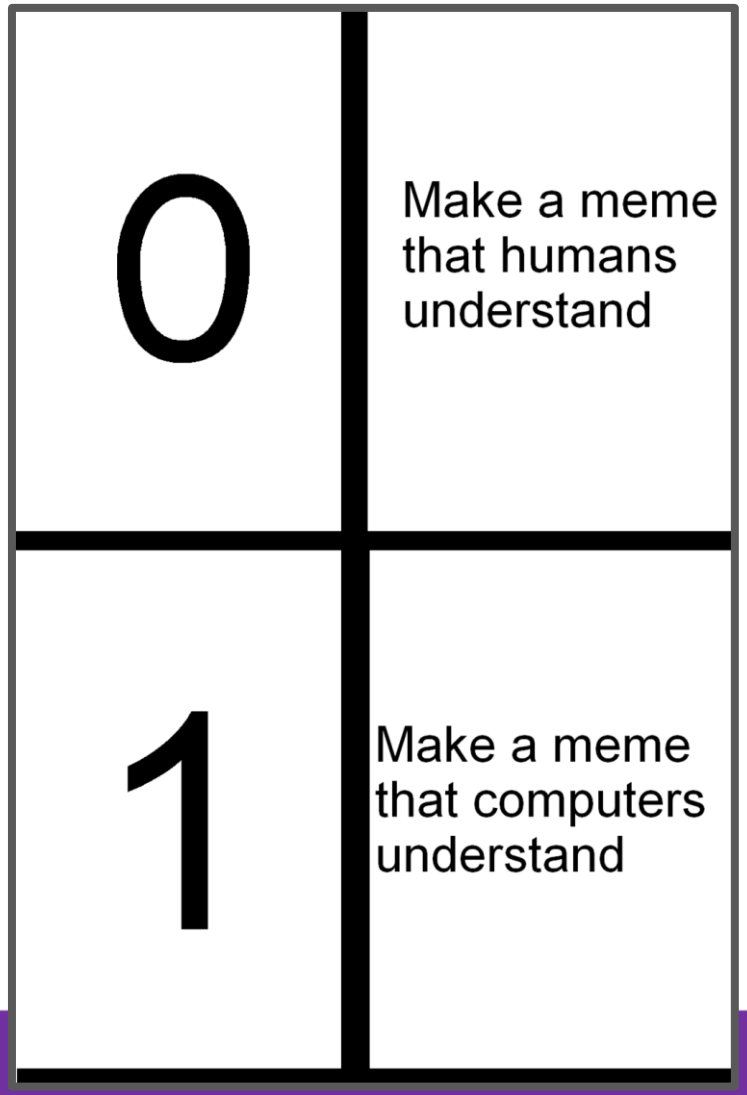
The Hardware Software Interface

CSE 351 Summer 2024

Instructor:
Ellis Haker

Teaching Assistants:
Naama Amiel
Micah Chang
Shananda Dokka
Nikolas McNamee
Jiawei Huang



