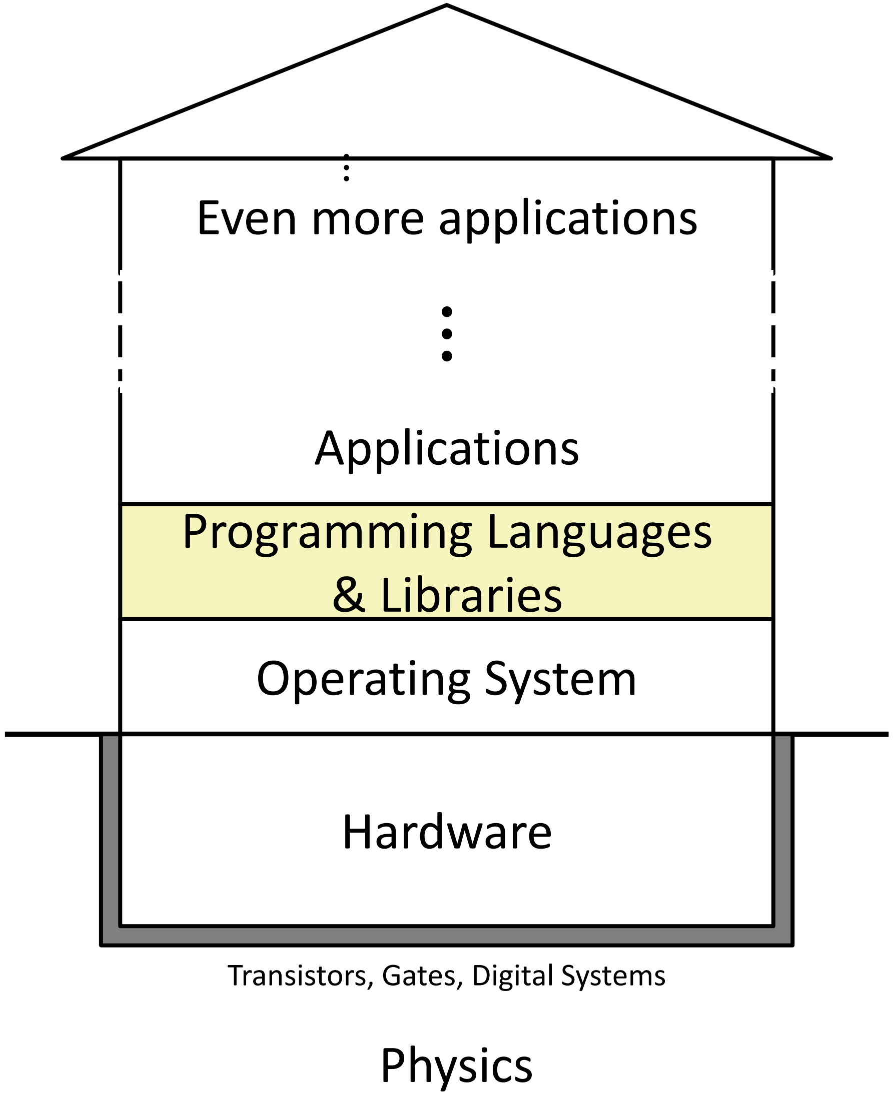
- How do we store information for other parts of the house of computing to access?
  - How do we represent data and what limitations exist?
  - What design decisions and priorities went into these encodings?



