

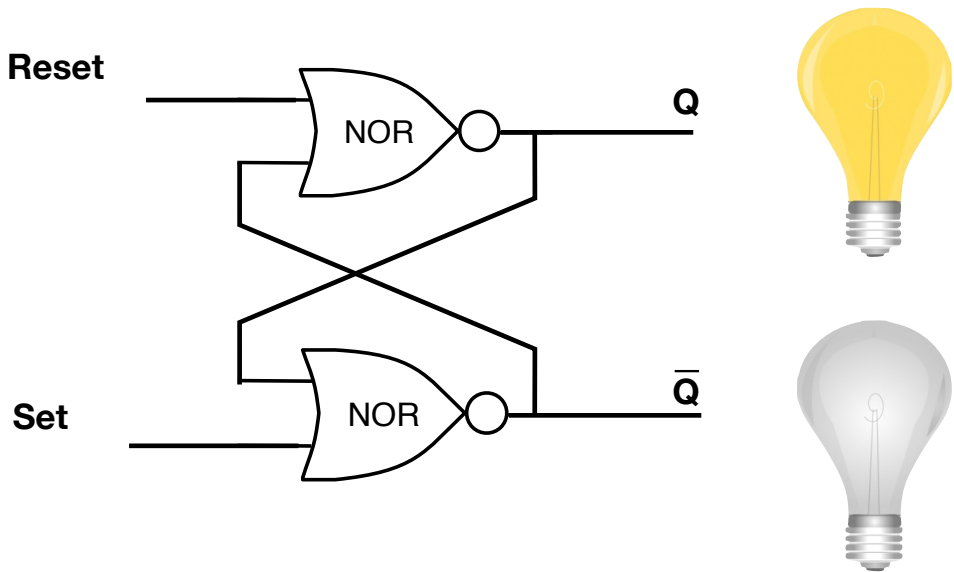
Organización del Procesador

Representando información con bits

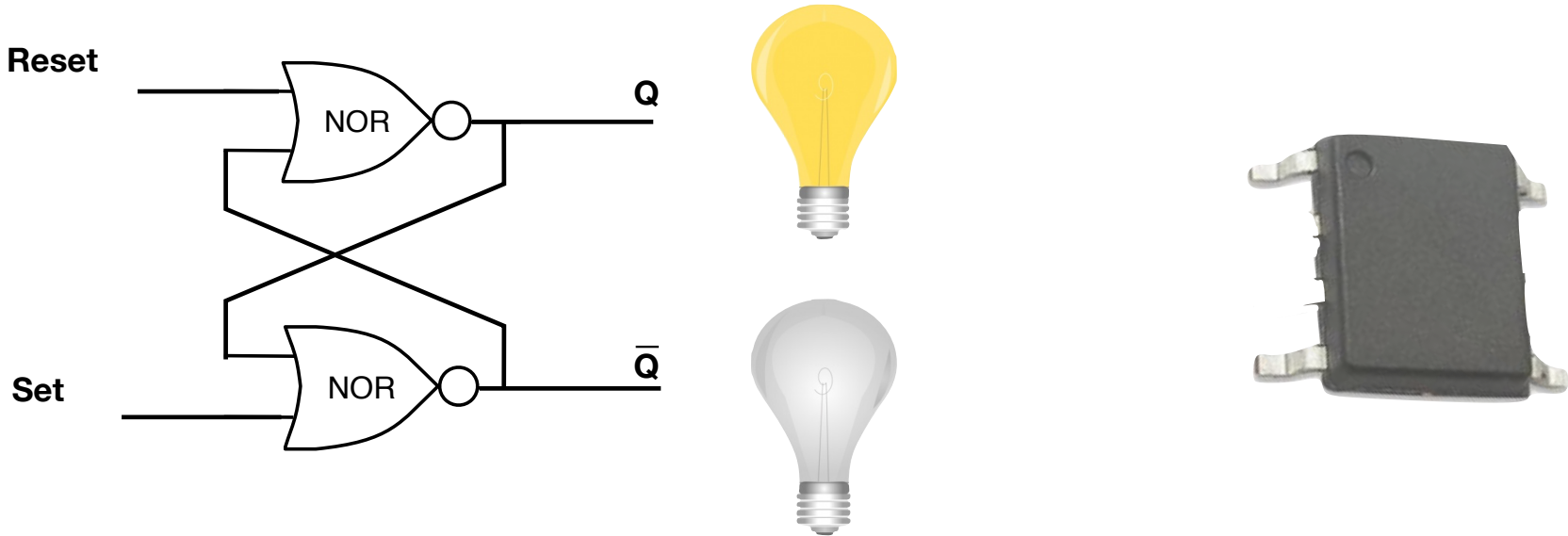
El camino a recorrer

- Un poco de Historia y Sistemas Numéricos
- Introducción a la Electrónica
- **Representación de la Información**
- Cómo computar utilizando la electricidad
- Funcionamiento abstracto de una computadora
- Assembly X86
- Micro-programación (cómo fabricar un procesador)
- Eficiencia
 - Pipelines
 - Memoria Caché
 - Memoria Virtual

Cómo representamos información con encendido y apagado, 0s y 1s



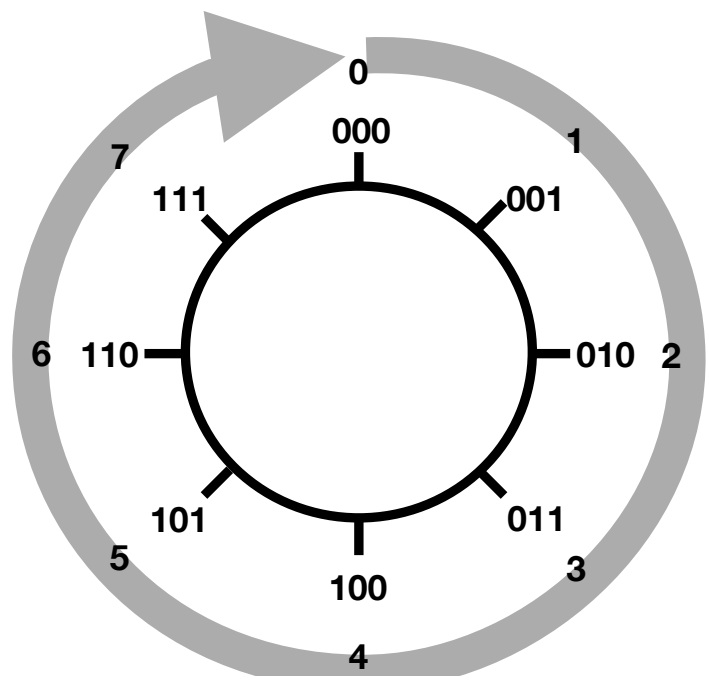
Cómo representamos información con encendido y apagado, 0s y 1s



Cuántas “cosas” puedo distinguir (representar) con 0s y 1s

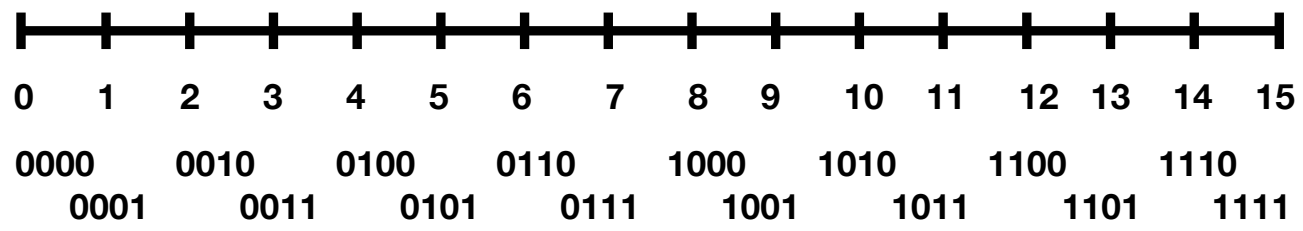
Cantidad de bits	Puedo representar	
1	2	
2	4	
3	8	
4	16	
5	32	
6	64	
7	128	
8	256	
9	512	
10	1.024	KiloByte (KB)
11	2.048	2KB
12	4.096	4KB
13	8.192	8KB
14	16.384	16KB
15	32.768	32KB
16	65.536	64KB
...
20	1.048.576	MegaByte (MB)
30	1.073.741.824	GigaByte (GB)
32	4.294.967.296	4GB
40	1.099.511.627.776	TeraByte (TB)
50	1.125.899.906.842.624	PetaByte (PB)
60	1.152.921.504.606.846.976	ExaByte (EB)
64	18.446.744.073.709.551.616	4EB

Representación de Números Naturales



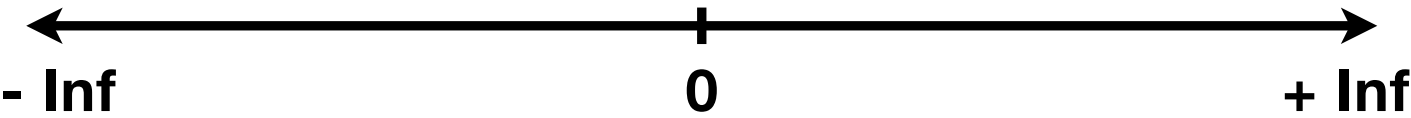
Rango de Representación

[0 a $2^n - 1$]

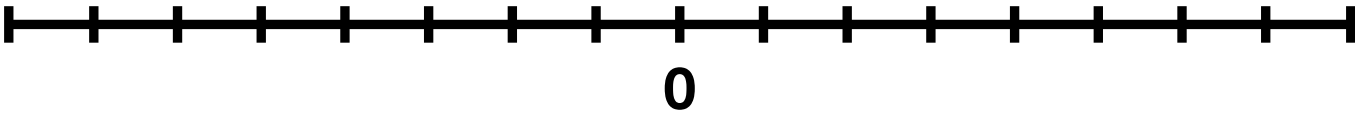


Ejemplos		
# bits	desde	hasta
3	0	7
5	0	31
10	0	1023
16	0	65.535
32	0	4.294.967.295

Representación de Números Enteros



Representación de Números Enteros



Representación de Números Enteros

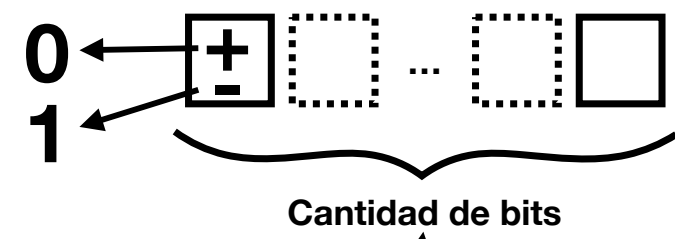
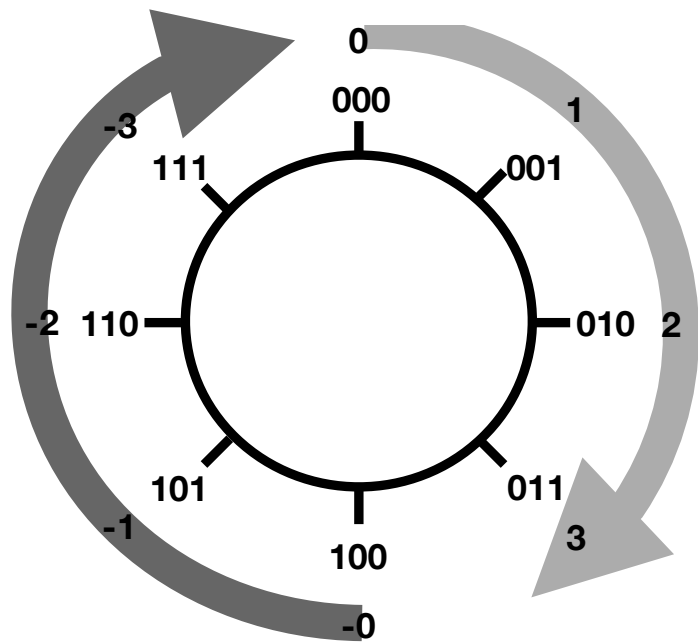
Signo y Magnitud

- Se utiliza un bit para representar el signo (bit de signo), usualmente “0” para positivos y “1” para negativos.
- Es “simple” para representar valores negativos, pero tiene una desventaja importante en la realización de operaciones aritméticas (basada en circuitos), ya que requiere considerar dos casos diferentes para la suma y resta de números.

Complemento a la Base

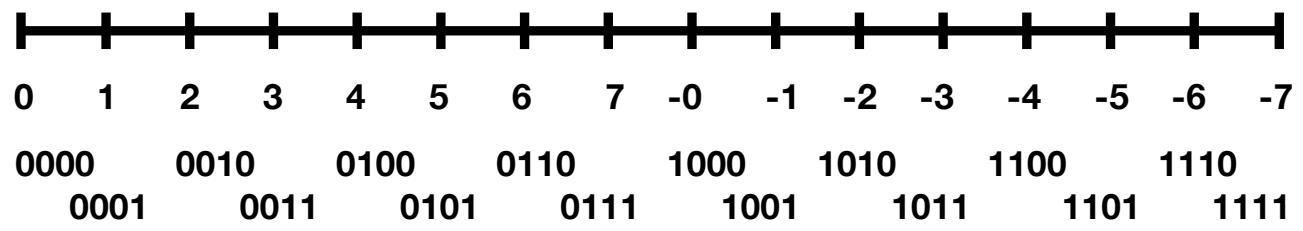
- Los números negativos se representan utilizando la forma complemento del número positivo en la misma base.
- Si bien existen algunas variantes, **complemento a dos** es el método más comúnmente utilizado en la representación de números enteros en sistemas digitales debido a su eficiencia en operaciones aritméticas.

Representación de Números Enteros - Signo y Magnitud



Rango de Representación

$$[-(2^{n-1})-1 \text{ a } (2^{n-1})-1]$$



Ejemplos		
# bits	desde	hasta
3	-3	3
5	-15	15
10	-511	511
16	-32767	32767
32	-2147483647	2147483647

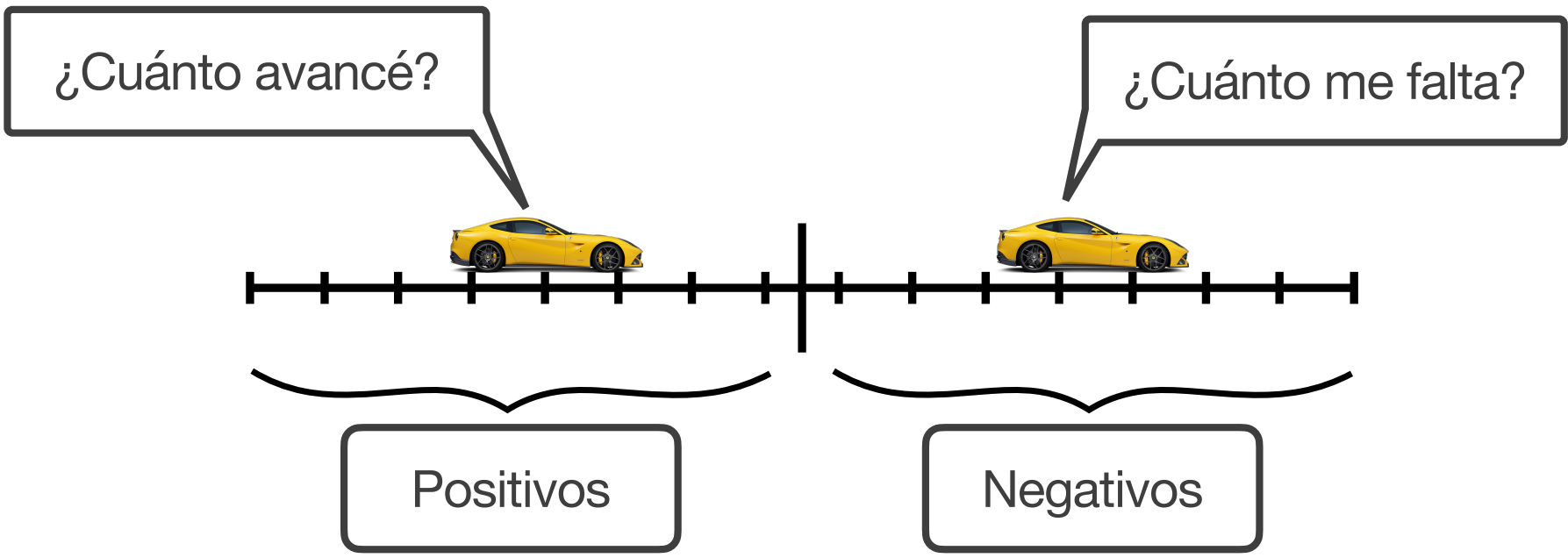
Representación de Números Enteros - Operaciones aritméticas

