

Representación de la Información

(Parte 2)

David Alejandro González Márquez

Clase disponible en: <https://github.com/fokerman/computingSystemsCourse>

Complemento a 2: Operaciones Básicas en binario

Si bien las operaciones se pueden realizar sobre cualquier notación de las vistas, vamos a hacer foco en **Complemento a 2**.

En Complemento a 2 podemos hacer cualquier operación básica de sumas y restas **directamente** usando la representación del número y sin hacer cuentas adicionales.

- Negado (NEG)
- Suma (+)
- Resta (-)
- Extensión de Signo
- Shifts, multiplicación y división por potencias de 2 (>> y <<)

Complemento a 2: Negado

Cómo obtener el inverso aditivo de un número en complemento a 2.

- ① Invertir bit a bit el número (NOT)
- ② Sumar uno al número (+1)

Complemento a 2: Negado

Cómo obtener el inverso aditivo de un número en complemento a 2.

- ① Invertir bit a bit el número (NOT)
- ② Sumar uno al número (+1)

0 0 0 1 0 1 0 1 → 21

Ejemplo:

Inverso aditivo de 21.

Complemento a 2: Negado

Cómo obtener el inverso aditivo de un número en complemento a 2.

- ① Invertir bit a bit el número (NOT)
- ② Sumar uno al número (+1)

$$\begin{array}{r} \text{NOT } 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1 \rightarrow 21 \\ \hline 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0 \end{array}$$

Ejemplo:

Inverso aditivo de 21.

Complemento a 2: Negado

Cómo obtener el inverso aditivo de un número en complemento a 2.

- ① Invertir bit a bit el número (NOT)
- ② Sumar uno al número (+1)

$$\begin{array}{r} \text{NOT } 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1 \rightarrow 21 \\ + \\ \hline 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0 \\ \\ \hline 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \end{array}$$

Ejemplo:

Inverso aditivo de 21.

Complemento a 2: Negado

Cómo obtener el inverso aditivo de un número en complemento a 2.

- ① Invertir bit a bit el número (NOT)
- ② Sumar uno al número (+1)

$$\begin{array}{r} \text{NOT } \boxed{0\ 0\ 0\ 1\ 0\ 1\ 0\ 1} \rightarrow 21 \\ + \quad \boxed{1\ 1\ 1\ 0\ 1\ 0\ 1\ 0} \\ \hline \boxed{1\ 1\ 1\ 0\ 1\ 0\ 1\ 1} \rightarrow -21 \end{array}$$

Ejemplo:

Inverso aditivo de 21.

Complemento a 2: Negado

Cómo obtener el inverso aditivo de un número en complemento a 2.

- ① Invertir bit a bit el número (NOT)
- ② Sumar uno al número (+1)

$$\begin{array}{r} \text{NOT } \boxed{0\ 0\ 0\ 1\ 0\ 1\ 0\ 1} \rightarrow 21 \\ + \quad \boxed{\overline{1\ 1\ 1\ 0\ 1\ 0\ 1\ 0}} \\ \hline \boxed{1\ 1\ 1\ 0\ 1\ 0\ 1\ 1} \rightarrow -21 \end{array}$$

Ejemplo:

Inverso aditivo de 21.

1 1 1 0 1 0 1 1

Inverso aditivo de –21.

Complemento a 2: Negado

Cómo obtener el inverso aditivo de un número en complemento a 2.

- ① Invertir bit a bit el número (NOT)
- ② Sumar uno al número (+1)

Ejemplo:

Inverso aditivo de 21.

Inverso aditivo de -21 .

$$\begin{array}{r} \text{NOT } 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1 \rightarrow 21 \\ + \\ \hline 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0 \\ \\ \hline 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \rightarrow -21 \end{array}$$

$$\begin{array}{r} \text{NOT } 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\ \hline 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0 \end{array}$$

Complemento a 2: Negado

Cómo obtener el inverso aditivo de un número en complemento a 2.

- ① Invertir bit a bit el número (NOT)
- ② Sumar uno al número (+1)

Ejemplo:

Inverso aditivo de 21.

Inverso aditivo de -21 .

$$\begin{array}{r} \text{NOT } 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1 \rightarrow 21 \\ + \\ \hline 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0 \\ \\ \hline 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \rightarrow -21 \end{array}$$

$$\begin{array}{r} \text{NOT } 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1 \\ + \\ \hline 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0 \\ \\ \hline 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1 \rightarrow 21 \end{array}$$

Complemento a 2: Negado

Ejemplos:

0 1 1 1 1 1 1 1 → 127

Complemento a 2: Negado

Ejemplos:

$$\begin{array}{r} \text{NOT } 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ + \quad \underline{\quad\quad\quad\quad\quad\quad\quad\quad\quad} \\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ \hline \quad \quad \quad \quad \quad \quad \quad \quad \quad 1 \\ \hline 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \end{array} \rightarrow -127$$

En 8 bits el máximo número representable en complemento a 2 es 127 y el más chico es -128. El rango es asimétrico y por lo tanto no es posible representar el 128 positivo.

Complemento a 2: Negado

Ejemplos:

$$\begin{array}{r}
 \text{NOT } \boxed{0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1} \rightarrow 127 \\
 + \quad \boxed{1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0} \\
 \hline
 \boxed{1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1} \rightarrow -127
 \end{array}$$

$$\begin{array}{r}
 \text{NOT } 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\
 + \quad \underline{0\ 1\ 1\ 1\ 1\ 1\ 1\ 0} \\
 \hline
 \quad \quad \quad \boxed{1} \\
 \hline
 \quad \quad \quad 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \rightarrow 127
 \end{array}$$

En 8 bits el máximo número representable en complemento a 2 es 127 y el más chico es -128. El rango es asimétrico y por lo tanto no es posible representar el 128 positivo.

Complemento a 2: Negado

Ejemplos:

$$\begin{array}{r} \text{NOT } 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \rightarrow 127 \\ + \ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ \hline 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \rightarrow -127 \end{array}$$

$$\begin{array}{r} \text{NOT } 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \rightarrow 1 \\ + \ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0 \\ \hline 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \rightarrow -1 \end{array}$$

$$\begin{array}{r} \text{NOT } 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\ + \ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0 \\ \hline 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \rightarrow 127 \end{array}$$

$$\begin{array}{r} \text{NOT } 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ + \ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ \hline 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \rightarrow 1 \end{array}$$

En 8 bits el máximo número representable en complemento a 2 es 127 y el más chico es -128. El rango es asimétrico y por lo tanto no es posible representar el 128 positivo.

Complemento a 2: Negado

Ejemplos:

$$\begin{array}{r} \text{NOT } 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \rightarrow 127 \\ + \quad 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ \hline 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \rightarrow -127 \end{array}$$

$$\begin{array}{r} \text{NOT } 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \rightarrow 1 \\ + \quad 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0 \\ \hline 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \rightarrow -1 \end{array}$$

$$0\ 0\ 1\ 0\ 0\ 0\ 1\ 1 \rightarrow 35$$

$$\begin{array}{r} \text{NOT } 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\ + \quad 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0 \\ \hline 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \rightarrow 127 \end{array}$$

$$\begin{array}{r} \text{NOT } 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ + \quad 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ \hline 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \rightarrow 1 \end{array}$$

En 8 bits el máximo número representable en complemento a 2 es 127 y el más chico es -128. El rango es asimétrico y por lo tanto no es posible representar el 128 positivo.

Complemento a 2: Negado

Ejemplos:

$$\begin{array}{r} \text{NOT } 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \rightarrow 127 \\ + \ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ \hline 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \rightarrow -127 \end{array}$$

$$\begin{array}{r} \text{NOT } 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \rightarrow 1 \\ + \ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0 \\ \hline 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \rightarrow -1 \end{array}$$

$$\begin{array}{r} \text{NOT } 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1 \rightarrow 35 \\ + \ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0 \\ \hline 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1 \rightarrow -35 \end{array}$$

$$\begin{array}{r} \text{NOT } 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \\ + \ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0 \\ \hline 0\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \rightarrow 127 \end{array}$$

$$\begin{array}{r} \text{NOT } 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ + \ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ \hline 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \rightarrow 1 \end{array}$$

$$\begin{array}{r} \text{NOT } 1\ 1\ 0\ 1\ 1\ 1\ 0\ 1 \\ + \ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0 \\ \hline 0\ 0\ 1\ 0\ 0\ 0\ 1\ 1 \rightarrow 35 \end{array}$$

En 8 bits el máximo número representable en complemento a 2 es 127 y el más chico es -128. El rango es asimétrico y por lo tanto no es posible representar el 128 positivo.

