Organización del computador

La unidad mínima de almacenamiento en una computadora es el BIT. Indica un pasaje o no de corriente.

Regla de números binarios: Voy llenando columnas de derecha a izquierda con uno, desplazándolo y poniendo uno detrás.

Base de un sistema de numeración: cantidad de símbolos disponibles de un sistema.

El número de símbolos de cualquier sistema se representa como 10.

Teorema Fundamental de la numeración:

Un número expresado en una base **b**, para traducirlo en base 10 debo asignarle un índice a cada número que conforma el número total. Para ello, comienzo asignándole 0 al número a la izquierda de la coma para luego seguir con el curso natural de los enteros:

Teorema Fundamental de la Numeración:

$$(\ldots ABC,DEF\ldots) = Ab^2 + Bb^1 + Cb^0 + Db^{-1} + Eb^{-2} + Fb^{-3}$$

Ej. $(423,1)_6 = (4x6^2 + 2x6^1 + 3x6^0 + 1x6^{-1})_{10}$

Cambio de Base

Convertir un número de base b a 10:

A. $N_b \rightarrow (\ )_{10}$

$$ABCD_b = (A\mathbf{b}^3 + B\mathbf{b}^2 + C\mathbf{b}^1 + D\mathbf{b}^0)_{10}$$

Ejemplos:

$34_6 = (3x6^1 + 4x6^0)_{10} = (22)_{10}$

$A7_{16} = (10x16^1 + 7x16^0)_{10} = (167)_{10}$

De base 10 a base b → divisiones sucesivas:

Consiste en realizar divisiones sucesivas hasta que el cociente sea menor que el divisor (base deseada), luego juntar el último cociente obtenido junto con todos los restos de abajo hacia arriba.

Ejemplos:

$34_{10} \rightarrow (\ )_2$

$$
\begin{array}{llllll}
34\ |\underline{2} & & & & & \\
\mathbf{0} & 17\ |\underline{2} & & & & \\
 & \mathbf{1} & 8\ |\underline{2} & & & \\
 & & \mathbf{0} & 4\ |\underline{2} & & \\
 & & & \mathbf{0} & 2\ |\underline{2} & \\
 & & & & \mathbf{0} & \mathbf{1}
\end{array}
$$

Hemos llegado al final de las divisiones sucesivas notar que $1 < 2 \rightarrow$
El resultado final es $\mathbf{(100010)_2}$

Ir de una base B a una base P: Para lograr esto, puedo convertir a base 10 (entendiendo el método a partir de las posiciones) y luego convertirlo a base P con las divisiones sucesivas.

Partes Fraccionarias

El subíndice será negativo y para pasar de base b a base 10 es la misma lógica:

A. $0,N_b \rightarrow (\ \ )_{10}$

$$0,ABC_{\ b} = (Ab^{-1} + Bb^{-2} + Cb^{-3})_{10}$$

Ejemplo:

$$0,132_4 = (1x4^{-1} + 3x4-2 + 2x4-3)10 = 0,4687510$$

Para pasar de base 10 a b:

B. $0{,}N_{10} \rightarrow (\ )_b$

Se resuelve mediante multiplicaciones sucesivas tomando de cada resultado la parte entera. Se termina cuando la parte fraccionaria es igual a cero. En caso de no llegar a este resultado cuantos más decimales se toman mayor precisión se alcanza.

Ejemplo:

$0{,}125_{10} \rightarrow (\ )_2$

$0{,}125 \times 2 = \mathbf{0}{,}250$ **0**
$0{,}250 \times 2 = \mathbf{0}{,}500$ **0**
$0{,}500 \times 2 = \mathbf{1}{,}000$ **1**

$\mathbf{0{,}125_{10} = 0{,}001_2}$

Cambio de base por potencia o raíz exacta

a) $b^x = p$

Se irán formando grupos de x dígitos y se hará el cambio para cada uno de estos grupos en forma independiente. Para la parte entera se empiezan a formar los grupos de derecha a izquierda en caso de ser necesario se completa con ceros a izquierda. Para la parte fraccionaria se comienza a formar los grupos de izquierda a derecha y de ser necesario se completa con ceros a derecha.

