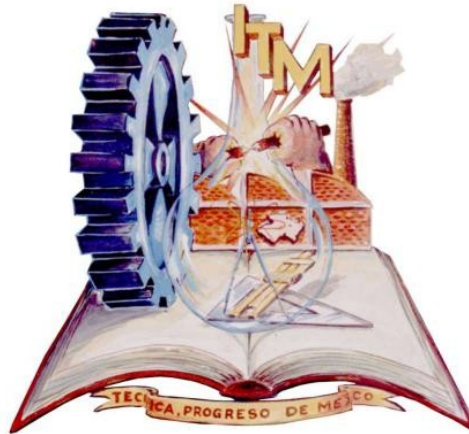


Instituto Tecnológico de Morelia



Microcontroladores **Representación de datos en las computadoras**

M.C.Miguelangel Fraga Aguilar

<http://sagitario.itmorelia.edu.mx/mfraga>
mfraga@itmorelia.edu.mx

Representaciones numéricas

- Representación de números sin signo
 - Binario natural
 - Hexadecimal
- Representación de números con signo
 - Complemento a dos
- Aritmética
- Operaciones booleanas y bit a bit (bitwise)

Números sin signo

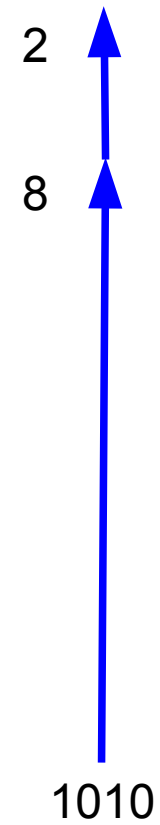
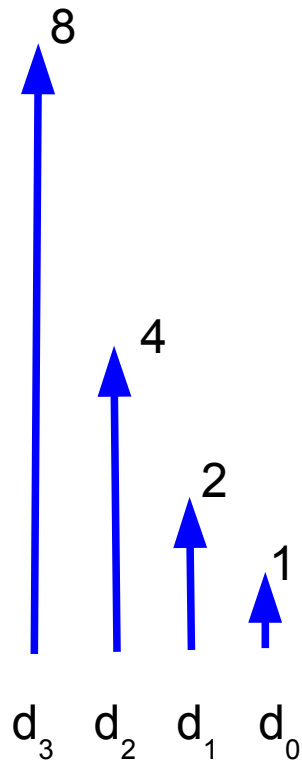
La representación de números enteros en sistema binario consiste en usar una cadena de n dígitos binarios donde la posición de cada dígito indica su peso.