Complemento a 2: Suma

Ejemplo de suma de dos números.

$$\begin{array}{r} + 39 \\ - 21 \\ \hline 60 \end{array}$$

$$\begin{array}{r} + 00100111 \\ - 00010101 \\ \hline \end{array}$$

$$\begin{array}{r} + 27 \\ - 15 \\ \hline \end{array}$$

Complemento a 2: Suma

Ejemplo de suma de dos números.

$$\begin{array}{r} + \quad 39 \\ + \quad 21 \\ \hline 60 \end{array} \qquad \begin{array}{r} + \quad 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \\ + \quad 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \\ \hline \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad 1 \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad 0 \end{array} \qquad \begin{array}{r} + \quad 27 \\ + \quad 15 \\ \hline \end{array}$$

Complemento a 2: Suma

Ejemplo de suma de dos números.

$$\begin{array}{r} + \quad 39 \\ + \quad 21 \\ \hline 60 \end{array} \qquad \begin{array}{r} + \quad 00100111 \\ + \quad 00010101 \\ \hline \quad 00 \end{array} \qquad \begin{array}{r} + \quad 27 \\ + \quad 15 \\ \hline \end{array}$$

Complemento a 2: Suma

Ejemplo de suma de dos números.

$$\begin{array}{r} + \quad 39 \\ + \quad 21 \\ \hline 60 \end{array} \qquad \begin{array}{r} + \quad 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \\ + \quad 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \\ \hline 1 \ 0 \ 0 \end{array} \qquad \begin{array}{r} + \quad 27 \\ + \quad 15 \\ \hline \end{array}$$

Complemento a 2: Suma

Ejemplo de suma de dos números.

$$\begin{array}{r} + \quad 39 \\ + \quad 21 \\ \hline 60 \end{array} \qquad \begin{array}{r} + \quad 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \\ + \quad 0 \ 0 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1 \\ \hline 1 \ 1 \ 0 \ 0 \end{array} \qquad \begin{array}{r} + \quad 27 \\ + \quad 15 \\ \hline C \end{array}$$

Complemento a 2: Suma

Ejemplo de suma de dos números.

$$\begin{array}{r} + \quad 39 \\ + \quad 21 \\ \hline 60 \end{array} \qquad \begin{array}{r} + \quad 00100111 \\ + \quad 00010101 \\ \hline 11100 \end{array} \qquad \begin{array}{r} + \quad 27 \\ + \quad 15 \\ \hline C \end{array}$$

Complemento a 2: Suma

Ejemplo de suma de dos números.

$$\begin{array}{r} + \quad 39 \\ + \quad 21 \\ \hline 60 \end{array} \qquad \begin{array}{r} + \quad 00100111 \\ + \quad 00010101 \\ \hline 111100 \end{array} \qquad \begin{array}{r} + \quad 27 \\ + \quad 15 \\ \hline C \end{array}$$

Complemento a 2: Suma

Ejemplo de suma de dos números.

$$\begin{array}{r} + \quad 39 \\ + \quad 21 \\ \hline 60 \end{array} \qquad \begin{array}{r} + \quad 00100111 \\ + \quad 00010101 \\ \hline 01111000 \end{array} \qquad \begin{array}{r} + \quad 27 \\ + \quad 15 \\ \hline C \end{array}$$

Complemento a 2: Suma

Ejemplo de suma de dos números.

$$\begin{array}{r} + \quad 39 \\ + \quad 21 \\ \hline 60 \end{array} \qquad \begin{array}{r} + \quad 00100111 \\ + \quad 00010101 \\ \hline 00111100 \end{array} \qquad \begin{array}{r} + \quad 27 \\ + \quad 15 \\ \hline 3C \end{array}$$

Complemento a 2: Resta

Ejemplo de resta de dos números.

$$\begin{array}{r} 39 \\ - 21 \\ \hline ?? \end{array}$$

$$\begin{array}{r} 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1 \\ - 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline ?\ ?\ ?\ ?\ ?\ ?\ ?\ ? \end{array}$$

$$\begin{array}{r} 27 \\ - 15 \\ \hline ?? \end{array}$$

Complemento a 2: Resta

Ejemplo de resta de dos números.

Complemento a 2: Resta

Ejemplo de resta de dos números.

$$\begin{array}{r} + \quad 39 \\ - 21 \\ \hline 18 \end{array}$$

$$\begin{array}{r} + \quad 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \\ - 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1 \\ \hline \end{array}$$

$$\begin{array}{r} + \quad 27 \\ - \text{EB} \\ \hline \end{array}$$

Complemento a 2: Resta

Ejemplo de resta de dos números.

$$\begin{array}{r} + \quad 39 \\ - 21 \\ \hline 18 \end{array} \qquad \begin{array}{r} + \quad 00100111 \\ 11101011 \\ \hline 0 \end{array} \qquad \begin{array}{r} + \quad 27 \\ \text{EB} \\ \hline \end{array}$$

Complemento a 2: Resta

Ejemplo de resta de dos números.

$$\begin{array}{r} + \quad 39 \\ - 21 \\ \hline 18 \end{array} \qquad \begin{array}{r} + \quad 00100111 \\ 11101011 \\ \hline 10 \end{array} \qquad \begin{array}{r} + \quad 27 \\ \text{EB} \\ \hline \end{array}$$

Complemento a 2: Resta

Ejemplo de resta de dos números.

$$\begin{array}{r} + \quad 39 \\ - 21 \\ \hline 18 \end{array} \qquad \begin{array}{r} + \quad 00100111 \\ 11101011 \\ \hline 010 \end{array} \qquad \begin{array}{r} + \quad 27 \\ \text{EB} \\ \hline \end{array}$$

Complemento a 2: Resta

Ejemplo de resta de dos números.

$$\begin{array}{r} + \\ 39 \\ - 21 \\ \hline 18 \end{array}$$

$$\begin{array}{r} + \\ \begin{array}{ccccccccc} & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ + & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ \hline & 0 & 0 & 1 & 0 \end{array} \end{array}$$

$$\begin{array}{r} + \\ 27 \\ - \text{EB} \\ \hline 2 \end{array}$$

Complemento a 2: Resta

Ejemplo de resta de dos números.

$$\begin{array}{r} + \quad 39 \\ - 21 \\ \hline 18 \end{array} \qquad \begin{array}{r} + \quad 00100111 \\ 11101011 \\ \hline 10010 \end{array} \qquad \begin{array}{r} + \quad 27 \\ EB \\ \hline 2 \end{array}$$

Complemento a 2: Resta

Ejemplo de resta de dos números.

$$\begin{array}{r} + \\ 39 \\ - 21 \\ \hline 18 \end{array}$$

$$\begin{array}{r} + \\ \begin{array}{ccccccccc} & 1 & & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ \hline 0 & 1 & 0 & 0 & 1 & 0 \end{array} \end{array}$$

$$\begin{array}{r} + \\ 27 \\ \hline \text{EB} \\ 2 \end{array}$$

Complemento a 2: Resta

Ejemplo de resta de dos números.

$$\begin{array}{r} + \\ 39 \\ - 21 \\ \hline 18 \end{array}$$

$$\begin{array}{r} + \\ \begin{array}{ccccccccc} 1 & 1 & & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ \hline 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array} \end{array}$$

$$\begin{array}{r} + \\ 27 \\ \hline \text{EB} \\ 2 \end{array}$$

Complemento a 2: Resta

Ejemplo de resta de dos números.

$$\begin{array}{r} + \\ 39 \\ - 21 \\ \hline 18 \end{array}$$

$$\begin{array}{r} + \\ \begin{array}{ccccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array} \end{array}$$

$$\begin{array}{r} + \\ 27 \\ \hline \text{EB} \\ 12 \end{array}$$

Complemento a 2: Resta

Ejemplo de resta de dos números.

$$\begin{array}{r} + \quad 39 \\ - 21 \\ \hline 18 \end{array} \qquad \begin{array}{r} + \quad 1111111 \\ 00100111 \\ \hline 11101011 \\ 00010010 \end{array} \qquad \begin{array}{r} + \quad 27 \\ EB \\ \hline 12 \end{array}$$

The middle column shows a binary subtraction operation. A circled '1' at the top left is labeled 'carry' with an orange arrow pointing to it. The result of the subtraction is 00010010. The rightmost column shows a subtraction of 27 from EB, resulting in 12.

El carry de la operación no se considera como parte del número resultado.

Complemento a 2: Resta

Ejemplo de resta de dos números.

The image displays three binary subtraction examples. The first example shows $39 - 21 = 18$ in binary. The second example shows $11111111 - 00100011 = 11010110$. The third example shows $27 - EB = 12$. Annotations include a 'carry' arrow pointing from the top of the first column to the second column in the first example, and a 'mismo signo' (same sign) bracket under the first two examples pointing to the sign bit of the result.

$$\begin{array}{r} 39 \\ + -21 \\ \hline 18 \end{array}$$
$$\begin{array}{r} 11111111 \\ + 00100011 \\ \hline 11010110 \end{array}$$
$$\begin{array}{r} 27 \\ - EB \\ \hline 12 \end{array}$$

El carry de la operación no se considera como parte del número resultado.
El resultado es correcto, ya que tienen el mismo bit de signo.

