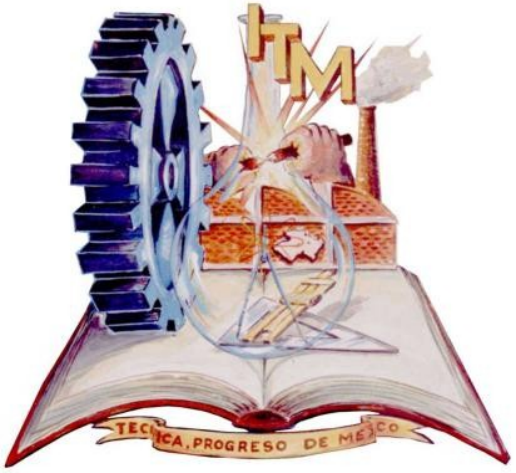
Instituto Tecnológico de Morelia



Electrónica Digital

Códigos numéricos

M.C. Miguelangel Fraga Aguilar

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.

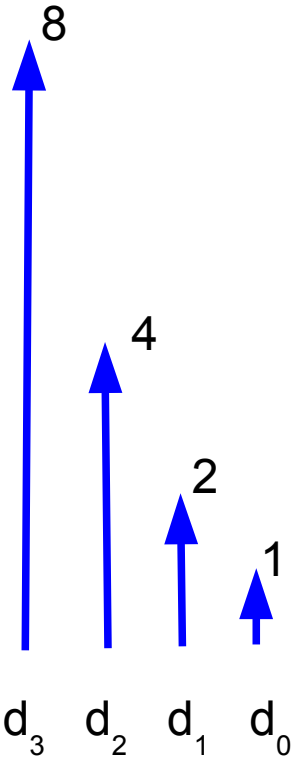
$$V = \sum_{i=0}^{n-1} d_i 2^i; \qquad d_i \in \{0,1\}$$

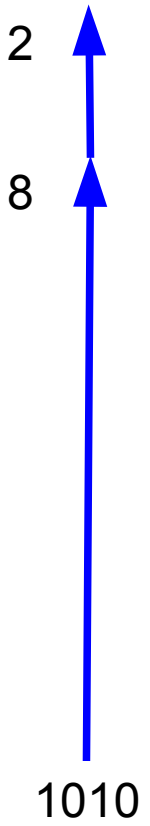
¿Se entiende La notación????