Lecture Outline

- Introduction
- Syllabus/Logistics
- Binary and Numerical Representation

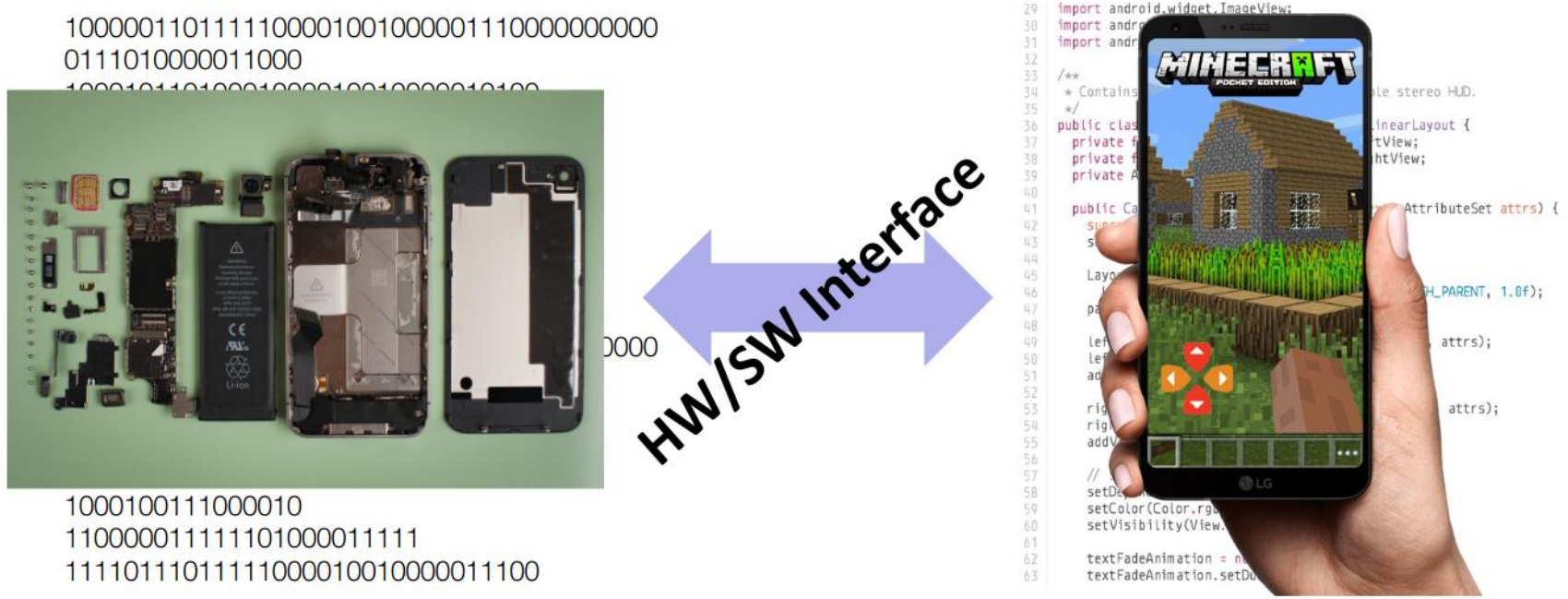
Introductions: Course Staff

- Instructor: Ellis
  - BS/MS student
  - Research focus in Operating Systems
  - Long-time TA, first-time instructor!
- TAs:
  - Naama, Micah, Shananda, Nikolas, and Jiawei
  - Available in section, office hours, and on Ed
- More than anything, we want you to feel
  - Welcome in this space
  - Able to learn and succeed in this course
  - Comfortable reaching out if you need help or want change

Introductions: You!

- ~70 students
- CSE majors, ECE majors, and more
  - Majority of the class are non-majors
  - Most of you will find almost everything in the course new
- Get to know each other! Help each other out!
  - Research suggests that group work supports learning
  - Working well with others is a valuable life skill
  - Diversity of perspectives expands your horizons
  - Take advantage of group work, where permissible, to learn, not just get a grade

