SISTEMAS DE REPRESENTACIÓN NUMÉRICA

Sistemas para la representación de números

Sistema numérico
- Conjunto de símbolos : desarrollo histórico
- Reglas para su organización
  - sistemas aditivos
  - sistemas posicionales

- Sistemas aditivos: El valor de cada símbolo es fijo, se pueden colocar en cualquier orden.
- Sistemas posicionales: El valor de cada dígito del número depende de su posición.

↳ Se define por:

- Símbolos
- Cantidad de símbolos : base
- Peso de cada posición
- Cantidad de posiciones.

Conversión entre diferentes bases

Conversión de cualquier base a base decimal

Cada número se escriba como un polinomio donde cada dígito es un coeficiente, que multiplica a la base, la cual está elevada a la posición del dígito.

Ejemplo: dígito en pos. 2 dígito en pos. 1 dígito en pos. 0.

$$342|_6 = \textcircled{3}\cdot 6^{\textcircled{2}} + \textcircled{4}\cdot 6^{\textcircled{1}} + \textcircled{2}\cdot 6^{\textcircled{0}} = 134|_{10}$$ → nº en decimal.

↳ nº a convertir

base original

Conversión de decimal a cualquier base

→ Hay varias formas:

- **Con los pesos**

i- Escribo los pesos de la base de destino: $\text{base}^{\textcircled{n}} | \text{base}^{n-1} | \ldots | \text{base}^{1} | \text{base}^{0}$.

↗ puede ser un valor hasta ∞.

ii- De izquierda a derecha voy analizando cada peso.

- Me detengo en el primer peso cuyo valor es menor o igual al número original.
- De izquierda a derecha analizo por cuánto puedo multiplicar a cada peso de forma tal que sumando las anteriores multiplicaciones el resultado sea menor o igual al número original.
- El resultado es el conjunto de multiplicadores de los pesos.

Ejemplo:

$134|_{10}$ a base 6:

- $6^{\textcircled{4}} | 6^{3} | 6^{2} | 6^{1} | 6^{0} = 1296 | 216 | 36 | 6 | 1$

↳ elijo a ojo: su resultado debe ser mayor al número original $(1296 > 134)$ ✓

- ¿1296 <= 134? No. → sigo
- ¿216 <= 134? No → sigo
- ¿36 <= 134? Sí → comienzo a contar.
- $x_{\max} \in \mathbb{N}$ tq $36.x <= 134$ → $x = 3$ → primer dígito de mi resultado.

↳ resultado = 3 _ _

- $x_{\max} \in \mathbb{N}$ tq. $6.x <= 134 - \underbrace{3.36}_{\text{la multiplicación anterior}}$ → $x = 4$ → segundo dígito de mi resultado

↳ resultado = 34 _

- $x_{\max} \in \mathbb{N}$ tq. $1.x <= 134 - \underbrace{3.36 - 4.6}_{\text{multiplicaciones anteriores}}$ → $x = 2$ → último dígito de mi resultado.

↳ resultado = 342

⇒ $134|_{10} = 342|_{6}$.

NOTA

Sistemas de representación numérica

- Con divisiones

i- Divido el número original por la base a la cual quiero convertir.

ii- El resultado lo divido por la base a la cual quiero convertir.

iii- Repito ii. hasta que el resultado sea 0.

iv- Escribo los restos de izq. a derecha de forma que el último resto sea el valor más a izquierda, y el primero sea el de más a derecha. El número obtenido es el resultado.

