Fundamentos de la Representación de la Información en Sistemas Informáticos

Introducción: Del Concepto a los Bits: La Jerarquía de la Abstracción Digital

Todo elemento de información que un sistema informático procesa, desde el texto de un correo electrónico y una hoja de cálculo financiera hasta una fotografía de alta resolución o un videojuego tridimensional, se reduce, en su nivel más fundamental, a una secuencia de ceros y unos. El propósito de estos apuntes es desvelar las capas de abstracción que permiten construir mundos digitales de inmensa complejidad a partir de esta simple dualidad. Este viaje nos llevará desde el concepto más básico de la información digital hasta la representación de contenido multimedia complejo, demostrando cómo cada nivel se construye sobre el anterior en una elegante jerarquía.

El átomo de este universo digital es el **bit**, una contracción del término inglés *binary digit* (dígito binario). Un bit es la unidad de información más pequeña y solo puede representar dos estados mutuamente excluyentes: encendido o apagado, verdadero o falso, alto o bajo voltaje y, de forma abstracta, 1 o 0.[1] La verdadera potencia de la computación no reside en el bit individual, sino en la capacidad de agrupar estas unidades en secuencias de longitud fija (como los
bytes, grupos de 8 bits) e interpretar estas secuencias mediante un conjunto de reglas y estándares bien definidos.

Esta interpretación es la clave de la abstracción en informática. Los temas que se abordarán no son conceptos aislados, sino que forman una clara jerarquía:

1. Los **bits** se agrupan para formar **números** mediante sistemas de numeración.
2. Ciertos **números** se asignan a caracteres específicos para representar **texto**.
3. Rejillas de **números**, donde cada número representa un color, forman **imágenes**.
4. Secuencias de **imágenes** (vídeo) y secuencias de **números** (sonido) dan lugar a la **información multimedia**.

Comprender esta "pila" de abstracciones es fundamental para entender no solo cómo funcionan los ordenadores, sino también para diagnosticar problemas y diseñar soluciones de software eficientes. Este documento guiará al estudiante a través de cada una de estas capas, construyendo un modelo mental coherente y robusto de la representación de la información.

Capítulo 1: Sistemas de Numeración: El Lenguaje de la Computadora

1.1 El Concepto de Base y el Sistema Posicional

Un sistema de numeración es un conjunto de símbolos y reglas que se utilizan para representar datos numéricos.[2] A lo largo de la historia, han existido diversos sistemas, pero los que dominan la computación y las matemáticas modernas se basan en dos conceptos cruciales: la
base y el **valor posicional**.
La **base** de un sistema de numeración es el número de dígitos o símbolos únicos que utiliza para representar cualquier cantidad.[4] Por ejemplo, el sistema que usamos en la vida cotidiana es de base 10, ya que emplea diez símbolos distintos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).
El **sistema posicional** es aquel en el que el valor de un dígito depende tanto de su propio símbolo como de la posición que ocupa dentro del número.[7] Cada posición representa una potencia de la base, incrementándose de derecha a izquierda. Por ejemplo, en el número decimal 372, el '2' está en la posición de las unidades (
100), el '7' en la de las decenas (101) y el '3' en la de las centenas (102). El valor total es la suma ponderada: (3×102)+(7×101)+(2×100). La invención del cero fue un hito crucial para el desarrollo de los sistemas posicionales, ya que permite indicar la ausencia de unidades en un orden determinado.[7] Esto contrasta con sistemas antiguos, como el egipcio, que eran aditivos y requerían símbolos distintos para cada orden de magnitud, volviéndose ineficientes para representar números grandes.[9]

1.2 Sistemas Fundamentales: Decimal, Binario, Octal y Hexadecimal

En el ámbito de la informática, cuatro sistemas de numeración son de vital importancia:

- **Sistema Decimal (Base 10):** Es el sistema universalmente utilizado por los seres humanos. Utiliza los dígitos del 0 al 9. Sirve como nuestro punto de referencia para comprender los demás sistemas.[2]
- **Sistema Binario (Base 2):** Es el lenguaje nativo de los ordenadores. Utiliza únicamente dos dígitos: 0 y 1. Su importancia radica en que estos dos estados se corresponden directamente con los dos estados de un circuito electrónico (apagado/encendido o bajo/alto voltaje), lo que simplifica enormemente el diseño del hardware.[1] Sin embargo, los números binarios largos son difíciles de leer y recordar para los humanos. Por ejemplo, el número decimal 250 se escribe en binario como
  11111010.

- **Sistema Octal (Base 8):** Utiliza los dígitos del 0 al 7. Históricamente fue popular en la computación temprana, y aunque su uso ha disminuido, sigue siendo un buen ejemplo de sistema de numeración compacto.[3]
- **Sistema Hexadecimal (Base 16):** Utiliza los dígitos del 0 al 9 y las letras de la A a la F para representar los valores del 10 al 15, respectivamente.[1] Es extremadamente común en programación y computación de bajo nivel, a menudo denotado con el prefijo 0x (por ejemplo, 0xFF es 255 en decimal).[14]

La verdadera razón de ser de los sistemas octal y hexadecimal no es simplemente ofrecer alternativas al binario, sino actuar como un **puente pragmático entre el lenguaje de la máquina y la legibilidad humana**. Los ordenadores operan exclusivamente en binario, pero para un programador, leer o escribir una dirección de memoria como 1111101011001110 es propenso a errores e ineficiente. La conversión entre binario y decimal requiere cálculos (sumas de potencias o divisiones sucesivas) que no son inmediatos.

Aquí es donde la relación matemática entre las bases se vuelve crucial: 8=23 y 16=24. Esta relación de potencia de dos permite una conversión casi instantánea. Para pasar de binario a octal, simplemente se agrupan los bits de tres en tres. Para pasar a hexadecimal, se agrupan de cuatro en cuatro.[2] Por ejemplo, la cadena binaria 1111101011001110 se puede agrupar de cuatro en cuatro como 1111 1010 1100 1110, lo que se traduce directamente a F A C E en hexadecimal. De este modo, el sistema hexadecimal no es solo "otro sistema", sino una interfaz de usuario optimizada para que los humanos lean y escriban valores binarios de forma compacta y con menos errores, siendo fundamental en áreas como el desarrollo de drivers, la depuración de memoria o la definición de colores en diseño web (CSS).[14]

1.3 Conversión de Binario a Decimal

Existen dos métodos principales para convertir un número del sistema binario al sistema decimal.

Método Formal: Suma de Potencias Ponderadas

Este método se basa en la definición de un sistema posicional. Cada dígito (bit) del número binario se multiplica por la base (2) elevada a la potencia de su posición. Las posiciones se numeran de derecha a izquierda, comenzando desde 0.[16] El valor decimal es la suma de todos estos productos.

La fórmula general es:

$$N_{10} = \sum_{i=0}^{n-1} d_i \times 2^i$$

donde d_i es el dígito en la posición i y n es el número total de dígitos.

Ejemplo Guiado: Convertir 101102 a decimal.

1. **Identificar posiciones:** El número tiene 5 dígitos. Las posiciones, de derecha a

izquierda, son 0, 1, 2, 3 y 4.
    - d0=0, d1=1, d2=1, d3=0, d4=1
2. Aplicar la fórmula:
   $N_{10}=(1\times2^4)+(0\times2^3)+(1\times2^2)+(1\times2^1)+(0\times2^0)$
3. Calcular potencias:
   $N_{10}=(1\times16)+(0\times8)+(1\times4)+(1\times2)+(0\times1)$
4. Sumar los productos:
   $N_{10}=16+0+4+2+0=22$
5. **Resultado:** $101102=22_{10}$

Método Práctico: La Tabla de Potencias de Dos

Este es un método visual y rápido que produce el mismo resultado. Es especialmente útil para cálculos mentales o rápidos sobre papel.

1. Se escribe una tabla con las potencias de dos, de derecha a izquierda:..., 128, 64, 32, 16, 8, 4, 2, 1.
2. Se coloca el número binario debajo de la tabla, alineando los dígitos.
3. Se suman únicamente los valores de la tabla que tienen un '1' debajo de ellos.[20]

Ejemplo Guiado: Convertir 1101012 a decimal.