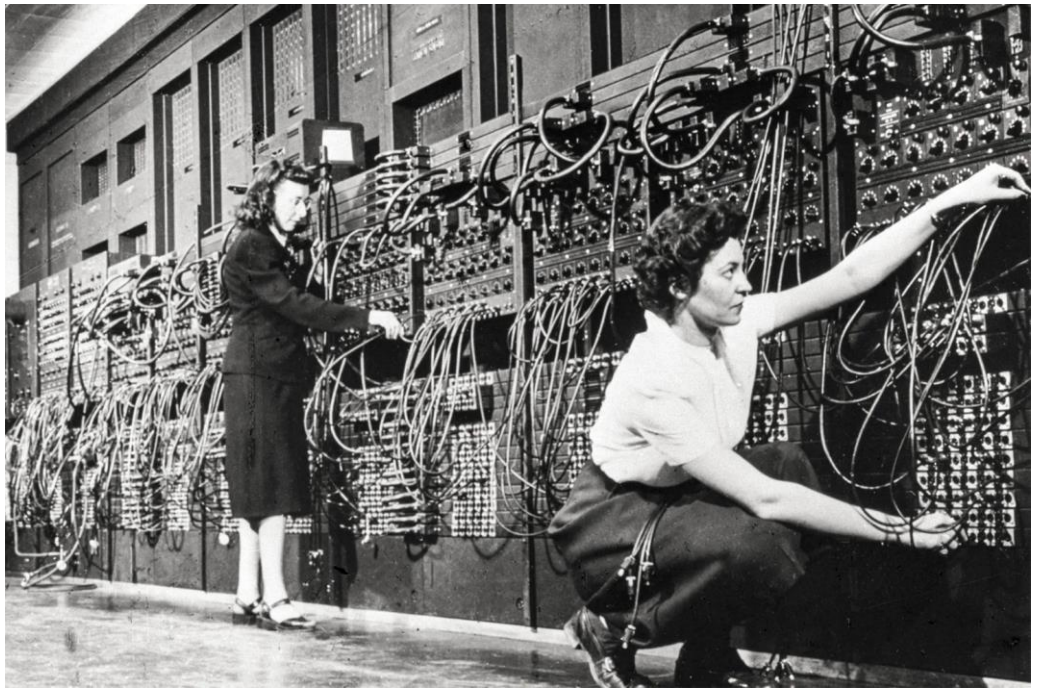
Introductions: CSE 351



- You'll learn the key abstractions "under the hood"
  - How does your source code become something the computer understands?
  - What happens as your computer is executing one or more programs?

Some History

- Hardware started out quite primitive
  - Programmed with very basic instructions
    - Very tedious!
- Software was also very basic
  - Programs reflected the actual hardware they ran on
  - Programmer had to specify each step manually



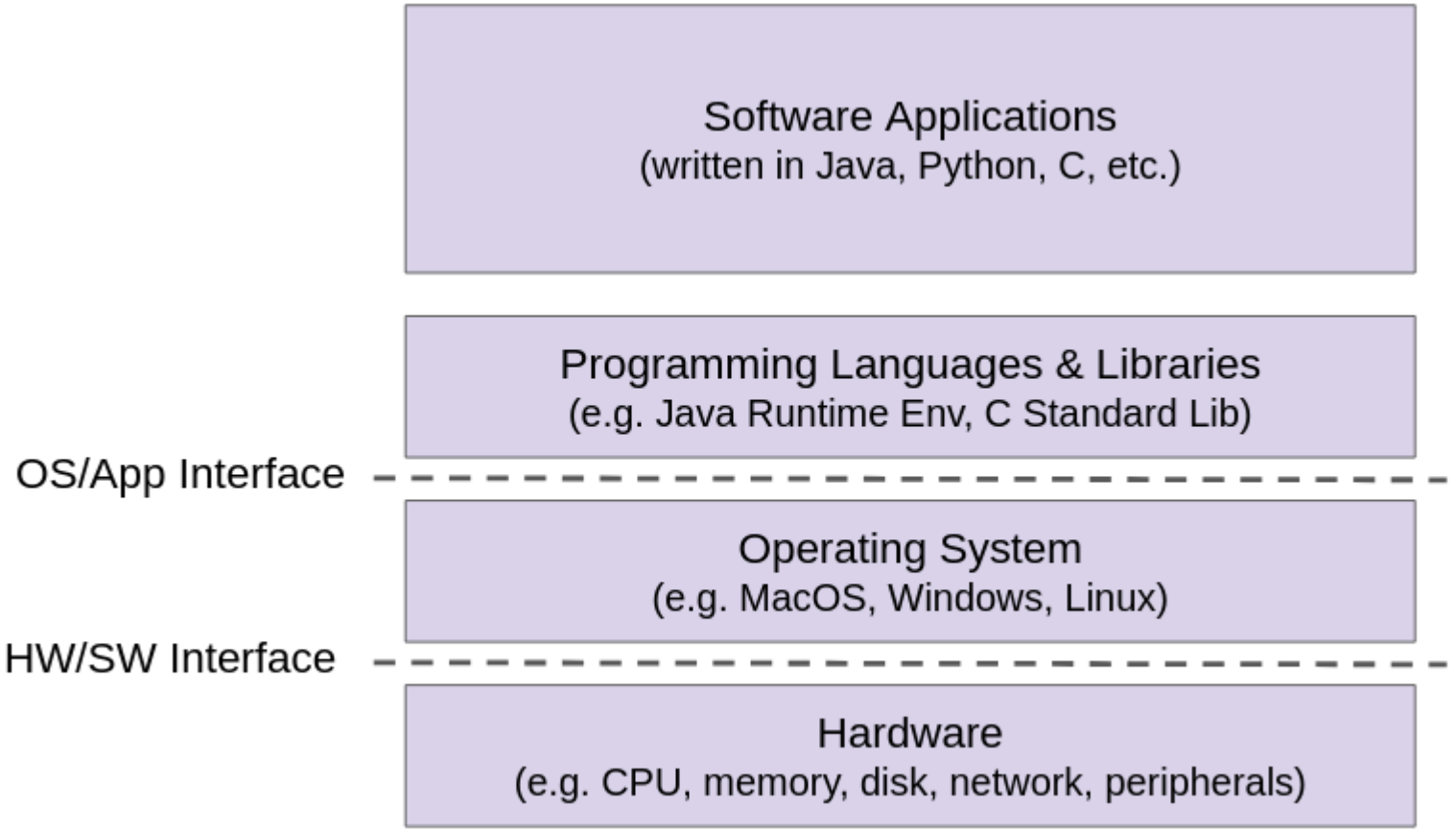
Programmers working on the ENIAC, circa 1946.

