Conversiones entre distintas bases numéricas y sumas en la ALU

Tags | Lab

Esta guía te proporcionará una comprensión completa sobre la conversión entre diferentes bases numéricas, específicamente entre base 10 (decimal), base 2 (binario), base 8 (octal) y base 16 (hexadecimal). También incluye una tabla comparativa para ayudarte a visualizar las representaciones de números en diferentes bases. Además, se explicará cómo realizar sumas en una Unidad Lógica Aritmética (ALU).

Índice

Introducción a las Bases Numéricas
¿Por Qué se Hace la Conversión Entre Bases?
¿Cómo Convertir Entre Bases?
- De Decimal (Base 10) a Otras Bases
- De Binario (Base 2) a Otras Bases
- De Octal (Base 8) a Otras Bases
- De Hexadecimal (Base 16) a Otras Bases

Tabla Comparativa de Números en Diferentes Bases

Magnitud Verdadera
Complemento a uno
Ejemplo: Complemento a Uno
Complemento a dos
Pasos para Calcular el Complemento a Dos
Ejemplo: Complemento a Dos
Sumas en la ALU
Ejemplos
Videos Complementarios

Introducción a las Bases Numéricas

Un sistema de numeración o base numérica es un conjunto de símbolos y reglas utilizados para representar cantidades. Las bases más comunes son:

- **Base 10 (Decimal):** Es el sistema numérico que usamos cotidianamente, compuesto por los dígitos del 0 al 9.
- **Base 2 (Binario):** Utiliza solo dos dígitos, 0 y 1. Es fundamental en informática.
- **Base 8 (Octal):** Usa los dígitos del 0 al 7.
- **Base 16 (Hexadecimal):** Utiliza 16 dígitos, del 0 al 9 y de la A a la F (donde A=10, B=11, ..., F=15).

¿Por Qué se Hace la Conversión Entre Bases?

La conversión entre bases numéricas es esencial para:

1. **Interacción con Sistemas Digitales:** Las computadoras operan en binario, pero los humanos prefieren trabajar con sistemas como el decimal.
2. **Optimización en Programación:** El hexadecimal permite una representación más compacta de números binarios.
3. **Diseño de Circuitos:** Se utiliza para diseñar y analizar circuitos lógicos.
4. **Depuración y Análisis de Código:** Es útil para interpretar direcciones de memoria y otros datos en programación.
5. **Eficiencia en Almacenamiento:** Facilita la representación eficiente de datos.

¿Cómo Convertir Entre Bases?

De Decimal (Base 10) a Otras Bases

▼ **A Binario (Base 2):** 45

1. Divide el número decimal por 2.
2. Anota el residuo.
3. Divide el cociente entre 2 y repite hasta que el cociente sea 0.
4. El número binario es la secuencia de residuos leída en orden inverso.

Ejemplo: Convertir 45 a binario:

$$45_{10} = 101101_2$$

▼ **A Octal (Base 8):** 345

1. Divide el número decimal por 8.
2. Anota el residuo.
3. Divide el cociente entre 8 y repite hasta que el cociente sea 0.
4. El número octal es la secuencia de residuos leída en orden inverso.

Ejemplo: Convertir 345 a octal:

$$345_{10} = 531_8$$

▼ **A Hexadecimal (Base 16):** 255

1. Divide el número decimal por 16.
2. Anota el residuo (usa A-F para valores mayores a 9).
3. Divide el cociente entre 16 y repite hasta que el cociente sea 0.
4. El número hexadecimal es la secuencia de residuos leída en orden inverso.

Ejemplo: Convertir 255 a hexadecimal:

$$255_{10} = FF_{16}$$

De Binario (Base 2) a Otras Bases

▼ **A Decimal (Base 10):** 101101

1. Multiplica cada dígito binario por 2^n (n es la posición del dígito, comenzando desde 0).
2. Suma los resultados.

Ejemplo: Convertir 101101 a decimal:

$$(1 * 2^5 + 0 * 2^4 + 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0)$$

$$32 + 8 + 4 + 1 = 45$$

$$101101_2 = 45_{10}$$

▼ **A Octal (Base 8):** 101101

1. Agrupa los dígitos binarios en grupos de tres, comenzando desde la derecha.
2. En caso de no poder formar grupos de tres, rellenar la izquierda del grupo con ceros hasta completarlo
3. Convierte cada grupo a su equivalente octal.