1. **Escribir la tabla de potencias:**

64	32	16	8	4	2	1

2. **Alinear el número binario:**

64	32	16	8	4	2	1
	1	1	0	1	0	1

3. Sumar las potencias correspondientes a los '1':
   $32+16+4+1=53$
4. **Resultado:** $1101012=53_{10}$

1.4 Conversión de Decimal a Binario

De manera análoga, existen dos métodos principales para la conversión inversa.

Método Formal: Divisiones Sucesivas

Este es el algoritmo formal y consiste en dividir el número decimal repetidamente por 2 hasta que el cociente sea 0. Los residuos de cada división (que solo pueden ser 0 o 1) forman el número binario. Es crucial leer estos residuos en orden inverso, desde el último obtenido hasta el primero.[16]

Ejemplo Guiado: Convertir 4310 a binario.

1. 43÷2=21 Residuo: **1** (Este será el bit menos significativo o LSB, *Least Significant Bit*)
2. 21÷2=10 Residuo: **1**
3. 10÷2=5 Residuo: **0**
4. 5÷2=2 Residuo: **1**
5. 2÷2=1 Residuo: **0**
6. 1÷2=0 Residuo: **1** (Este será el bit más significativo o MSB, *Most Significant Bit*)

El proceso termina cuando el cociente es 0. Ahora, se leen los residuos de abajo hacia arriba.
Resultado: 4310=1010112

Método Práctico: Restas Sucesivas con la Tabla

Este método es el inverso del método práctico anterior y es muy intuitivo.

1. Se escribe la tabla de potencias de dos.
2. Se busca la mayor potencia de dos que sea menor o igual al número decimal. Se coloca un '1' en esa posición de la tabla.
3. Se resta el valor de esa potencia al número decimal para obtener un nuevo resto.
4. Se repite el proceso con el resto, buscando la mayor potencia de dos que sea menor o igual a él, hasta que el resto sea 0.
5. Las posiciones de la tabla que no se usaron se rellenan con '0'.[22]

Ejemplo Guiado: Convertir 8910 a binario.

1. **Tabla de potencias:** | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
2. El número es 89. La mayor potencia de dos menor o igual a 89 es 64.
    - Poner un **1** en la posición del 64.
    - Resto: 89−64=25.
    - Resultado parcial: 1.....
3. El resto es 25. La mayor potencia de dos menor o igual a 25 es 16.
    - Poner un **1** en la posición del 16.
    - Resto: 25−16=9.
    - Resultado parcial: 1.1....
4. El resto es 9. La mayor potencia de dos menor o igual a 9 es 8.
    - Poner un **1** en la posición del 8.
    - Resto: 9−8=1.
    - Resultado parcial: 1.1.1..
5. El resto es 1. La mayor potencia de dos menor o igual a 1 es 1.
    - Poner un **1** en la posición del 1.
    - Resto: 1−1=0.
    - Resultado parcial: 1.1.1...1
6. Rellenar los huecos con 0. Las posiciones de 128, 32, 4 y 2 no se usaron.
   Resultado: 8910=010110012 (o 10110012 sin el cero a la izquierda).

1.5 Conversiones Rápidas: Binario ↔ Octal ↔ Hexadecimal

Como se mencionó anteriormente, la conversión entre estos sistemas es directa gracias a su relación de potencias de dos.

- **De Binario a Octal y Hexadecimal:**
    - **Octal:** Se agrupan los bits de 3 en 3, comenzando desde la derecha. Si el último grupo de la izquierda no tiene 3 bits, se completa con ceros a la izquierda (*padding*). Cada grupo de 3 bits se convierte a su dígito octal equivalente.[3]
    - **Hexadecimal:** El proceso es idéntico, pero se agrupan los bits de 4 en 4.[15]

Ejemplo Guiado: Convertir 1101011102 a octal y hexadecimal.

- A Octal (grupos de 3):
  110 101 110
  6 5 6
  Resultado: 6568
- A Hexadecimal (grupos de 4):
  0001 1010 1110 (se añade padding de tres ceros a la izquierda)
  1 A E
  Resultado: 1AE16
- De Octal y Hexadecimal a Binario:
  El proceso es el inverso. Cada dígito octal se expande a su representación binaria de 3 bits, y cada dígito hexadecimal a su representación de 4 bits.

Ejemplo Guiado: Convertir 7B216 a binario.

1. Separar los dígitos: 7, B, 2
2. Convertir cada uno a binario de 4 bits:
    - 7⟶0111
    - B⟶1011
    - 2⟶0010
3. Unir las secuencias de bits: 011110110010
   Resultado: 7B216=111101100102 (se pueden omitir los ceros iniciales).

1.6 Ejercicios Prácticos y Guiados

Para consolidar los conocimientos, se proponen los siguientes ejercicios. Se recomienda intentar resolverlos utilizando ambos métodos (formal y práctico) para ganar fluidez.

Ejercicios Propuestos:

1. Convertir los siguientes números binarios a decimal:
    - a) 11012
    - b) 10011012
    - c) 111111112
    - d) 101010102
2. Convertir los siguientes números decimales a binario:
    - a) 2910

    - b) 15010
    - c) 25510
    - d) 9910
3. Convertir el número binario 111001011011012 a:
    - a) Octal
    - b) Hexadecimal
4. Convertir el número hexadecimal A4F16 a:
    - a) Binario
    - b) Decimal (Sugerencia: convierta primero a binario y luego a decimal).
5. Convertir el número octal 5728 a:
    - a) Binario
    - b) Decimal

Ejercicios Guiados (Soluciones Detalladas):

- **Solución 1.a) 11012 a decimal (Método Tabla):**

8	4	2	1
1	1	0	1

Suma: $8 + 4 + 1 = 13$. Resultado: 13_{10}.

- **Solución 2.a) 2910 a binario (Método Divisiones):**
    - 29÷2=14 (Residuo 1)
    - 14÷2=7 (Residuo 0)
    - 7÷2=3 (Residuo 1)
    - 3÷2=1 (Residuo 1)
    - 1÷2=0 (Residuo 1)

      Leyendo de abajo hacia arriba: 111012.
- **Solución 3) 111001011011012 a octal y hexadecimal:**
    - **A Octal (grupos de 3):** 011 100 101 101 101 ⟶ 345558
    - **A Hexadecimal (grupos de 4):** 0011 1001 0110 1101 ⟶ 396D16

A continuación se presenta una tabla de referencia para facilitar la verificación y el estudio de los patrones de conversión.

Decimal	Binario (8 bits)	Hexadecimal	Octal
0	00000000	0	0
1	00000001	1	1
2	00000010	2	2
3	00000011	3	3
4	00000100	4	4
5	00000101	5	5
6	00000110	6	6

7	00000111	7	7
8	00001000	8	10
9	00001001	9	11
10	00001010	A	12
11	00001011	B	13
12	00001100	C	14
13	00001101	D	15
14	00001110	E	16
15	00001111	F	17
16	00010000	10	20
...	...	...	...
32	00100000	20	40

Capítulo 2: Almacenamiento Interno de Datos Numéricos

Una vez comprendidos los sistemas de numeración, el siguiente paso es entender cómo un ordenador utiliza secuencias de bits de longitud fija (por ejemplo, 8, 16, 32 o 64 bits) para almacenar diferentes tipos de números.

2.1 Representación de Números Enteros

Enteros sin Signo (Unsigned Integers)

Esta es la forma más directa de representación. Un número entero sin signo (es decir, un número natural, incluyendo el cero) se almacena como su equivalente binario directo. Con un número fijo de n bits, se pueden representar 2n valores distintos, desde 0 hasta 2n−1.[29]

- **Ejemplo (8 bits):** Se pueden representar 28=256 valores.
    - El valor mínimo es 0, representado como 00000000.
    - El valor máximo es 28−1=255, representado como 11111111.

Enteros con Signo (Signed Integers)

El principal desafío es cómo representar el signo (positivo o negativo) utilizando únicamente

ceros y unos.[31] A lo largo de la historia de la computación, se han propuesto varios métodos, pero uno ha prevalecido por su eficiencia.