Ejemplo:

$134|_{10}$ a base 6:

- 134 |6 • 22 |6 • 3 |6 → 3 4 2
14 ㉒ → ④ ③ → ③ 0
②

⇒ $134|_{10} = 342|_6$

Conversión de cualquier base a cualquier otra base

i- Convierto el número original a un número en base decimal

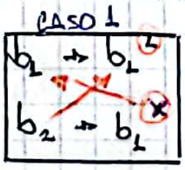
ii- Convierto el número en decimal a la base final

→ sí o sí paso por decimal

Conversión de una base a otra, siendo una de ellas una potencia de la otra

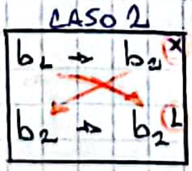
i- Escribo la relación entre las bases:

CASO 1



ó

CASO 2



→ la relación es de x

↳ x en b_1 equivalen a 1 en b_2

↳ x en b_2 equivalen a 1 en b_1

→ Tengo dos posibles casos

- **Caso 1: $b_1 < b_2$**

i- Agrupo el número original de a x dígitos, de derecha a izquierda

ii- Cada grupo lo convierto a b_2

iii- El conjunto de los resultados es el resultado final

Ejemplo:

$1011101111|_2$ a base 16

- $2 \to 2^1$ → relación de 4.
  $16 \to 2^4$

- $\underbrace{10}\underbrace{1110}\underbrace{1111}$
  → $0010|_2 = 2 = 2|_{16}$.
  → $1110|_2 = 14 = E|_{16}$ → 2EF
  → $1111|_2 = 15 = F|_{16}$

⇒ $1011101111|_2 = 2EF|_{16}$

- **Caso 2: $b_1 > b_2$**

i- Cada dígito lo escribo con x dígitos → Ese grupo es la conversión del dígito original en b_1 a b_2.

ii- El conjunto de dígitos es el resultado final

Ejemplo:

$841|_9$ a base 3

- $9 \to 3^2$ → relación de 2
  $3 \to 3^1$

- $\underbrace{8}\underbrace{4}\underbrace{1}$
  → $8 = 22|_3$
  → $4 = 11|_3$ → 221101
  → $1 = 01|_3$

⇒ $841|_9 = 221101|_3$

Caso especial: números con parte fraccionaria.

Sistema de punto fijo

- Puede aplicarse a una base cualquiera

Conversión de cualquier base a base decimal

→ igual, pero los exponentes después de la coma son negativos

Ejemplo: $54,38|_9 = 5.9^1 + 4.9^0 + 3.9^{-1} + 8.9^{-2} = 49,43|_{10}$.

Conversión de base decimal a base binaria

i. Calculo la parte entera

ii. Calculo la parte decimal de la siguiente forma:

1. La parte decimal la multiplico por 2
2. Del resultado guardo la parte entera (1 o 0) y la parte decimal.
3. Repito hasta que la parte decimal sea nula, o hasta donde quiera → el resultado es el entero + los enteros obtenidos.

Ejemplo: $20,32|_{10}$:

- $20|_{10} = 10100|_2$.
- $0,32 \rightarrow$

$$\begin{aligned} 0,32 \times 2 &= \textcircled{0},64 \\ 0,64 \times 2 &= \textcircled{1},28 \\ 0,28 \times 2 &= \textcircled{0},56 \\ 0,56 \times 2 &= \textcircled{1},12 \\ 0,12 \times 2 &= \textcircled{0},24 \\ 0,24 \times 2 &= \textcircled{0},48 \end{aligned} \rightarrow 010100$$

$\Rightarrow 20,32|_{10} \simeq 10100,010100$ ...

→ Los resultados no son precisos:
- puedo calcular el error específico de cada caso (valor original menos valor calculado)
- puedo calcular cota de máximo error haciendo $2^{-\text{cant. decimales}}$

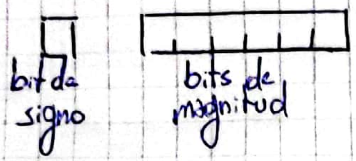
Sistema de punto flotante

→ Lo vemos más adelante

Representación de enteros con signo

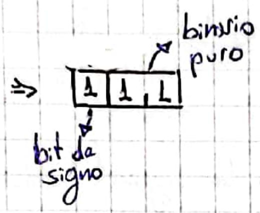
Magnitud y signo (Signed Magnitude)

- **Representación**



i. Escribo el número a convertir en binario puro.
ii. Le agrego a la izquierda (en el bit de signo) un 0 si es positivo o un 1 si es negativo

Ejemplo: $-3|_{10}$: · $3|_{10} = 11|_2$
· $-3|_{10} = 111|_2$ ⇒



- **Rango**: $(-2^{n-1}+1)|_{10} \leq x \leq (2^{n-1}-1)|_{10}$ y doble valor para el 0.