The Hardware/Software Interface Topic Group 2

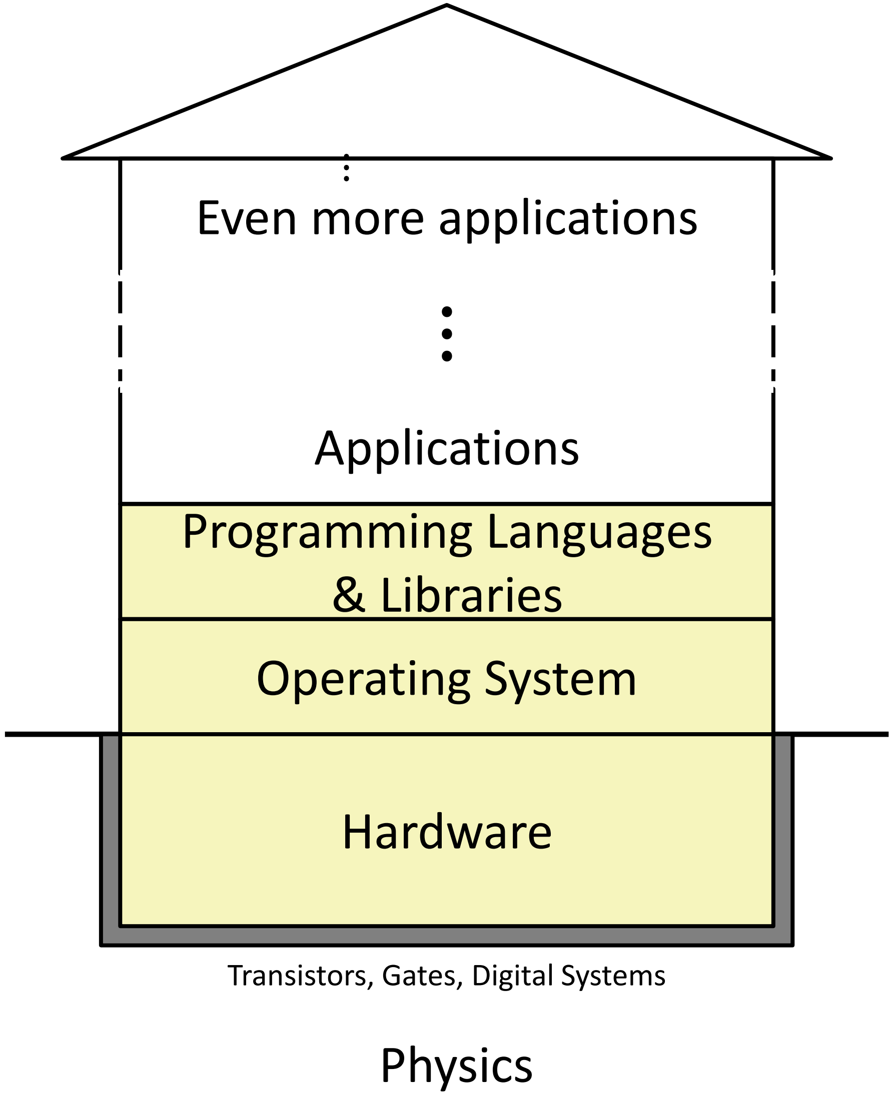
- Topic Group 2: **Programs**
  - x86-64 Assembly, Procedures, Stacks, Executables

- How are programs created and executed on a CPU?
  - How does your source code become something that your computer understands?
  - How does the CPU organize and manipulate local data?



The Hardware/Software Interface Topic Group 3

- Topic Group 3: **Scale & Coherence**
  - Caches, Memory Allocation, Processes, Virtual Memory
- How do we maintain logical consistency in the face of more data and more processes?
  - How do we support control flow both within many processes and things external to the computer?
  - How do we support data access, including dynamic requests, across multiple processes?



Course Learning Objectives

- At the end of this course, students should be able to:
  - Describe the multi-step process by which a high-level program becomes a stream of instructions executed by a processor;
  - Describe the basic organization of the memory hierarchy and the effect of its parameters on system performance;
  - Trace the execution of assembly code (x86-64), map the assembly to high-level language constructs, and write simple pieces of assembly programs;
  - Write C code using pointers to create and manipulate data structures;
  - Write (or rewrite) code to take advantage of the computer execution model to improve execution efficiency;
  - Debug small-ish C and assembly programs using GDB;
  - Perform basic navigation and file system operations in Linux using a terminal;
  - Explain the basic role of an operating system in executing processes;
  - Identify some of the ways that computers and their design principles affect society today.

