UNIVERSIDAD POLITECNICA DE VALENCIA

etsinf

DISCA

Fundamentos de los computadores

Tema 6. Representación de la información

Objetivos

- Conocer los sistemas binario, octal y hexadecimal y sus relaciones.
- Conocer las operaciones aritméticas básicas en el sistema binario.
- Conocer las distintas formas de representación de la información en el computador

Índice

- Sistemas de numeración
  - Binario. Recordatorio
- Operaciones binarias básicas
- Representación de números enteros
  - Convenio de representación: Signo y Magnitud
  - Convenio de representación: Complemento a 2
  - Convenio de representación: Exceso Z
- Representación de números reales
  - Coma fija
  - Coma flotante. Formato estándar IEEE 754

Introducción

- Representación externa
  - Empleada por las personas: sistema decimal, caracteres alfanuméricos, gráficos, etc.

- Representación interna
  - Utilizada por el ordenador: sistema binario, código ASCII para los caracteres, etc.

Sistemas de numeración: recordatorio FCO

- Conceptos a recordar de los sistemas de numeración
  - Sistemas de numeración posicionales, base, factor de escala
  - Sistema binario: sistema posicional de base 2

- Cambios de base: de una base B cualquiera a decimal
  - Desarrollo del polinomio de potencias de la base
    - $a_{-f}\ a_{-f+1} \ldots a_{-1}$ es la parte fraccionaria (f dígitos)
    - $a_0\ a_1 \ldots a_{e-1}$ es la parte entera (e dígitos)

$$N = \sum_{i=-f}^{e-1} a_i B^i$$

- Cambios de base: de decimal a una base B cualquiera
  - Divisiones sucesivas entre la base B para la parte entera
  - Multiplicaciones sucesivas por la base B para la parte fraccionaria
  - La coma separa en las dos bases la parte entera de la fraccionaria

Sistemas de numeración

- Además del sistema binario, se utilizan también:
  - Octal (base 8 = 2^3)
    - **Cada dígito octal representa un grupo de exactamente 3 bits**
    - ***Dígitos octales: 0, 1, 2, 3, 4, 5, 6, 7***
  - Hexadecimal (base 16 = 2^4)
    - **Cada dígito hexadecimal representa un grupo de exactamente 4 bits**
    - ***Dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (=10_{10}), B (=11_{10}), C (=12_{10}), D (=13_{10}), E (=14_{10}), F (=15_{10})***

- Y su uso está extendido por
  - La facilidad de conversión a / desde binario, y
  - Porque permiten representar largas secuencias de bits con pocos dígitos (más fáciles de manejar que una secuencia de bits)

Cambio de bases binaria, octal, hexadecimal FCO

- Considerando que las bases octal y hexadecimal son potencias de 2 (la base binaria), se puede demostrar que
  - En octal (base 2^3) un dígito representa un grupo de 3 bits
  - En hexadecimal (base 2^4) un dígito representa un grupo de 4 bits
  - En estos dos casos, el cambio de una representación a otra se realiza utilizando una tabla, agrupando los bits en bloques de 3 o 4

Octal	Binario
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Hexadecimal	Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Hex.	Binario
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Cambio de bases binaria, octal, hexadecimal FCO

- Cambio a/de binario desde/a octal y hexadecimal
  - Cuando el grupo de 3/4 bits no está completo, se llena con ceros
    - Ceros a la izquierda si los bits son de la parte entera
    - Ceros a la derecha si los bits son de la parte fraccionaria
  - Un grupo de bits nunca puede incluir la coma
    - No se pueden mezclar bits de la parte entera y de la fraccionaria en el mismo grupo
    - Empezar las agrupaciones alrededor de la coma

bit de relleno

$11100011011{,}10000001_2 = 111\ 000\ 011\ 011\ ,\ 100\ 000\ 010_2 = 7033{,}402_8$

$11100011011{,}10000001_2 = 1110\ 0001\ 1011\ ,\ 1000\ 0001_2 = E1B{,}81_{16}$

Cambio de bases binaria, octal, hexadecimal FCO

- Ejemplo. Dado N = $F9,E3B_{16}$
  - Pasar a decimal
  - Pasar a binario

$N = F9,E3B_{16} =$

$F_{16} \times 16^1 + 9_{16} \times 16^0 + E_{16} \times 16^{-1} + 3_{16} \times 16^{-2} + B_{16} \times 16^{-3} =$

$= 15_{10} \times 16^1 + 9_{10} \times 16^0 + 14_{10} \times 16^{-1} + 3_{10} \times 16^{-2} + 11_{10} \times 16^{-3}$

$= 249,8894042969_{10}$

$N = F9,E3B_{16} = \quad 1111\ 1001\ ,\ 1110\ 0011\ 1011_2$

Cambio de bases binaria, octal, hexadecimal FCO

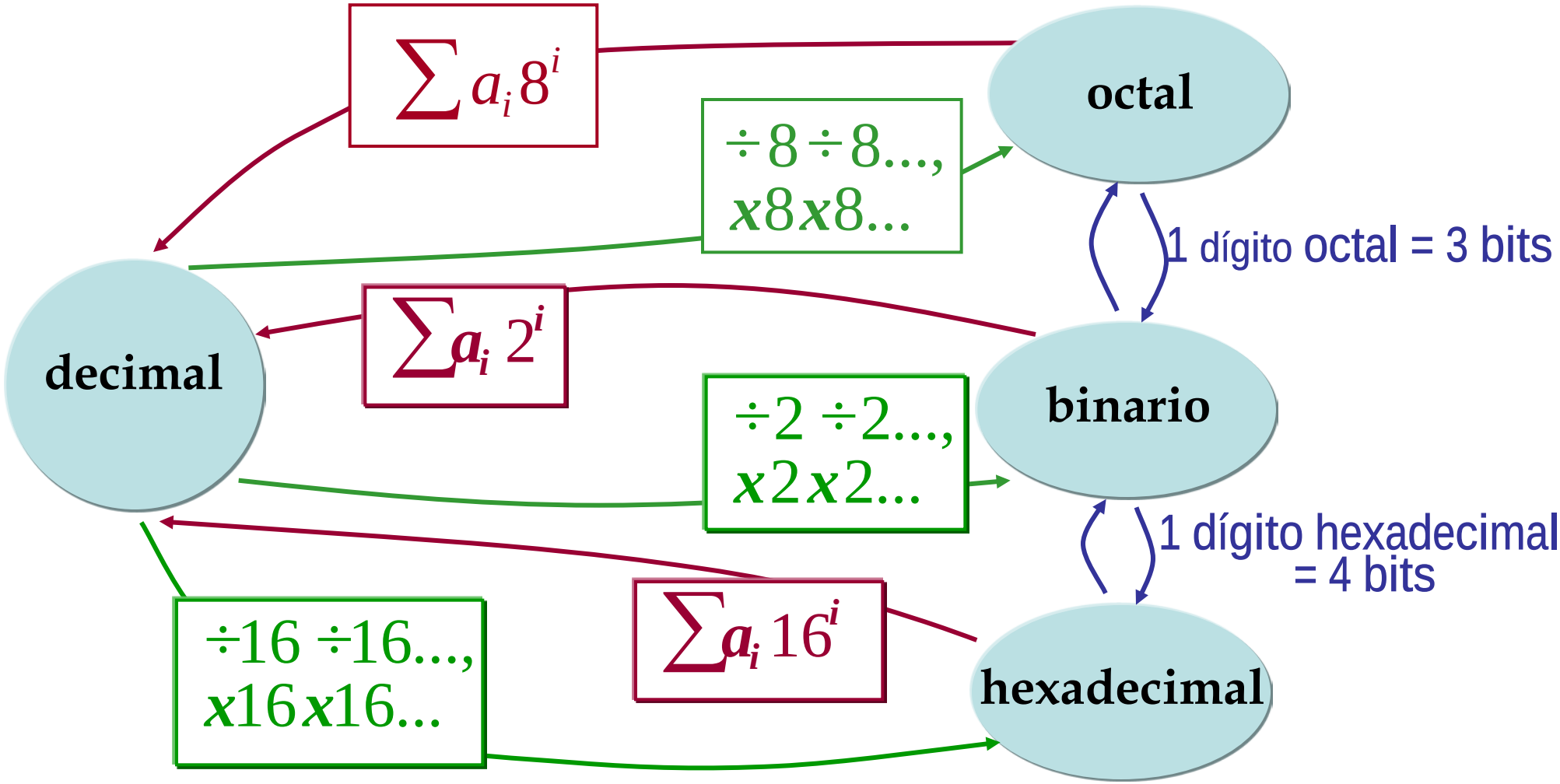
- Cambio de octal a binario y hexadecimal
  - $15,36_8$ = 001 101 , 011 110 $_2$
  - $15,36_8$ = 001 101 , 011 110 $_2$ =