i	4	3	2	1	0
Peso	16	8	4	2	1
Bit	1	0	1	1	0

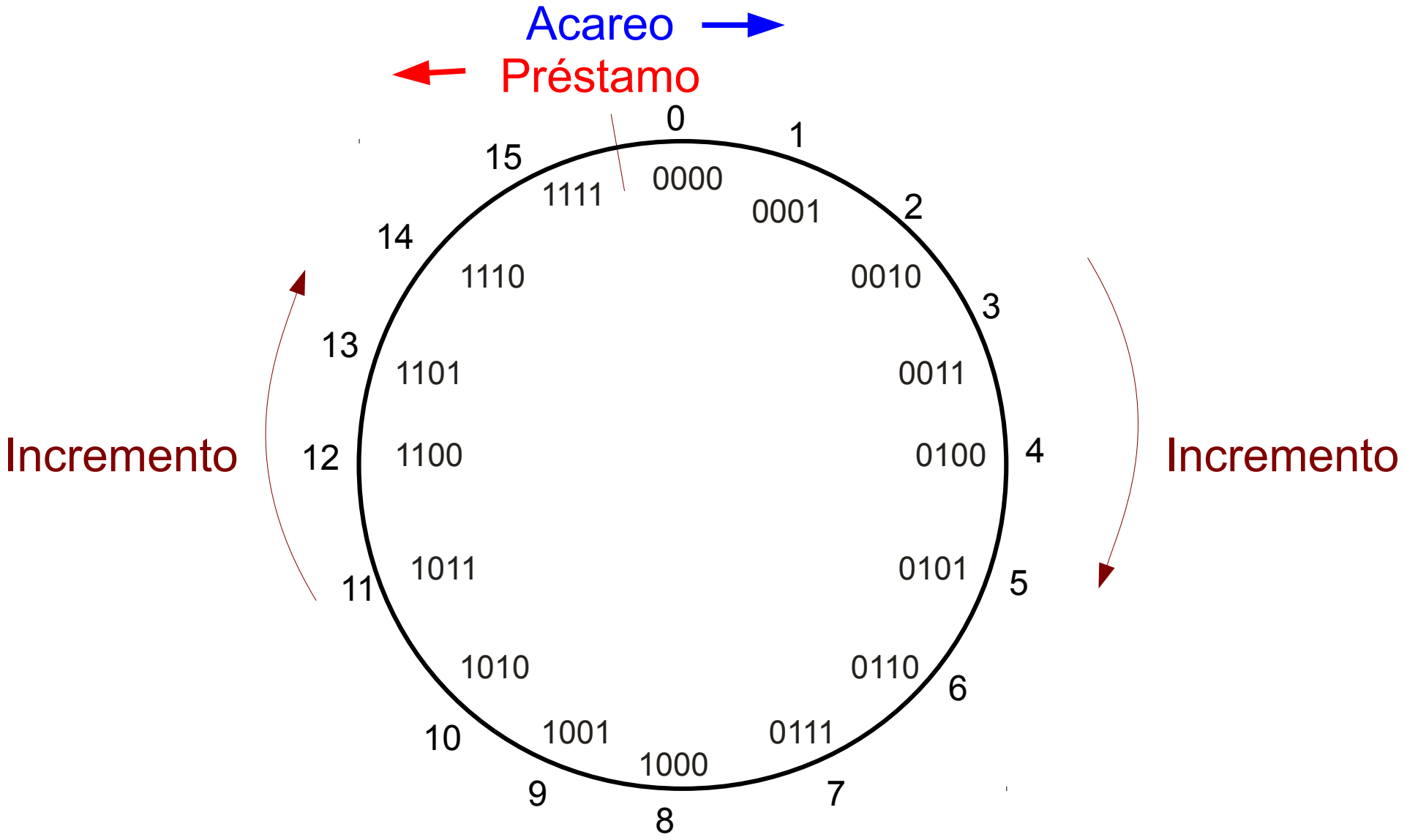
Rango representable: 0 … 2^n-1

Números sin signo como vectores





Número finito de dígitos
Circulo numérico



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

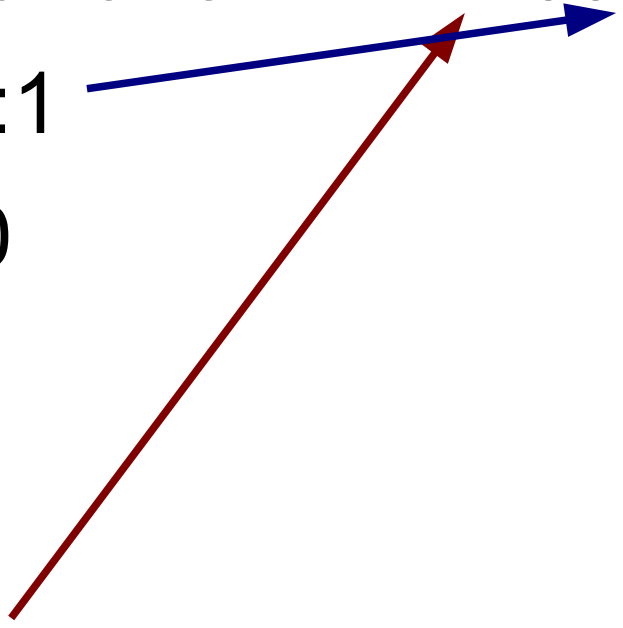
  25/2=12:1

  12/2=6:0

  6/2=3:0

  3/2=1:1

  1/2=0:1



Bit menos significativo

Bit más significativo

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

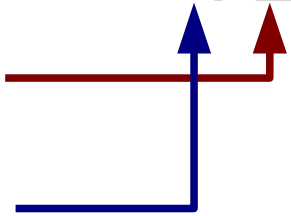
$$V = \sum_{i=0}^{n-1} d_i 16^i ; \qquad d_i \in \{0 \ldots F\}$$

