How does a computer store letters?

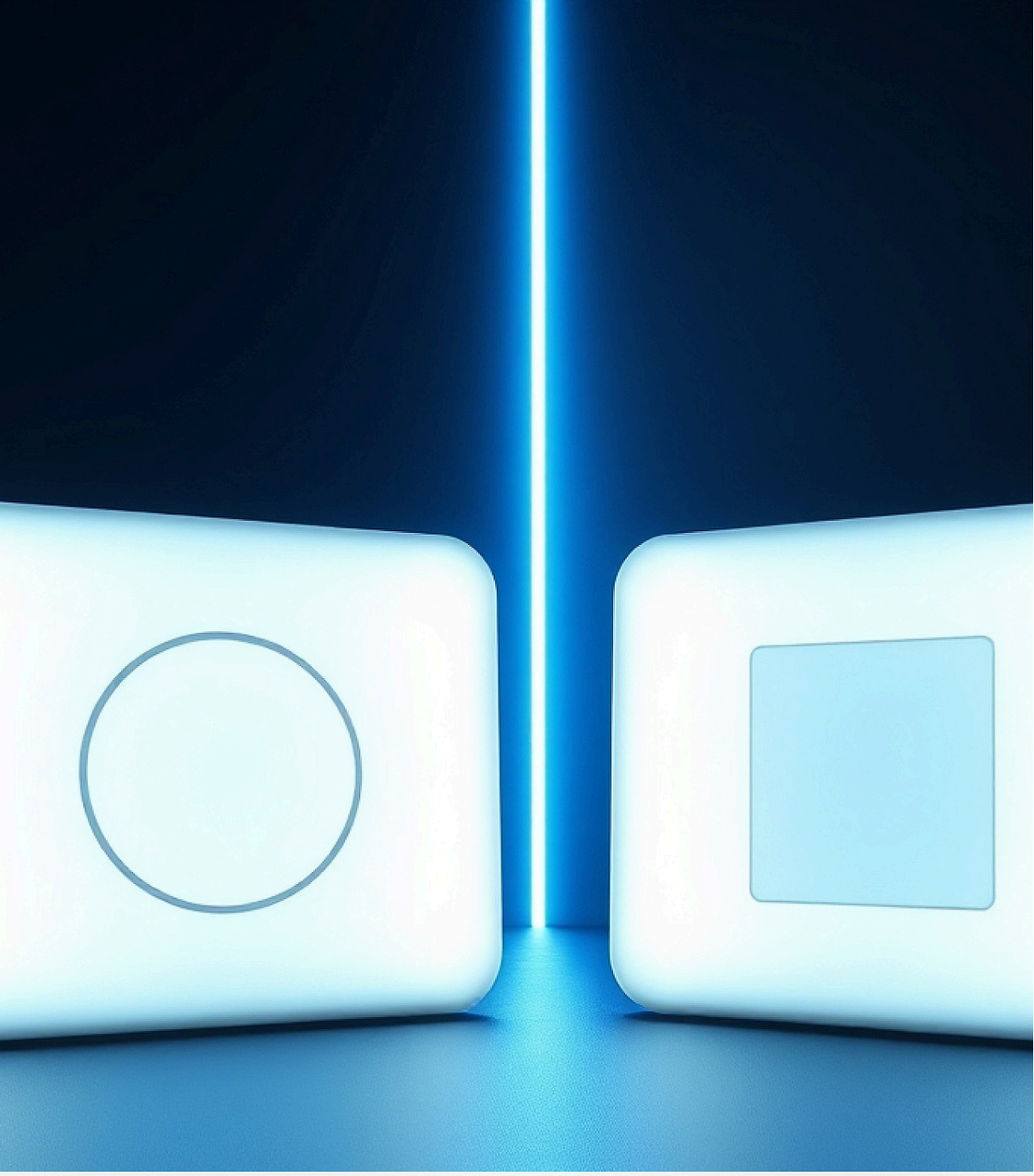


www.yzokul.com.tr

How does a computer store letters?