- Sistema Signo-Magnitud:
  Este es el enfoque más intuitivo. Se reserva el bit más a la izquierda (el MSB) para representar el signo: 0 para positivo y 1 para negativo. Los n−1 bits restantes representan la magnitud (el valor absoluto) del número en binario puro.32
    - **Ejemplo (+25 y -25 en 8 bits):**
      - +2510=000110012 (Signo '0', Magnitud '0011001')
      - −2510=100110012 (Signo '1', Magnitud '0011001')
    - **Desventajas:** A pesar de su simplicidad conceptual, este sistema tiene dos problemas graves que lo hacen ineficiente en la práctica.
      1. **Doble representación del cero:** Existen un "+0" (00000000) y un "-0" (10000000). Esto desperdicia una combinación de bits y complica las comparaciones.[31]
      2. **Complejidad aritmética:** Las operaciones de suma y resta requieren una lógica compleja. El hardware (la Unidad Aritmético-Lógica o ALU de la CPU) tendría que comprobar los signos de los operandos antes de decidir si debe sumar o restar las magnitudes, lo cual es costoso y lento.[33]
- Sistema de Complemento a 2 (Ca2):
  Este es el método estándar utilizado por la práctica totalidad de los ordenadores modernos para representar enteros con signo.33 Soluciona los problemas del sistema signo-magnitud de una manera elegante.
    - **Representación:**
      - Los **números positivos** se representan igual que en signo-magnitud: el bit de signo es 0 y el resto es la magnitud en binario.
      - Los **números negativos** se representan mediante un procedimiento específico.
    - **Procedimiento para obtener el Ca2 de un número negativo:**
      1. Se parte del valor absoluto del número (su versión positiva) en binario, usando el número total de bits disponibles (ej. 8 bits).
      2. Se invierten todos los bits (se cambian los 0 por 1 y los 1 por 0). Este paso se conoce como **complemento a 1**.
      3. Se suma 1 al resultado de la inversión.[35]

Ejemplo Guiado: Representar −4510 en Ca2 de 8 bits.

1. Representar +4510 en 8 bits:
   4510=001011012
2. Invertir los bits (Complemento a 1):
   11010010
3. Sumar 1:
   11010010 + 1 = 11010011
4. **Resultado:** La representación de −4510 en Ca2 de 8 bits es 110100112.

Nótese que el bit más significativo sigue funcionando como un indicador de signo (0 para

positivo, 1 para negativo), pero el valor de la magnitud no es directo.

- **Ventajas del Complemento a 2:**
    1. **Única representación del cero:** 00000000 es el único cero. El antiguo "-0" (10000000 en signo-magnitud) ahora representa el número negativo más pequeño del rango (ej. -128 en 8 bits).[38]
    2. **Aritmética simplificada:** La ventaja crucial es que la resta se puede realizar como una suma. Para calcular A−B, la ALU simplemente calcula A+(Ca2 de B) y descarta cualquier acarreo final. Esto significa que el mismo circuito de hardware puede realizar tanto sumas como restas, simplificando enormemente el diseño de la CPU.[34]

Con n bits, el rango de representación en Ca2 es de −(2n−1) a 2n−1−1. Para 8 bits, el rango es de -128 a +127.

2.2 Representación de Números Reales (Coma Flotante)

Mientras que los enteros tienen una representación binaria exacta, los números reales (aquellos con una parte fraccionaria) presentan un desafío fundamental: el conjunto de los números reales es infinito y continuo. Un sistema digital, con un número finito de bits, no puede representar perfectamente todos los valores posibles. Por ejemplo, números como π o 1/3 tienen infinitos decimales. La computación resuelve este problema mediante una **aproximación** estandarizada.

Es crucial entender que la representación en coma flotante no es una forma de almacenar números reales de manera *perfecta*, sino una manera estandarizada y eficiente de almacenar una **aproximación** muy útil de ellos.[30] Esta naturaleza aproximada es la causa de muchos comportamientos aparentemente extraños en la programación, como el hecho de que en muchos lenguajes, la operación

0.1 + 0.2 no sea exactamente igual a 0.3. Esto se debe a que 0.1, 0.2 y 0.3 no pueden representarse de forma exacta en binario, y la suma de las aproximaciones de los dos primeros no coincide con la aproximación del tercero.

El Estándar IEEE 754

Para asegurar la compatibilidad entre diferentes sistemas, la industria adoptó el **estándar IEEE 754**, que define cómo se deben representar y operar los números en coma flotante.[42] El estándar se basa en la notación científica, que representa un número como:

Signo×Mantisa×BaseExponente

El estándar IEEE 754 define varios formatos de precisión. El más común para empezar es el de **precisión simple (32 bits)**, que divide los 32 bits de la siguiente manera [45]:

- **Bit de Signo (1 bit):** El bit más a la izquierda. 0 para números positivos, 1 para

negativos.

- **Exponente (8 bits):** Representa la magnitud del número. Para poder almacenar exponentes tanto positivos como negativos (ej. 1.2×105 vs 1.2×10−5), no se usa Ca2, sino un formato llamado **exceso-127**. El valor real del exponente se obtiene restando 127 al valor almacenado: Exponentereal=Exponentealmacenado−127. Esto permite una comparación más rápida de los números.
- **Mantisa o Significando (23 bits):** Representa los dígitos significativos del número. Para maximizar la precisión, los números se **normalizan**. Esto significa que la coma se desplaza hasta que haya un único '1' a su izquierda (forma 1.fraccio´n). Dado que este '1' siempre está presente en un número normalizado, no es necesario almacenarlo. Se conoce como el **bit implícito** o *hidden bit*, lo que en la práctica otorga 24 bits de precisión a la mantisa.[47]

Ejemplo Guiado: Codificar −118.62510 en formato IEEE 754 de 32 bits.[46]

1. **Signo:** El número es negativo, por lo tanto, el bit de signo es **1**.
2. **Convertir a binario:**
    - Parte entera: 11810=11101102.
    - Parte fraccionaria: 0.625×2=1.25 (acarreo **1**); 0.25×2=0.5 (acarreo **0**); 0.5×2=1.0 (acarreo **1**). La parte fraccionaria es .1012.
    - Número completo: 1110110.1012.
3. Normalizar el número: Desplazar la coma binaria hasta que solo quede un '1' a la izquierda.
   1110110.101=1.110110101×26.
   El desplazamiento fue de 6 posiciones, por lo que el exponente real es 6.
4. Calcular el exponente en exceso-127:
   Exponentealmacenado=Exponentereal+127=6+127=133.
   Ahora, convertir 133 a binario de 8 bits: 13310=100001012.
5. Obtener la mantisa:
   La mantisa son los dígitos a la derecha de la coma en el número normalizado: 110110101.
   Se debe rellenar con ceros a la derecha hasta completar los 23 bits requeridos: 11011010100000000000000.
6. **Unir las tres partes:**
    - Signo: 1
    - Exponente: 10000101
    - Mantisa: 11011010100000000000000
      Resultado Final: 1 10000101 11011010100000000000000

El estándar también define representaciones para valores especiales como cero, infinito y "NaN" (Not a Number), que resultan de operaciones como la división por cero.[43]

2.3 Ejercicios Prácticos y Guiados

1. Representar los siguientes números enteros en formato **Complemento a 2 de 8 bits**:
    - a) +6010
    - b) −6010
    - c) −110
    - d) −12810
2. Dado el número en Ca2 de 8 bits 110011002, ¿cuál es su valor decimal?
3. **Ejercicio Guiado: Codificar 12.510 en IEEE 754 de 32 bits.**
    - **Paso 1: Signo.** El número es positivo. Bit de signo = **0**.
    - **Paso 2: Binario.** 1210=11002. 0.510=.12. Número completo: 1100.12.
    - **Paso 3: Normalización.** 1100.1=1.1001×23. El exponente real es 3.
    - **Paso 4: Exponente en exceso-127.** 3+127=130. 13010=100000102.
    - **Paso 5: Mantisa.** La parte fraccionaria es 1001. Rellenamos hasta 23 bits: 10010000000000000000000.
    - **Resultado:** 0 10000010 10010000000000000000000.

Capítulo 3: Representación de Texto: De Símbolos a Bytes

La representación de texto es la capa de abstracción que se construye sobre la representación numérica. La idea fundamental es simple: asignar un número único a cada carácter (letra, número, símbolo de puntuación, etc.). El ordenador almacena estos números, y el software (editor de texto, navegador web) los utiliza para mostrar los caracteres correspondientes en la pantalla.

