



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO®

TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TLAXIACO

**REPORTE DE INVESTIGACION DE
INTERPRETACION DE 0 Y 1 A NIVEL DE HARDWARE**

Presenta:

22620061 García Santiago Sandy Marissa

Materia:

Arquitectura de computadoras

Carrera:

Ingeniería En Sistemas Computacionales

Docente:

Ing. Osorio Salinas Edward

Grupo:

5BS

Tlaxiaco, Oaxaca, A 14 de septiembre de 2024.

"Educación, Ciencia y Tecnología, Progreso día con día" ®



INDICE

Introducción	3
Cómo se implementa en hardware:.....	4
¿Cómo funciona el sistema binario?	5
Sistema binario en las computadoras	7
De binario al ensamblador y a los lenguajes de programación	11
El lenguaje ensamblador.....	12
Los lenguajes de programación	12
Todo es binario, al menos hasta la computación cuántica	13
Conclusión	15
Bibliografías:.....	15

Introducción

La interpretación de los números 0 y 1 a nivel de hardware es fundamental en el funcionamiento de los sistemas digitales y los dispositivos electrónicos modernos. En el ámbito de la informática, estos dos valores forman la base del ****sistema binario****, el lenguaje en el que las computadoras procesan y almacenan información. Sin embargo, para que los sistemas digitales puedan operar, es necesario que estos dígitos binarios se traduzcan en señales físicas que los componentes electrónicos puedan manipular.

En el nivel más básico, los ceros y unos representan dos estados distintos de un sistema, generalmente asociados con ausencia y presencia de voltaje o corriente eléctrica. Estas señales son interpretadas por transistores, las unidades más básicas de procesamiento en los circuitos integrados. Los transistores actúan como interruptores electrónicos, donde un estado de bajo voltaje se interpreta como un "0" y un estado de alto voltaje como un "1".

A través de esta simple representación de dos estados, es posible realizar operaciones lógicas complejas utilizando puertas lógicas y otros circuitos electrónicos. Estos componentes permiten que las computadoras y otros dispositivos electrónicos realicen cálculos, tomen decisiones y almacenen información en forma de bits. La capacidad de traducir ceros y unos en señales físicas tangibles es lo que ha hecho posible el desarrollo de la electrónica digital y, en última instancia, la revolución informática.

A nivel de hardware, los números 0 y 1 representan **bits** que son la base del sistema binario utilizado en la informática y la electrónica digital. Estos bits corresponden a dos estados físicos distintos que los dispositivos electrónicos pueden reconocer y procesar. Los estados de 0 y 1 se interpretan de la siguiente manera:

1. **0**: Generalmente representa un estado de **baja** energía, que en un circuito electrónico puede interpretarse como **ausencia de voltaje** o un nivel de voltaje bajo. Esto puede asociarse con apagado, inactividad o falta de corriente en un componente.
2. **1**: Representa un estado de **alta** energía, que en un circuito puede interpretarse como **presencia de voltaje** o un nivel de voltaje alto. En este caso, hay corriente o el componente está en un estado activo.

Cómo se implementa en hardware:

- **Transistores**: La interpretación de 0 y 1 ocurre a través de **transistores**, que son pequeños interruptores electrónicos que controlan el flujo de corriente. Los transistores pueden estar "abiertos" (permiten el paso de corriente, representando un 1) o "cerrados" (bloquean la corriente, representando un 0).
- **Puertas lógicas**: Los transistores forman **puertas lógicas** (AND, OR, NOT, etc.), que procesan los bits y realizan operaciones lógicas. Estas puertas toman varias entradas de bits y producen una salida también en forma de 0 o 1, dependiendo de la operación que realizan.
- **Memorias**: Las memorias como la **RAM** o las **memorias flash** almacenan bits. En estos dispositivos, el estado de 0 o 1 puede estar representado por la carga eléctrica almacenada en un pequeño condensador (en la RAM) o por una célula de memoria en estado de conducción o no (en la memoria flash).

Los números 0 y 1 a nivel de hardware corresponden a diferentes estados físicos de los componentes electrónicos, principalmente el voltaje o la corriente eléctrica, lo que permite que los circuitos digitales ejecuten operaciones lógicas y almacenen información.