Complemento a 2: Suma Overflow

Ejemplo de suma de dos números, donde su resultado no es representable en 8 bits.

$$\begin{array}{r} + \\ 78 \\ \hline 78 \end{array}$$

$$\begin{array}{r} + \\ 01001110 \\ \hline 01001110 \end{array}$$

$$\begin{array}{r} + \\ 4E \\ \hline 4E \end{array}$$

Complemento a 2: Suma Overflow

Ejemplo de suma de dos números, donde su resultado no es representable en 8 bits.

$$\begin{array}{r} + \quad 78 \\ + \quad 78 \\ \hline 156 \end{array} \qquad \begin{array}{r} + \quad 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \\ + \quad 0 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \\ \hline 1 \ 1 \ 0 \ 0 \end{array} \qquad \begin{array}{r} + \quad 4E \\ + \quad 4E \\ \hline C \end{array}$$

Complemento a 2: Suma Overflow

Ejemplo de suma de dos números, donde su resultado no es representable en 8 bits.

$$\begin{array}{r} + \quad \quad \quad 0 \quad 1 \quad \quad \quad 1 \quad 1 \quad 1 \\ 78 \quad + \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \\ 78 \quad \quad \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \\ \hline 156 \quad \quad \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \end{array}$$
$$\begin{array}{r} + \quad \quad \quad 1 \\ 4E \quad + \quad 4E \\ \hline 9C \end{array}$$

Complemento a 2: Suma Overflow

Ejemplo de suma de dos números, donde su resultado no es representable en 8 bits.

$$\begin{array}{r} & \begin{array}{r} 0 & 1 & 1 & 1 & 1 \\ + & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ \hline & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{array} \\ + & \begin{array}{r} 78 \\ 78 \\ \hline +156 \end{array} \end{array}$$

diferente signo

$$\begin{array}{r} & \begin{array}{r} 1 \\ 4E \\ 4E \\ \hline 9C \end{array} \end{array}$$

Complemento a 2: Suma Overflow

Ejemplo de suma de dos números, donde su resultado no es representable en 8 bits.

The diagram shows three additions:

- $78 + 78 = 156$: The sum is 156, which is represented as 10011100 in binary. A circled '0' and a circled '1' above the first two columns are connected by a line labeled "carry's diferentes" (different carries). A circled '1' below the third column is connected by a line labeled "diferente signo" (different sign) to the circled '1' in the result.
- $01001110 + 01001110 = 10011100$: The sum is 10011100 in binary.
- $4E + 4E = 9C$: The sum is 9C in binary.

Los últimos dos carry de la operación no coinciden → **Overflow**.

Complemento a 2: Suma Overflow

Ejemplo de suma de dos números, donde su resultado no es representable en 8 bits.

The diagram shows three additions:

- $78 + 78 = 156$ (in decimal)
- $010001110 + 010001110 = 100011100$ (in binary)
- $4E + 4E = 9C$ (in hexadecimal)

Annotations highlight carry issues:

- A bracket labeled "diferente signo" (different sign) points from the first addition's result to the second addition's result.
- Two circles labeled "0" and "1" are connected by a line labeled "carry's diferentes" (different carries) above the second addition.
- A red line through the second addition's result indicates an error.
- A red line through the third addition's result indicates an error.

Los últimos dos carry de la operación no coinciden → **Overflow**.

El resultado es incorrecto, la suma de dos números positivos no puede ser negativa.

Complemento a 2: Suma Overflow

Ejemplo de suma de dos números, donde su resultado no es representable en 8 bits.

$$\begin{array}{r} + \quad -78 \\ -78 \\ \hline -156 \end{array} \qquad \begin{array}{r} + \quad 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0 \\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0 \\ \hline \end{array} \qquad \begin{array}{r} + \quad \text{B2} \\ \text{B2} \\ \hline \end{array}$$

Complemento a 2: Suma Overflow

Ejemplo de suma de dos números, donde su resultado no es representable en 8 bits.

$$\begin{array}{r} + \quad -78 \\ -78 \\ \hline -156 \end{array} \qquad \begin{array}{r} + \quad 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0 \\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0 \\ \hline 0\ 1\ 0\ 0 \end{array} \qquad \begin{array}{r} + \quad \text{B2} \\ \text{B2} \\ \hline 4 \end{array}$$

Complemento a 2: Suma Overflow

Ejemplo de suma de dos números, donde su resultado no es representable en 8 bits.

$$\begin{array}{r} \begin{array}{r} & 1 & 0 & 1 & 1 & 1 \\ + & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ -78 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ \hline -156 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} & \begin{array}{r} 1 \\ B2 \\ B2 \\ \hline 64 \end{array} \end{array}$$

Complemento a 2: Suma Overflow

Ejemplo de suma de dos números, donde su resultado no es representable en 8 bits.

$$\begin{array}{r} \begin{array}{r} & 1 & 0 & 1 & 1 & 1 \\ + & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ -78 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ \hline -156 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \\ \text{diferente signo} \end{array}$$
$$\begin{array}{r} \begin{array}{r} & 1 \\ + & B2 \\ B2 \\ \hline 64 \end{array} \end{array}$$

Complemento a 2: Suma Overflow

Ejemplo de suma de dos números, donde su resultado no es representable en 8 bits.

The diagram illustrates the binary addition of two 8-bit numbers, both representing -78 (B2). The addition is performed as follows:

$$\begin{array}{r} -78 \\ + -78 \\ \hline -156 \end{array}$$

The result, -156, is shown in binary as 101100100. A circled '0' at the beginning indicates it is a negative number. An annotation 'diferente signo' (different sign) points to this circled '0'. Another annotation 'carrys diferentes' (different carries) points to the two carry bits at the top of the column, which are both 1s (circled), while the sum bit is 0 (circled).

+ B2
B2
64

Los últimos dos carry de la operación no coinciden → **Overflow**.

Complemento a 2: Suma Overflow

Ejemplo de suma de dos números, donde su resultado no es representable en 8 bits.

The diagram illustrates the binary addition of two 8-bit numbers, both representing -78 (B2). The numbers are:

$$\begin{array}{r} -78 \\ + -78 \\ \hline -156 \end{array}$$

The result, -156, is shown in red. A circled '0' at the top left indicates the sign bit. A circled '1' at the top right indicates the carry bit. A circled '0' at the bottom left indicates the result's sign. A circled '0' at the bottom right indicates the result's value. An orange bracket labeled 'diferente signo' connects the circled sign bits. An orange bracket labeled 'carrys diferentes' connects the circled carry bits. A red horizontal line through the bottom row of digits highlights the error.

+ + +

-78 1 0 1 1 1 0 0 1 0
-78 1 0 1 1 0 0 1 0

-156 0 1 1 0 0 1 0 0

diferente signo carries diferentes

+ + +

B2 1 B2
B2 B2
64

Los últimos dos carry de la operación no coinciden → **Overflow**.

El resultado es incorrecto, la suma de dos números negativos no puede ser positiva.

Complemento a 2: Extensión de signo

Para aumentar la cantidad de bits de un número en complemento a 2 es simple.

Se debe **copiar** el bit más significativo del número tantas veces como sea necesario.
A esta acción la llamamos extensión de signo.

Ejemplos:

$$45d = 00101101b =$$

$$-18d = 11101110b =$$

$$-1d = 1111b =$$

Complemento a 2: Extensión de signo

Para aumentar la cantidad de bits de un número en complemento a 2 es simple.

Se debe **copiar** el bit más significativo del número tantas veces como sea necesario.
A esta acción la llamamos extensión de signo.

Ejemplos:

$$45d = 00101101b = \textcolor{orange}{0}0101101b$$

$$-18d = 11101110b =$$

$$-1d = 1111b =$$

Complemento a 2: Extensión de signo

Para aumentar la cantidad de bits de un número en complemento a 2 es simple.

Se debe **copiar** el bit más significativo del número tantas veces como sea necesario.
A esta acción la llamamos extensión de signo.

Ejemplos:

$$45d = 00101101b = \textcolor{teal}{00000000} \textcolor{orange}{0}0101101b$$

$$-18d = 11101110b =$$

$$-1d = 1111b =$$

Complemento a 2: Extensión de signo

Para aumentar la cantidad de bits de un número en complemento a 2 es simple.

Se debe **copiar** el bit más significativo del número tantas veces como sea necesario.
A esta acción la llamamos extensión de signo.

Ejemplos:

$$45d = 00101101b = \textcolor{teal}{00000000} \textcolor{orange}{0}0101101b$$

$$-18d = 11101110b = \textcolor{orange}{1}1101110b$$

$$-1d = 1111b =$$

Complemento a 2: Extensión de signo

Para aumentar la cantidad de bits de un número en complemento a 2 es simple.

Se debe **copiar** el bit más significativo del número tantas veces como sea necesario.
A esta acción la llamamos extensión de signo.

Ejemplos:

$$45d = 00101101b = \textcolor{teal}{00000000} \textcolor{orange}{0}0101101b$$

$$-18d = 11101110b = \textcolor{teal}{11111111} \textcolor{orange}{1}1101110b$$

$$-1d = 1111b =$$

Complemento a 2: Extensión de signo

Para aumentar la cantidad de bits de un número en complemento a 2 es simple.

Se debe **copiar** el bit más significativo del número tantas veces como sea necesario.
A esta acción la llamamos extensión de signo.

Ejemplos:

$$45d = 00101101b = \textcolor{teal}{00000000} \textcolor{orange}{0}0101101b$$

$$-18d = 11101110b = \textcolor{teal}{11111111} \textcolor{orange}{1}1101110b$$

$$-1d = 1111b = \textcolor{orange}{1111b}$$

Complemento a 2: Extensión de signo

Para aumentar la cantidad de bits de un número en complemento a 2 es simple.