3.1 La Evolución de la Codificación: ASCII, Unicode y UTF-8

- **ASCII (American Standard Code for Information Interchange):** Fue uno de los primeros y más influyentes estándares de codificación. Utiliza 7 bits, lo que le permite representar 27=128 caracteres distintos. Esto incluye las letras mayúsculas y minúsculas del alfabeto inglés, los números del 0 al 9, símbolos de puntuación y caracteres de control no imprimibles. ASCII fue suficiente para la computación temprana en el mundo angloparlante, pero resultó completamente inadecuado para representar la vasta diversidad de idiomas del mundo, que incluyen acentos (á, é, í), caracteres únicos (ñ, ç) y alfabetos completamente diferentes (cirílico, griego, árabe, kanji).[50]
- **Unicode:** Para resolver las limitaciones de ASCII y de cientos de otras codificaciones incompatibles, se creó el estándar Unicode. Es importante entender que **Unicode no es una codificación, sino un conjunto de caracteres universal**. Su objetivo es proporcionar un número único y unívoco, llamado **punto de código** (*code point*), para cada carácter de cada idioma, pasado o presente.[53] Los puntos de código se escriben

en hexadecimal con el prefijo "U+", por ejemplo, U+0041 para la letra 'A' y U+00F1 para la 'ñ'. Unicode puede representar más de un millón de caracteres, incluyendo alfabetos, símbolos matemáticos e incluso emojis.

- **UTF-8 (Unicode Transformation Format - 8-bit):** Si Unicode es el "qué" (el mapa de caracteres), las codificaciones como UTF-8 son el "cómo" (cómo se almacenan esos puntos de código en bytes). UTF-8 es la codificación de Unicode más utilizada en el mundo, especialmente en la web.[58] Su diseño es ingenioso y es la clave de su éxito:
    - **Longitud variable:** Utiliza de 1 a 4 bytes para representar un punto de código Unicode. Los caracteres más comunes requieren menos bytes, lo que lo hace eficiente en cuanto a espacio.[59]
    - **Retrocompatibilidad con ASCII:** Los primeros 128 puntos de código de Unicode son idénticos a los de ASCII. UTF-8 codifica estos 128 caracteres usando un solo byte, que es numéricamente idéntico a su código ASCII. Esto significa que cualquier texto ASCII válido es también un texto UTF-8 válido, lo que facilitó enormemente la transición de sistemas antiguos a Unicode.[51]

En resumen: **ASCII** es un conjunto de caracteres antiguo y limitado. **Unicode** es un conjunto de caracteres universal y moderno que asigna un número a cada símbolo. **UTF-8** es el método más popular para almacenar esos números de Unicode en bytes.

3.2 ¿Por qué aparecen "caracteres extraños"? Entendiendo el Mojibake

El fenómeno de ver texto corrupto o caracteres sin sentido (como ProgramaciÃ³n en lugar de Programación) se conoce como **Mojibake**.[52] Este problema no es aleatorio; es el resultado determinista de una discrepancia de codificación: un texto que fue guardado (codificado) con un estándar es leído (decodificado) utilizando un estándar diferente.
Para entenderlo, es necesario analizar el proceso a nivel de bytes. Sigamos la cadena causal que lleva a un error común en sistemas Windows:

1. **Objetivo:** Almacenar la palabra "Programación". El carácter problemático es la 'ó'.
2. **Del Carácter al Punto de Código Unicode:** En el estándar Unicode, el carácter 'ó' tiene asignado el punto de código U+00F3.
3. **Del Punto de Código a los Bytes en UTF-8:** La codificación UTF-8 tiene reglas para convertir puntos de código en secuencias de bytes. El punto de código U+00F3 (que en binario es 11110011) cae en el rango que requiere 2 bytes. La regla de codificación produce la siguiente secuencia de bytes:
    - Byte 1: 11000011 (hexadecimal: C3)
    - Byte 2: 10110011 (hexadecimal: B3)
      Por lo tanto, en un archivo guardado como UTF-8, el carácter 'ó' se representa físicamente por la secuencia de dos bytes C3 B3.
4. **El Error: La Interpretación Incorrecta:** Ahora, un programa (un editor de texto, un navegador, una base de datos) abre este archivo, pero está configurado para interpretarlo usando la codificación Windows-1252 (también conocida como ANSI en

configuraciones de Windows para Europa Occidental). Esta codificación es de un solo byte; asume que cada byte representa un carácter completo.

5. **Decodificación Byte por Byte:**
    - El programa lee el primer byte, C3. Consulta su tabla de caracteres Windows-1252 y encuentra que el código C3 corresponde al carácter 'Ã'.
    - A continuación, lee el segundo byte, B3. Consulta la misma tabla y encuentra que el código B3 corresponde al carácter '³'.
6. **Resultado:** Los dos bytes que en UTF-8 representan correctamente un solo carácter ('ó') son interpretados erróneamente por Windows-1252 como dos caracteres distintos ('Ã³'). El texto en pantalla muestra ProgramaciÃ³n.

Este análisis demuestra que los "caracteres extraños" son una consecuencia lógica y predecible de aplicar la tabla de decodificación incorrecta a una secuencia de bytes.

3.3 Laboratorio Práctico: El Caso de los Acentos en Visual Studio Code (Windows)

Este ejercicio práctico permite a los alumnos replicar y solucionar un problema de codificación de Mojibake, solidificando la comprensión teórica.

Objetivo: Observar directamente el efecto de una discrepancia de codificación y aprender a corregirla.

Pasos Guiados:

1. **Crear el archivo:** Abra Visual Studio Code. Cree un nuevo archivo (Archivo > Nuevo archivo de texto).
2. Añadir contenido: Escriba el siguiente texto en el editor:
   La solución a la codificación es la unificación.
3. **Verificar la codificación por defecto:** Observe la barra de estado en la esquina inferior derecha de la ventana de VS Code. Por defecto, debería mostrar "UTF-8".[63] Esto indica que VS Code está interpretando y guardará el archivo usando la codificación UTF-8.
4. **Guardar en UTF-8:** Guarde el archivo con el nombre prueba_utf8.txt. Cierre el archivo y vuelva a abrirlo. El texto debería mostrarse perfectamente, con todos sus acentos.
5. **Provocar el error de codificación:** Con el archivo prueba_utf8.txt abierto, haga clic en la etiqueta "UTF-8" de la barra de estado. Aparecerá un menú en la parte superior.
6. Seleccione la opción **"Guardar con codificación"** (*Save with Encoding*).[65]
7. En la lista de codificaciones que aparece, desplácese y seleccione **"Western (Windows 1252)"**. VS Code guardará una nueva versión del archivo utilizando esta codificación. Puede guardar con un nuevo nombre, como prueba_windows1252.txt.
8. **Observar el Mojibake:** Cierre el archivo prueba_windows1252.txt. Ahora, fuerce a VS Code a interpretarlo como si fuera UTF-8. Para ello, haga clic de nuevo en la etiqueta de codificación (que ahora podría decir "Windows 1252") y seleccione **"Reabrir con codificación"** (*Reopen with Encoding*). En la lista, elija "UTF-8".

9. Análisis del resultado: El texto aparecerá corrupto. Verá algo similar a:
   La soluciÃ³n a la codificaciÃ³n es la unificaciÃ³n.
   Este es el Mojibake. Los bytes que representan 'ó' y 'ú' en la codificación Windows-1252 están siendo decodificados erróneamente según las reglas de UTF-8.
10. **Solución:** Para corregir la visualización, simplemente vuelva a reabrir el archivo con la codificación correcta. Haga clic en la etiqueta de codificación, seleccione "Reabrir con codificación" y elija "Western (Windows 1252)". El texto volverá a la normalidad. Para convertirlo permanentemente a un estándar moderno, guárdelo de nuevo con la codificación UTF-8.

Este ejercicio demuestra de forma práctica que un archivo de texto no es solo una secuencia de caracteres, sino una secuencia de bytes que requiere una "clave" (la codificación correcta) para ser interpretada correctamente.

Capítulo 4: Representación de Imágenes: Píxeles vs. Ecuaciones

Las imágenes digitales son otro pilar de la información computacional. Se dividen en dos categorías fundamentalmente diferentes según cómo se almacena la información visual: imágenes de mapa de bits e imágenes vectoriales.