Ejemplo:

$10110_2 \rightarrow (\ )_8$ $2^3 = 8$ **tomo de 3 dígitos**

$(\mathbf{0}10 \mid 110)_2 = 26_8$
$110_2 = 6_8$
$010_2 = 2_8$

b) $b^{1/x} = p$

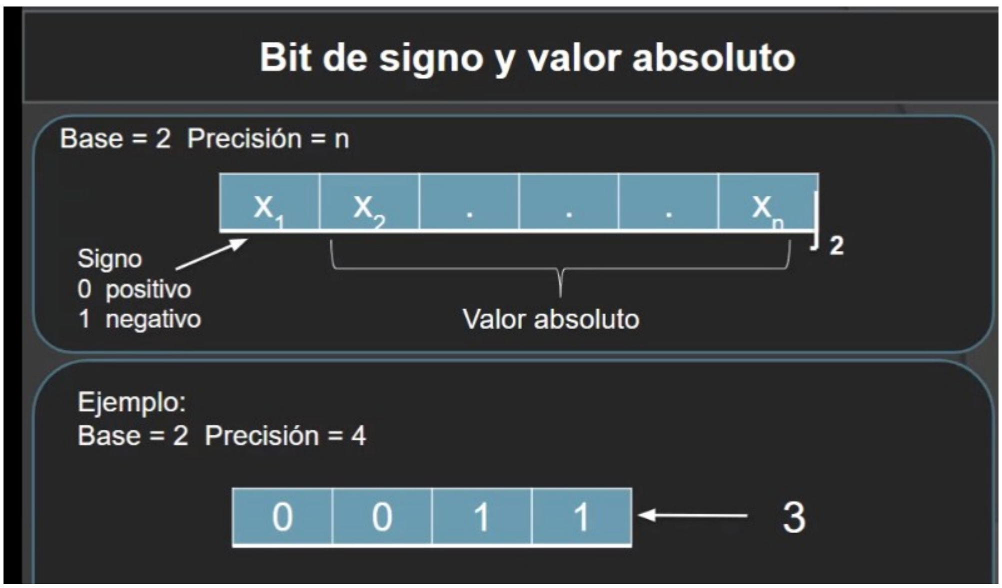
Cada dígito en la base b se expandirá en x dígitos en la base p.

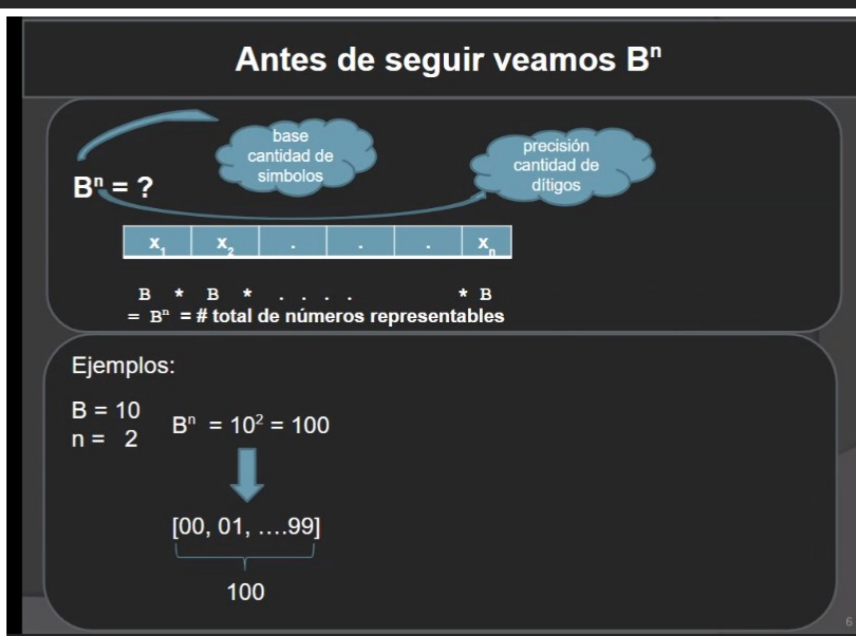
Ejemplo:

$AE75_{16} \rightarrow (\ \ )_2$ $16^{1/4} = 2$ **se expande en 4 dígitos binarios**

$5_{16} = 0101_2$
$7_{16} = 0111_2$
$E_{16} = 1110_2$
$A_{16} = 1010_2$ **$(1010111001110101)_2$**

Bit de signo y valor absoluto





Complemento a 1

Paso por paso

Base = 2   precisión = 4
Representar -6

1. Pasar valor absoluto a base 2: $|-6_{10}| = 6_{10} = 110_2$
2. Completar con 0 a izquierda hasta completar n: 0110
3. Si es negativo, complementar (hacer NOT): 1001

