Data Representation and Binary Numbers

•••

Beste Filiz Yuksel

To make computers easier to build and keep them reliable, everything is represented using just two values.

These two values are represented as 0 and 1.

But on a computer they are represented by two states e.g., in memory a low or high voltage is used to store each 0 or 1.

Computer Storage: Binary System

→Computers represent everything as binary numbers: sequences of ones and zeros

→A bit is a unit of information: 0 or 1

→“Bit” is short for “binary digit”

→A bit is like a lightswitch or a warning light: ON or OFF

→A sequence of bits looks like: 010000100101101111011101100011101

→All data in computers is stored this way:

numbers, text, photos, music, videos, …

Data Representation: Binary System

Every file, picture, download, digital recording, web page is just a whole lot of bits.

These binary digits are what make digital technology *digital*!

Representing numbers in Base 10 (decimal) system

The number system that humans normally use is in base 10 (also known as decimal).

In decimal, the value of each digit in a number depends on its place in the number. For example, in \$123, the 3 represents \$3, whereas the 1 represents \$100. Each place value in a number is worth 10 times more than the place value to its right.

10^4	10^3	10^2	10^1	10^0
10000	1000	100	10	1

Representing numbers in Base 2 (binary) system

The number system that computers use is in base 2 (also known as binary).

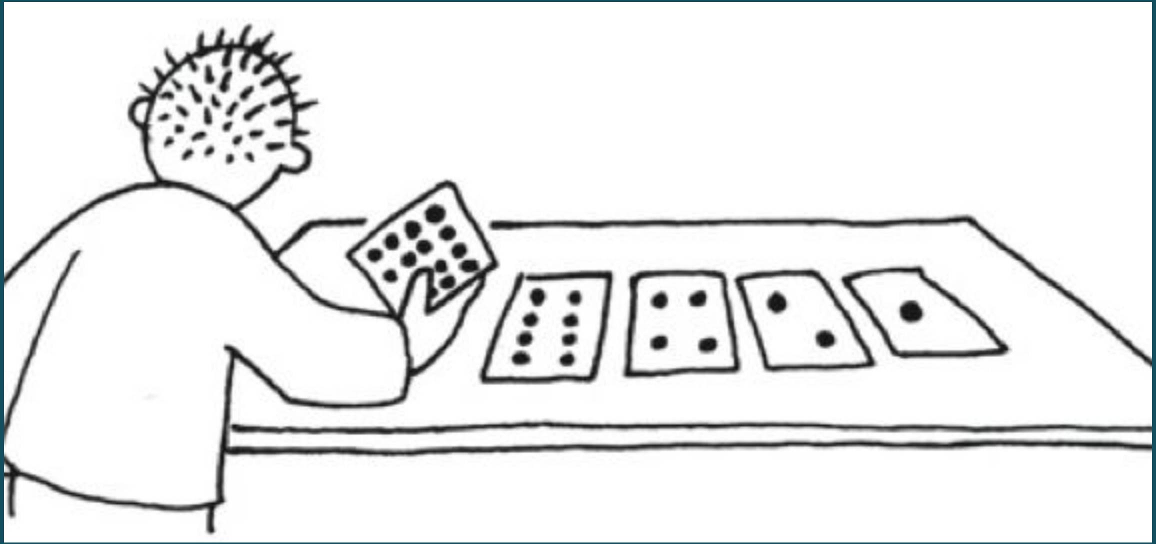
In binary, the value of each digit in a number depends on its place in the number.

Each place value in a number is worth 2 times more than the place value to its right.

