

Grupo ARCOS

uc3m | Universidad **Carlos III** de Madrid

Tema 2

Representación de la información

Estructura de Computadores
Grado en Ingeniería Informática



Contenidos

1. Introducción

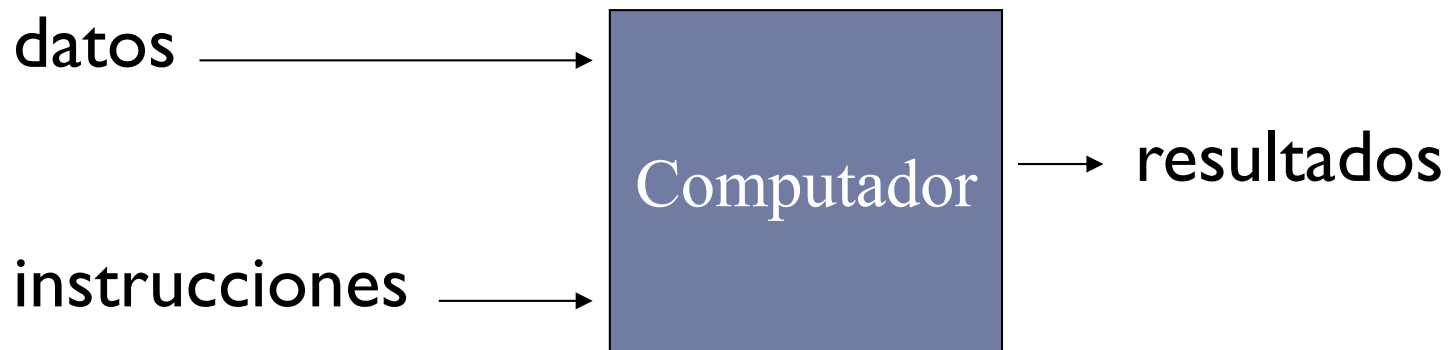
1. Motivación y objetivos
2. Sistemas posicionales

2. Representaciones

1. Alfanuméricas: caracteres y cadenas
2. Numéricas: naturales y enteras
3. Numéricas: coma fija
4. Numéricas: coma flotante: estándar IEEE 754

Introducción: computador

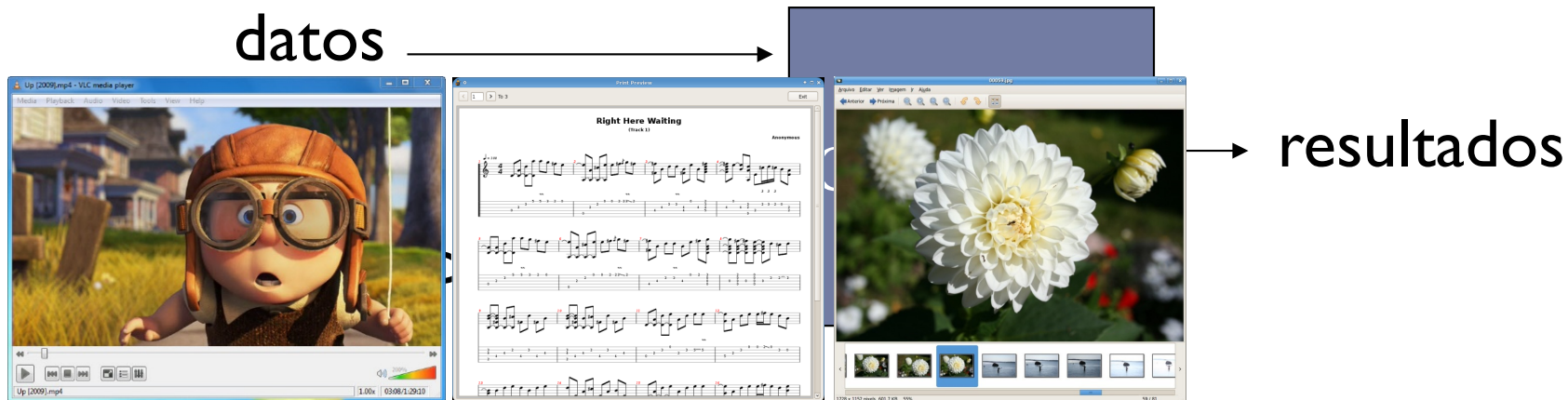
- ▶ Un computador es una máquina destinada a procesar datos.