Ejemplo: Convertir 101101 a octal:

$$101|101 = 5|5$$

$$101101_2 = 55_8$$

▼ **A Hexadecimal (Base 16):** 101101

1. Agrupa los dígitos binarios en grupos de cuatro, comenzando desde la derecha.
2. En caso de no poder formar grupos de cuatro, rellenar la izquierda del grupo con ceros hasta completarlo
3. Convierte cada grupo a su equivalente hexadecimal.

Ejemplo: Convertir 101101 a hexadecimal:

$$0010|1101 = 2|D$$

$$101101_2 = 2D_{16}$$

De Octal (Base 8) a Otras Bases

▼ **A Decimal (Base 10):** 531

1. Multiplica cada dígito octal por 8^n (n es la posición del dígito).
2. Suma los resultados.

Ejemplo: Convertir 531 a decimal:

$$(5 * 8^2 + 3 * 8^1 + 1 * 8^0)$$

$$320 + 24 + 1 = 345$$

$$531_8 = 345_{10}$$

▼ **A Binario (Base 2):** 531

1. Convierte cada dígito octal a su equivalente binario de 3 bits.

Ejemplo: Convertir 531 a binario:

$$5|3|1 = 101|011|001$$

$$531_8 = 101011001_2$$

▼ **A Hexadecimal (Base 16):** 531

1. Convierte el octal a binario.
2. Agrupa los bits binarios en grupos de cuatro para convertir a hexadecimal.

Ejemplo: Convertir 531 a hexadecimal:

$$5|3|1 = 101|011|001 = 101011001$$

$$0001|0101|1001 = 1|5|9$$

$$531_8 = 159_{16}$$

De Hexadecimal (Base 16) a Otras Bases

▼ **A Decimal (Base 10):** 2D

1. Multiplica cada dígito hexadecimal por 16^n (n es la posición del dígito).
2. Suma los resultados.

Ejemplo: Convertir 2D a decimal:

$$(2 * 16^1 + 13 * 16^0)$$

$$32 + 13 = 45$$

$$2D_{16} = 45_{10}$$

▼ **A Binario (Base 2):** 2D

1. Convierte cada dígito hexadecimal a su equivalente binario de 4 bits.

Ejemplo: Convertir 2D a binario:

$$2|D = 0010|1101$$

$$2D_{16} = 00101101_2$$

▼ **A Octal (Base 8):** 2D

1. Convierte el hexadecimal a binario.
2. Agrupa los bits binarios en grupos de tres para convertir a octal.

Ejemplo: Convertir 2D a octal:

$$2|D = 0010|1101 = 00101101$$

$$000|101|101 = 055 = 55$$

$$2D_{16} = 55_8$$

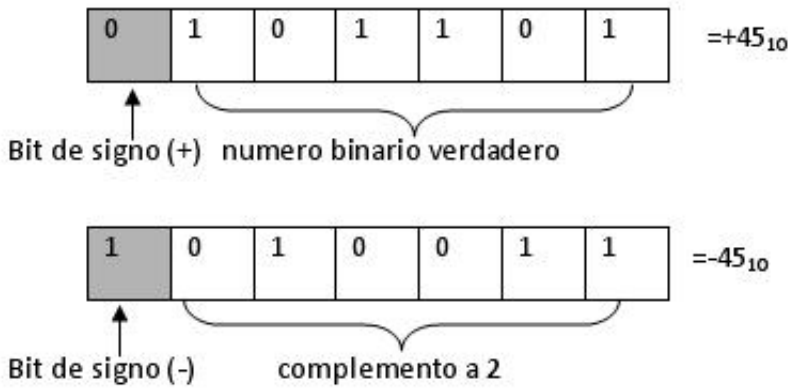
Tabla Comparativa de Números en Diferentes Bases

Decimal (Base 10)	Binario (Base 2)	Octal (Base 8)	Hexadecimal (Base 16)
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6

Decimal (Base 10)	Binario (Base 2)	Octal (Base 8)	Hexadecimal (Base 16)
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Magnitud Verdadera

Consiste en la representación del número binario tal cual es; es decir, se realiza la conversión por el método conocido. Los dígitos binarios se colocan, dentro del registro, justificados a la derecha, el signo se coloca justificado a la izquierda, si hay casillas intermedias vacías, se llenan con cero.



