Representación de Números

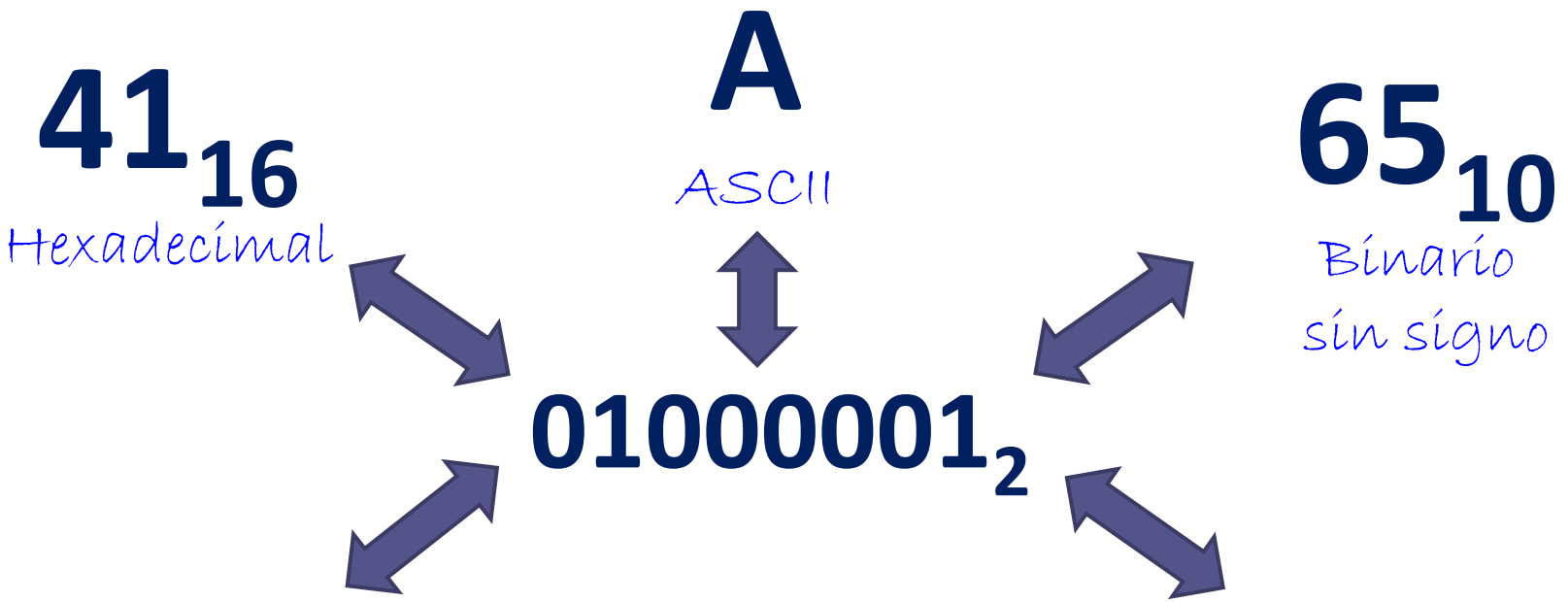
Explicación Práctica

Programación I - 2023

Facultad de Informática y Facultad de Ingeniería - UNLP

Representación de Números

Una cadena de bits (secuencia de dígitos binarios) puede tener múltiples interpretaciones (representar cosas distintas):



Representación de Números

Si tiene muchas interpretaciones, ¿Cómo sé cual es la correcta?

- El contexto de trabajo determina que solo una única representación es la correcta.

Como programadores cuando definimos el tipo de una variable establecemos:

- Que tipo de operaciones se pueden realizar.
- Como interpretar la secuencia de bits que componen la variable.
- Cuantos bits componen la secuencia.

Ejemplo:

- String , Char → Ascii
- Real, Single, Double → Punto Flotante
- Integer, Smallint, Shortint → Complemento a 2
- Cardinal, Word, Byte → Binario puro o sin signo

Rango y Resolución

Rango: es un intervalo que abarca el número “más pequeño” y el número “más grande” que se puede representar .

Resolución: es la diferencia que existe entre un número y su consecutivo. A veces la representación no es igual en todo el rango, por lo que se establece:

- ***Resolución extremo inferior:*** La diferencia más pequeña entre un número y su consecutivo.
- ***Resolución extremo superior:*** La diferencia más grande entre un número y su consecutivo.

Rango y Resolución

En las representaciones de números naturales y enteros (BSS, BCS, Ca1, Ca2, Ex2) la resolución es constante y vale 1.

En las representaciones de punto fijo la resolución es constante y depende de la cantidad de bits para representar los decimales.

En las representaciones de punto flotante la resolución es variable y es pequeña (mínima) para números pequeños y es grande (máxima) para números grandes.

Binario Puro o Sin Signo

En BSS (Binario sin signo) solo pueden representarse números positivos.

Para n bits puede representar 2^n números: 1 cero y 2^n-1 positivos.

Para n=10 ➔ 2^{10}=1024 ➔ 1 cero y 1023 positivos.

Rango: [0..1023]

Resolución: 1

Binario Puro o Sin Signo

Conversión BSS a Decimal: multiplicar cada dígito binario por 2 elevado a la posición que ocupa (en el sistema binario). Luego sumar todos los valores.

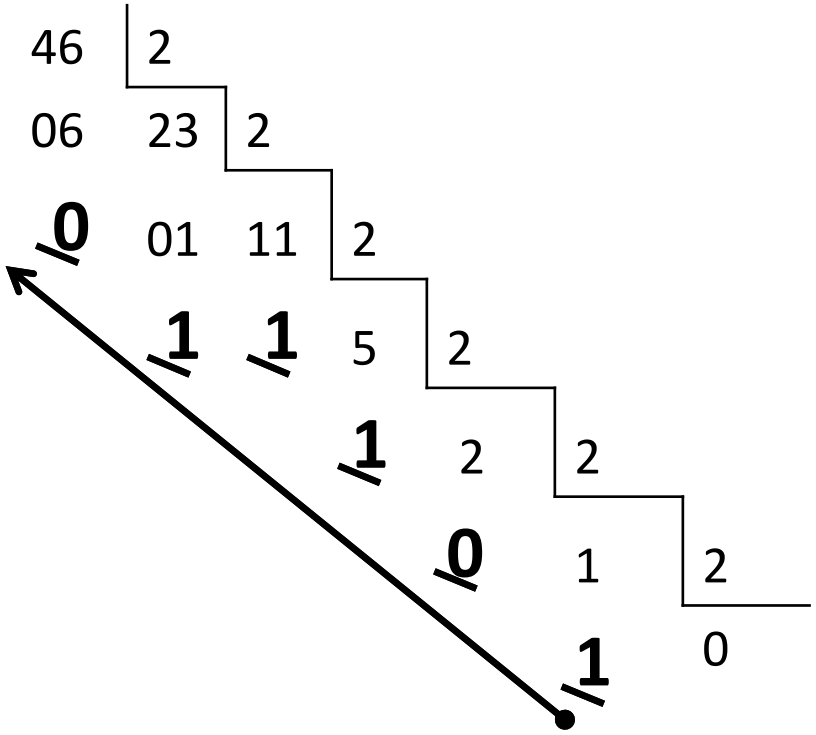
1 0 1 1 1 0

$1x2^5$ $0x2^4$ $1x2^3$ $1x2^2$ $1x2^1$ $0x2^0$

$32 + 0 + 8 + 4 + 2 + 0 = \mathbf{46_{10}}$

Binario Puro o Sin Signo

Conversión Decimal a BSS: dividir repetidamente el número decimal por 2 hasta que el cociente sea 0. Armar la secuencia de bits con todos los restos, comenzando por el último (es el más significativo o de mayor peso):



$101110_2 \rightarrow 46_{10}$

Módulo y Signo o Binario con Signo

En BCS (Binario Con Signo) pueden representarse números negativos y positivos. Tiene 2 representaciones de 0.