    (y ahora, agrupando los bits de 4 en 4)

    1101 , 0111 1000 $_2$ = $D,78_{16}$

Cambios de base. Resumen

FCO



Representación de números naturales FCO

- Se denominan números naturales a aquellos números sin signo y sin parte fraccionaria
- Los números naturales se representan en el ordenador directamente por medio de su correspondiente valor en binario
- Es lo que se denomina “binario natural”
- Ejemplo:

valores

binario	natural
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

Representación de números naturales FCO

- El rango de representación viene dado por el intervalo de números representables en un formato
- En este caso, formato = longitud en bits
- Con un total de n bits se pueden representar los valores [0, 2^n - 1]
- Es posible que al operar exista desbordamiento (el resultado no es representable con n bits)

Operaciones binarias básicas con naturales

- Las operaciones aritméticas básicas (suma, resta, multiplicación y división) utilizan en binario las mismas reglas que en decimal, excepto la referente a la base
- En la suma de dos o más bits:
  - Se produce bit de accareo (*carry*) cuando la suma de los operandos es igual o superior a la base (>= 2 en binario, y no >= 10)
- En la resta de dos o más bits:
  - El préstamo (*borrow*) se calcula como base + minuendo - sustraendo (2 + minuendo - sustraendo, y no 10 + minuendo - sustraendo)

Operaciones binarias básicas con naturales

- Como funciones combinacionales que son, se pueden formalizar en una tabla de verdad
- Suma de dos bits

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10 (*carry*=1)

A	B	*Carry*	Suma
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Al circuito correspondiente a esta tabla de verdad se le conoce como semisumador o "*Half Adder*" (HA):

$S = A \oplus B$, $Carry = A \cdot B$

Operaciones binarias básicas con naturales

- Como funciones combinacionals que son, se pueden formalizar en una tabla de verdad
- Resta de dos bits

0 - 0 = 0
0 - 1 = 1 (y 1 de préstamo)
1 - 0 = 1
1 - 1 = 0

A	B	Resta	Préstamo
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Operaciones binarias básicas con naturales

- Para sumar dos o más números de N bits,
  - El *carry* generado en una columna es un bit más a sumar en la columna siguiente

Bits de *Carry* 0 1 1 1 1 1 0 1

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

Sumandos

Resultado

Operaciones binarias básicas con naturales

- Para restar dos o más números de N bits
  - El resto generado en una columna es un bit más a restar en la columna siguiente

	0 1 0 0 0 1 0 1	Minuendo
–	0 0 0 1 1 1 0 1	Sustraendo
Préstamos	0 1 1 1 0 0 0	
	0 0 1 0 1 0 0 0	Resultado

Desbordamiento en las operaciones

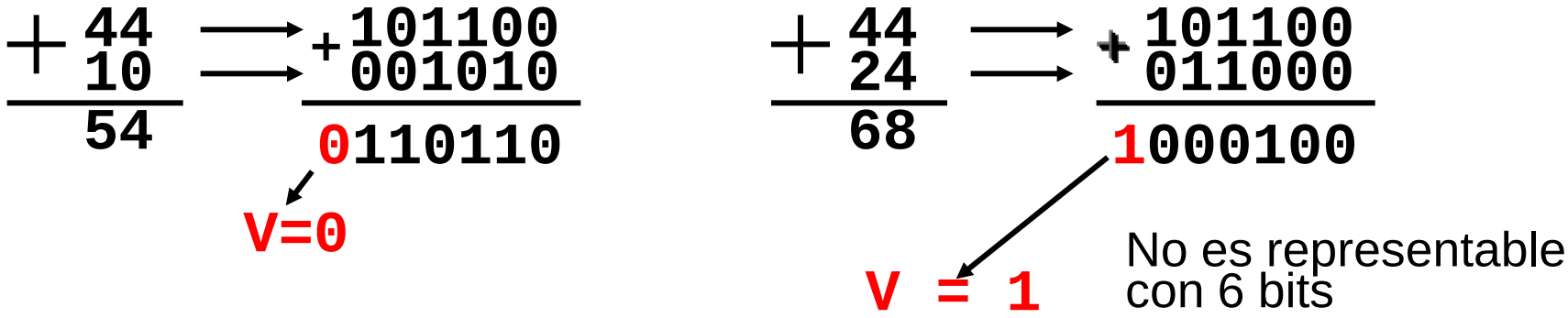
FCO

- Desbordamiento (*overflow*)
  - Ocurre cuando el resultado de una operación no es representable en el formato utilizado (queda fuera del rango de representación)
    - Puede ocurrir en cualquier convenio o formato de representación
    - La limitación en el número de bits empleados por almacenar los datos crea este problema
    - Es una condición o adjetivo sobre el valor del resultado. Indica que la secuencia de bits calculada como resultado no es válida, y que por lo tanto no hay resultado
- Flag o bandera de desbordamiento (V o OV)
  - Salida de un circuito aritmético para indicar la situación de desbordamiento en la operación

