How Computers Store Information

and seeding the idea of problem solving

Sukrit Gupta

January 11, 2024

Outline

Acknowledgement and disclaimer

What you will see today is how I feel about these things today.
All opinions (if any) are mine.
I am human and make mistakes (sometimes more than the average human).

Section 1

Bits

Human and digital communication

- How do humans communicate (when there is no digital media involved)?
- Verbal and written forms?
- How do we communicate with computers and vice versa?
- What language do they understand?
- **Bits** = **Bi**nary Digi**ts**

Why bits?

- Computers have evolved over several decades. Why did they not just have decimal digits?
- What is the primary input to computers?
- Electricity! Electricity can be used to represent information. How?
- Computers store electricity to represent information. Presence = 1; absence = 0.

BTW, why are most modern computers binary?

- In electronics, a voltage level or current flow is a way to represent a value.
- Let's suppose that we want to try out the decimal system instead of the binary system. How would we represent this in electronics?
- For a range of 0-5V voltage, we could divide it into 10 steps (0V, 0.5V, 1.0V, and so on). So what's the problem?
- $Energy = \int_t Power = \int_t Current \times Voltage$. So more voltage means more energy expended. But we can always make the range smaller (0-1V instead of 0-5V)?
- Electronic signals are not always steady and can be influenced by the surroundings internal circuits. Thereby, 0.5 could become 0.75 and it would be hard to understand whether it is 1 or 2.

Counting with bits

- How many numbers can you count with 1 bit?
- How about 2 bits?
- Now how about 3 bits?
- Now how about 10,000 bits? 2^{10000}
- Now how about 1 million bits?

Section 2

Different representations of numbers

Let us have a closer look

- In the decimal system, the number 125 = 100*1 + 10*2 + 1*5. 10^0, 10^1, 10^2, ... are the bases. What are the bases in the binary system?
- In the binary system, the bases are 2^0, 2^1, 2^2, ...
- How do we think about the number 6?
- $6 = 2^2*1 + 2^1*1 + 2^0*0$.

Exercise: Convert decimal 77 to binary number

Decimal number 77 = $2^6*1 + 2^5*0 + 2^4*0 + 2^3*1 + 2^2*1 + 2^1*0 + 2^0*1$

= 1001101

Binary, octal, decimal, hexadecimal number systems

Table: Conversion

Base 2 (binary)	Base 10 (decimal)	Base 8 (octal)	Base 16 (hexadecimal)
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	10	8
1001	9	11	9
1010	10	12	A
1011	11	13	B
1100	12	14	C
1101	13	15	D
1110	14	16	E
1111	15	17	F

In modern computing and digital electronics, the most commonly used bases are decimal (base 10), binary (base 2), octal (base 8), and hexadecimal (base 16).

Data conversion in different bases

I gave a few examples of how to convert from decimal to binary. How to do this for:

- Decimal to Octal
- Decimal to Hexadecimal

Exercise: Convert decimal 77 to octal and hexadecimal forms

Binary form:
Decimal number 77 = 2^6*1 + 2^5*0 + 2^4*0 + 2^3*1 + 2^2*1 + 2^1*0 + 2^0*1
= 1001101

Octal form:
Decimal number 77 = 8^2*1 + 8^1*1 + 8^0*5.
= 115

Hexadecimal form:
Decimal number 77 = 16^1*4 + 16^0*D.
= 4D

Okay, we have seen how computers represent numbers. How about alphabets?

Section 3

Universal Character Encoding Standards

ASCII: American Standard Code for Information Interchange

Dec	Char	Dec	Char	Dec	Char
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	_	127	DEL

Figure: The (incomplete) ASCII table[1]

- We can give any arbitrary code to digits/alphabets and use the code to represent them.
- Can anyone tell me how many bits do you need to represent an ASCII character?
- Actually, ASCII started with 7 bits but ended up having 8 bits (1 parity bit) later. 8 bits = 1 byte.
- What does 89 79 33 mean?

[1] https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html

ASCII is still insufficient. Why?

- Doesn't support other languages.
- Lot of languages use accents that are absent in ASCII.
- Emoticons are also missing!

Unicode comes into picture

- So Unicode was introduced in the late 1980s.
- It uses higher number of bits to represent characters/symbols/emoticons.
- As of Jan, 2023 (Unicode 14.0), there are 149,186 characters that are covered in Unicode.

