

CONCEPTO 1 DE 4

De la electricidad al código

Cómo una computadora convierte electricidad en todo lo que ves en pantalla

¿QUÉ HACE UNA COMPUTADORA?

Una computadora no "piensa". Mueve electricidad. Dentro del procesador hay miles de millones de componentes diminutos llamados **transistores**. Cada transistor funciona igual que un interruptor de luz: deja pasar la corriente, o no la deja pasar.

A ese estado le pusieron un valor: cuando hay corriente es **1**, cuando no hay es **0**. De ahí surge la unidad básica de toda la informática:

DEFINICIÓN

Bit — la unidad mínima de información. Puede tomar uno de dos valores posibles: **0** (sin corriente) o **1** (con corriente). Viene de *binary digit*.

Los procesadores modernos tienen más de 10.000 millones de transistores apagándose y prendiéndose miles de millones de veces por segundo. Cuando escuchás "procesador de 3 GHz", significa que puede hacer 3.000 millones de ciclos por segundo. Cada ciclo mueve bits.

DE BITS A TEXTO: EL BYTE Y EL ACUERDO UNICODE

Con un solo bit tenemos dos opciones (0 o 1). Con **8 bits juntos** — lo que llamamos un **byte** — tenemos $2^8 = 256$ **combinaciones** posibles. El desafío era acordar qué significa cada combinación.

💡 LA ANALOGÍA DEL CÓDIGO MORSE

Pensalo de esta manera: si entre nosotros definimos que dos aplausos significa "Hola", acabamos de crear un código basado en impulsos. Con las computadoras es exactamente lo mismo: el mundo se reunió a ponerse de acuerdo sobre qué iba a significar cada combinación de números. Así decidieron que ciertos números forman la letra H, la letra O, o un emoji.

El primer acuerdo se llamó **ASCII** (1963): usaba combinaciones de 8 bits y permitía 128 caracteres básicos — letras inglesas, números y símbolos. Funcionó bien hasta que el mundo quiso escribir en árabe, chino o usar emojis. 256 combinaciones no alcanzaban.

Entonces vino **Unicode**: expandió el acuerdo usando combinaciones de hasta 4 bytes, lo que permite albergar más de **150.000 caracteres** de todos los idiomas del mundo.

H O L A

01001000

01001111

01001100

01000001

Ese acuerdo no vive en un solo lugar — está implementado en el teclado, el sistema operativo, los archivos de tipografía y la pantalla. Todos contruidos siguiendo las mismas reglas. Por eso cuando escribís H en cualquier computadora del mundo, aparece H.

DE BYTES A TODO LO DEMÁS

Con bytes representamos letras. Pero también representamos **colores**. Una pantalla es una cuadrícula de millones de puntos microscópicos llamados **píxeles**. Cada píxel tiene tres pequeñas bombillas: Roja (R), Verde (G) y Azul (B). La computadora controla la intensidad de cada una usando bytes.

El rojo puro es $R=255, G=0, B=0$ — máxima corriente al rojo, ninguna al verde ni al azul. El blanco es $R=255, G=255, B=255$. El negro es $R=0, G=0, B=0$.

¿Cómo aparece una letra en pantalla? Cuando presionás "H" en el teclado, pasan cuatro cosas:



Con colores formamos píxeles. Con píxeles, imágenes. Con imágenes en secuencia, video. Con números también representamos sonido (muestras de amplitud de onda, miles por segundo).

Todo lo digital — absolutamente todo — son bytes. Los bytes son bits. Los bits son transistores. Los transistores son electricidad.

TAMAÑOS DE ARCHIVO: DE BYTES A TERABYTES

A medida que agrupamos bytes, los nombramos para manejarlos más fácil:

KB Kilobyte ~1.000 bytes Un mail de texto	MB Megabyte ~1.000 KB Una canción MP3	GB Gigabyte ~1.000 MB Una película HD	TB Terabyte ~1.000 GB Miles de películas
--	--	--	---

Cuando en AWS configurás el disco de una instancia EC2 o un bucket en S3, estos son los valores que vas a manejar.

LAS CAPAS DE TRADUCCIÓN

Escribir en ceros y unos sería imposible para humanos. Por eso se construyeron capas de traducción: cada una más legible para personas, pero por abajo siempre termina siendo bits que el procesador ejecuta.

Python / JS / Go	if usuario.pin == correcto: dejar_entrar()	VOS ESCRIBÍS ESTO
Bytecode	LOAD_FAST · COMPARE_OP · POP_JUMP	TRADUCCIÓN INTERMEDIA
Assembly	MOV EAX, [EBP-4] · CMP EAX, 1	CASI MÁQUINA
Machine code	10001011 01000101 11111100 ...	EL PROCESADOR EJECUTA

COMPILADO VS INTERPRETADO

Hay dos formas en que un lenguaje puede hacer esa traducción:

Compilado

El código se traduce **todo de una vez** antes de ejecutarse. El resultado es un archivo ejecutable que corre directamente en el procesador. Más rápido en ejecución.

Go, C, C++, Rust

Interpretado

El código se traduce y ejecuta **línea por línea** en el momento. Necesita que el intérprete esté instalado para correr. Más flexible y portátil.

Python, JavaScript, Ruby

PARA RECORDAR

Cuando instalás Go o Python, lo que instalás es el traductor. Sin él, el archivo `.go` o `.py` es solo texto — la computadora no sabe qué hacer con él. Cuando en AWS lanzás una instancia y querés correr tu app Python, lo primero que hacés es instalar Python en esa máquina.