Ejemplos de notación hexadecimal

- 1010 1100 b → AC h
- 0101 0110 b → 56h
- 126d → 7E h

  126/16=7:14

  7/16=0:7



- BEBAh → 1011 1110 1011 1010b
- BEBAh → $11*16^3+14*16^2+11*16^1+10*16^0=$

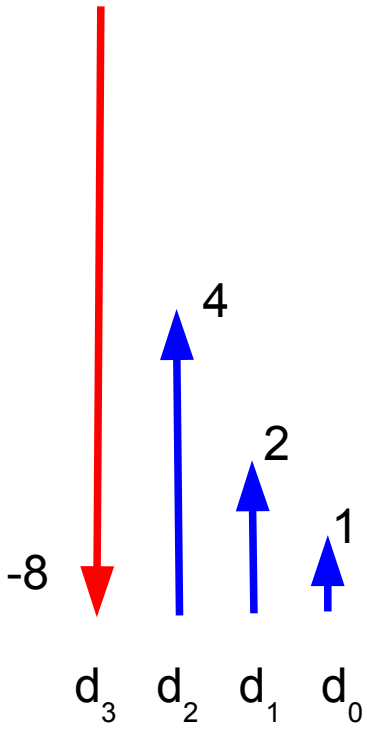
  $11*4096+14*256+11*16+10*1=48826$

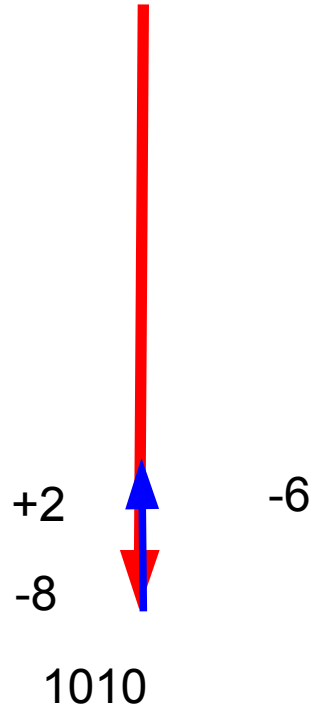
Números con signo en Complemento a dos