¿Cómo funciona el sistema binario?

Para entender el sistema de numeración binario primero pensemos en el sistema decimal que usamos a diario. En él tenemos 10 dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9) y vamos contando en ese orden hasta que los agotamos (o sea, cuando llegamos al nueve y no hay más dígitos). Entonces nos toca combinarlos en números de dos dígitos, así que comenzamos en el siguiente de la lista (1) y volvemos a contar del 0 al 9 (10, 11, 12, 13, 14, 15, 16, 17, 18 y 19).

Cuando terminamos esta secuencia vamos al siguiente dígito de la lista y volvemos a comenzar: 20, 21, 22, 23... y cuando terminamos, vamos al siguiente: 30, 31, 32... Y cuando terminamos con todos, creamos números de 3 dígitos: 100. Y volvemos a comenzar.

En el sistema binario no tenemos 10 dígitos, solo 2: 0 y 1. Pero el proceso es el mismo. Veamos:

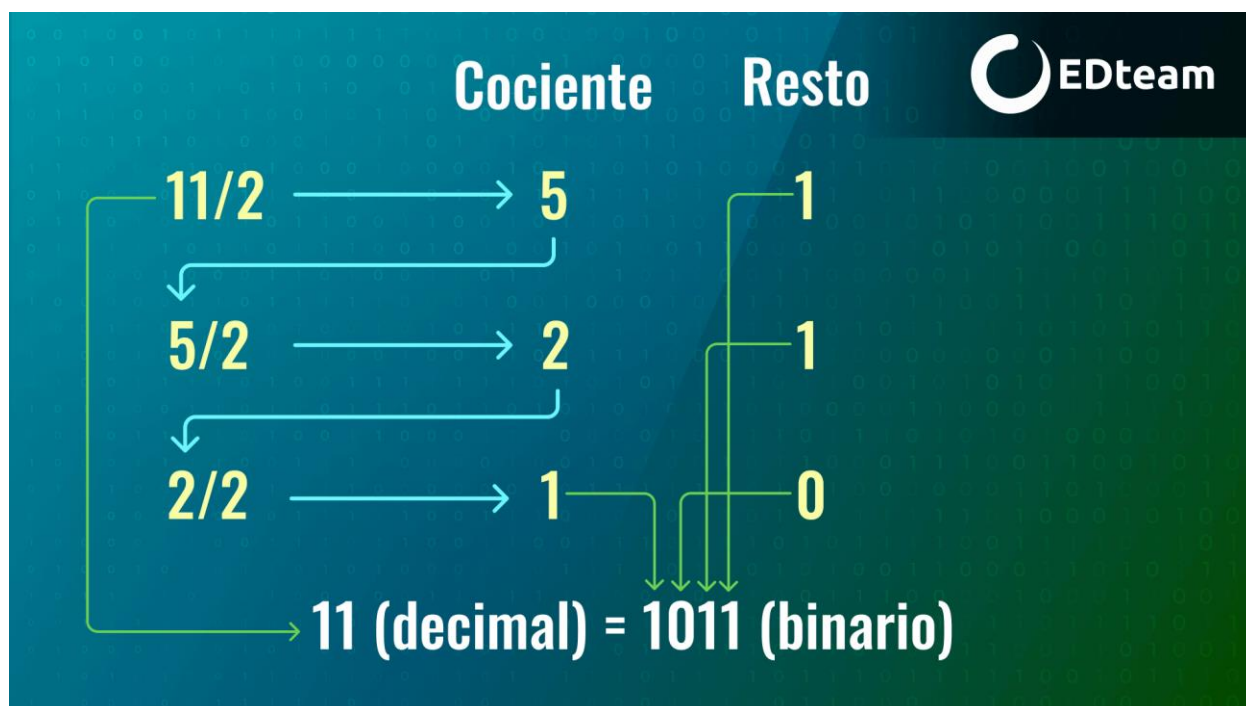
- Contamos 0 y 1.
- Si queremos ir al siguiente número (el que en decimal sería 2), nos toca crear un número de 2 dígitos: 10
- El tres es fácil porque podemos cambiar el 0 por 1 y nos queda 11.
- Para el cuatro nos quedamos sin dígitos, así que creamos un número de tres dígitos: 100.
- El cinco es fácil, solo sumamos uno: 101
- Para el seis sumamos uno. Pero ya sabes que sumar 1 da 10, así que 6 es 110
- El siete es sumar uno: 111
- Y para el 8 nos quedamos sin dígitos, así que toca crear un número de 4 dígitos: 1000



