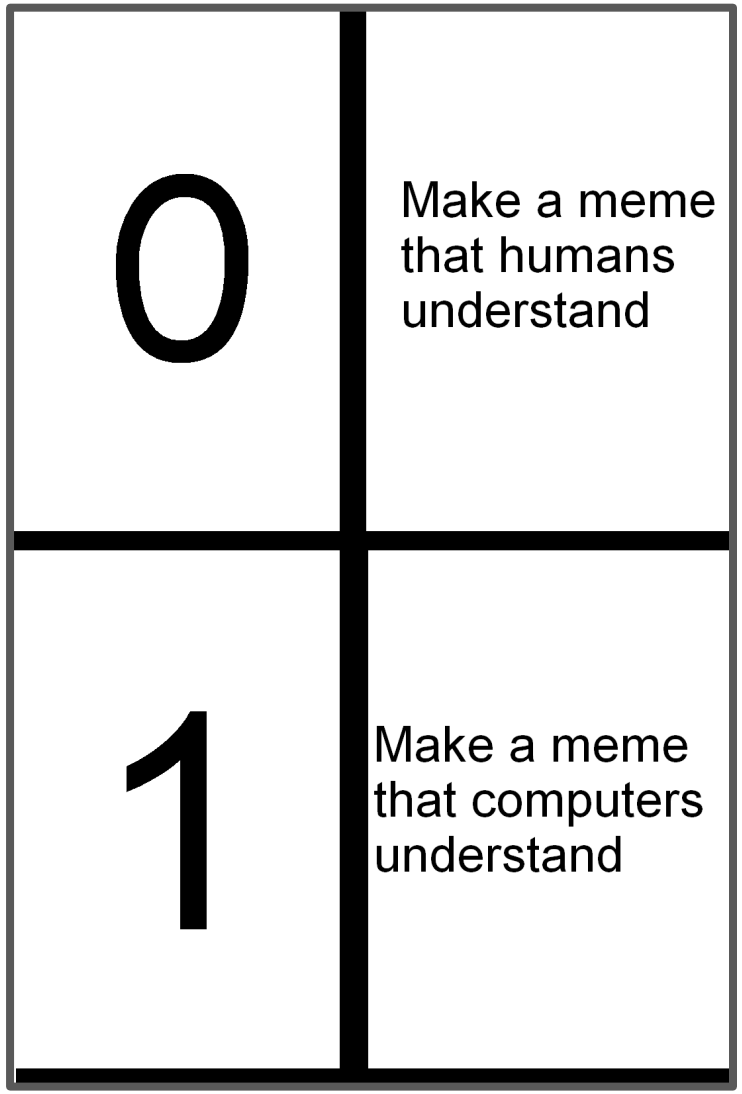
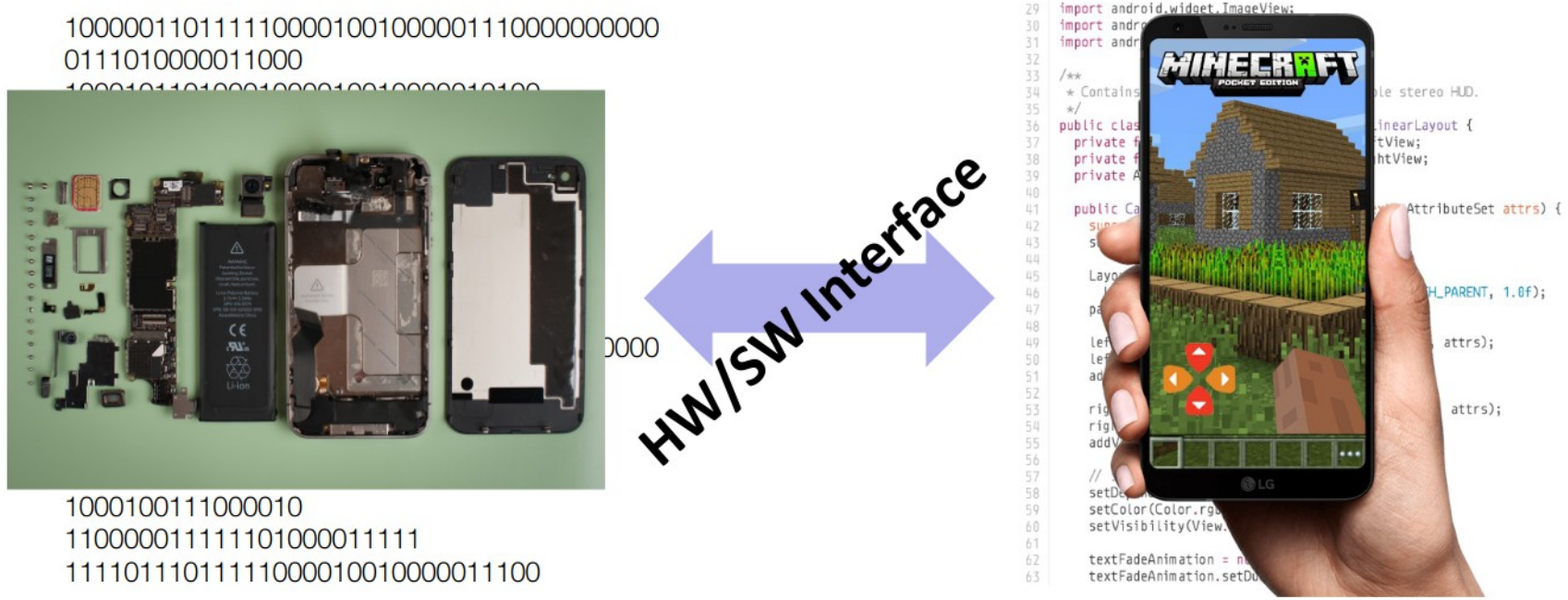
Introduction to Computer Science