Desbordamiento en números naturales FCO

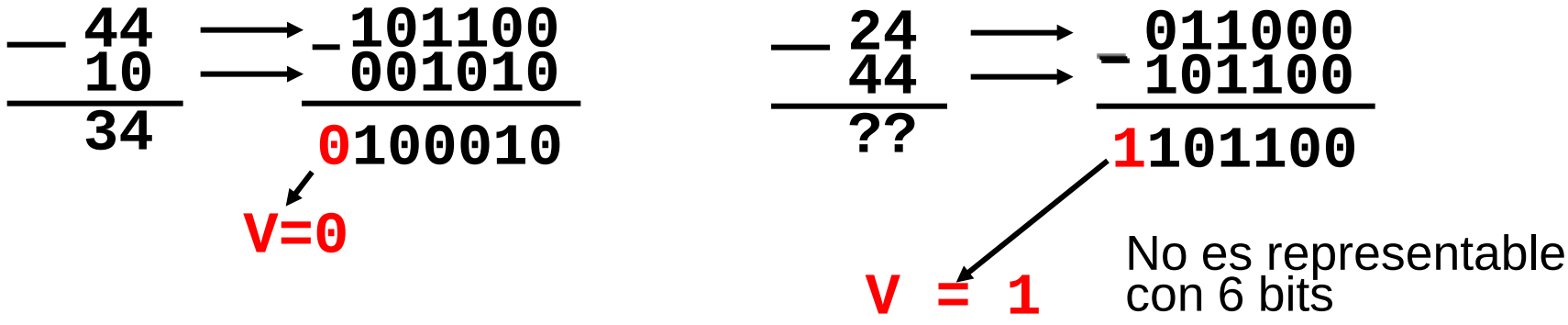
- Desbordamiento en números naturales
  - Ocurre cuando el *carry* de lo ultima etapa es 1, indicando que sería necesario por lo menos un bit más para representar el resultado
  - Expresión algebraica del flag, V = Cn-1

- Ejemplo. Naturales representados con 6 bits



Desbordamiento en números naturales FCO

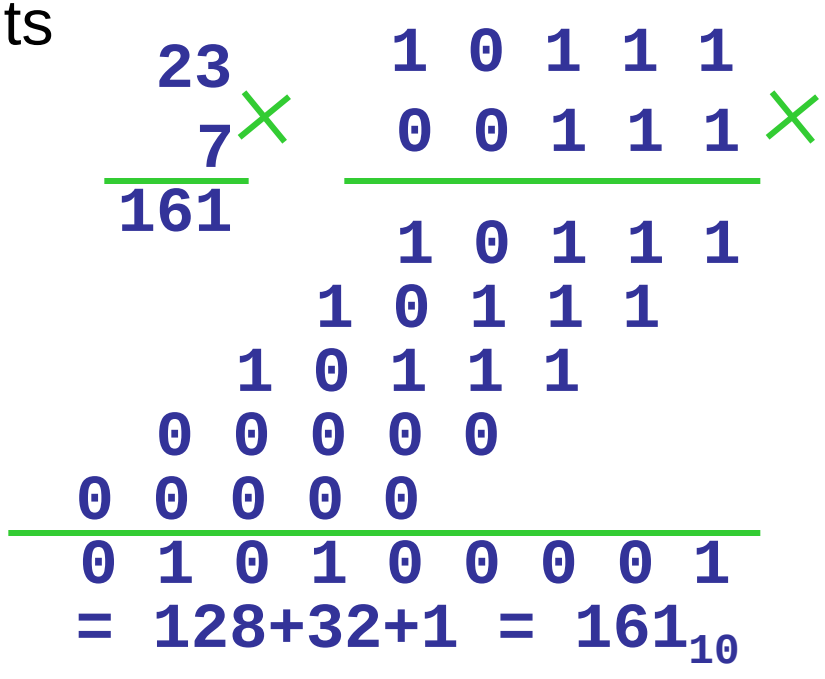
- Ejemplo



- Ejercicios
  - Sumar 101110001_2 y 001110110_2
  - Restar 101110001_2 y 001110110_2
  - Sumar 101110001_2, 001110110_2 y 011001100_2

Operaciones binárias básicas

- Multiplicación
  - Al multiplicar dos números de n y m bits respectivamente, el resultado siempre podrá representarse con un máximo de n+m bits



Operaciones binárias básicas

- Multiplicación
  - Ejercicios

```
      1 0 0 1 1
      1 0 1 0 1  x
-------------------
      1 0 0 1 1
    1 0 0 1 1
  1 0 0 1 1
-------------------
1 1 0 0 0 1 1 1 1
```

```
      1 0 0 0 1
      1 1 1 0 1  x
-------------------
      1 0 0 0 1
    1 0 0 0 1
  1 0 0 0 1
-------------------
1 1 1 1 0 1 1 0 1
```

