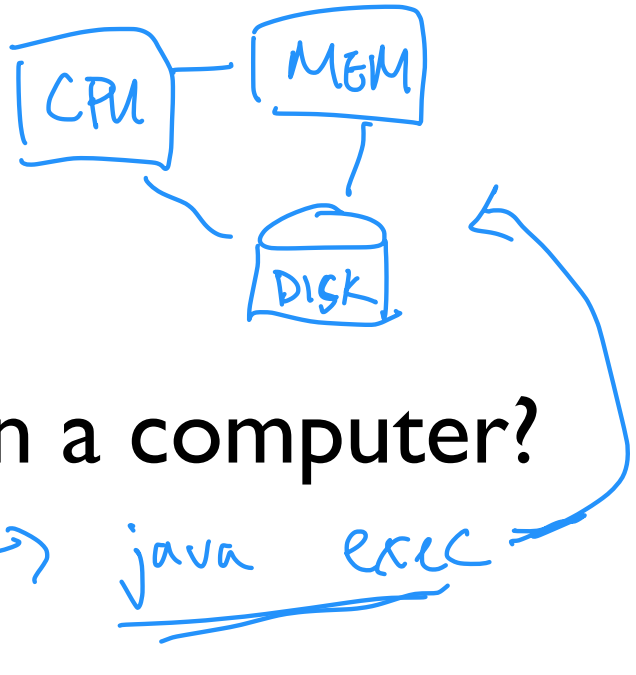
Lecture 1: Introduction

CSE 29: Systems Programming and Software Tools

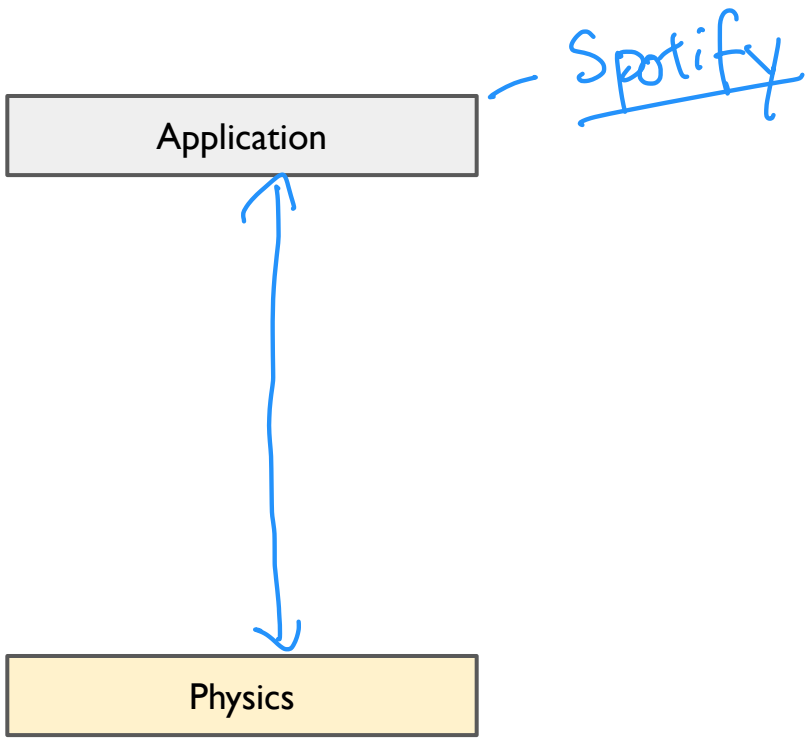
Olivia Weng