Si te fijas, aumentamos un dígito solo en las potencias de 2 (2, 4, 8, 16, 32, 64, 128, 256, 512, 1024) ¿Te suenan estos números?

Una forma muy sencilla de convertir cualquier número decimal a binario es hacer divisiones sucesivas y quedarnos con el resto y el cociente final. Por ejemplo, si queremos pasar 11 a binario:

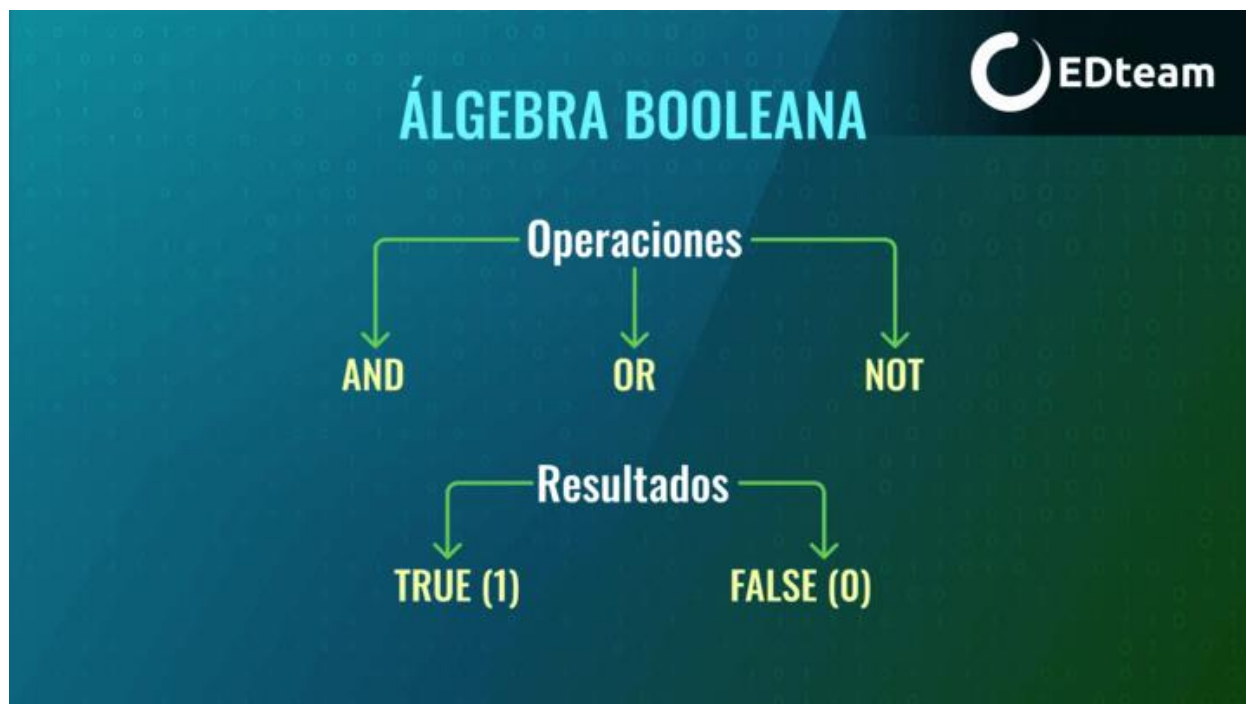
1. Primero dividimos 11 entre 2. Nos da 5 y el resto es 1. Conservamos el 1.
2. Dividimos 5 entre 2. Nos da 2 y el resto es 1. Conservamos el 1.
3. Dividimos 2 entre 2. Nos da 1 y el resto es 0. Como la división terminó conservamos el 0 y también el cociente 1.
4. Por lo tanto, 11 en binario entonces es 1011.