Separa signo (bit más a la izquierda) y mantisa (número).

Para n bits puede representar 2^n números: 2 ceros y 2^{n-1}-1 positivos y negativos.

Para n=10 ➔ 2^{10}=1024 ➔ 2 ceros y 511 positivos y 511 negativos.

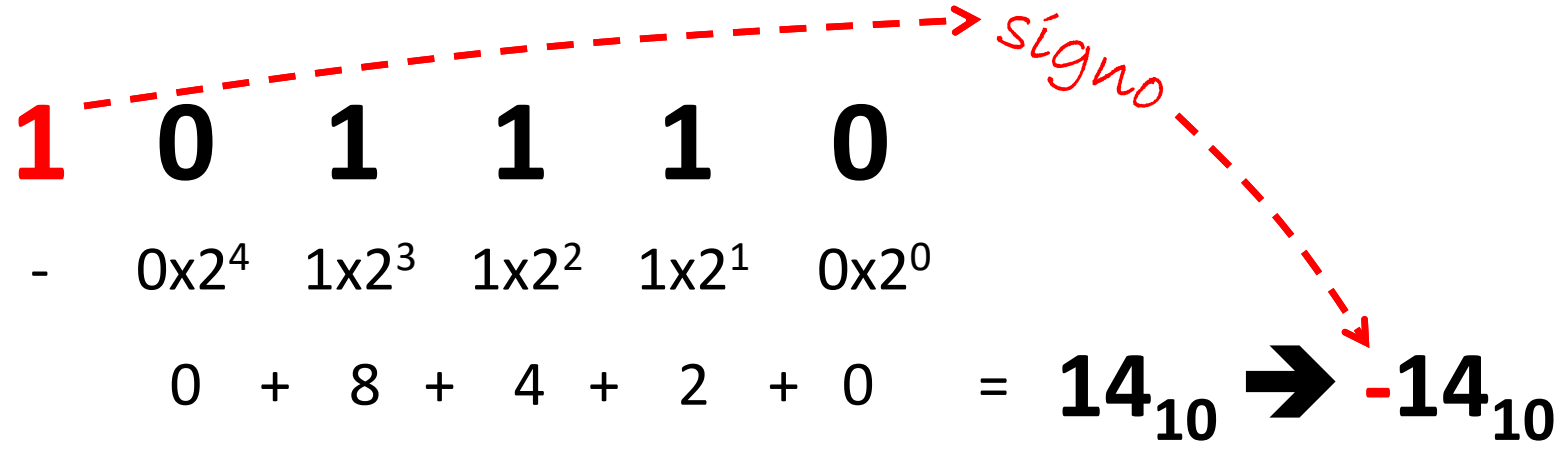
Rango: [-511..511]

Resolución: 1

Módulo y Signo o Binario con Signo

Conversión BCS a Decimal:

- positivo : convertir como BSS a decimal.
- negativo: ignorar el signo del número y convertir de igual forma que BSS a decimal. Agregar el signo al número decimal.



Módulo y Signo o Binario con Signo - Ejemplo

Dado un sistema BCS de 4 bits convertir a su representación decimal los siguientes números: 0100_2 , 0101_2 , 1000_2 , 1010_2

Como 0100_2 y 0101_2 son positivos convierto como BSS:

0100_2 → $1x2^2 + 0x2^1 + 0x2^0$ → 4 + 0 + 0 → 4

0101_2 → $1x2^2 + 0x2^1 + 1x2^0$ → 4 + 0 + 1 → 5

Como 1000_2 y 1010_2 son negativos, ignoro signo y convierto como BSS y luego agrego signo:

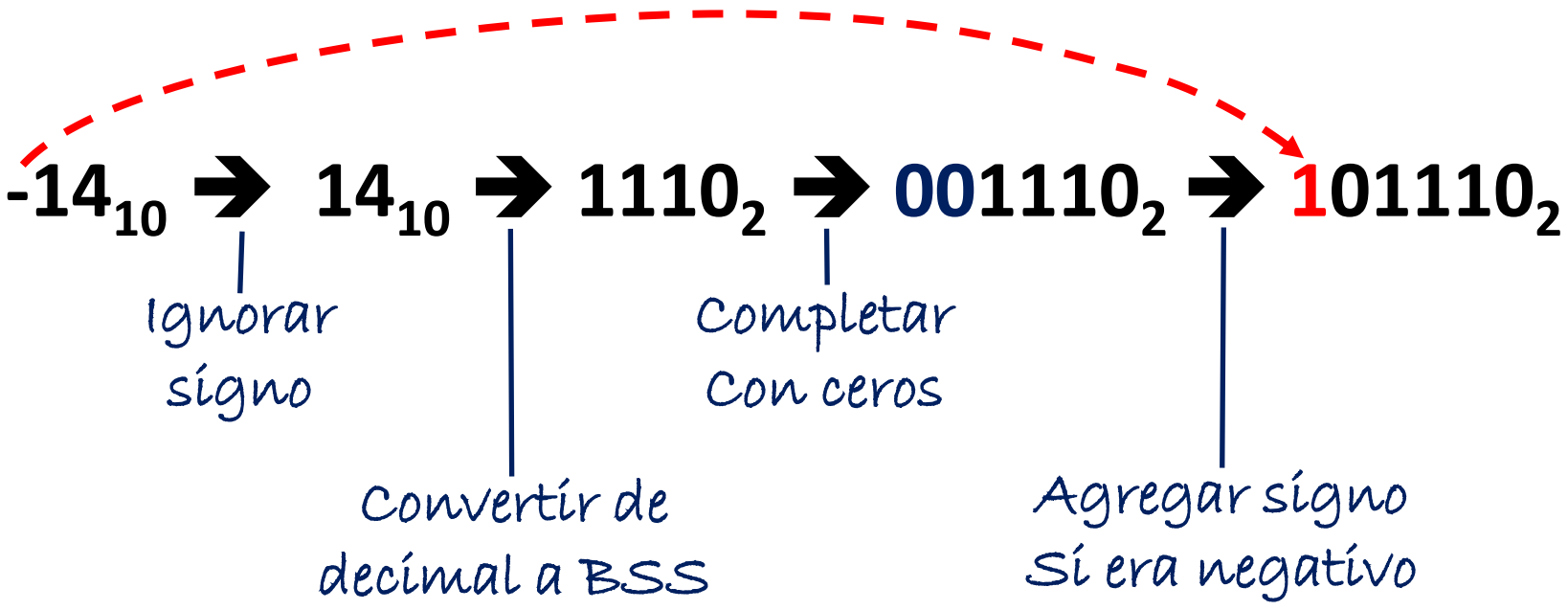
1000_2 → $0x2^2 + 0x2^1 + 0x2^0$ → 0 + 0 + 0 → 0 → -0 → 0

1010_2 → $0x2^2 + 1x2^1 + 0x2^0$ → 0 + 2 + 0 → 2 → -2

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

Módulo y Signo o Binario con Signo

Conversión Decimal a BCS: ignorar el signo del número decimal y convertir igual que de decimal a BSS. Agregar ceros a la izquierda hasta completar los bits del sistema (en el ejemplo 6 bits). Si el signo del número decimal es negativo reemplazar con un 1 el bit de más a la izquierda de número binario.