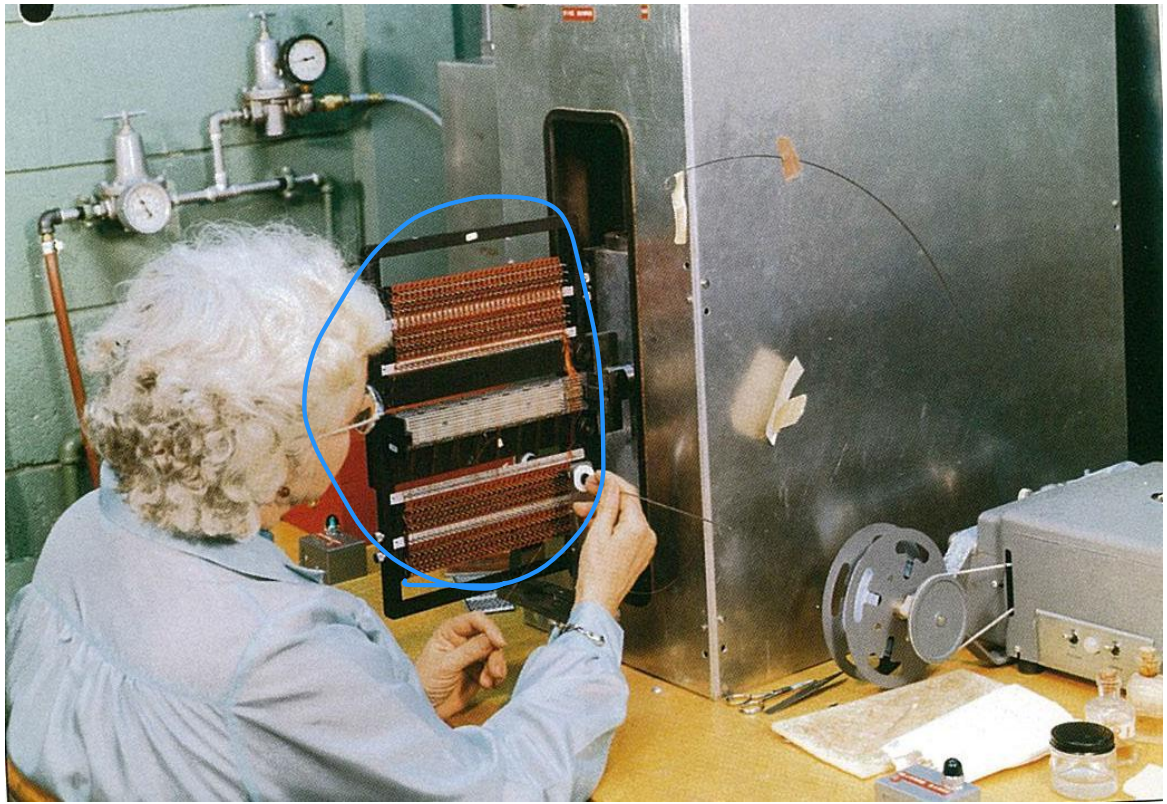
How does a program run on a computer?

write java → Compile javac → java exec

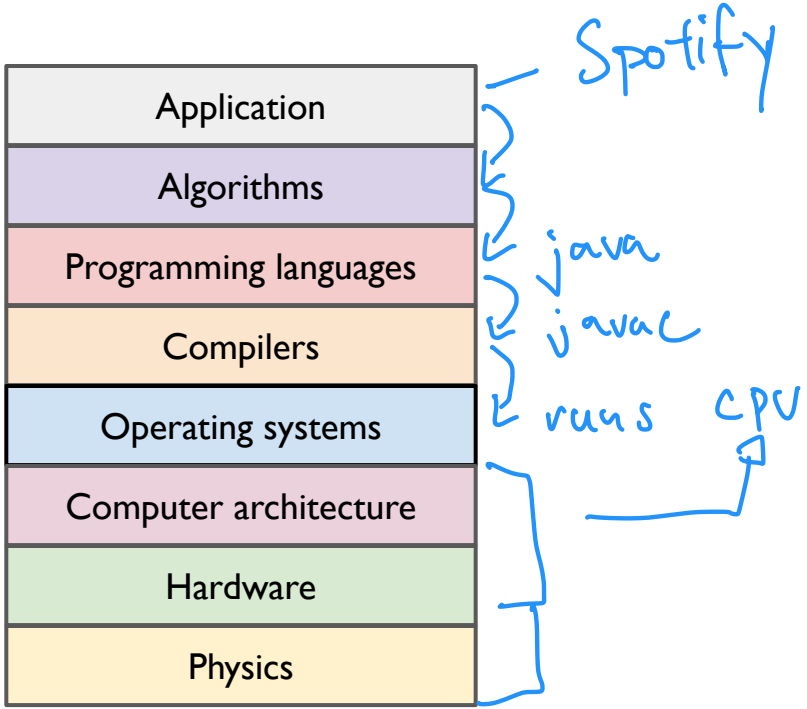
How does a program run on a computer?