- 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

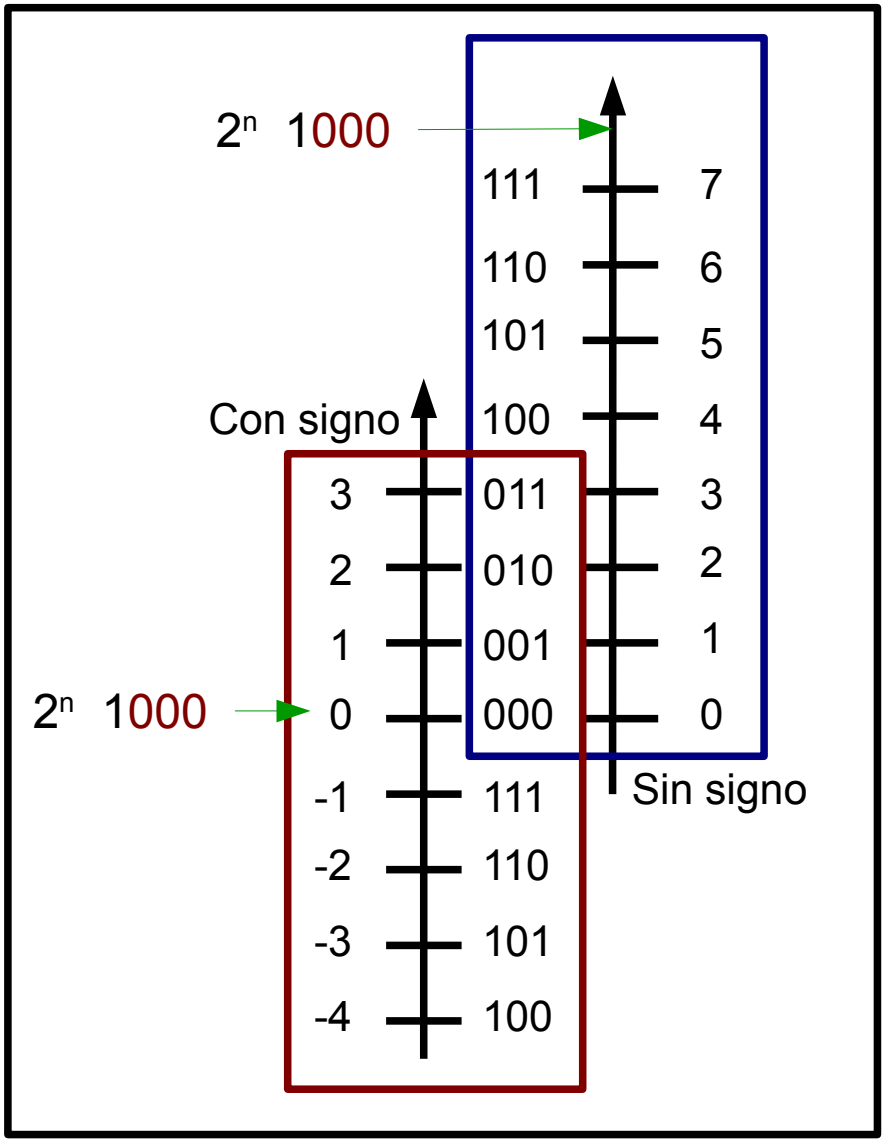
Números con signo como vectores

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

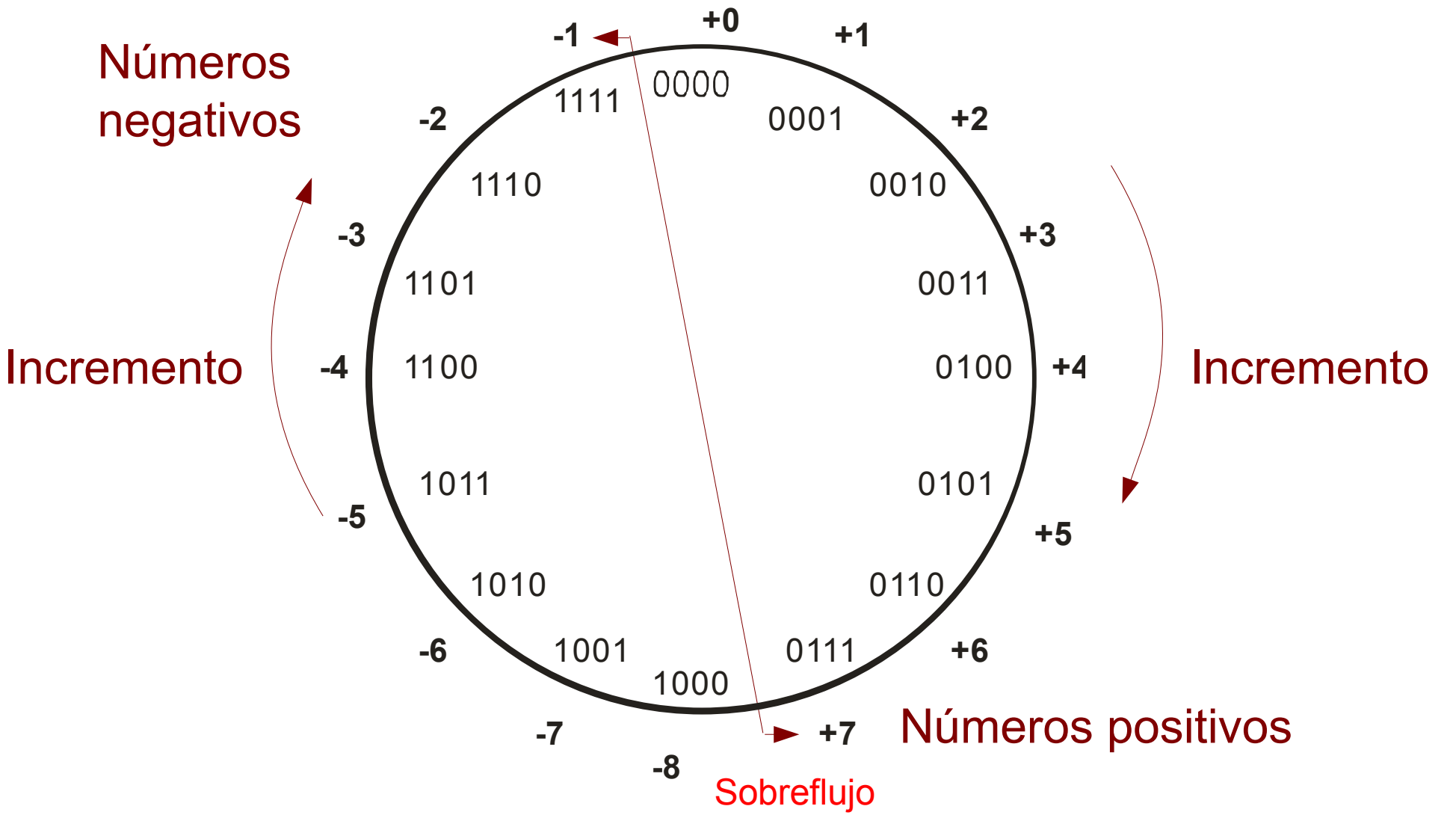
También se puede calcular negando todos los bits y sumándole 1, o por inspección copiando los bits hasta el primer uno de derecha a izquierda, y el negado de los restantes bits.