- ▶ Se aplican unas instrucciones y se obtiene unos resultados

Introducción: computador

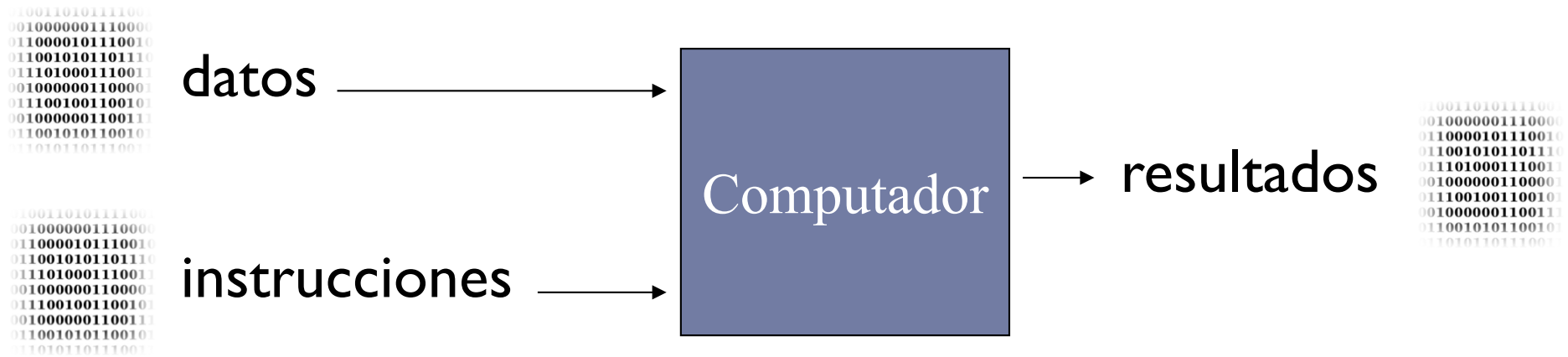
- ▶ Un computador es una máquina destinada a procesar datos.



- ▶ Se aplican unas instrucciones y se obtiene unos resultados
- ▶ Los datos/información pueden ser de **distintos tipo**

Introducción: computador

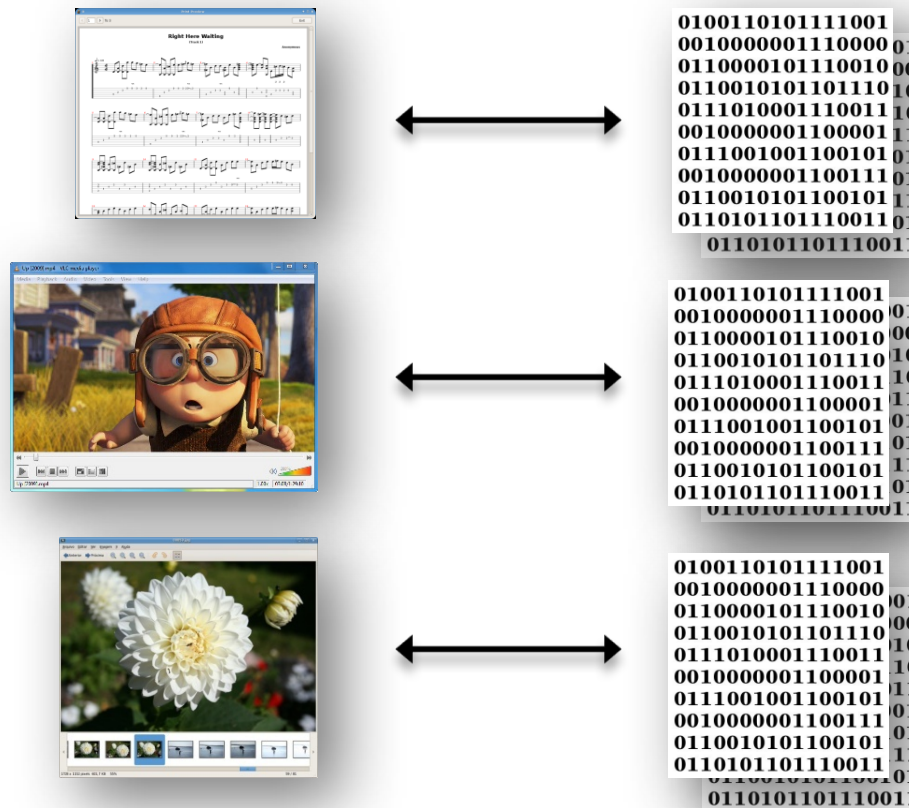
- ▶ Un computador es una máquina destinada a procesar datos.