Dealing with physics straight up…

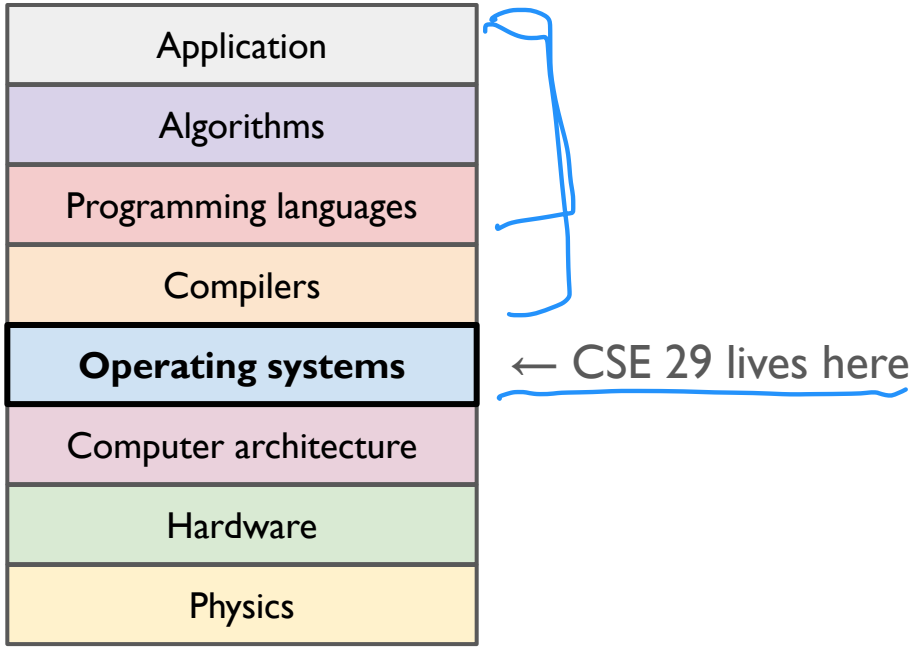


How does a program run on a computer?



How does a program run on a computer?

- Abstraction allows us to reason about complex things in a simpler way
  - CS levels of abstraction (a simple version)



Why take CSE 29 and learn systems programming?

Why take CSE 29 and learn systems programming?

- Learn how computers run programs
  - Even multiple programs at a time

- Supplement your programming skills with systems knowledge
  - Learn how to program in C!

- Take a step from being a programmer to being an engineer

drive → mechanic

Outline

- Logistics

- Intro to binary representation

Staff

- Instructor: Olivia Weng
  - Office hours: On course website
  - Co-taught with: Joe Politz
- 3 TAs shared across both classes:
  - Nick Petrone
  - Joey Wu
  - Danial Zuberi
- 4 tutors shared across both classes:
  - Miles Davis
  - Sam Gonzalez
  - Elena Tomson
  - Kevin Yang

Class Website / Syllabus

- https://ucsd-cse29.github.io/ss1-25/

Lecture

- Total 20 lectures

- Attendance is important and impacts your grade
  - + grade modifier: 17-20 lectures
  - no grade modifier: 13-16 lectures
  - - grade modifier: 9-12 lectures
  - One letter grade lower: 0-8 lectures

Lab

- Attendance is important and impacts your grade
  - Grade details on the website
- A lot happens in lab
- Lab work is completed in groups
  - Missing lab or showing up late will slow down your group