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

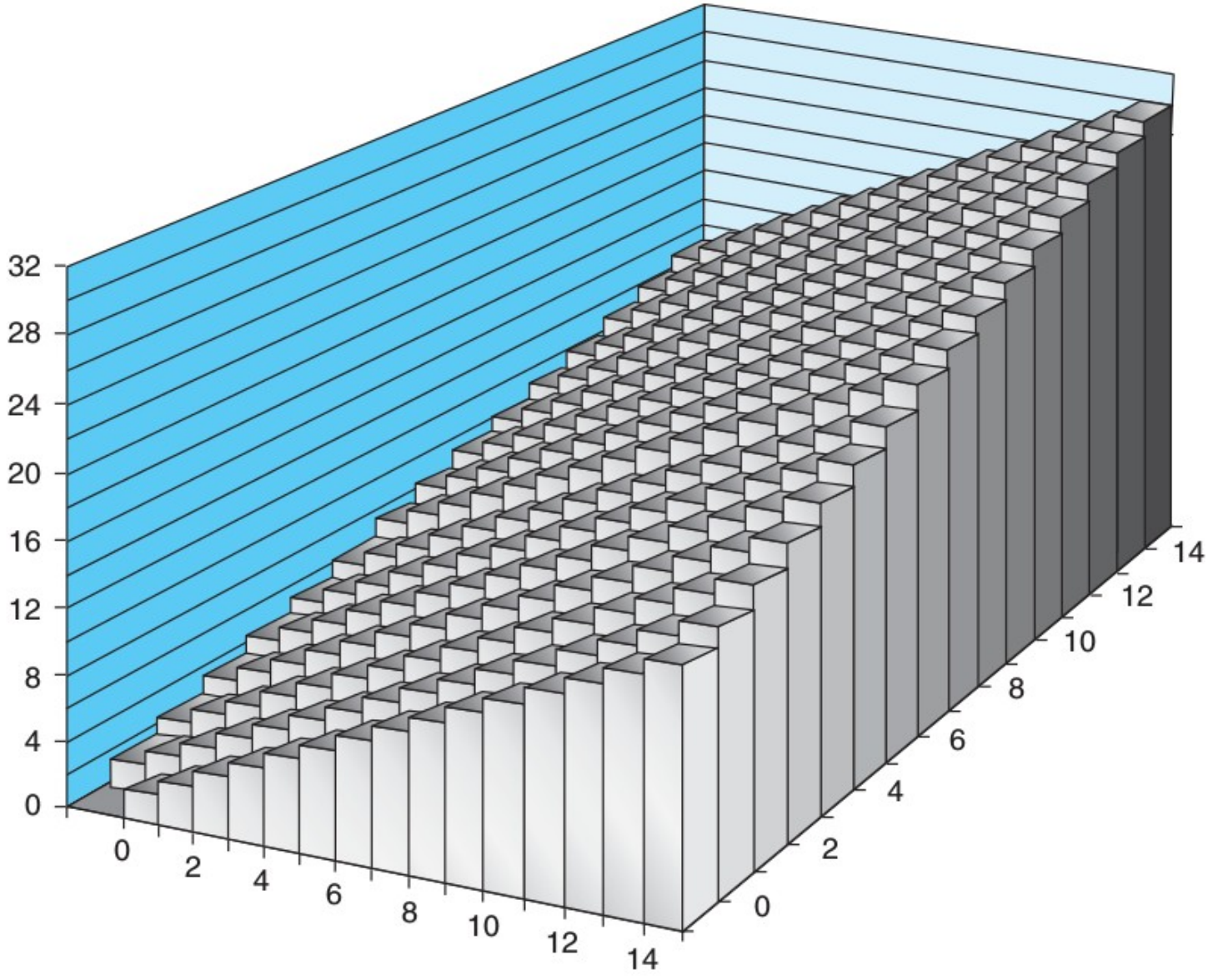
Se resuelven los problemas del complemento a uno

(+3) 0011	(-3) 1101	(-5) 1011	(+7) 0111
+ (+2) 0010	+ (-2) 1110	+ (+3) 0011	+ (-5) 1011
(+5) 0101	(-5) 11011	(-2) 1110	(+2) 10010