0	1	0	1
0	0	0	1
<hr/>			

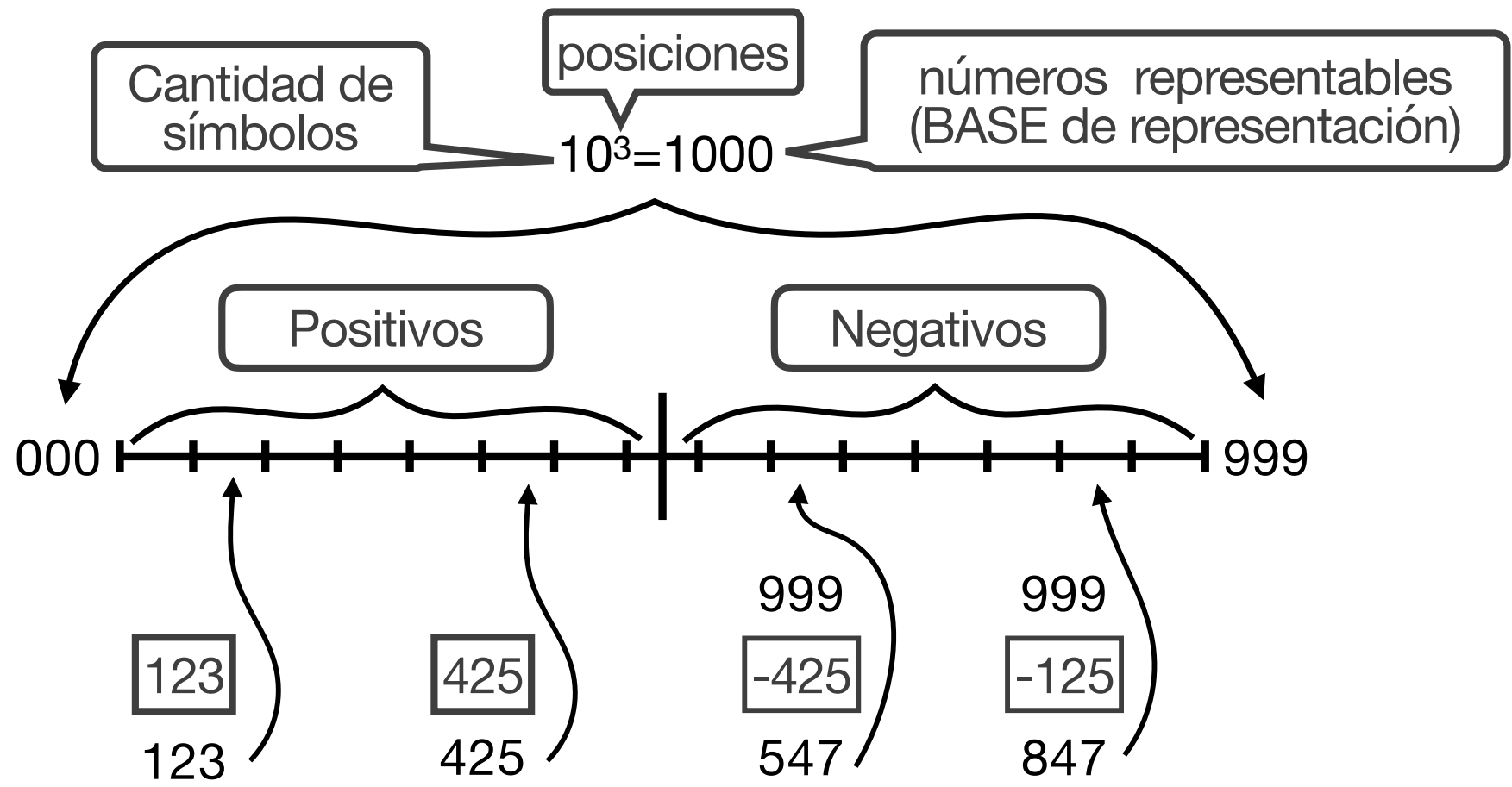
0	1	0	1
1	0	0	1
<hr/>			

1	1	0	1
1	0	0	1
<hr/>			

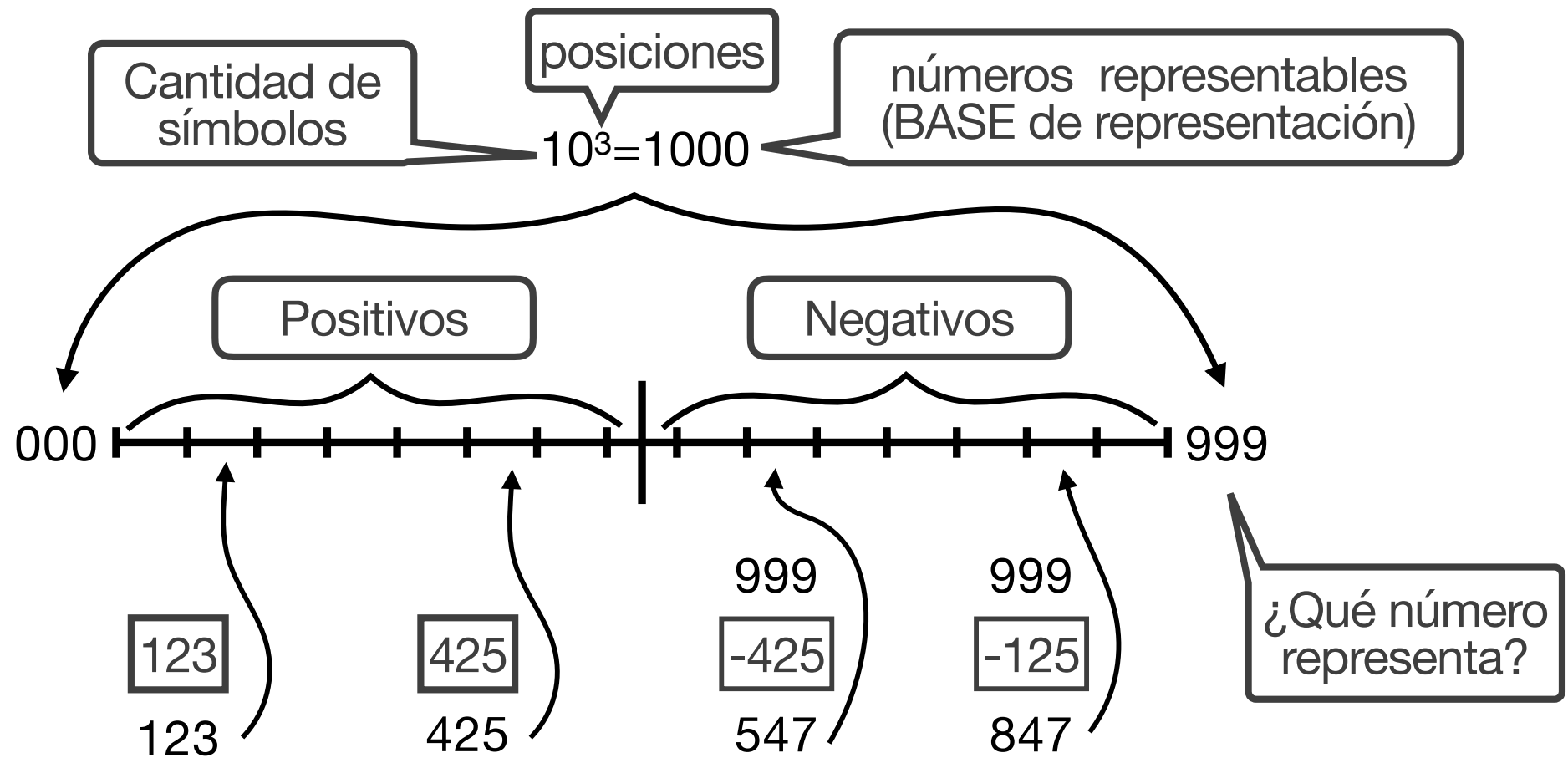
Representación de Números Enteros - Complemento a la Base



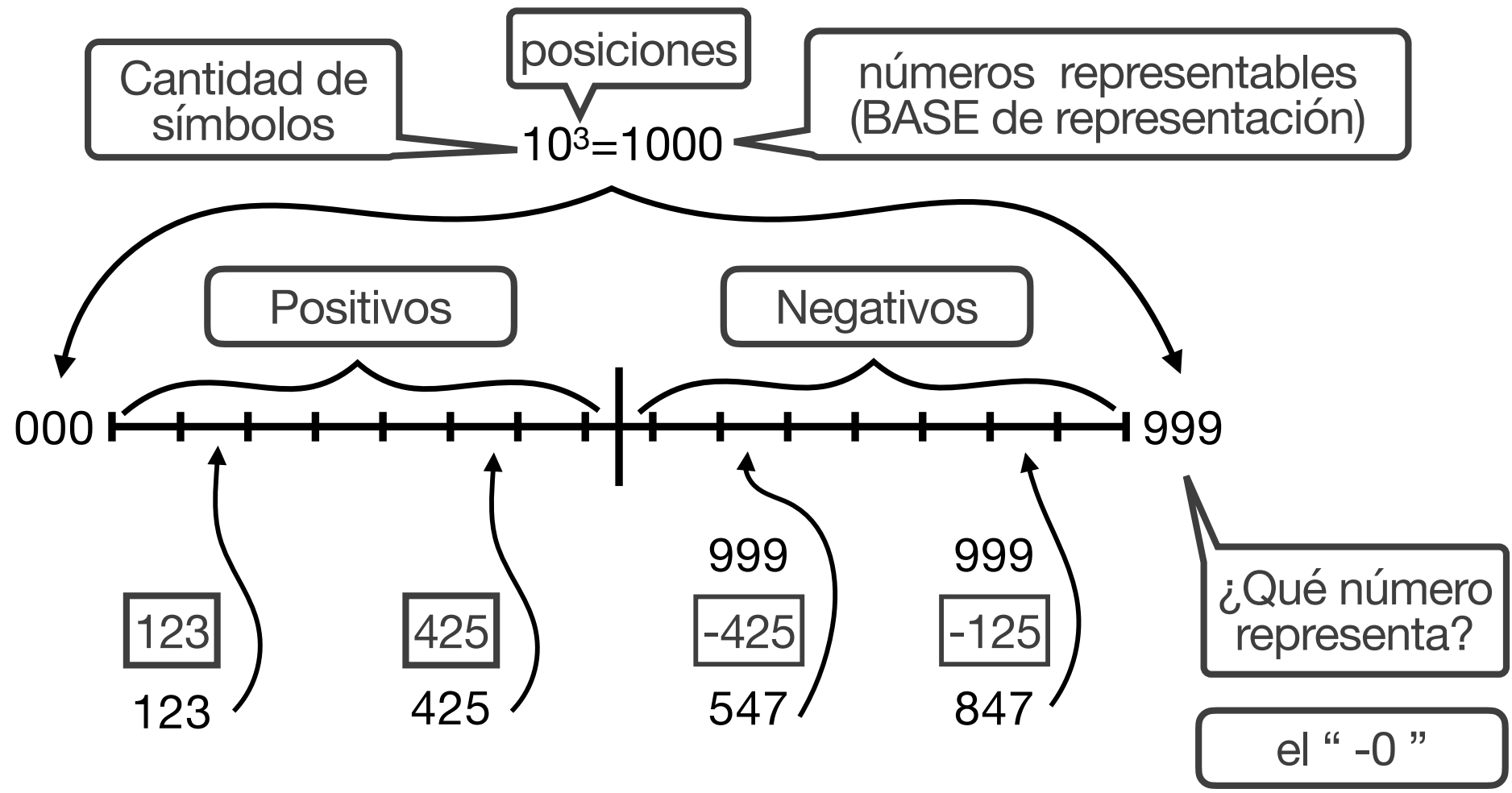
Representación de Números Enteros - Complemento a la Base (-1)



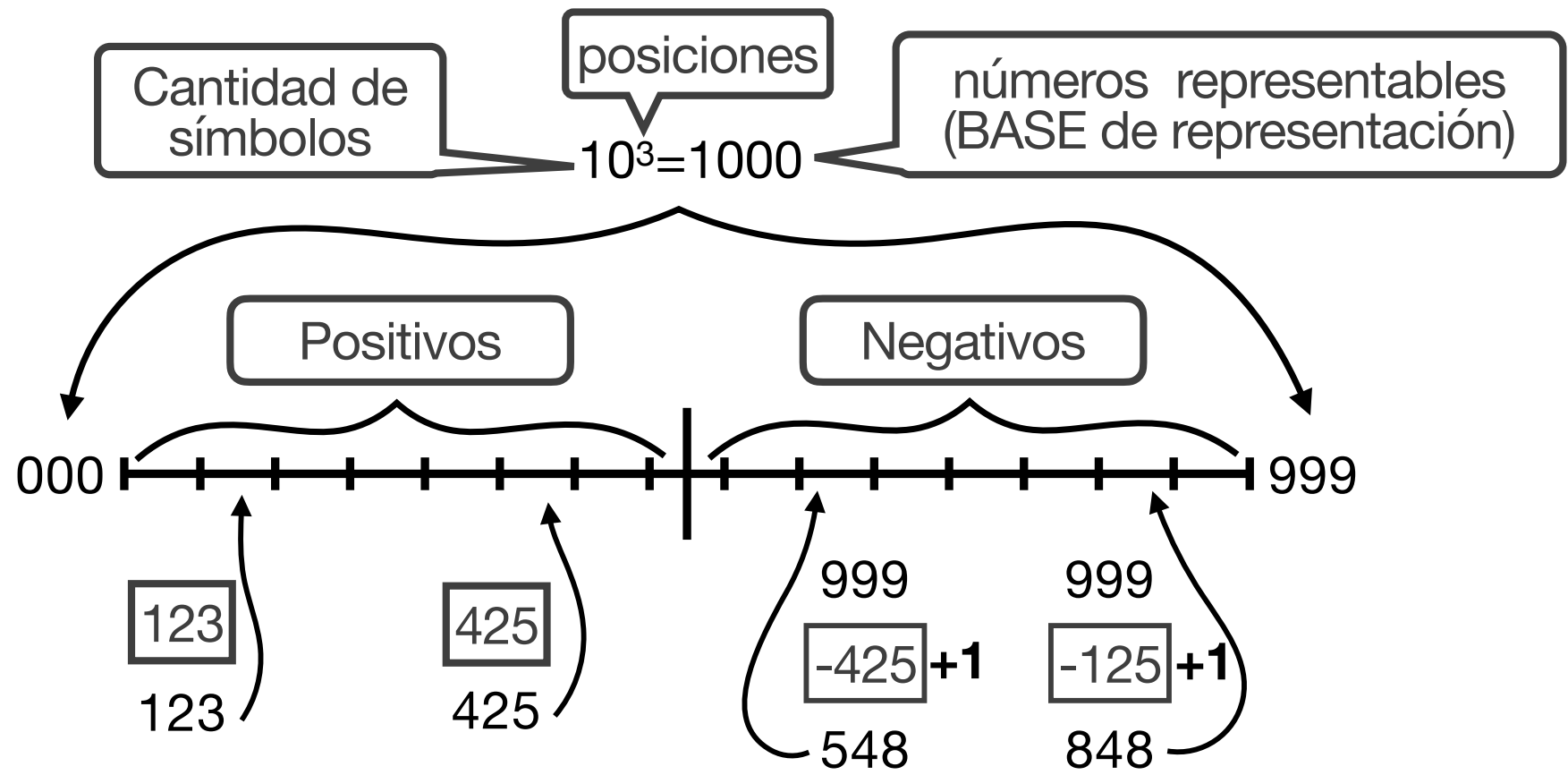
Representación de Números Enteros - Complemento a la Base (-1)