Exams

- 3 exams that will take place in Weeks 2, 3, 4, and 5
  - @ Triton Testing Center's Computer-Based Testing Facility (CBTF)
  - Makeup exams (up to 2) will be available during Finals (Friday, August 1st)

- By July 3, sign up @ prairietest.com to familiarize yourself with the testing center
  - Login with your UCSD credentials
  - Familiarization session is on Thursday, July 3rd
  - Bring your UCSD ID with you

How to Pass CSE 29

- Participate in lecture and lab

- Do the problem sets on your own
  - The exams will be very similar to the problems sets

Problem sets

- First problem set will be released before lab 1
  - Due before Week 2's lab
  - All following problems sets will be due by the next lab

- Lab 1 goes over key tools you will need to complete pset 1

Checking in

- Fill out the welcome survey!
  - Available on the website
  - Due by beginning of lab 1

- I'd like to meet with everyone once throughout the summer and chat about:
  - Questions & concerns
  - Course feedback
  - What is on your mind
  - The breeze

- I will send out a scheduling link soon to meet everyone by the end of week 5
  - Counts as attending 1 lecture!

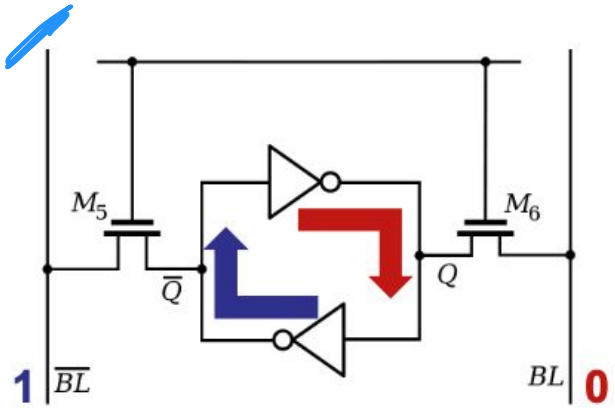
Take attendance

Time to start teaching

How do we store a string of characters inside a computer?

How do computers store characters?

- SRAM cell
  - SRAM = Static random access memory
  - Stores state of a single bit: 0 or 1
    - High state: 1
    - Low state: 0



- Why?
  - Cheap: made of sand (silicon)
  - Fast: read/write billions of times per second
  - Accurate: nearly always correctly returns the stored bit
  - Small: fit billions inside a computer chip