Módulo y Signo o Binario con Signo - Ejemplo

Convertir los siguientes números en decimal al sistema BCS de 4 bits: 8,6, -4

8 no se puede convertir porque no esta en el rango (-7 a 7)

Como 6 es positivo convierto como decimal a BSS:

6/2 = 3 R=0 ➔ 3/2 = 1 R=1 ➔ 1/2 = 0 R=1 ➔ 0110_2

Como -4 es negativo elimino el signo, convierto como decimal a BSS y luego cambio el signo:

4/2=2 R=0 ➔ 2/2=1 R=0 ➔ 1/2=0 R=1 ➔ 0100_2 ➔ 1100_2

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

Complemento a 1

En Ca1 (complemento a 1) pueden representarse números negativos y positivos. Tiene 2 representaciones de 0. El signo es el bit más izquierdo. 0 es positivo y 1 es negativo.

Para n bits puede representar 2^n números: 2 ceros y 2^{n-1}-1 positivos y negativos.

Para n=10 ➔ 2^{10}=1024 ➔ 2 ceros y 511 positivos y 511 negativos.

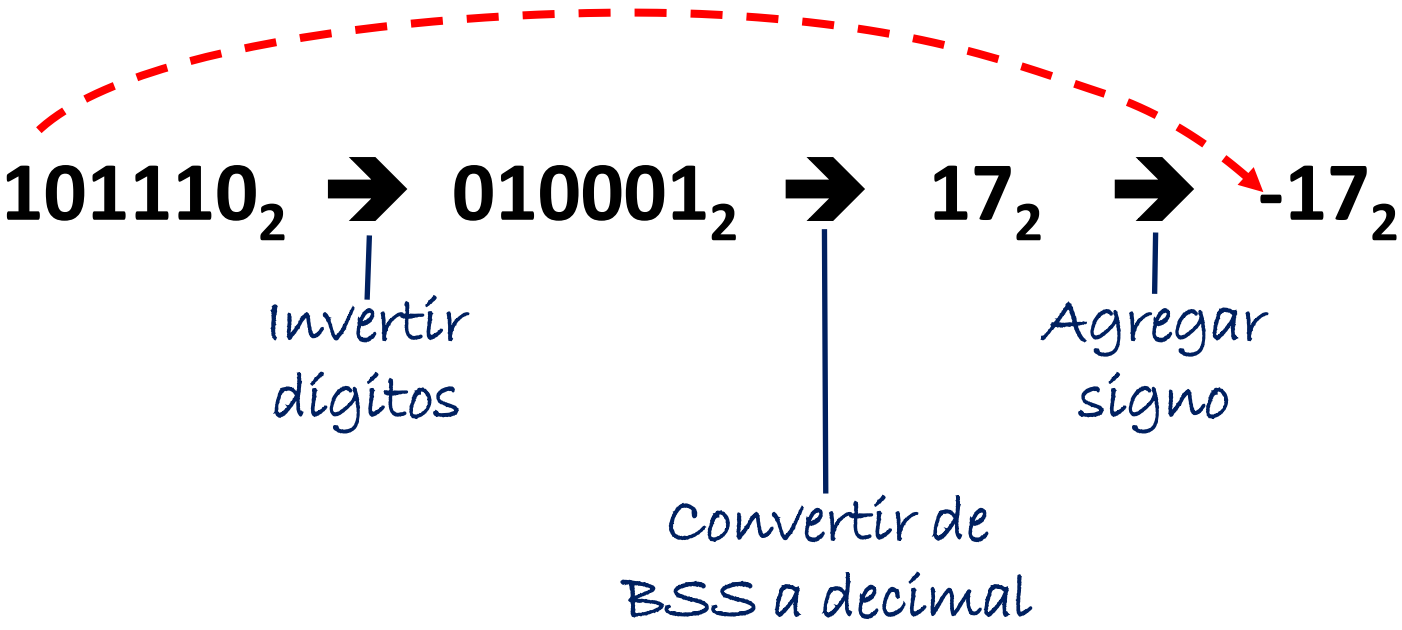
Rango: [-511..511]

Resolución: 1

Complemento a 1

Conversión Ca1 a Decimal – alternativa 1:

- positivo : convertir como BSS a decimal.
- negativo: invertir todos los bits. Convertir como BSS a decimal. Agregar signo a número decimal.



Complemento a 1

Conversión Ca1 a Decimal – alternativa 2:

sin importar si fuera negativo o positivo, se puede convertir de manera similar como BSS. Solo hay que modificar el factor por el que se multiplica el bit mas significativo. En vez de utilizar 2^{n-1} se debe utilizar $-(2^{n-1}-1)$.

En nuestro ejemplo como usamos 6 bits, n vale 6 por lo que el factor a multiplicar sería: $-(2^{n-1}-1) = -(2^5-1) = -31$

1 0 1 1 1 0

$1x-(2^5-1)$ $0x2^4$ $1x2^3$ $1x2^2$ $1x2^1$ $0x2^0$

$-31 + 0 + 8 + 4 + 2 + 0 = \mathbf{-17_{10}}$

Complemento a 1 - Ejemplo

Dado un sistema Ca1 de 4 bits convertir a su representación decimal los siguientes números: 0110_2 , 0011_2 , 1000_2 , 1010_2

Como 0011_2 y 0110_2 son positivos convierto como BSS:

0011_2 → $0x2^2 + 1x2^1 + 1x2^0$ → 0 + 2 + 1 → 3

0110_2 → $1x2^2 + 1x2^1 + 0x2^0$ → 4 + 2 + 0 → 6

Como 1000_2 y 1010_2 son negativos, invierto los bits, convierto como BSS y luego agrego signo:

1000_2 → 111_2 → $1x2^2 + 1x2^1 + 1x2^0$ → 4+2+1→ 7 → -7

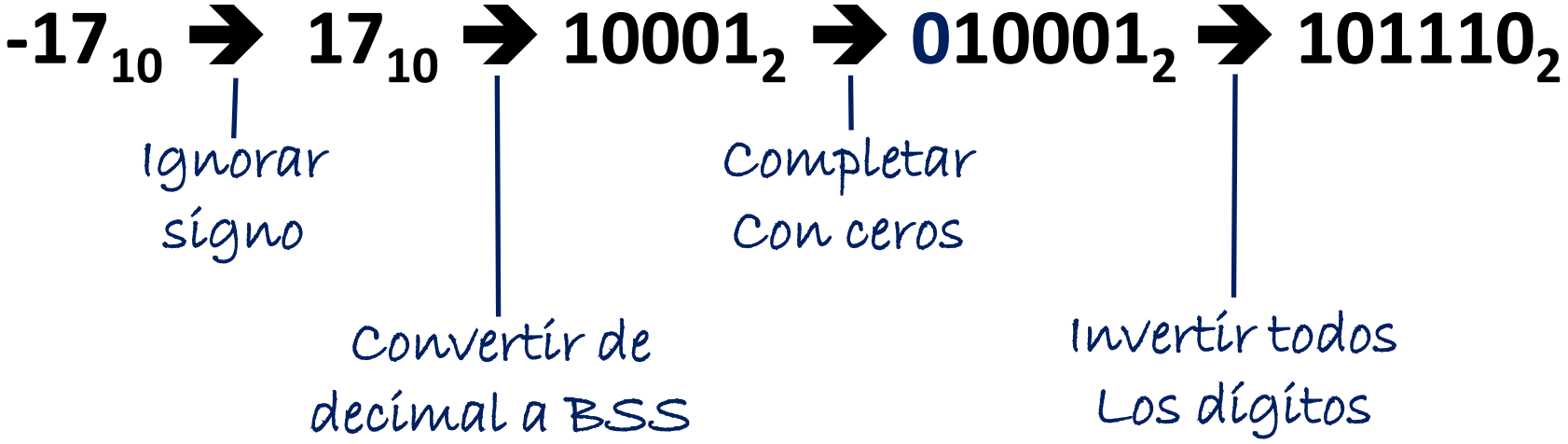
1010_2 → 101_2 → $1x2^2 + 0x2^1 + 1x2^0$ → 4+0+1→ 5 → -5

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-7
1001	-6
1010	-5
1011	-4
1100	-3
1101	-2
1110	-1
1111	-0

Complemento a 1

Conversión Decimal a Ca1:

- positivo : convertir como BSS a decimal.
- negativo: ignorar el signo del número decimal y convertir igual que de decimal a BSS. Agregar ceros a la izquierda hasta completar los bits del sistema (en el ejemplo 6 bits). Invertir todos los bits.