$$V = \sum_{i=0}^{n-1} d_i 2^i; \quad d_i \in \{0,1\}$$

<i>i</i>	4	3	2	1	0
Peso	16	8	4	2	1
Bit	1	0	1	1	0

Rango representable: $0 \dots 2^n - 1$

Números sin signo como vectores

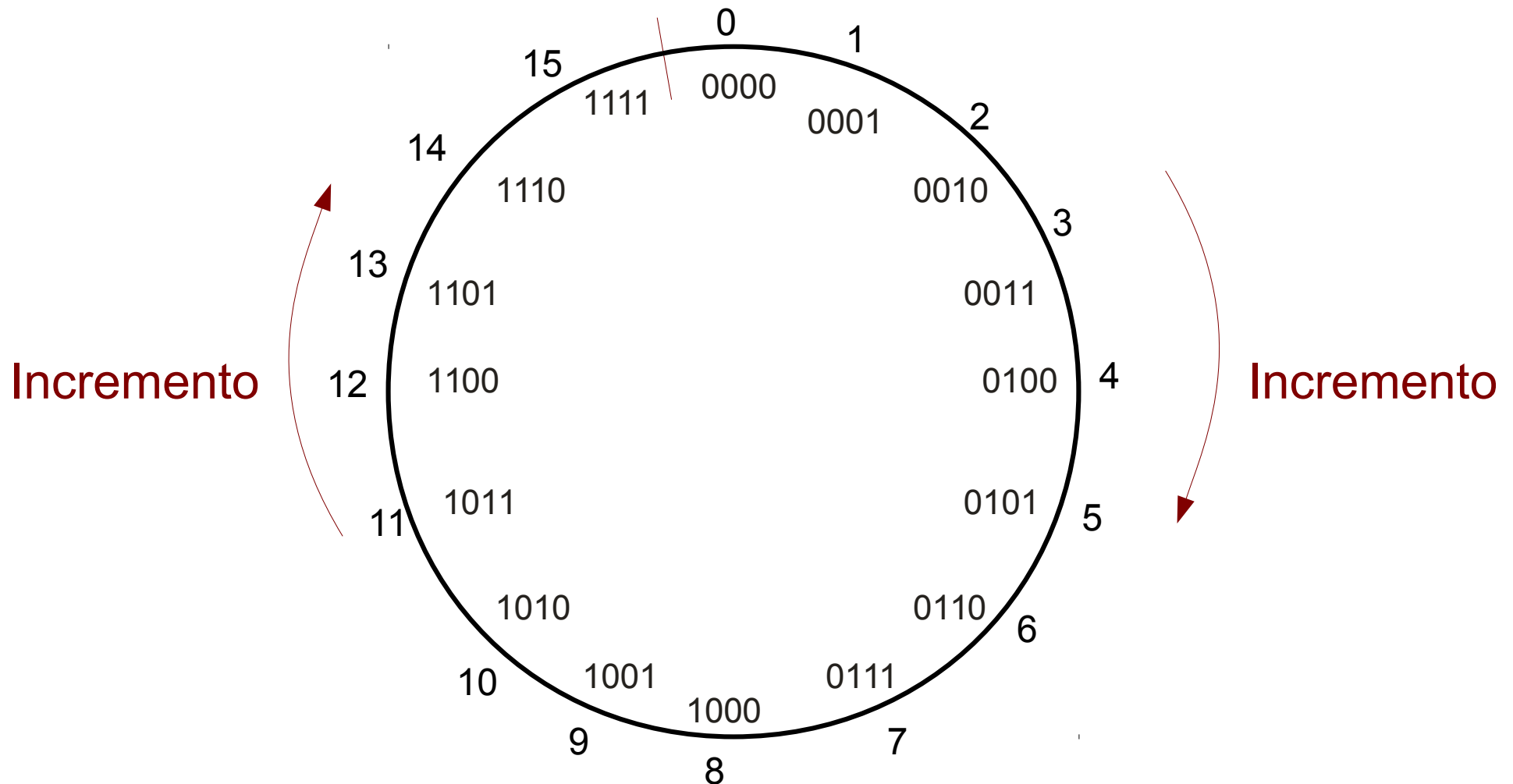


Número finito de dígitos

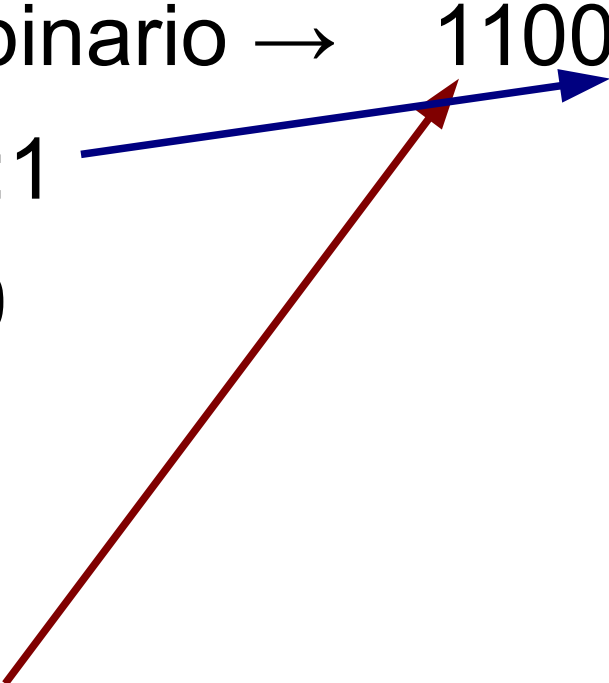
Círculo numérico

Acareo →

← Préstamo



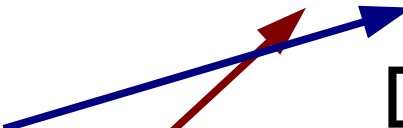
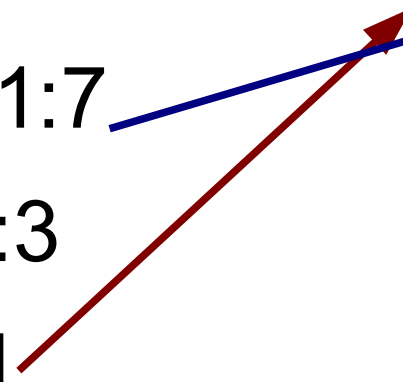
Ejemplos de conversiones

- 1000 1100 b $\rightarrow 1*2^7+1*2^3+1*2^2 = 128+8+4=140$
- 0011 0101 b $\rightarrow 1*2^5+1*2^4+1*2^2+1*2^0 = 32+16+4+1=53$
- 25 d $\rightarrow$ binario $\rightarrow$ 11001
 - 25/2=12:1
 - 12/2=6:0
 - 6/2=3:0
 - 3/2=1:1
 - 1/2=0:1

Bit menos significativo

Bit más significativo

Ejemplos de conversiones 2

- $732_o \rightarrow 7 \cdot 8^2 + 3 \cdot 8^1 + 2 \cdot 8^0 = 7 \cdot 64 + 3 \cdot 8 + 2 = 474$
- $CAB_h \rightarrow 12 \cdot 16^2 + 10 \cdot 16^1 + 11 \cdot 16^0 =$
 $12 \cdot 256 + 10 \cdot 16 + 11 = 3243$
- $95_d \rightarrow \text{octal} \rightarrow 137$
 $95/8 = 11:7$  Dígitos menos significativo
 $11/8 = 1:3$
 $1/8 = 0:1$  Dígitos más significativo

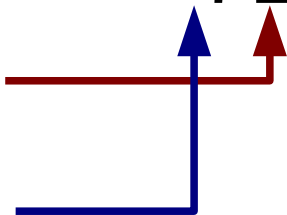
Notación Hexadecimal

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

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

$$V = \sum_{i=0}^{n-1} d_i 16^i; \quad d_i \in \{0 \dots F\}$$

Ejemplos de notación hexadecimal

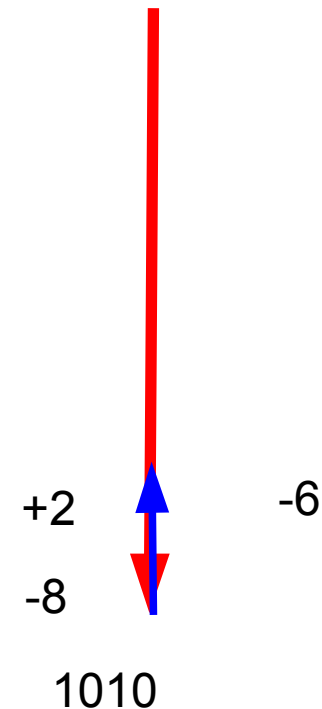
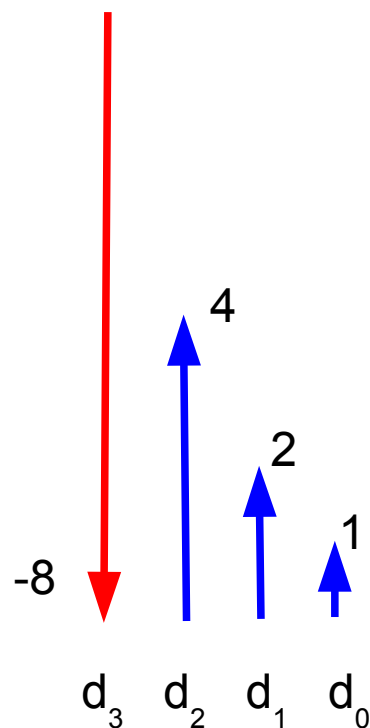
- 1010 1100 b $\rightarrow$ AC h
- 0101 0110 b $\rightarrow$ 56h
- 126d $\rightarrow$
126/16=7:14
7/16=0:7
7E h

- BEBAh $\rightarrow$ 1011 1110 1011 1010b
- BEBAh $\rightarrow$ $11 \cdot 16^3 + 14 \cdot 16^2 + 11 \cdot 16^1 + 10 \cdot 16^0 =$
 $11 \cdot 4096 + 14 \cdot 256 + 11 \cdot 16 + 10 \cdot 1 = 48826$