Complemento a la base menos uno

En binario: Complemento a 1 (One's complement)

- **Representación**
  - Si el número es positivo lo escribo en binario puro
  - Si el número es negativo lo escribo como $\text{base}^{\text{cant. digitos}} - 1 - |\text{número}|$.

    ↳ en binario puedo hacer la resta o invertir bit a bit

- **Rango**: $(-2^{n-1}+1)|_{10} \leq x \leq (2^{n-1}-1)|_{10}$ y doble valor para el 0

NOTA

Complemento a la base

En binario: Complemento a 2 (two's complement)

- **Representación**
  - Si el número es positivo, lo escribo en binario puro.
  - Si el número es negativo, lo escribo como $\text{base}^{\text{cant. dígitos}} - |\text{número}|$

    ↳ en binario puedo hacer la resta o invertir bit a bit y sumar 1

- **Rango**: $(-2^{n-1})|_{10} \leq x \leq (2^{n-1}-1)|_{10}$ y un sólo valor para el 0.

Decimal	Complemento a la base (C_b)	Complemento a la base - 1 (C_{b-1})	Magnitud y signo
-8	1000	—	—
-7	1001	1000	1111
-6	1010	1001	1110
-5	1011	1010	1101
-4	1100	1011	1100
-3	1101	1100	1011
-2	1110	1101	1010
-1	1111	1110	1001
0	0000	1111 ó 0000	1000 ó 0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	0101	0101
6	0110	0110	0110
7	0111	0111	0111

Flags

- **Carry**: bit más a la izquierda del resultado de una operación.
  - si c = 1 ⇒ se fue de rango (error)
  - se utiliza al operar con binario puro (sin signo)
- **Overflow**: se calcula comparando los últimos dos carry outs de una suma ("lo que me llevo").
  - si v = 1 ⇒ se fue de rango

    ↳ en complemento a 2 lo veo cuando sumo dos números de igual signo (y el resultado tiene signo opuesto.
  - se utiliza al operar con binario signado

Zero: vale 1 si el resultado es igual a 0.

Signo (negated): vale 1 si el resultado es negativo.

Paridad: Analizo la cant. de dígitos 1. Si es una cantidad par entonces vale 0, si es impar vale 1.

Ej:

$$\begin{array}{r} 110 \\ 1011 \\ +\ 110 \\ \hline \boxed{1}001 \end{array} \rightarrow \text{sin signo} \Rightarrow \text{miro solo c.}$$

$\hookrightarrow c = 1$

$$\begin{array}{r} \overset{\text{sale}}{①}\overset{\text{entra}}{①}010110 \\ \hline 01010110 \\ +\ 11010010 \\ \hline 100101000 \end{array} \rightarrow \text{con signo, complemento a 2.} \Rightarrow \text{miro solo V}$$

$\hookrightarrow V = 0$

Operaciones

Para operar con números con una cantidad distinta de dígitos, extiendo signo:
- Si es binario puro o positivo, extiendo a izquierda con 0s.
- Si es negativo, extiendo a izquierda con 1s.

Suma

No puedo sumar n° con diferentes bases uno del otro.