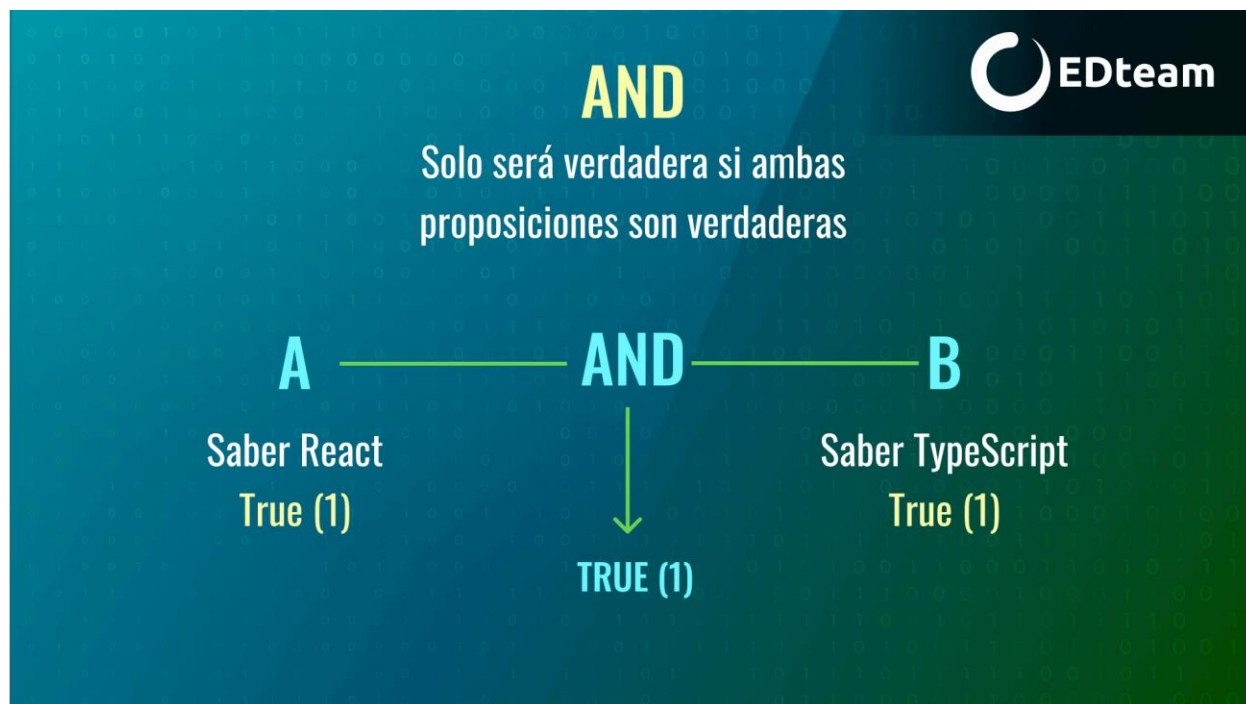
Sistema binario en las computadoras

En 1854, George Boole publicó "An Investigation of the Laws of Thought" (una investigación sobre las leyes del pensamiento) en la que desarrolla la famosa álgebra booleana que es una teoría fundamental para la informática. El álgebra booleana consiste de tres operaciones lógicas bastante sencillas: AND, OR y NOT.