Se debe **copiar** el bit más significativo del número tantas veces como sea necesario.
A esta acción la llamamos extensión de signo.

Ejemplos:

$$45d = 00101101b = \textcolor{teal}{00000000} \textcolor{orange}{0}0101101b$$

$$-18d = 11101110b = \textcolor{teal}{11111111} \textcolor{orange}{1}1101110b$$

$$-1d = 1111b = \textcolor{teal}{1111111111111111} \textcolor{orange}{1}111b$$

Complemento a 2: Shifts

Las operaciones de *shift* corresponden a mover bits desde sus posiciones más significativas a las menos significativas, o a la inversa.

Lógico a derecha

Complemento a 2: Shifts

Las operaciones de *shift* corresponden a mover bits desde sus posiciones más significativas a las menos significativas, o a la inversa.

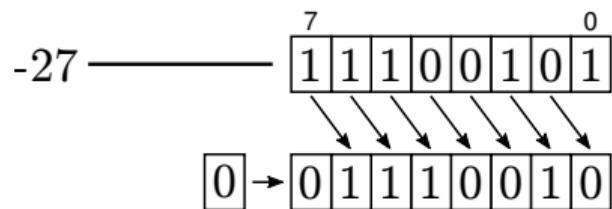
Lógico a derecha

$$-27 \quad \underline{\quad} \quad \begin{array}{|c|c|c|c|c|c|c|c|c|} \hline & 7 & & & & & & 0 \\ \hline 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ \hline \end{array}$$

Complemento a 2: Shifts

Las operaciones de *shift* corresponden a mover bits desde sus posiciones más significativas a las menos significativas, o a la inversa.

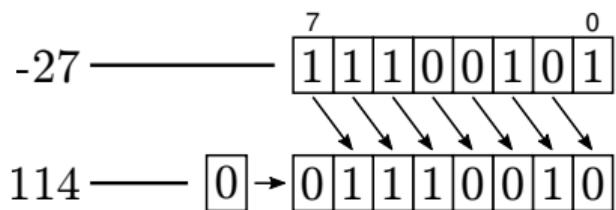
Lógico a derecha



Complemento a 2: Shifts

Las operaciones de *shift* corresponden a mover bits desde sus posiciones más significativas a las menos significativas, o a la inversa.

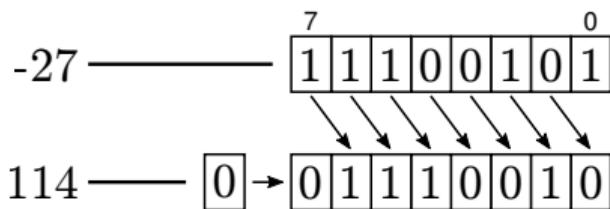
Lógico a derecha



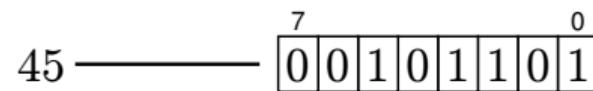
Complemento a 2: Shifts

Las operaciones de *shift* corresponden a mover bits desde sus posiciones más significativas a las menos significativas, o a la inversa.

Lógico a derecha



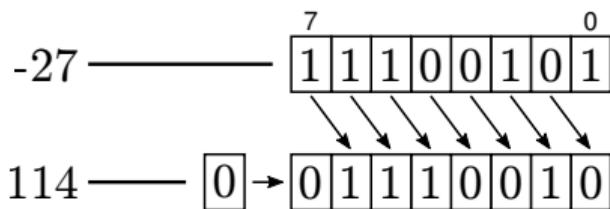
Lógico a izquierda (mul por 2)



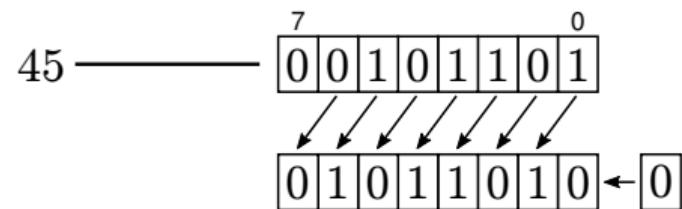
Complemento a 2: Shifts

Las operaciones de *shift* corresponden a mover bits desde sus posiciones más significativas a las menos significativas, o a la inversa.

Lógico a derecha



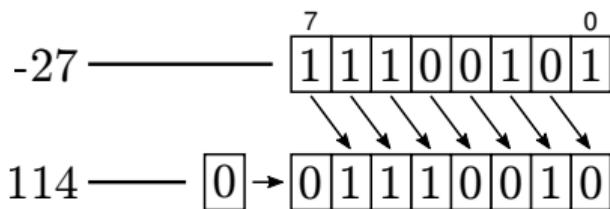
Lógico a izquierda (mul por 2)



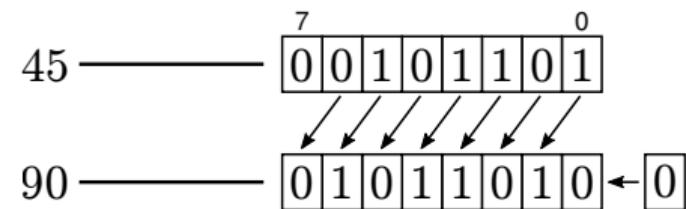
Complemento a 2: Shifts

Las operaciones de *shift* corresponden a mover bits desde sus posiciones más significativas a las menos significativas, o a la inversa.

Lógico a derecha



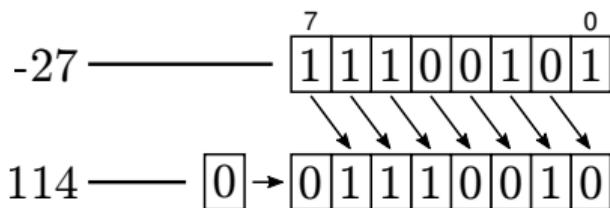
Lógico a izquierda (mul por 2)



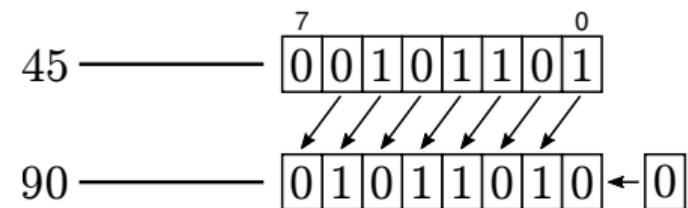
Complemento a 2: Shifts

Las operaciones de *shift* corresponden a mover bits desde sus posiciones más significativas a las menos significativas, o a la inversa.

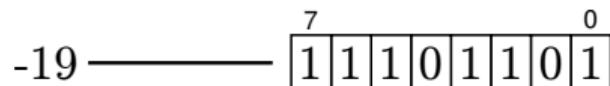
Lógico a derecha



Lógico a izquierda (mul por 2)



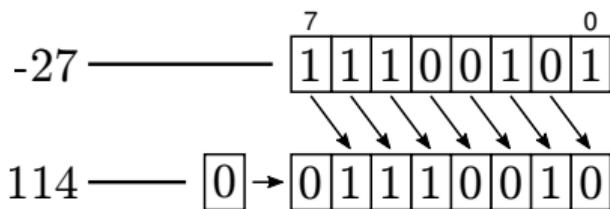
Aritmético a derecha (div por 2)



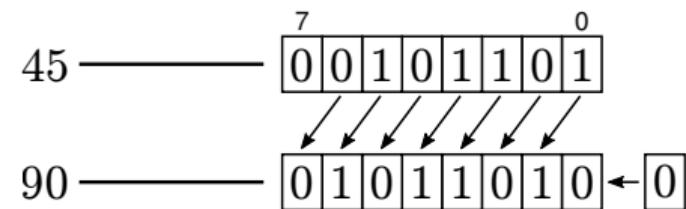
Complemento a 2: Shifts

Las operaciones de *shift* corresponden a mover bits desde sus posiciones más significativas a las menos significativas, o a la inversa.

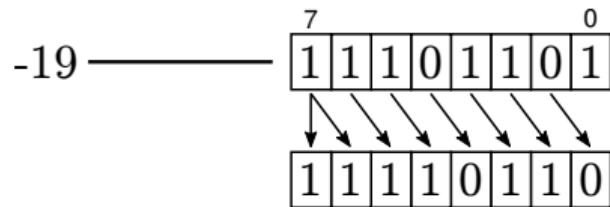
Lógico a derecha



Lógico a izquierda (mul por 2)



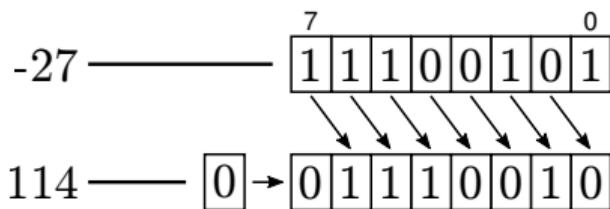
Aritmético a derecha (div por 2)



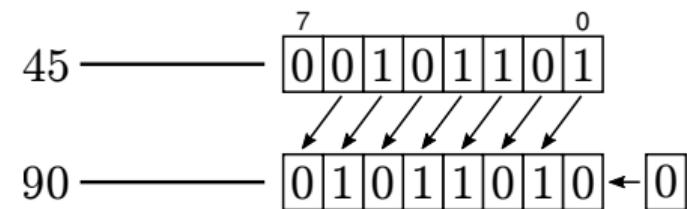
Complemento a 2: Shifts

Las operaciones de *shift* corresponden a mover bits desde sus posiciones más significativas a las menos significativas, o a la inversa.