How does a computer store letters? The short answer: it turns each letter into a number according to a fixed rule, and then saves that number as bits. For example, the letter A becomes the number 65, and 65 is saved as a pattern of 0s and 1s. A letter's stored number is not the same as the shape you see on the screen.



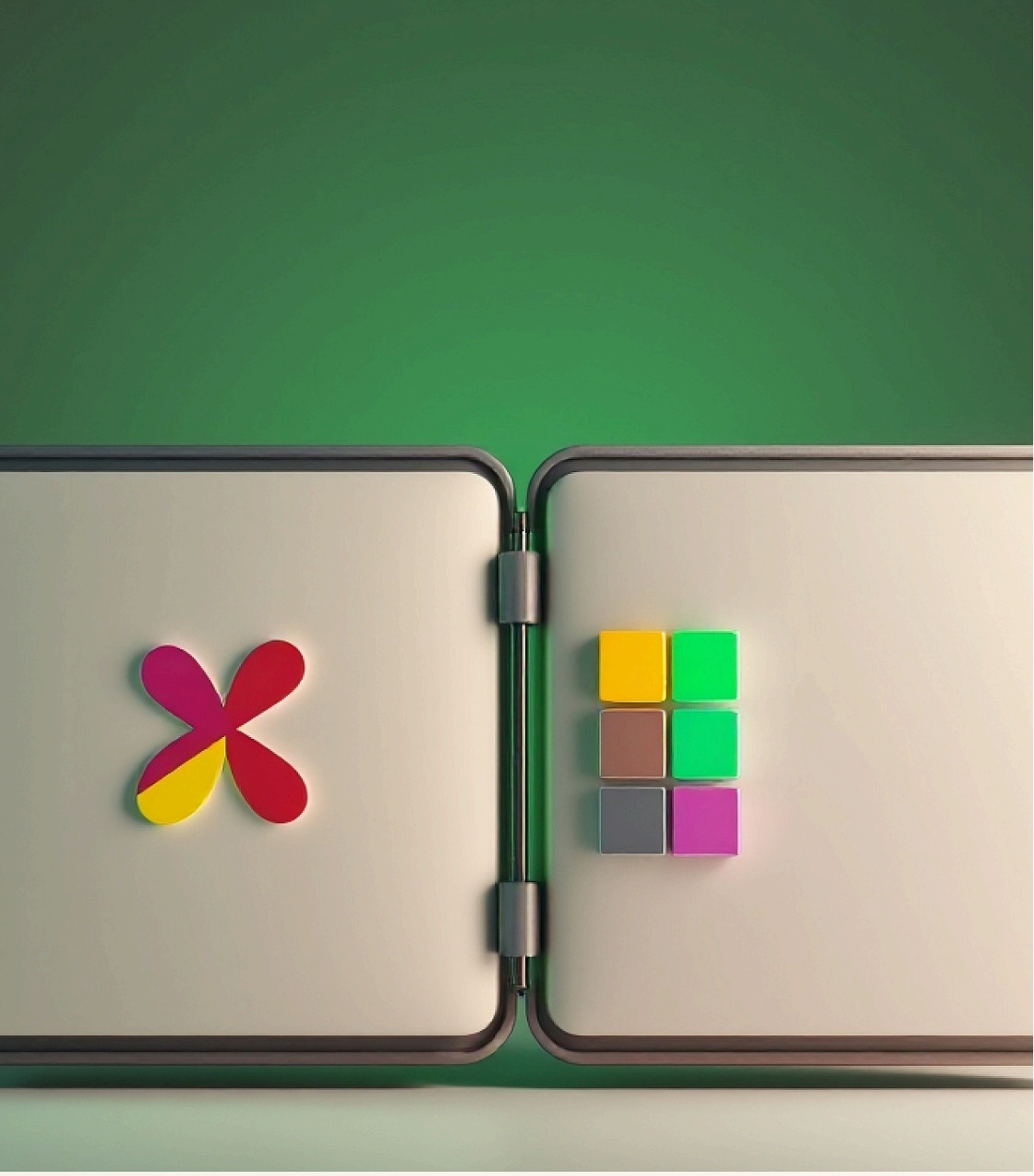
A computer's memory holds only two states, which we write as 0 and 1. A single 0 or 1 is called a bit, the smallest unit of information a computer can store. Everything a computer keeps must be turned into bits, because there is nothing else it can hold. Letters, photos, and sounds all become bits in the end.