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1

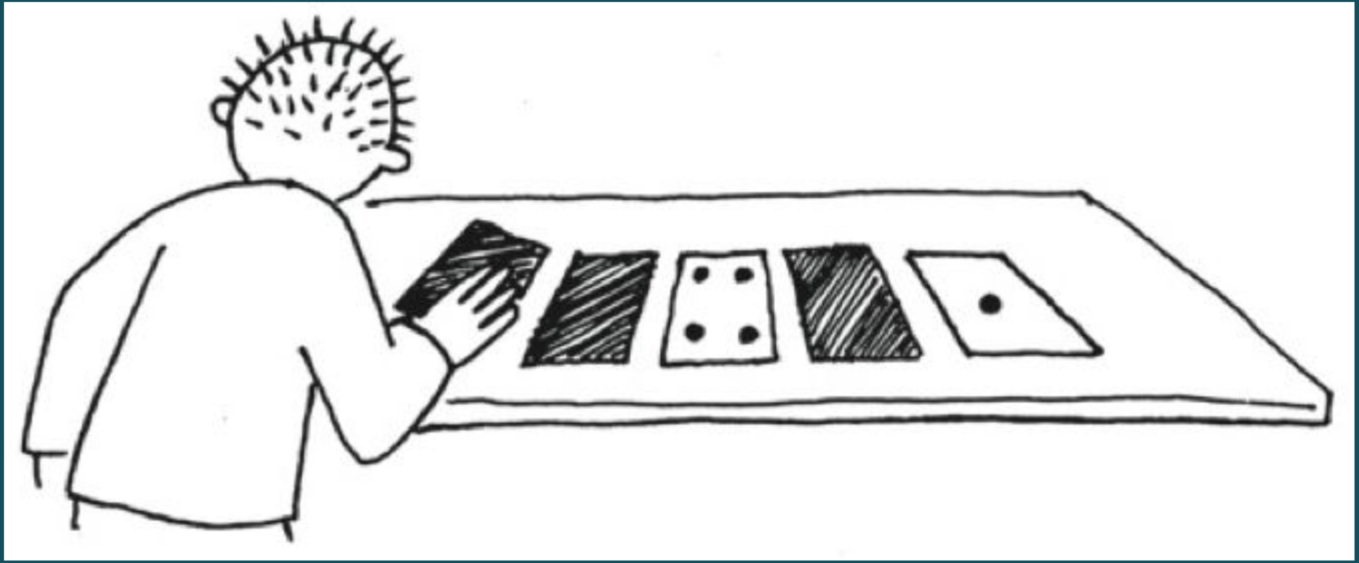
Class Exercise: Representing Numbers Using Bits

Lay out your cards:



Class Exercise: Representing Numbers Using Bits

Flip your cards so that 5 dots show only (don’t re-order your cards):