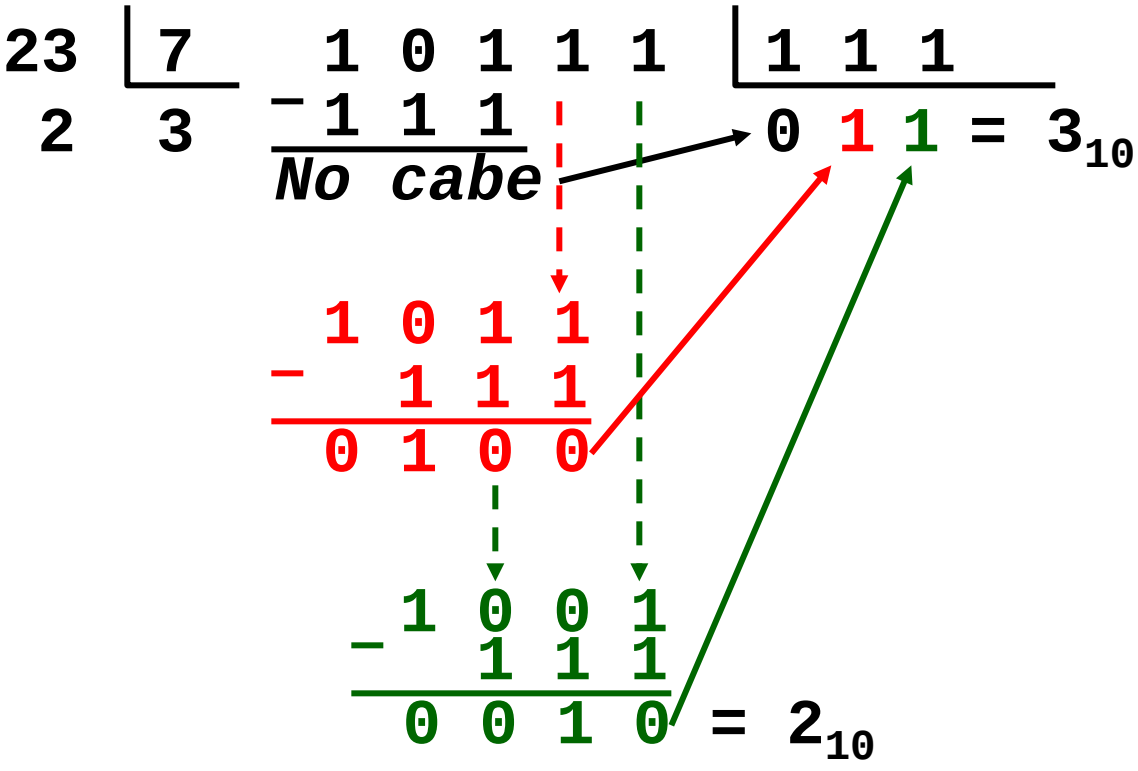
Operaciones binárias básicas

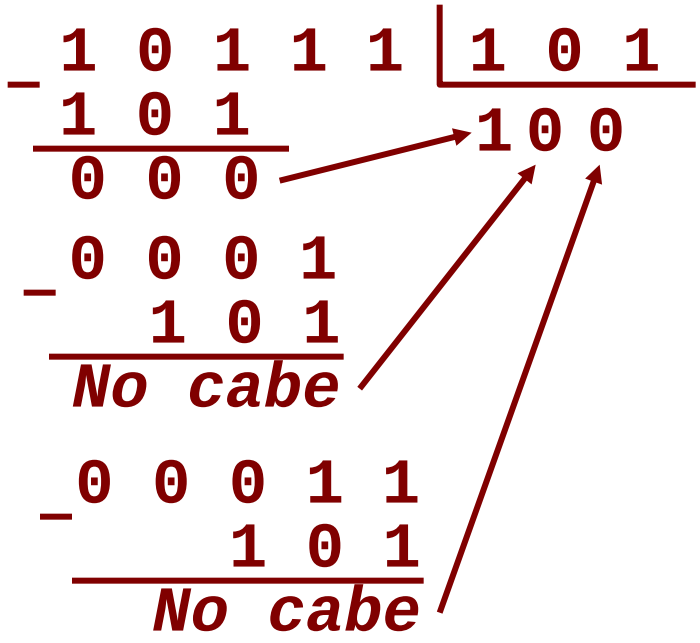
FCO

- División
  - El algoritmo de división es el mismo que en decimal, excepto que cuando el dígito del cociente no es cero, solo puede ser un 1 (es decir, en su caso el dígito del cociente es seguro un 1) cuando en decimal hay que probar entre 1 y 9.

Operaciones binárias básicas

- División
  - Ejemplos





Operaciones binárias básicas

- Complemento a 1 de un número X de n bits (sin signo)
  - Definición: $Ca1(X) = 2^n - X - 1$
  - Se puede calcular también invirtiendo los bits de X
- Complemento a 2 de un número X de n bits (sin signo)
  - Definición: $Ca2(X) = 2^n - X$
  - Se puede calcular también invirtiendo los bits de X y sumando 1,
  - y también como $Ca2(X) = Ca1(X) + 1$

Operaciones binárias básicas

FCO

- Los complementos son operaciones reversibles,
  - Ca1(Ca1(X)) = X
    - Ca1(Ca1(X)) = 2^n - (2^n - X - 1) - 1 = 2^n - 2^n + X + 1 - 1 = X
  - Ca2(Ca2(X)) = X
    - Ca2(Ca2(X)) = 2^n - (2^n - X) = 2^n - 2^n + X = X
- Ejercicios:
  - Ca1(111000101) = 000111010
  - Ca1(000111010) = 111000101
  - Ca2(111000101) = 000111011
  - Ca2(000111011) = 111000101

Representación de enteros

- Números enteros
  - números con signo pero sin parte fraccionaria
- Problema de representación
  - Al intentar almacenar un número entero en la memoria del ordenador, este no almacena signos como “+” o “-”, solo bits.

Representación de enteros

FCO

- Solución
  - Definir convenios de representación, reglas arbitrarias que nos permiten almacenar valores tanto positivos como negativos utilizando únicamente secuencias de 1s y 0s
- Extensión de signo
  - Capacidad de ampliar de forma sencilla (sin realizar operaciones aritméticas) el número de bits utilizado para representar una determinada cantidad y que mantenga el criterio de representación utilizado

Representación de enteros

FCO

- Criterio 1: Signo y Magnitud
  - Se reserva el bit MSB para representar el signo del número;
    - MSB = 0 para los números positivos
    - MSB = 1 para los números negativos
  - El resto de bits representa el valor absoluto en binario natural
  - Rango de representación simétrico [- (2^{n-1}-1), + 2^{n-1}-1]
  - Hay dos ceros, uno “positivo” y otro “negativo”
  - Antes de sumar o restar hay que analizar el signo de los operandos

Representación de enteros

- Criterio 1: Signo y Magnitud
  - Se puede hacer extensión de signo sin más que añadir los ceros que sean necesarios a la izquierda de la magnitud, manteniendo el bit de signo como el MSB

```
-3 =>   1 11          +3 =>   0 11
        1 011                 0 011
        1 0011                0 0011
        1 00011               0 00011
```

S. y Mag.	Decimal
0 000…000	+0
0 000…001	+1
0 000…010	+2
0 000…011	+3
…	…
0 111…110	$+(2^{n-1} - 2)$
0 111…111	$+(2^{n-1} - 1)$

S. y Mag.	Decimal
1 000…000	-0
1 000…001	-1
1 000…010	-2
1 000…011	-3
…	…
1 111…110	$-(2^{n-1} - 2)$
1 111…111	$-(2^{n-1} - 1)$

Representación de enteros

- Criterio 2: Complemento a 2
  - No debemos confundir este convenio de representación con la operación matemática Ca2
  - Utiliza dos métodos diferentes, en función del signo del número. Para una representación con n bits:
    - Los positivos se representan por su valor absoluto, en binario natural, con n-1 bits, y se le añade un 0 a la izquierda.
    - Los negativos se representan por el Ca2 (positivo)
  - Ejemplo con 5 bits. Representar +4 y -4
  - +4 con 4 bits = 0100; añadimos el cero: $00100_{c2} = +4_{c2}$
  - -4 = Ca2 (+4) = Ca2 (00100) = $11100_{c2} = -4_{c2}$

Representación de enteros

FCO

- Criterio 2: Complemento a 2
  - En la representación resultante, el bit MSB indica el signo
    - MSB=0 para los números positivos
    - MSB=1 para los números negativos
  - Rango de representación asimétrico [- 2^{n-1}, + 2^{n-1}-1]
  - Hay un único cero

Compl. A 2	Decimal
0000…000	+0
0000…001	+1
0000…010	+2
0000…011	+3
…	…
0111…110	+(2^{n-1} - 2)
0111…111	+(2^{n-1} - 1)

Compl. A 2	Decimal
1000…000	-2^{n-1}
1000…001	-(2^{n-1} - 1)
…	…
1111…100	-4
1111…101	-3
1111…110	-2
1111…111	-1

Representación de enteros

FCO

- Criterio 2: Complemento a 2
  - Las operaciones de suma y resta no necesitan realizar ajustes para obtener el resultado de la operación representado según este convenio de representación. Esto hace que sea el convenio más utilizado cuando se trata de operar con números enteros
  - Es posible realizar extensión de signo, replicando el bit de signo

```
-2 => 110          +2 => 010
      1110                0010
      11110               00010
      111110              000010
```

Representación de enteros

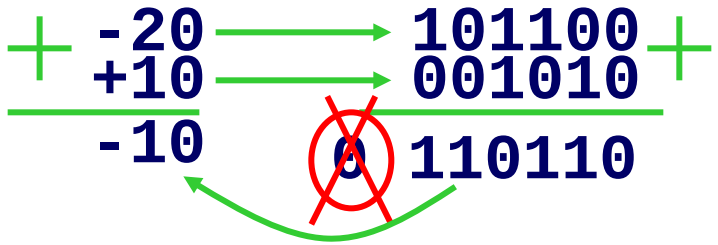
- Sumas y restas en Ca2
  - Sean A y B dos números enteros (positivos o negativos) representados según el convenio de Ca2
  - Para calcular R=A+B y obtener R en el convenio de representación Ca2, realizamos una suma binaria e ignoraremos el *carry* final
  - Para calcular R=A-B y obtener R en el convenio de representación Ca2, realizamos la suma binaria A + Ca2(B) e ignoramos el *carry* final