Números con signo en Complemento a dos

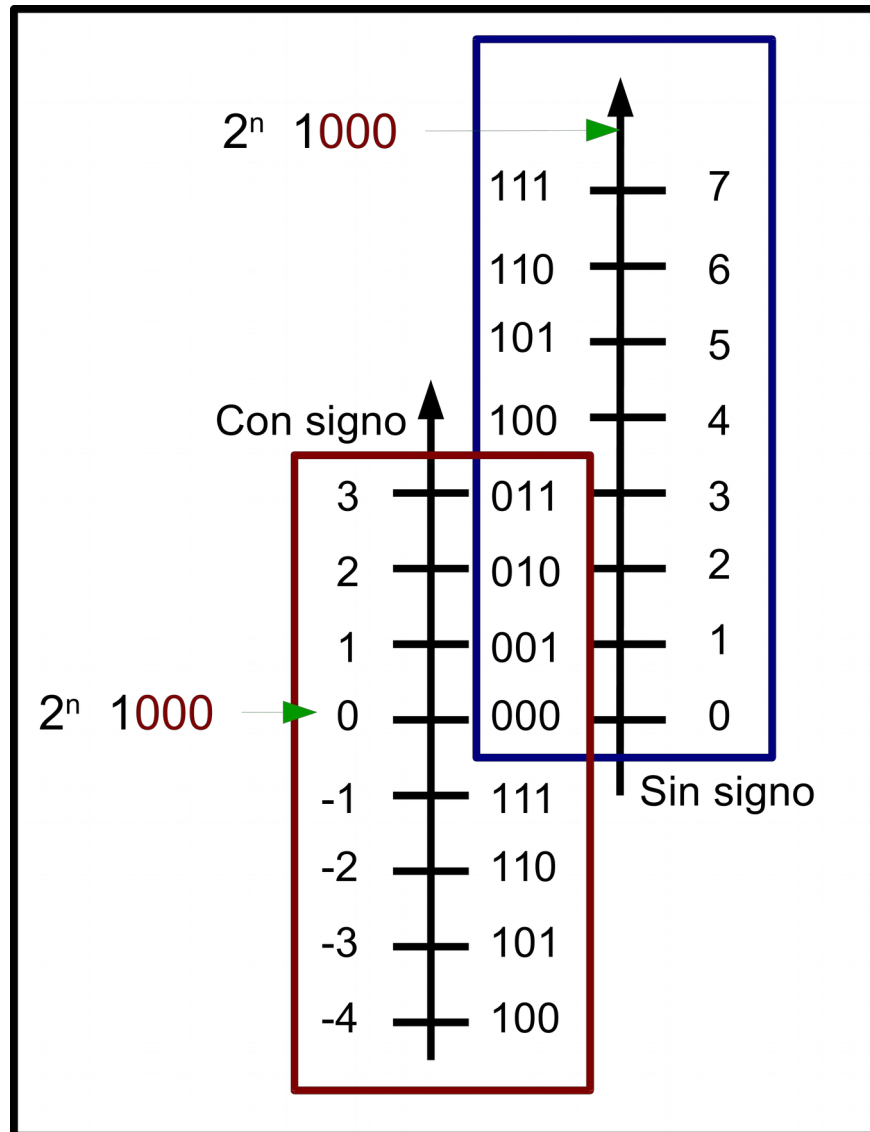
- Se asigna la mitad de las combinaciones numéricas a los números negativos
- Todos los números positivos tienen el bit más significativo en 0
- Todos los números negativos tienen el bit más significativo en 1
- Se usa una combinación de las positivas para almacenar al cero, por lo que el intervalo que se puede representar no es simétrico

Números con signo como vectores

- El bit más significativo se le da un peso negativo de 2^{n-1} , donde n es el número de bits.



Recta numérica



Complemento a dos

- Se aprovecha el hecho de que al usar aritmética de tamaño fijo, cualquier acarreo se descarta. Por ejemplo: si se trabaja con cuatro bits y se suma $Fh+1=0h$, ya que el acarreo del ultimo bit se descarta. Ya que al sumarle 1 al Fh se obtuvo el 0, este corresponde con el -1 en la notación de complemento a dos.

Complemento a dos

- Para saber que número negativo corresponde a uno positivo, se resta este al cero con acarreo descartado. Por ejemplo, en cuatro bits, para saber a que número corresponde al -3, se calcula el resultado de $10h - 3h = Dh$
- Esta resta es la definición de la operación de complemento dos, y en general, se puede definir para n bits como:

$$C_2(V) = 2^n - V$$

Donde C es el complemento a dos de V a n bits

Complemento a dos

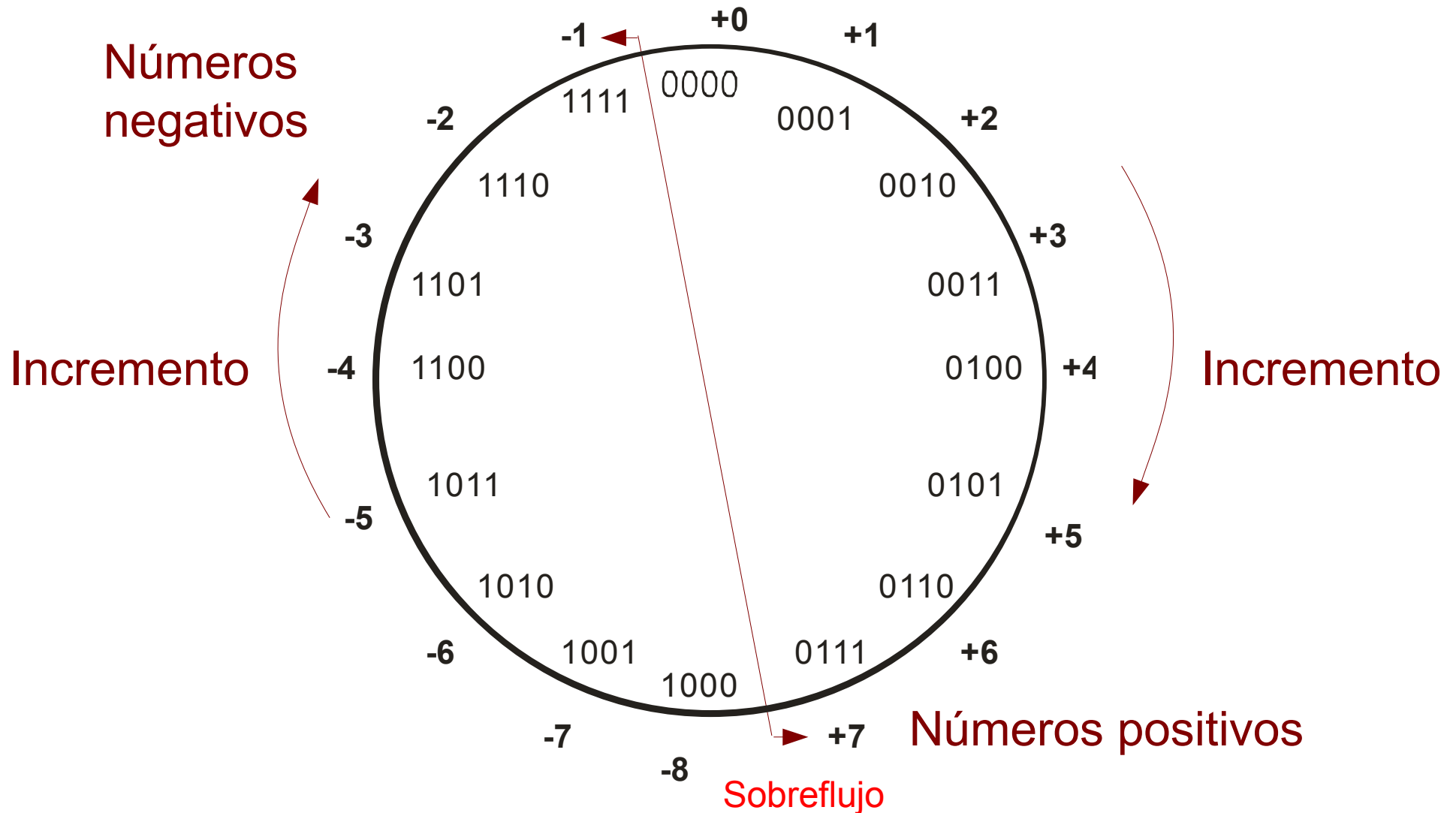
También se puede calcular negando todos los bits y sumándole 1, o por inspección copiando los bits hasta el primer uno de derecha a izquierda, y el negado de los restantes bits.