Some History pt 2

- As time went on, programming became more abstract
  - Assembly language: basic set of instructions
  - Early high-level languages: C, FORTRAN, etc.
    - Data types, arrays, loops, etc.
    - Still closer to the hardware than modern languages
  - Now: Java, Python, etc.
    - Lots of convenient features!
    - Don’t have to know much about the hardware to program

Brian Kernighan and Dennis Ritchie, creators of the original C standard.

Layers of Computing



Course Perspective

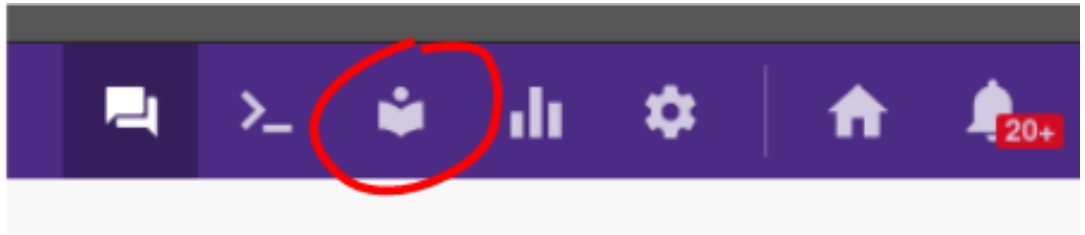
- CSE 351 will make you a better programmer
  - Learn how software really works
    - Understand some of the abstractions that exist between hardware and software, why they exist, and how they build upon each other
- Why is this important?
  - Better debugging
  - Better basis for evaluation performance

- You might miss Java, but keep an open mind!
  - Take note of anything that piques your interest. There’s lots of classes you can take to explore this space further

Syllabus/Logistics

Important Tools

- **Course Website:** https://courses.cs.Washington.edu/courses/cse351/24su
  - Syllabus, schedule, lab specs, etc.
- **Ed course**
  - Discussion: announcements, ask and answer questions
  - Lessons: readings, lecture polls, homework
- **Gradescope:** lab submissions, take-home quizzes
- **Canvas:** surveys, grade book
- **Panopto:** lecture recordings