Complemento a 1 - Ejemplo

Convertir los siguientes números en decimal al sistema Ca1 de 4 bits: -8, 3, -4

-8 no se puede convertir porque no esta en el rango (-7 a 7)

Como 3 es positivo convierto como decimal a BSS:

3/2 = 1 R=1 ➔ 1/2 = 0 R=1 ➔ 0011_2

Como -4 es negativo elimino el signo, convierto como decimal a BSS y luego invierto todos los bits

4/2=2 R=0 ➔ 2/2=1 R=0 ➔ 1/2=0 R=1 ➔ 0100_2 ➔ 1011_2

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-7
1001	-6
1010	-5
1011	-4
1100	-3
1101	-2
1110	-1
1111	-0

Complemento a 2

En Ca2 (complemento a 2) pueden representarse números negativos y positivos. Tiene 1 representación de 0.

El signo es el bit más izquierdo. 0 es positivo y 1 es negativo.

Para n bits puede representar 2^n números: 1 cero y 2^{n-1}-1 positivos y 2^{n-1} negativos.

Para n=10 ➔ 2^{10}=1024 ➔ 1 cero y 511 positivos y 512 negativos.

Rango: [-512..511]

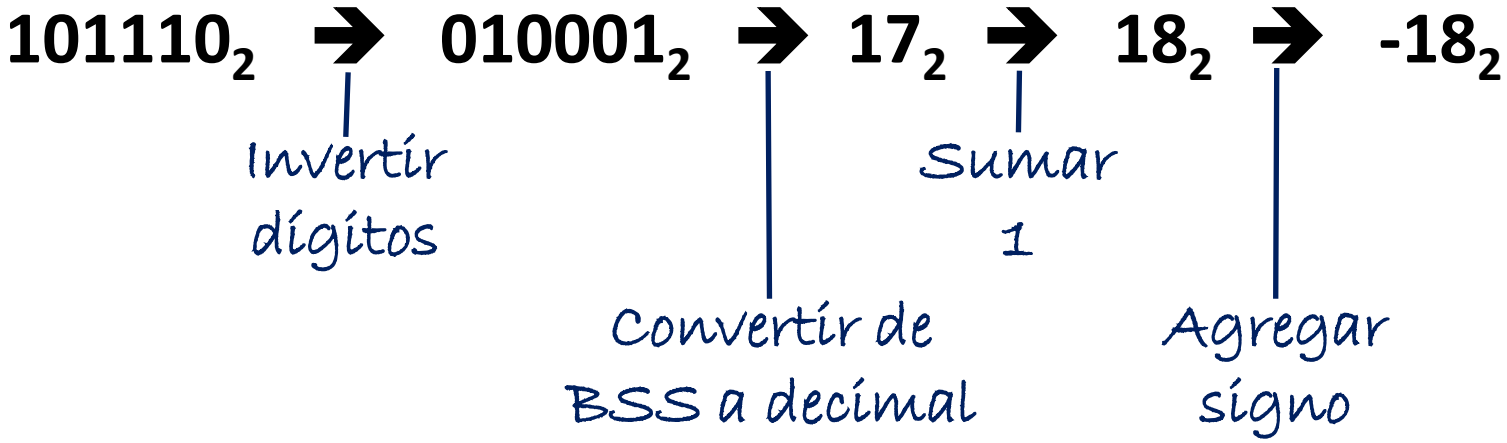
Resolución: 1

Complemento a 2

Conversión Ca2 a Decimal – alternativa 1:

- positivo : convertir como BSS a decimal.
- negativo: invertir todos los bits. Convertir como BSS a decimal. sumar 1. Agregar signo al número decimal.

Conversión Ca2 a Decimal para negativos:



Complemento a 2

Conversión Ca2 a Decimal – alternativa 2:

sin importar si fuera negativo o positivo, se puede convertir de manera similar como BSS. Solo hay que modificar el factor por el que se multiplica el bit mas significativo. En vez de utilizar 2^{n-1} se debe utilizar -2^{n-1}.

En nuestro ejemplo como usamos 6 bits, n vale 6 por lo que el factor a multiplicar sería: $-2^{n-1} = -2^5 = -32$

1	**0**	**1**	**1**	**1**	**0**
$1x-2^5$	$0x2^4$	$1x2^3$	$1x2^2$	$1x2^1$	$0x2^0$

$$-32 + 0 + 8 + 4 + 2 + 0 = \mathbf{-18_{10}}$$

Complemento a 2 - Ejemplo

Dado un sistema Ca2 de 4 bits convertir a su representación decimal los siguientes números: 0110_2 , 0011_2 , 1111_2 , 1010_2

Como 0011_2 y 0110_2 son positivos convierto como BSS:

0011_2 → $0x2^2 + 1x2^1 + 1x2^0$ → 0 + 2 + 1 → 3

0110_2 → $1x2^2 + 1x2^1 + 0x2^0$ → 4 + 2 + 0 → 6

Como 1111_2 y 1010_2 son negativos, invierto los bits, sumo 1, convierto como BSS y luego agrego signo:

1111_2 → 000_2 → 001_2 → $0x2^2 + 0x2^1 + 1x2^0$ → 1 → -1

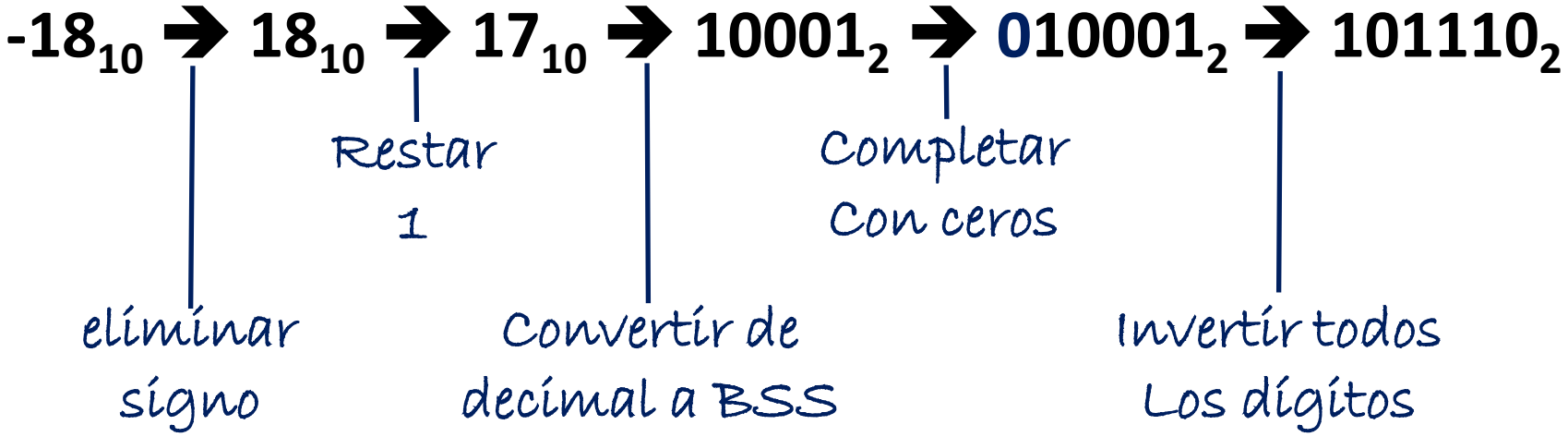
1010_2 → 101_2 → 110_2 → $1x2^2 + 1x2^1 + 0x2^0$ → 6 → -6