Indicar número almacenado en 1101

1. Si primer bit es 1 (es negativo), complementar (hacer NOT): 0010
2. Pasar a base 10: $0010_2 = 2_{10}$
3. Indicar el número según signo y valor obtenidos: -2

Complemento a 1

Rango de Representación

Minimo: $-(2^{n-1}-1)$
Maximo: $2^{n-1}-1$

Desventajas

- Doble representación del 0

Ventajas

- Rango simétrico
- Permite operar aritméticamente sumando el "end-around carry"

```
11
 0101       5
+1110    + -1
 ----      --
 0011   ≠   4
+   1
 ----
 0100   =   4
```

Complemento a la Base

Base = B Precisión = n

$Rep(x_b) = x_b$ si $x \geq 0$
$Rep(x_b) = Cb(|x_b|)$ si $x < 0$

Ejemplos:

$B = 10 \quad n = 2 \Rightarrow B^n = 10^2 = 100$

Rep(3) = 03
Rep(-3) = Cb(3) = 100 – 3 = 97
Rep(-1) = Cb(1) = 100 – 1 = 99
Rep(-50) = Cb(50) = 100 – 50 = 50
Rep(50) => NO SE PUEDE
=> 49 es el mayor positivo representable

¿ Pero que es Cb?

$Cb(r) + r = B^n$
$\Longrightarrow Cb(r) = B^n - r$

Notar que:
Si
k es complemento de r
$\Rightarrow k + r = B^n$

=> r es complemento de k

13

Complemento a la Base

Rango de Representación

Mínimo: $-(B^n/2)$
Máximo: $(B^n/2) - 1$

Desventajas
- Rango asimétrico (un negativo más)

Ventajas
- Única representación del 0
- Permite operar aritméticamente

Complemento a la Base

Permite operar aritméticamente:

Sumas (Con B = 10 y n = 2)

```
5 + 2   (es 7)        5 + (-2)   (es 3)        -5 + 2   (es -3)             -5 + (- 2)   (es -7)

05 + 02               05 + 98                  95 + 02                      95 + 98

00                     11                      00                            11
 05                     05                      95                            95
+02                    + 98                    + 02                          + 98
 ---                    ---                     ---                           ---
007  (A)               103  (B)                097  --> -3  (C)              193  --> -7  (D)
```

Restas: al plantear A-B se transforma en A+Bcomp

```
5 - 2   (es 3)        5 - (-2)   (es 7)        95 - 02   (es -7)            -5 - (- 2)   (es -3)

05 - 02               05 - 98                  95 - 02                      95 - 98
05 + Cb(2)            05 + Cb(98)              95 + Cb(2)                   95 + Cb(98)
= 05 + 98             = 05 + 02                = 95 + 98                    = 95 + 02
= 103                 = 07                     = 193                        = 97
(B)                   (A)                      (D)
```

Permite operar aritméticamente: CONCLUSIÓN

A - B

Se trabaja como

A + Comp(B)

***** NO IMPORTA EL SIGNO DE B ******

Complemento a 2

Base = 2 Precisión = n

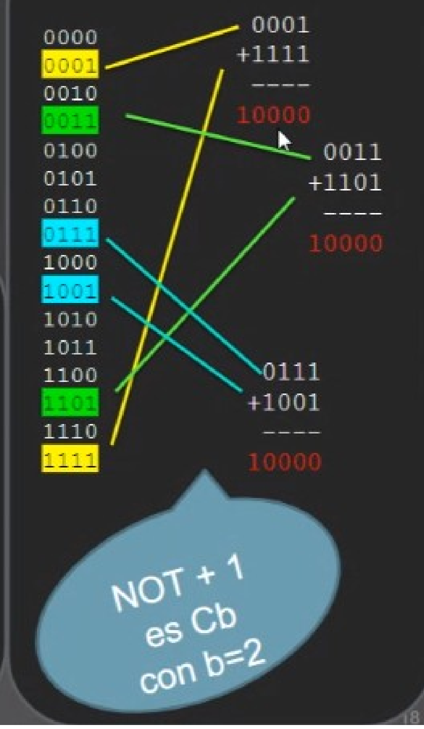
$Rep(x_{10}) = x_2$ si $x \geq 0$
$Rep(x_{10}) = NOT(|x_2|) + 1$ si $x < 0$

Ejemplos:

B = 2   n = 4

Rep (3) = 0011 Rep(-3) = NOT(0011) + 1 = 1100 + 1 = 1101 Rep(-1) = NOT(0001) + 1 = 1110 + 1 = 1111	Rep(-8) = NOT(1000) + 1 = 0111 + 1 = 1000 Rep(8) => NO SE PUEDE => 7 = $0111_{|2}$ es el mayor positivo representable

Veamos...