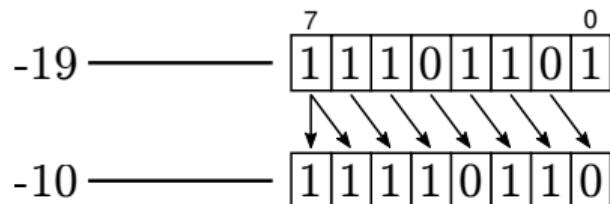
Lógico a derecha



Lógico a izquierda (mul por 2)



Aritmético a derecha (div por 2)



Números Fraccionarios

Existen múltiples formas de codificar números fraccionarios.
Vamos a analizar solo dos de ellas.

Números Fraccionarios

Existen múltiples formas de codificar números fraccionarios.

Vamos a analizar solo dos de ellas.

- **Punto fijo**

La cantidad de bits destinados para la parte entera y fraccionaria del número son fijas.

Ejemplo en decimal:

238,35 (3 dígitos para la parte entera y 2 dígitos para la parte fraccionaria)

Números Fraccionarios

Existen múltiples formas de codificar números fraccionarios.

Vamos a analizar solo dos de ellas.

- **Punto fijo**

La cantidad de bits destinados para la parte entera y fraccionaria del número son fijas.

Ejemplo en decimal:

238,35 (3 dígitos para la parte entera y 2 dígitos para la parte fraccionaria)

- **Flotante**

Se representa un número acotado por una cantidad de bits fija y se usa un factor de escala para calcular su magnitud.

Ejemplo en decimal:

$$0,3\cancel{8}9 \cdot 10^{\cancel{0}2} = 0,389 \cdot 100 = 38,9 \quad (3 \text{ dígitos para la fracción y } 2 \text{ dígitos para el exponente})$$

Convertir desde binario a decimal

Para convertir la parte fraccionaria tenemos que operar con las inversas de las potencias de 2.
Por ejemplo,

$$100100,101_2 \mapsto$$

Convertir desde binario a decimal

Para convertir la parte fraccionaria tenemos que operar con las inversas de las potencias de 2.
Por ejemplo,

$100100,101_2 \mapsto$

6	5	4	3	2	1	0
0	1	0	0	1	0	0

Convertir desde binario a decimal

Para convertir la parte fraccionaria tenemos que operar con las inversas de las potencias de 2.
Por ejemplo,

$100100,101_2 \mapsto$

$$\begin{array}{ccccccc} 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ * & * & * & * & * & * & * \\ 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 \end{array}$$

Convertir desde binario a decimal

Para convertir la parte fraccionaria tenemos que operar con las inversas de las potencias de 2.
Por ejemplo,

$100100,101_2 \mapsto$

$$\begin{array}{cccccccc} 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ * & * & * & * & * & * & * \\ 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 \\ | & | & | & | & | & | & | \\ 64 & 32 & 16 & 8 & 4 & 2 & 1 \end{array}$$

Convertir desde binario a decimal

Para convertir la parte fraccionaria tenemos que operar con las inversas de las potencias de 2.
Por ejemplo,

$100100,101_2 \mapsto$

$$\begin{array}{cccccccccc} & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ * & * & * & * & * & * & * & * \\ 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 + \\ | & | & | & | & | & | & | \\ 64 & 32 & 16 & 8 & 4 & 2 & 1 \end{array}$$

Convertir desde binario a decimal

Para convertir la parte fraccionaria tenemos que operar con las inversas de las potencias de 2.
Por ejemplo,

$100100,101_2 \mapsto$

$$\begin{array}{cccccccc|cccccc} 6 & 5 & 4 & 3 & 2 & 1 & 0 & \dots & -1 & -2 & -3 & -4 & -5 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & | & 1 & 0 & 1 & 0 & 0 \\ * & * & * & * & * & * & * & | & & & & & \\ 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 + & & & & & & & | & & & & & \\ \downarrow & | & & & & & \\ 64 & 32 & 16 & 8 & 4 & 2 & 1 & | & & & & & \end{array}$$

Convertir desde binario a decimal

Para convertir la parte fraccionaria tenemos que operar con las inversas de las potencias de 2.
Por ejemplo,

$100100,101_2 \mapsto$

$$\begin{array}{cccccccc|ccccccccc} 6 & 5 & 4 & 3 & 2 & 1 & 0 & & -1 & -2 & -3 & -4 & -5 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & & 1 & 0 & 1 & 0 & 0 \\ * & * & * & * & * & * & * & & * & * & * & * & * \\ 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^5} = \\ | & | & | & | & | & | & | & | & | & | & | & | & | \\ 64 & 32 & 16 & 8 & 4 & 2 & 1 & 0,5 & 0,25 & 0,125 & 0,0625 & 0,03125 & \\ \end{array}$$

Convertir desde binario a decimal

Para convertir la parte fraccionaria tenemos que operar con las inversas de las potencias de 2.
Por ejemplo,

$100100,101_2 \mapsto$

$$\begin{array}{cccccccc|ccccccccc} 6 & 5 & 4 & 3 & 2 & 1 & 0 & & -1 & -2 & -3 & -4 & -5 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & & 1 & 0 & 1 & 0 & 0 \\ * & * & * & * & * & * & * & & * & * & * & * & * \\ 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 & & & & & & & + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^5} & = 32 + 4 + 0,5 + 0,125 = 36,625 \\ | & | & | & | & | & | & | & & | & | & | & | & | \\ 64 & 32 & 16 & 8 & 4 & 2 & 1 & & 0,5 & 0,25 & 0,125 & 0,0625 & 0,03125 \end{array}$$

Convertir desde binario a decimal

Para convertir la parte fraccionaria tenemos que operar con las inversas de las potencias de 2.
Por ejemplo,

$$100100,101_2 \mapsto 36,625_{10}$$

$$\begin{array}{cccccccc|cccccccc} 6 & 5 & 4 & 3 & 2 & 1 & 0 & -1 & -2 & -3 & -4 & -5 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ * & * & * & * & * & * & * & * & * & * & * & * \\ 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 + \frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^5} = 32 + 4 + 0,5 + 0,125 = 36,625 \\ | & | & | & | & | & | & | & | & | & | & | & | \\ 64 & 32 & 16 & 8 & 4 & 2 & 1 & 0,5 & 0,25 & 0,125 & 0,0625 & 0,03125 \end{array}$$

Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto$$

Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto$$

$$0,65625$$

Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto$$

$$\begin{array}{r} 0,65625 \\ \times 2 \\ \hline 1,3125 \end{array}$$

Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto$$

$$\begin{array}{r} 0,65625 \\ 1,3125 \\ 0,625 \end{array} \times 2$$

Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto$$

$$\begin{array}{r} 0,65625 \\ 1,3125 \\ 0,625 \\ 1,25 \end{array} \xrightarrow{x2} \begin{array}{r} 1,3125 \\ 0,625 \\ 1,25 \end{array}$$

Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto$$

$$\begin{array}{r} 0,65625 \\ 1,3125 \\ 0,625 \\ 1,25 \\ 0,5 \end{array} \xrightarrow{x2} \begin{array}{r} 1,3125 \\ 0,625 \\ 1,25 \\ 0,5 \end{array}$$

Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto$$

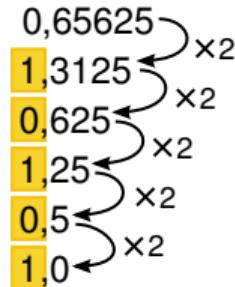
$$\begin{array}{r} 0,65625 \\ 1,3125 \\ 0,625 \\ 1,25 \\ 0,5 \\ 1,0 \end{array} \xrightarrow{\times 2} \begin{array}{r} 1,3125 \\ 0,625 \\ 1,25 \\ 0,5 \\ 1,0 \end{array}$$

Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto 0,10101_2$$

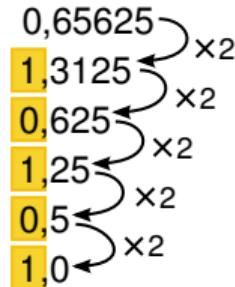


Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto 0,10101_2 \qquad 0,1_{10} \mapsto$$



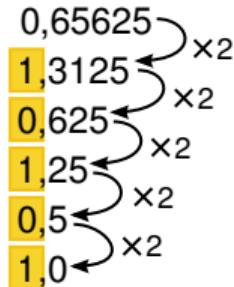
Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto 0,10101_2$$

$$0,1_{10} \mapsto$$



$$0,1$$

Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto 0,10101_2 \quad 0,1_{10} \mapsto$$

$$\begin{array}{r} 0,65625 \\ \times 2 \\ \hline 1,3125 \\ \times 2 \\ \hline 0,625 \\ \times 2 \\ \hline 1,25 \\ \times 2 \\ \hline 0,5 \\ \times 2 \\ \hline 1,0 \end{array}$$

$$\begin{matrix} 0,1 \\ 0,2 \end{matrix} \xrightarrow{x_2}$$

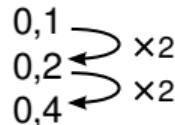
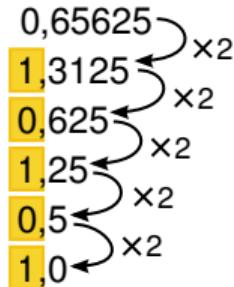
Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto 0,10101_2$$

$$0,1_{10} \mapsto$$



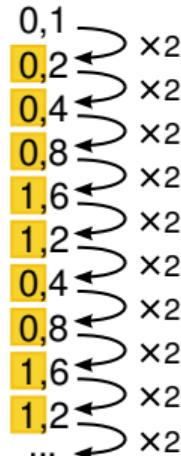
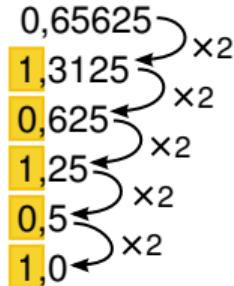
Convertir desde decimal a binario

Para convertir la parte fraccionaria a binario tenemos que multiplicar por dos y tomar la parte entera de la operación.