Existe una sola representación para el cero y existe un número negativo más que los positivos

Se resuelven los problemas del complemento a uno

(+3) 0011	(-3) 1101	(-5) 1011	(+7) 0111
+ (+2) 0010	+ (-2) 1110	+ (+3) 0011	+ (-5) 1011
<u> </u>	<u> </u>	<u> </u>	<u> </u>
(+5) 0101	(-5) <u>1</u> 1011	(-2) 1110	(+2) <u>1</u> 0010

Circulo Numérico Complemento a dos



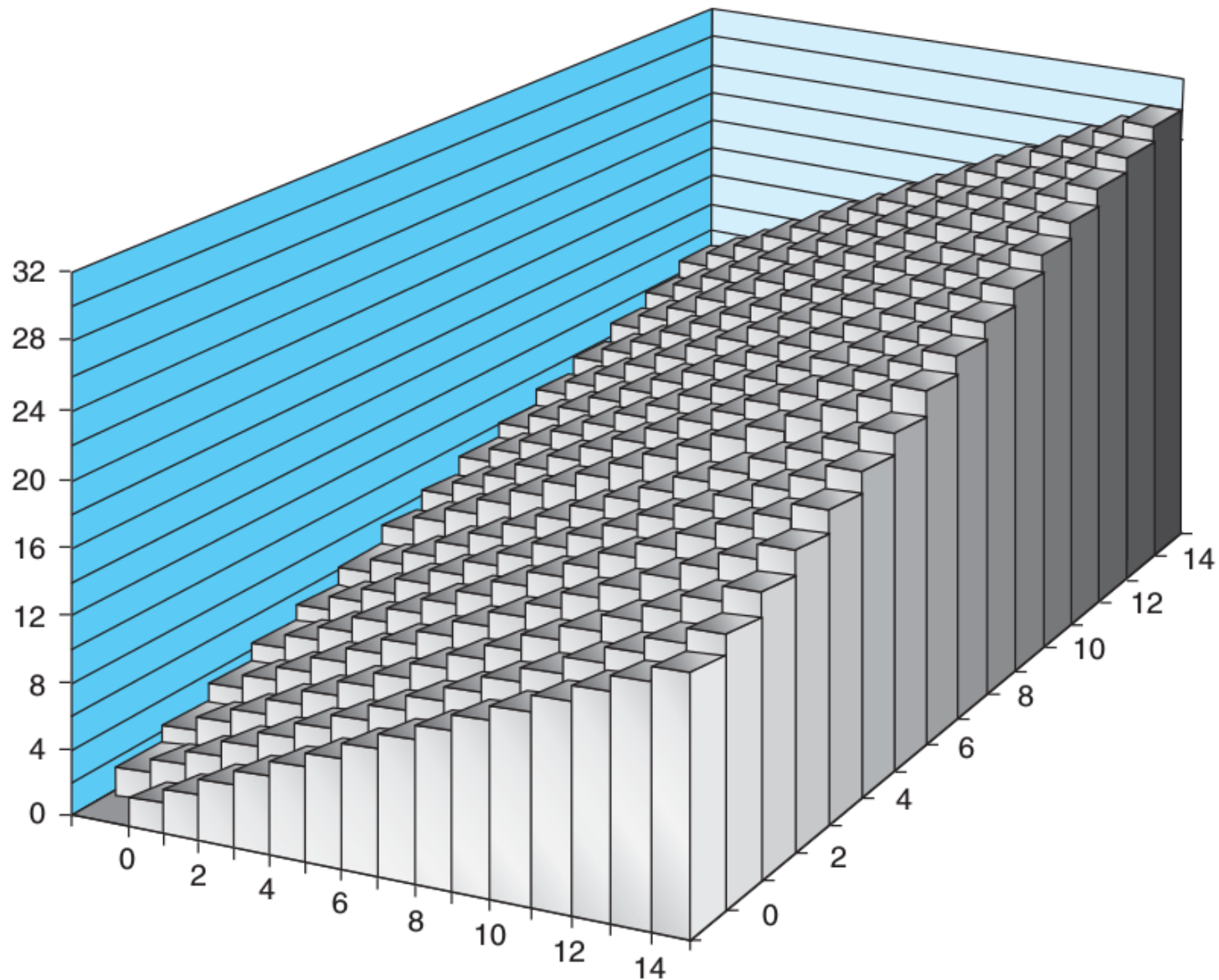
Representación de números con signo

- Represente el 4567 decimal en 16 bits con signo:
 - Como es positivo, solo hay que convertirlo a binario: **11D7h**
- Represente el -8912 decimal en 16 bits con signo:
 - El equivalente en binario es 22D0h
 - Es negativo, hay que obtener el complemento a dos: $10000h - 22D0h = \text{DD30h}$