$$A - B = A + (-B) = A + Ca2(B)$$

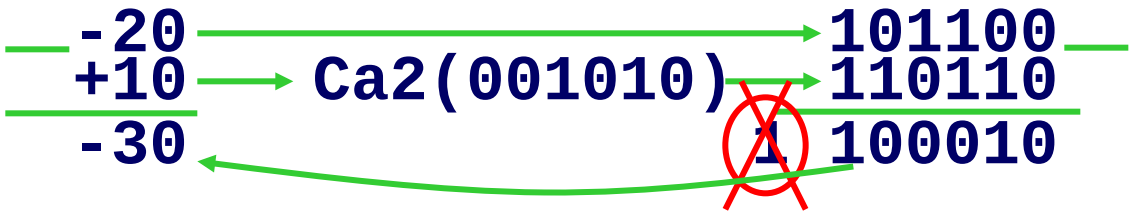
Representación de enteros

- Ejercicio: A=-20, B=+10. Calcular, representando en Ca2 y con 6 bits

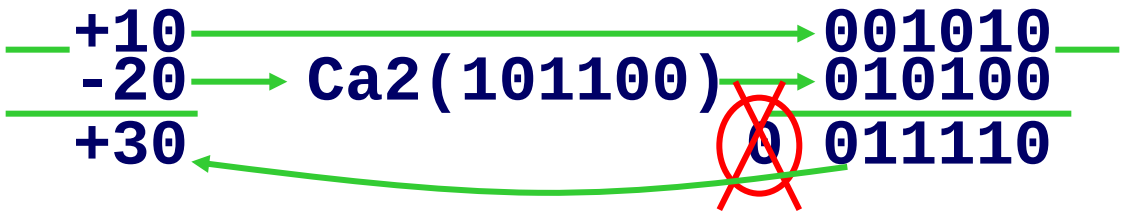
- A+B



- A-B



- B-A



Desbordamiento en las operaciones en Ca2

FCO

- Desbordamiento al operar con números en Ca2
  - Ocurre cuando el resultado de una suma en Ca2 no es representable, bien porque se sale por la derecha (positivo demasiado grande), bien porque se sale por la izquierda (negativo demasiado grande)
  - Los humanos lo pueden detectar mirando el bit de signo: la suma de dos positivos da negativo, o la suma de dos negativos da positivo
  - Los computadores necesitan un circuito para detectar e informar del desbordamiento.

Desbordamiento en las operaciones en Ca2

- Desbordamiento en Ca2

$$A + B = \begin{array}{cccccc} C_{n-1} & C_{n-2} & \cdots & & & \\ A_{n-1} & A_{n-2} & \cdots & A_1 & A_0 & \\ B_{n-1} & B_{n-2} & \cdots & B_1 & B_0 & + \\ \hline \cancel{C_{n-1}} & R_{n-1} & R_{n-2} & \cdots & R_1 & R_0 \end{array}$$

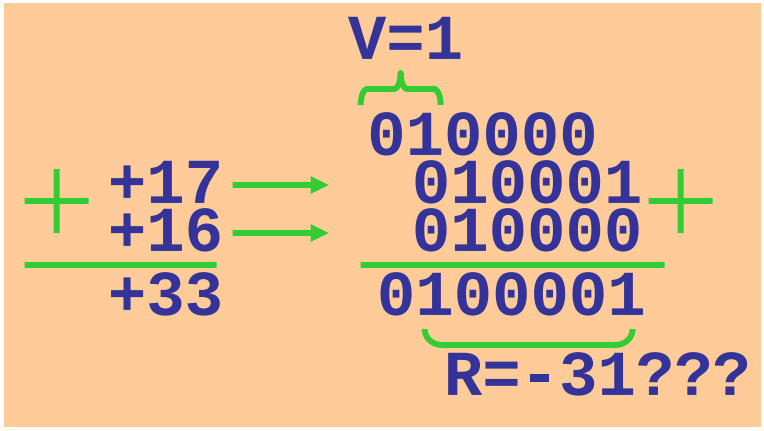
A_{n-1}	B_{n-1}	C_{n-2}	C_{n-1}	R_{n-1}	
0	0	0	0	0	A y B >= 0
0	0	1	0	1	→ Desb. si R < 0
0	1	0	0	1	
0	1	1	1	0	A>=0 y B<0 o viceversa
1	0	0	0	1	Desbordamiento imposible
1	0	1	1	0	
1	1	0	1	0	A y B < 0
1	1	1	1	1	→ Desb. si R>=0

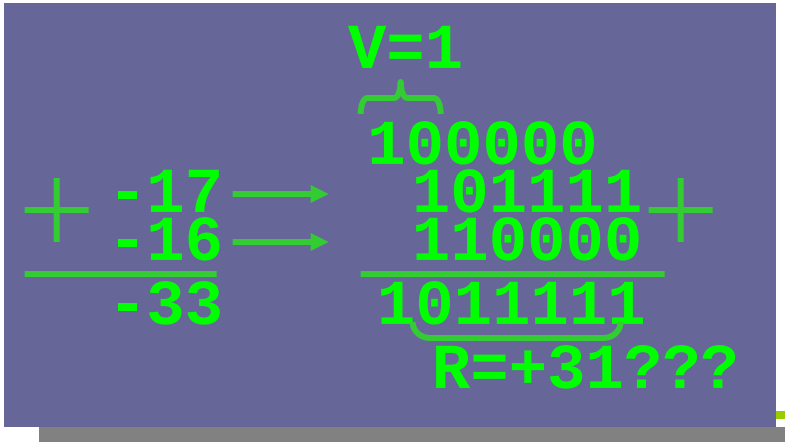
$$V = C_{n-1} \oplus C_{n-2}$$

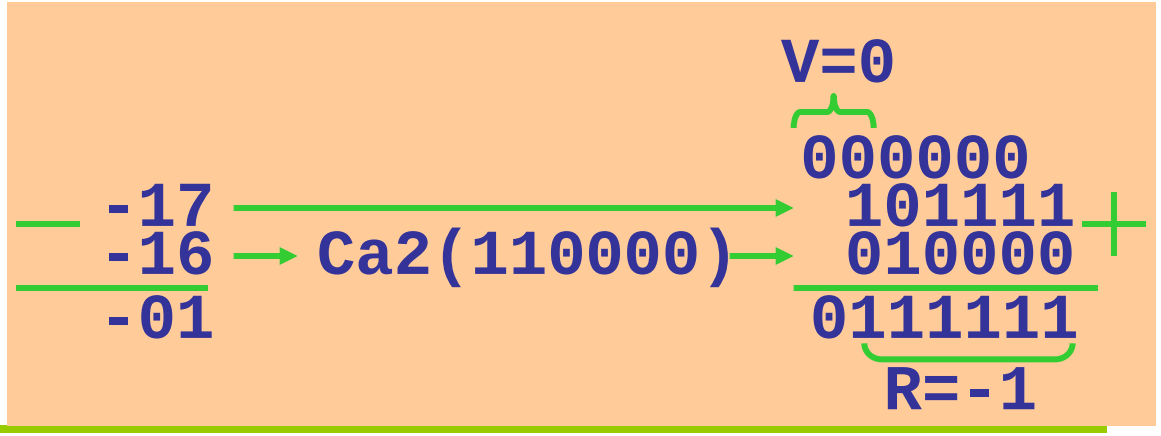
Desbordamiento en las operaciones en Ca2

FCO

- Desbordamiento al operar con números en Ca2.
  - Ejemplos: realice las siguientes operaciones con 6 bits