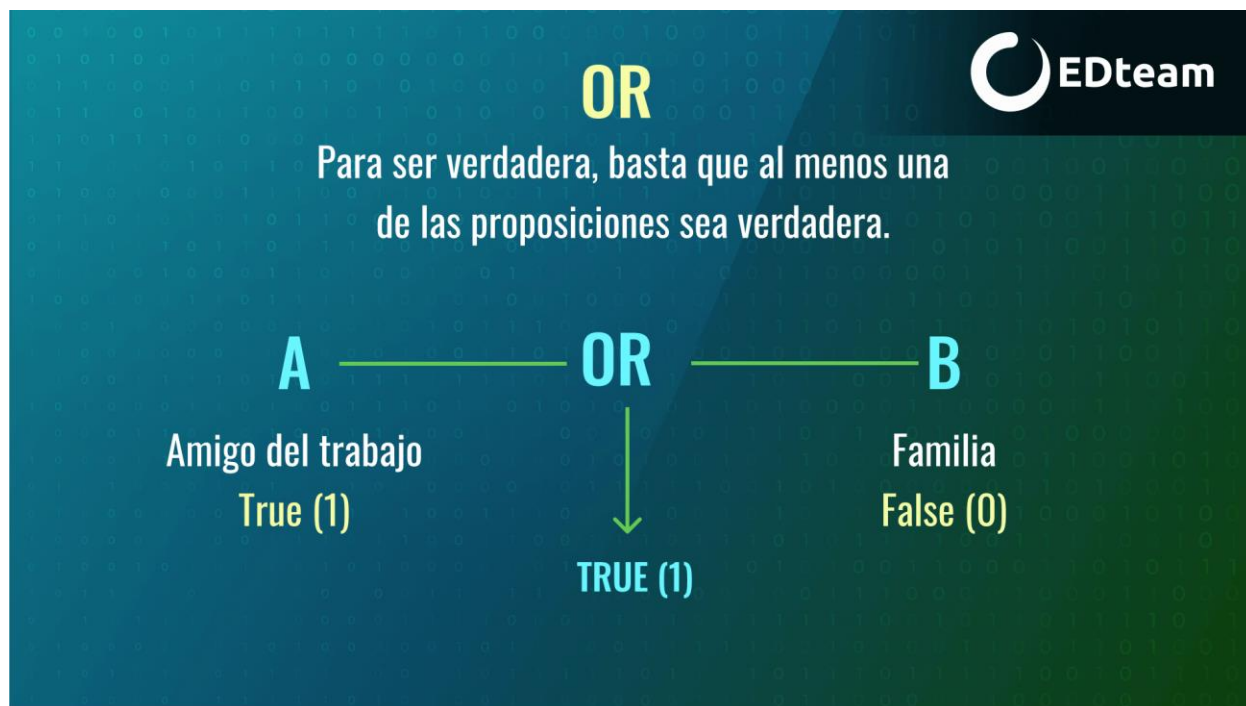
En este sistema, los resultados de las operaciones solo tienen dos valores posibles: verdadero (true o 1) y falso (false o 0). Por eso, en honor a George Boole, le llamamos booleanos (o boolean) a los tipos de datos verdaderos y falsos.



La operación AND será verdadera (1) solo cuando ambas proposiciones son verdaderas (1 y 1). Por ejemplo, si en EDteam estamos buscando un programador frontend que sepa React y TypeScript es una operación AND. Porque no le bastará saber una de las dos para pasar. Tiene que, sí o sí, saber ambas.



La operación OR, en cambio, solo requiere que una de las dos proposiciones sea verdadera para ser verdadera. Por ejemplo, si estás organizando una fiesta, puedes invitar a amigos del trabajo o a tu familia. No es necesario que sean a la vez del trabajo y de tu familia.