Complemento a 2

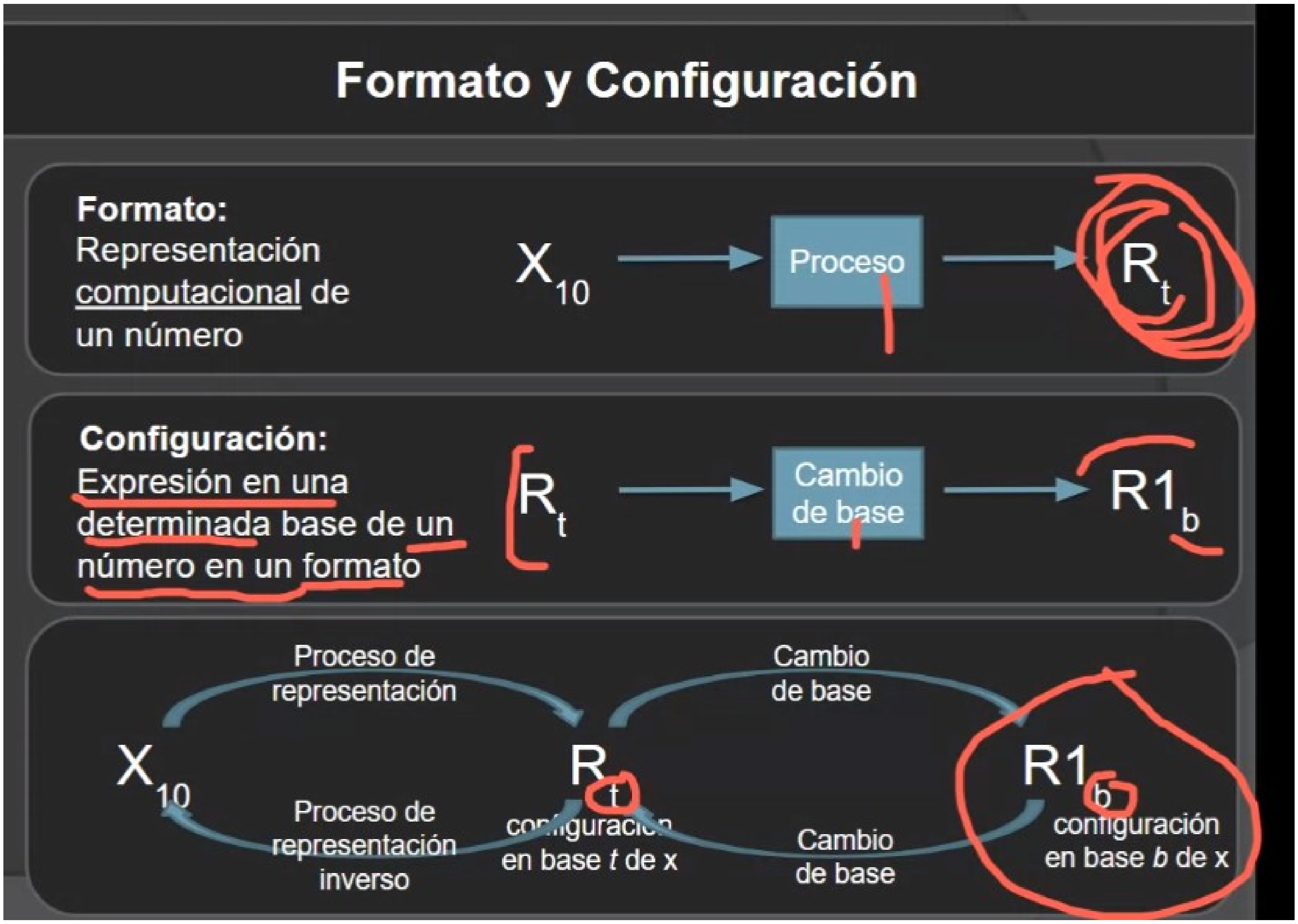
Paso por paso

Base = 2   precisión = 4
Representar -6

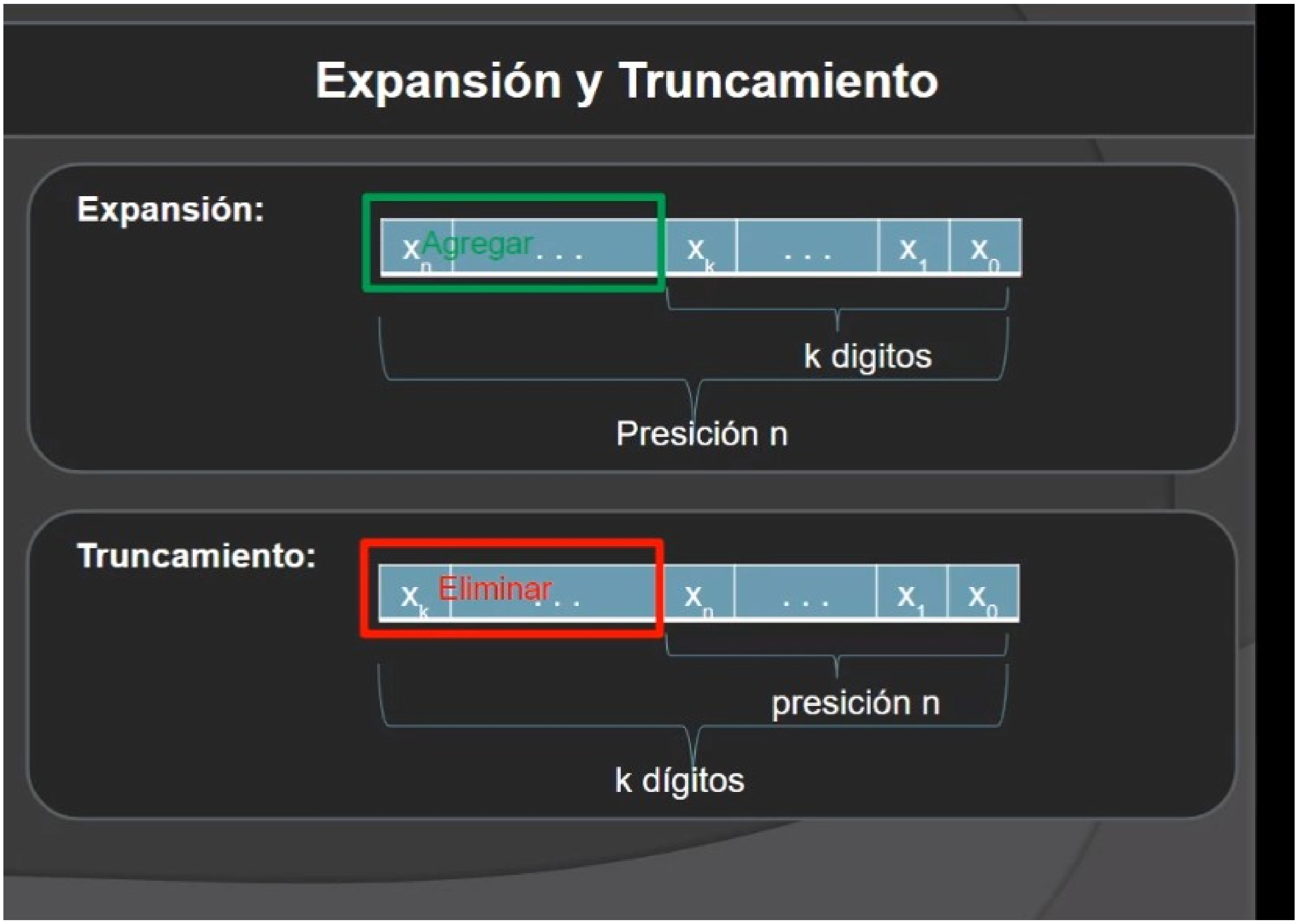
1. Pasar valor absoluto a base 2: $|-6_{10}| = 6_{10} = 110_2$
2. Completar con 0 a izquierda hasta completar n: 0110
3. Si es negativo, complementar (hacer NOT + 1): 1001 + 1 = 1010

