Sistemas de numeración, representación de la información y de números enteros

Tema 2

¿Qué sabrás al final del capítulo?

- Representar un número en binario
- Convertir de decimal a binario y viceversa
- Convertir a hexadecimal y octal
- Realizar cambios de base tanto de la parte entera como de la parte fraccionaria en ambos sentidos
- El código de representación alfanumérico ASCII
- Representar un número entero (con signo) de distintas formas
  - Signo magnitud
  - Complemento a 1
  - Complemento a 2
- Realizar operaciones básicas con números enteros

Sistema numérico decimal

El sistema numérico decimal tiene 10 dígitos: del 0 al 9

El sistema numérico decimal tiene como base el 10: cada posición tiene un peso de 10

$\ldots 10^5\ 10^4\ 10^3\ 10^2\ 10^1\ 10^0.\ 10^{-1}\ 10^{-2}\ 10^{-3}\ 10^{-4}\ 10^{-5}\ldots$

$\mathbf{14.2} = \mathbf{1} \times 10^1 + \mathbf{4} \times 10^0 + \mathbf{2} \times 10^{-1}$

Sistema numérico binario

El sistema numérico binario tiene 2 dígitos: 0 y 1

El sistema numérico binario tiene como base el 2: cada posición tiene un peso de 2

$\ldots 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0\ .\ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ 2^{-5}\ \ldots$

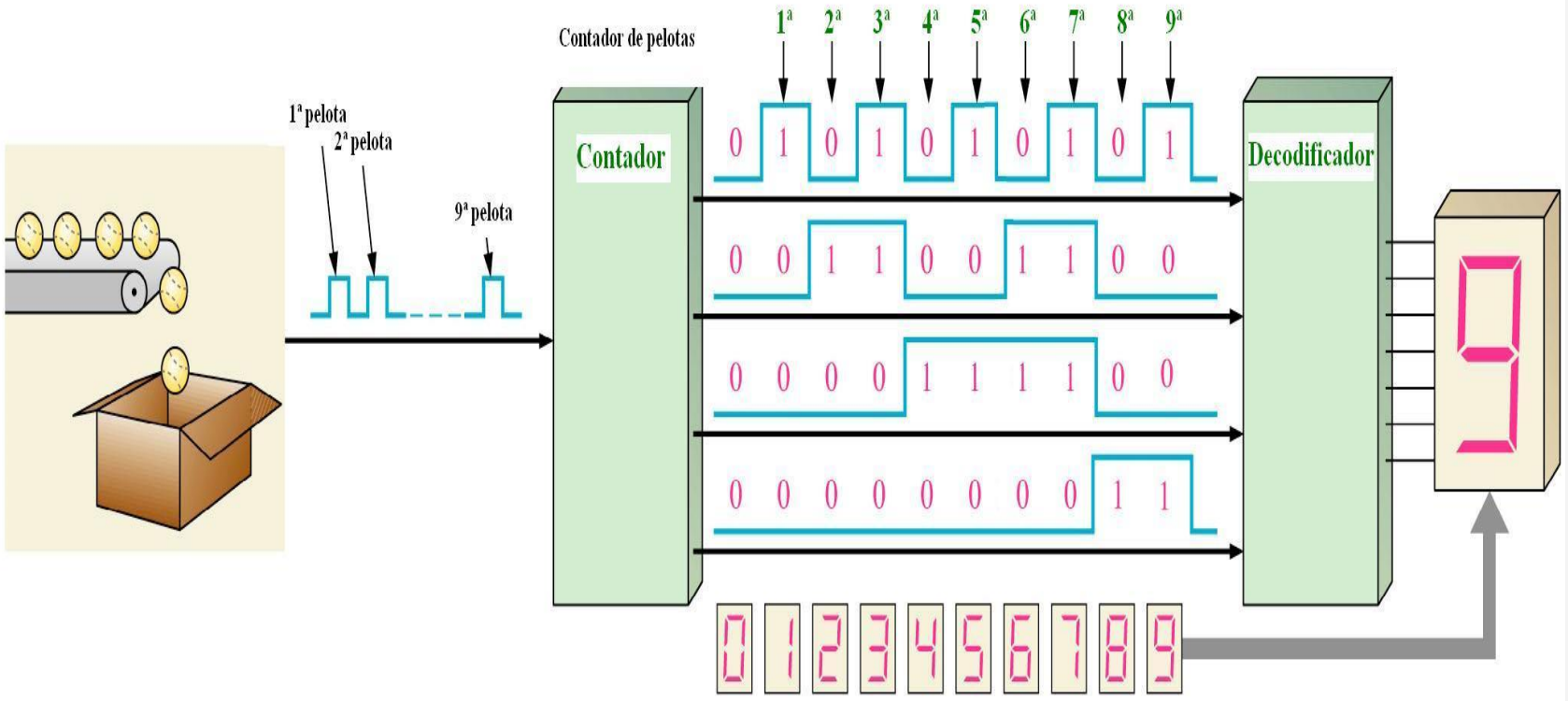
$10111.101 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} +$