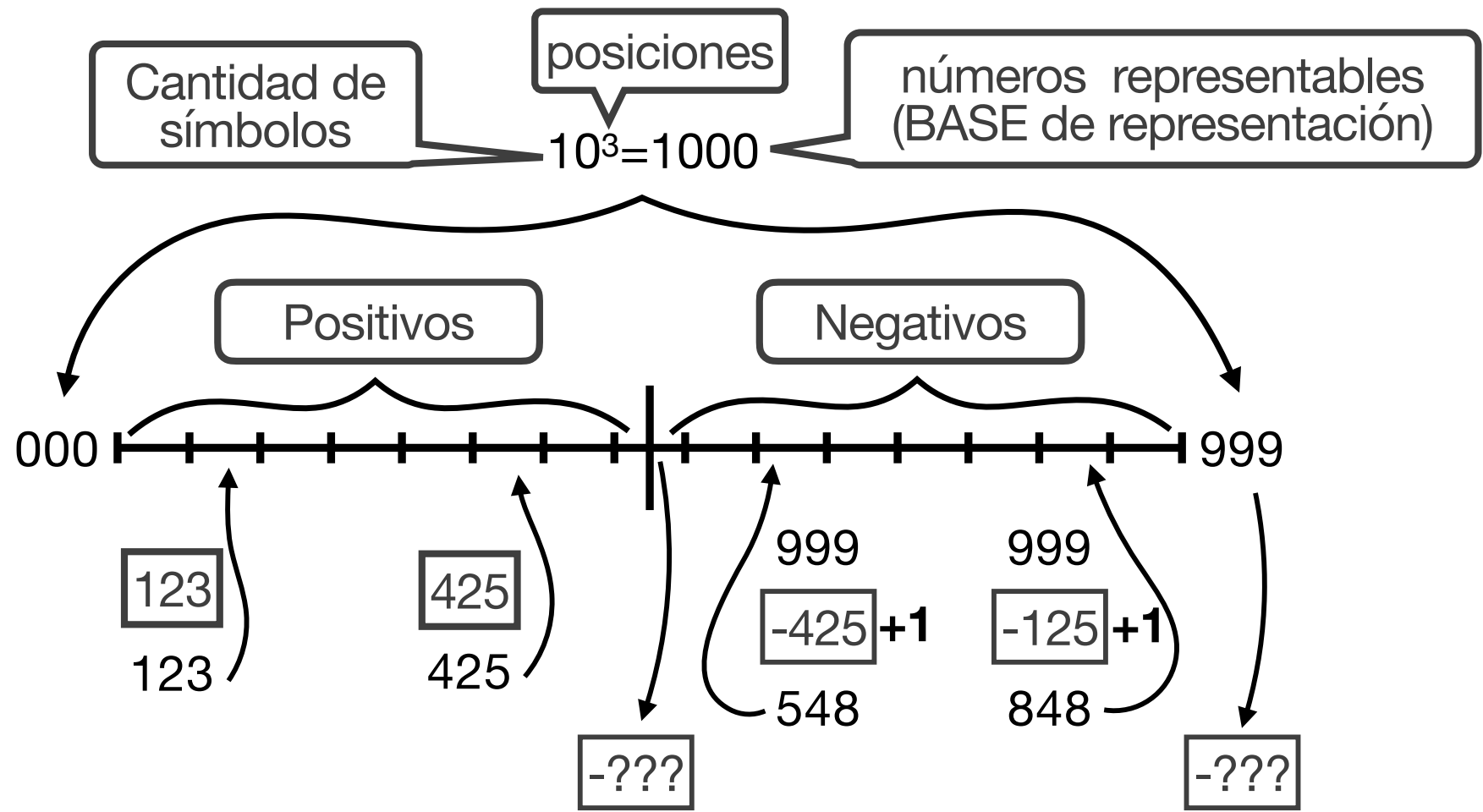
Representación de Números Enteros - Complemento a la Base (-1)



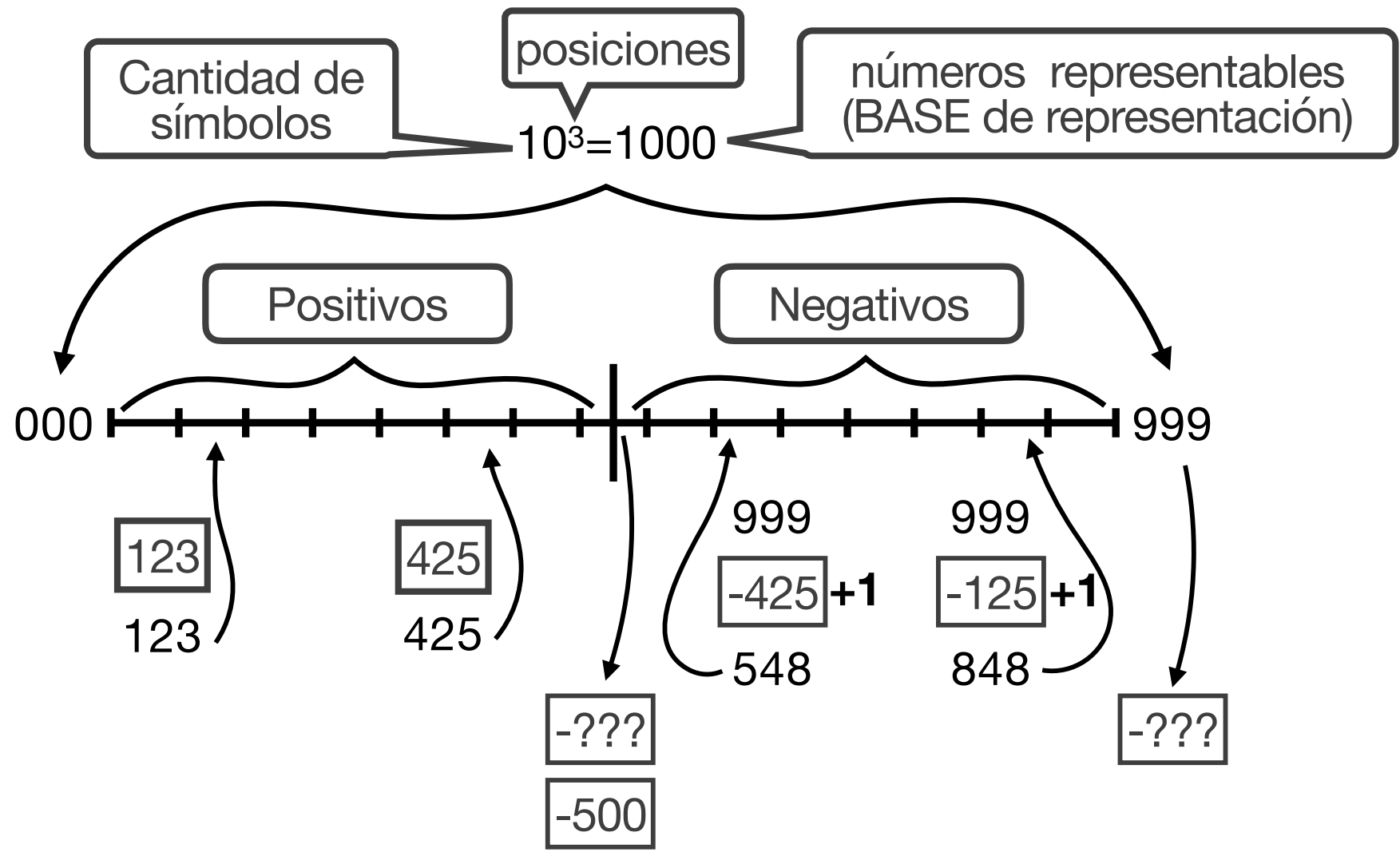
Representación de Números Enteros - Complemento a la Base



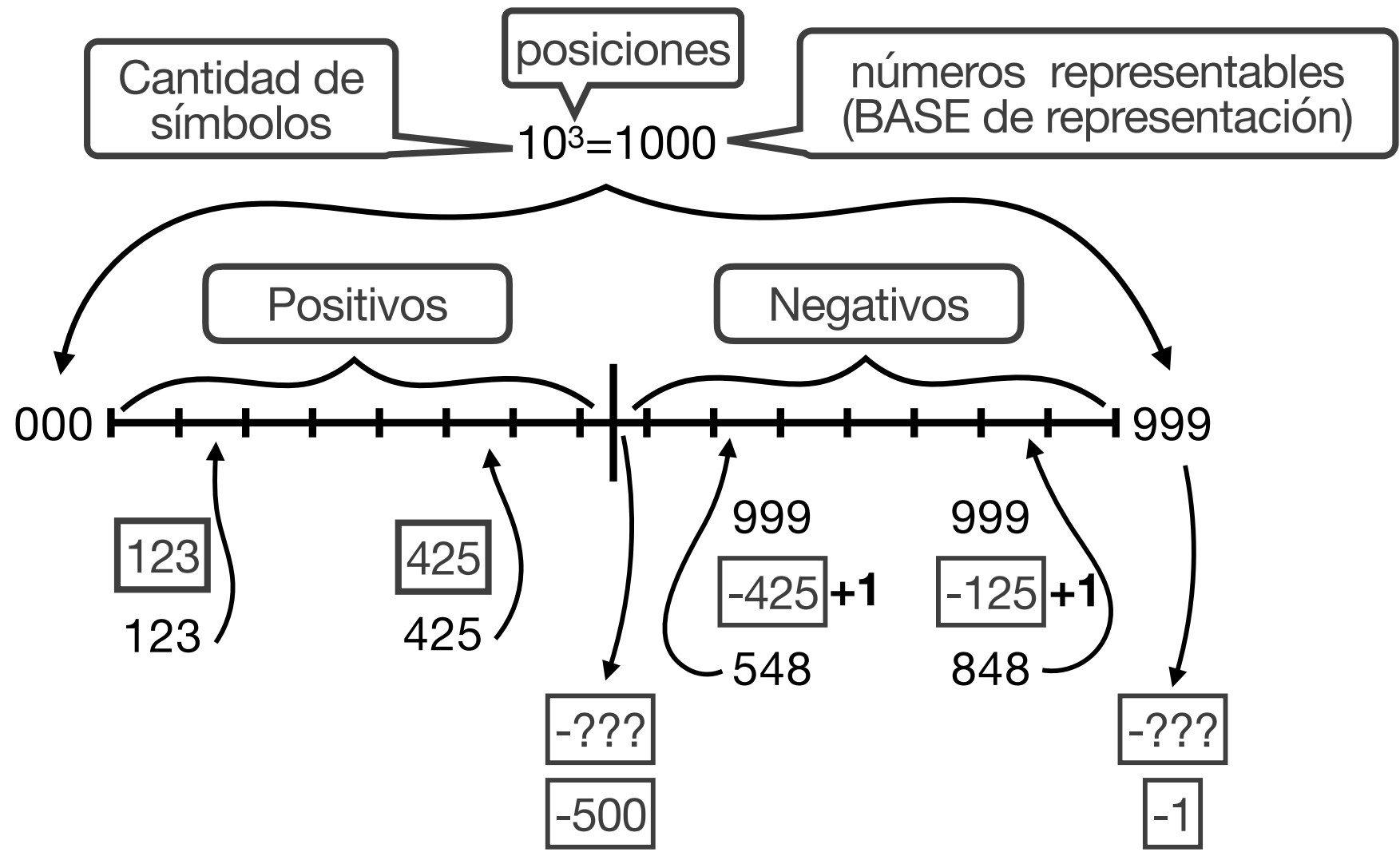
Representación de Números Enteros - Complemento a la Base



Representación de Números Enteros - Complemento a la Base



Representación de Números Enteros - Complemento a la Base



Representación de Números Enteros - Complemento a la Base

Operaciones

123

+

123

1	2	3
1	2	3



Representación de Números Enteros - Complemento a la Base

Operaciones

123

+

123

1	2	3
1	2	3
<hr/>		
2	4	6

Representación de Números Enteros - Complemento a la Base

Operaciones

$$\boxed{123} + \boxed{123}$$

1	2	3
---	---	---

1	2	3
---	---	---

2	4	6
---	---	---

$$\boxed{123} - \boxed{48} = \boxed{123} + \boxed{-48}$$

Representación de Números Enteros - Complemento a la Base

Operaciones

123 + 123

1	2	3
1	2	3
<hr/>		
2	4	6

123 - 48 = 123 + -48 + 1
+ 952

1	2	3
9	5	2
<hr/>		

Representación de Números Enteros - Complemento a la Base

Operaciones

123 + 123

1	2	3
1	2	3
<hr/>		
2	4	6

123 - 48 = 123 + -48⁹⁹⁹₊₁
+ 952

1	2	3
9	5	2
<hr/>		
0	7	5

Representación de Números Enteros - Complemento a la Base

Operaciones

123 + 123

1	2	3
1	2	3
<hr/>		
2	4	6

123 - 48 = 123 + -48 + 1
+ 952

1	2	3
9	5	2
<hr/>		
0	7	5

-123 + -123

Representación de Números Enteros - Complemento a la Base

Operaciones

123 + 123

1	2	3
---	---	---

1	2	3
---	---	---

2	4	6
---	---	---

123 - 48 = 123 + -48⁺¹
+ 952

1	2	3
---	---	---

9	5	2
---	---	---

0	7	5
---	---	---

-123⁺¹ + -123⁺¹
875 875

8	7	7
---	---	---

8	7	7
---	---	---

Representación de Números Enteros - Complemento a la Base

Operaciones

123

 +

123

1	2	3
1	2	3
<hr/>		
2	4	6

123

 -

48

 =

123

 +

-48

⁺¹_{+ 952}

1	2	3
9	5	2
<hr/>		
0	7	5

-123

⁺¹₈₇₅ +

-123

⁺¹₈₇₅

8	7	7
8	7	7
<hr/>		
7	4	4

Representación de Números Enteros - Complemento a la Base

Operaciones

123 + 123

123

123

123

123

123

123

246

123 - 48 = 123 + -48 + 1

123

-48

952

123

952

075

-123 + 1

-123 + 1

-123

-123

875

875

875

875

744 + 1

-256

Representación de Números Enteros - Complemento a la Base (a dos)

Cómo usar complemento a la base en sistema binario (ejemplo con 4 bits)

-6

Representación de Números Enteros - Complemento a la Base (a dos)

Cómo usar complemento a la base en sistema binario (ejemplo con 4 bits)

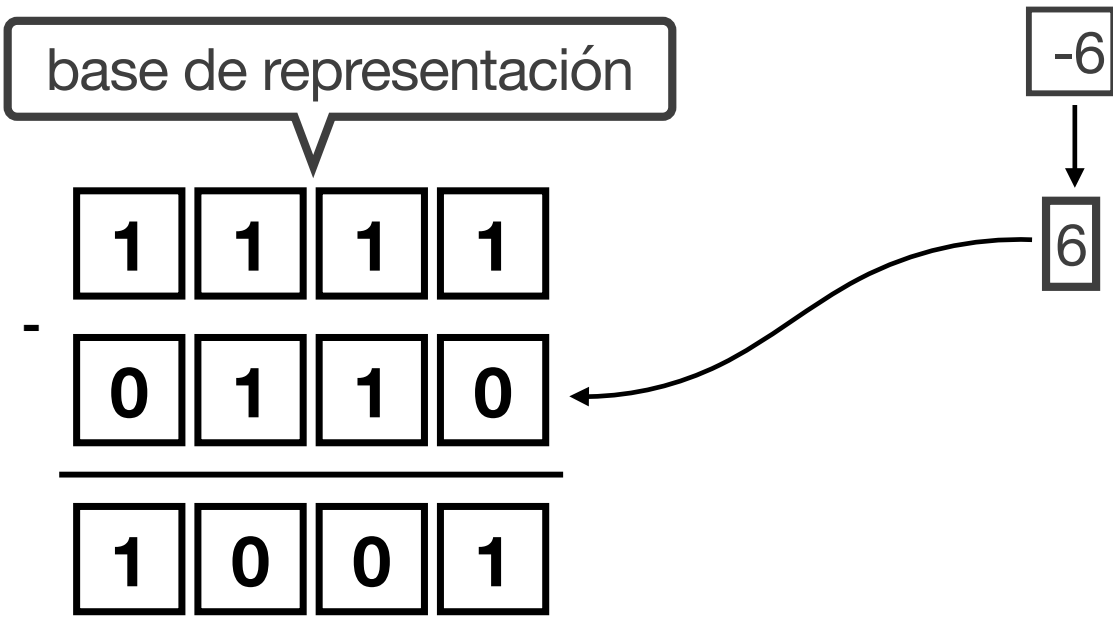
base de representación

-6



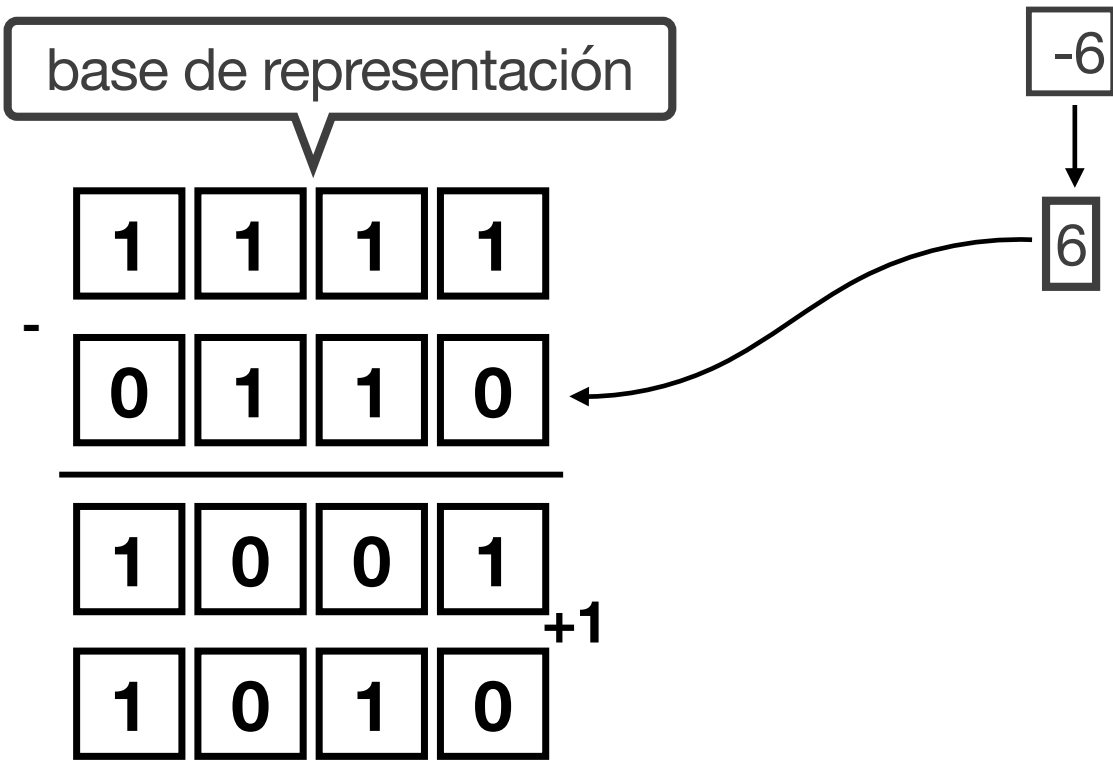
Representación de Números Enteros - Complemento a la Base (a dos)