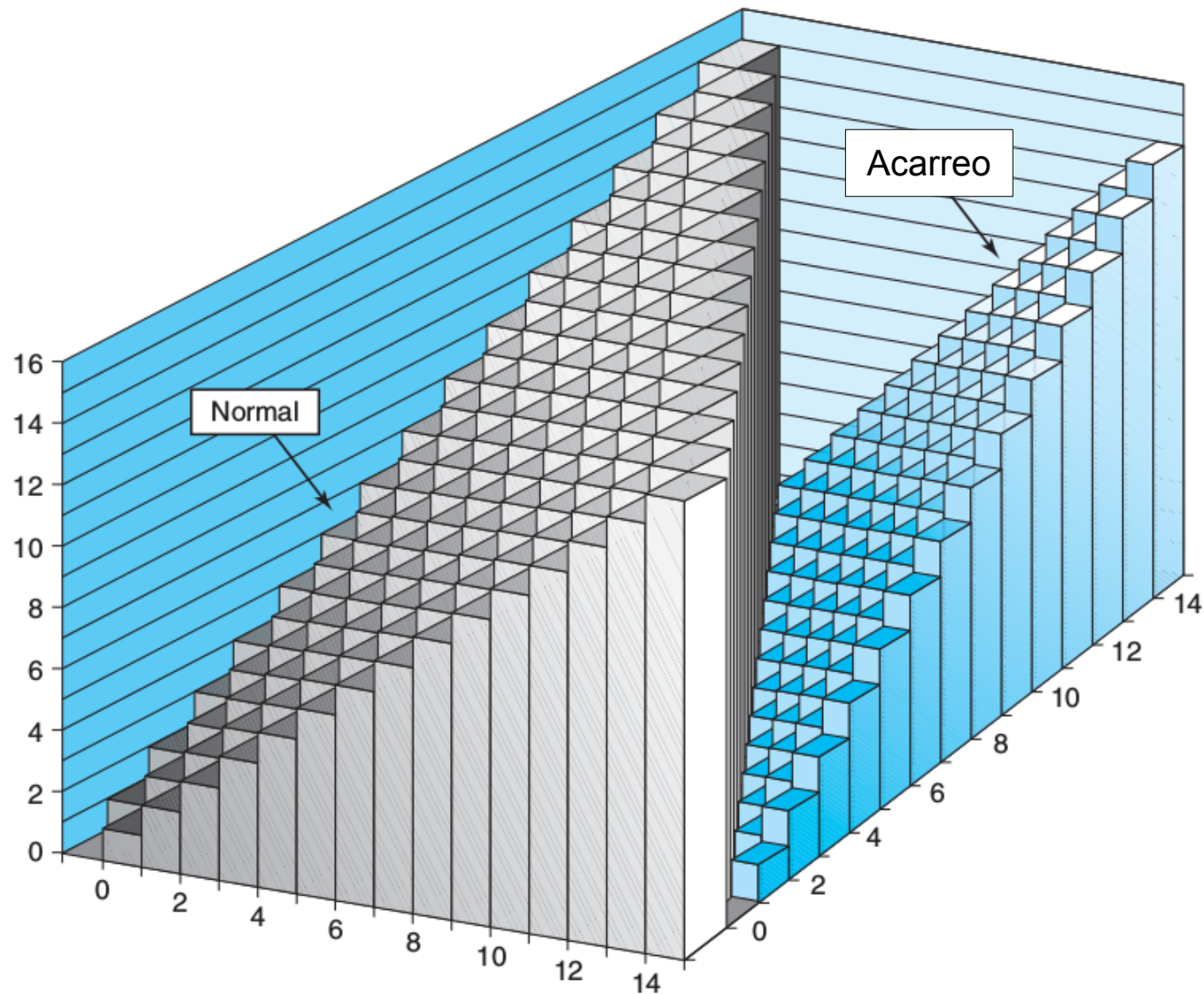
Representación de números con signo (2)

- ¿A que número decimal equivale el 5D6Bh como número con signo de 16 bits?
 - El bit de signo es cero, por lo que es positivo y equivale al 23915 decimal
- ¿A que número decimal equivale el 9C6Bh como número con signo de 16 bits?
 - El bit de signo de uno, por lo que es negativo. Su complemento a dos es 10000h-9C6Bh=6395h
 - Convirtiendo a decimal 9C6Bh es -25493 dec.

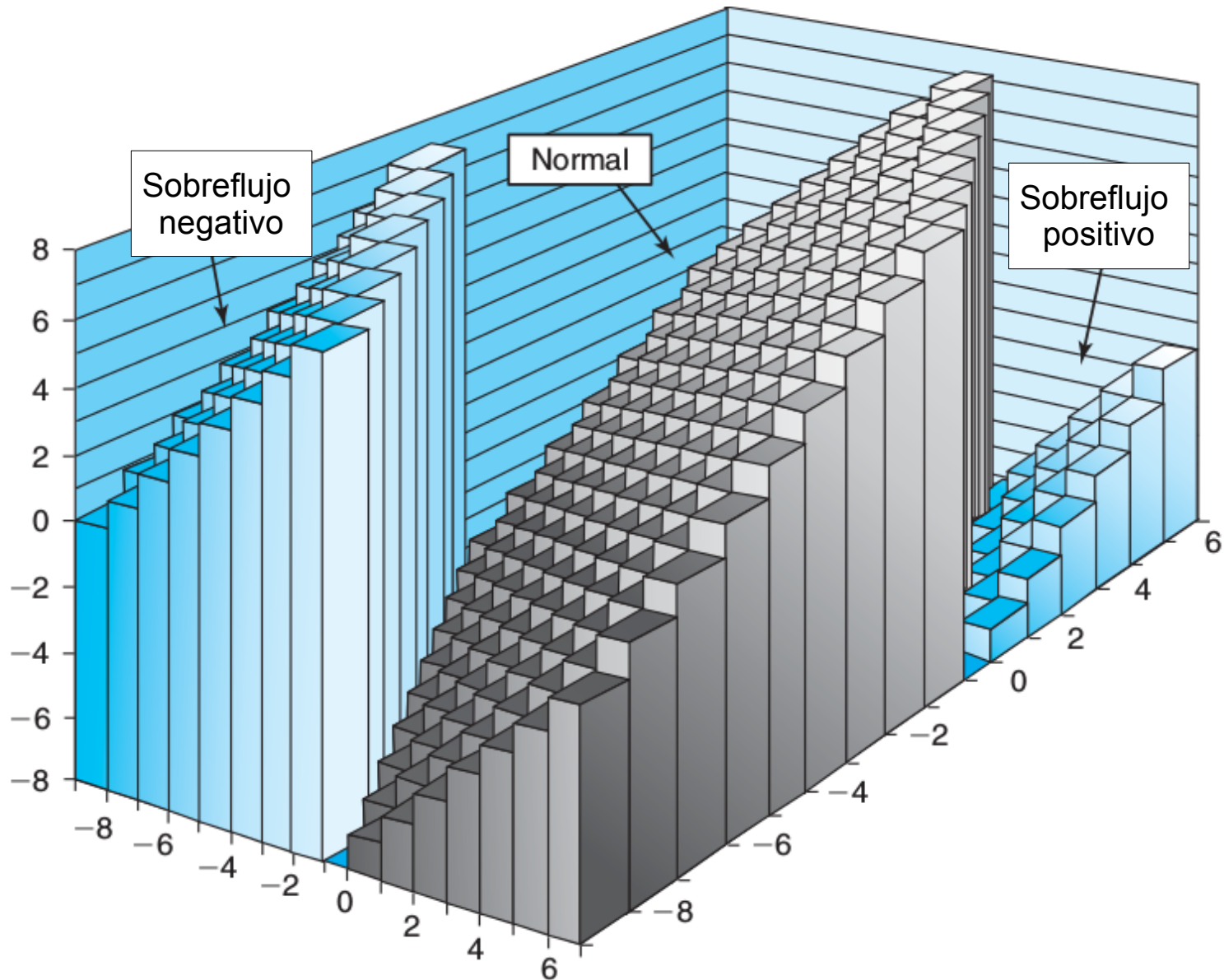
Suma entera datos de 4 bits, resultado en 5 bits



Suma entera datos de 4 bits, resultado en 4 bits



Suma entera con signo, datos de 4 bits, resultado en 4 bits



Sobreflujo

$$\begin{array}{rcccccl} & 0 & 1 & 1 & 1 & & \\ & \nearrow & \nearrow & \nearrow & \nearrow & & \\ & & 0 & 1 & 1 & 1 & (+7) \\ + & 0 & 0 & 1 & 1 & & (+3) \\ \hline & 1 & 0 & 1 & 0 & & (-6) \end{array}$$

$$\begin{array}{rcccccl} & 1 & 0 & 0 & 0 & & \\ & \nearrow & \nearrow & \nearrow & \nearrow & & \\ & & 1 & 1 & 0 & 0 & (-4) \\ + & 1 & 0 & 1 & 1 & & (-5) \\ \hline & 0 & 1 & 1 & 1 & & (+7) \end{array}$$