4.1 Imágenes de Mapa de Bits (Raster): Un Mosaico de Píxeles

Una imagen de mapa de bits, también conocida como imagen *raster* o *bitmap*, es una rejilla o matriz rectangular de pequeños puntos de color llamados **píxeles** (*picture elements*).[67] Esencialmente, la imagen es un mosaico donde cada tesela (píxel) tiene un color específico. El archivo de imagen almacena la información de color de cada uno de estos píxeles, uno por uno.

Para entender una imagen de mapa de bits, es necesario conocer tres conceptos clave:

- **Resolución:** Es una medida de la densidad de píxeles en la imagen, comúnmente expresada en **píxeles por pulgada** (PPI, *Pixels Per Inch*) para pantallas, o **puntos por pulgada** (DPI, *Dots Per Inch*) para impresión. Una resolución más alta significa más píxeles en la misma área, lo que se traduce en una imagen con mayor detalle y nitidez.[71]
- **Dimensiones:** Se refiere al número total de píxeles de ancho y alto de la imagen (por ejemplo, 1920x1080 píxeles).
- **Profundidad de Color:** Es el número de bits utilizados para representar el color de un único píxel. Una mayor profundidad de bits permite que la imagen contenga una paleta de colores más amplia y rica.[68]
    - **1 bit:** 2 colores (blanco y negro).
    - **8 bits:** 28=256 colores (típico de imágenes GIF).

    - **24 bits (Color Verdadero):** 224≈16.7 millones de colores. Utiliza 8 bits para cada uno de los tres canales de color: Rojo, Verde y Azul (RGB). Es el estándar para fotografías (JPEG).
    - **32 bits:** Similar a 24 bits, pero añade 8 bits adicionales para un canal alfa, que define el nivel de transparencia de cada píxel (formato PNG).

La principal desventaja de las imágenes de mapa de bits es su **escalabilidad limitada**. Como la imagen está definida por un número fijo de píxeles, al intentar ampliarla, el software simplemente agranda cada píxel. Esto provoca que los píxeles individuales se vuelvan visibles, creando un efecto de "dientes de sierra" o bloques, conocido como **pixelación**, que degrada la calidad de la imagen.[72]

4.2 Imágenes Vectoriales: Gráficos Definidos por Matemáticas

A diferencia de los mapas de bits, una imagen vectorial no está compuesta por una rejilla de píxeles. En su lugar, se define mediante un conjunto de objetos geométricos (puntos, líneas, curvas de Bézier, polígonos) que son descritos por ecuaciones matemáticas.[69] El archivo de imagen no almacena el color de millones de píxeles, sino las "instrucciones" para dibujar la imagen: "dibuja una línea desde el punto A al punto B con un grosor de 2px y color azul", "dibuja un círculo con centro en C, radio R y relleno rojo".

La ventaja fundamental de este enfoque es la **escalabilidad perfecta**. Cuando se redimensiona una imagen vectorial, el software no estira píxeles. En su lugar, recalcula las fórmulas matemáticas para el nuevo tamaño y vuelve a dibujar la imagen desde cero. Como resultado, la imagen mantiene una nitidez y claridad perfectas, sin importar cuánto se amplíe o se reduzca.[72] Además, como solo se almacenan las fórmulas, los archivos suelen ser mucho más pequeños que los mapas de bits de alta resolución.

4.3 Comparativa y Aplicaciones: ¿Cuándo Elegir Cada Formato?

La elección entre un formato de mapa de bits y uno vectorial depende enteramente de la naturaleza de la imagen y su uso previsto.

- **Mapa de Bits es ideal para:**
    - **Fotografías y arte realista:** La estructura de píxeles es perfecta para capturar los matices sutiles, las gradaciones de color complejas y las texturas del mundo real.[69] Es prácticamente imposible replicar una fotografía con vectores.
    - **Edición a nivel de píxel:** Programas como Adobe Photoshop permiten manipular píxeles individuales para retoques fotográficos detallados.
- **Vectorial es ideal para:**
    - **Logotipos e iconografía:** Un logotipo debe poder usarse en una tarjeta de visita y en una valla publicitaria sin perder calidad. La escalabilidad de los vectores es esencial para esto.[69]

- **Ilustraciones, tipografías y diagramas:** Gráficos con formas limpias y colores planos se benefician de la precisión y la escalabilidad de los vectores.
- **Diseño web y de interfaces:** Los iconos y gráficos en formato SVG (Scalable Vector Graphics) se adaptan a cualquier tamaño de pantalla y resolución sin degradarse.

La siguiente tabla resume las diferencias clave entre ambos formatos.

Característica	Mapa de Bits (Raster)	Vectorial
Unidad Básica	Píxel (un punto de color en una rejilla) [69]	Objeto matemático (punto, línea, curva, polígono) [74]
Escalabilidad	Limitada. Pierde calidad (se pixela) al ampliar [72]	Infinita. Se puede redimensionar a cualquier tamaño sin pérdida de calidad [77]
Tamaño de Archivo	Grande, especialmente en alta resolución (depende del número de píxeles) [69]	Pequeño (depende de la complejidad de las formas, no del tamaño de la imagen) [77]
Edición	A nivel de píxel (retoque, filtros) [67]	A nivel de objeto (modificar formas, colores, trazos) [77]
Casos de Uso Típicos	Fotografías, imágenes realistas, arte digital complejo [75]	Logotipos, iconos, ilustraciones, tipografía, diagramas, infografías [69]
Formatos Comunes	JPEG, PNG, GIF, BMP, TIFF [71]	SVG, AI, EPS, PDF [69]

Capítulo 5: Una Mirada a la Representación Multimedia

La representación multimedia combina los conceptos anteriores para digitalizar sonido y vídeo, las formas más complejas de información que manejamos.

5.1 Digitalización de Sonido: Capturando la Onda Analógica

El sonido en el mundo físico es una onda analógica: una vibración continua que se propaga a través de un medio como el aire. Para que un ordenador pueda almacenar y procesar sonido, esta onda continua debe convertirse en una secuencia de números discretos (digitales). Este proceso se llama digitalización o conversión analógico-digital (ADC) y se basa en dos parámetros fundamentales [82]:

- **Muestreo (Sampling):** Consiste en medir la amplitud (el "volumen") de la onda sonora a intervalos de tiempo muy cortos y regulares. El número de mediciones o "muestras" que se toman por segundo se conoce como la **frecuencia de muestreo**, y se mide en

Hercios (Hz) o Kilohercios (kHz).[82] Una frecuencia de muestreo más alta captura la forma de la onda con mayor fidelidad, lo que resulta en un sonido de mayor calidad. El estándar para los CD de audio es de 44,100 muestras por segundo (44.1 kHz), que es suficiente para capturar todo el rango de frecuencias audibles por el oído humano.

- **Cuantificación (Quantization):** Cada una de las muestras tomadas es un valor de amplitud. La cuantificación es el proceso de asignar un valor numérico discreto a cada una de estas muestras. La precisión de esta asignación viene determinada por la **profundidad de bits** (*bit depth*). La profundidad de bits es el número de bits que se utilizan para representar el valor de cada muestra.[83]
    - Una mayor profundidad de bits permite representar un mayor número de niveles de amplitud, lo que se traduce en un mayor **rango dinámico** (la diferencia entre los sonidos más suaves y los más fuertes) y una menor distorsión, conocida como **ruido de cuantificación**.
    - El estándar de CD utiliza una profundidad de 16 bits, que permite 216=65,536 niveles de amplitud distintos. En la producción de audio profesional, es común trabajar con 24 o incluso 32 bits para una mayor precisión.

En resumen, la digitalización del sonido convierte una onda continua en una larga secuencia de números. La calidad de esta conversión depende de con qué frecuencia se mide la onda (frecuencia de muestreo) y con qué precisión se registra cada medición (profundidad de bits).

5.2 Digitalización de Vídeo: El Cine en el Ordenador

El vídeo digital es, en esencia, una ilusión de movimiento creada al mostrar una secuencia de imágenes fijas a una velocidad suficientemente alta. Cada una de estas imágenes es un **fotograma** (*frame*), que es una imagen de mapa de bits.[86]

- **Tasa de Fotogramas (Frame Rate):** Es la velocidad a la que se muestran estos fotogramas, medida en **fotogramas por segundo** (fps). El cine tradicional usa 24 fps, mientras que la televisión y los vídeos digitales suelen usar 30 o 60 fps para una mayor fluidez.