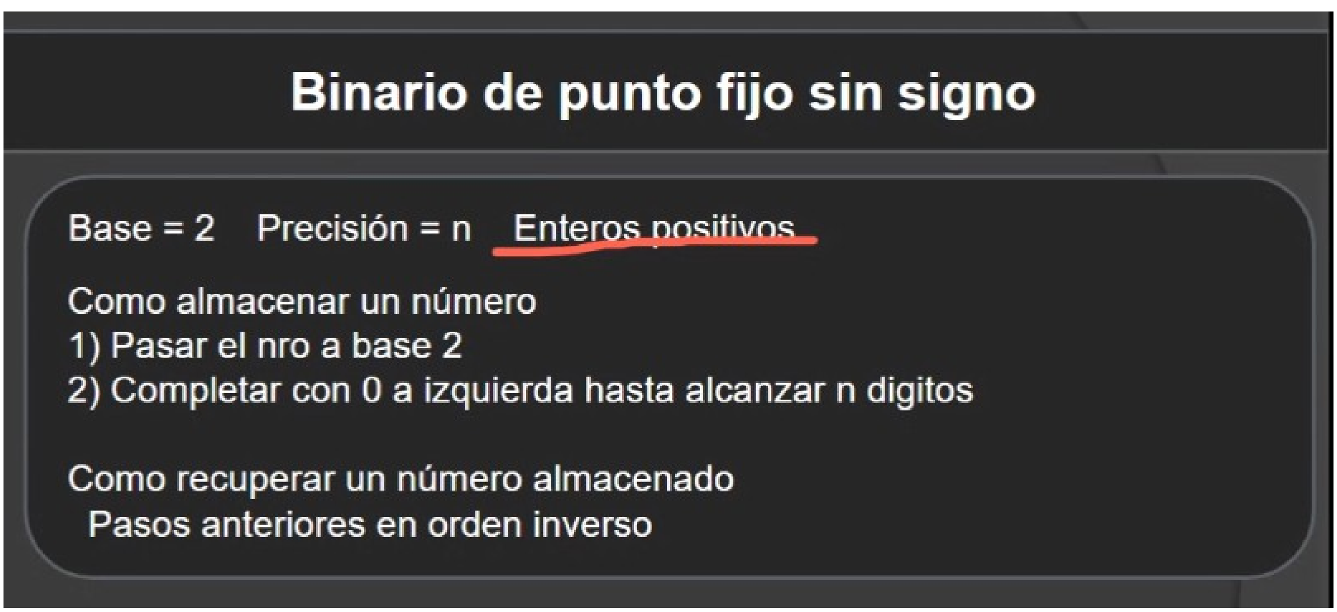
Indicar número almacenado en 1101

1. Si primer bit es 1 (es negativo), complementar (hacer NOT+1): 0010 + 1 = 0011
2. Pasar a base 10 los bits: $0011_2 = 3|_{10}$
3. Indicar el número según signo y valor obtenidos: -3


Formato y Configuración
Formato:
Representación computacional de un número
X10
Proceso
Rt
Configuración:
Expresión en una determinada base de un número en un formato
Rt
Cambio de base
R1b
Proceso de representación
Cambio de base
X10
Rt
R1b
Proceso de representación inverso
configuración en base t de x
Cambio de base
configuración en base b de x

A partir de una señal del mundo real, le doy formato para que lo lea la maquina. La configuración se la da para que lo pueda leer el usuario final. Se busca achicar o agrandar expresiones, según la necesidad.





Binario de punto fijo con signo

Base = 2   Precisión = n   Enteros positivos y negativos

Es la implementación del método complemento a 2

Como almacenar un número
1) Pasar el nro a base 2
2) Completar con 0 a izquierda hasta alcanzar n digitos
3) Si el nro es negativo, complementar usando método de "complemento a 2" (Not +1)

Como recuperar un número almacenado
1) Si el primer bit es 1 (es negativo), complementar.
2) Quitar 0 a izquierda.
3) Pasar a base 10 y colocar el signo que corresponda.

Binario de punto fijo con signo

Validación Overflow en operaciones aritméticas

```
B=2   n=4
```

Resolver 7 + 1

$7_{|10} = 0111_{|2}$
$1_{|10} = 0001_{|2}$

```
 0111
  0111
+ 0001
  ----
  1000
```

Ultimos 2 acarreos distintos
=> OVERFLOW

Resolver 7 - 1

$7_{|10} = 0111_{|2}$
$1_{|10} = 0001_{|2}$

Hallo C(1) para hacer 7+C(1)