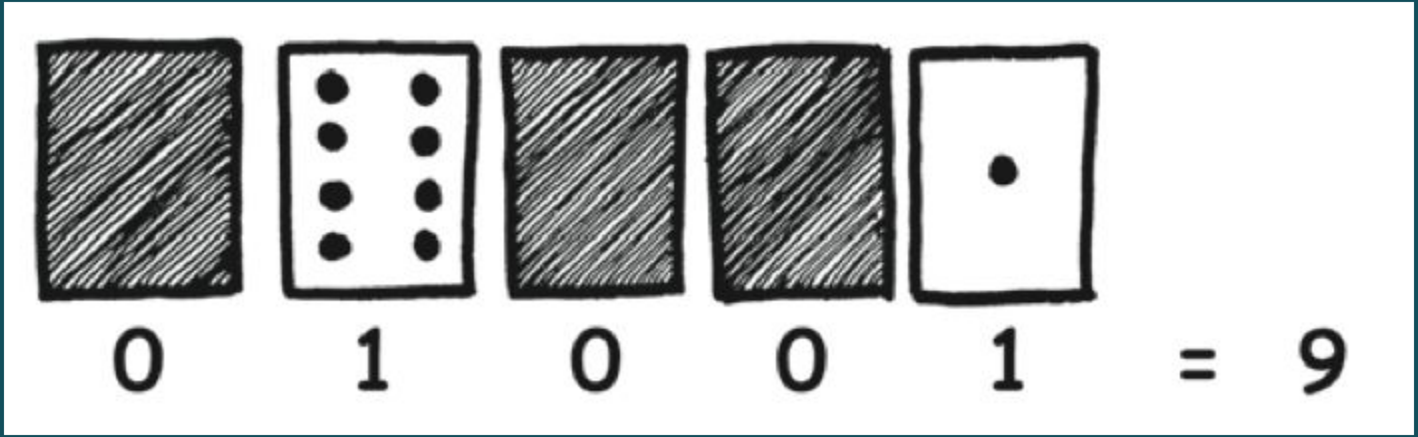


Binary Number System

Now flip your cards so that 9 dots show

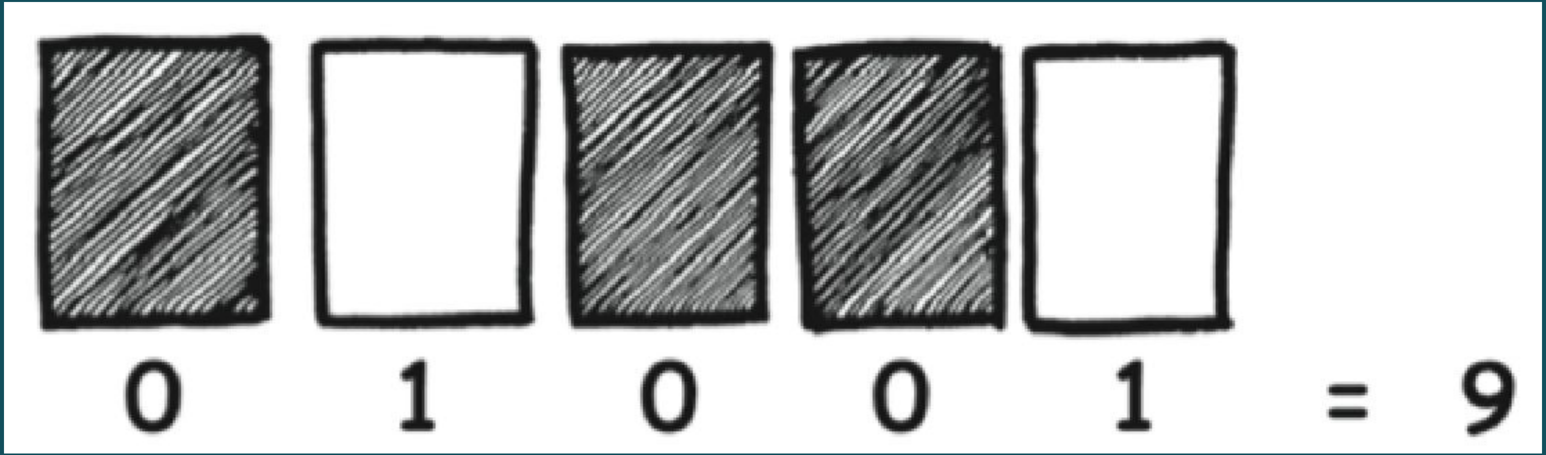
Binary numbers use 0 (zero) and 1 to represent whether a card is face-up or face-down: 0 = face-down, 1 = face-up.

Example: 01001 represents 9



Binary Number System

Don’t really need the dots on the cards. Just need card’s position and whether it is facing up or down.



Class Exercise: Representing Numbers Using Bits

Flip cards so 10 dots show. This represents 10.

Represent these numbers: 3, 12, 19, 0, 7

Is there more than one way to represent any number?

What is the biggest number you can represent?

What is the smallest?

Are there any numbers between the biggest and smallest that you can’t represent?

How many numbers in total can you represent with 5 cards?

Counting Beyond 31

- If you added a sixth card, how many dots would it have?

  What would be the biggest number your cards could represent? And the smallest?

  How many numbers total can you have with six bits?

- If you added a seventh next card, how many dots would it have?

  What would be the biggest number your cards could represent?

  How many numbers total can you have with seven bits?

Counting Beyond 31

Only a few cards are needed to count up to very big numbers.

Six cards (6 bits) can count 0 to 63.

Seven cards (7 bits) can count 0 to 127.

Eight cards (8 bits) can count 0 to 255.

Nine cards (9 bits) can count 0 to 511.

Ten cards (10 bits) can count 0 to 1023.

N cards (N bits) can count 0 to $2^N - 1$.

Bits and Bytes

Bits are usually organized into groups of 8:

Eight bits is called a ***byte.***

1024 bytes in one kilobyte (1KB)

1024 KB in one megabyte (1MB)

1024MB in one gigabyte (GB)

1024GB in one terabyte (TB)

But hard drive manufacturers round down to make things easier and provide less storage space.

This means:

1000 bytes in one kilobyte (1KB)

1000KB in one megabyte (1MB)

1000MB in one gigabyte (GB)

1000GB in one terabyte (TB)

But computers store data digitally that represent more than just numbers, such as:

Text

Images

Music

What’s in a Byte?

Depends on how you ***interpret*** it.