= concepts

Course Learning Objectives: Concepts

- At the end of this course, students should be able to:
  - Describe the multi-step process by which a high-level program becomes a stream of instructions executed by a processor;
  - Describe the basic organization of the memory hierarchy and the effect of its parameters on system performance;
  - Trace the execution of assembly code (x86-64), map the assembly to high-level language constructs, and write simple pieces of assembly programs;
  - Write C code using pointers to create and manipulate data structures;
  - Write (or rewrite) code to take advantage of the computer execution model to improve execution efficiency;
  - Debug small-ish C and assembly programs using GDB;
  - Perform basic navigation and file system operations in Linux using a terminal;
  - Explain the basic role of an operating system in executing processes;
  - Identify some of the ways that computers and their design principles affect society today.

Course Learning Objectives: Tools

= concepts
= tools

- At the end of this course, students should be able to:
  - Describe the multi-step process by which a high-level program becomes a stream of instructions executed by a processor;
  - Describe the basic organization of the memory hierarchy and the effect of its parameters on system performance;
  - Trace the execution of assembly code (x86-64), map the assembly to high-level language constructs, and write simple pieces of assembly programs;
  - Write C code using pointers to create and manipulate data structures;
  - Write (or rewrite) code to take advantage of the computer execution model to improve execution efficiency;
  - Debug small-ish C and assembly programs using GDB;
  - Perform basic navigation and file system operations in Linux using a terminal;
  - Explain the basic role of an operating system in executing processes;
  - Identify some of the ways that computers and their design principles affect society today.

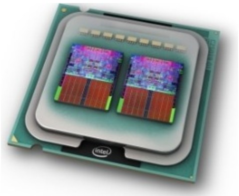
Course Learning Objectives: Skills

= concepts
= tools
= skills

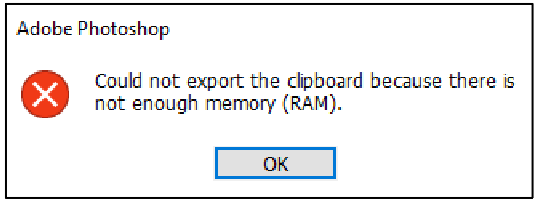
- At the end of this course, students should be able to:
  - Describe the multi-step process by which a high-level program becomes a stream of instructions executed by a processor;
  - Describe the basic organization of the memory hierarchy and the effect of its parameters on system performance;
  - Trace the execution of assembly code (x86-64), map the assembly to high-level language constructs, and write simple pieces of assembly programs;
  - Write C code using pointers to create and manipulate data structures;
  - Write (or rewrite) code to take advantage of the computer execution model to improve execution efficiency;
  - Debug small-ish C and assembly programs using GDB;
  - Perform basic navigation and file system operations in Linux using a terminal;
  - Explain the basic role of an operating system in executing processes;
  - Identify some of the ways that computers and their design principles affect society today.

Applications of 351 Material

- Examples you might encounter in the "real world":
  - Shopping for a new CPU (*e.g.*, # of cores, size of cache)

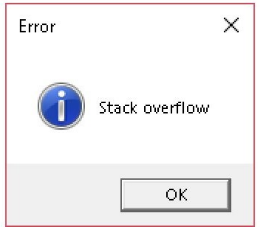


  - Running out of memory/RAM from running too many programs for too long



  - Game modding / reverse engineering

  - CPU/GPU benchmarks using the unit of a GFLOP
  - Stack overflow (not the website)



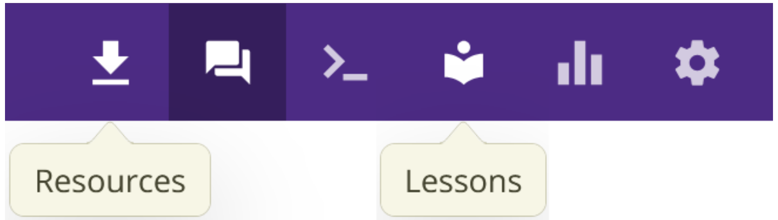
  - News about security vulnerabilities (*e.g.*, Heartbleed, Cloudbleed)

Lecture Outline (2/3)