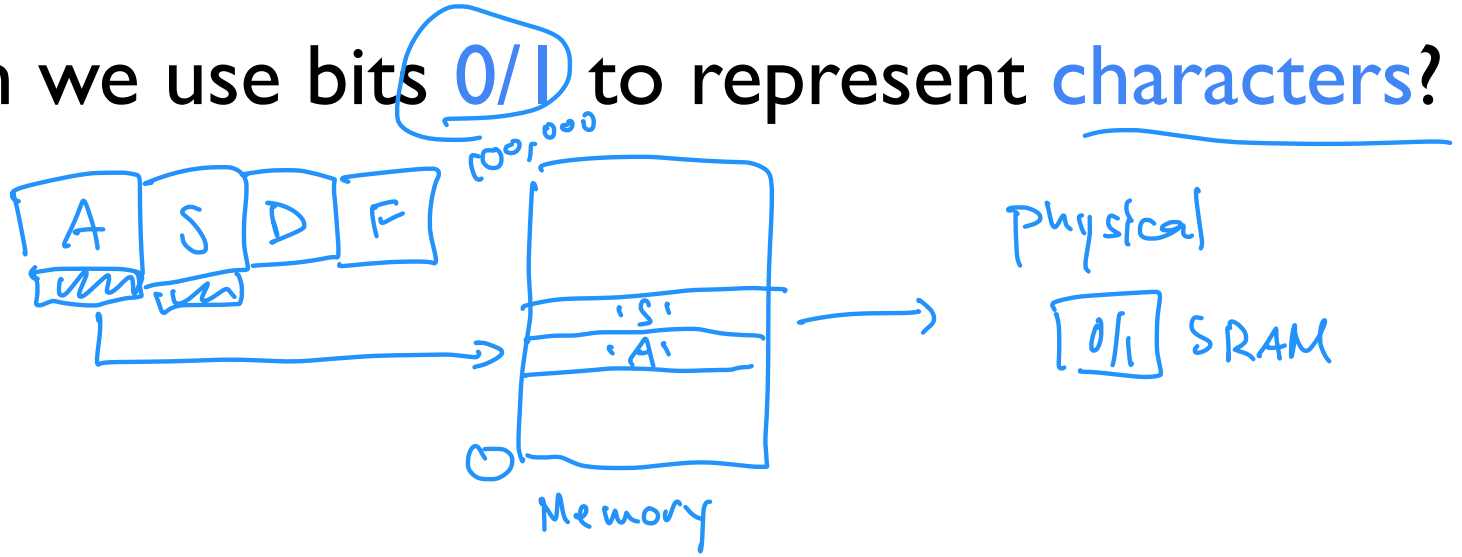
0 = A

1 = B

00 = C

11 = D

How can we use bits 0/1 to represent characters?