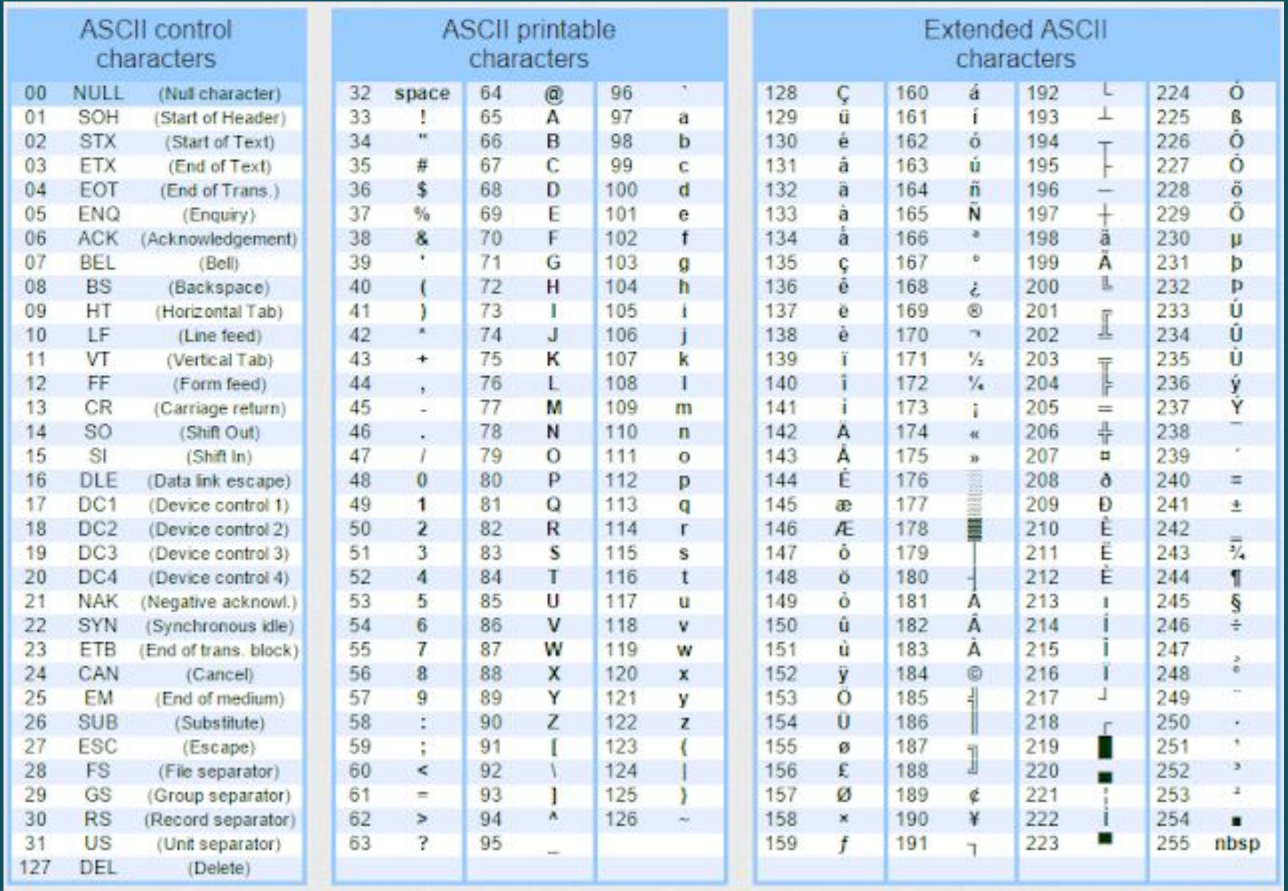
One byte can represent:	Two bytes can represent:
→Integer: 0 – 255, or -128 – 127	Integer: 0 – 65535, or -32768 – 32767
→Text character: a-z, A-Z, $, ; …	Text character: symbols from many languages
→Image pixel: 256 different colors	Image pixel: 65536 different colors

Negative numbers are encoded using *two’s complement*.

Real numbers are encoded in *floating-point notation.*

Bytes as Characters (Text): ASCII

American Standard Code for Information Interchange



ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Bytes as Characters (Text): ASCII

American Standard Code for Information Interchange

For example, the word “computers” (all lower-case) would be

01100011 01101111 01101101 01110000 01110101 01110100 01100101 01110010 01110011.

This is because "c" is "01100011", "o" is "01101111", and so on.

Have a look at the ASCII table above to check that that’s right!

Concluding Remarks

While binary notation is used extensively, and you may often hear the data being referred to as being “0’s and 1’s”, it is important to remember that a computer does *not* store 0’s and 1’s, it has no way of doing this.

Itis just using physical mechanisms such as high and low voltage.

Concluding Remarks

Computer scientists don't spend a lot of time reading bits themselves, but knowing how they are stored is really important because it affects the amount of space that data will use, the amount of time it takes to send the data to a friend, and the quality of what is being stored.