$+ 0 \times 2^{-2} + 1 \times 2^{-3}$

Expresión binaria del 0 a 15

DECIMAL	BINARIO			
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

¡Sabemos contar en binario!



Otros sistemas “posicionales”

- Octal (base 8)
  - Ocho dígitos [0…7]
- Hexadecimal (base 16)
  - Dieciséis dígitos [0…9A…F]

Los primeros 256 números en las bases más importantes (I)

Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin
0	0	000	00000000	16	10	020	00010000	32	20	040	00100000	48	30	060	00110000
1	1	001	00000001	17	11	021	00010001	33	21	041	00100001	49	31	061	00110001
2	2	002	00000010	18	12	022	00010010	34	22	042	00100010	50	32	062	00110010
3	3	003	00000011	19	13	023	00010011	35	23	043	00100011	51	33	063	00110011
4	4	004	00000100	20	14	024	00010100	36	24	044	00100100	52	34	064	00110100
5	5	005	00000101	21	15	025	00010101	37	25	045	00100101	53	35	065	00110101
6	6	006	00000110	22	16	026	00010110	38	26	046	00100110	54	36	066	00110110
7	7	007	00000111	23	17	027	00010111	39	27	047	00100111	55	37	067	00110111
8	8	010	00001000	24	18	030	00011000	40	28	050	00101000	56	38	070	00111000
9	9	011	00001001	25	19	031	00011001	41	29	051	00101001	57	39	071	00111001
10	A	012	00001010	26	1A	032	00011010	42	2A	052	00101010	58	3A	072	00111010
11	B	013	00001011	27	1B	033	00011011	43	2B	053	00101011	59	3B	073	00111011
12	C	014	00001100	28	1C	034	00011100	44	2C	054	00101100	60	3C	074	00111100
13	D	015	00001101	29	1D	035	00011101	45	2D	055	00101101	61	3D	075	00111101
14	E	016	00001110	30	1E	036	00011110	46	2E	056	00101110	62	3E	076	00111110
15	F	017	00001111	31	1F	037	00011111	47	2F	057	00101111	63	3F	077	00111111

Los primeros 256 números en las bases más importantes (II)

Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin
64	40	100	01000000	80	50	120	01010000	96	60	140	01100000	112	70	160	01110000
65	41	101	01000001	81	51	121	01010001	97	61	141	01100001	113	71	161	01110001
66	42	102	01000010	82	52	122	01010010	98	62	142	01100010	114	72	162	01110010
67	43	103	01000011	83	53	123	01010011	99	63	143	01100011	115	73	163	01110011
68	44	104	01000100	84	54	124	01010100	100	64	144	01100100	116	74	164	01110100
69	45	105	01000101	85	55	125	01010101	101	65	145	01100101	117	75	165	01110101
70	46	106	01000110	86	56	126	01010110	102	66	146	01100110	118	76	166	01110110
71	47	107	01000111	87	57	127	01010111	103	67	147	01100111	119	77	167	01110111
72	48	110	01001000	88	58	130	01011000	104	68	150	01101000	120	78	170	01111000
73	49	111	01001001	89	59	131	01011001	105	69	151	01101001	121	79	171	01111001
74	4A	112	01001010	90	5A	132	01011010	106	6A	152	01101010	122	7A	172	01111010
75	4B	113	01001011	91	5B	133	01011011	107	6B	153	01101011	123	7B	173	01111011
76	4C	114	01001100	92	5C	134	01011100	108	6C	154	01101100	124	7C	174	01111100
77	4D	115	01001101	93	5D	135	01011101	109	6D	155	01101101	125	7D	175	01111101
78	4E	116	01001110	94	5E	136	01011110	110	6E	156	01101110	126	7E	176	01111110
79	4F	117	01001111	95	5F	137	01011111	111	6F	157	01101111	127	7F	177	01111111