Por ejemplo,

$$0,65625_{10} \mapsto 0,10101_2$$

$$0,1_{10} \mapsto 0,000110011\dots_2$$

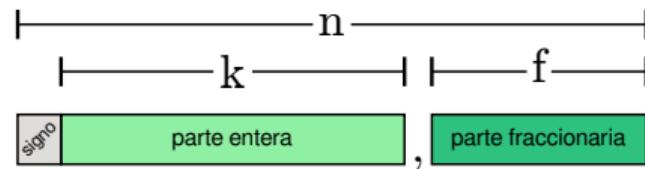


¡Moraleja! No todas las magnitudes pueden ser representadas de forma exacta entre cambios de base.

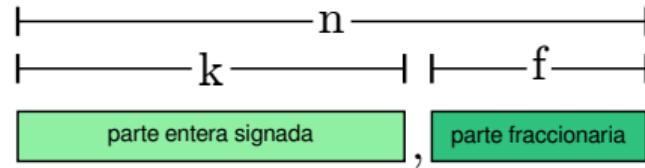
Punto Fijo

Utilizamos n bits en total para representar el número.

- Un bit de signo, k para la parte entera y f para la parte fraccionaria.



- Incluso, el signo puede ser codificado en la parte entera.
Por ejemplo, codificado como complemento a 2.



Punto Fijo - Ejemplo

Suponer una codificación de la forma:

signo (1 bit) + parte entera (9 bits) + parte fraccionaria (6 bits)

Ejemplo:

El número: 0011001010001001

Punto Fijo - Ejemplo

Suponer una codificación de la forma:

signo (1 bit) + parte entera (9 bits) + parte fraccionaria (6 bits)

Ejemplo:

El número: 0011001010001001

Lo interpretamos como: (0) 011001010. 001001

Punto Fijo - Ejemplo

Suponer una codificación de la forma:



Ejemplo:

El número: 0011001010001001

Lo interpretamos como: (0) 011001010. 001001

(0) → Positivo

Punto Fijo - Ejemplo

Suponer una codificación de la forma:



Ejemplo:

El número: 0011001010001001

Lo interpretamos como: (0) 011001010 . 001001

(0) → Positivo

011001010 → 202

Punto Fijo - Ejemplo

Suponer una codificación de la forma:



Ejemplo:

El número: 0011001010001001

Lo interpretamos como: (0) 011001010 . 001001

(0) → Positivo

011001010 → 202

001001 → 0.140625

Punto Fijo - Ejemplo

Suponer una codificación de la forma:

signo (1 bit) + parte entera (9 bits) + parte fraccionaria (6 bits)

Ejemplo:

El número: 0011001010001001

Lo interpretamos como: (0) 011001010. 001001

(0) → Positivo

011001010 → 202

001001 → 0.140625

Luego, 0011001010001001 → +202.140625

Punto Fijo - Ejemplo

Suponer una codificación de la forma:



Ejemplo:

El número: 0011001010001001

Lo interpretamos como: (0) 011001010. 001001

(0) → Positivo

011001010 → 202

001001 → 0.140625

Luego, 0011001010001001 → +202.140625

El número: 1000000110000111

Punto Fijo - Ejemplo

Suponer una codificación de la forma:



Ejemplo:

El número: 0011001010001001

Lo interpretamos como: (0) 011001010. 001001

(0) → Positivo

011001010 → 202

001001 → 0.140625

Luego, 0011001010001001 → +202.140625

El número: 1000000110000111

Lo interpretamos como: (1) 000000110. 000111 → -6.109375

(1) → Negativo

000000110 → 6

000111 → 0.109375

Luego, 1000000110000111 → -6.109375

Punto Flotante

Se representan mediante tres campos,

fracción: Dígitos representados. Comúnmente interpretado como $0.x \dots x$ o $1.x \dots x$.
Al formato $1.x \dots x$, también se lo conoce como *significando*.

exponente: Indica la escala del número, es decir, dónde se ubica el punto fraccionario.
Los exponentes pueden ser negativos y representar números aún más chicos.

signo: Signo de la magnitud representada.

Punto Flotante

Se representan mediante tres campos,

fracción: Dígitos representados. Comúnmente interpretado como $0.x \dots x$ o $1.x \dots x$.

Al formato $1.x \dots x$, también se lo conoce como *significando*.

exponente: Indica la escala del número, es decir, dónde se ubica el punto fraccionario.

Los exponentes pueden ser negativos y representar números aún más chicos.

signo: Signo de la magnitud representada.

Ejemplos:

- bit de signo + fracción como $0.x \dots x$ + exponente en signo más magnitud → (sig) $0.\text{fracción} \cdot 2^{(s)\text{exp}}$

Múltiples representaciones del mismo número.

- bit de signo + fracción como $1.x \dots x$ + exponente en notación exceso → (sig) $1.\text{fracción} \cdot 2^{\text{exp}-e}$

Representación única de los números. El cero se representa por convención.

Punto Flotante - Rango de representación

La representación en punto flotante no es uniforme sobre la recta numérica.



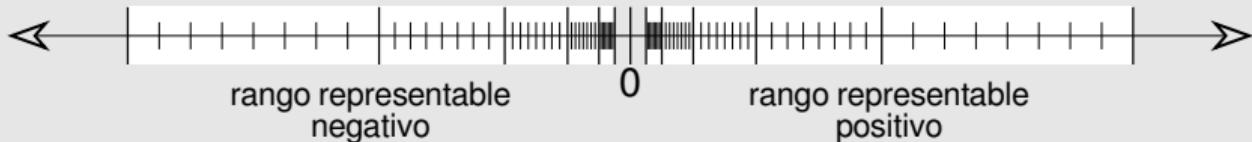
Punto Flotante - Rango de representación

La representación en punto flotante no es uniforme sobre la recta numérica.



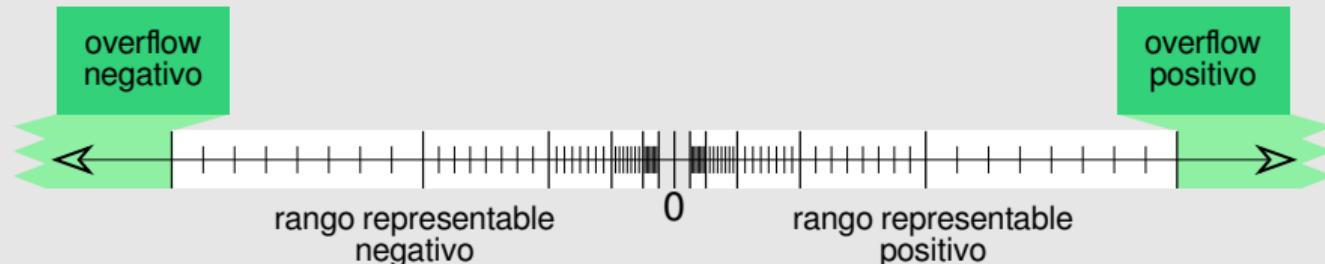
Punto Flotante - Rango de representación

La representación en punto flotante no es uniforme sobre la recta numérica.



Punto Flotante - Rango de representación

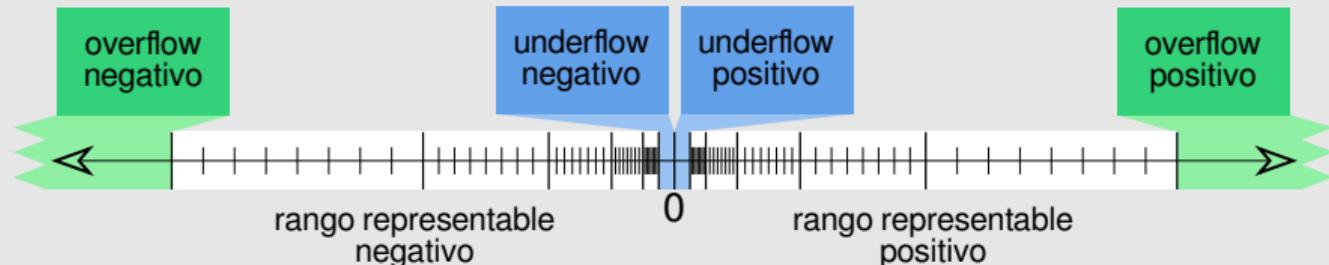
La representación en punto flotante no es uniforme sobre la recta numérica.