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

Complemento a 2

Conversión Decimal a Ca2 – alternativa 1:

- positivo : convertir como Decimal a BSS.
- negativo: eliminar el signo del número decimal. Restar 1. Convertir igual que de decimal a BSS. Agregar ceros a la izquierda hasta completar los bits del sistema (en el ejemplo 6 bits). Invertir todos los bits.



Complemento a 2 - Ejemplo

Convertir los siguientes números en decimal al sistema Ca2 de 4 bits: -8, 2, -4

-8 SI se puede convertir porque esta en el rango (-8 a 7)

Como 2 es positivo convierto como decimal a BSS:

2/2 = 1 R=0 ➔ 1/2 = 0 R=1 ➔ 0010_2

Como -4 y -8 son negativos elimino el signo, resto 1 convierto como decimal a BSS y luego invierto todos los bits

-4 ➔ 4 ➔ 3 ➔ 3/2=1 R=0 ➔ 1/2=0 R=1 ➔ 0011_2 ➔ 1100_2

-8 ➔ 8 ➔ 7 ➔ 7/2=3 R=1 ➔ 3/2=1 R=1 ➔ 1/2=0 R=1 ➔

➔ 0111_2 ➔ 1000_2

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

Exceso a 2

En Ex2 (Exceso a 2) pueden representarse números negativos y positivos. Tiene 1 representación del 0.

El signo es el bit más izquierdo. 1 es positivo y 0 es negativo (Al revés que en BCS, Ca1 y Ca2)

Para n bits puede representar 2^n números: 1 ceros y 2^{n-1}-1 positivos y 2^{n-1} negativos (Igual que Ca2).

Para n=10 ➔ 2^{10}=1024 ➔ 1 cero y 511 positivos y 512 negativos.

Rango: [-512..511]

Resolución: 1

Exceso a 2 - Conversión Ex2 a Decimal

La conversión es igual cuando el número es negativo o positivo.

Para realizar la conversión hay que determinar el número de exceso. Para n bits el exceso es 2^{n-1} (para nuestro ejemplo de 6 bits n es 6, el exceso es $2^{6-1} = 2^5 = 32$).

Conversión Ex2 a Decimal: convertir igual que de decimal a BSS. Restar el número de exceso al valor decimal.

$$101110_2 \rightarrow 46_{10} \rightarrow 14_{10}$$

Convertir de BSS a decimal

Restar Exceso (32)

Exceso a 2 - Ejemplo

Dado un sistema Ex2 de 4 bits convertir a su representación decimal los siguientes números: 0011_2 , 0110_2 , 1111_2 , 1010_2

El exceso del sistema es $2^{4-1} = 2^3 = 8$

Positivos y negativos se tratan igual: convierto como BSS a decimal y resto el exceso:

- 0011_2 → $0x2^3+0x2^2+1x2^1+1x2^0$ → 0+0+2+1 → 3 → -5
- 0110_2 → $0x2^3+1x2^2+1x2^1+0x2^0$ → 0+4+2+0 → 6 → -2
- 1111_2 → $1x2^3+1x2^2+1x2^1+1x2^0$ → 8+4+2+1 → 15 → 7
- 1010_2 → $1x2^3+0x2^2+1x2^1+0x2^0$ → 8+0+2+0 → 10 → 2

0000	-8
0001	-7
0010	-6
0011	-5
0100	-4
0101	-3
0110	-2
0111	-1
1000	0
1001	1
1010	2
1011	3
1100	4
1101	5
1110	6
1111	7

Exceso a 2 - Conversión Decimal a Ex2

La conversión es igual cuando el número es negativo o positivo.

Para realizar la conversión hay que determinar el número de exceso. Para n bits el exceso es 2^{n-1} (para nuestro ejemplo de 6 bits n es 6, el exceso es $2^{6-1} = 2^5 = 32$).

Conversión Decimal a Ex2: Sumar el número de exceso al valor decimal. Convertir igual que de decimal a BSS.

14_{10} → 46_{10} → 101110_2

Sumar Exceso (32)

Convertir de Decimal a BSS

Exceso a 2 - Ejemplo

Convertir los siguientes números en decimal al sistema Ex2 de 4 bits: -8, 2, -4

El exceso del sistema es $2^{4-1} = 2^3 = 8$

Positivos y negativos se tratan igual: sumo el exceso y convierto como de decimal a BSS:

2 ➔ 10 ➔ 10/2=5 R=0 ➔ 5/2=2 R=1➔ 2/2=1 R=0 ➔
➔ 1/2=0 R=1 ➔ 1010_2

-8 ➔ 0 ➔ 0/2=0 R=0 ➔ 0000_2

-4 ➔ 4 ➔ 4/2=2 R=0➔ 2/2=1 R=0 ➔ 1/2=0 R=1 ➔ 0100_2

0000	-8
0001	-7
0010	-6
0011	-5
0100	-4
0101	-3
0110	-2
0111	-1
1000	0
1001	1
1010	2
1011	3
1100	4
1101	5
1110	6
1111	7

Suma y Resta de Números Binarios

- Las sumas y las restas en binario funcionan igual que en decimal, solo que hay que tener en cuenta que hay 2 dígitos en vez de 10

- Hay que tener en cuenta que la representación del los bits en las computadoras es finita

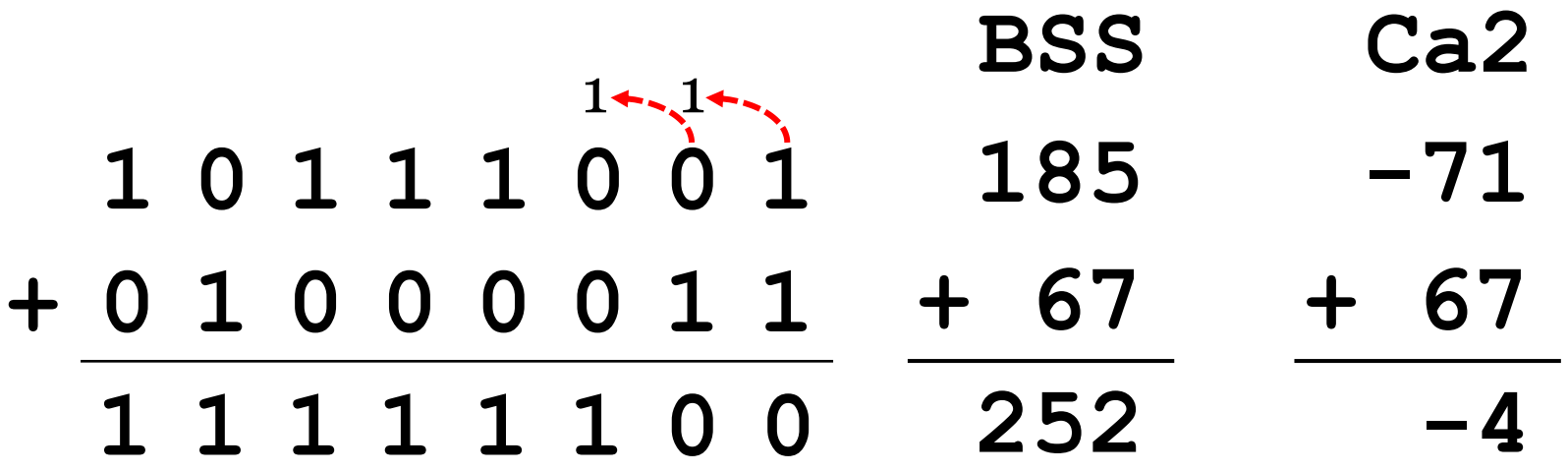
- Cuando hacemos operaciones matemáticas sobre una cantidad limitada de bits pude suceder que el resultado requiera mas bits que los disponibles