Los primeros 256 números en las bases más importantes (III)

Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin
128	80	200	10000000	144	90	220	10010000	160	A0	240	10100000	176	B0	260	10110000
129	81	201	10000001	145	91	221	10010001	161	A1	241	10100001	177	B1	261	10110001
130	82	202	10000010	146	92	222	10010010	162	A2	242	10100010	178	B2	262	10110010
131	83	203	10000011	147	93	223	10010011	163	A3	243	10100011	179	B3	263	10110011
132	84	204	10000100	148	94	224	10010100	164	A4	244	10100100	180	B4	264	10110100
133	85	205	10000101	149	95	225	10010101	165	A5	245	10100101	181	B5	265	10110101
134	86	206	10000110	150	96	226	10010110	166	A6	246	10100110	182	B6	266	10110110
135	87	207	10000111	151	97	227	10010111	167	A7	247	10100111	183	B7	267	10110111
136	88	210	10001000	152	98	230	10011000	168	A8	250	10101000	184	B8	270	10111000
137	89	211	10001001	153	99	231	10011001	169	A9	251	10101001	185	B9	271	10111001
138	8A	212	10001010	154	9A	232	10011010	170	AA	252	10101010	186	BA	272	10111010
139	8B	213	10001011	155	9B	233	10011011	171	AB	253	10101011	187	BB	273	10111011
140	8C	214	10001100	156	9C	234	10011100	172	AC	254	10101100	188	BC	274	10111100
141	8D	215	10001101	157	9D	235	10011101	173	AD	255	10101101	189	BD	275	10111101
142	8E	216	10001110	158	9E	236	10011110	174	AE	256	10101110	190	BE	276	10111110
143	8F	217	10001111	159	9F	237	10011111	175	AF	257	10101111	191	BF	277	10111111

Los primeros 256 números en las bases más importantes (IV)

Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin	Dec	Hex	Oct	Bin
192	C0	300	11000000	208	D0	320	11010000	224	E0	340	11100000	240	F0	360	11110000
193	C1	301	11000001	209	D1	321	11010001	225	E1	341	11100001	241	F1	361	11110001
194	C2	302	11000010	210	D2	322	11010010	226	E2	342	11100010	242	F2	362	11110010
195	C3	303	11000011	211	D3	323	11010011	227	E3	343	11100011	243	F3	363	11110011
196	C4	304	11000100	212	D4	324	11010100	228	E4	344	11100100	244	F4	364	11110100
197	C5	305	11000101	213	D5	325	11010101	229	E5	345	11100101	245	F5	365	11110101
198	C6	306	11000110	214	D6	326	11010110	230	E6	346	11100110	246	F6	366	11110110
199	C7	307	11000111	215	D7	327	11010111	231	E7	347	11100111	247	F7	367	11110111
200	C8	310	11001000	216	D8	330	11011000	232	E8	350	11101000	248	F8	370	11111000
201	C9	311	11001001	217	D9	331	11011001	233	E9	351	11101001	249	F9	371	11111001
202	CA	312	11001010	218	DA	332	11011010	234	EA	352	11101010	250	FA	372	11111010
203	CB	313	11001011	219	DB	333	11011011	235	EB	353	11101011	251	FB	373	11111011
204	CC	314	11001100	220	DC	334	11011100	236	EC	354	11101100	252	FC	374	11111100
205	CD	315	11001101	221	DD	335	11011101	237	ED	355	11101101	253	FD	375	11111101
206	CE	316	11001110	222	DE	336	11011110	238	EE	356	11101110	254	FE	376	11111110
207	CF	317	11001111	223	DF	337	11011111	239	EF	357	11101111	255	FF	377	11111111

Rangos

- Rango de representación
  - con n bits pueden representarse $\mathbf{2^n}$ números diferentes
  - con n bits el rango será: $\mathbf{[2^n - 1 .. 0]}$
  - Ejemplo: 4 bits -> 2^4 (16_{10}) números diferentes,

    en el rango [15 .. 0]
- ¿Bits necesarios para representar un rango de N números decimales?
  - Respuesta: $\mathbf{n = \log_2 N}$
  - Ejemplo: 16_{10} -> $n = \log_2 16 = 4$ bits