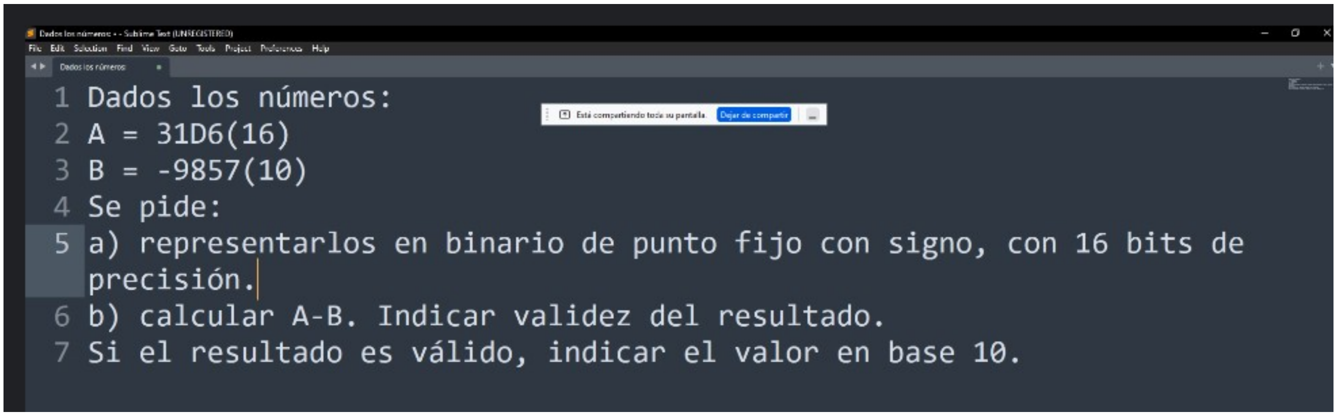
```
NOT(0001) = 1110
          +    1
            ----
            1111
```

comp

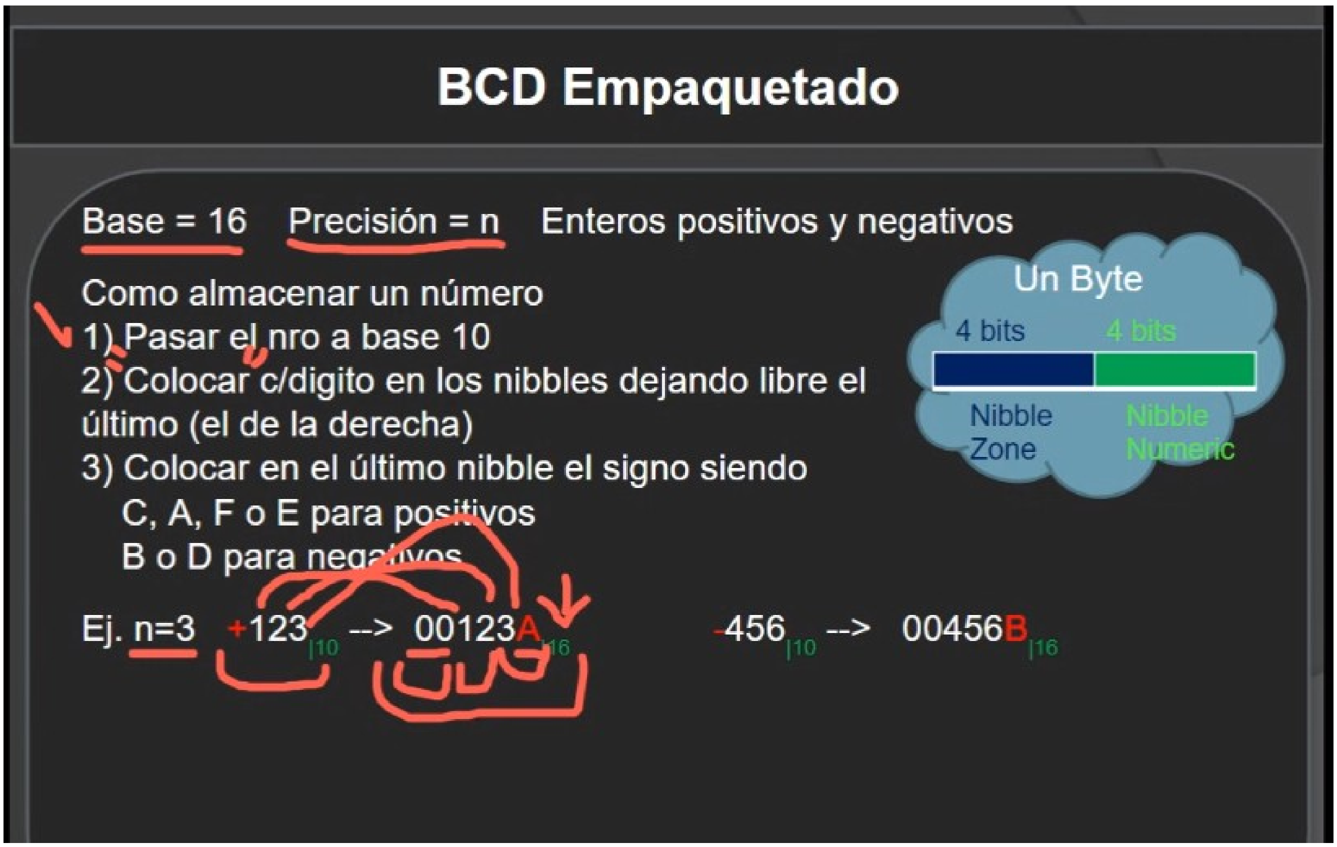
Ahora sumo

```
 1111
  0111
+ 1111
  ----
  0110
```