Suma y Resta de Números Binarios

- Es importante destacar que tanto la suma como la resta de 2 números en representación BSS y CA2 se realiza de la misma manera y da resultados correctos
- La mayor parte de las computadoras trabajan con ambas representaciones
- En un lenguaje de alto nivel, el tipo de una variable indica al compilador el tipo de representación y la implementación de las operaciones correspondiente:
  - **integer / int** define representación en CA2
  - **cardinal / unsigned int** define representación en BSS

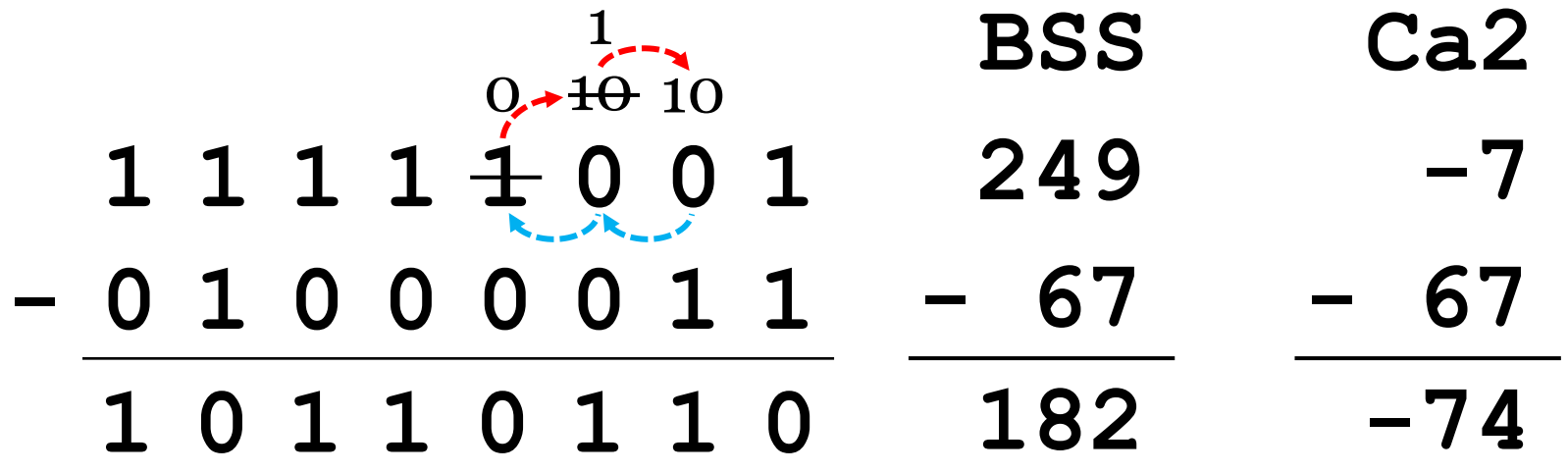
Suma de Números Binarios

- Para ver un ejemplo asumamos una representación de 8 bits (BSS o CA2):



Resta de Números Binarios

- Para ver un ejemplo asumamos una representación de 8 bits (BSS o CA2):



Flags o Banderas de Estado

- Los flags son bits que el procesador modifica cada vez que se ejecuta una operación aritmética o lógica

- Se utilizan para:
  - Determinar relaciones entre 2 valores ordinales (números o caracteres) como menor, igual o mayor
  - Verificar errores o condiciones para cambiar el rumbo del programa a través de instrucciones de salto condicional

Flags o Banderas de Estado – Cero (Z)

- El flag Z indica que el resultado de una operación es cero:
  - Z = 1 cuando el resultado es 0 (todos los bits son 0)
  - Z = 0 cuando el resultado es 1 (alguno de los bits no es 0)
- Se utiliza para:
  - determinar si 2 valores son iguales ya que la resta de 2 números son iguales
  - en lenguaje ensamblador, en bucles (tipo for) donde se va decrementando un valor se utiliza para condición de salida

Flags o Banderas de Estado – Negativo (N)

- El flag N indica que el resultado de una operación es negativo:
  - N = 1 cuando el bit mas significativo del resultado es 1
  - N = 0 cuando el bit mas significativo del resultado es 0

- Se utiliza para:
  - determinar la desigualdad de 2 valores en Ca2. La resta de A – B da negativa cuando B es mayor que A o A es menor que B

Flags o Banderas de Estado – Acarreo (C)

- El flag C indica que se produjo carry (acarreo) en una suma o borrow (prestamo) en una resta. C = 1 cuando:
  - La suma de los bits mas significativos de dos números produce 1 bit de acarreo
  - La resta de los bits mas significativos de dos números requiere 1 bit prestado
- El flag de Carry se utiliza para:
  - determinar la desigualdad de 2 valores en BSS. La resta de A – B da borrow cuando B es mayor que A o A es menor que B

Flags o Banderas de Estado – Acarreo (C)

- El flag de Carry se utiliza para:
  - Para determinar que un resultado de suma o resta es incorrecto en complemento BSS
- Ejemplos de suma y resta para BSS en 8 bits (tener en cuenta que el rango está entre 0 y 255) :

C=1 (Acarreo)

10000000	128
+ 10000000	+ 128
00000000	0

El resultado debe ser 256 y es 0

C=1 (Borrow)

10000000	128
- 10000001	- 129
11111111	255

El resultado debe ser -1 y es 255

Flags o Banderas de Estado – Overflow (V)

- El flag V indica que se produjo un desborde (faltan bits) en la representación. V = 1 cuando:
  - En la suma, los bits de signo de los operandos son iguales pero distintos del bit de signo del resultado.
  - En la resta, los bits de signo del minuendo y el sustraendo son diferentes y el bit de signo del resultado es igual al del sustraendo.
  - Notar que si se piensa en la resta como la suma del opuesto, se puede usar la regla de los signos de la suma que es más fácil de recordar

Flags o Banderas de Estado – Overflow (V)

- Ejemplos de suma y resta para Ca2 en 8 bits (tener en cuenta que el rango está entre -128 y 127) :

V=1 (Overflow)

```
  01000000     64
+ 01000000   + 64
----------   ----
  10000000   -128
```

El resultado debe ser 128 y es -128

V=1 (Overflow)

```
  10000000   -128
- 00000001 -    1
----------   ----
  01111111    127
```

El resultado debe ser -129 y es 127

Representación en Punto Flotante

El objetivo es poder representar números muy pequeños y muy grandes de una manera flexible