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

Quizzes

- Three throughout the quarter
- Take-home on Gradescope
- Will have ~1 week to complete

- More information to come when we get to the first quiz

Assignments

- **Pre-lecture readings:** Ed
  - Graded on completion, due 1pm before lecture
- **Lecture polls:** Ed
  - Conducted during class, graded on completion, due 1pm before the next lecture
- **Homework:** Ed
  - Unlimited attempts, due 1pm two lectures after it's assigned
- **Labs:** Course website/Gradescope (except for Lab0)
  - ~1-2 weeks per lab, due at 11:59pm
- Assignments need to be turned in **before** the time listed. If the clock reads 11:59pm when you turn in your lab, it's late!

Late Policies

- Labs can be turned in **up to 2 days** after the deadline
  - You get **5** free late days to use throughout the quarter
  - After your free late days are used up, we will deduct 10% per day

- **We cannot accept Ed assignments after the due date!**
  - You only need to complete 80% of the readings and lecture polls for full credit
  - Your lowest homework score will be dropped

- If you have DRS accommodations that contradict these policies, I will contact you
  - Please email me if I don't reach out by the end of today

Extensions, Accommodations, Help

- We're here to help! We want you to have the best experience possible
- Extenuating circumstances
  - Students (and staff) face an extremely varied set of circumstances
  - We will try to be accommodating, but **the earlier you reach out, the better**
  - For formal accommodations go through Disability Resources for Students (DRS)
    - They can help you even for temporary disabilities (injury, etc.)
- Don't suffer in silence – talk to a staff member!
  - Go to office hours
  - Ask on the Ed board
  - Request a 1-on-1 meeting

Grade Breakdown

- **Pre-lecture readings:** 7%
- **Homework:** 20%
- **Labs:** 40%
  - Can be done individually or in pairs
- **Quizzes:** 30% (10% each)
- **Participation:** 3%
  - Includes answering lecure polls, participation in section, office hours, and Ed

Administrivia/TODOs

- Explore the course website, and read the syllabus
- Make sure you can connect to the CSE Linux environment (attu/calgary)
- (Optionally) sign up for CSE 391: System and Software Tools
- Due **Friday** (6/21)
  - Pre-course Survey (on Canvas), due 1pm
  - HW0 (Ed) due 1pm
  - RD1 & RD2 (Ed) due 1pm
- Due **Monday** (6/24)
  - RD3 (Ed) due 1pm
  - HW1 (Ed) due 1pm
  - Lab0 (Ed) due 11:59pm
- Reminder: **no class on Wednesday** due to holiday**!**

Binary

ex: in base 4, we have the symbols 1-3. after that, we go to 10 (=4), 11 (=5), etc.

Base Definitions (Review)

- If we're in base *b*, that means we have *b* possible symbols to use
  - In decimal (base 10), we have the digits 0-9
- Each digit represents a power of *b*
  - The rightmost digit always represents 1 (b^0), and it increases as we read left

    Ex: compare the number written "351" in base 10 vs base 6

$10^2 = 100$	$10^1 = 10$	$10^0 = 1$
3	5	1

In base 10, this numeral means we have 3 100s, 5 10s, and a 1.

$6^2 = 36$	$6^1 = 6$	$6^0 = 1$
3	5	1

In base 6, this numeral means we have 3 36s, 5 6s, and a 1.

Base Conversions pt 1 (Review)

- To convert from base *b* to decimal, multiply each digit by its corresponding place value, then add them

  Ex: find the decimal value of 246_8 (note, the subscript “8” denotes the base):

$8^2 = 64$	$8^1 = 8$	$8^0 = 1$
2	4	6

So the total value is 2*64 + 4*8 +6*1 = **166**

Base Conversions pt 2 (Review)

- To convert a number *N* from decimal to base *b*:

1. Find the highest power *i* such that $b^i < N$
2. Divide *N* by b^i and round down to the nearest integer. The result is the *i*th digit of our final number
3. Take the remainder, and repeat step 2 with *i*-1 until you reach *i*=0