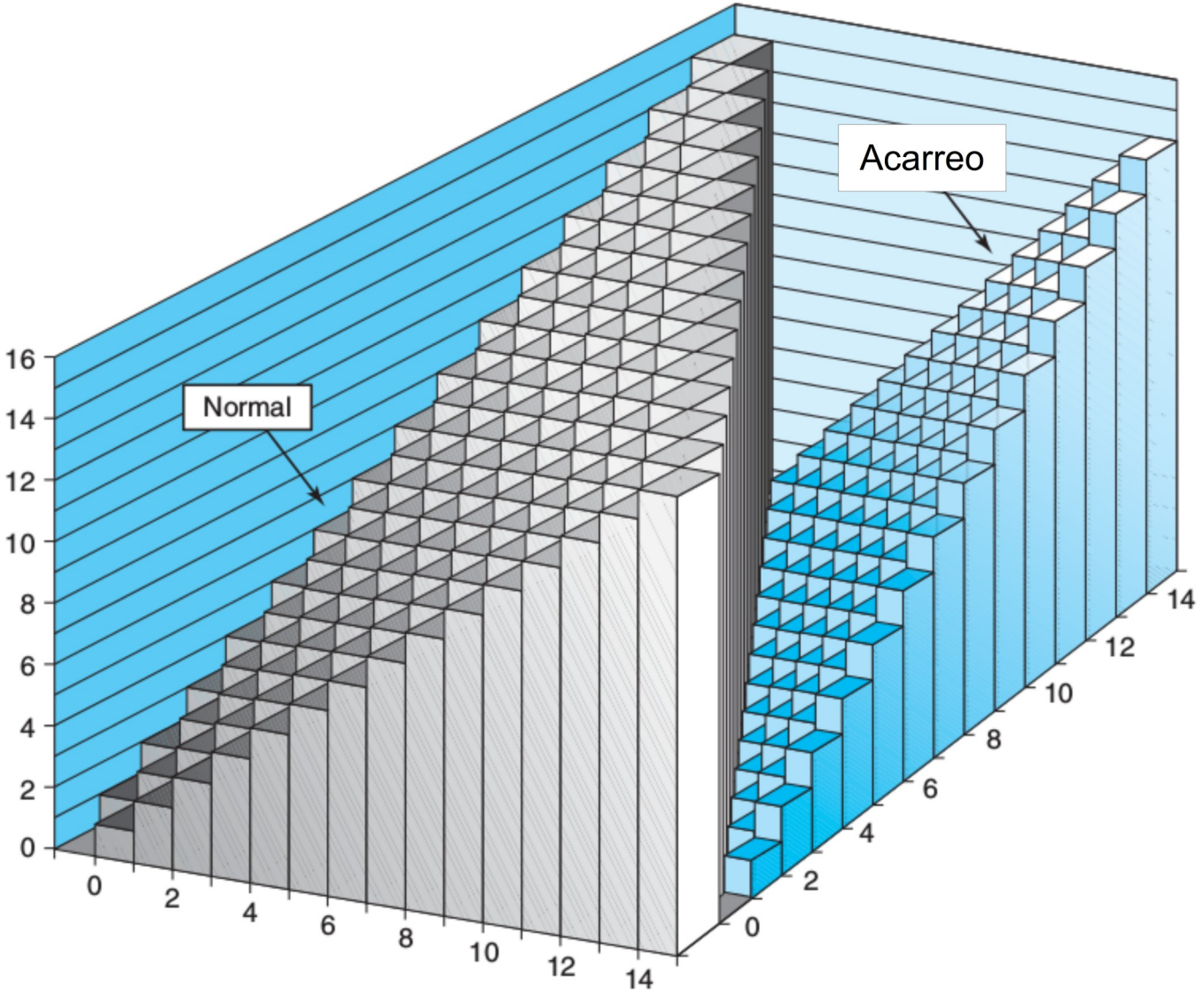
Circulo Numérico Complemento a dos



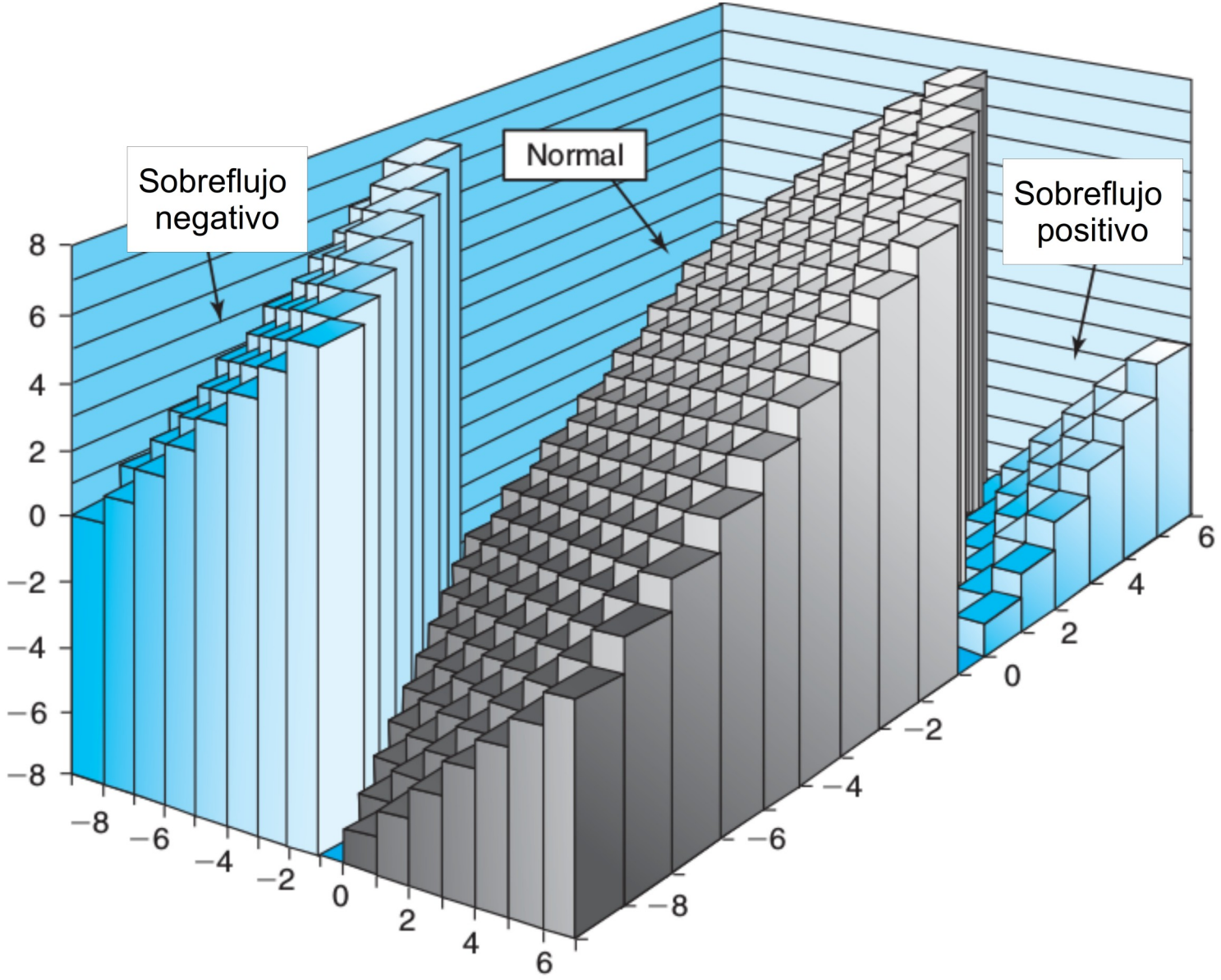