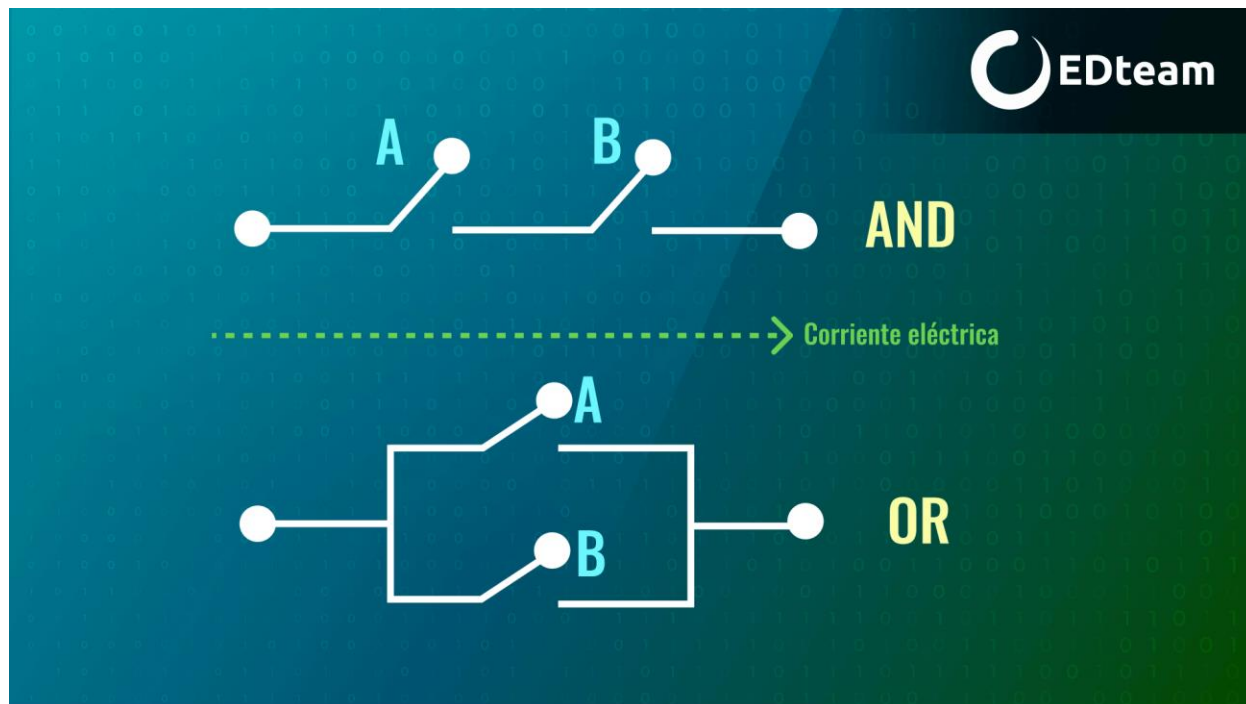


Mientras que la operación NOT implica negar la proposición. Por ejemplo, podrías decir que NO se admiten niños en tu fiesta.

Estas tres operaciones son la base de cómo funciona nuestro pensamiento cuando analizamos una situación y tomamos decisiones. Sin embargo, se convirtió (sin que su autor se lo proponga) en la base para la informática y las computadoras.

En 1937, Claude Shannon publicó el artículo “Análisis simbólico de circuitos de relés y de conmutación”. Una obra que ha sido considerada la base de la informática moderna, pues unía el álgebra booleana con los circuitos. Mientras trabajaba en los laboratorios Bell quedó maravillado de como se usaban los relés y los interruptores para manejar el tráfico de las llamadas telefónicas y redireccionarlas. Y se le ocurrió que usando ese principio y el álgebra de Boole se podría ejecutar cualquier operación matemática usando interruptores, que no son más que ceros (apagado) y unos (encendidos).

Por ejemplo, una operación AND consistiría de dos interruptores en serie. Tienen que estar los dos encendidos (1) para que pase la corriente. Mientras que un OR serían dos interruptores en paralelo. Basta que uno de los dos esté prendido (1) para que pase la corriente.



Este principio tan sencillo, replicado cientos o miles de veces en muchos interruptores, podía resolver cualquier problema matemático o lógico y sería la base para las computadoras.

De este trabajo derivan las famosas puertas lógicas, que son una forma de implementar en un circuito electrónico el álgebra booleana usando los principios de Claude Shannon.

Casi al mismo tiempo, en 1936, Alan Turing en su artículo “Acerca de números computables” presenta el primer modelo teórico de una computadora, con entrada, salida y un CRUD completo y también basada en un sistema binario.

Y aun así las primeras computadoras fueron decimales y no binarias, el sistema binario es más fácil de implementar en un circuito electrónico, mientras que el decimal es más fácil en un sistema electromecánico. Por ejemplo, dos de las primeras computadoras de la historia: el Harvard Mark I y el Mark II fueron decimales.