El sobre flujo se detecta como la OR exclusiva de los acarreos de los dos bits más significativos

Resta como suma del complemento a dos

- La suma de números con signo es idéntica a la de números binarios sin signo.
- La mayoría de las computadoras implementan la resta como la suma del minuendo con el complemento a dos del sustraendo

$\text{resultado} = \text{minuendo} + \text{not}(\text{sustraendo}) + 1$

- El acarreo es el negado del préstamo de la resta

Calcular 1010 1100b - 1100 1010b

- Complemento a dos de 1100 1010 es 0011 0110

- Resta binaria

$$\begin{array}{r} 0 \quad 0111 \quad 100 \quad \text{acarreo} \\ \quad 1010 \quad 1100 \\ + \quad 0011 \quad 0110 \\ \hline \quad 1110 \quad 0010 \end{array}$$

$$\begin{array}{r} 1 \quad 1000 \quad 010 \quad \text{préstamo} \\ \quad 1010 \quad 1100 \\ - \quad 1100 \quad 1010 \\ \hline \quad 1110 \quad 0010 \end{array}$$

Ejercicios

Determine el Estado de las banderas de Cero, acarreo, sobreflujo y signo (negativo)

a) 67E2h	b) FA4Eh	c) 8A3Ch
+ 3F5Ah	+ 45ABh	+ B3C1h

Represente los siguientes números en la notación de complemento a dos de 8 bits: a) +76, b)-28 c) -1, d) -100

A que número equivale el F325h en como número con signo y sin signo de 16 bits

Rangos de datos garantizados por el estándar C99

C data type	Minimum	Maximum
char	−127	127
unsigned char	0	255
short [int]	−32,767	32,767
unsigned short [int]	0	65,535
int	−32,767	32,767
unsigned [int]	0	65,535
long [int]	−2,147,483,647	2,147,483,647
unsigned long [int]	0	4,294,967,295
long long [int]	−9,223,372,036,854,775,807	9,223,372,036,854,775,807
unsigned long long [int]	0	18,446,744,073,709,551,615

Rangos de datos en C en una maquina de 32 bits

C data type	Minimum	Maximum
char	−128	127
unsigned char	0	255
short [int]	−32,768	32,767
unsigned short [int]	0	65,535
int	−2,147,483,648	2,147,483,647
unsigned [int]	0	4,294,967,295
long [int]	−2,147,483,648	2,147,483,647
unsigned long [int]	0	4,294,967,295
long long [int]	−9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long long [int]	0	18,446,744,073,709,551,615

Rangos de datos en C en una maquina de 64 bits

C data type	Minimum	Maximum
char	−128	127
unsigned char	0	255
short [int]	−32,768	32,767
unsigned short [int]	0	65,535
int	−2,147,483,648	2,147,483,647
unsigned [int]	0	4,294,967,295
long [int]	−9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long [int]	0	18,446,744,073,709,551,615
long long [int]	−9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long long [int]	0	18,446,744,073,709,551,615

Conversión (cast) entre sin signo y con signo

- Los bits se mantienen idénticos, solo cambia la forma en que se interpretan

```
short int v=-12345; //CFC7h
```

```
unsigned short int uv=(unsigned short)v;
```

```
printf("v=%hd, uv=%hu\n", v, uv);
```

- Resultado:

```
v = -12345, uv = 53191
```

Cambio de tamaño (Extensión y truncamiento)

- Truncamiento: Solo se conservan los bits menos significativos. Ejemplo: 1234h $\rightarrow$ 34h
- Extensión sin signo: se añaden ceros. Ejemplo: 34h $\rightarrow$ 0034h
- Extensión con signo: se repite el bit de signo.
 - Ejemplo 1: 34h $\rightarrow$ 0034h
 - Ejemplo 2: 81h $\rightarrow$ FF81h

Datos multibyte

- La mayoría de las maquinas tienen localidades de memoria de 8 bits
- Para almacenar datos de más de 8 bits se debe usar más de una localidad de memoria
- Big-Endian: Se almacena el Byte más significativo en la dirección más baja
- Little-Endian: Se almacena el Byte menos significativo en la dirección más baja

Ejemplo de datos multibyte

12345678h

- Big Endian



- Little Endian



Representación de cadenas de caracteres

- Se usa un código para representar cada carácter
 - ASCII – 1968, 7bits
 - ISO-8859-1, Windows-1252, Mac OS Roman
 - Unicode: UTF-8, UTF-16 y UTF32
- Se coloca un carácter en seguida del otro en direcciones de memoria contiguas
- Ejemplo: "HOLA"

48h	4Fh	4Ch	41h
-----	-----	-----	-----

Operaciones bit a bit en C

- And bit a bit &

A	B	A&b
0	0	0
0	1	0
1	0	0
1	1	1

- Propiedades

$$A \& 0 = 0$$

$$A \& 1 = A$$

- Or bit a bit |

A	B	A b
0	0	0
0	1	1
1	0	1
1	1	1

- Propiedades

$$A | 0 = A$$

$$A | 1 = 1$$

Operaciones bit a bit en C (2)

- XOR bit a bit $\wedge$

A	B	$A \wedge B$
0	0	0
0	1	1
1	0	1
1	1	0

- Not bit a bit $\sim$

A	$\sim A$
0	1
1	0

- Propiedades

$$A \wedge 0 = A$$

$$A \wedge 1 = \text{not } A$$

Poner un bit en 1

- Se utilizan las propiedades de la operación |

$$A|0=A$$

$$A|1=1$$

- Se usa una mascara con 1 en los bits que quieren modificarse y 0 en los que no deben modificarse

- Ejemplo: poner bit 2 en 1

`var|=0b0000 0100; //Constante en binario`

`var|=0x04; //Constante en hexadecimal`

Poner un bit en 0

- Se utilizan las propiedades de la operación &
 $A \& 0 = 0$
 $A \& 1 = A$
- Se usa una mascara con 0 en los bits que quieren modificarse y 1 en los que no deben modificarse
- Ejemplo: poner bit 2 en 0
 $\text{var} \&= 0b1111\ 1011; \text{var} \&= \sim 0b0000\ 0100;$
 $\text{var} \&= 0xFB; \text{var} \&= \sim 0x04;$

Negar un bit

- Se utilizan las propiedades de la operación $\wedge$

$$A \wedge 0 = A$$

$$A \wedge 1 = \text{not } A$$

- Se usa una mascara con 1 en los bits que quieren modificarse y 0 en los que no deben modificarse