Conversión a base 10

- Expresar en forma polinomial y operar en base 10

- Ejemplos:
  - $BC92_{16)} = 11x16^3 + 12x16^2 + 9x16^1 + 2x16^0 = 48274_{10)}$
  - $101100.11_{2)} = 1x2^5 + 1x2^3 + 1x2^2 + 1x2^{-1} + 1x2^{-2} = 32 + 8 + 4 + 0,5 + 0,25 = 44,75_{10)}$

Conversión Decimal a Binario

Método de suma de pesos

Pesos binarios

256 128 64 32 16 8 4 2 1

357 = 256 + 64 + 32 + 4 + 1 ⟶ 101100101

Pesos binarios

1024 512 256 128 64 32 16 8 4 2 1

1937 = 1024 + 512 + 256 + 128 + 16 + 1 ⟶ 11110010001

Método de las divisiones sucesivas por 2 (fundamentos)

$$Dividendo = Divisor * Cociente + Resto$$

12	8
4	1

$12 = 8*1+4$ o bien $12 = 1*8^1 + 4*8^0$

92	8	
4	11	8
	3	1

$92 = 8*11+4$ como $11 = 8*1+3$

Sustituyendo, se puede decir que

$92 = 8*(8*1+3)+4$ o bien

$92 = \mathbf{1}*8^2 + \mathbf{3}*8^1 + \mathbf{4}*8^0$

Método de las divisiones sucesivas por 2 (ejemplos)

- $19 \mid 2$ — $1\ 9 \mid 2$ — $1\ 4 \mid 2$ — $0\ 2 \mid 2$ — $0\ 1$

  $19_{10)} = \mathbf{10011_{2)}}$

- $45 \mid 2$ — $1\ 22 \mid 2$ — $0\ 11 \mid 2$ — $1\ 5 \mid 2$ — $1\ 2 \mid 2$ — $0\ 1$

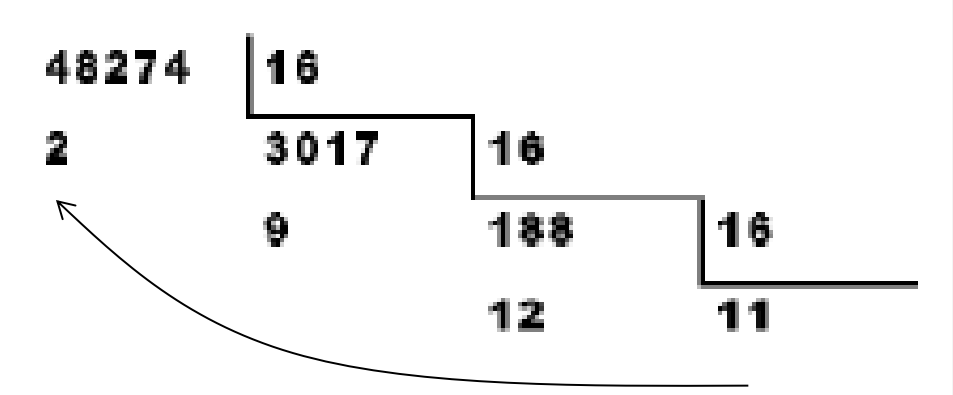
  $45_{10)} = \mathbf{101101_{2)}}$

La operación finaliza cuando el cociente es menor que la base, en nuestro caso, menor que 2

Se toma el último cociente y los restos en orden contrario a como han ido apareciendo

De base 10 a una base genérica *a*

- Se divide por la base *a* sucesivamente, tomando el último cociente y los restos en orden inverso
- Ejemplo:
  - $48274_{10)} = BC92_{16)}$



Entre bases 2 y 2^n

- De la base 2 a la base 2^n:
  - Agrupando los bits de n en n, de derecha a izquierda
  - Ejemplo: 179 -> base 8

$$179 = 10110011_2 = (010)\,(110)\,(011) = 263_8$$

- De la base 2^n a la base 2:
  - Expandiendo cada dígito por los n bits correspondientes
  - Ejemplo: $B3_{16}$ -> base 2

$$B3_{16} = (1011)(0011) = 10110011_2$$

Conversión de fracciones decimales a binario

Método de suma de pesos

Pesos binarios

64 32 16 8 4 2 1 .5 .25 .125 .0625

$95.6875_{10)} = 64 + 16 + 8 + 4 + 2 + 1 + .5 + .125 + .0625$

$1011111.1011_{2)}$

Método de divisiones y multiplicaciones

- División por 2 repetida de la parte entera da el entero en binario
- Multiplicación por 2 repetida de la fracción da la fracción binaria