Suma entera datos de 4 bits, resultado en 5 bits



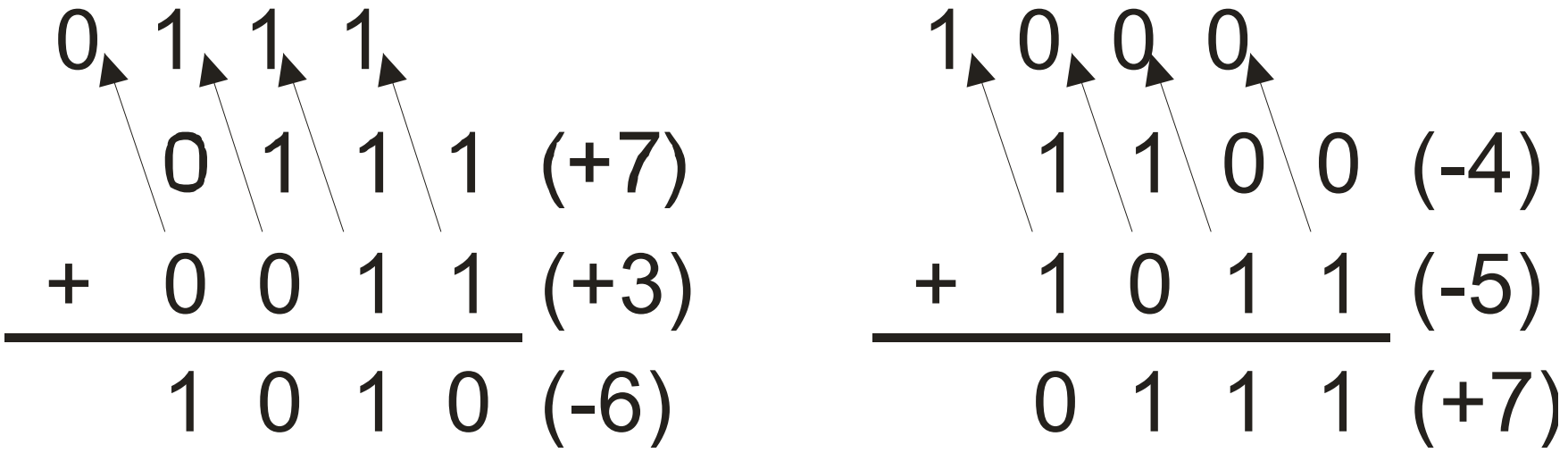
Suma entera datos de 4 bits, resultado en 4 bits