- Course Introduction
- **Course Policies**
  - https://courses.cs.washington.edu/courses/cse351/25au/syllabus.html
- Binary and Numerical Representation

Bookmarks

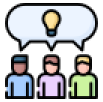
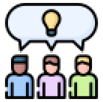
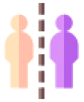
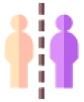
- Website: https://courses.cs.washington.edu/courses/cse351/25au/
  - Schedule, policies, materials, tutorials, assignment specs, etc.
- Ed Course: https://edstem.org/us/courses/80075/
  - Discussion: announcements, ask and answer questions
  - Lessons: pre-lecture readings, lecture polls, homework



Find Ed Lessons in the top-right corner of Ed.

- Linked from website and Ed
  - Canvas: surveys, grade book, Zoom links
  - Gradescope: lab submissions, exams
  - Panopto: lecture recordings

Grading

- **Pre-Lecture Readings:** 10%, collaboration allowed 
  - Can reveal solution after one attempt (completion)
- **Homework:** 20% total, collaboration allowed 
  - Unlimited submission attempts (autograded correctness)
- **Labs:** 35% total, partners optional 
  - Last submission graded (correctness)
- **Exams:** Midterm (12%) and Final (20%) 
  - In-person paper exams; individual
  - Midterm clobber policy!
- **EPA:** Effort, Participation, and Altruism (3%)

Group Work in 351

- Get to know each other and help each other out
  - More fun! Valuable life skill! Expand your horizons with diversity of perspectives!
- Group work will be *emphasized* in this class
  - Lecture & section will have group work time – will gain the most if you participate!
    - TAs will circle around the room and interact with groups
    - Raise your hand to get the attention of a staff member
  - Most assignments allow collaboration – talking to classmates will help you synthesize concepts and terminology
    - *The major takeaways for this course will be the ability to explain the major concepts verbally and/or in writing to others*
  - However, the responsibility for learning falls on *you*

Generative AI in 351

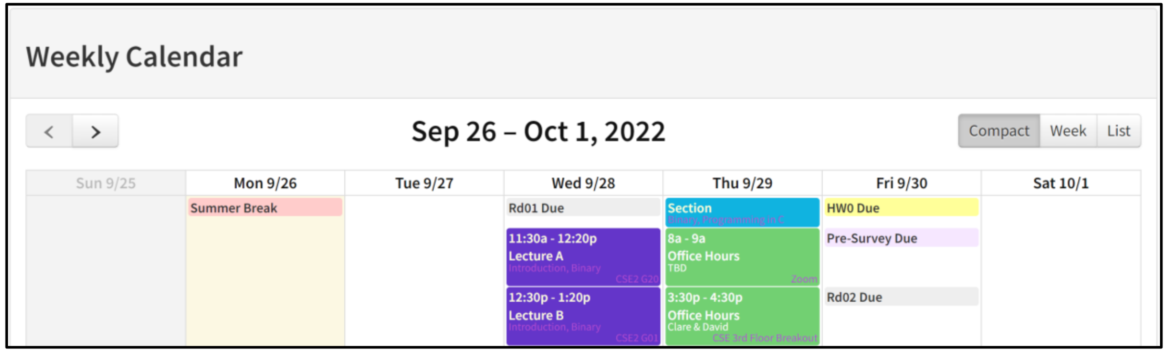
- Can use for concepts
  - *e.g.*, re-explain a concept, summarize notes
  - Note that some specific terminology may differ for this course vs. what's on the Internet, so be careful!

- Cannot use to generate code, but can explain/check code you wrote
  - This includes Copilot code writing suggestions 🙅

- You will likely have a better time talking to a staff member
  - We've been there before! We know what you're going through.