- Ejemplo:
  - 0.6875_{10} -> base 2 -> 0.1011_{2}

0,6875x2=1,375	parte entera 1
0,375x2=0,75	parte entera 0
0,75x2=1,5	parte entera 1
0,5x2=1,0	parte entera 1

Tabla resumen

Sist. Destino / Sist. Origen	Decimal	Binario	Octal	Hexadecimal
Decimal		Divisiones sucesivas por la base (2)	Divisiones sucesivas por la base (8)	Divisiones sucesivas por la base (16)
Binario	**a)** Multiplicación por potencias de la base (2)		Agrupación de a 3 bits, comenzando desde la derecha.	Agrupación de a 4 bits, comenzando desde la derecha.
Octal	**a)** Multiplicación por potencias de la base (8)	Expresar cada cifra octal con su binario de 3 bits equivalente.		Convertir el octal a binario, y luego éste a hexadecimal.
Hexadecimal	**a)** Multiplicación por potencias de la base (16)	Expresar cada cifra octal con su binario de 4 bits equivalente.	Convertir el hxadecimal a binario, y luego éste a octal.	

Representación de información alfanumérica I
(American Standart Code for Information Interchange, ASCII)

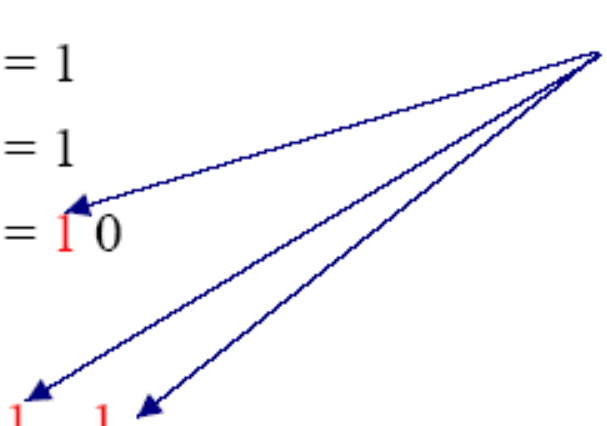
ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
0	0	NUL	16	10	DLE	32	20	(espacio)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

Representación de información alfanumérica II
(American Standart Code for Information Interchange, ASCII)

ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	□

Suma Binaria

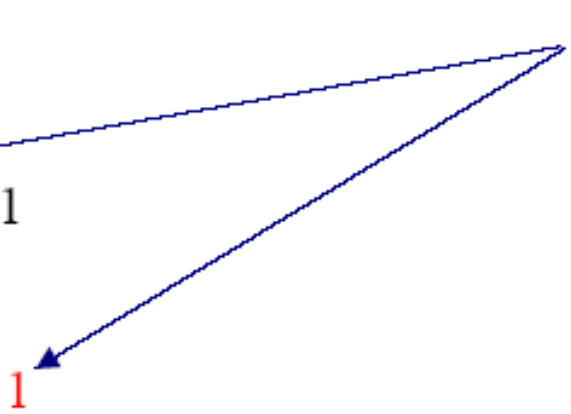
- 4 reglas básicas
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 1 0
- Ejemplo:




```
    1   1
      1   1   1    <-  3 bits
+     0   0   1    <-  3 bits
-------------------
  1   0   0   0    <-  4 bits
```

Resta Binaria

- 4 reglas básicas
  - 0 - 0 = 0
  - 1 - 1 = 0
  - 1 - 0 = 1
  - 1 0 - 1 = 1
- Ejemplo:



1

$^{0}\not{1}$ 0 1 <- 3 bits

\- 0 1 1 <- 3 bits

---

0 1 0 <- 3 bits

Multiplicación Binaria

- 4 reglas básicas
  - 0 x 0 = 0 0 x 1 = 0
  - 1 x 0 = 0 1 x 1 = 1
- Ejemplo:

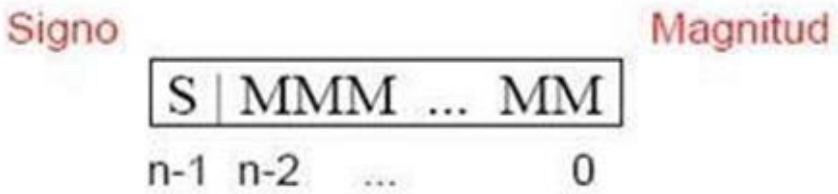
```
          1   1   1    <-  3 bits
    x     1   0   1    <-  3 bits
    ------------------
          1   1   1
      0   0   0
  1   1   1
  -------------------------
1   0   0   0   1   1    <-  6 bits
```

Números negativos

- Hasta ahora sólo hemos utilizado binarios enteros positivos
  - con n bits -> $0 \leq N \leq 2^n - 1$
- ¿Cómo representar números negativos?
  - 2 alternativas
    - *Signo-Magnitud*
    - *Complemento*

Signo-magnitud

- El primer bit representa el signo
  - 0 -> signo positivo, 1 -> signo negativo
- El resto de la cifra representa la magnitud (valor absoluto)



- Ejemplos:

$7_{10} \rightarrow 111_2$

$+7_{10} \rightarrow 0111_2$

$-7_{10} \rightarrow 1111_2$

El número de bits debe ser tal que 2 elevado a dicho número supere a 7 . En nuestro caso, 3.
Si se considera el signo, el número de bits debe ser 4

Signo-magnitud

- Rango representable con n bits
  - sin signo : $[2^n-1 \,..\, 0] \rightarrow$ total = 2^n
  - signo-magnitud : $[-(2^{n-1}-1) \,..\, (2^{n-1}-1)] \rightarrow$ total = 2^n
- Ejemplo: 3 bits

Representación	Signo-Magnitud	Sin signo
000	0	0
001	1	1
010	2	2
011	3	3
100	-0	4
101	-1	5
110	-2	6
111	-3	7

Doble representación para el 0

Signo-magnitud

- Suma y resta:
  - Caso 1: los números son de signo contrario.
    - Se resta el mayor del menor. El signo resultante será el del mayor.
    - $4 - 3 \rightarrow 0100 + 1011 \rightarrow +(100 - 011) = +1 \rightarrow 0001$
    - $-4 + 3 \rightarrow 1100 + 0011 \rightarrow -(100 - 011) = -1 \rightarrow 1001$
  - Caso 2: los números son del mismo signo
    - Se suman ambos. El signo resultante será el de los operandos.
    - Puede producirse desbordamiento **(overflow)**
    - $-2-3 \rightarrow 1010 + 1011 \rightarrow -(010 + 011) = -(101)$ => **correcto**
    - $4 + 5 \rightarrow 0100 + 0101 \rightarrow +(100 + 101) = +(1001)$ => **overflow**

En el caso del overflow, el problema se soluciona con 1 bit más

Signo-magnitud

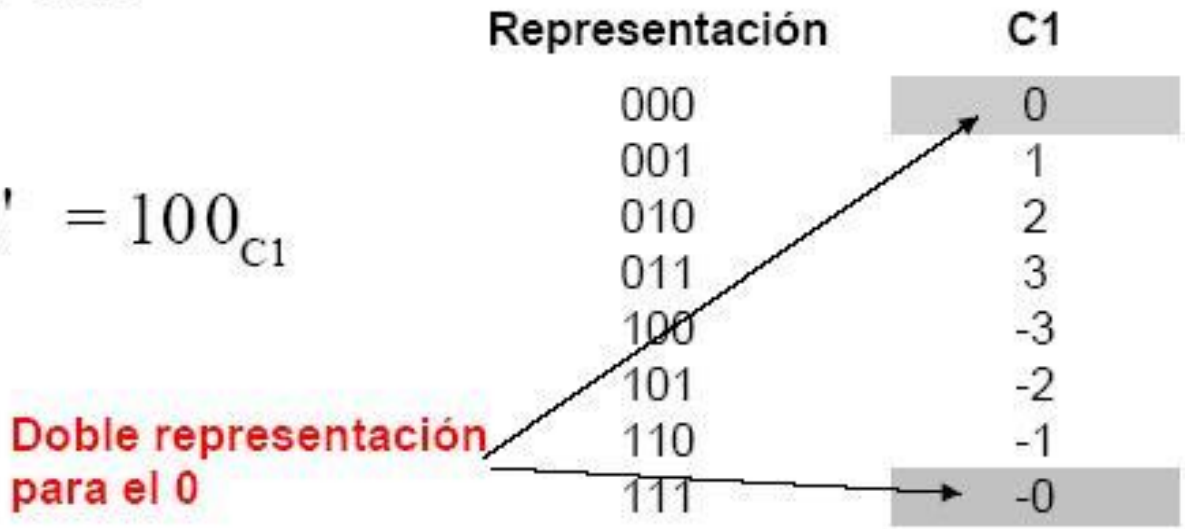
- Producto:
  - Se multiplican ambos operandos. Si son del mismo signo el resultado es positivo. Si no, es negativo.
- El tamaño del resultado es la suma de los tamaños de los operandos
  - n bits * m bits = n+m bits
- No hay overflow (desbordamiento)