- ▶ Se aplican unas instrucciones y se obtiene unos resultados
- ▶ Los datos/información pueden ser de **distintos tipo**
- ▶ Un computador solo usa una representación: **binario**.

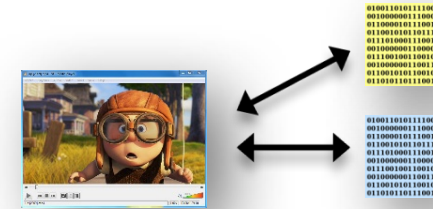
Introducción: representación de la información

- ▶ El uso de una **representación** permite transformar los distintos tipos de información en binario (y viceversa)



Necesitaremos...

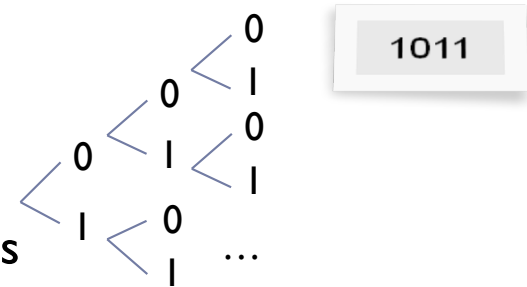
- ▶ Conocer posibles representaciones:



Introducción: características de la información

- ▶ Un **computador maneja un conjunto finito de valores**

- ▶ Tipo binario (dos estados)
- ▶ Finito (representación acotada)
 - ▶ N° de bits de palabra del computador
 - ▶ Con **n** bits se pueden codificar **2ⁿ** valores distintos



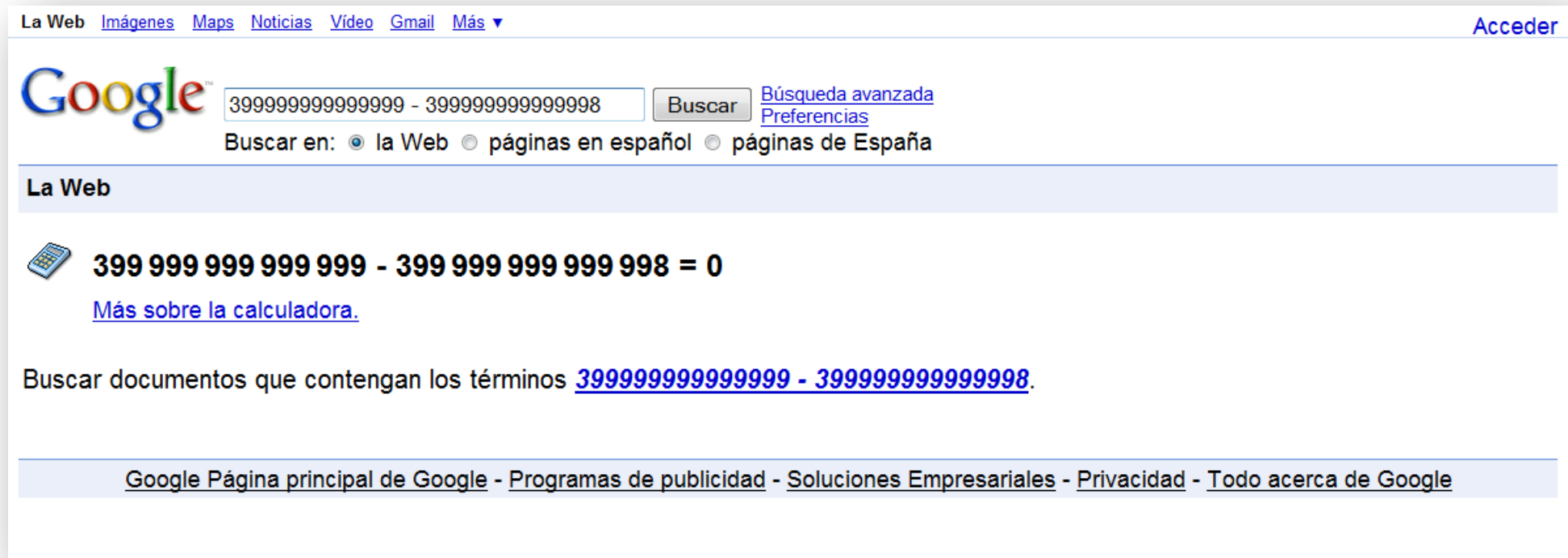
- ▶ Hay algunos tipos de información que son **infinitos**

- ▶ Imposible representar todos los valores de los números naturales, reales, etc.