Bits are grouped into sets of eight, and each set is called a byte. A byte can hold 256 different patterns of 0s and 1s, so it can stand for 256 different numbers, from 0 to 255. That is enough room to give every common letter, digit, and punctuation mark its own number. One byte often holds one character.



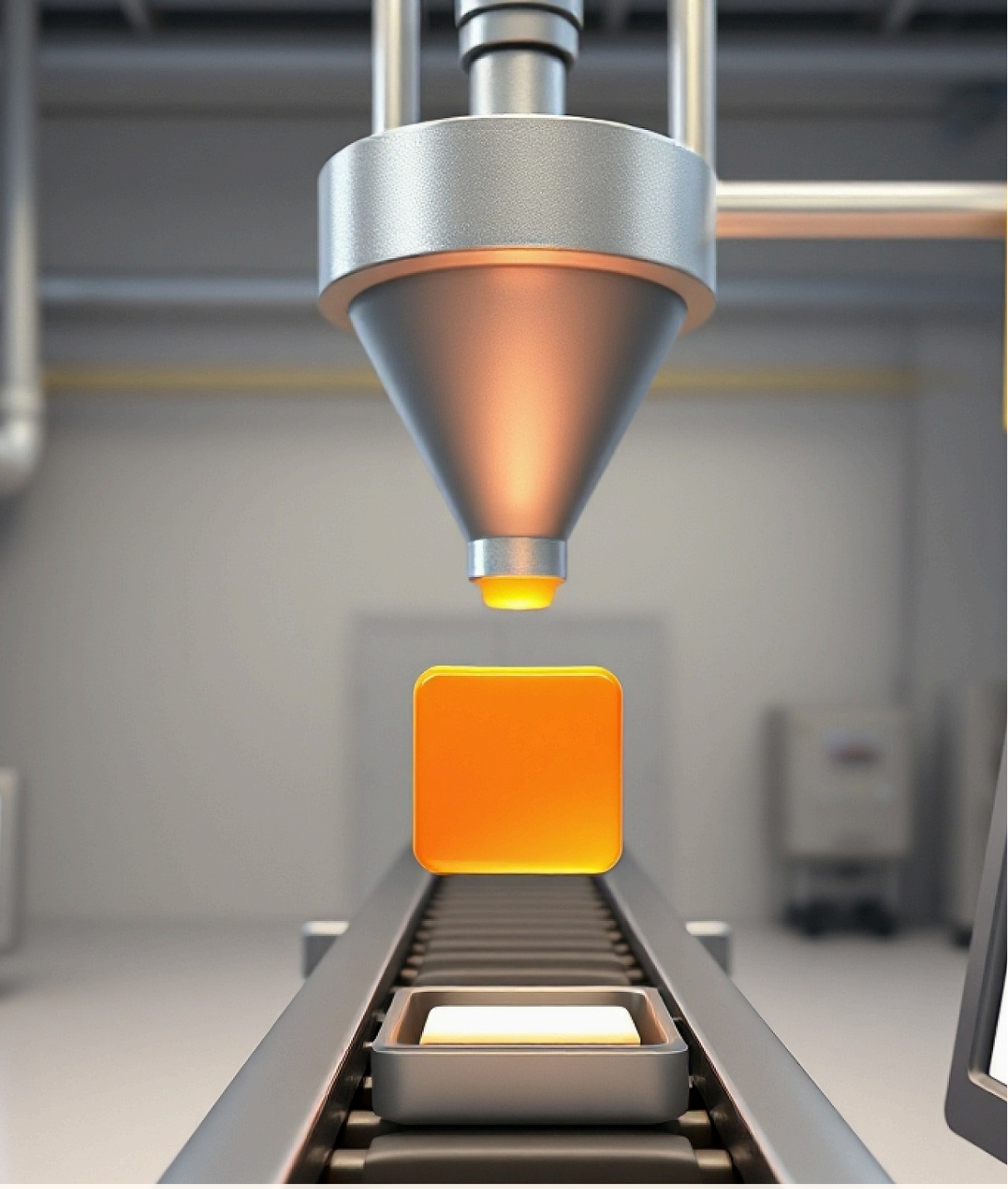
A letter shape like A cannot go into memory directly, because memory only holds bits. Something must turn the letter into a number first, and that number can then be written as bits. Without this conversion, the computer would have no way to keep the letter. The visible shape is made later, when the text is shown.