Complemento

- No es necesario distinguir entre la suma y la resta
- Dos tipos
  - Complemento a la base 2 - 1 (*complemento a 1*)
  - Complemento a la base 2 (*complemento a 2*)

Complemento a 1

- Los números positivos se representan igual que en signo-magnitud. Los negativos se representan complementando todos los bits
- Rango : $[-(2^{n-1}-1) .. (2^{n-1}-1)] \rightarrow$ total = 2^n
- Ejemplos: n = 3 bits
  - $+3 \rightarrow 011$
  - $-3 \rightarrow (011)' = 100_{C1}$

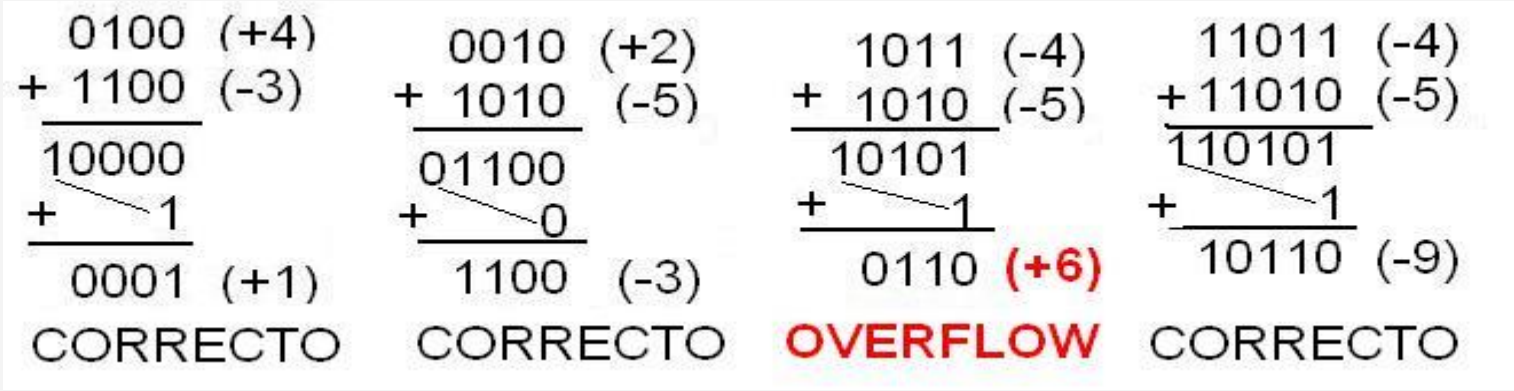


Complemento a 1

- Suma y resta:

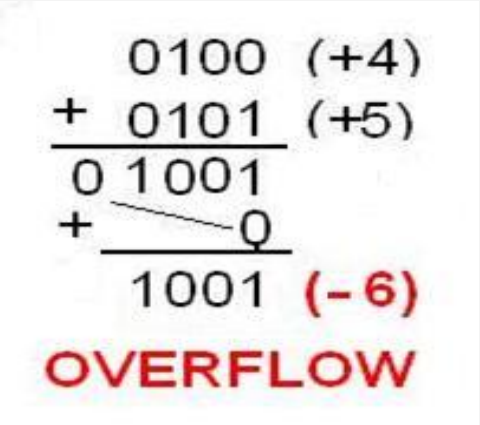
  Siempre se suma, y se suma **TAMBIÉN** el bit de acarreo (**RECIRCULAR**)

  - Signo resultante en la suma de números enteros:
    - El de los operandos si son del mismo signo. Si el signo sale distinto que el de los operandos: **overflow.**
    - El del mayor de los dos, si son de distinto signo