- ▶ La representación elegida tiene **limitaciones**

Ejemplo 1: la calculadora de Google con 15 dígitos...



The screenshot shows the Google search interface. At the top, there are navigation links: "La Web", "Imágenes", "Maps", "Noticias", "Video", "Gmail", and "Más". On the right, there is a link to "Acceder". The Google logo is on the left, and the search bar contains the text "399999999999999 - 399999999999998". To the right of the search bar is a "Buscar" button and links for "Búsqueda avanzada" and "Preferencias". Below the search bar, there are radio buttons for "la Web" (selected), "páginas en español", and "páginas de España". A blue bar with the text "La Web" is below the search options. The main content area shows a calculator icon, the result "399 999 999 999 999 - 399 999 999 999 998 = 0", and a link "Más sobre la calculadora.". Below this, there is a search suggestion: "Buscar documentos que contengan los términos 399999999999999 - 399999999999998". At the bottom, there is a footer with links: "Google", "Página principal de Google", "Programas de publicidad", "Soluciones Empresariales", "Privacidad", and "Todo acerca de Google".

<http://www.20minutos.es/noticia/415383/0/google/restar/error/>

Ejemplo 2: la profundidad de color...

1 bit	2 colores
4 bits	16 colores
8 bits	256 colores



<http://platea.pntic.mec.es/~lgonzale/tic/imagen/conceptos.html>

Ejemplo 2: la profundidad de color...

1 bit	2 colores
4 bits	16 colores
8 bits	256 colores



<http://platea.pntic.mec.es/~lgonzale/tic/imagen/conceptos.html>

Ejemplo 2: la profundidad de color...

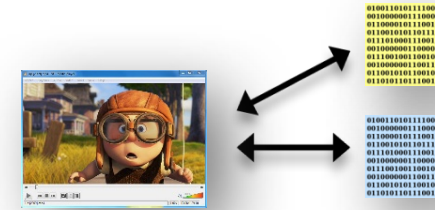
1 bit	2 colores
4 bits	16 colores
8 bits	256 colores



<http://platea.pntic.mec.es/~lgonzale/tic/imagen/conceptos.html>

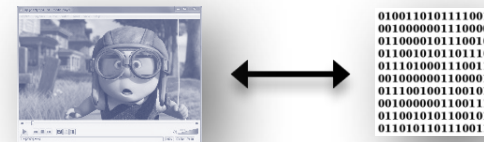
Necesitaremos...

- ▶ Conocer **posibles representaciones**:



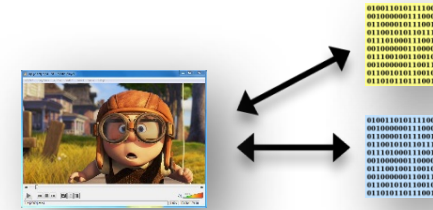
- ▶ Conocer las **características** de las mismas:

- ▶ Limitaciones



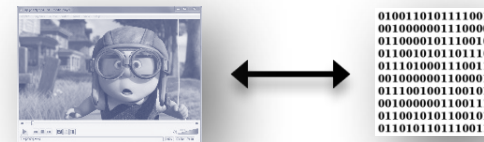
Necesitaremos...

- ▶ Conocer **posibles representaciones**:

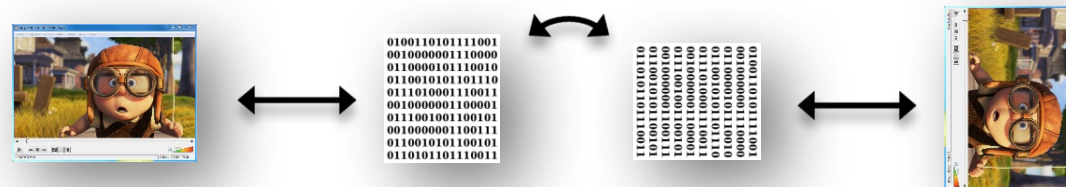


- ▶ Conocer las **características** de las mismas:

- ▶ Limitaciones



- ▶ Conocer **cómo operar** con la representación:



Contenidos

1. **Introducción**

1. Motivación y objetivos
2. **Sistemas posicionales**

2. Representaciones

1. Alfanuméricas: caracteres y cadenas
2. Numéricas: naturales y enteras
3. Numéricas: coma fija
4. Numéricas: coma flotante: estándar IEEE 754

Sistemas de representación posicionales

- ▶ Un número se define por una **cadena de dígitos**, estando **afectado** cada uno de ellos por un **factor de escala** que **depende** de la **posición** que ocupa en la cadena.