Cómo usar complemento a la base en sistema binario (ejemplo con 4 bits)



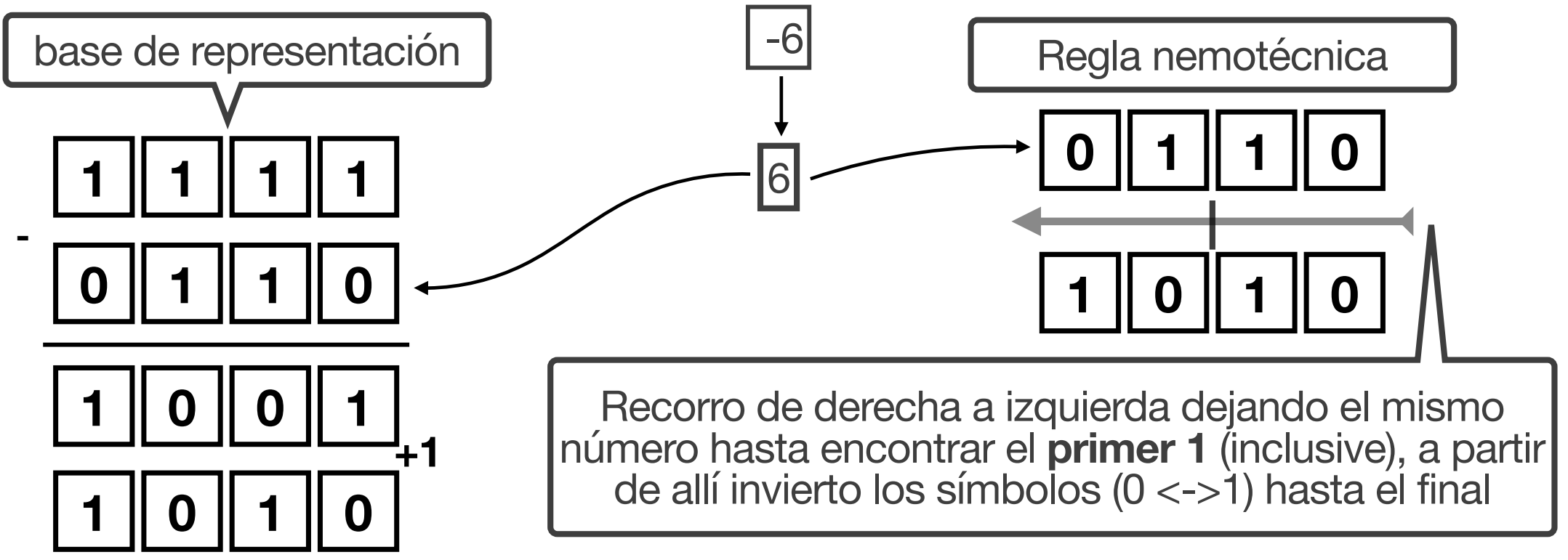
Representación de Números Enteros - Complemento a la Base (a dos)

Cómo usar complemento a la base en sistema binario (ejemplo con 4 bits)



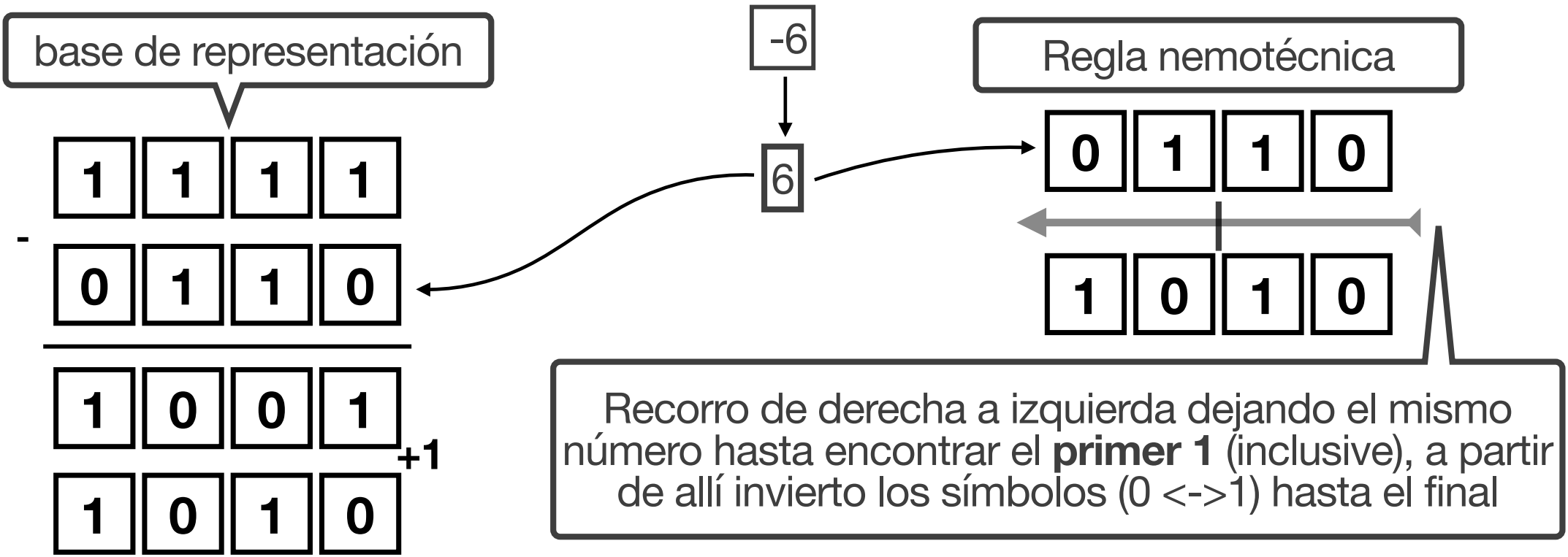
Representación de Números Enteros - Complemento a la Base (a dos)

Cómo usar complemento a la base en sistema binario (ejemplo con 4 bits)



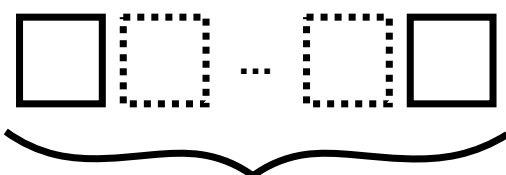
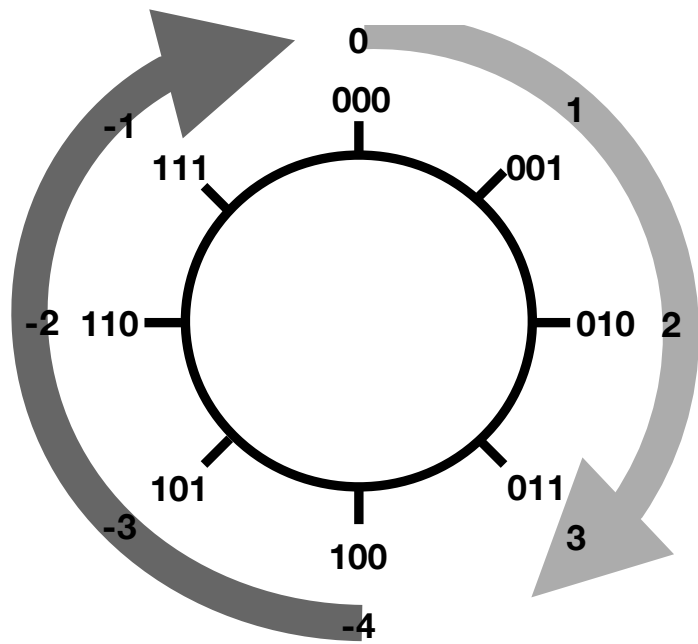
Representación de Números Enteros - Complemento a la Base (a dos)

Cómo usar complemento a la base en sistema binario (ejemplo con 4 bits)



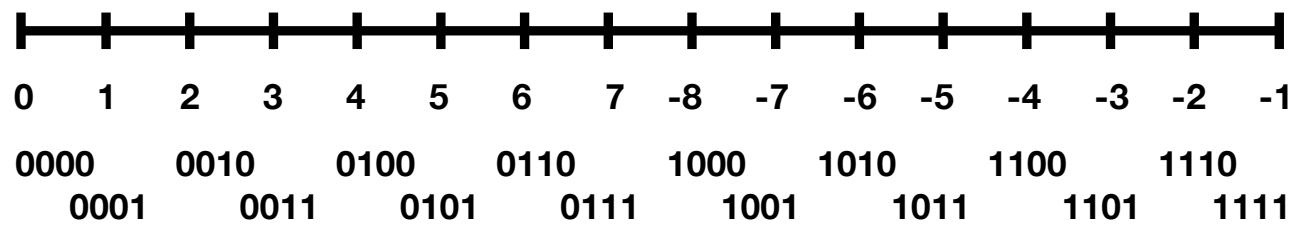
Dado un número ¿ ***cómo saber qué valor representa ?***
 Si comienza con “1” es un valor negativo, lo complemento y obtengo el valor

Representación de Números Enteros - Complemento a la Base (a dos)



Rango de Representación

$$\left[-(2^{n/2}) \text{ a } (2^{n/2}) - 1 \right]$$



Ejemplos		
# bits	desde	hasta
3	-4	3
5	-16	15
10	-512	511
16	-32768	32767
32	-2147483648	2147483647

Representación de Números Enteros - Operaciones: Multiplicación

123

*

27

1	2	3
0	2	7
<hr/>		

Representación de Números Enteros - Operaciones: Multiplicación

123

 *

27

Doble de tamaño

1	2	3
0	2	7

--	--	--	--	--	--

Representación de Números Enteros - Operaciones: Multiplicación

123

*

27

1

2

3

0

2

7

8

6

1

Representación de Números Enteros - Operaciones: Multiplicación

123

*

27

			1	2	3
			0	2	7
			8	6	1
		2	4	6	

Representación de Números Enteros - Operaciones: Multiplicación

123

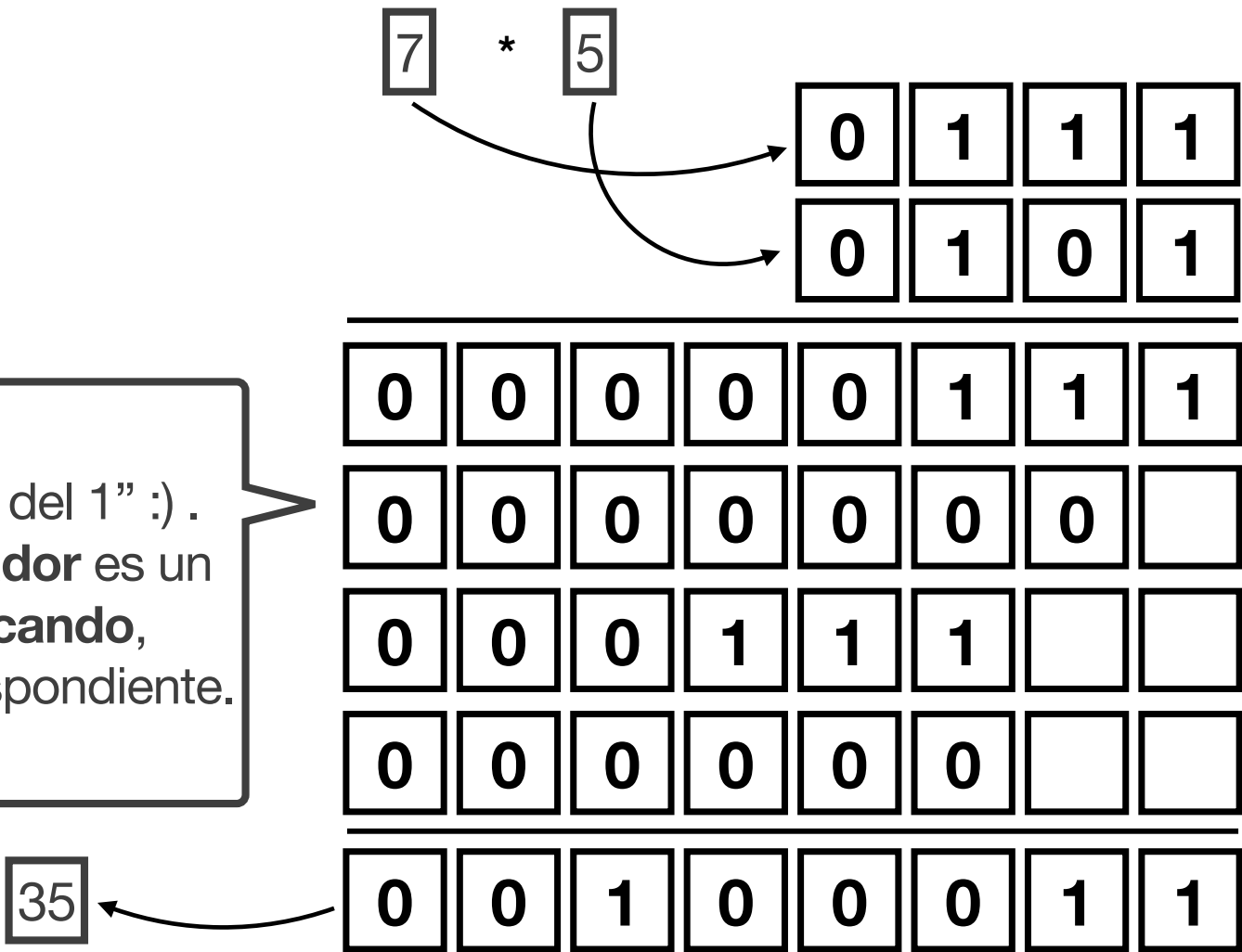
*

27

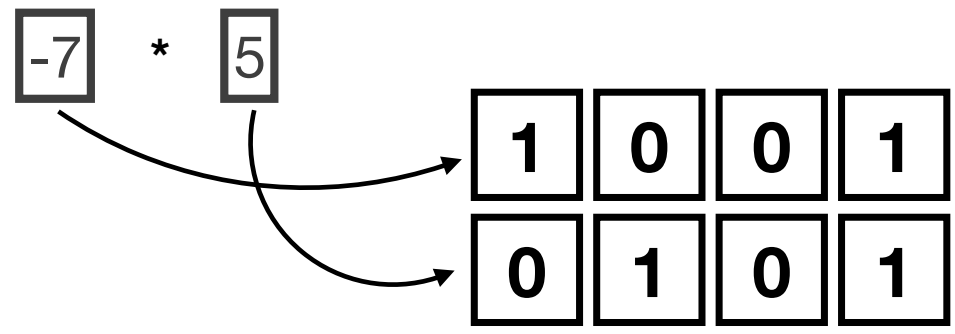
			1	2	3
			0	2	7
			8	6	1
		2	4	6	
		3	3	2	1

Representación de Números Enteros - Operaciones: Multiplicación

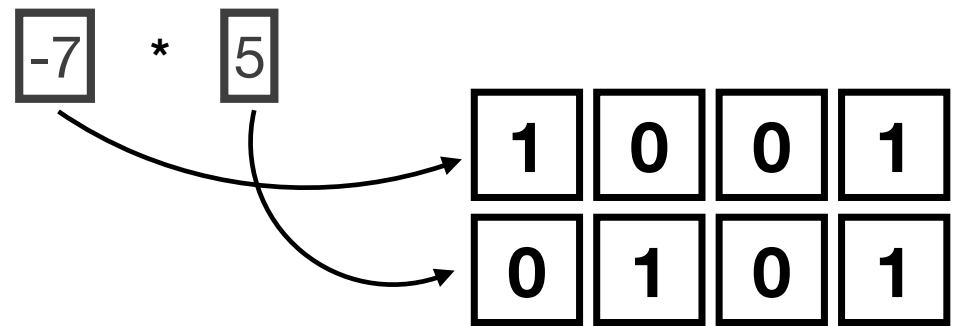
Sólo tenemos “tabla de 0 y del 1” :) .
Si la posición del **multiplicador** es un
“1” sumamos el **multiplicando**,
“corrido” a la posición correspondiente.