Office Hours

- Check Weekly Calendar on website for scheduled office hours:
  - In-person or virtual, but NOT hybrid
  - Zoom meeting links found in Zoom tab within Canvas



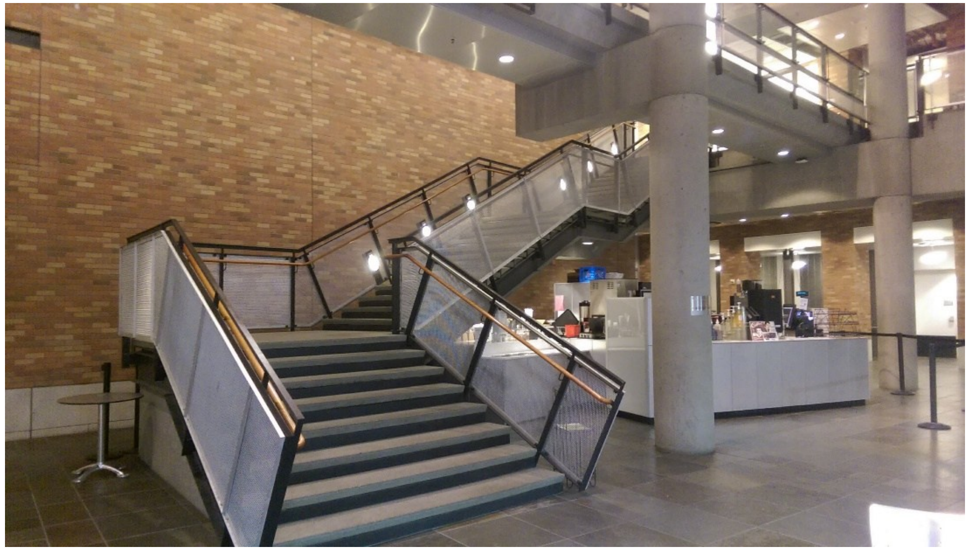
- *All* office hours will use a Google Sheets queue:
  - Fill out first 3 columns to enter queue:

Name(s)	Category	Description	Time Queued	Staff	Status
Example 1	Concept	Question about floating point encoding range.		Justin	Done
Example 2	Debugging	Lab 5: running into a segfault in mm_malloc after reaching end of the heap.		Justin	Done
Example 3	Spec	Lab 1a: confusion over within same block examples		Justin	Done
Example 4	Tools	GDB: how do I examine memory on the stack?		Justin	Done

- We encourage you to chat with other students if the TAs are busy!
  - Keep it high level, no sharing code or answers

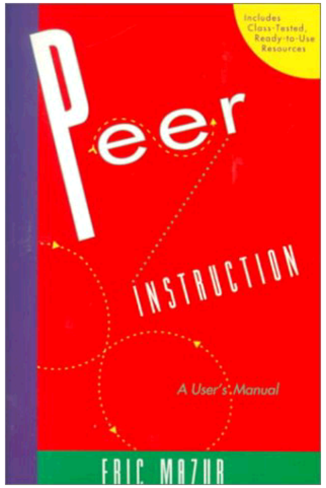
In-Person Office Hours

- Mostly Allen breakouts
  - Up the stairs in the CSE Atrium (Allen Center, not Gates)
  - The open areas with the whiteboard walls are the breakouts!



- Amber's OHs in CSE 204
  - Second floor, right across bathrooms near the main elevators

Lecture Polls and Discussions



- Increase learning, test your understanding, increase student interactions, makes the class more engaging and fun
  - Lot of research supports its effectiveness:
- Polls on technical material will be multiple-choice and short answer
  - You haven't mastered the material yet; mistakes are part of the process!
- Discussion questions will be more open-ended
  - Be respectful of others' opinions and experiences
- Respond on Lecture Ed lesson for credit and we will use ***random call*** to solicit responses from audience
  - Don't need to be correct, just want the feedback of what was discussed

Extensions, Accommodations, Help

- Extenuating circumstances
  - Students (and staff) face an extremely varied set of environments and circumstances
  - For formal accommodations, go through Disability Resources for Students (DRS)
  - We will try to be accommodating otherwise, but *the earlier you reach out*, the better
- Don't suffer in silence – talk to a staff member!
  - We have a 1-on-1 meeting request form