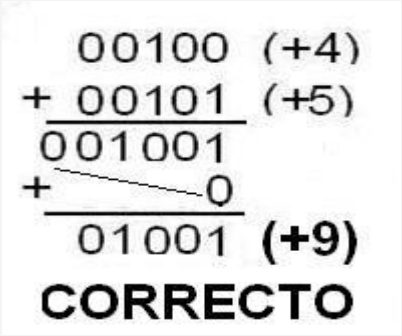
Complemento a 1

- Más casos de OVERFLOW



Como se puede ver en los dos casos de **OVERFLOW**, el número que se interpreta como complemento a 1 es erróneo.

En estos casos, se necesita 1 bit más



Complemento a 1

- Producto:
  - Como en signo-magnitud
- Extensión del signo
  - positivos: 3 -> $011_2 = 0...011_2$
  - negativos: -3 -> $100_2 = 1...100_2$

Complemento a 2

- Los números positivos se representan igual que en signo-magnitud. Los negativos como 2^n - el número.
- Rango : $[-(2^{n-1}) .. (2^{n-1}-1)] \rightarrow$ total = 2^n
- Ejemplos: n = 4 bits
  - +7 $\rightarrow 0111$
  - -7 $\rightarrow (2^4 - 7)_{10} = (16 - 7)_{10} = 9_{10}$

    $9_{10} = 1001_{C2}$
- Operaciones aritméticas:
  - como en complemento a 1

NO SE NECESITA RECIRCULAR EL ACARREO

Número	c a 2	Binario
0	0 000	0
+1	0 001	1
- 1	1 111	15
+2	0 010	2
- 2	1 110	14
+3	0 011	3
- 3	1 101	13
+4	0 100	3
- 4	1 100	12
+5	0 101	5
- 5	1 011	11
+6	0 110	6
- 6	1 010	10
+7	0 111	7
- 7	1 001	9
- 8	1 000	8

Complemento a 2

- Otras formas de calcular el complemento a 2:
  - C2(número) = C1(número) + 1

    Ej: Ca2(-90)=Ca1(-90)+1=10100101+1=10100110
  - De derecha a izquierda copiar todos los bits hasta encontrar el primer '1'. A partir de él complementar el resto de bits.
    - $+90_{Ca2}$= 0 1 0 1 1 0 1 0
      ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
      -90_{Ca2}= 1 0 1 0 0 1 1 0

Hasta el 1er. '1' todos los bits son iguales

El 1er. '1' se mantiene

El resto de valores se complementa

Complemento a 2 (ejemplos)

```
  0100  (+4)
+ 1101  (-3)
------
  0001  (+1)
```

CORRECTO

```
  0010  (+2)
+ 1011  (-5)
------
  1101  (-3)
```

CORRECTO

```
  1100  (-4)
+ 1011  (-5)
------
  0111  (+7)
```

OVERFLOW

```
 11100  (-4)
+11011  (-5)
-------
 10111  (-9)
```

CORRECTO

Comparación

Bits	Números	Rango SM	Rango C1	Rango C2	Sin Signo
4	$2^4 = 16$	[-7.. +7]	[-7.. +7]	[-8 .. +7]	[0 .. 15]
8	$2^8 = 256$	[-127.. +127]	[-127 .. +127]	[-128 .. +127]	[0 .. 255]
16	$2^{16} = 65536$	[-32767.. +32767]	[-32767.. +32767]	[-32768.. +32767]	[0 .. 65535]
N	2^n	$[-2^{n-1}-1 .. +2^{n-1}-1]$	$[-2^{n-1}-1 .. +2^{n-1}-1]$	$[-2^{n-1} .. +2^{n-1}-1]$	$[0 .. +2^n-1]$

Ya sabes…

- Representar un número en binario
- Convertir de decimal a binario y viceversa
- Convertir a hexadecimal y octal
- Cambiar de base números fraccionarios tanto a base 10 como desde base 10
- Cambiar de base entre bases potencia de 2 sin pasar por base 10
- Conocer el código ASCII
- Hacer operaciones aritméticas en binario
- Calcular el complemento a 1 y el complemento a 2 de un número binario
- Expresar números negativos en signo-magnitud, complemento a 1 y complemento a 2
- Hacer operaciones aritméticas en binario con números con signo