Ex: convert the number 123 to base 4:

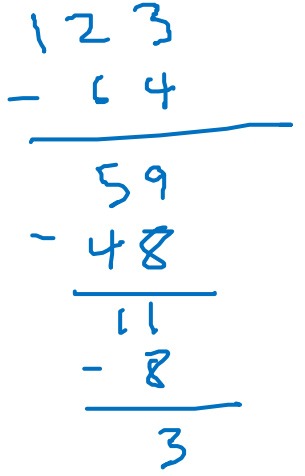
$4^3 = 64$, 123/64 = **<u>1</u>**, remainder = **59**

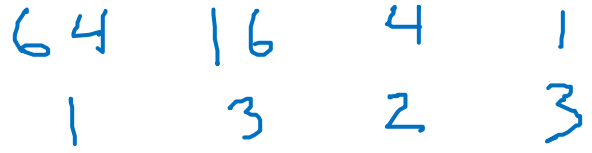
$4^2 = 16$, 59/16 = **<u>3</u>**, remainder = **11**

$4^1 = 4$, 11/4 = **<u>2</u>**, remainder = **3**

$4^0 = 1$, 3/1 = **<u>3</u>**

So the final answer is $\mathbf{\underline{1323}}_{\underline{4}}$





Review Question: Base Conversion

What is the *decimal value* of 107_8?

↖ means this # is written in base 8

A. 71

B. 87

C. 107

D. 568

$8^2 \quad 8^1 \quad 8^0$

$1 \quad 0 \quad 7_8$

$= 1 \cdot 8^2 + 0 \cdot 8^1 + 7 \cdot 8^0 = 64 + 7 = 71$

Binary

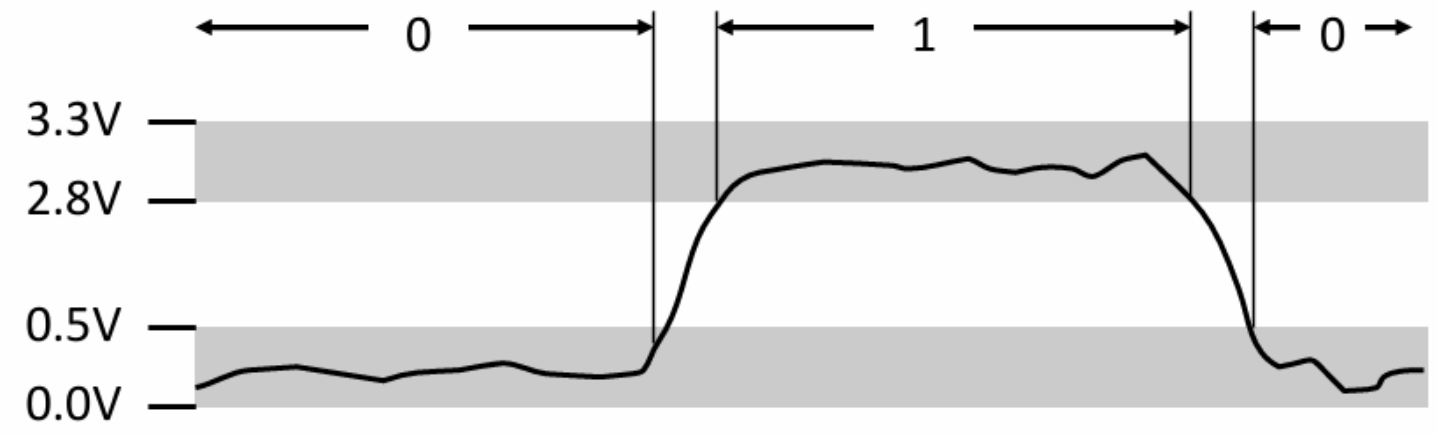
- *Humans* think about numbers in base 10, but *computers* “think” about numbers in base 2 (**binary**)

- A binary digit is called a **bit**
- A group of 4 bits is called a **nibble**
- A group of 8 bits is called a **byte**

Why Binary?

