Representing Data

Shaan Fulton

June 30, 2025

Data as Bits

Computers must store data. This requires translating physical states into complex data states. There are many ways to do this.

For instance, we could design an electronic component that has 128 different switches so that we can easily store all the ASCII characters (we could maybe extend it later). We'll call it the Super Switch 128. You would simply string these together to store strings of data. But how would we design this component? And mass produce it? As it turns out, this is quite difficult. And what if we wanted to add more characters? The emoji-capable computer may require re-architecting to support a Super Switch 1000!

It is much easier to store just two voltage states: on and off. The vacuum tube, as well as the transistor, does precisely this. But how do we store the number 2? Seems we can only store 0 and 1. Interestingly, we've already solved this problem. In pre-school, in fact. We only have 10 digits to play with (0,1,2,3,4,5,6,7,8,9). Yet somehow, we can represent trillions of numbers! This concept is the **positional number system**. We represent different digit values based on their position!

There are many **positional number systems**. The main parameter between them is their base. We're used to base-10: We have 10 digits, so each positional shift adds 10 so we can reset our rightmost digit to 0 and make 11 as well as the numbers beyond. With only two digits available in our on and off (binary) system, we can switch to base-2. Now we carry over with each positional shift adding 2. So we go: 0, 1, 10, 11, 100, 101, 110, 111, etc.

So with just 8 on and off (binary) transistors, we could represent $2^8 = 256$ (recall combinatorics) unique combinations! That's enough to represent all

128 ASCII characters. So we just make a bunch of these little 8 transistor components instead!

And there's really not too many better ways to do it available right now. We store units of memory in little 8 bit (8 transistor) slots called bytes. Each byte has 256 possible combinations, which is sufficient for ASCII.

Bits Summary

It's easy to build an electronic component that stores just two states. Therefore, if we wish to store characters, we must represent them in a base-2 system. In a binary base-2 system, we need only 8 bits to allow for 256 possible combinations. Thus a single character can be represented by just 8 bits, or one byte. These can be strung together to form any string of relevant data.

Binary (8 bits)	**Decimal Base-10**	**ASCII Character**
01000001	65	A
01000010	66	B
01000011	67	C
01000100	68	D
01000101	69	E
01000110	70	F
01000111	71	G

Table 1: Example of 8-bit binary representing ASCII characters