- ▶ Dada una base de numeración b , un número X se define como la cadena de dígitos:

$$X = (\dots x_2 x_1 x_0, x_{-1} x_{-2} \dots)_b \quad \text{Con } 0 \leq x_i < b$$

con una lista de pesos asociados:

$$P = (\dots b^2 b^1 b^0 \quad b^{-1} b^{-2} \dots)_b$$

Sistemas de representación posicionales

- ▶ Un número se define por una **cadena de dígitos**, estando **afectado** cada uno de ellos por un **factor de escala** que **depende** de la **posición** que ocupa en la cadena.

- ▶ Dada una base de numeración b , un número X se define como la cadena de dígitos:

$$X = (\dots x_2 x_1 x_0, x_{-1} x_{-2} \dots)_b \quad \text{Con } 0 \leq x_i < b$$

con una lista de pesos asociados:

$$P = (\dots b^2 b^1 b^0 \quad b^{-1} b^{-2} \dots)_b$$

- ▶ Su valor es:

$$V(X) = \sum_{i=-\infty}^{+\infty} b^i \cdot x_i = \dots \underset{\text{bolsa}}{b^2} \cdot x_2 + \underset{\text{bolsa}}{b^1} \cdot x_1 + \underset{\text{bolsa}}{b^0} \cdot x_0 + \underset{\text{bolsa}}{b^{-1}} \cdot x_{-1} + \underset{\text{bolsa}}{b^{-2}} \cdot x_{-2} \dots$$

Sistemas de representación posicionales

▶ Decimal

$$X = \quad 9 \quad 7 \quad 3 \quad 1 \\ \quad \dots 10^3 \ 10^2 \ 10^1 \ 10^0$$

▶ Binario

$$X = \quad 0 \ 1 \ 0 \ 1 \\ \quad \dots 2^3 \ 2^2 \ 2^1 \ 2^0$$

▶ Hexadecimal

$$X = \quad 1 \quad F \quad A \quad 8 \\ \quad \dots 16^3 \ 16^2 \ 16^1 \ 16^0$$

Sistemas de representación posicionales

▶ Decimal

$$X = \quad 9 \quad 7 \quad 3 \quad 1 \\ \dots 10^3 \ 10^2 \ 10^1 \ 10^0$$

▶ Binario

$$X = \quad 0 \ 1 \ 0 \ 1 \\ \dots 2^3 \ 2^2 \ 2^1 \ 2^0$$

▶ Hexadecimal

$$X = \quad 1 \quad F \quad A \quad 8 \\ \dots 16^3 \ 16^2 \ 16^1 \ 16^0$$

Paso de binario a hexadecimal:

- ▶ Agrupar de 4 en 4 bits, de derecha a izquierda
- ▶ Cada 4 bits es el valor del dígito hexadecimal

▶ Ej.: $\quad 1 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 1$
 $\quad 0x \quad \underline{1010} \quad \underline{0101}$
 $\quad \quad \quad A \quad \quad 5$

Sistemas de representación posicionales

▶ Decimal

$$X = \begin{array}{cccc} 9 & 7 & 3 & 1 \\ \dots & 10^3 & 10^2 & 10^1 & 10^0 \end{array}$$

▶ Binario

$$X = \begin{array}{cccc} 0 & 1 & 0 & 1 \\ \dots & 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

▶ Hexadecimal

$$X = \begin{array}{cccc} 1 & F & A & 8 \\ \dots & 16^3 & 16^2 & 16^1 & 16^0 \end{array}$$

¿?



Ejercicio

- ▶ Representar 342 en binario:

256	128	64	32	16	8	4	2	1
?	?	?	?	?	?	?	?	?