El principal problema del vídeo digital es el enorme tamaño de los datos. Un solo segundo de vídeo sin comprimir a resolución Full HD (1920x1080) y 24 bits de color a 30 fps puede ocupar cientos de megabytes. Almacenar o transmitir esto por internet sería impracticable. La solución es la **compresión**.

La Solución: Códecs y Contenedores

Para manejar el vídeo digital de forma eficiente, se utilizan dos componentes que a menudo se confunden:

- **Códec (Codificador-Decodificador):** Un códec es un **algoritmo** de software o

hardware diseñado para **comprimir** (codificar) los datos de vídeo para reducir su tamaño y **descomprimir** (decodificar) para su reproducción.[87] Los códecs modernos utilizan técnicas muy sofisticadas para eliminar la redundancia espacial (dentro de un mismo fotograma) y temporal (entre fotogramas consecutivos). Por ejemplo, en lugar de almacenar cada fotograma completo, un códec puede almacenar un fotograma clave y, para los siguientes, solo la información que ha cambiado (ej. el movimiento de un personaje sobre un fondo estático). Códecs populares incluyen H.264 (AVC) y su sucesor más eficiente, H.265 (HEVC).

- **Contenedor:** Es el **formato de archivo** que envuelve y organiza los diferentes flujos de datos que componen un archivo multimedia. Un archivo de vídeo típico contiene un flujo de vídeo (comprimido por un códec), uno o más flujos de audio (también comprimidos por un códec de audio como AAC), y a menudo metadatos como subtítulos o información de capítulos.[87] Formatos de contenedor comunes son .MP4, .MKV, .AVI y .MOV.

Una buena analogía es pensar en un libro: el **códec** es el idioma en el que está escrito el texto (inglés, español), mientras que el **contenedor** es el formato físico del libro (tapa dura, rústica), que agrupa las páginas (vídeo), el índice (metadatos) y quizás un CD de audio adjunto (audio). Para poder leer el libro, necesitas entender el idioma (tener el códec correcto) y poder abrir el libro (que tu reproductor sea compatible con el contenedor).

Conclusión: La Abstracción Digital Unificada

A lo largo de estos capítulos, hemos realizado un viaje desde el nivel más bajo de la computación hasta las complejas representaciones multimedia. Este recorrido no ha sido una colección de temas inconexos, sino la ascensión por una **jerarquía de abstracción** que define la esencia de la informática moderna.

Partimos del **bit**, la unidad de información más simple, para entender cómo, mediante las reglas de los **sistemas de numeración**, se pueden construir todos los números que un ordenador puede manejar. Vimos que esta representación numérica no es monolítica, sino que se adapta a las necesidades: enteros sin signo, el eficiente sistema de complemento a 2 para enteros con signo, y la aproximación estandarizada de IEEE 754 para números reales.

Sobre esta base numérica, se erigen las siguientes capas. El **texto** cobra vida cuando asignamos números (puntos de código Unicode) a los símbolos de todos los idiomas, y lo almacenamos eficientemente con codificaciones como UTF-8. Las **imágenes** se materializan como vastas rejillas de números que definen el color de cada píxel, o como elegantes ecuaciones matemáticas en el caso de los gráficos vectoriales. Finalmente, el **sonido y el vídeo** se digitalizan convirtiendo ondas y secuencias de imágenes en gigantescos conjuntos de datos numéricos, cuyo manejo solo es posible gracias a los ingeniosos algoritmos de compresión de los códecs.

Cada capa oculta la complejidad de la anterior, permitiendo a los programadores y usuarios interactuar con conceptos de alto nivel como "carácter", "color" o "sonido", sin necesidad de

manipular directamente los ceros y unos subyacentes. Comprender esta estructura, desde el bit hasta el vídeo, es comprender el lenguaje fundamental de la tecnología digital. Es la clave para pasar de ser un mero usuario de la tecnología a un creador capaz de entender, diagnosticar y construir las herramientas del futuro.

Obras citadas

1. Sistemas de numeración Binario, Octal y Hexadecimal, fecha de acceso: septiembre 19, 2025, https://victorvillamon.files.wordpress.com/2014/10/sistemas-de-numeracic3b3n-binario-octal-y-hexadecimal.pdf
2. Sistemas de Numeración y Conversiones | Binario - Octal - Hexadecimal - YouTube, fecha de acceso: septiembre 19, 2025, https://www.youtube.com/watch?v=nhhvcUfYSIM
3. SISTEMAS DE NUMERACIÓN Sistema de numeración decimal: 8·103 2·102 4·101 5·100 9·10−1 7·10−2=8245,97, fecha de acceso: septiembre 19, 2025, https://gc.scalahed.com/syllabus/cloud/visor.php?container=L1IS109_1102_675_37121_0&object=Sistemas%20de%20numeraci%C3%B3n.pdf
4. repository.unad.edu.co, fecha de acceso: septiembre 19, 2025, https://repository.unad.edu.co/reproductor-ova/10596_11599/sistemas_de_numeracin.html#:~:text=La%20base%20indica%20el%20n%C3%BAmero,num%C3%A9ricos%20(llamado%20Sistema%20num%C3%A9rico).
5. ¿Cuál es la base de los diferentes sistemas de numeración? - Quora, fecha de acceso: septiembre 19, 2025, https://es.quora.com/Cu%C3%A1l-es-la-base-de-los-diferentes-sistemas-de-numeraci%C3%B3n
6. es.wikipedia.org, fecha de acceso: septiembre 19, 2025, https://es.wikipedia.org/wiki/Base_(aritm%C3%A9tica)#:~:text=En%20un%20sistema%20de%20numeraci%C3%B3n,s%C3%ADmbolos%20presentes%20en%20dicho%20sistema.
7. Sistema decimal y binario - Portal Académico CCH, fecha de acceso: septiembre 19, 2025, https://portalacademico.cch.unam.mx/cibernetica1/sistemas-de-numeracion/sistema-decimal-y-binario
8. Respuestas de Flexi - ¿Qué es el sistema numérico posicional? | CK-12 Foundation, fecha de acceso: septiembre 19, 2025, https://www.ck12.org/flexi/es/grado-6/diferencia-de-enteros-al-utilizar-una-recta-numerica/que-es-el-sistema-numerico-posicional/
9. Sistema de numeración - Concepto, tipos, características y ejemplos, fecha de acceso: septiembre 19, 2025, https://concepto.de/sistema-de-numeracion/
10. CAPITULO TRES 3. LOS SISTEMAS DE NUMERACIÓN, fecha de acceso: septiembre 19, 2025, https://www.cs.buap.mx/~andrex/ensamblador/sistemas-de-numeracion.pdf
11. Sistemas de Numeracion - CORE, fecha de acceso: septiembre 19, 2025, https://core.ac.uk/download/235864347.pdf