- Easy to store and read
  - For larger bases, noise would lead to inaccuracies
- Other bases are possible, but not yet viable
  - The Setun was a ternary (base 3) computer built in the USSR in 1958
  - UW research into DNA storage (base 4: A, C, G, T)
  - Quantum

ex: if we were in base 10, we'd need to split this range of voltage into 10 groups — too small



Binary and Hex (Review)

- Binary is inconvenient for humans, so computer scientists often write numbers in **base 16** (hexadecimal)
  - 16 digits: 0-9, A-F
  - Why hex?
    - Easy conversion, each hex digit = 4 bits!
      - 2 hex digits = 1 byte
- We use prefixes to denote common bases
  - 0b = binary
  - 0x = hex
  - No prefix = decimal

ex: 0, ..., 9, A, B, ..., F, 10 in base 16
= 0, ..., 9, 10, 11, ... 15, 16 in base 10

Common Base Conversion (Review)

- hex -> binary: translate each digit according to chart, then drop **leading 0s**
  - Ex: 0x2D = 0b0010 1101 = 0b101101
- binary -> hex: break into groups of 4 bits from right to left, add **leading 0s** if necessary, then translate
  - Ex: 0b101101 = 0b0010 1101 = 0x2D
- **Note:** does not work for decimal conversions!

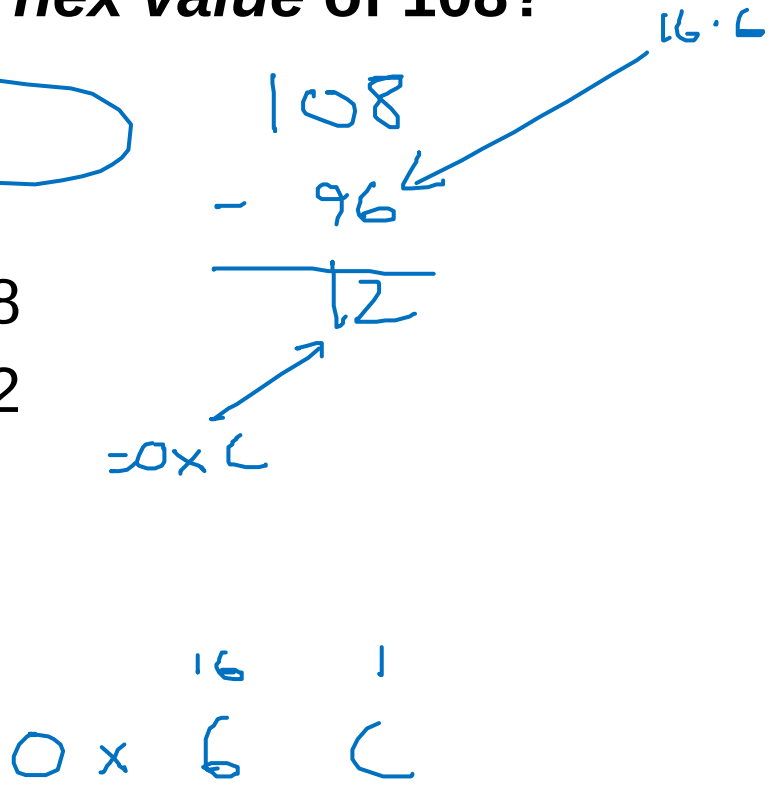
- You will probably have this table memorized by the end of the quarter :)

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

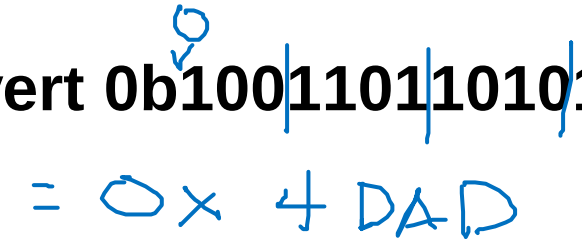
Review Questions: Binary, Hex, and Decimal

What is the *hex value* of 108?