- Ejemplo: negar el bit 2

`var ^= 0b0000 0100; //Constante en binario`

`var ^= 0x04; //Constante en hexadecimal`

Teoremas de DeMorgan

$$\overline{A+B} = \bar{A} \cdot \bar{B}$$

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

A	B	$\overline{A+B}$	$\overline{A \cdot B}$	$\bar{A}$	$\bar{B}$	$\bar{A} + \bar{B}$	$\bar{A} \cdot \bar{B}$
0	0	1	1	1	1	1	1
0	1	0	1	1	0	0	1
1	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0

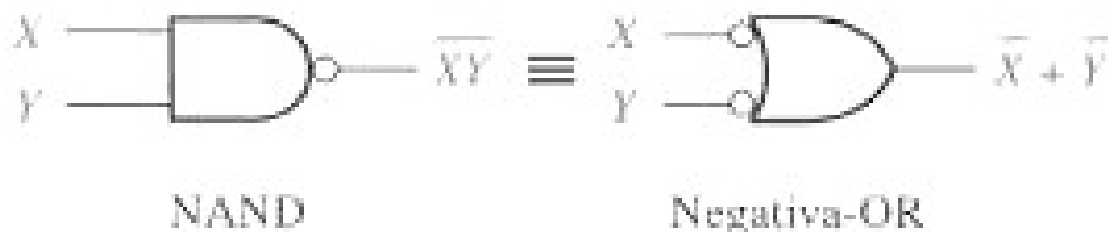
Teoremas de DeMorgan (2)

$$\overline{A+B} = \bar{A} \cdot \bar{B}$$

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

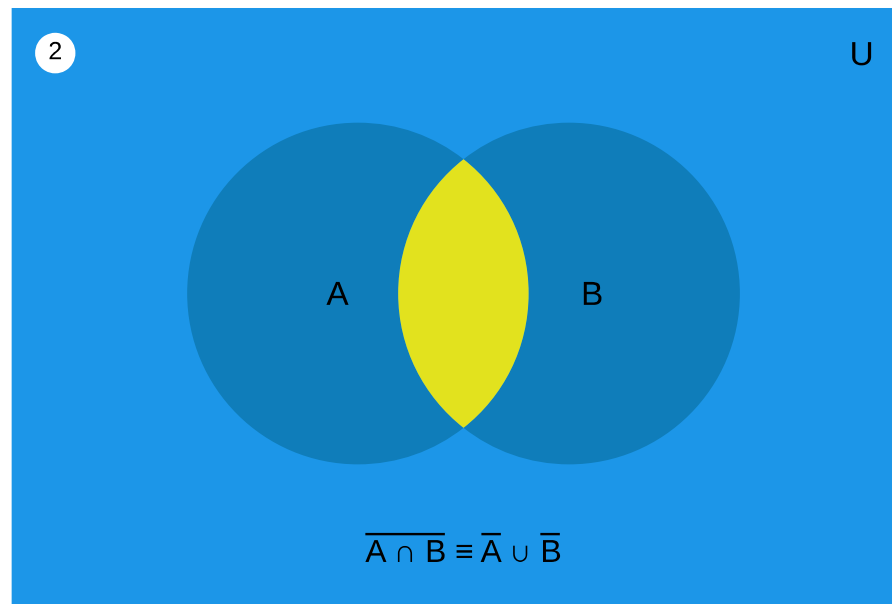
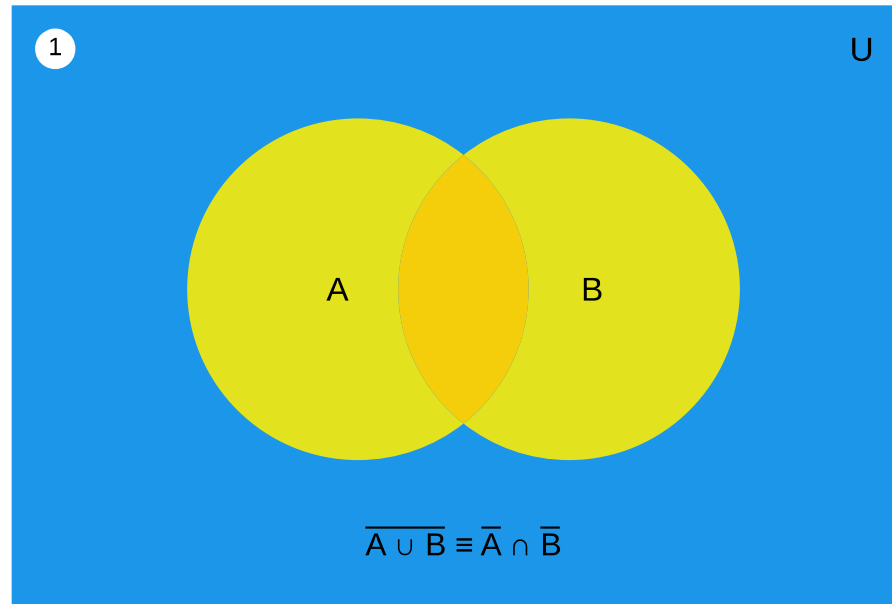


Entradas		Salida	
X	Y	$\overline{X+Y}$	$\bar{X}\bar{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0



Entradas		Salida	
X	Y	$\overline{XY}$	$\bar{X} + \bar{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Teoremas de DeMorgan (3)



Operaciones de corrimiento

- Corrimiento a la izquierda: los bits se recorren una posición hacia el bit más significativo, el bit más significativo sale de la variable y entra un cero al bit menos significativo



- El segundo operando es el número de veces que se repite el corrimiento
- Ejemplo:

```
unsigned char a=0b 0010 0110;
```

```
a=a<<1; //a==0b 0100 1100;
```

Corrimiento lógico (sin signo) a la derecha

- Los bits se recorren una posición hacia el bit menos significativo, el bit menos significativo sale de la variable y entra un cero al bit más significativo



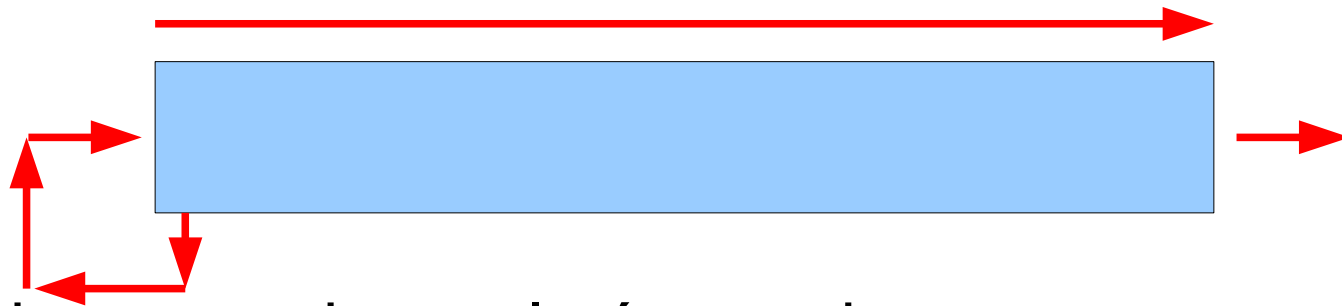
- El segundo operando es el número de veces que se repite el corrimiento
- Ejemplo:

```
unsigned char a=0b 1010 0110;
```

```
a=a>>1; //a==0b 0101 0011;
```