- **overflow:** Magnitudes que superan el límite máximo absoluto representable.

Punto Flotante - Rango de representación

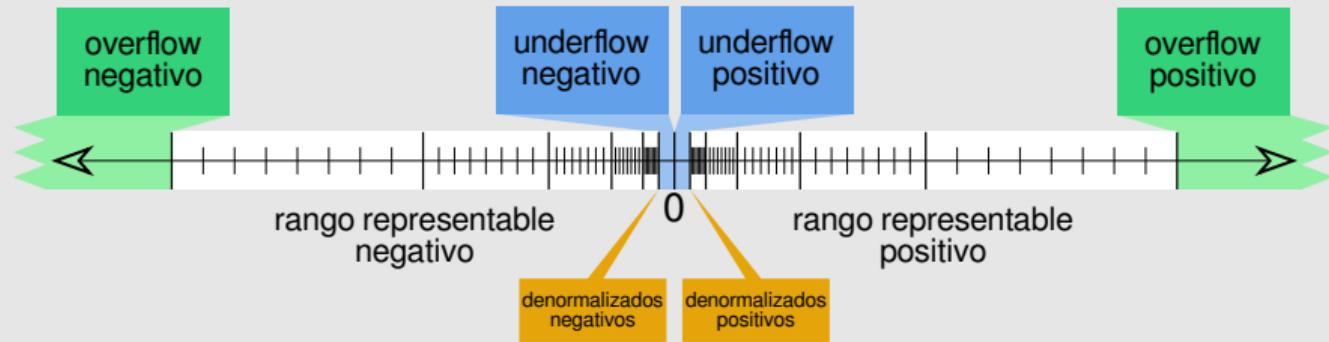
La representación en punto flotante no es uniforme sobre la recta numérica.



- **overflow:** Magnitudes que superan el límite máximo absoluto representable.
- **underflow:** Magnitudes más chicas que el mínimo absoluto representable distinto de cero.

Punto Flotante - Rango de representación

La representación en punto flotante no es uniforme sobre la recta numérica.



- **overflow:** Magnitudes que superan el límite máximo absoluto representable.
- **underflow:** Magnitudes más chicas que el mínimo absoluto representable distinto de cero.
- **denormalizado:** La *fracción* se interpreta comenzando por 0, permite representar números de magnitud muy pequeña.

Punto Flotante - Ejemplo

Suponer una codificación de la forma:

signo (1bit)

+ fracción (9 bits)

+ exponente (6 bits)

Ejemplo:

El número: 0001100101001001

fracción: Se interpreta como 0.

fracción

exponente: Se interpreta en complemento a 2

Punto Flotante - Ejemplo

Suponer una codificación de la forma:

signo (1bit)

+ fracción (9 bits)

+ exponente (6 bits)

Ejemplo:

El número: 0001100101001001

Lo interpretamos como: (0)0.001100101 · 2⁰⁰¹⁰⁰¹

fracción: Se interpreta como 0.

fracción

exponente: Se interpreta en complemento a 2

Punto Flotante - Ejemplo

Suponer una codificación de la forma:

signo (1bit) + fracción (9 bits) + exponente (6 bits)

Ejemplo:

El número: 0001100101001001

Lo interpretamos como: (0)0.⁰⁰¹¹⁰⁰¹⁰¹ · 2⁰⁰¹⁰⁰¹

(0) → Positivo

fracción: Se interpreta como 0.

fracción

exponente: Se interpreta en complemento a 2

Punto Flotante - Ejemplo

Suponer una codificación de la forma:

signo (1bit)

+ fracción (9 bits)

+ exponente (6 bits)

Ejemplo:

El número: 0001100101001001

Lo interpretamos como: (0)0.001100101 · 2⁰⁰¹⁰⁰¹

(0) → Positivo

0.001100101 → 0.197265625

fracción: Se interpreta como 0.

fracción

exponente: Se interpreta en complemento a 2

Punto Flotante - Ejemplo

Suponer una codificación de la forma:

signo (1bit)

+ fracción (9 bits)

+ exponente (6 bits)

Ejemplo:

El número: 0001100101001001

Lo interpretamos como: (0)0.001100101 · 2⁰⁰¹⁰⁰¹

(0) → Positivo

0.001100101 → 0.197265625

001001 → 9

fracción: Se interpreta como 0.

fracción

exponente: Se interpreta en complemento a 2

Punto Flotante - Ejemplo

Suponer una codificación de la forma:

signo (1bit)

+ fracción (9 bits)

+ exponente (6 bits)

fracción: Se interpreta como 0.

fracción

exponente: Se interpreta en complemento a 2

Ejemplo:

El número: 0001100101001001

Lo interpretamos como: (0)0.001100101 · 2⁰⁰¹⁰⁰¹

(0) → Positivo

0.001100101 → 0.197265625

001001 → 9

Luego, 0001100101001001 → 0.197265625 · 2⁹ → 101

Punto Flotante - Ejemplo

Suponer una codificación de la forma:

signo (1bit)

+ fracción (9 bits)

+ exponente (6 bits)

fracción: Se interpreta como 0.

fracción

exponente: Se interpreta en complemento a 2

Ejemplo:

El número: 0001100101001001

Lo interpretamos como: (0)0.001100101 · 2⁰⁰¹⁰⁰¹

(0) → Positivo

0.001100101 → 0.197265625

001001 → 9

Luego, 0001100101001001 → 0.197265625 · 2⁹ → 101

El número: 1000000011000111

Punto Flotante - Ejemplo

Suponer una codificación de la forma:

signo (1bit)

+ fracción (9 bits)

+ exponente (6 bits)

fracción: Se interpreta como 0.

fracción

exponente: Se interpreta en complemento a 2

Ejemplo:

El número: 0001100101001001

Lo interpretamos como: (0)0.001100101 · 2⁰⁰¹⁰⁰¹

(0) → Positivo

0.001100101 → 0.197265625

001001 → 9

Luego, 0001100101001001 → 0.197265625 · 2⁹ → 101

El número: 1000000011000111

Lo interpretamos como: (1)0.000000011 · 2⁰⁰⁰¹¹¹

(1) → Negativo

0.000000011 → 0.005859375

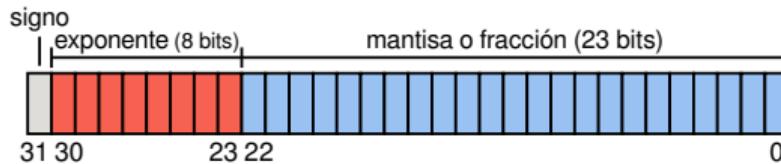
000111 → 7

Luego, 1000000011000111 → -0.005859375 · 2⁷ → -0.75

Punto Flotante - IEEE 754

Una de las codificaciones en punto flotante más utilizada es el estándar IEEE 754. Permite definir números **float** (32 Bits) y números **double** (64 Bits).

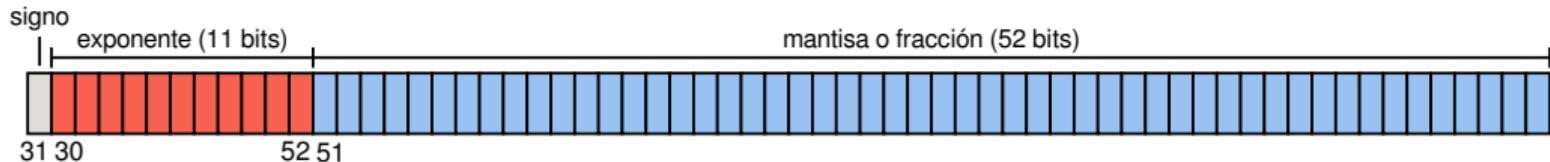
32 Bits



(signo) 1.**fracción** · $2^{\text{exponente} - \text{exceso}}$

para 32 bits exceso=127
para 64 bits exceso=1023

64 Bits



Punto Flotante - Ejemplo IEEE 754

signo + fracción + exponente

Ejemplo: (IEEE 754)

signo	exponente	fracción
0	10000100	0110111011110101110001

$$(\text{signo}) \text{.fracción} \cdot 2^{\text{exponente}-127} =$$

$$= (+)1.0110111011110101110001 \cdot 2^{10000100-127} =$$

$$= +45.869999$$

Representación de caracteres

Existen muchas formas de representar **codificar** caracteres.

Las codificaciones se basan en **tablas**, que indican qué bits corresponden a cada carácter.