Representación de enteros

- Criterio 3: Exceso Z
  - Se elige un valor arbitrario denominado exceso (Z). El binario natural de Z representará el cero
  - Cualquier entero A se representa por el binario natural A+Z. Eso implica que A+Z >= 0 para poder representar en binario natural
  - Ejemplo: representa +4 y -4 con 6 bits y Exceso 31

$$+4_{10} = 31 + 4 = 35 = 100011_{z31}$$
$$-4_{10} = 31 - 4 = 27 = 011011_{z31}$$

Representación de enteros

- Criterio 3: Exceso Z
  - Rango de representación asimétrico [- Z, + 2^n - 1 - Z]
  - Cada valor de Z define un rango de representación diferente
  - Se recomienda asignar a Z el valor $2^{n-1} – 1$, donde n = número de bits
  - Hay un único cero

Exceso 2^{n-1}-1	Decimal
0000…000	$-(2^{n-1} - 1)$
0000…001	$-(2^{n-1} - 2)$
0000…010	$-(2^{n-1} - 3)$
0000…011	$-(2^{n-1} - 4)$
…	…
0111…110	-1
0111…111	0

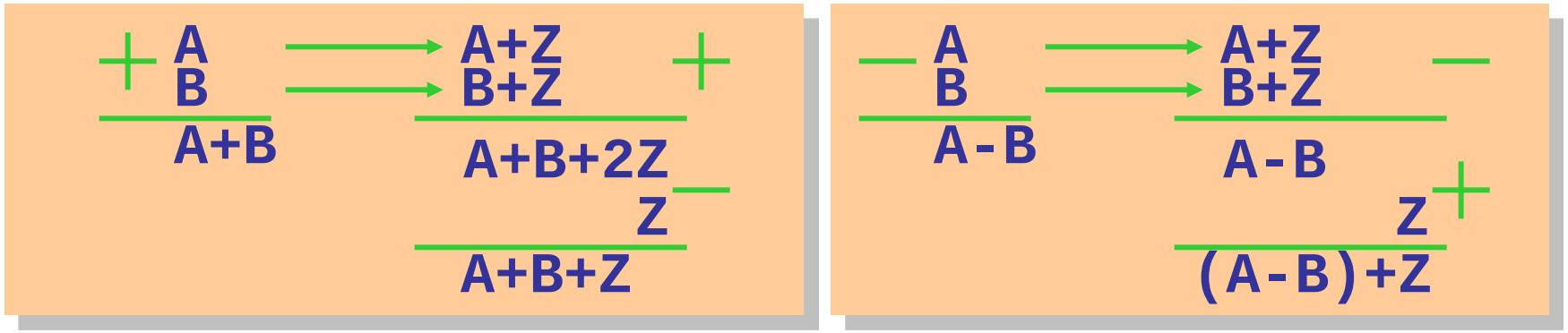
Exceso 2^{n-1}-1	Decimal
1000…000	+1
1000…001	+2
…	…
1111…100	$+(2^{n-1} - 3)$
1111…101	$+(2^{n-1} - 2)$
1111…110	$+(2^{n-1} - 1)$
1111…111	$+2^{n-1}$

Representación de enteros

- Criterio 3: Exceso Z
  - En la representación resultante, el bit MSB no indica el signo, ya que la representación depende del valor de Z
  - La ventaja de este criterio es que las cantidades representadas están ordenadas de menor a mayor. Es decir, desde la mayor cantidad negativa hasta la mayor cantidad positiva. Esto permite comparar dos números con un sencillo circuito y sin hacer operaciones aritméticas
  - No es posible hacer extensión de signo sin hacer operaciones matemáticas, ya que habitualmente el valor del exceso depende de la cantidad de bits disponibles.

Representació d'enters

- Criterio 3: Exceso Z
  - Al hacer operaciones de suma o resta, el resultado no está representado de acuerdo al criterio, por lo que hay que ajustarlo



Representación de enteros

- Ejemplos (n=6 bits)

Decimal	*SyM*	*Ca2*	*Exceso 31*
0	0 00000	000000	011111
+1	0 00001	000001	100000
-1	1 00001	111111	011110
+5	0 00101	000101	100100
-5	1 00101	111011	011010
+31	0 11111	011111	111110
-31	1 11111	100001	000000
+32	----	----	111111
-32	----	100000	----
+33	----	----	----
-33	----	----	----
	[-31,+31]	***[-32,+31]***	***[-31,+32]***

Representación de enteros

- Resumen

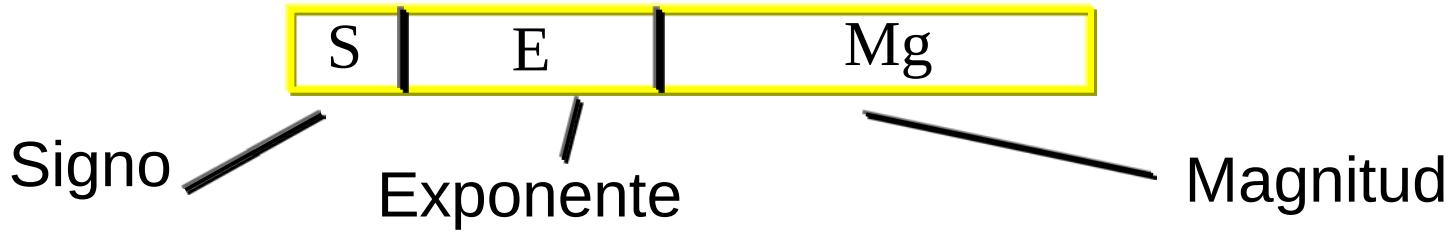
	Signo y Magnitud	Ca2	Exceso 2^{n-1}-1
Rango	$[-(2^{n-1}-1), +2^{n-1}-1]$	$[-2^{n-1}, +2^{n-1}-1]$	$[-(2^{n-1}-1), +2^{n-1}]$
Ext. signo	Sí (con cuidado)	Sí	No
Inconvenientes	Sumas y restas complicadas +0 y -0		Sumas y restas complicadas
Ventajas	Facilidad de uso	Facilidad para operar	Facilidad para comparar

Representación de reales

- Dos maneras de representar los números reales:
  - Coma fija
  - Coma flotante
- Coma fija: se dedican *n* bits para la parte entera y *p* bits para la parte fraccionaria. *n* y *p* son fijas.
  - $R = e_{n-1}\ e_{n-2}\ \ldots\ e_1\ e_0\ ,\ f_{-1}\ f_{-2}\ \ldots\ f_{-p+1}\ f_p$
  - La aritmética coincide con la aritmética entera
  - Problema a causa de la longitud fija de los campos:
    - Con un formato de coma fija de 3 dígitos para la parte entera y 3 más para la parte fraccionaria, hacemos el cálculo $000{,}125_{10} \times 000{,}001_{10}$
    - $000{,}125_{10} \times 000{,}001_{10} = 000{,}000125_{10} = \text{Truncando} = 000{,}000_{10}$