Fall 2024



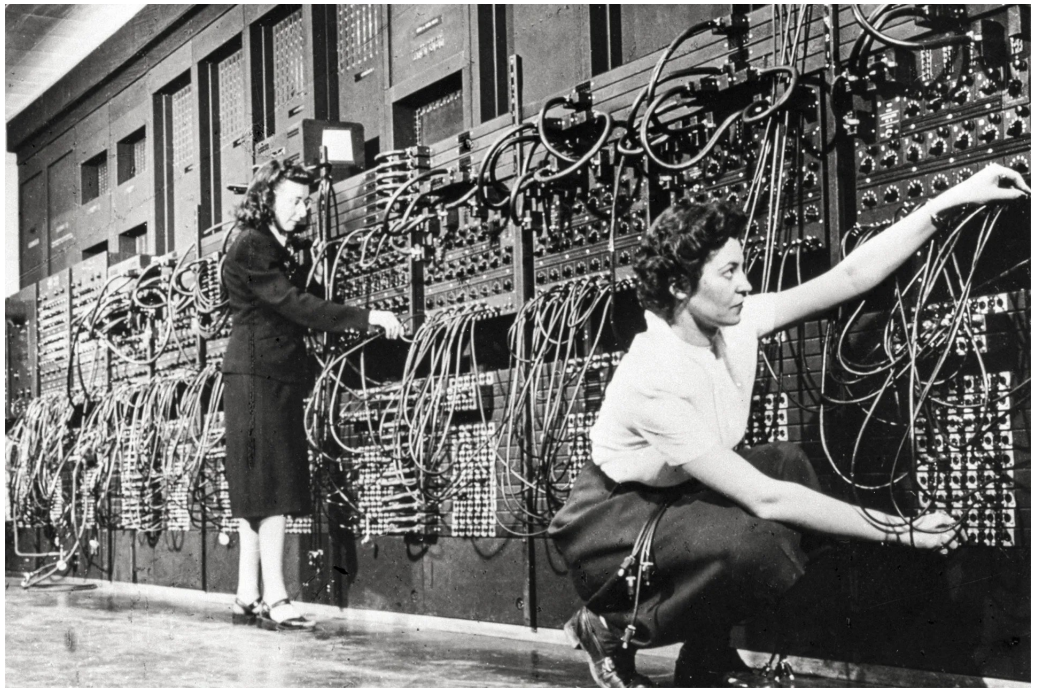
Introduction



- 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