Dependiendo de la cantidad de bits/bytes usados para codificar cada carácter, pueden ser:

- de tamaño fijo
- de tamaño variable

Algunos ejemplos

- **ASCII**: Fija, 1 byte. Aunque solo se usan 7 bits para codificar caracteres.
- **UTF-8**: Variable, 1 a 4 bytes. Codificación Unicode de longitud variable.
- **UTF-16**: Variable, 2 o 4 bytes. Codificación Unicode optimizado para caracteres multilingües.
- **UTF-32**: Fija, 4 bytes. Codificación Unicode simple.
- **Latin-1 (ISO-8859-1)**: Fija, 1 byte. Caracteres latinos, tildes, diéresis, cedilla, eñe, etc.
- **GB 18030**: Variable, 1 a 4 bytes. Estándar utilizado en China.

Representación de caracteres - ASCII

Dec	Hex																
0	00	NUL	16	10	DLE	32	20	48	30	0	64	40	@	80	50	P	
1	01	SOH	17	11	DC1	33	21	!	49	31	1	65	41	A	81	51	Q
2	02	STX	18	12	DC2	34	22	"	50	32	2	66	42	B	82	52	R
3	03	ETX	19	13	DC3	35	23	#	51	33	3	67	43	C	83	53	S
4	04	EOT	20	14	DC4	36	24	\$	52	34	4	68	44	D	84	54	T
5	05	ENQ	21	15	NAK	37	25	%	53	35	5	69	45	E	85	55	U
6	06	ACK	22	16	SYN	38	26	&	54	36	6	70	46	F	86	56	V
7	07	BEL	23	17	ETB	39	27	'	55	37	7	71	47	G	87	57	W
8	08	BS	24	18	CAN	40	28	(56	38	8	72	48	H	88	58	X
9	09	HT	25	19	EM	41	29)	57	39	9	73	49	I	89	59	Y
10	0A	LF	26	1A	SUB	42	2A	*	58	3A	:	74	4A	J	90	5A	Z
11	0B	VT	27	1B	ESC	43	2B	+	59	3B	;	75	4B	K	91	5B	[
12	0C	FF	28	1C	FS	44	2C	,	60	3C	<	76	4C	L	92	5C	\
13	0D	CR	29	1D	GS	45	2D	-	61	3D	=	77	4D	M	93	5D]
14	0E	SO	30	1E	RS	46	2E	.	62	3E	>	78	4E	N	94	5E	^
15	0F	SI	31	1F	US	47	2F	/	63	3F	?	79	4F	O	95	5F	_

Representación de caracteres - UTF-8

Los caracteres se codifican según el rango al que pertenezcan.

Los primeros 127 corresponden a la codificación ASCII.

#bytes	desde	hasta	byte 1	byte 2	byte 3	byte 4
1	0	127	0xxxxxxx			
2	128	2047	110xxxxx	10xxxxxx		
3	2048	65535	1110xxxx	10xxxxxx	10xxxxxx	
4	65536	1114111	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

El primer byte indica cuántos bytes a continuación se deben leer (mismo prefijo).

No importa el byte que se lea, siempre se puede interpretar parcialmente el carácter.

Ejemplos:

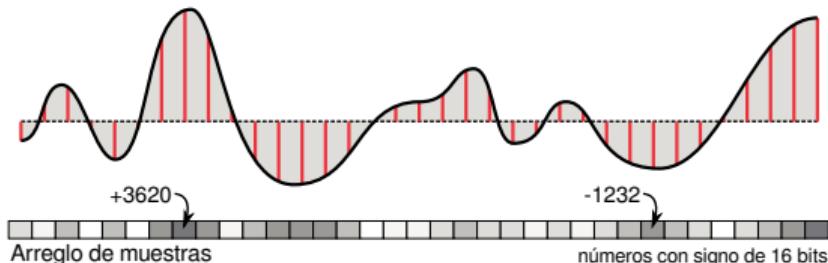
U+03A3 → ce a3 → GREEK CAPITAL LETTER SIGMA → Σ

U+2197 → e2 86 97 → NORTH EAST ARROW → ↗

U+10890 → f0 90 a2 90 → NABATAEAN LETTER FINAL LAMEDH → ՚

Representación de datos

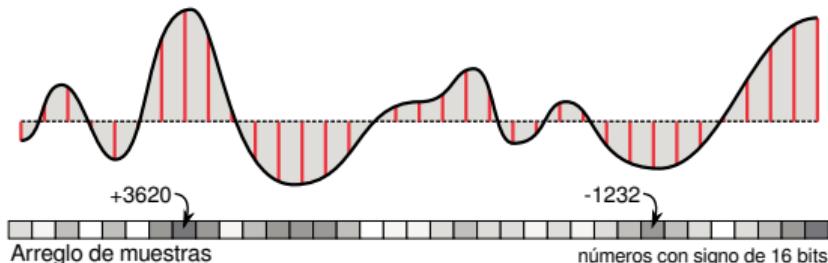
Sonido



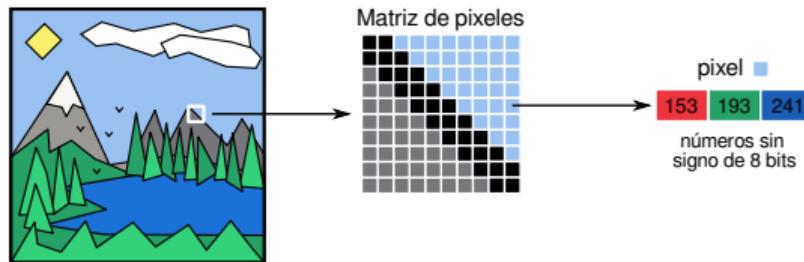
Ejemplo: Formato WAV
Almacena muestras de señales de audio sin comprimir, permite frecuencias de muestreo de 44kHz a 16 bits.

Representación de datos

Sonido



Imagen



Ejemplo: Formato WAV

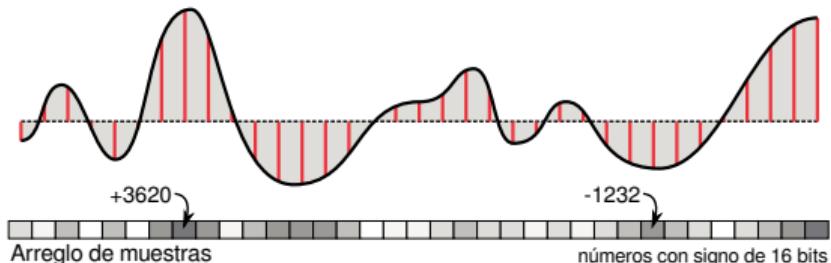
Almacena muestras de señales de audio sin comprimir, permite frecuencias de muestreo de 44kHz a 16 bits.

Ejemplo: Formato BMP

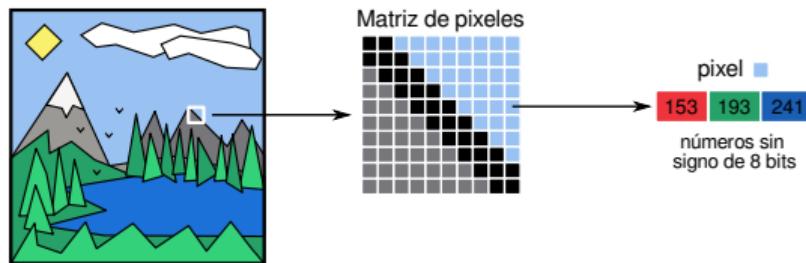
Guarda mapas de bits sin compresión. Permite guardar imágenes en escala de grises y en colores de 24 o 32 bits.

Representación de datos

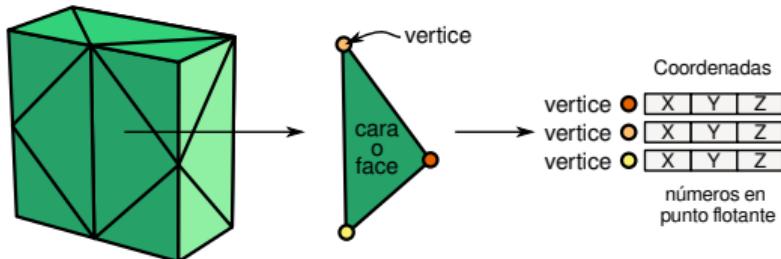
Sonido



Imagen



Diseño 3D



Ejemplo: Formato WAV

Almacena muestras de señales de audio sin comprimir, permite frecuencias de muestreo de 44kHz a 16 bits.

Ejemplo: Formato BMP

Guarda mapas de bits sin compresión. Permite guardar imágenes en escala de grises y en colores de 24 o 32 bits.

Ejemplo: Formato STL

Permite representar superficies 3D por medio de la descripción de triángulos en coordenadas cartesianas.

Bibliografía

- Tanenbaum, "Organización de Computadoras. Un Enfoque Estructurado", 4ta Edición, 2000.
 - **Apéndice - Números Binarios** - Páginas 631-640
 - **Apéndice - Números de Punto Flotante** - Páginas 643-650
- Null, "Essentials of Computer Organization and Architecture", 5th Edition, 2018.
 - **Chapter 2 - Data Representation in Computer Systems**
 - 2.4 - Signed Integer Representation
 - 2.5 - Floating-Point representation
 - 2.6 - Character Codes

Ejercicios

Con lo visto, ya pueden resolver todos los ejercicios de la Práctica 1.

¡Gracias!

Recuerden leer los comentarios adjuntos
en cada clase por aclaraciones.