Figure: Represented by Unicode: 127846

Okay, enough about text. What about images?

Images

Figure: [1]

Can anyone guess what videos are?

[1] Image courtesy: The Office (US)

Sound

Sound Waves in the Air $->$ Transducer $->$ Electrical Form $->$ Analog to Digital Converters $->$ Digital Signal

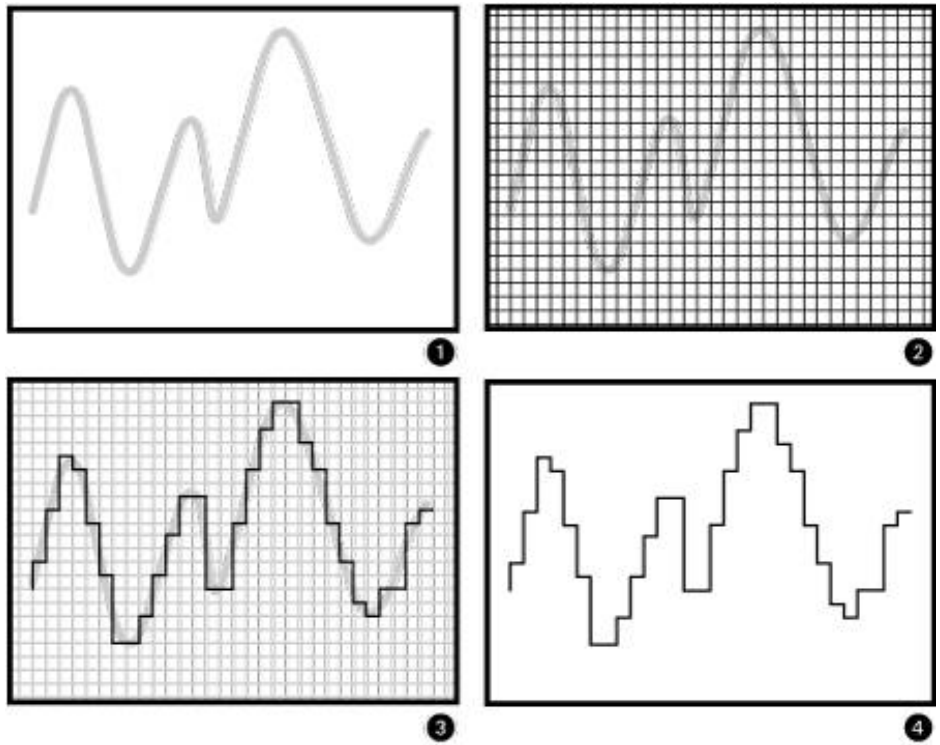

Figure: [2]

[2]Image courtesy: Sound in Digital Form

Section 4

Python (and why learn it!)

Brief History of Python

- We use Python as a vehicle to present concepts related to computational problem solving and thinking.
- I am not a fan of Python, just want you start your programming journey with something where I can convey computational ideas instead of focusing on the syntax all the time.
- Python is a living language. Since its introduction by Guido von Rossum in 1990, it has undergone many changes.
- With the arrival of Python 2.0 in 2000, a large number of people began developing libraries that interfaced seamlessly with Python, and continuing support and development of the Python ecosystem became a community-based activity.
- Python 3.0 was released at the end of 2008. Cleaned up many of the inconsistencies in the design of the various releases of Python 2 (often referred to as Python 2.x). Not backward compatible.

Upsides of learning Python

- **It's old**: Python has been around since the nineties. That doesn't only mean that it has had plenty of time to grow. It has also acquired a large and supportive community.
- **It's beginner-friendly**: The syntax of Python is very human-readable and many other features make it easy for beginners.
- **It's versatile**: Since Python has been around for so long, developers have made a package for every purpose. These days, you can find a package for almost everything.

Downsides of Python

- **Speed**: Python is slow (Like, really slow). On average, you'll need about 2–10 times longer to complete a task with Python than with any other language.
- **Mobile Development**: As we're witnessing the shift from desktop to smartphone, it's clear that we need robust languages to build mobile software.
- **Runtime Errors**: A Python script isn't compiled first and then executed. Instead, it compiles every time you execute it, so any coding error manifests itself at runtime. This leads to poor performance, time consumption, and the need for a lot of tests.

Thank you!