Como anécdota, en el Mark II trabajaba el equipo de Grace Hooper y un día la máquina dejó de funcionar y buscando el problema encontraron una polilla en uno de los interruptores. A partir de entonces se popularizó el término “bug” (insecto) para referirse a los errores de programación.

En 1943 se creó la primera computadora de la historia: Colossus. Esta computadora era binaria, electrónica y tomaba elementos de la máquina de Turing. Sin embargo, estuvo aplicada solo a descifrar códigos Nazi y se mantuvo en total secreto. Y cuando terminó la guerra fueron destruidas todas las máquinas Colossus.

En 1946 se lanzó en Estados Unidos el ENIAC, que fue considerada la primera computadora de la historia (porque nadie sabía de la existencia de Colossus) a pesar de que era decimal y no binaria. John Von Neumann había trabajado como asesor en la construcción del ENIAC y se dio cuenta de que los cálculos podían ser más rápidos usando el sistema binario, pero no tenía el poder para cambiar el diseño del ENIAC. Así que su siguiente proyecto fue una computadora binaria, luego lideró un nuevo proyecto, fue una computadora binaria y de programa almacenado, es decir, que tenga memoria.

Esta computadora se llamó EDVAC y se lanzó en 1951, y a pesar de que se habían construido varias computadoras antes, el EDVAC es la primera en tener todos los componentes de una computadora moderna: es electrónica, binaria y de programa almacenado (es decir, tiene una memoria para almacenar los programas en lugar de tener que reconfigurarla físicamente moviendo cables e interruptores).

Desde entonces todas las computadoras usaron el sistema binario porque era más eficiente en circuitos electrónicos en los que se podía controlar el voltaje para diferenciar ceros de unos. Primero se hizo con tubos de vacío y luego se pasó al estado sólido con los transistores y los chips. Si quieres saber más, no te pierdas como se transformó la arena en empresas multimillonarias.

De binario al ensamblador y a los lenguajes de programación

El binario es excelente para las computadoras porque pueden hacer cálculos a velocidades increíbles, pues solo emplean dos valores (1 y 0) en lugar de cientos o miles. Pero, para los programadores es una tortura escribir en binario. Imagínate, tenías una

tarjeta en la que podías hacer agujeros que representaban un 1, y la falta de agujero un 0. Si eras programador, escribías tu código en varias tarjetas y luego hacías fila con tus tarjetas para que un operador las introduzca en la computadora.

Y ay de ti si encontraban un error porque te tocaba empezar de nuevo revisando tarjeta por tarjeta (no había un debugger que te diga en qué lugar estaba el error). No era cool ser programador en esos tiempos, era un trabajo más de paciencia que de talento.

El lenguaje ensamblador

Este proceso tan lento se conocía como “procesamiento por lotes” y aunque no podía cambiarse por la falta de potencia de las computadoras que no podían procesar varias órdenes a la vez sino una detrás de otra; lo que sí se podía era dejar de torturarnos escribiendo código binario.

Así que el siguiente paso fue crear un lenguaje que fuera más fácil de entender y de recordar y así nació el lenguaje ensamblador. Este lenguaje es un conjunto de órdenes directas al procesador (como hacer cálculos matemáticos) escritas en abreviaturas que hacen mucho más fácil programar.

Este lenguaje consta de dos partes: el lenguaje ensamblador en sí mismo, y el ensamblador, que es un programa que se encarga de convertir las instrucciones de este lenguaje en binario. Porque las computadoras no entienden ensamblador, solo entienden ceros y unos.

Los lenguajes de programación

Y aunque el lenguaje ensamblador es una forma más sencilla frente a programar en binario, seguía siendo un trabajo arduo y lento, por lo que se crearon los lenguajes de programación de alto nivel. Alto nivel significa que son más fáciles de comprender para los seres humanos. Mientras que bajo nivel significa que son menos fáciles de entender, pero con más control sobre cada instrucción dada al procesador.

Pero, como ya te habrás dado cuenta, estos lenguajes deben ser convertidos a ceros y unos porque las computadoras no entienden ni Python ni JavaScript ni Java ni Go, ni ninguno de estos lenguajes. Solo entienden ceros y unos. Y para convertir los lenguajes de programación a binario se creó un programa llamado “compilador”.