To-Do List

- Admin
  - Explore/read the course website *thoroughly*, especially the syllabus
  - Check that you can access Ed Discussion & Lessons
  - **Get your machine set up to access the CSE Linux environment (`attu` or `cancun`) *as soon as possible***
  - Optionally, sign up for CSE 391: System and Software Tools

- Assignments
  - Pre-Course Survey and HW0 due Friday (9/26)
  - HW1 and Lab 0 due Monday (9/29)
  - Pre-lecture Readings due before each lecture @ 11 am
  - Lecture activities are due before NEXT lecture @ 11 am

Lecture Outline (3/3)

- Course Introduction
- Course Policies
  - https://courses.cs.washington.edu/courses/cse351/25au/syllabus.html
- **Binary and Numerical Representation**

Common Bases (Review)

- *Humans* think about numbers in base 10, but *digital computers* "think" about numbers in base 2 (**binary**)
  - Symbols: 0, 1
  - Common binary denominations
    - A binary digit is known as a ***bit***
    - A group of 8 bits is called a ***byte***
- Hexadecimal (hex, for short) is base 16 = 2^4
  - Every hex digit is 4 bits
  - Symbols? 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, **A, B, C, D, E, F**
  - Example: What is $A5_{16}$ in base 10?
    - $A5_{16} = (10 \times 16^1) + (5 \times 16^0) = 165_{10}$

Base 10	Base 2	Base 16
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

Polling Questions

- What is the *decimal value* of the numeral 107_8?
  A. 71
  B. 87
  C. 107
  D. 568

- Represent 0b100110110101101 in hex.

- What is the decimal number 108 in hex?
  A. 0x6C
  B. 0xA8
  C. 0x108
  D. 0x612

- Represent 0x3C9 in binary.

Numerical Encoding (Review)

- **You can represent *any* countable set of things using numbers**
  - Create the encoding scheme by assigning a unique numeral to each element
  - Example: English Letters such as CSE→0x435345, yay→0x796179
  - Example: Emoticons such as 😃 0x0, 😞 0x1, 😎 0x2, 😇 0x3, 😈 0x4, 🙋 0x5

- With x digits in base b, how many "things" can you represent?
  - Example: With 3 bits → 2^3 = 8 things
  - Example: With 2 hex digits → 16^2 = 256 things

- If you are determining your numeral width to represent N things, you will need $b^x \geq N$
  - Example: 5 bits for alphabet because 2^5 = 32 > 26

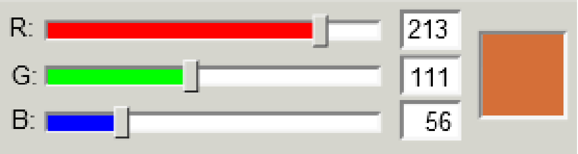
Digital Computer Data

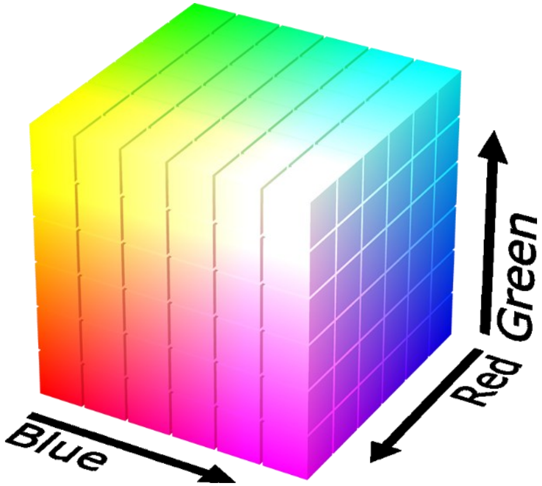
- Anything with a numerical encoding scheme can be stored in binary and used in a digital computer!
  - We have created countless different encoding schemes that work for and between specific pieces of hardware and software