- número muy pequeño: $1{,}23 \times 10^{-10}$
- número muy grande : $1{,}23 \times 10^{10}$

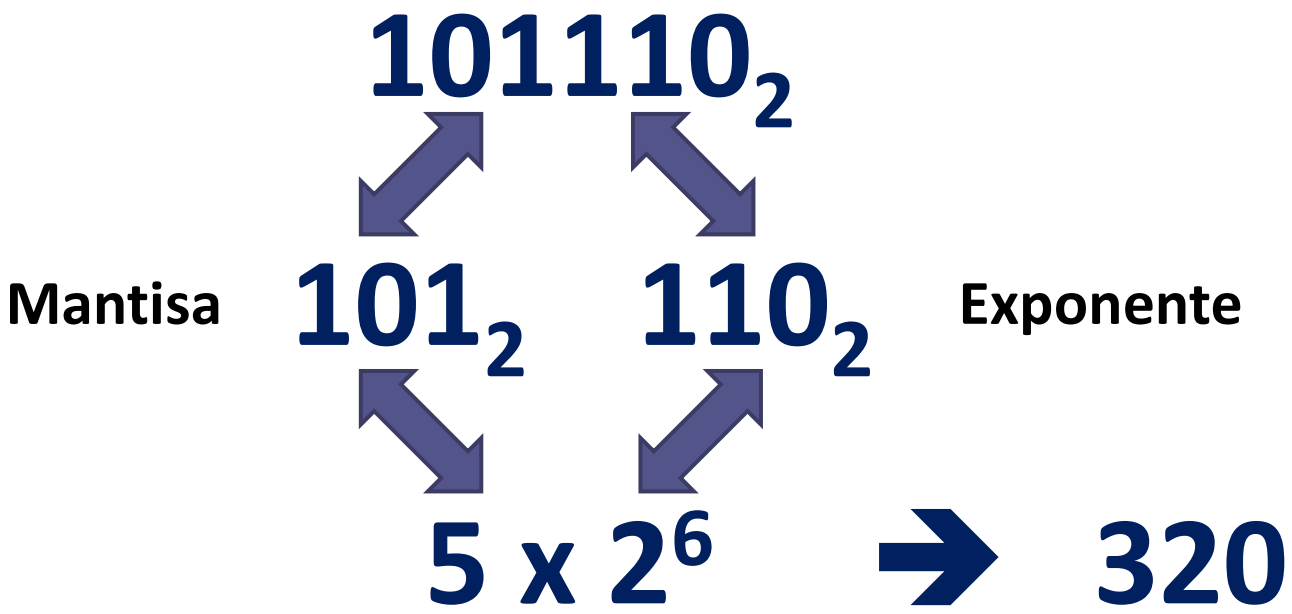
Si utilizamos un sistema binario, la forma general para expresar este tipo de número es : $M \times 2^{E}$

- M se denomina mantisa y puede estar representada en BSS, BCS, Ca1, Ca2, Ex2 o fraccionaria (sin normalizar, normalizada o normalizada con bit implícito)
- E es el exponente y puede estar representada en BSS, BCS, Ca1, Ca2, Ex2.

Representación en Punto Flotante

En esta representación un número binario está compuesto por un grupo de bits que representan la mantisa y un grupo de bits que representan al exponente.

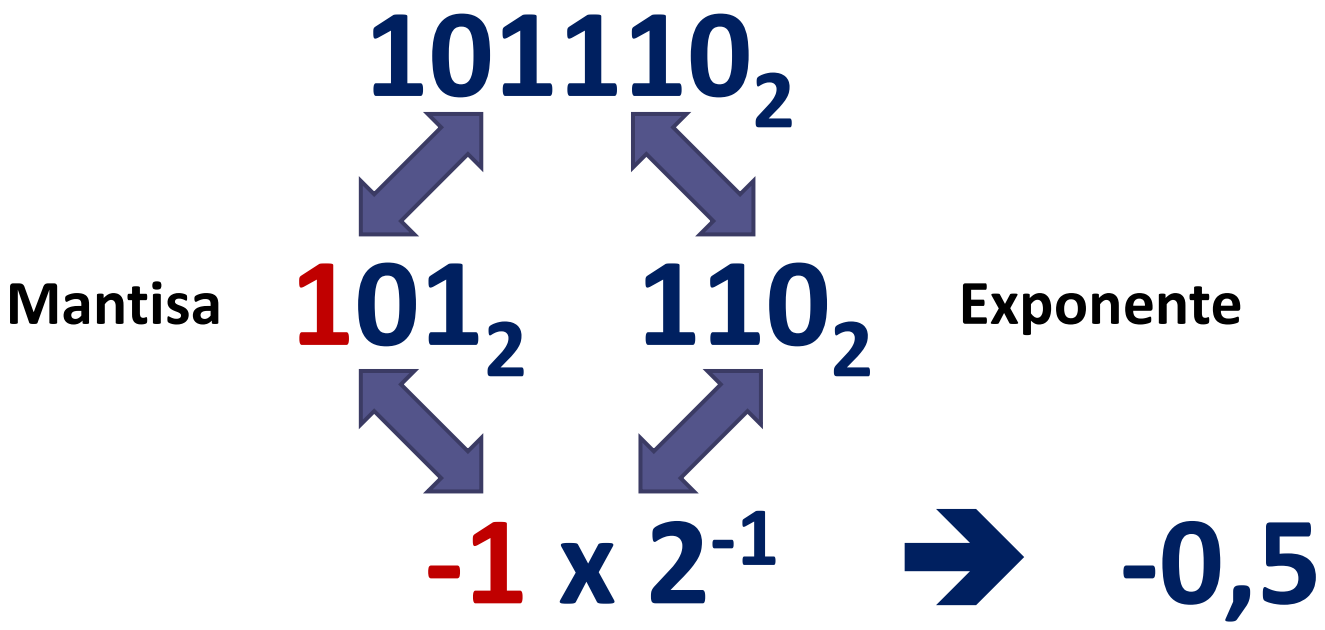
Por ejemplo en un sistema con mantisa en BSS de 3 bits y exponente BSS de 3 bits



Representación en Punto Flotante

Para ver otro ejemplo tomemos la misma secuencia de bits y cambiemos la representación de la mantisa y el exponente.