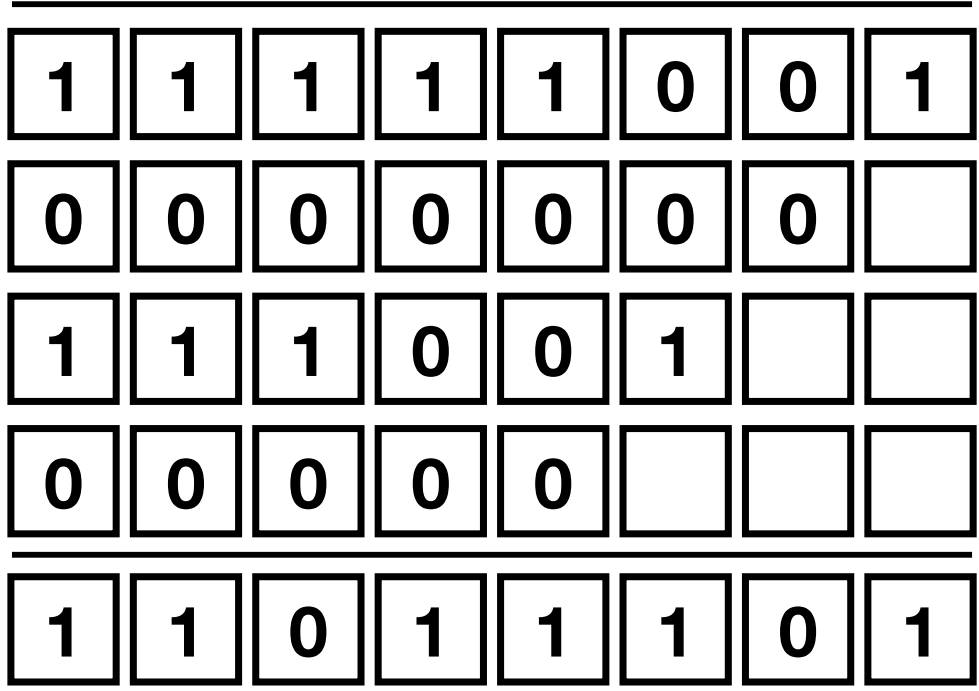
Representación de Números Enteros - Operaciones: Multiplicación



Representación de Números Enteros - Operaciones: Multiplicación



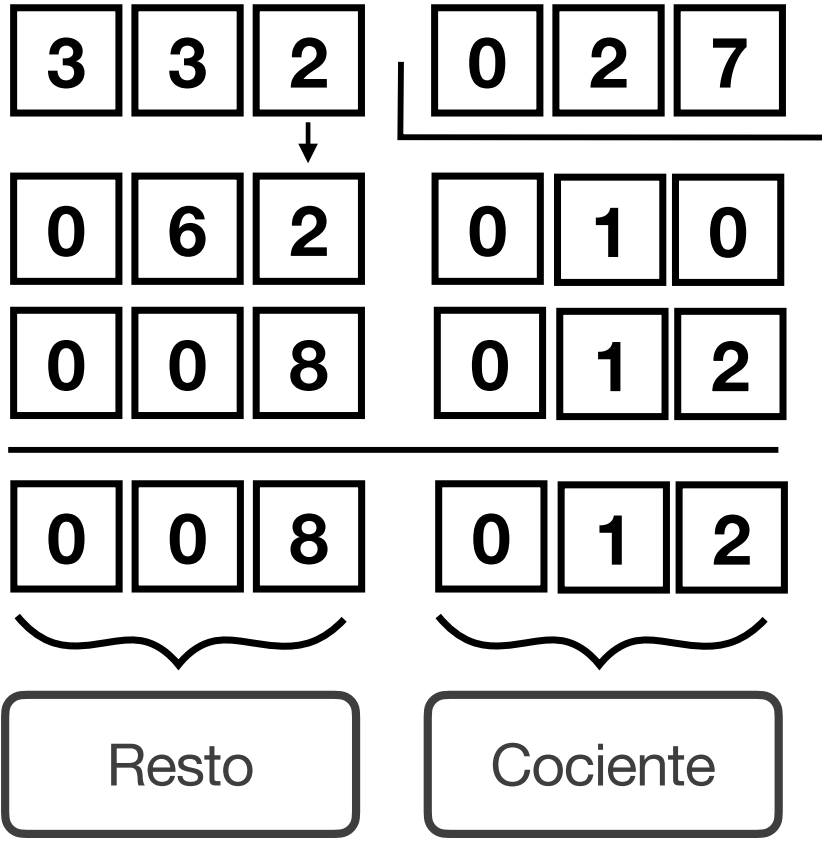
Como las restas son sumas, utilizamos el mismo procedimiento, pero debemos **extender el signo**.



-35

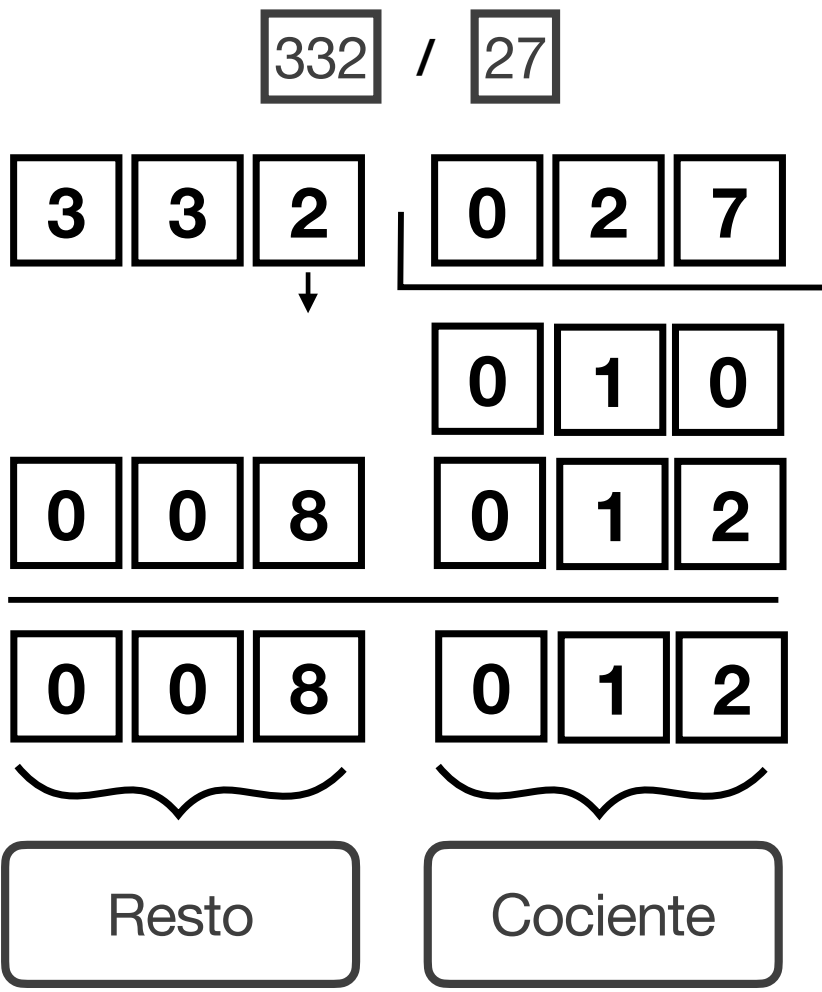
Representación de Números Enteros - Operaciones: División

332 / 27



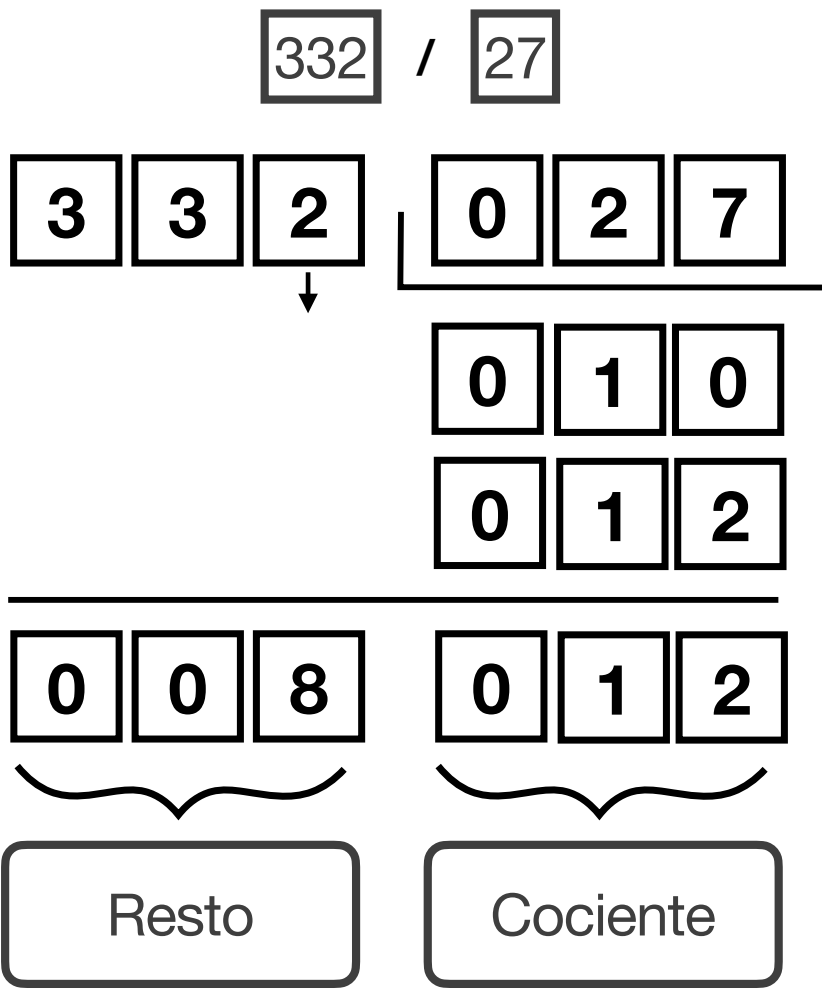
Máximas restas sucesivas.

Representación de Números Enteros - Operaciones: División



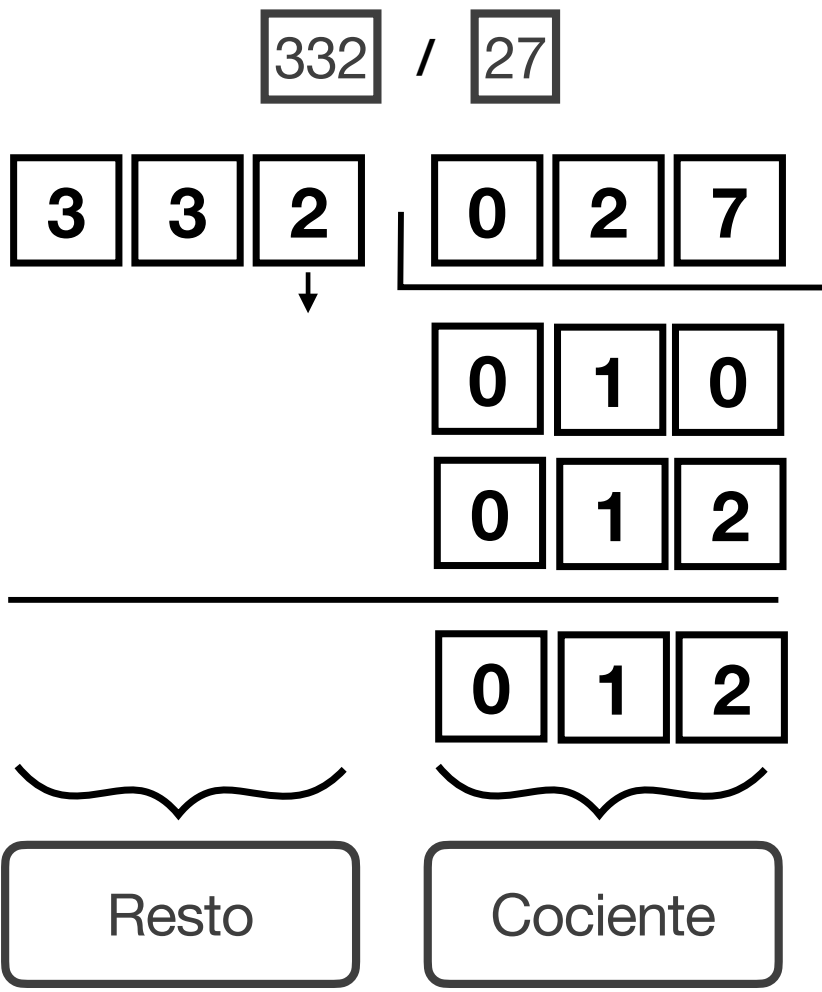
Máximas restas sucesivas.

Representación de Números Enteros - Operaciones: División



Máximas restas sucesivas.

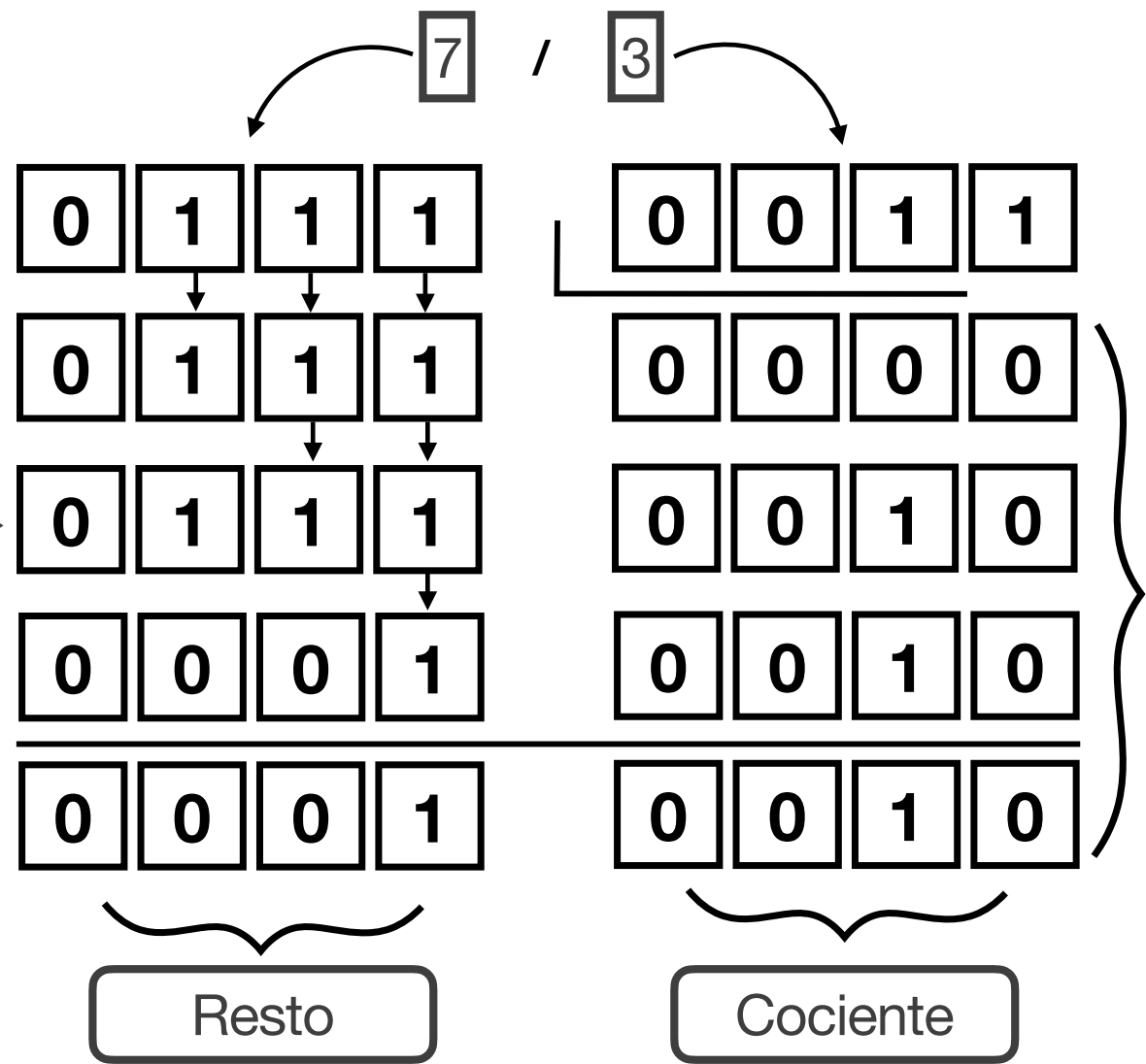
Representación de Números Enteros - Operaciones: División



Máximas restas sucesivas.