The rule that gives every letter and symbol its own number is called character encoding. Think of it as a codebook: a fixed list where each character points to one number. Both the program that writes the text and the program that reads it must use the same codebook. If they use different ones, the numbers will be misunderstood.



A simple real codebook is called ASCII. In ASCII, the letter A is the number 65, B is 66, C is 67, and so on through the alphabet. Each of these numbers is saved as one byte, a pattern of eight bits. For example, 65 is written in binary as 01000001. ASCII also gives numbers to digits and punctuation.



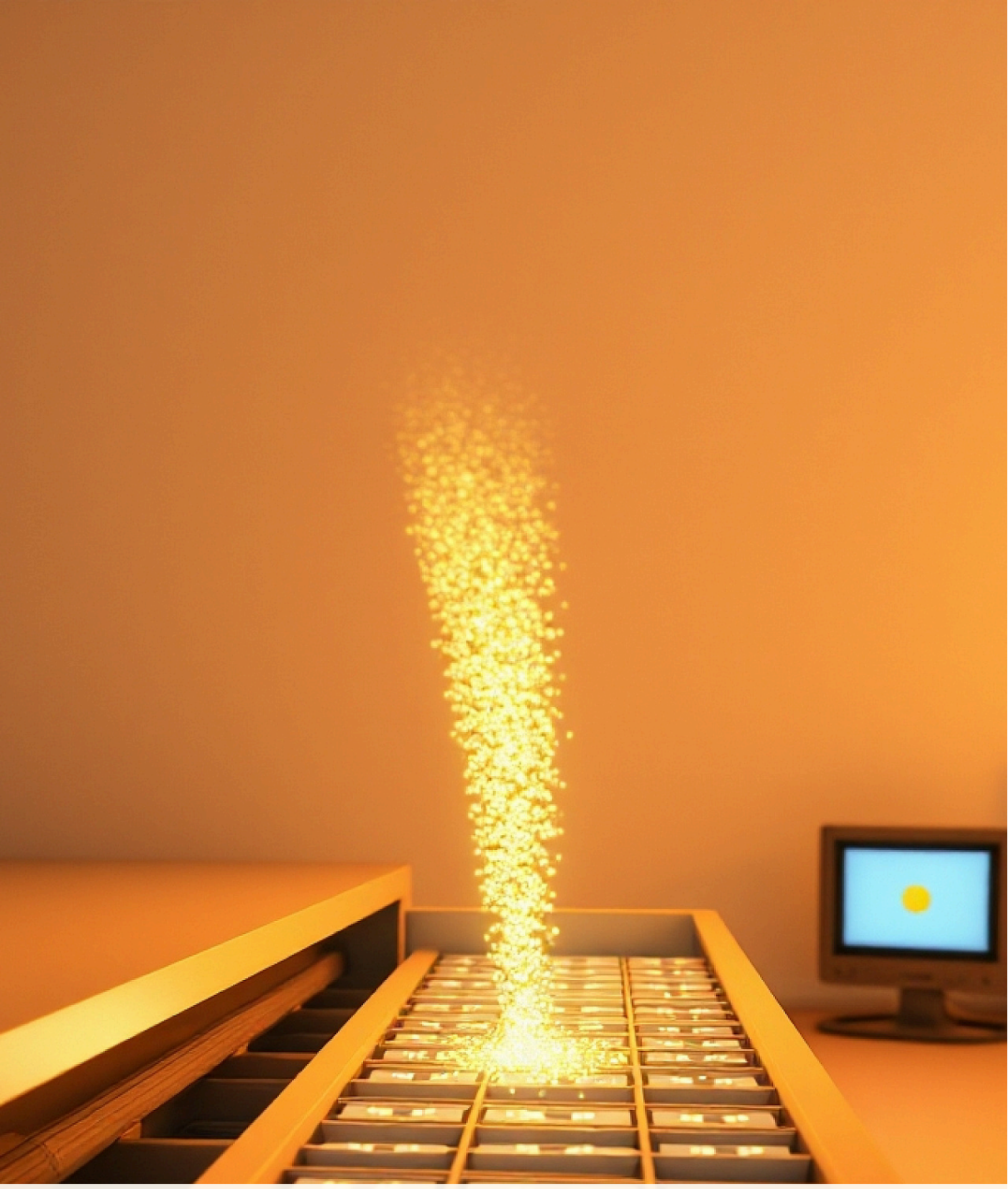
Follow one letter, A, as it is stored. The computer looks up A in the codebook and finds the number 65. Now the computer holds a number instead of a letter. This lookup happens for every character in the text, one after another. The letter itself is no longer needed once its number is known.