A. 0x6C
B. 0xA8
C. 0x108
D. 0x612



Convert 0b1001101101010101 to hex.



Convert 0x3C9 to binary.

Numerical Encoding

- You can represent anything countable using numbers!
  - *But* you need to agree on an encoding
- Computers store all data as a binary number
- Examples:
  - Decimal Integers: 0→0b0, 1→0b1, 2→0b10, etc.
  - English Letters: CSE→0x435345, yay→0x796179
  - Emojis: 😀 0x0, 😞 0x1, 😎 0x2, 😇 0x3, 😈 0x4, 🤦 0x5

So What's it Mean?

A sequence of bits can have many meanings!

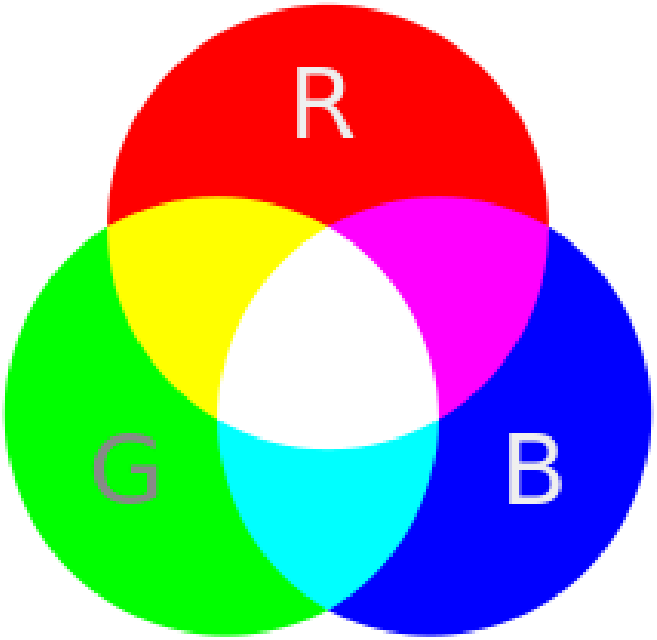
- Consider the hex sequence `0x4E6F21`
  - Possible interpretations include:
    - The decimal number 5120257
    - The real number $7.203034*10^{-39}$
    - The characters "No!"
    - The background color of this slide (sort of an olive green?)
- It's up to the program/programmer to decide how to interpret the sequence of bits

Binary Encoding Example - Colors

- RGB - Red, Green Blue
  - Additive model, represent amount of each color of light
    - 1 byte (8 bits) for each color

Ex: **Blue** = **0x0000FF**, **White** = **0xFFFFFF**

Dark Purple = **0x7030a0**



Binary Encoding Example - Text

- American Standard Information Exchange (ASCII)
- Unicode (international)

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

Source: www.LookupTables.com

Binary Encoding Example - Files and Programs

- At the lowest level, all digital data is stored as bits!
- Layers of abstraction keep everything comprehensible to humans
  - Data/files are groups of bits interpreted by a program
  - Program is *also* a sequence of bits interpreted by the CPU

- You can see the hex representation of your files in a text editor - try it!
  - Vim: `%!xxd`
  - Emacs: `M-x hexl-mode`

Summary

- All computer data is stored in **binary**
  - Humans think in **decimal**, have to convert between bases
  - **Hex** as a more human-readable base that's easy to convert
- Binary can represent *anything!*
  - Program needs to know how to interpret the bits

Some fun topics we will touch on (vote on Ed!)

A. What is a GFLOP and why is it used in computer benchmarks?
B. How and why does running many programs for a long time eat into your memory (RAM)?
C. What is stack overflow and how does it happen?
D. Why does your computer slow down when you run out of *disk* space?
E. What was the flaw behind the original Internet worm, the Heartbleed bug, and the Cloudbleed bug?
F. What is the meaning behind the different CPU specifications? (*e.g.*, # of cores, # and size of cache)