Ultimos 2 acarreos iguales
=> VALIDO


Dados los números:
A = 31D6(16)
B = -9857(10)
Se pide:
a) representarlos en binario de punto fijo con signo, con 16 bits de precisión.
b) calcular A-B. Indicar validez del resultado.
Si el resultado es válido, indicar el valor en base 10.

4 digitos binarios, es un digito hexagecimal. Se entiende con la práctica.
Atenti con la configuración y formato.



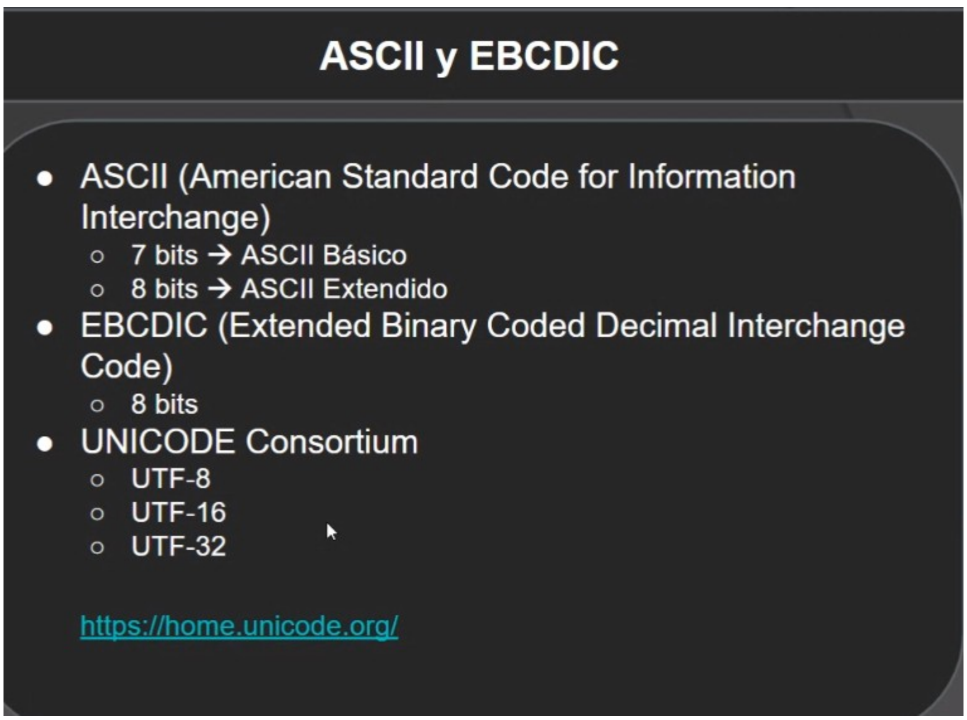
```
123|10

0000 0000 0001 0010 0011 1010 |2
lo paso a bas 16
0    0    1    2    3    A

0111 1011 <- la conf binaria del 123 en bpf c/s de 8 bits

  7B  <- la conf hexa del 123 en bpf c/s 8 bits

123A  <- la conf hexa del 123 en empaq de 2 bytes
0001 0010 0011 1010 <- conf binaria del 123 en empaq de 2
bytes
```



Archivo Edición Formato Ver Ayuda

```
Dado A que representa la configuración en base 8 de un número almacenado en formato BPF C/S de 24 bits y B que representa
la configuración en base 10 de otro número almacenado en formato zoneado de 3 bytes

A = 166140(8)
B = 16053714(10)

a. Indicar cuáles son los números almacenados en base 10.

i) Hallar la confg binaria del bpf c/s => paso de baso 8 a base 2

1   6   6   1   4   0(8) =
001 110 110 001 100 000(2)
000000001110110001100000



/**********************************
1   6   6   1   4   0(8)   bpf c/s 16 bits!!!!! supongamos
1 110 110 001 100 000(2)
/**********************************
```