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



Sobreflujo



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

Ejercicios

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

a)	b)	c)
67E2h	FA4Eh	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=%d,uv=%u\n",v,uv);
```

- Resultado:

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

Cambio de tamaño (Extensión y truncamiento)

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

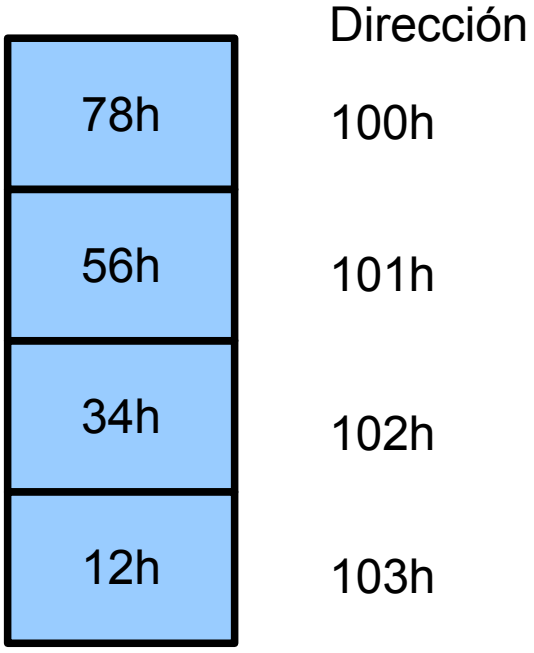
Datos multibyte

- 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

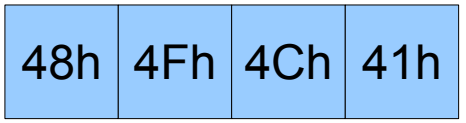
- Big Endian

- 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”



Código ASCII

USASCII code chart

b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 ↓ b_3 ↓ b_2 ↓ b_1 ↓ Row ↓ / Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Código BCD

- Se utilizan 4 bits para representar un dígito decimal, solo se permite las combinaciones correspondientes del 0 al 9
- BCD No empacado: un dígito por byte

  $01\ 02\ 05\ 07_{16}$

- BCD empacado: dos dígitos por byte

  $12\ 57_{16}$

Código BCD Aiken

- Similar al BCD pero con los pesos distribuidos en otro orden
- Logra la simetria de 4 y 5, 3 y 6, 2 y 7, 1 y 8, 0 y 9
- Útil en resta y división

DECIMAL	Aiken
	2421
0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

Código Exceso 3

- Se obtiene sumando “3” a cada combinación del código BCD natural.
- Al igual que el código Aiken cumple con la misma característica de simetría. Cada cifra es el complemento a 9 de la cifra simétrica en todos sus dígitos.

DECIMAL	BCD	Exceso 3
	8421	
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Código Gray

- Es un código no ponderado
- Entre una combinación de dígitos y la siguiente, sólo hay una diferencia de un dígito
- Se usa en sistemas de posición

Decimal	Binario	GRAY
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

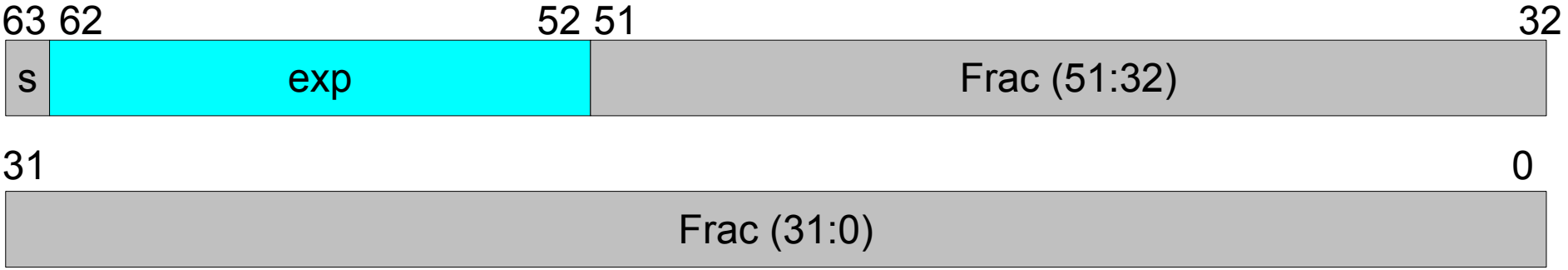
Formato de punto flotante IEEE 754-1985

- Un numero *V* se representa por los campos *s*, *M* y *E*, de forma que $V=(-1)^s x M x 2^E$.
- *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-ε o entre 0.0 y 1.0-ε
- *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=exp-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-sesgo, no -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)