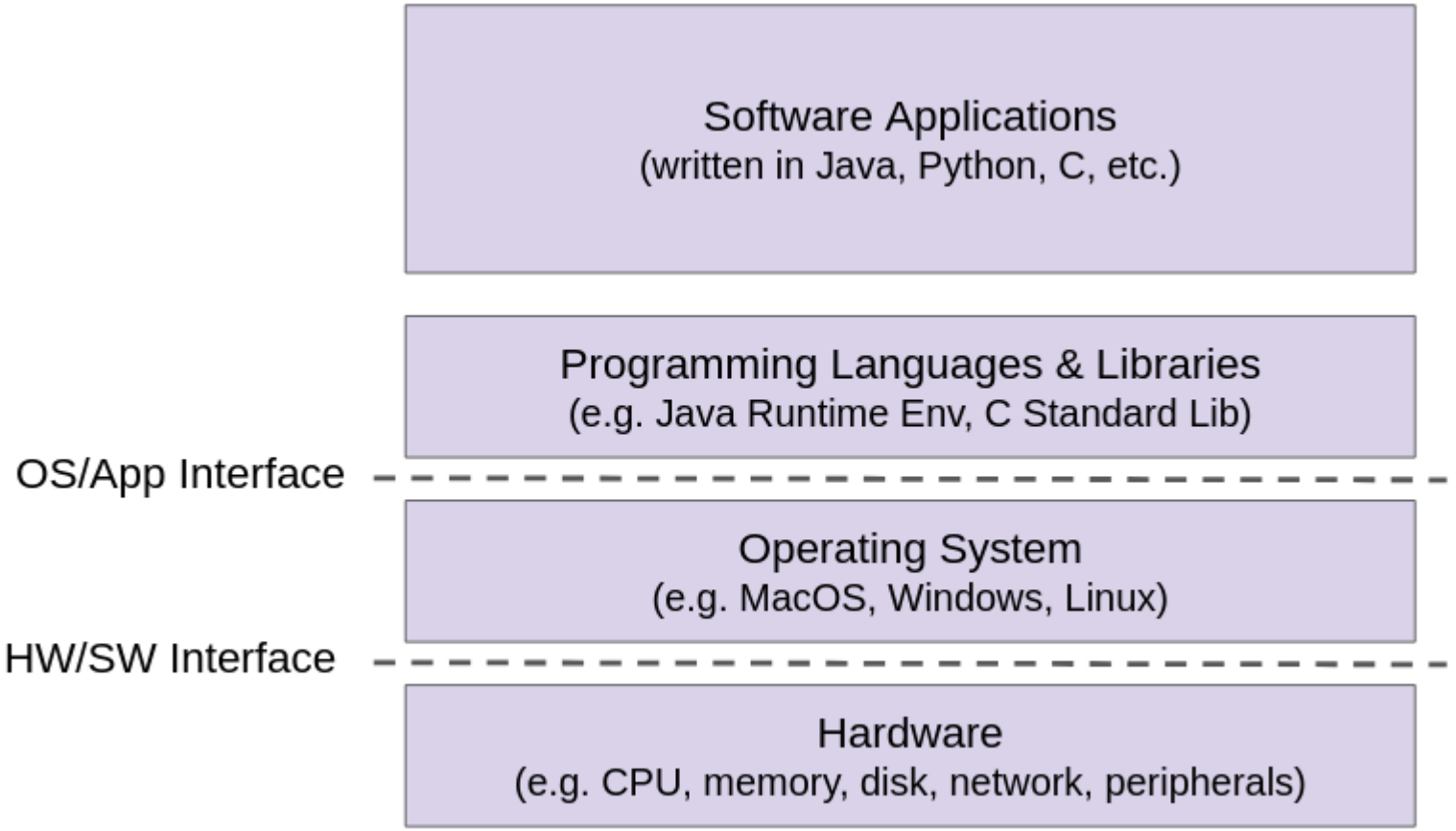
Programmers working on the ENIAC, circa 1946.