Complemento a uno

El **complemento a uno** es una operación que se realiza en números binarios para invertir sus bits. Es útil en la representación de números negativos en algunas arquitecturas de sistemas digitales.

- Donde hay un `0`, lo cambias a `1`.

- Donde hay un `1`, lo cambias a `0`.
- El bit signo no se modifica

Ejemplo: Complemento a Uno

Supongamos que queremos encontrar el complemento a uno del número binario 10100110:

1. **Número original (10100110):**
   - Es un número binario de 8 bits.
2. **Invertir los bits:**
   - 1 → 0
   - 0 → 1

$$10100110 \rightarrow 11011001$$

Entonces, el complemento a uno de 10100110 es 11011001.

Complemento a dos

El **complemento a dos** es un método comúnmente utilizado en informática para representar números enteros negativos en formato binario. Es preferido sobre el complemento a uno porque elimina la ambigüedad del cero y simplifica las operaciones aritméticas.

Pasos para Calcular el Complemento a Dos

1. Representa el número en binario.
2. Encuentra el complemento a uno: Invierte todos los bits (convierte 0 en 1 y 1 en 0).
3. Suma 1 al resultado del complemento a uno.

Ejemplo: Complemento a Dos

Supongamos que queremos encontrar el complemento a dos del número binario `-101` (que es -5 en decimal) en un sistema de 4 bits:

1. **Número original (1101):**

- Este es el número en binario que queremos convertir a su complemento a dos.

2. **Encuentra el complemento a uno:**
   - Invertimos los bits de 1101.

$$1101 \rightarrow 1010$$

3. **Suma 1 al resultado:**
   - Ahora sumamos 1 al complemento a uno para obtener el complemento a dos.

$$1010 + 0001 = 1011$$

Entonces, el complemento a dos de 1101 (que es -5) es 1011, que representa −5 en un sistema de 4 bits.

Sumas en la ALU

Las computadoras actuales utilizan el Convenio de complemento a dos para representar las cantidades numéricas, el cual consiste en:

- Si la cantidad es positiva, se representa en magnitud verdadera.
- Si la cantidad es negativa, se representa en complemento a dos.

https://www.youtube.com/watch?v=uLulA91jt_A

Ejemplos

▼ **8+5** utilizando `7` bits

- Convertir los numeros en base decimal, a binario.

$$8 = 0001000$$

$$5 = 0000101$$

- Realizar la suma

$$8 + 5 = 0001101$$

▼ **14 + -3** utilizando `7` bits

- Convertir los numeros en base decimal, a binario.

$$14 = 0001110$$

$$-3 = 1000011$$

- En caso de tener numeros negativos aplicar complemento a uno y luego complemento a dos
  - Aplicar complemento a uno

$$-3 = 1000011 = 1111100$$

  - Aplicar complemento a dos

$$-3 \rightarrow 1000011 \rightarrow 1111100 + 1 \rightarrow 1111101$$

- Realizar la suma

$$0001110 + 1111101 = 10001011$$

- En caso la suma supere los bits indicados solo tomar en cuenta de derecha a izquierda la cantidad indicada, ignorando el resto.

$$10001011_{(8bits)} \rightarrow 0001011_{(7bits)}$$

Para comprobar la respuesta, se debe convertir a base 10 utilizando el **método** mencionado anteriormente en esta guía. En caso de que la respuesta sea negativa, se puede realizar la conversión a base 10 excluyendo el bit de signo. Luego, para obtener el valor real en base 10, se debe aplicar el proceso de complemento a uno y complemento a dos a la respuesta original en binario.

Videos Complementarios

https://www.youtube.com/watch?v=hIs3A6gGz2w

https://www.youtube.com/shorts/iQX3MvuaJ1c