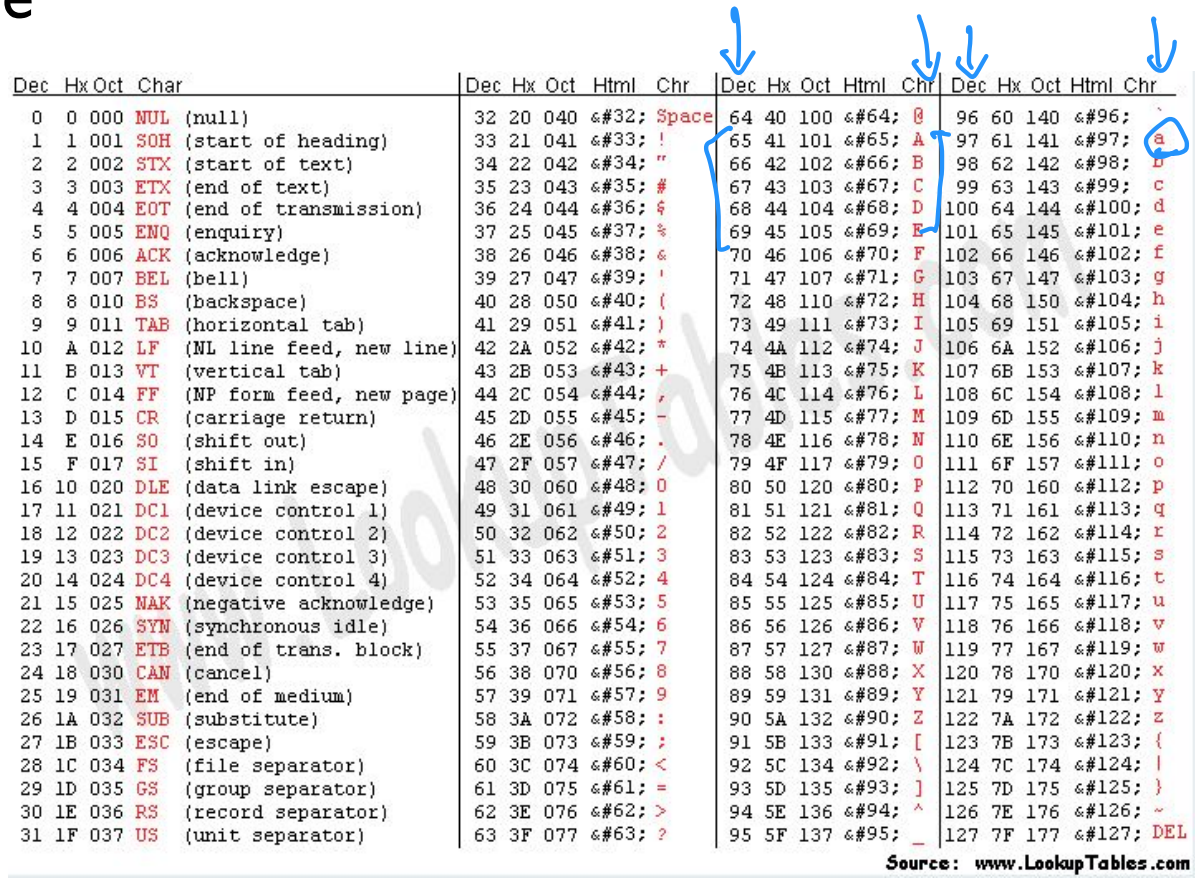


How can we use bits 0/1 to represent characters?

- Binary (0/1) number system
  - Use only 0 and 1 to represent numbers!

- Assign a number to each character
  - ASCII Encoding: [0-127] represents all English characters

ASCII Table



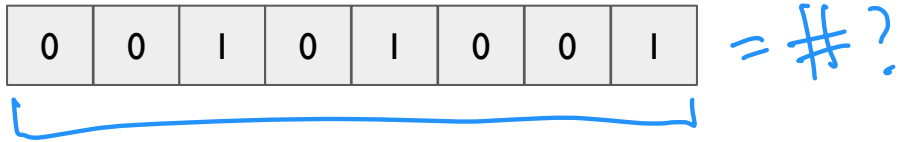
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of 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

How to represent numbers in binary?

- Two (binary) electrical states
  - High (on) = 1
  - Low (off) = 0

- A number is an array of RAM cells in binary states:



How to represent numbers in binary?

- What number is this?

128

$1 \times 10^2 + 2 \times 10^1 + 8 \times 10^0$

$100 + 20 + 8$

$= 128$

Decimal

Base 10

$0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$32 + 8 + 1 = 41$

How to represent numbers in binary?

- What does each bit's binary state mean?
  - Recall:
    - High (on) = 1
    - Low (off) = 0

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	1	0	1	0	0	1

= 41

How to represent numbers in binary?

- What number is this?

128 16 2

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	0	1	0	0	1	0

you

$= 146$

$0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

How to represent numbers in binary?

- What number is this?

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	0	1	0	0	1	0

= 146

How to represent numbers in binary?

- What number is this?

4 + 2

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	1	1	0

= 6

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	1	0	1	0	0

= 84

64 + 16 + 4

How to represent numbers in binary?

- What number is this?

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	1	1	0

= 6

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	0	1	0	1	0	0

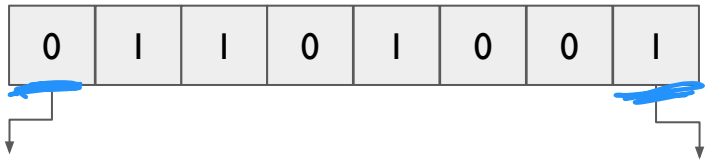
= 84

How to represent numbers in binary?

- Some binary definitions:

1 byte = 8 bits

*1 nibble = 4 bits



Most significant bit (MSB)

Least significant bit (LSB)

- Why is it called MSB/LSB?

How many values can be represented in binary?

1 bit

0 1

$2^{1 \text{ bit}}$ values

0

1

2 bits

0 1 0 1

0 0 = 0

0 1 = 1

1 0 = 2

1 1 = 3

$2^{2 \text{ bits}} = 4$ values

3 bits

2^3

How many values can be represented in binary?

1 bit

Binary	Value
0	0
1	1

$2^{1 \text{ bit}} = 2$ values

2 bits

Binary	Value
00	0
01	1
10	2
11	3

$2^{2 \text{ bits}} = 4$ values

3 bits

Binary	Value
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

$2^{3 \text{ bits}} = 8$ values

Demo 1

- `hello world!`

- `print_abc_char_and_int()`

- `uint8_t bin8_to_dec(char bin_arr[])`