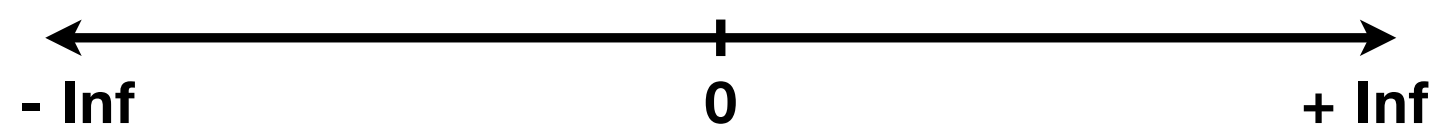
Representación de Números Enteros - Operaciones: División

Restas sucesivas, como sólo tenemos “tablas del 0 y 1”, o puedo restarlo una vez o no puedo restarlo :) .

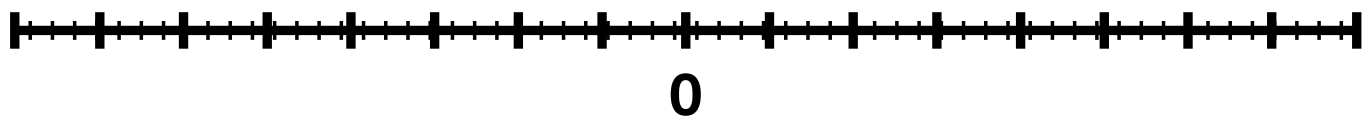


repito tantas veces como cantidad de posiciones

Representación de Números Racionales



Representación de Números Racionales



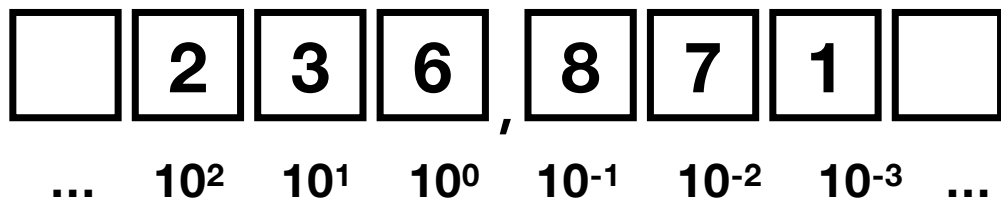
Representación de Números Racionales

¿ cómo podemos representar el 236,871 ?

--	--	--	--	--	--	--	--

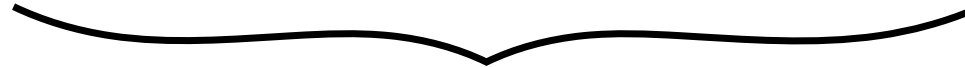
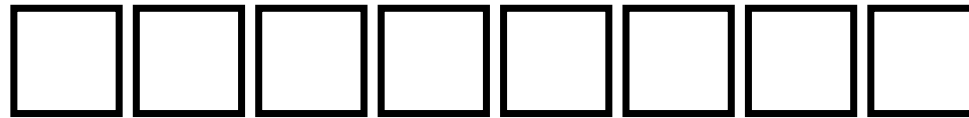
Representación de Números Racionales

¿ cómo podemos representar el 236,871 ?



Representación de Números Racionales

¿ cómo podemos representar el 5,82 ?

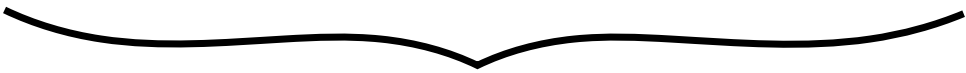


cantidad de bits

Representación de Números Racionales (Punto fijo)

¿ cómo podemos representar el 5,84 ?

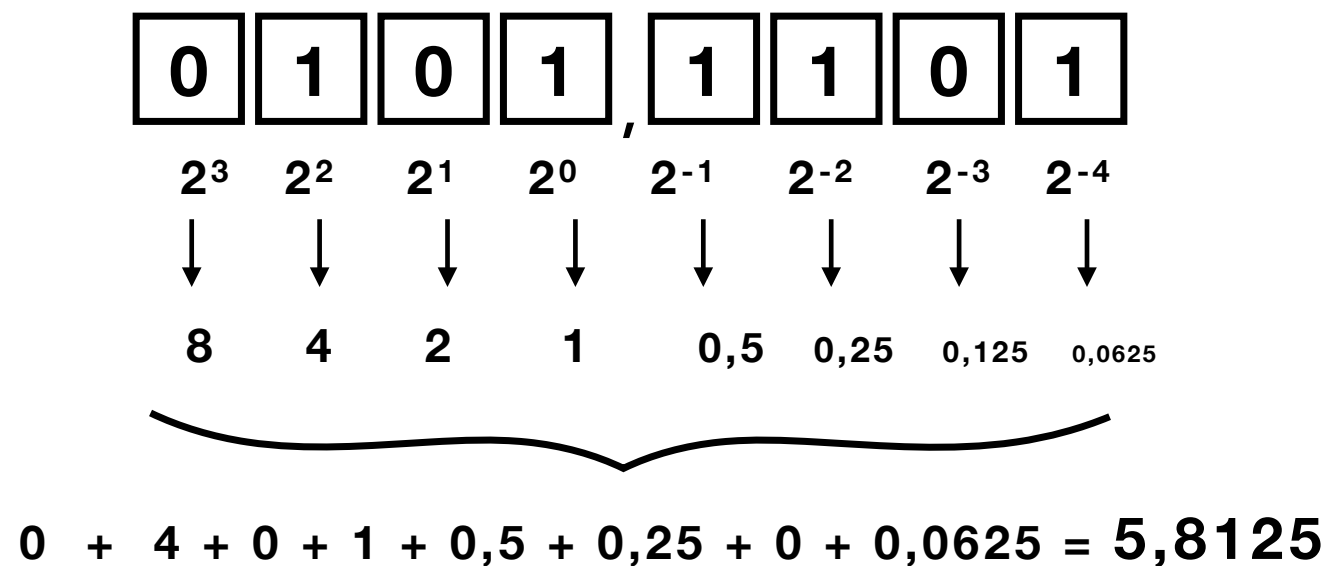
	↓	↓	↓	↓	↓	↓	↓	
	8	4	2	1	0,5	0,25	0,125	0,0625



$$0 + 4 + 0 + 1 + 0,5 + 0,25 + 0 + 0,0625 = 5,8125$$

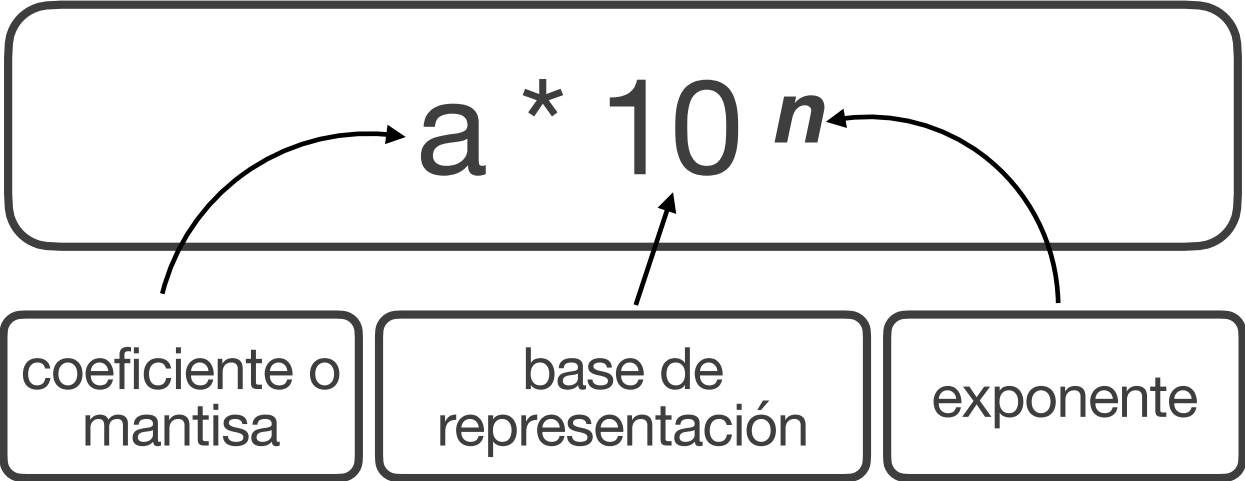
Representación de Números Racionales (Punto fijo)

¿ cómo podemos representar el 5,84 ?



Representación de Números Racionales con notación científica

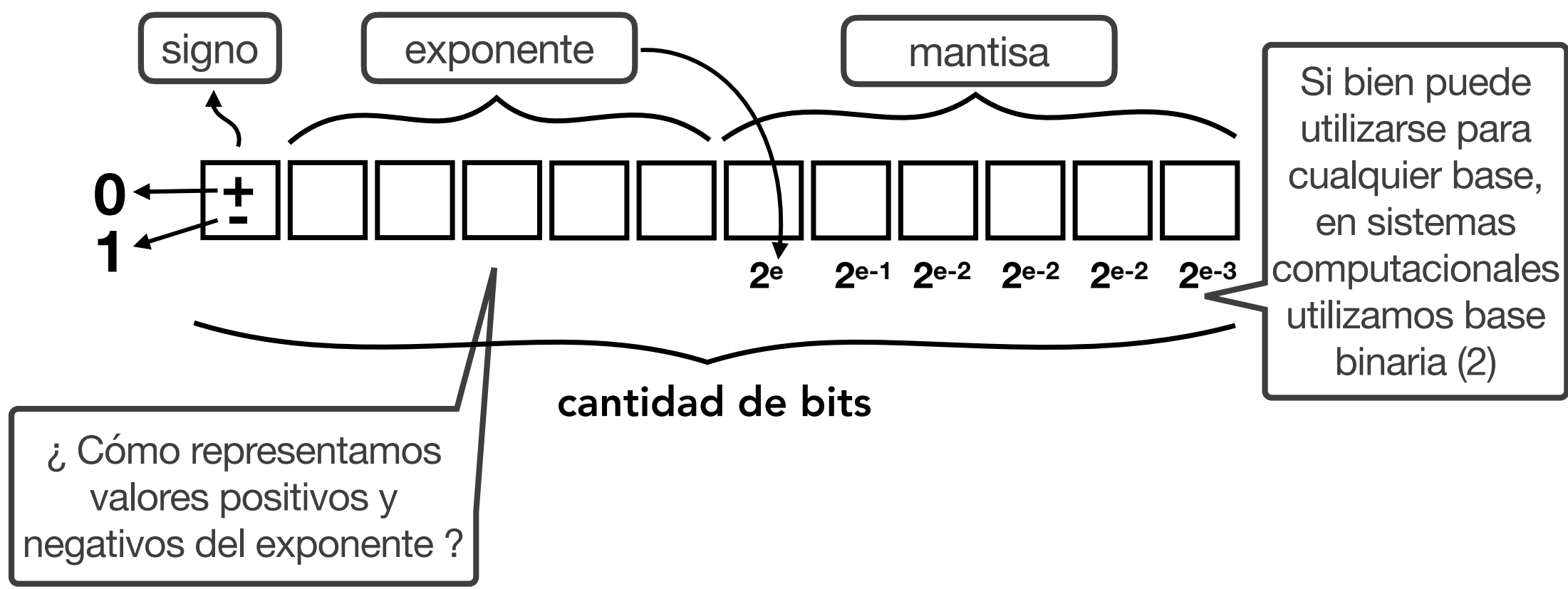
La notación científica es una forma estandarizada de expresar números muy grandes o muy pequeños de una manera más compacta y fácil de manejar.



Ejemplos	
236,871	$2,36871 * 10^2$
0,000025	$2,5 * 10^{-5}$
6.720.000.000	$6,72 * 10^9$
0.0000000000000045	$4,5 * 10^{-14}$

Representación de Números Racionales con punto flotante

La representación con **punto flotante** permite expresar números tanto enteros como fraccionarios, incluyendo valores extremadamente grandes o pequeños, de una manera eficiente y precisa.



Representación de Números Racionales con punto flotante - convenciones

Normalización: es un proceso fundamental para garantizar una representación eficiente y precisa de los números reales. Ya que podríamos tener en algunos casos más de una representación para un mismo número.

Extensiones: Además, la representación con punto flotante provee la representación de **infinito** (positivo y negativo). Para ello se destina una configuración particular del exponente (cuando es todos 1). Cuando el número no es posible representarlo con la cantidad de bits, pasa a ser **Infinito**. Operaciones con infinito: infinito + ó - número = infinito.

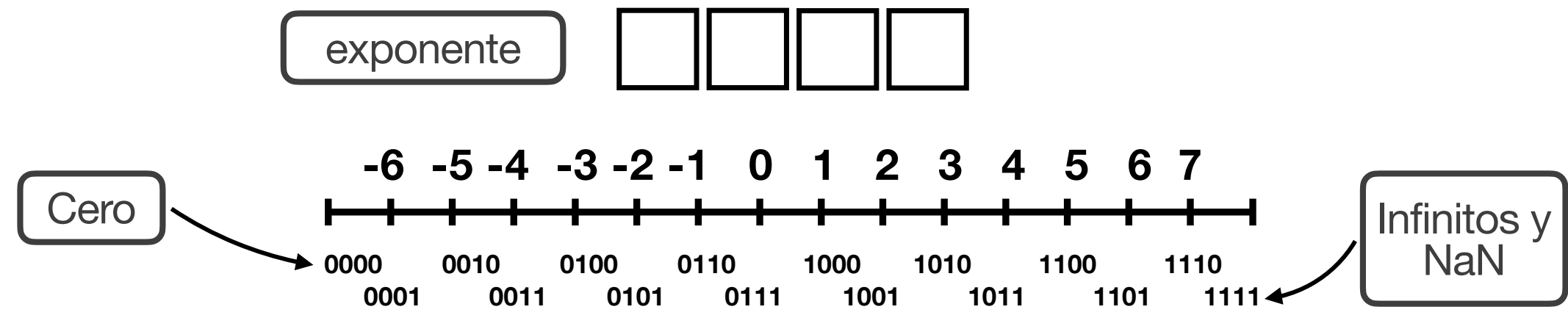
¿Cuál sería el resultado de infinito - infinito?

Representación de Números Racionales con punto flotante - convenciones

Casos especiales de representación			
Número	Bit Signo	Exponente	Mantiza
+ infinito	0	todos 1	todos 0
- infinito	1	todos 1	todos 0
cero	0	todos 0	todos 0
No es un Número (NaN)	0	todos 1	al menos un 1

Representación de Números Racionales con punto flotante - Polarización

La **polarización** es una forma de representar números positivos y negativos. En la representación con polarización, se suma un valor fijo (llamado **sesgo** o **bias**) al exponente real para obtener el valor final del exponente que se almacena en el formato de punto flotante. El objetivo de este enfoque es permitir que el exponente sea almacenado de manera eficiente y se pueda utilizar tanto para números positivos como negativos sin tener que utilizar el sistema complemento a la base.



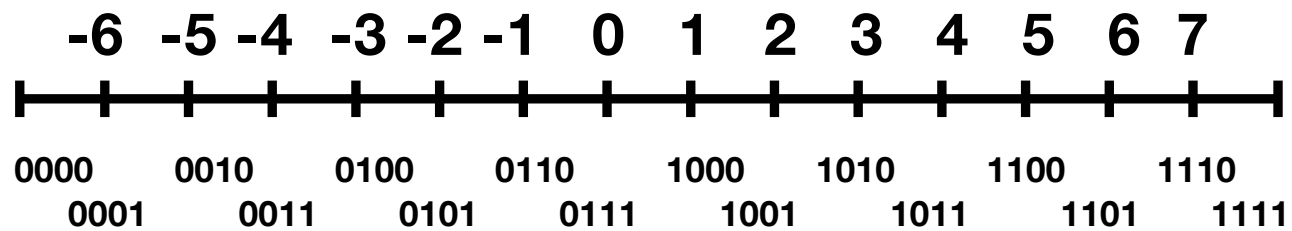
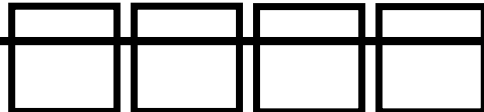
¿Cual es el polarizante (sesgo o bias)? y ¿Cómo calculamos un exponente?