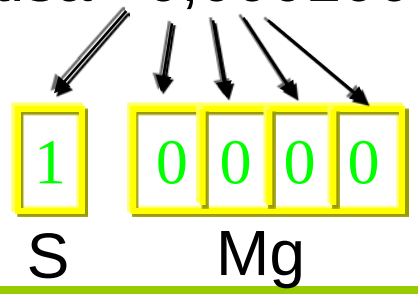
Representación de reales

- Coma flotante
  - Un número real R se representa mediante la expresión:
  - $R = M \times B^E$
    - M es la mantisa, con q bits, fraccionaria, en coma fija y con signo (SyM): $m_{q-1}\ m_{q-2}\ \ldots\ m_1\ m_0$
    - B es la base de la potencia, generalmente 2
    - E es el exponente, con p bits, entero, y generalmente representado en exceso $2^{p-1} - 1$: $e_{p-1}\ e_{p-2}\ \ldots\ e_1\ e_0$
  - La representación en la memoria del ordenador suele ser:



Representación de reales

- Coma flotante
  - La aritmética es más compleja que la aritmética entera
    - Hay circuitos específicos (unidades de coma flotando, FPU: *Floating Point Unit*)
- Normalización de la mantisa
  - El objetivo de normalizar la mantisa es no perder bits significativos (1) desperdiciando espacio para almacenar bits no significativos (0).
  - Ejemplo: almacenar la mantisa –0,00010001 con un formato de 5 bits

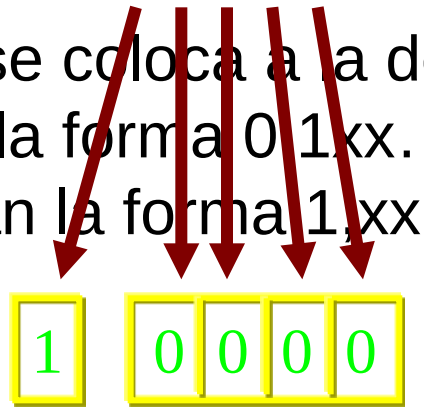


Representación de reales

- Normalización de la mantisa
  - Al normalizar la representación de la mantisa se asegura que se almacenan el número más grande posible de unos
  - El primer bit significativo (1) se coloca alrededor de la coma. Para moverla habrá que modificar el valor del exponente
  - Ejemplo: mantisa normalizada: $-1{,}0001 \times 2^{-4}$
  - Si el primer bit significativo se coloca a la derecha de la coma, los números tendrán la forma 0,1xx…x. Si se coloca a la izquierda, tendrán la forma 1,xx…x

Representación de reales

- Normalización de la mantisa
  - Al normalizar la representación de la mantisa se asegura que se almacenan el número más grande posible de unos
  - El primer bit significativo (1) se coloca alrededor de la coma. Para moverla habrá que modificar el valor del exponente
  - Ejemplo: mantisa normalizada: $-1{,}0001 \times 2^{-4}$
  - Si el primer bit significativo se coloca a la derecha de la coma, los números tendrán la forma 0,1xx…x. Si se coloca a la izquierda, tendrán la forma 1,xx…x

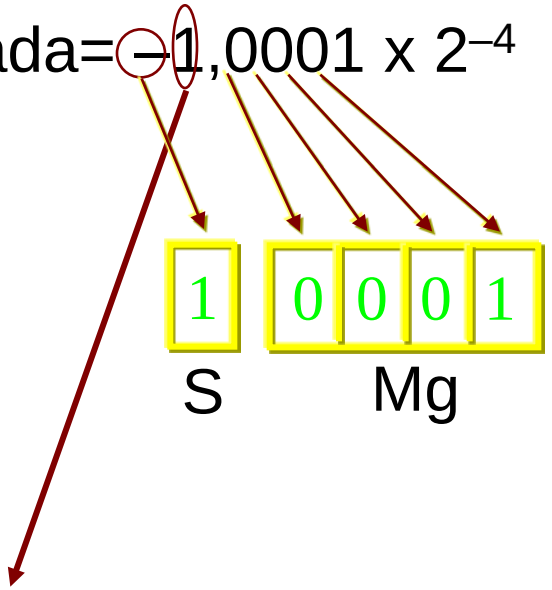


Representación de reales

- Técnica del bit implícito
  - Dado que se asegura la posición del primer bit significativo, no hace falta almacenarlo (aunque se deberá tener en cuenta en las operaciones)
  - Ejemplo: mantisa normalizada= $-1{,}0001 \times 2^{-4}$

Representación en memoria

con la técnica de bit implícito:



No se almacena por ser siempre 1!

Rep. de reales: estándar IEEE 754

FCO

- Formato estándar IEEE 754
  - Surge para facilitar el transvase de información entre distintas máquinas
  - Dos versiones:

Precisión	Total bits	Signo	Exponente	Magnitud
Simple	32	1	8	23*
Doble	64	1	11	52*

*Hay un bit adicional que no se almacena (bit implícito)

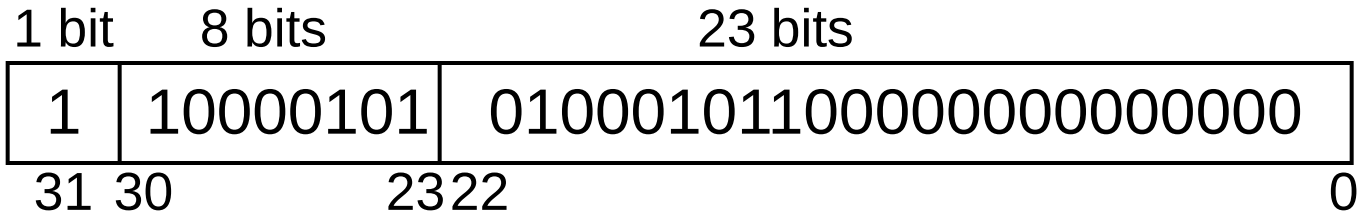
Rep. de reales: estándar IEEE 754

FCO

- Características
  - La mantisa se normaliza en la forma 1,xx…x. Se representa en signo y magnitud y se utiliza la técnica del bit implícito
  - El exponente está en exceso $2^{n-1}-1$ (127 ó 1023).
  - El orden de los campos es S, E y M. La base B es 2
  - Los números “normales” tienen la forma: $\pm 1,M \times 2^{E-127}$
  - Existen números “denormalizados” que tienen la forma $\pm 0,M \times 2^{-126}$

Rep. de reales: estándar IEEE 754

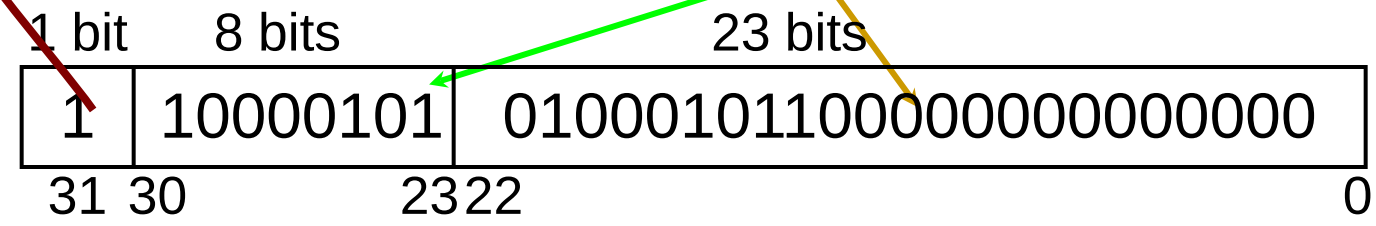
- Ejemplo.
  - Represente el número - 81,375 en formato IEEE de simple precisión (formato normalizado)
    - Parte entera: $- 81_{10} = 1010001_2$
    - Parte decimal: $0{,}375_{10} = 011_2$
    - Mantisa = $- 1010001{,}011_2$
    - Normalizada = $-1{,}010001011_2 \times 2^{+6}$
    - Exponente = 6_{10}
    - Exponente en exceso 127 = $133_{10} = 10000101_2$