Por ejemplo en un sistema con mantisa en BCS de 3 bits y exponente Ca1 de 3 bits



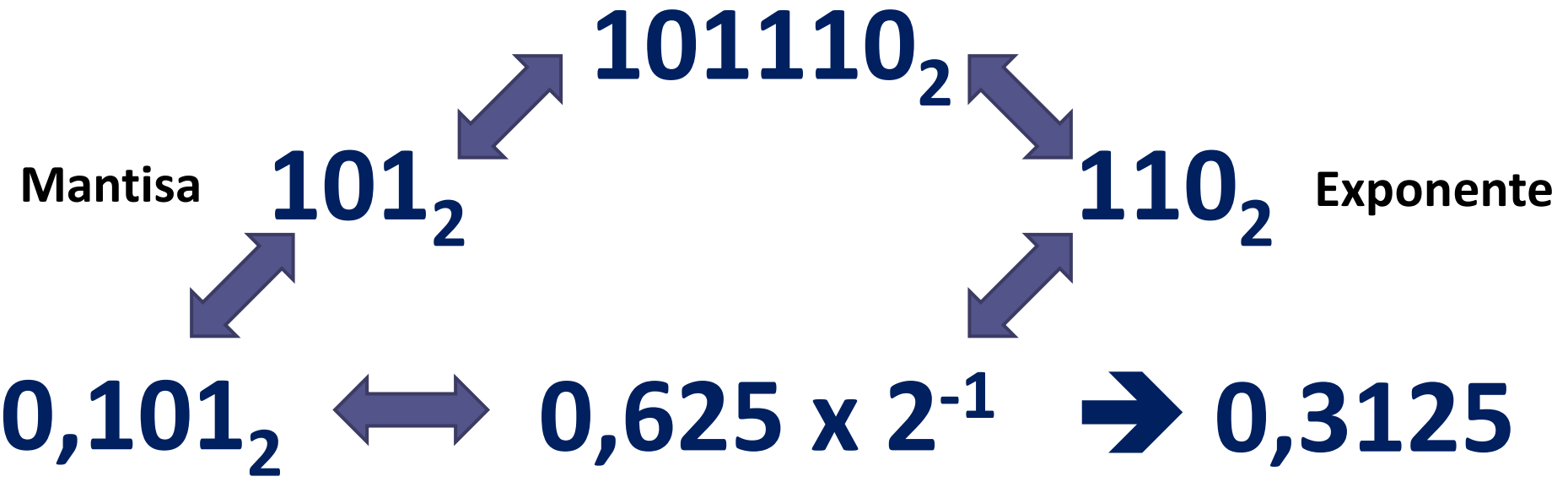
Punto Flotante – Mantisa Fraccionaria

La mantisa fraccionaria asume un punto decimal implícito a la izquierda del primer bit.

Los dígitos representan fracciones de potencias binarias.

Por ejemplo, la mantisa 10011_2 equivale a $0,10011_2$ que equivale a $1x2^{-1} +1x2^{-4} +1x2^{-5} = 0,5 + 0,0625 + 0,03125 = 0,59375$

Por ejemplo en un sistema con mantisa en BSS de 3 bits y exponente Ca1 de 3 bits



Mantisa Fraccionaria Normalizada

Desventaja: La mantisa fraccionaria (sin normalizar) tiene muchas representaciones de un mismo número.

Por ejemplo en un sistema con mantisa en BSS de 3 bits y exponente BCS de 3 bits tenemos:

100000_2 ➔ $(1x2^{-1}) \times 2^0$ ➔ 0,5 x 1 ➔ 0,5

010001_2 ➔ $(1x2^{-2}) \times 2^1$ ➔ 0,25 x 2 ➔ 0,5

001010_2 ➔ $(1x2^{-4}) \times 2^2$ ➔ 0,125 x 4 ➔ 0,5

Una mantisa normalizada es aquella cuyo bit izquierdo es 1. Cualquier otra representación que comience con 0 se considera inválida. Esto asegura que una representación única de cada número (y por lo tanto comparar 2 números por igual es simple).

Mantisa Fraccionaria Normalizada

La mantisa fraccionaria (sin normalizar) puede ser normalizada. Dada la siguiente mantisa que se quiere normalizar:

010001_2 ➔ $(1x2^{-2}) \times 2^1$ ➔ 0,25 x 2 ➔ 0,5

Desplazamos la mantisa a izquierda (equivale a multiplicar x 2):

100001_2 ➔ $(1x2^{-1}) \times 2^1$ ➔ 0,50 x 2 ➔ 1

Ahora hay que ajustar el exponente para conservar el número original. Restamos 1 al exponente (equivale a dividir x 2):

100000_2 ➔ $(1x2^{-1}) \times 2^0$ ➔ 0,50 x 1 ➔ 0,5

Mantisa Normalizada con bit implícito

Desventaja: La mantisa fraccionaria normalizada desperdicia el bit más izquierdo, ya que este siempre debe valer 1.

Como todas las mantisas deben estar normalizadas, al punto decimal implícito se le agrega el uno normalizado para aprovechar el bit izquierdo.

Por ejemplo, la mantisa normalizada con bit implícito 011000_2 ➔ $0,1011000_2$ ➔ $1x2^{-1}+1x2^{-3}+1x2^{-4} = 0,5+0,125+0,0625 = 0,6875$

Cabe notar que la representación de la mantisa tiene 6 bits, pero al momento de interpretar su valor tiene 7 (el bit implícito).

Suma en Punto Flotante

Para sumar números en punto flotante se deben igualar los exponentes y ajustar las mantisas antes de sumar. ¿Por qué?

Si dos números tienen el mismo exponente, se puede realizar la suma en binario de manera convencional

Si dos mantisas A y B tienen el mismo exponente i, entonces:

$$\mathbf{A \times 2^i + B \times 2^i = (A + B) \times 2^i}$$

Ejemplo de mantisa fraccionaria y exponente BSS de 3 bits :

$100001_2 \rightarrow (1 \times 2^{-1}) \times 2^1 \rightarrow 0{,}5 \times 2 \rightarrow 1{,}0$

$+ \; 010001_2 \rightarrow (1 \times 2^{-2}) \times 2^1 \rightarrow 0{,}25 \times 2 \rightarrow + \; 0{,}5$

$110001_2 \rightarrow (1 \times 2^{-1} + 1 \times 2^{-2}) \times 2^1 \rightarrow 0{,}75 \times 2 \rightarrow 1{,}5$

Suma en Punto Flotante

Resolver la siguiente suma para un sistema de punto flotante con mantisa fraccionaria en BSS de 6 bits y exponente en BCS 4 bits:

1000001001 + 1000000000

Interpretamos 1000001001:

1000001001 ➔ $(1x2^{-1}) \times 2^{-1}$ ➔ 0,5 x 0,5 ➔ 0,25

Interpretamos 0100000000:

1000000000 ➔ $(1x2^{-2}) \times 2^{0}$ ➔ 0,5 x 1 ➔ 0,50

Sabemos que el resultado debe ser 0,25+0,50 ➔ 0,75

Suma en Punto Flotante

Para resolver la suma hay 2 alternativas: igualar los exponentes al del primer operando o igualar los exponentes al segundo operando.

Primera alternativa: igualando el exponente de 1000001001 (0,25) a 0000

Llevamos 1001 a 0000 sumando 1 (equivale a multiplicar por 2):

1000000000 ➔ $(1x2^{-1}) \times 2^{0}$ ➔ 0,5 x 1 ➔ 0,50

Para mantener el número original, ajustamos la mantisa desplazando a derecha (equivale a dividir por 2):

0100000000 ➔ $(1x2^{-2}) \times 2^{0}$ ➔ 0,25 x 1 ➔ 0,25 (valor original)

Suma en Punto Flotante

Realizamos la suma:

$$\begin{array}{rll} 0100000000 & \rightarrow & 0{,}50 \\ + \; 1000000000 & \rightarrow & + \; 0{,}25 \\ \hline 1100000000 \rightarrow (1\times2^{-1} + 1\times2^{-2}) \times 2^{0} \rightarrow 0{,}75 \times 1 & \rightarrow & 0{,}75 \end{array}$$

El resultado es correcto!

Suma en Punto Flotante

Segunda alternativa: igualando el exponente de 1000000000 (0,5) a 1001

Llevamos 0000 a 1001 restando 1 (equivale a dividir por 2):

1000001001 ➔ $(1x2^{-1}) \times 2^{-1}$ ➔ 0,5 x 0,5 ➔ 0,25

Para mantener el número original, ajustamos la mantisa desplazando a izquierda (equivale a multiplicar por 2):

0000001001 ➔ $(0) \times 2^{-1}$ ➔ 0 ➔ 0 (valor original es 0,5)

Se pierde precisión! No conviene esta alternativa

Hay que elegir la que pierde menos dígitos

A veces con las 2 alternativas se pierde precisión

¿Preguntas?