Representación de Números Racionales con punto flotante - Polarización

Complemento a la Base

La **polarización** es una forma de representar números positivos y negativos. En la representación con polarización, se suma un valor fijo (llamado **sesgo** o **bias**) al exponente real para obtener el valor final del exponente que se almacena en el formato de punto flotante. El objetivo de este enfoque es permitir que el exponente sea almacenado de manera eficiente y se pueda utilizar tanto para números positivos como negativos sin tener que utilizar el sistema complemento a la base.

exponente



Cero

Infinitos y NaN

¿Cual es el polarizante (sesgo o bias)? y ¿Cómo calculamos un exponente?

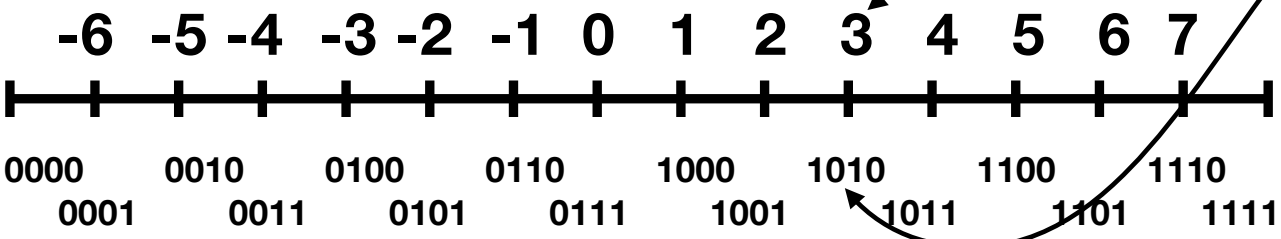
Representación de Números Racionales con punto flotante - Polarización

Polarizante: se calcula en base a la cantidad total de la representación dividido 2 menos 1.

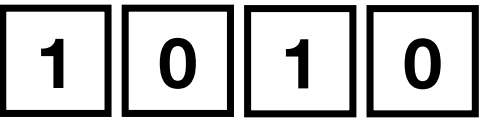
Ejemplos con sistema binario		
4 bits	$(2^4/2)-1$	7
8 bits	$(2^{12}/2)-1$	127
11 bits	$(2^{12}/2)-1$	1023

Ejemplo: si necesitamos indicar que el exponente es **3** en una representación con **4** bits, debemos sumarle **7**.

$3 + 7 = 10 = 1010$

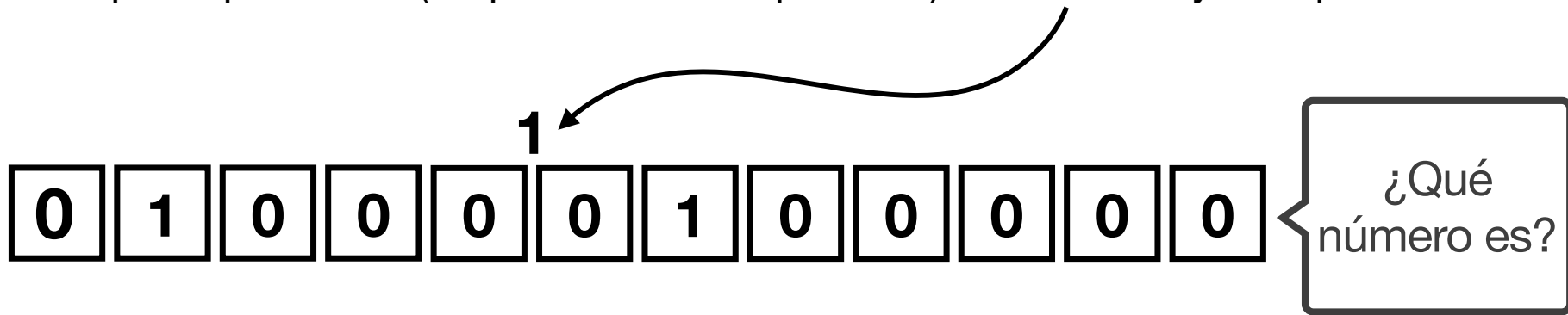


exponente



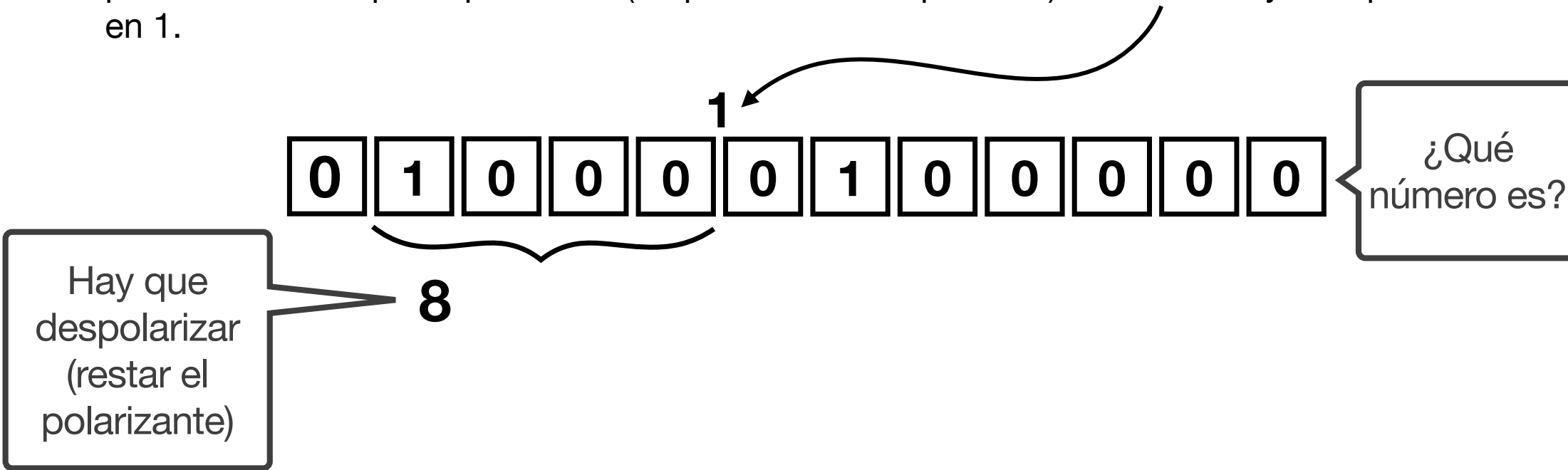
Representación de Números Racionales con punto flotante - Bit Oculto

Dado que todos los números a representar tienen **al menos un 1** en la mantisa. Se toma por convención que el primer bit (al que refiere el exponente) **está oculto** y siempre está en 1.



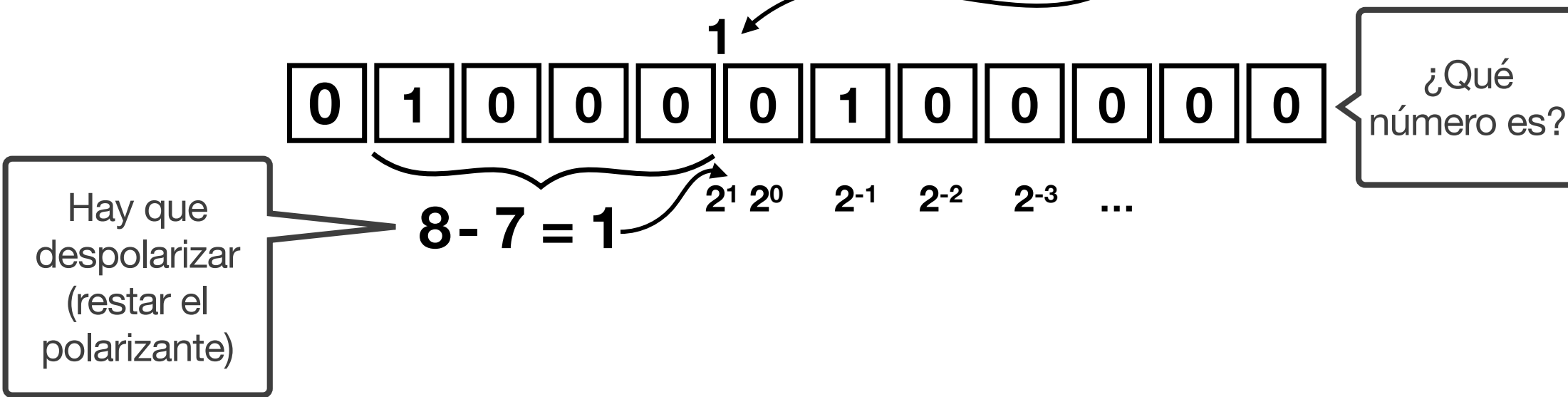
Representación de Números Racionales con punto flotante - Bit Oculto

Dado que todos los números a representar tienen **al menos un 1** en la mantisa. Se toma por convención que el primer bit (al que refiere el exponente) **está oculto** y siempre está en 1.



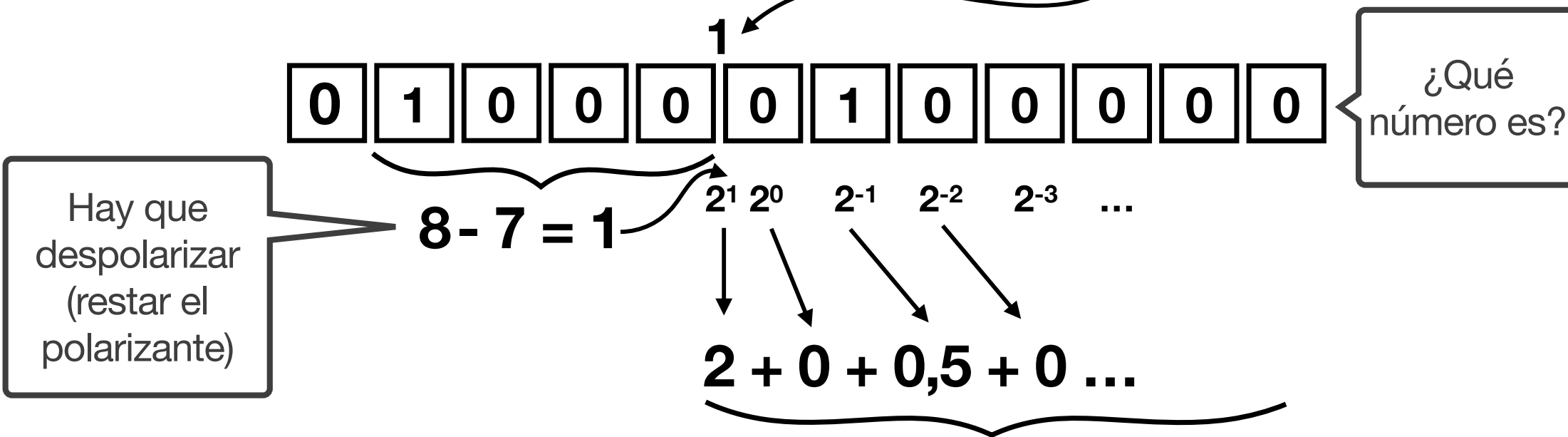
Representación de Números Racionales con punto flotante - Bit Oculto

Dado que todos los números a representar tienen **al menos un 1** en la mantisa. Se toma por convención que el primer bit (al que refiere el exponente) **está oculto** y siempre está en 1.



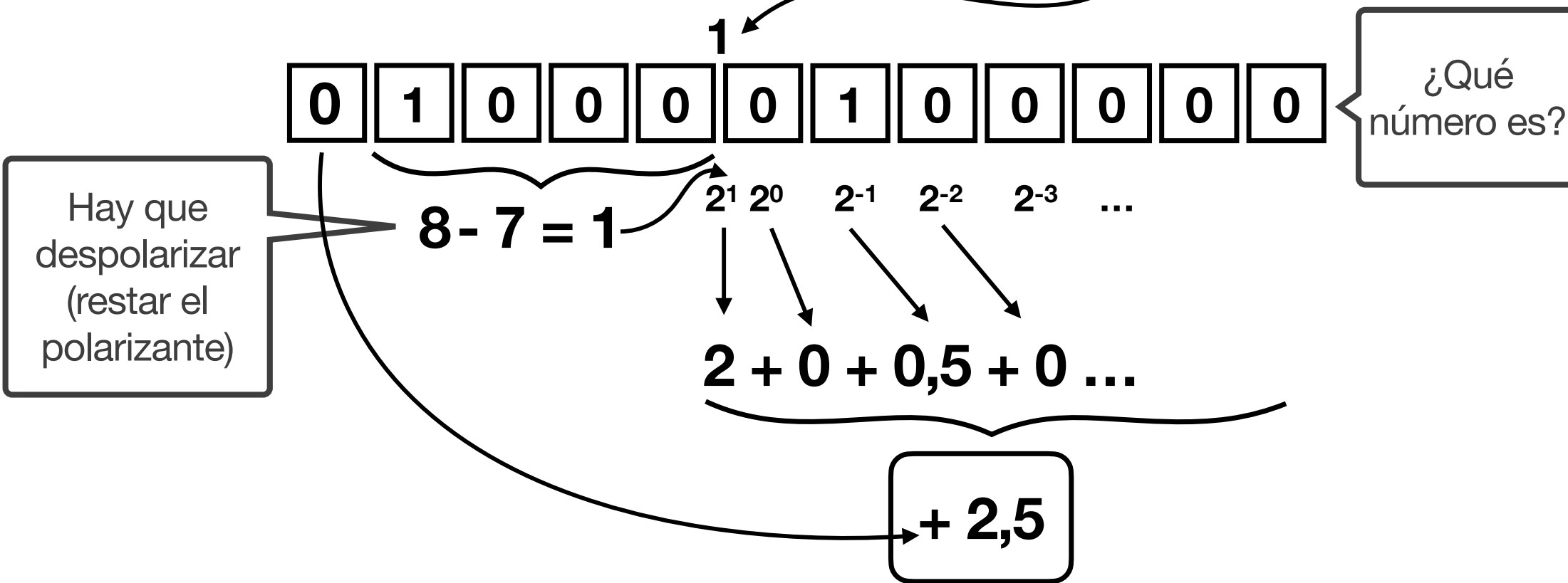
Representación de Números Racionales con punto flotante - Bit Oculto

Dado que todos los números a representar tienen **al menos un 1** en la mantisa. Se toma por convención que el primer bit (al que refiere el exponente) **está oculto** y siempre está en 1.



Representación de Números Racionales con punto flotante - Bit Oculto

Dado que todos los números a representar tienen **al menos un 1** en la mantisa. Se toma por convención que el primer bit (al que refiere el exponente) **está oculto** y siempre está en 1.