Some History

- 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

- This course 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

Binary

Base Definitions

- 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

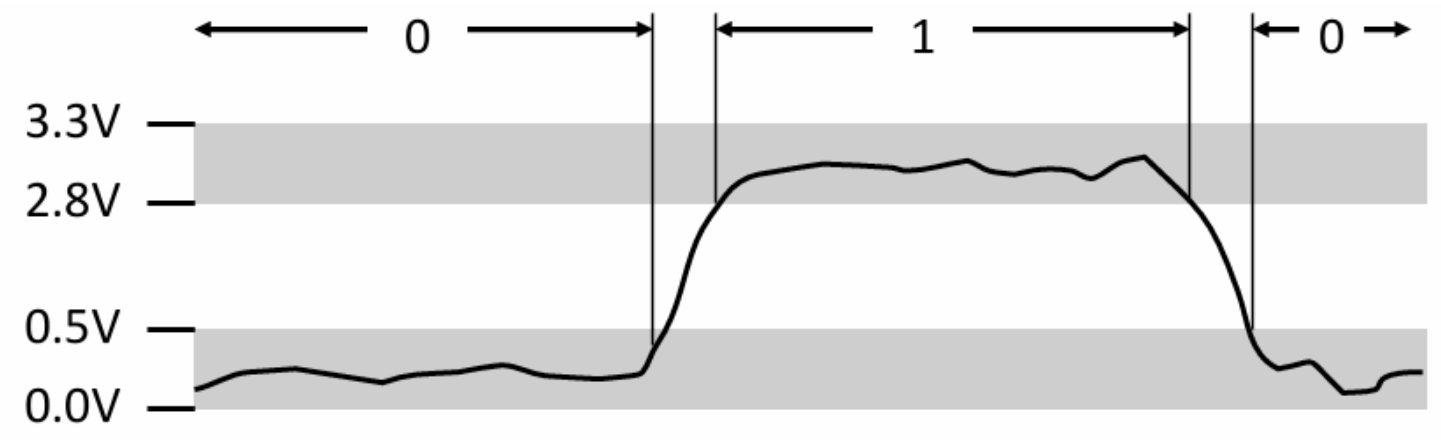
In base 6, this numeral means we have 3 36s, 5 6s, and a 1.

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?



Binary and Hex

- 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

Common Base Conversion

- 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!

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

What is the *hex value* of 108?

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

Convert 0b100110110101101 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!

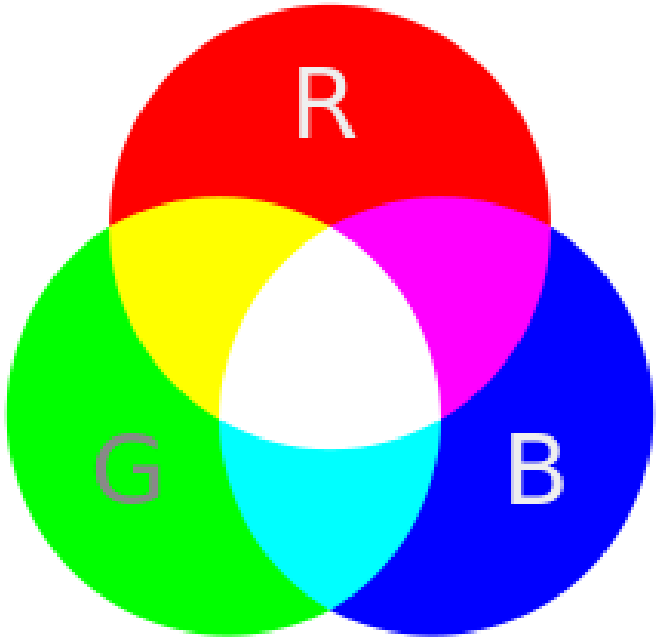
- Consider the hex sequence 0x4E6F21
  - Possible interpretations include:
    - The decimal number 5120257
    - The real number 7.203034*10-39
    - The characters "No!"
    - The background color of this slide (sort of an olive green?)
- It's up to the program/programmer to decide how to interpret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

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