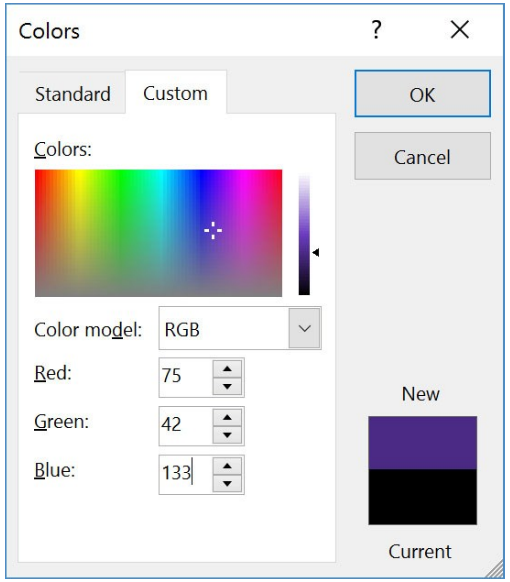
- *A sequence of bits can have many meanings!*
  - Consider the hex sequence 0x4E6F21, whose common interpretations include:
    - The decimal number 5140257
    - The real number 7.203034×10^{-39}
    - The characters "No!"
    - The background color of this text
  - It is up to the program/programmer to decide how to interpret the sequence of bits

Binary Encoding – Colors

- RGB – Red, Green, Blue
  - Additive color model (light): byte (8 bits) for each color
  - Commonly seen in hex (in HTML, photo editing, etc.)
  - Examples: **Blue**→0x0000FF, **Gold**→0xFFD700, →0xFFFFFF, **Deep Pink**→0xFF1493







Binary Encoding – Characters/Text (1/2)

- ASCII Encoding (www.asciitable.com)
  - American Standard Code for Information Interchange

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

What's Missing?

Binary Encoding – Characters/Text (2/2)

- ASCII Encoding (www.asciitable.com)
  - *American* Standard Code for Information Interchange
- Created in 1963
  - Memory was expensive, 32KB in brand new machines
  - *Economic incentive* to use fewer bits for encoding
- **Design Goals:**
  - Represent everything on an *American* typewriter as *efficiently* as possible
  - Organize similar characters together
    - Numbers, uppercase, lowercase, then other stuff

Binary Encoding – Unicode & Emoji

- Unicode Standard is managed by the Unicode Consortium
  - "Universal language" that uses 1-4 bytes to represent a much larger range of characters/languages, including emoji
  - Adds new emojis every year, though adoption often lags: □ (orca)
    - https://emojipedia.org/new/

- Emojipedia demo: http://www.emojipedia.org
  - Taco:  (added 2015)
  - Code points: U+1F32E
  - Display (as of 2023):

Discussion Question

- Discuss the following question(s) in groups of 3-4 students
  - I will call on a few groups afterwards so please be prepared to share out
  - Be respectful of others' opinions and experiences

- The Unicode Consortium publicly solicits proposals from the public for new emoji to add to future standards
  - What do you think some of the decision factors are (or should be) in how many and which ones to add?
  - Voting is done by a combination of paid members consisting of companies, institutions, and individuals – how do you feel about who has control and how they gained that control?
    - https://home.unicode.org/membership/members/

Summary

- Humans think about numbers in decimal; computers think about numbers in binary
  - Base conversion: digit d in position i in base b has a decimal value of $d \times b^i$
    - Changing bases does *not* change value; just different representations
  - Hexadecimal (base 16, prefix `0x`) is more human-readable than binary (base 2, prefix `0b`)
  - Unit of data in a computer is **1 byte = 8 bits** = 2 hex digits
- Binary encoding can represent *anything*!
  - Computer/program needs to know how to interpret the bits
  - Encodings aren't "neutral"; priorities are baked in

Base 10	Base 2	Base 16
0	0b0000	0x0
1	0b0001	0x1
2	0b0010	0x2
3	0b0011	0x3
4	0b0100	0x4
5	0b0101	0x5
6	0b0110	0x6
7	0b0111	0x7
8	0b1000	0x8
9	0b1001	0x9
10	0b1010	0xA
11	0b1011	0xB
12	0b1100	0xC
13	0b1101	0xD
14	0b1110	0xE
15	0b1111	0xF