Representación de Números Racionales con punto flotante - Estándares

Estándares (IEEE 1985)

Single: 32bits

- 1 bit Signo
- 8 bits Exponente.
- 23 bits Mantisa

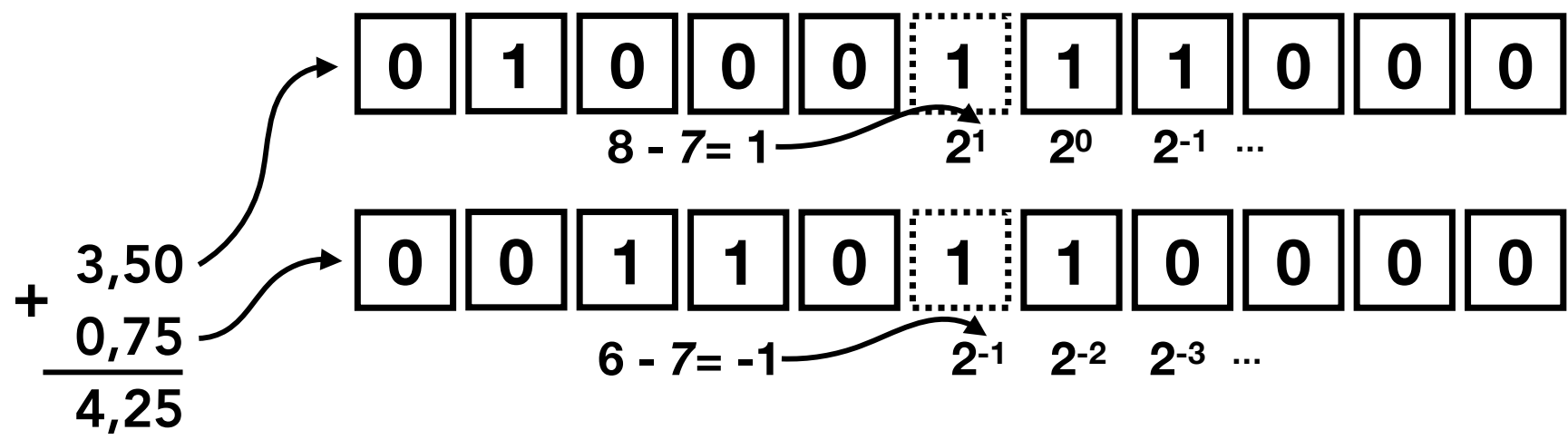
Double: 64bits

- 1 bit Signo
- 11 bits Exponente.
- 52 bits Mantisa

Representación de Números Racionales con punto flotante - Operaciones

Algoritmo de Suma/Resta:

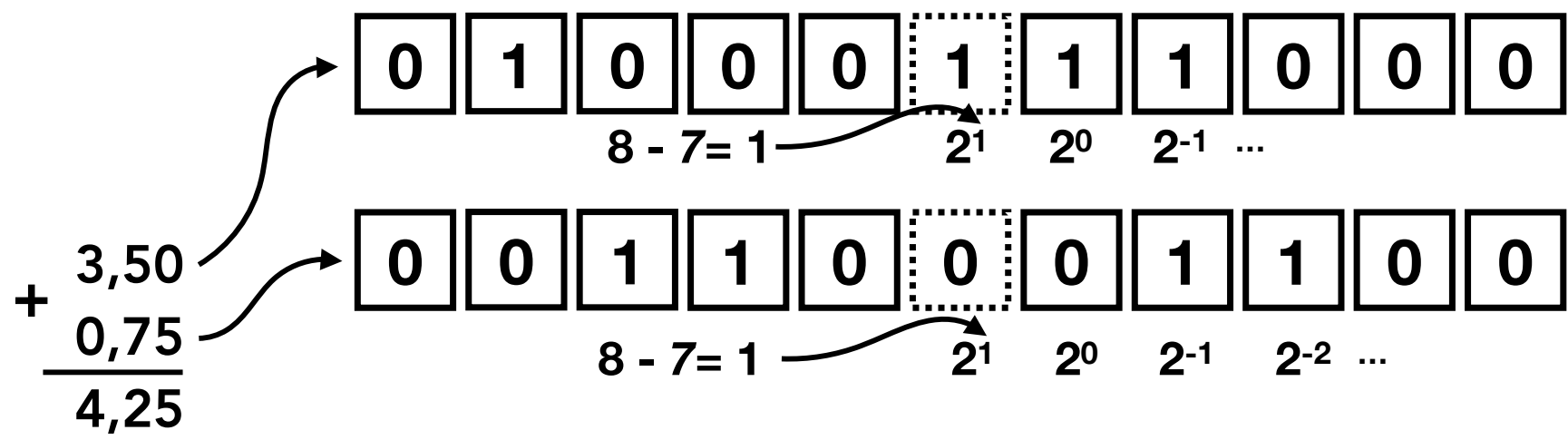
- 1) Elegir el operando con **menor exponente**, *alienar* la mantisa hacia la derecha incrementando el exponente hasta que éste **coincida** con el del otro operando. (Recordar el bit oculto)
- 2) Operar con las mantisas como la **suma/resta** de enteros.
- 3) Modificar el **signo** si fuese necesario.
- 4) **Normalizar** si fuese necesario.



Representación de Números Racionales con punto flotante - Operaciones

Algoritmo de Suma/Resta:

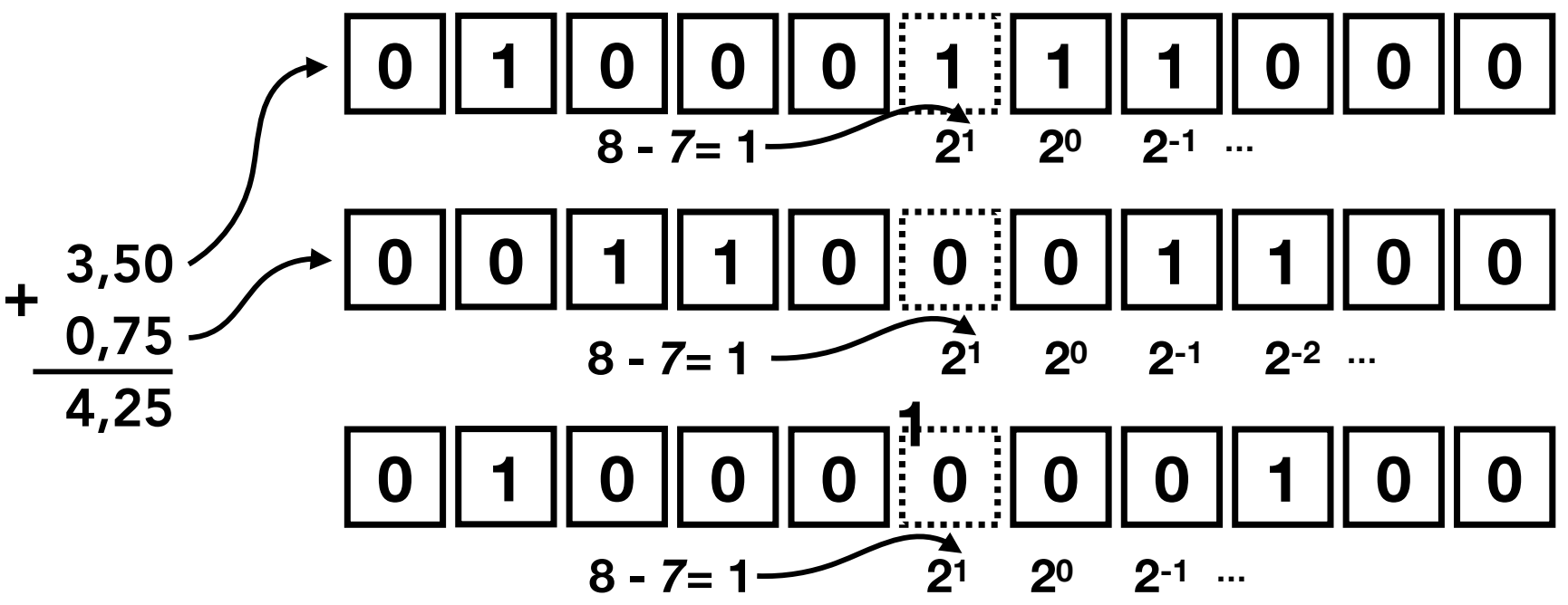
- 1) Elegir el operando con **menor exponente**, *alienar* la mantisa hacia la derecha incrementando el exponente hasta que éste **coincida** con el del otro operando. (Recordar el bit oculto)
- 2) Operar con las mantisas como la **suma/resta** de enteros.
- 3) Modificar el **signo** si fuese necesario.
- 4) **Normalizar** si fuese necesario.



Representación de Números Racionales con punto flotante - Operaciones

Algoritmo de Suma/Resta:

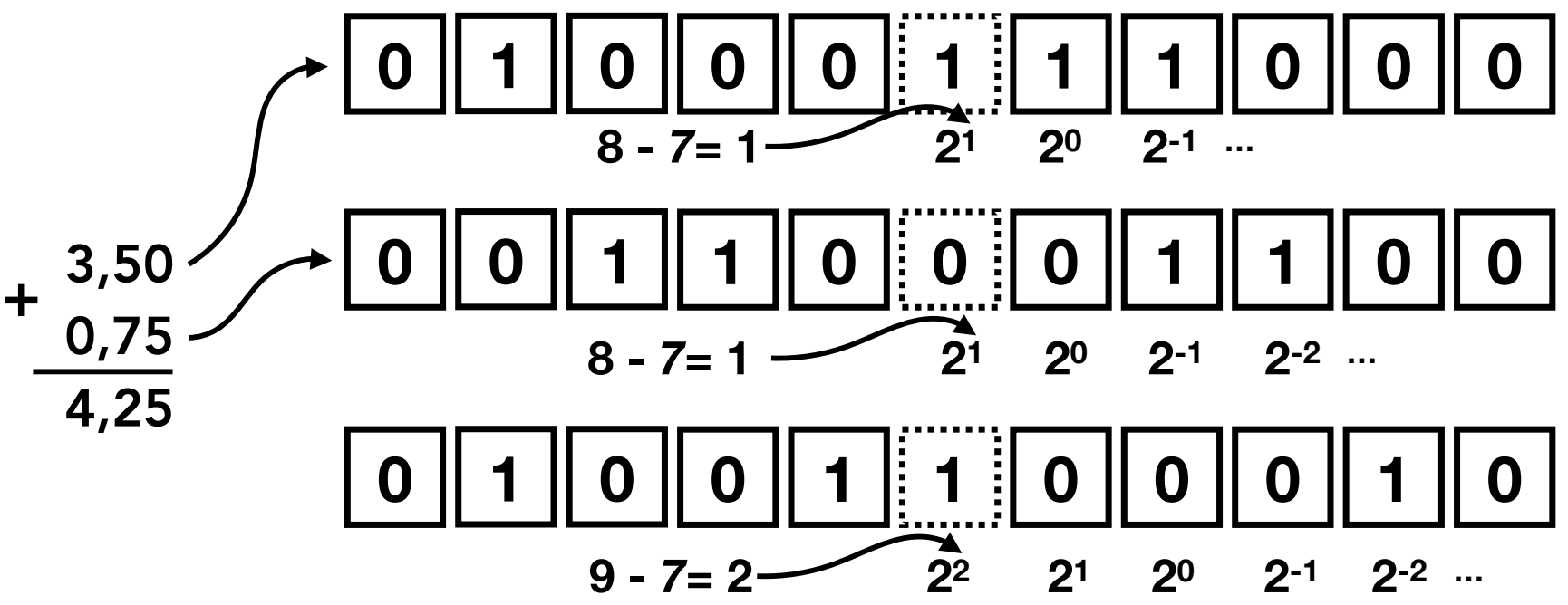
- 1) Elegir el operando con **menor exponente**, *alienar* la mantisa hacia la derecha incrementando el exponente hasta que éste **coincida** con el del otro operando. (Recordar el bit oculto)
- 2) Operar con las mantisas como la **suma/resta** de enteros.
- 3) Modificar el **signo** si fuese necesario.
- 4) **Normalizar** si fuese necesario.



Representación de Números Racionales con punto flotante - Operaciones

Algoritmo de Suma/Resta:

- 1) Elegir el operando con **menor exponente**, *alienar* la mantisa hacia la derecha incrementando el exponente hasta que éste **coincida** con el del otro operando. (Recordar el bit oculto)
- 2) Operar con las mantisas como la **suma/resta** de enteros.
- 3) Modificar el **signo** si fuese necesario.
- 4) **Normalizar** si fuese necesario.



Representación de Números Racionales con punto flotante - Operaciones

Algoritmo de Multiplicación:

- 1) Sumar los exponentes, restando una vez la polaridad para mantenerla.
(Ejemplo IEEE Single -127)
- 2) Multiplicar las mantisas
- 3) Modificar el signo si fuese necesario
- 4) Normalizar si fuese necesario

Algoritmo de División:

- 1) Restar los exponentes, sumando una vez la polaridad para mantenerla.
(Ejemplo IEEE Single -127)
- 2) Multiplicar las mantisas
- 3) Modificar el signo si fuese necesario
- 4) Normalizar si fuese necesario

Representación de Caracteres

ABC

ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) en 1967.
Fue sucesor de los códigos Baudot (telegrafía 5bits) y el código Murray (desarrollado para las máquinas de escribir “typewriter”).

- **32** caracteres de control
- **10** dígitos
- **52** letras (mayúsculas y minúsculas)
- **32** caracteres especiales
- **1** espacio

Representación de Caracteres

ASCII (utiliza 8 bits = 1 byte)

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

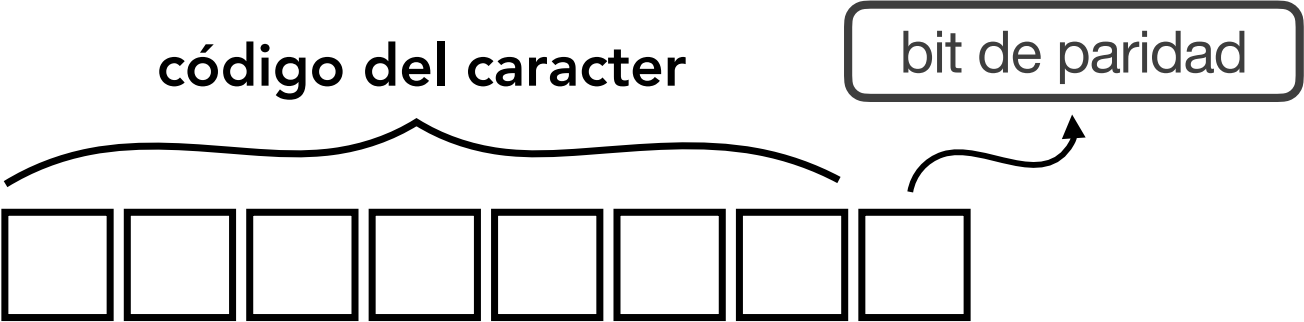
Representación de Caracteres

ASCII (utiliza 8 bits = 1 byte)

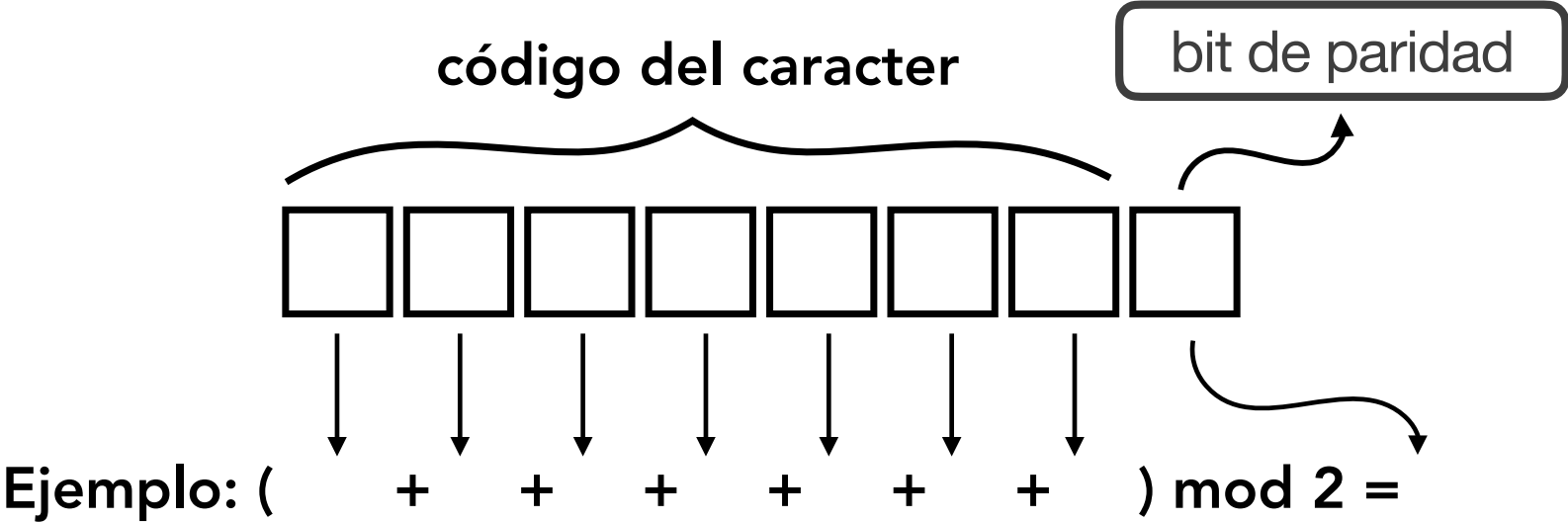
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

¿ y el 8bit ?

Representación de Caracteres - Bit de Paridad



Representación de Caracteres - Bit de Paridad



Representación de Caracteres

ABC

ASCII Extendido (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) en 1967.

Fue sucesor de los códigos Baudot (telegrafía 5bits) y el código Murray (desarrollado para las máquinas de escribir “typewriter”).

- **32** caracteres de control
- **10** dígitos
- **52** letras (mayúsculas y minúsculas)
- **32** caracteres especiales
- **1** espacio

Representación de Caracteres

ABC

ASCII Extendido en 1980.

Debido a la masificación de las computadoras se comenzó a utilizar el último bit para poder representar una mayor cantidad de símbolos: ñ Ñ Ç etc. . **256** símbolos, es compatible con ASCII

Unicode fue presentado en 1991

- utiliza 16 bits
- permite codificar todos los símbolos del mundo
- provee un mecanismo de extensión que permite codificar millones de caracteres
- es compatible con ASCII
- existen variantes UTF-8, UTF-16