El primer compilador fue creado por Grace Hooper en 1952 para un lenguaje llamado Flow-Matic que no llegó a ser utilizado ampliamente. El primer lenguaje de alto nivel de la historia, utilizado en muchas computadoras y empresas, fue Fortran, lanzado en 1957. Y el segundo lenguaje fue COBOL, lanzado en 1959 y que hasta la fecha se sigue usando muchísimo, sobre todo en el mundo financiero.

Todo es binario, al menos hasta la computación cuántica

Aunque en la década del 60 se empezó a usar el código binario por su eficiencia, el poder de cómputo actual es tan grande que bien podría usarse el sistema decimal sin problemas. Pero, implicaría cambiar todo lo que ya existe: desde la arquitectura de procesadores, el lenguaje ensamblador, los compiladores, programas, sistemas operativos, drivers, todo. Es un trabajo que no vale la pena, así que seguiremos usando sistema binario en la computación.

Excepto en la computación cuántica, que usa qubits en lugar de bits y es enormemente más poderosa que la computación binaria. Aunque es tema de otro blog, **así que deja tu comentario si quieres que hablemos de la computación cuántica.**

Y, para terminar, ya que hablamos de bits, ¿qué son los bits? Son la abreviación de **B**inary **D**igit y no es más que una unidad de información, es decir, un cero o un uno. Una anécdota curiosa es que el término que se usaba antes era **bd** por las iniciales de Binary Digit. Sin embargo, fue John Tukey en 1946 quien creó el término bit, que luego fue popularizado por Claude Shannon.

Y ya que los bits son la unidad básica de la información, se usan para calcular las unidades de almacenamiento, siento la más básica, el byte que equivale a 8 bits. Y como cada bit tiene dos valores posibles (0 y 1), un byte tiene 256 valores posibles (2^8). Los famosos 256 valores que están en todas partes en la informática, desde las direcciones IP, la notación de colores RGB, o la representación del alfabeto en ASCII.

A partir de los bytes se definen las siguientes unidades de almacenamiento de la siguiente manera:

- 1 Kilobyte (KB) = 1024 bytes (2^{10} bytes)

- 1 Megabyte (MB) = 1024 kilobytes (2^{20} bytes)
- 1 Gigabyte (GB) = 1024 megabytes (2^{30} bytes)
- 1 Terabyte (TB) = 1024 gigabytes (2^{40} bytes)
- 1 Petabyte (PB) = 1024 terabytes (2^{50} bytes)
- 1 Exabyte (EB) = 1024 exabytes (2^{60} bytes)

Y todo, absolutamente todo lo que ves en la computadora, debe traducirse a ceros y unos (o bits). Por ejemplo, en una imagen, cada pixel es traducido a bits que son ceros y unos. Lo mismo para los videos, audios, PDF, código de programación, etc.

Conclusión

A nivel de hardware, los números 0 y 1 forman la base del sistema binario, permitiendo que los dispositivos electrónicos interpreten y procesen información mediante dos estados físicos distintos. El valor 0 representa la ausencia de voltaje o corriente, mientras que el 1 indica su presencia. Estos bits se implementan en circuitos electrónicos a través de transistores, puertas lógicas y memorias, lo que permite realizar cálculos, ejecutar instrucciones y almacenar datos de manera eficiente.

Este sistema ha sido esencial en la evolución de la informática, desde los primeros transistores hasta las complejas arquitecturas de los ordenadores modernos. Gracias a la simplicidad del sistema binario, las computadoras pueden realizar operaciones lógicas y matemáticas a gran velocidad, lo que ha facilitado el desarrollo de lenguajes de programación y sistemas avanzados. La traducción de los números binarios a señales físicas es el núcleo de la operación de los sistemas electrónicos, lo que ha permitido el progreso continuo de la tecnología informática.

Bibliografías:

¿Por qué las computadoras solo entienden 0 y 1? (Código binario). (s. f.). EDteam - En Español Nadie Te Explica Mejor. <https://ed.team/blog/por-que-las-computadoras-solo-entienden-0-y-1-codigo-binario>