Corrimiento aritmético (con signo) a la derecha

- Los bits se recorren una posición hacia el bit menos significativo, el bit menos significativo sale de la variable y el bit más significativo se conserva



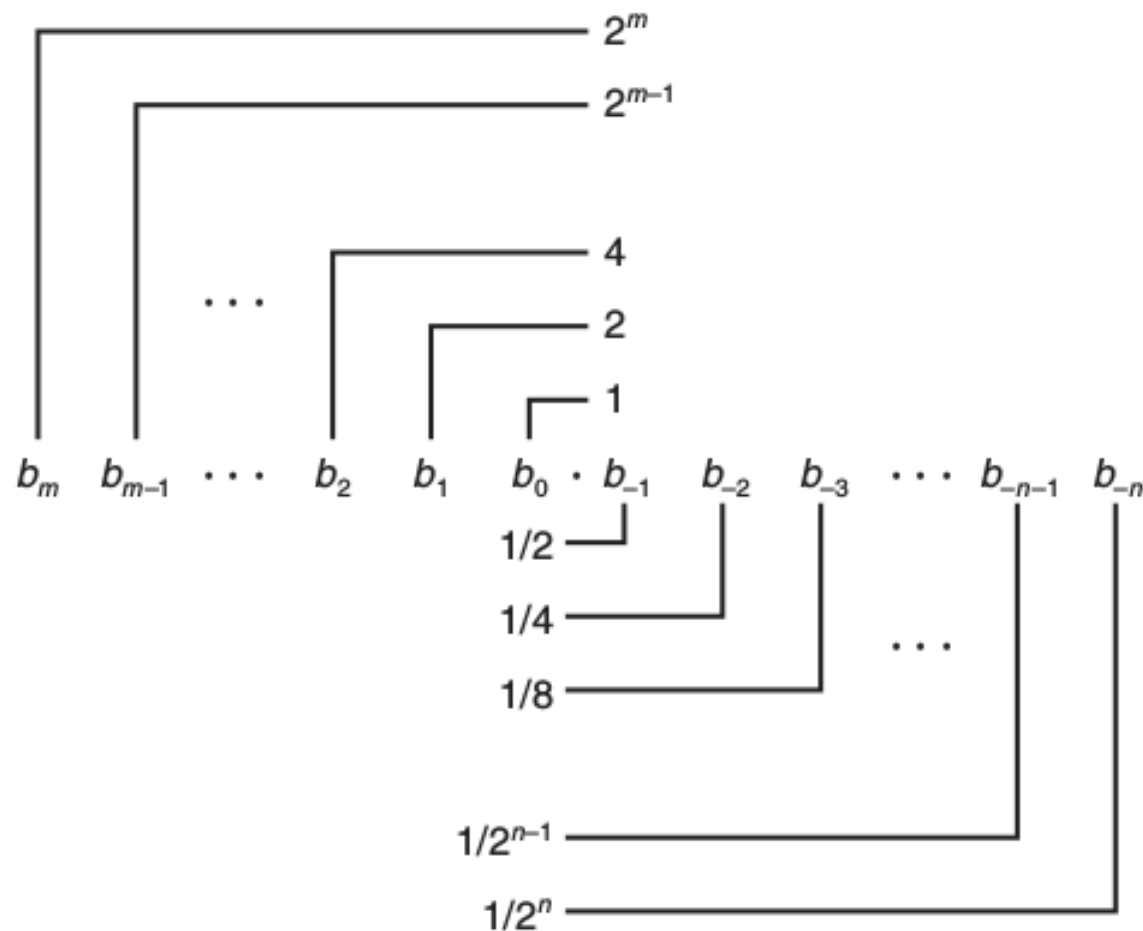
- El segundo operando es el número de veces que se repite el corrimiento
- Ejemplo:

```
char a=0b 1010 0110;
```

```
a=a>>1; //a==0b 1101 0011;
```

Números binario fraccionarios

- Bits a la derecha del punto binario tienen un peso que es una potencia negativa de 2



Formato de punto flotante IEEE 754-1985

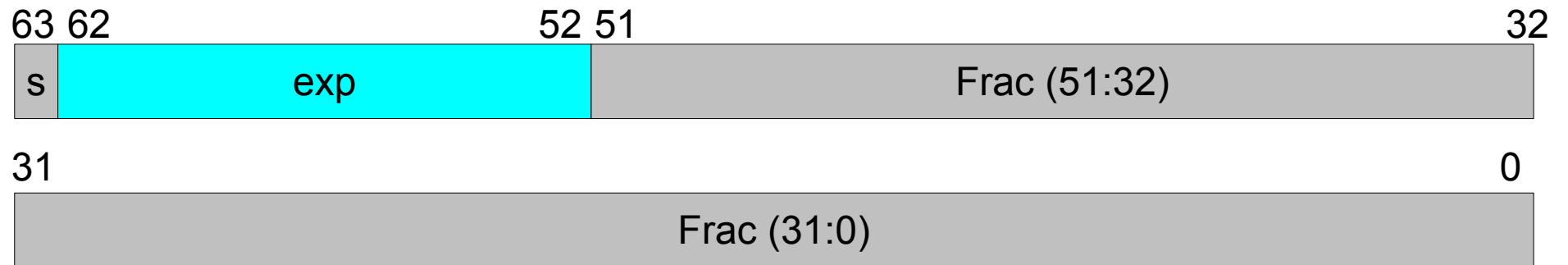
- Un numero V se representa por los campos s , M y E , de forma que $V=(-1)^s \times M \times 2^E$.
- s determina el signo: $s=1$ para un número negativo, $s=0$ para uno positivo
- M es la mantisa, una numero fraccionario con rango entre 1.0 y $2.0-\epsilon$ o entre 0.0 y $1.0-\epsilon$
- E es el exponente al que se eleva la base

Formatos IEEE comunes

- Precisión sencilla (float)



- Precisión doble (double)



Categorías de valores de punto flotante en precisión sencilla

- Normalizado



- Denormalizado



- Infinito



- NaN



Valores normalizados

- El exponente se obtiene como $E = \text{exp} - \text{sesgo}$, donde sesgo es $2^{k-1} - 1$ (127 para single y 1023 para double)
- La mantisa se considera normalizada, es decir, con solo un uno a la izquierda del punto binario. Este uno queda implícito, no se incluye en el campo f.
- $1.0 < M < 2.0$

Valores denormalizados

- La mantisa M se encuentra desnormalizada, sin el uno implícito. $0 < M < 1$
- El exponente es $E = 1 - \text{sesgo}$, no $-\text{sesgo}$ para compensar por el uno implícito
- El $+0.0$ es representado por todos los bits en cero. Existe un -0.0 . Ambos son casos especiales de valores denormalizados
- Los valores normalizados permiten representar números igualmente espaciados cercanos a cero (subflujo gradual)