- Calculo el módulo $= 2^{\text{cant. dígitos}}$
- Obtengo el rango
  - $(-2^{n-1}+1)|_{10} \leq x \leq (2^{n-1}-1)|_{10}$ para MyS y Complemento a la base -1.
  - $(-2^{n-1})|_{10} \leq x \leq (2^{n-1}-1)|_{10}$ para Complemento a la base.
- Hago la suma de la manera normal, situando el primer número arriba del segundo, de manera que en cada columna estén los dígitos del mismo peso.
  - Por cada columna:
    - Agarro el dígito mayor
    - Calculo cuánto falta para llegar al valor de la base $\Rightarrow (\text{base} - \text{dígito}_1) = d$
    - Calculo con el dígito menor su suma con el último carry (si lo hubiera) $\Rightarrow (\text{dígito}_2 + \text{carry}) = b$
      - Si $b \geq d \Rightarrow$ voy a "llevarme" un 1
        en la columna, en la fila del resultado coloco $(b-d)$
      - Si $b < d \Rightarrow$ me "llevo" 0.
        en la columna, en la fila del resultado coloco $(d-b)$

NOTA

- Con los flags veo fácil si se fue o no de rango o sino puedo pasar todo a decimal y fijarme a mano

→ Si tengo punto fijo es lo mismo

Resta

Puedo hacerlo de manera directa o sumando con valores negativos.
↳ casi siempre lo más fácil.

- Convierto el segundo número (sustraendo), en la representación que corresponde, al negativo.
  - Si trabajo con C_{b-1} entonces le aplico el C_{b-1}.
  - Si trabajo con C_b entonces le aplico C_b.
- Realizo la suma. → Si trabajo con C_{b-1} ⇒ al resultado le sumo el carry: ese el verdadero resultado.

Observaciones

- El resultado está en representación; para saber su valor lo tengo que calcular.
  → El carry lo ignoro y lo descarto.
- Es posible, al sumar dos números de igual signo, que haya un desborde y que el resultado tenga signo opuesto ⇒ está mal el resultado, lo ignoro.

División

- Desplazo a derecha: divido por la base.

Ej: 010110 → 22
001011 → 11
} → dividí por la base = 2.

Multiplicación

- Desplazo a izquierda: multiplico por la base

Ej: 010110 → 22
101100 → 44
} → multipliqué por la base = 2.

EJEMPLOS

- 26 → Binario puro / Cb / Cb-1 = 011010.
- -26 → Cb = 100110
  ↳ Cb-1 = 100101
- 19 → Binario puro / Cb / Cb-1 = 010011
- -19 → Cb = 101101
  ↳ Cb-1 = 101100

Rango: Binario: $0 \leq x \leq (2^6-1) \Rightarrow 0 \leq x \leq 63$.

$Cb: (-2^5) \leq x \leq (2^5-1) \Rightarrow -32 \leq x \leq 31$

$Cb\text{-}1: (-2^5+1) \leq x \leq (2^5-1) \Rightarrow -31 \leq x \leq 31$

- 19 + 26 = → **CASO DESBORDE** → Mismo que si hago -19-26 =

$$\begin{array}{r} 1001 \\ \hline 010011 \\ +\ 011010 \\ \hline 101101 \end{array}$$

- En binario puro = 45 ✓
  C = 0 ✓
- En Cb = -19 ⊗
  V = 1 → error fuera de rango
- En Cb-1 = -18 ⊗
  V = 1 → error fuera de rango.

- 19-26 = → **CASO RESTA NORMAL**

$$\begin{array}{r} 11 \\ \hline 010011 \\ +\ 100110 \\ \hline 111001 \end{array} \rightarrow Cb. \rightarrow -7 ✓$$

$$\begin{array}{r} 111 \\ \hline 010011 \\ +\ 100101 \\ \hline 111000 \end{array} \rightarrow Cb\text{-}1 \rightarrow -7 ✓$$

- 26-19 = **CASO HAY CARRY EN Cb-1**

$$\begin{array}{r} 11 \\ \hline 011010 \\ +\ 101101 \\ \hline 1\,000111 \end{array} \rightarrow Cb \rightarrow 7 ✓$$

$$\begin{array}{r} 11 \\ \hline 011010 \\ 101100 \\ \hline ①000110 \end{array} \rightarrow Cb\text{-}1$$

$$\begin{array}{r} \hookrightarrow 000110 \\ +\ ① \\ \hline 000111 \end{array} \rightarrow 7 ✓$$

NOTA