12. Sistemas de numeración (Binario, Octal, Hexadecimal, Decimal) | PPSX - SlideShare, fecha de acceso: septiembre 19, 2025, https://es.slideshare.net/slideshow/sistemas-de-numeracin-10566789/10566789
13. Sistema octal - Wikipedia, la enciclopedia libre, fecha de acceso: septiembre 19, 2025, https://es.wikipedia.org/wiki/Sistema_octal
14. Sistema hexadecimal: te explicamos qué es este sistema numérico ..., fecha de acceso: septiembre 19, 2025, https://www.ionos.com/es-us/digitalguide/servidores/know-how/sistema-hexadecimal/
15. Bases Numéricas, fecha de acceso: septiembre 19, 2025, https://triton.astroscu.unam.mx/fruiz/introduccion/introduccion_computacion/Bases%20Nume%CC%81ricas.pdf
16. Sistema binario - Wikipedia, la enciclopedia libre, fecha de acceso: septiembre 19, 2025, https://es.wikipedia.org/wiki/Sistema_binario
17. ¿Cómo convertir un número binario en decimal? (paso a paso) - EDteam, fecha de acceso: septiembre 19, 2025, https://ed.team/blog/como-convertir-un-numero-binario-en-decimal
18. Respuestas de Flexi - ¿Cómo convertir binario a decimal? | CK-12 Foundation, fecha de acceso: septiembre 19, 2025, https://www.ck12.org/flexi/es/grado-8/convertir-decimales-periodicos-en-fracciones/como-convertir-binario-a-decimal/
19. 1 Procedimiento para Pasar de Binario A Decimal y Viceversa | PDF - Scribd, fecha de acceso: septiembre 19, 2025, https://es.scribd.com/document/716298193/1-Procedimiento-para-pasar-de-binario-a-decimal-y-viceversa
20. Convertir Binario a Decimal - YouTube, fecha de acceso: septiembre 19, 2025, https://www.youtube.com/watch?v=lQpDcrMrlsE
21. Conversión binario a decimal - UTN-FRRQ, fecha de acceso: septiembre 19, 2025, https://frrq.cvg.utn.edu.ar/mod/resource/view.php?id=14349
22. [Método Fácil] Convertir de Binario a Decimal y viceversa. - YouTube, fecha de acceso: septiembre 19, 2025, https://www.youtube.com/watch?v=c-hyLLdDt7I
23. CONVERTIR UN NÚMERO BINARIO A DECIMAL Super facil - Para principiantes - YouTube, fecha de acceso: septiembre 19, 2025, https://www.youtube.com/watch?v=jCWfMlWSXrM
24. ed.team, fecha de acceso: septiembre 19, 2025, https://ed.team/blog/sistemas-binarios-y-decimales#:~:text=Convertir%20un%20n%C3%BAmero%20decimal%20a,%2F%202%20%3D%205879%20Residuo%3A%201
25. Tema 1.3.2 Conversión de decimal a binario - Sistemas Digitales I, fecha de acceso: septiembre 19, 2025, https://cursos.clavijero.edu.mx/cursos/038_sgl/modulo1/contenidos/tema1.3.2.html
26. Sistema Decimal El sistema decimal emplea diez diferentes dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9). Por esto se dice que la “b, fecha de acceso: septiembre 19, 2025, http://fcaenlinea.unam.mx/anexos/1364/anexo_B_sistemas_de_numeracion_u2Act

1.pdf
27. Cómo pasar de DECIMAL a BINARIO (RÁPIDO Y FÁCIL!!!) | Ejercicios Resueltos de conversión [2025] - YouTube, fecha de acceso: septiembre 19, 2025, https://www.youtube.com/watch?v=nHT9p4madjo
28. Convertir de Decimal a Binario Usando Potencias de 2 | Paso a Paso - YouTube, fecha de acceso: septiembre 19, 2025, https://www.youtube.com/watch?v=LJzw2h7cSGg
29. Tipos de datos numéricos - Visual Basic - Microsoft Learn, fecha de acceso: septiembre 19, 2025, https://learn.microsoft.com/es-es/dotnet/visual-basic/programming-guide/language-features/data-types/numeric-data-types
30. Representación de Números, fecha de acceso: septiembre 19, 2025, http://users.df.uba.ar/dmitnik/metodosnumericos/algoritmos/RepresentacionNumerica.html
31. Representación de enteros - Organización de Computadoras, fecha de acceso: septiembre 19, 2025, http://orga.blog.unq.edu.ar/wp-content/uploads/sites/5/2018/04/orga_clase5_apuntesistemasdenumeracion.pdf
32. Representación de los números en la computadora. - FaMAF, fecha de acceso: septiembre 19, 2025, https://www.famaf.unc.edu.ar/~serra/representacion-numeros.pdf
33. Number Systems - HyperPhysics, fecha de acceso: septiembre 19, 2025, http://hyperphysics.phy-astr.gsu.edu/hbasees/Electronic/number2.html
34. www.reddit.com, fecha de acceso: septiembre 19, 2025, https://www.reddit.com/r/learnprogramming/comments/ask96a/what_is_twos_complement_and_what_is_the_use_of_it/?tl=es-419#:~:text=El%20complemento%20a%20dos%20es%20una%20forma%20de%20representar%20enteros,para%20sumar%20enteros%20sin%20signo.&text=0101%20%2B%201011%20%3D%200%2C%20por,1011%20%3D%20%2D5%20por%20definici%C3%B3n.
35. Tema 1.4.3 Representación de números con signo usando el ..., fecha de acceso: septiembre 19, 2025, https://cursos.clavijero.edu.mx/cursos/038_sgl/modulo1/contenidos/tema1.4.3.html
36. Datos, Operadores y Expresiones: Complemento a dos - Aula Virtual, fecha de acceso: septiembre 19, 2025, https://aulavirtual.fio.unam.edu.ar/mod/book/view.php?id=35570&chapterid=7899
37. Aritmética modular para entender el complemento a 2 - Víctor Iglesias, fecha de acceso: septiembre 19, 2025, https://www.victoriglesias.net/complemento-a-2-aritmetica-modular/
38. utn frsf – tecnicatura superior en mecatronica – sistemas digitales capitulo iii – aritmetica binaria 1/8, fecha de acceso: septiembre 19, 2025, https://frrq.cvg.utn.edu.ar/mod/resource/view.php?id=4732
39. Representación de enteros: Complemento a 2 | | UPV - YouTube, fecha de acceso: septiembre 19, 2025, https://www.youtube.com/watch?v=xyGXtdJpiDg
40. Describa las ventajas de la representación en complemento a 2 frente a la

representación signo-magnitud. a) R, fecha de acceso: septiembre 19, 2025, https://www.cartagena99.com/recursos/alumnos/apuntes/201401_P2.pdf
41. La aritmética de punto flotante puede dar un resultado inexacto en Excel - Microsoft Learn, fecha de acceso: septiembre 19, 2025, https://learn.microsoft.com/es-es/office/troubleshoot/excel/floating-point-arithmetic-inaccurate-result
42. Clase Nº6 Norma Ieee 754 1 | PDF | Aritmética | Arquitectura de Computadores - Scribd, fecha de acceso: septiembre 19, 2025, https://es.scribd.com/presentation/546156183/Clase-N%C2%BA6-Norma-Ieee-754-1
43. Antonio Bello - Tutorías UNED - IEEE 754 - El estándar IEEE para la aritmética en coma flotante, fecha de acceso: septiembre 19, 2025, https://www6.uniovi.es/~antonio/uned/ieee754/formato.html
44. IEEE 754 - Wikipedia, la enciclopedia libre, fecha de acceso: septiembre 19, 2025, https://es.wikipedia.org/wiki/IEEE_754
45. Informática de precisión simple, doble, múltiple y mixta - AMD, fecha de acceso: septiembre 19, 2025, https://www.amd.com/es/resources/articles/single-precision-vs-double-precision-main-differences.html
46. IEEE754 - Manjarrés, fecha de acceso: septiembre 19, 2025, https://www.manjarr.es/ieee754
47. Módulo 5 - Norma IEEE-754, fecha de acceso: septiembre 19, 2025, https://cs.uns.edu.ar/~ldm/mypage/data/oc/apuntes/2019-modulo5.pdf
48. Punto Flotante - Estándar IEEE 754 - IBCM, fecha de acceso: septiembre 19, 2025, http://ibcm.blog.unq.edu.ar/wp-content/uploads/sites/5/2015/08/apunte_punto_flotante.pdf
49. Representación de punto flotante de IEEE - Microsoft Learn, fecha de acceso: septiembre 19, 2025, https://learn.microsoft.com/es-es/cpp/build/ieee-floating-point-representation?view=msvc-170
50. Cómo se relaciona Unicode con estándares anteriores como ASCII y EBCDIC - IBM, fecha de acceso: septiembre 19, 2025, https://www.ibm.com/docs/es/i/7.5.0?topic=wu-how-unicode-relates-prior-standards-such-as-ascii-ebcdic
51. ASCII y UTF-8 - Apuntes de Programador, fecha de acceso: septiembre 19, 2025, https://apuntes.de/golang/ascii-y-utf8/
52. Mojibake, when encoding goes wrong | by Thomas Lamiraud - Medium, fecha de acceso: septiembre 19, 2025, https://medium.com/@thomas.lamiraud/mojibake-when-encoding-goes-wrong-0958d0631883
53. Cadenas en memoria - entender ASCII y Unicode - Tutkit.com, fecha de acceso: septiembre 19, 2025, https://www.tutkit.com/es/tutoriales-de-texto/11358-entender-cadenas-en-la-memoria-ascii-y-unicode
54. Qué es Unicode? Cómo se utiliza y qué ventajas tiene? | Lenovo España, fecha de