The number 65 is then written in binary as a pattern of 0s and 1s: 01000001. That bit pattern is what actually gets saved in the file. Binary is just a way of writing numbers using only the two digits 0 and 1. Every number the computer stores is written this way, because memory only holds bits.



The bit patterns for each letter are written one after another into a text file. If the text is the word CAT, the file holds the bits for C, then the bits for A, then the bits for T. On the disk, the file holds only these bits, nothing else. The disk does not store the shapes of the letters.



Opening the file means reading the bits again. The computer takes the bits in groups of eight, one byte at a time, and turns each byte back into a number. Now it has numbers again, not letters. The next job is to turn those numbers back into characters you can read.



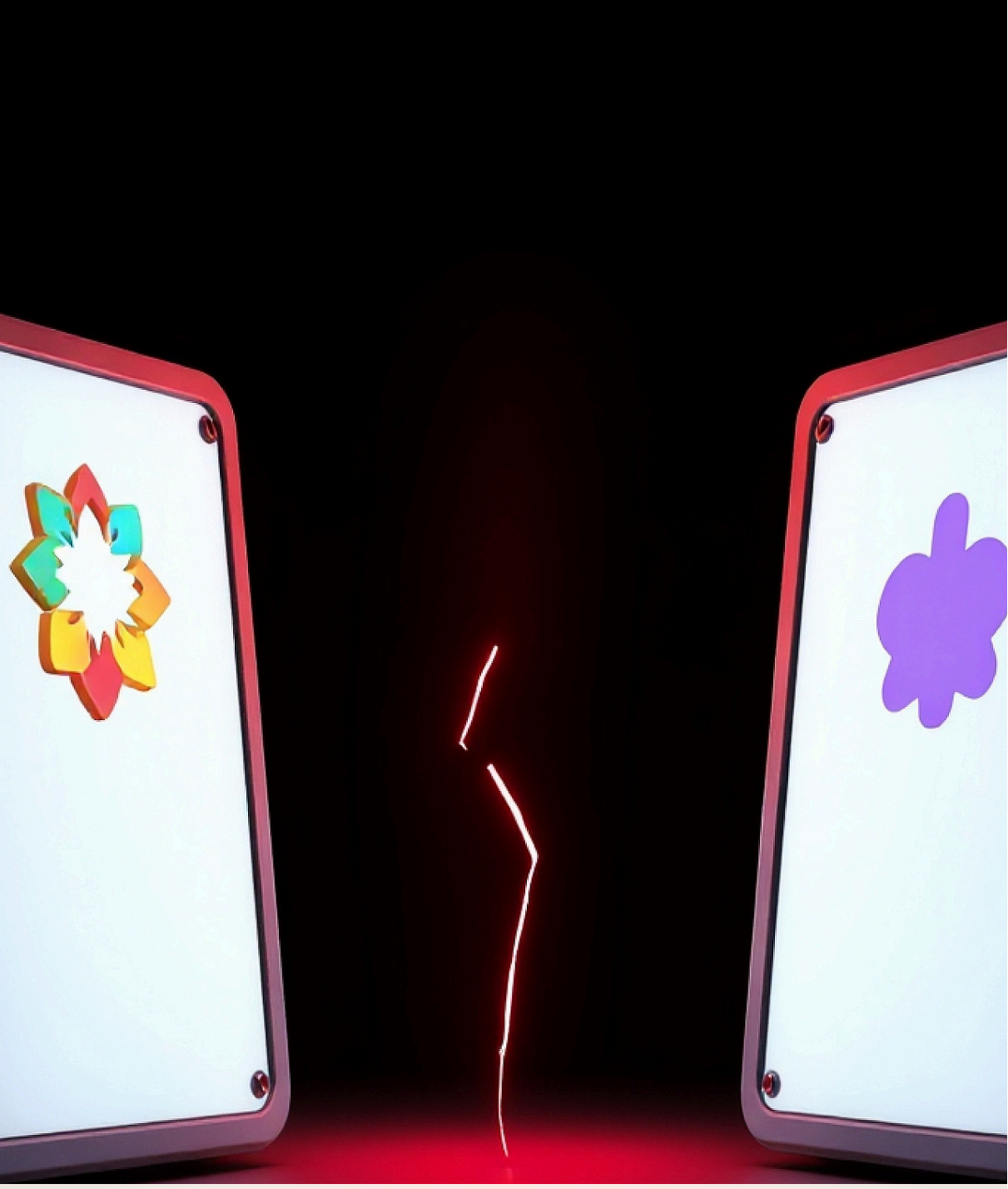
Turning numbers back into letters is called decoding. The computer looks up each number in the same codebook that was used to write the file. The number 65 is found in the codebook and becomes the letter A again. Decoding is simply the reverse of the encoding that happened when the text was saved.



The screen draws the visible shape of each letter using a font. A font is a set of drawings, one for each character, that the screen can paint. The stored number only tells the computer which shape to draw; it does not contain the shape itself. So the number 65 tells the screen to draw the shape for A.



The program writing the file and the program reading it must use the same encoding. When they agree, each number points to the intended letter, and the text appears correctly. This agreement is the whole reason encoding rules are written down and shared. Same codebook on both sides means no confusion.



If the reader uses a different codebook, the numbers point to wrong letters. A byte meant to be A might be read as a completely different symbol. The text then looks corrupted, full of strange or unexpected symbols. The bits are not damaged; they are simply being interpreted by the wrong rule.



A common mistake is to think the stored number for a letter is the picture you see. It is not. The number is only a reference, a label that says which character is meant. The shape is drawn separately by the screen using a font. So the code and the shape are two different things.



Letters become numbers by an encoding rule, numbers are saved as bits, and reading uses the same rule to turn bits back into letters. For example, A becomes 65, 65 becomes 01000001, and the file holds those bits. Remember that the code is not the shape: the screen draws the shape separately.