Rep. de reales: estándar IEEE 754

- Ejemplo
  - Represente el número - 81,375 en formato IEEE de simple precisión (formato normalizado)
    - Parte entera: $-81_{10} = 1010001_2$
    - Parte decimal: $0{,}375_{10} = 011_2$
    - Mantisa = -1010001.011_2
    - Normalizada = $-1{,}010001011_2 \times 2^{+6}$
    - Exponente = 6_{10}
    - Exponente en exceso 127 = $133_{10} = 10000101_2$

Bit implícito que no se almacena



Rep. de reales: estándar IEEE 754

FCO

- Ejercicios
  - La representación de 5,6875 en IEEE 754 de simple precisión es
    - $+5{,}687510 = 101{,}1011_2 = 101{,}1011 \times 2^0 = 1{,}011011 \times 2^2$

S	E	Mg
0	10000001	01101100000000000000000

Rep. de reales: estándar IEEE 754

FCO

- Ejercicios
  - Si la secuencia de bits (en hexadecimal) es C0B60000, se está representando el valor

S	E	Mg
1	10000001	01101100000000000000000

Rep. de reales: estándar IEEE 754

FCO

- Casos especiales del formato estándar
  - Si E = 00…0 y M = 0, se está representando el valor 0, que no es representable utilizando el formato normalizado
  - Si E = 00…0 y M != 0, se está representando un valor muy pequeño en formato desnormalizado: $\pm 0,M \times 2^{-126}$
  - Si E = 11…1 y M = 0, se está indicando que el resultado de la operación no es representable. Será $+\infty$ o $-\infty$ según el signo
  - Si E = 11…1 y M != 0, se está indicando que el resultado de una operación no tiene sentido (ejemplo, 0÷0). Este tipo de resultados se conocen como NaN (“Not a Number”)
- Las secuencias 00…000 y 11…111 del exponente no sirven para representar números “normales”

Rep. de reales: estándar IEEE 754

FCO

- Rango de representación
  - Determinado principalmente por el número de bits asignados a E
- Precisión: distancia entre dos valores representables consecutivos
  - Determinado principalmente por el número de bits asignados a M
- Posibles desbordamientos al representar valores
  - *Overflow*. Cuando el número está tan alejado del 0 (valor absoluto muy grande) que no se puede representar
  - *Underflow*. Cuando el número está tan próximo al 0 (valor absoluto muy pequeño) que no se puede representar

Rep. de reales: estándar IEEE 754

FCO

- Rango de representación de los números normalizados
  - Del rango original [-127, +128] (o [-1023, +1024]) del exponente, las secuencias binarias 00…000 y 11…111 son indicaciones de casos especiales, y no sirven para representar números normales
  - Por lo tanto, el rango disponible es [-126, +127] (o [-1022, +1023])
  - En valor absoluto, el número más pequeño representable en simple precisión es $1{,}00\ldots00_2 \times 2^{-126}$
  - En valor absoluto, el número más grande representable en simple precisión es $1{,}11\ldots111_2 \times 2^{+127} \approx 1{,}0 \times 2^{+128}$

Rep. de reales: estándar IEEE 754

FCO

- Rango de representación de los números desnormalizados
  - Los números desnormalizados ofrecen otro conjunto de números representables, con el valor del exponente fijo: -126 (o -1022)
  - En valor absoluto, el número más pequeño en simple prec. es $0{,}00\ldots001_2 \times 2^{-126} = 1{,}0 \times 2^{-126-23} = 1{,}0 \times 2^{-149}$
  - En valor absoluto, el número más grande en simple prec. es $0{,}11\ldots111_2 \times 2^{-126} \approx 1{,}0 \times 2^{-126}$

    que coincide con el inicio del rango de los números normalizados

Rep. de reales: estándar IEEE 754

FCO

- Rango completo en simple precisión

$$0 \cup \left[\pm 2^{-149}, \pm 2^{-126}\right[\cup \left[\pm 2^{-126}, \pm 2^{+128}\right[\approx \left[\pm 2^{-149}, \pm 2^{+128}\right]$$

Representación de caracteres

FCO

- Son caracteres
  - Las letras (“a”, …, ”z”, “A”, …, “Z”)
  - Los dígitos (“0”, …, “9”)
  - Los signos de puntuación (“.”, “,”, “;”, …)
  - Y los símbolos especiales (“*”, “&”, “$”, …)
- Para representar caracteres se asigna un código numérico a cada uno de los caracteres por medio de una tabla
- El ordenador siempre trabaja con los códigos, nunca con los símbolos gráficos

Representación de caracteres

FCO

- Las características de una representación son
  - Longitud en bits de los códigos
  - Número de caracteres representable
  - La asignación de código a cada carácter (la tablade caracteres)
- A.S.C.I.I. (*American Standard Code for Information Interchange*)
  - Longitud fija, igual para todos los códigos
  - ASCII original. Longitud de código de 7 bits
  - ASCII extendido. Ampliación para caracteres internacionales, con una longitud de código de 8 bits

Representación de caracteres

FCO

- E.B.C.D.I.C. (*Extended Binary Coded Decimal Interchange Code*)
  - Surge en 1964 con el sistema IBM S360
  - Longitud fija de 8 bits
  - Solo se usa en algunos sistemas mainframe
- Unicode: Universalidad, Uniformidad, Unicidad
  - Tratamiento de textos en diferentes lenguas
  - Textos en lenguas muertas y otras disciplinas
  - Se pueden gastar 8, 16 y 32 bits
  - Es el futuro, ahora!

Representació de caracteres

FCO

- Tabla ASCII original (7 bits)

	0	16	32	48	64	80	96	112
+0	*NUL*	*DLE*	*SP*	0	@	P	`	p
+1	*SOH*	*DC1*	!	1	A	Q	a	q
+2	*STX*	*DC2*	"	2	B	R	b	r
+3	*ETX*	*DC3*	#	3	C	S	c	s
+4	*EOT*	*DC4*	$	4	D	T	d	t
+5	*ENQ*	*NAK*	%	5	E	U	e	u
+6	*ACK*	*SYN*	&	6	F	V	f	v
+7	*BEL*	*ETB*	'	7	G	W	g	w
+8	*BS*	*CAN*	(	8	H	X	h	x
+9	*HT*	*EM*	)	9	I	Y	i	y
+10	*LF*	*SUB*	*	:	J	Z	j	z
+11	*VT*	*ESC*	+	;	K	[	k	{
+12	*FF*	*FS*	,	<	L	\	l	|
+13	*CR*	*GS*	-	=	M	]	m	}
+14	*S0*	*RS*	.	>	N	^	n	~
+15	*S1*	*US*	/	?	O	_	o	*DEL*

El codigo ASCII de “z” es

112 + 10 = 122

UNIVERSIDAD POLITECNICA DE VALENCIA

etsinf

DISCA

Fundamentos de los computadores

Tema 6. Representación de la información