acceso: septiembre 19, 2025, https://www.lenovo.com/es/es/glossary/what-is-unicode/
55. www.lenovo.com, fecha de acceso: septiembre 19, 2025, https://www.lenovo.com/es/es/glossary/what-is-unicode/#:~:text=Unicode%20es%20un%20sistema%20de,incluidos%20alfabetos%2C%20ideogramas%20y%20s%C3%ADmbolos.
56. ¿Qué es Unicode? | Loqate | ES, fecha de acceso: septiembre 19, 2025, https://www.loqate.com/es/blog/que-es-unicode/
57. Unicode - Wikipedia, la enciclopedia libre, fecha de acceso: septiembre 19, 2025, https://es.wikipedia.org/wiki/Unicode
58. learn.microsoft.com, fecha de acceso: septiembre 19, 2025, https://learn.microsoft.com/es-es/windows/apps/design/globalizing/use-utf8-code-page#:~:text=UTF%2D8%20es%20la%20p%C3%A1gina,basadas%20en%20XML%20y%20*nix.
59. ¿Qué es UTF-8 y qué ventajas tiene? | Blog de Arsys, fecha de acceso: septiembre 19, 2025, https://www.arsys.es/blog/utf8
60. UTF-8 - Wikipedia, la enciclopedia libre, fecha de acceso: septiembre 19, 2025, https://es.wikipedia.org/wiki/UTF-8
61. Unicode: Qué es, cómo funciona y sus aplicaciones - GoDaddy, fecha de acceso: septiembre 19, 2025, https://www.godaddy.com/resources/es/crearweb/unicode-que-es-como-funciona-aplicaciones
62. Mojibake - Revealing Errors, fecha de acceso: septiembre 19, 2025, https://revealingerrors.com/mojibake
63. Visual Studio Code. Elegir Lenguaje y Codificación. - YouTube, fecha de acceso: septiembre 19, 2025, https://www.youtube.com/watch?v=ssTN6PcY6d0
64. Descripción de la codificación de archivo en VS Code y PowerShell - Microsoft Learn, fecha de acceso: septiembre 19, 2025, https://learn.microsoft.com/es-es/powershell/scripting/dev-cross-plat/vscode/understanding-file-encoding?view=powershell-7.5
65. Change the encoding of a file in Visual Studio Code - Stack Overflow, fecha de acceso: septiembre 19, 2025, https://stackoverflow.com/questions/30082741/change-the-encoding-of-a-file-in-visual-studio-code
66. Cambiar formato de codificación en Visual Studio Code - Devidence - Medium, fecha de acceso: septiembre 19, 2025, https://medium.com/@devidence/cambiar-formato-de-codificaci%C3%B3n-en-visual-studio-code-26d16596cf75
67. Gráfico de Mapa de Bits: ¿qué es? | Lenovo México, fecha de acceso: septiembre 19, 2025, https://www.lenovo.com/mx/es/glosario/mapa-grafico-de-bits/
68. MAPA DE BITS - elhacker.INFO, fecha de acceso: septiembre 19, 2025, https://www.elhacker.info/Cursos/Cursos%20PhotoShop/Retoque%20fotografico%20con%20Photoshop/unidad2/U2_ADJ_01_INFORMACION_DE_BIT.pdf
69. Diferencia entre la imagen de mapa de bits e imagen vectorial - Blog de Marcaprint, fecha de acceso: septiembre 19, 2025,

https://www.marcaprint.com/blog/diferencia-entre-bits-y-vectorial/
70. ¿Vectores o mapas de bits? Que diferencia hay entre una imagen vectorial y un Mapa de Bits. - YouTube, fecha de acceso: septiembre 19, 2025, https://www.youtube.com/watch?v=zoyKwCmkWsY
71. 2.4.2. Imágenes de mapas de bits, fecha de acceso: septiembre 19, 2025, https://libros.uvq.edu.ar/spm/242_imgenes_de_mapas_de_bits.html
72. Imagen vectorial o mapa de bits: pros y contras - IONOS, fecha de acceso: septiembre 19, 2025, https://www.ionos.com/es-us/digitalguide/paginas-web/diseno-web/imagen-vectorial-o-mapa-de-bits-pros-y-contras/
73. Diferencia entre bitmap y vector | Gráficas Papallona Valencia, fecha de acceso: septiembre 19, 2025, https://graficaspapallona.com/diferencia-entre-bitmap-y-vector/
74. ¿Qué son imágenes vectoriales? Usos, importancia y formatos - Aula Creactiva, fecha de acceso: septiembre 19, 2025, https://www.aulacreactiva.com/que-son-imagenes-vectoriales-en-diseno/
75. Los gráficos o imágenes vectoriales: características y usos - UNIR, fecha de acceso: septiembre 19, 2025, https://www.unir.net/revista/diseno/graficos-imagenes-vectoriales/
76. Gráficos vectoriales escalables - Wikipedia, la enciclopedia libre, fecha de acceso: septiembre 19, 2025, https://es.wikipedia.org/wiki/Gr%C3%A1ficos_vectoriales_escalables
77. Guía Completa sobre Imágenes Vectoriales: Qué Son, Ventajas, Creación y Uso en el Diseño | Pixartprinting, fecha de acceso: septiembre 19, 2025, https://www.pixartprinting.es/blog/guia-sobre-imagenes-vectoriales/
78. Descubre el Poder del Arte Vectorial: Consejos Esencial, fecha de acceso: septiembre 19, 2025, https://es.strikingly.com/blog/posts/descubre-el-poder-del-arte-vectorial-consejos-esenciales
79. ¿Qué diferencia hay entre una imagen vectorial y un mapa de bits? { Micro Conocimiento by @Mazzima - YouTube, fecha de acceso: septiembre 19, 2025, https://www.youtube.com/watch?v=RZywV73MDGM
80. Diferencias entre imágenes vectoriales y mapa de bits | El Blog de PCG, fecha de acceso: septiembre 19, 2025, https://www.pcgprint.com/blog-diferencias-imagen-vectoriale-mapa-de-bits/
81. Tipos de mapas de bits - Windows Forms - Microsoft Learn, fecha de acceso: septiembre 19, 2025, https://learn.microsoft.com/es-es/dotnet/desktop/winforms/advanced/types-of-bitmaps
82. La digitalización del sonido | Musicalecer. Creación, edición y producción musical, fecha de acceso: septiembre 19, 2025, https://musicalecer.com/el-sonido-digital/la-digitalizacion-del-sonido/
83. Profundidad de bits y frecuencia de muestreo comprension del impacto en el rendimiento del DAC - FasterCapital, fecha de acceso: septiembre 19, 2025, https://fastercapital.com/es/contenido/Profundidad-de-bits-y-frecuencia-de-mu

estreo--comprension-del-impacto-en-el-rendimiento-del-DAC.html
84. Frecuencia de muestreo y profundidad de bits en audio digital - Wood and Fire, fecha de acceso: septiembre 19, 2025, https://woodandfirestudio.com/es/sample-rate-bit-depth/
85. ¿Qué es la profundidad de bits en audio? - eMastered, fecha de acceso: septiembre 19, 2025, https://emastered.com/es/blog/bit-depth-in-audio
86. ¿Qué son los códecs de vídeo? | Guía básica de la cámara - fotokoch.de, fecha de acceso: septiembre 19, 2025, https://www.fotokoch.de/es/kamera-basics-was-sind-videocodecs.html
87. Qué es un códec de vídeo: Todo lo que debe saber - Dacast, fecha de acceso: septiembre 19, 2025, https://www.dacast.com/es/blog-es/que-es-un-codec-de-video/
88. Codificadores de Video: Cómo Funcionan y Su Papel en el Streaming y la Videovigilancia, fecha de acceso: septiembre 19, 2025, https://flussonic.com/es/blog/news/videokoder
89. ¿Qué son los CODEC & FORMATOS de VÍDEO? Todo lo que DEBES SABER sobre la COMPRESIÓN de VIDEO - YouTube, fecha de acceso: septiembre 19, 2025, https://www.youtube.com/watch?v=GlJtUxL8zzc&pp=0gcJCRsBo7VqN5tD