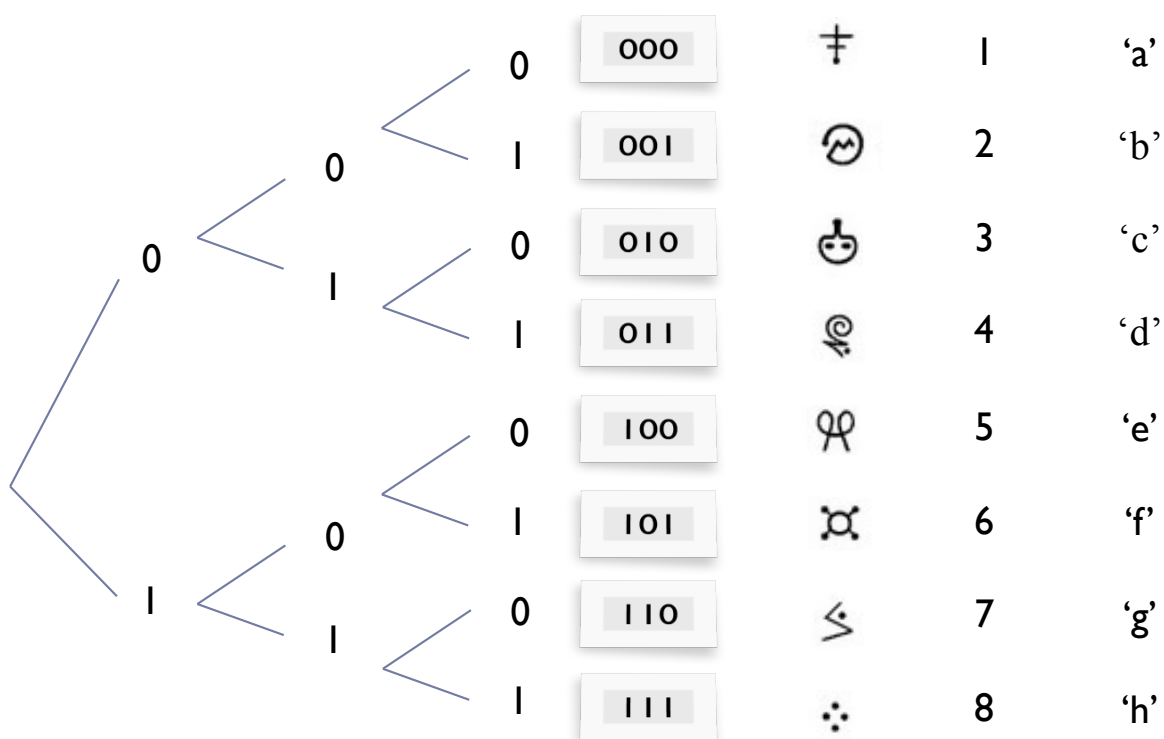
Ejercicio (solución)

- ▶ Representar 342 en binario:

256	128	64	32	16	8	4	2	1
1	0	1	0	1	0	1	1	0

Ejemplo

- ▶ Con 3 dígitos binarios, se pueden representar 8 símbolos:



Sistemas de representación posicionales

- ▶ ¿Cuántos valores se pueden representar con n bits?
- ▶ ¿Cuántos bits se necesitan para representar m 'valores'?
- ▶ Con n bits,
si el valor mínimo representable corresponde al número 0,
¿Cuál es el máximo valor numérico representable?

Sistemas de representación posicionales

- ▶ **¿Cuántos valores** se pueden representar con **n bits**?
 - ▶ 2^n
 - ▶ Ej.: con 4 bits se pueden representar 16 valores
- ▶ **¿Cuántos bits** se necesitan para representar **m 'valores'**?
 - ▶ **$\log_2(n)$ por exceso**
 - ▶ Ej.: para representar 35 valores se necesitan 6 bits
- ▶ Con n bits,
si el valor mínimo representable corresponde al número 0,
¿Cuál es el máximo valor numérico representable?
 - ▶ $2^n - 1$



Ejercicio

- ▶ Calcular el valor de (23 unos):

|||||||||||||||||||||||₂

Ejercicio (solución)

- ▶ Calcular el valor de (23 unos):

$$\text{|||||} \dots \text{|||||}_2$$

$$X = 2^{23} - 1$$

Truco:

$$\begin{array}{r} \text{|||||} \dots \text{|||||}_2 = X \\ + \text{00000000000000000000000}_2 = 1 \\ \hline \text{10000000000000000000000}_2 = 2^{23} \end{array}$$

$$X = 2^{23} - 1$$

Ejemplo: operaciones

► Sumar en binario:

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

Ejemplo: operaciones

► **Sumar** en binario:

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

► **Restar** en binario:

$$\begin{array}{r} \\ \\ \\ \\ \\ - \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \\ \hline 0 \quad 0 \quad 0 \quad 0 \quad 1 \end{array}$$

1 → 1 →

Contenidos

1. Introducción

1. Motivación y objetivos
2. Sistemas posicionales

2. Representaciones

1. **Alfanuméricas: caracteres y cadenas**
2. Numéricas: naturales y enteras
3. Numéricas: coma fija
4. Numéricas: coma flotante: estándar IEEE 754

Representación alfanumérica

- ▶ Cada carácter se codifica con un byte.
- ▶ Para n bits $\Rightarrow 2^n$ caracteres representables:

# bits	# caracteres	Incluye...	Ejemplo
6	64	<ul style="list-style-type: none">• 26 letras: a...z• 10 números: 0...9• Puntuación: .,;:...• Especiales: + - [...	BCDIC
7	128	<ul style="list-style-type: none">• añade mayúsculas y caracteres de control	ASCII
8	256	<ul style="list-style-type: none">• añade letras acentuadas, ñ, caracteres semigráficos	EBCDIC ASCII extendido
16	34.168	<ul style="list-style-type: none">• Añade distintos idiomas (chino, árabe,...)	UNICODE

Ejemplo: tabla ASCII (7 bits)

ASCII value	Character	Control character	ASCII value	Character	ASCII value	Character	ASCII value	Character
000	(null)	NUL	032	(space)	064	@	096	
001	☺	SOH	033	!	065	A	097	a
002	☻	STX	034	"	066	B	098	b
003	♥	ETX	035	#	067	C	099	c
004	♦	EOT	036	\$	068	D	100	d
005	♣	ENQ	037	%	069	E	101	e
006	♠	ACK	038	&	070	F	102	f
007	(beep)	BEL	039	'	071	G	103	g
008	■	BS	040	(072	H	104	h
009	(tab)	HT	041)	073	I	105	i
010	(line feed)	LF	042	*	074	J	106	j
011	(home)	VT	043	+	075	K	107	k
012	(form feed)	FF	044	,	076	L	108	l
013	(carriage return)	CR	045	-	077	M	109	m
014	♪	SO	046	.	078	N	110	n
015	☼	SI	047	/	079	O	111	o
016	▲	DLE	048	0	080	P	112	p
017	▴	DC1	049	1	081	Q	113	q
018	↕	DC2	050	2	082	R	114	r
019		DC3	051	3	083	S	115	s
020	#	DC4	052	4	084	T	116	t
021	\$	NAK	053	5	085	U	117	u
022	■	SYN	054	6	086	V	118	v
023	↕	ETB	055	7	087	W	119	w
024	↕	CAN	056	8	088	X	120	x
025	↕	EM	057	9	089	Y	121	y
026	↑	SUB	058	:	090	Z	122	z
027	←	ESC	059	::	091	[123	{
028	(cursor right)	FS	060	<	092	\	124	
029	(cursor left)	GS	061	=	093]	125	}
030	(cursor up)	RS	062	>	094	^	126	~
031	(cursor down)	US	063	?	095	_	127	☑

Copyright 1998, JimPrice.Com Copyright 1982, Loading Edge Computer Products, Inc.

Ejemplo: tabla ASCII (7 bits)

caracteres de control

ASCII value	Character	Control character	ASCII value	Character	ASCII value	Character	ASCII value	Character
000	(null)	NUL	032	(space)	064	@	096	
001	☺	SOH	033	!	065	A	097	a
002	☻	STX	034	"	066	B	098	b
003	♥	ETX	035	#	067	C	099	c
004	♦	EOT	036	\$	068	D	100	d
005	♣	ENQ	037	%	069	E	101	e
006	♠	ACK	038	&	070	F	102	f
007	(beep)	BEL	039	'	071	G	103	g
008	■	BS	040	(072	H	104	h
009	(tab)	HT	041)	073	I	105	i
010	(line feed)	LF	042	*	074	J	106	j
011	(home)	VT	043	+	075	K	107	k
012	(form feed)	FF	044	,	076	L	108	l
013	(carriage return)	CR	045	-	077	M	109	m
014	🎵	SO	046	.	078	N	110	n
015	☼	SI	047	/	079	O	111	o
016	▲	DLE	048	0	080	P	112	p
017	▴	DC1	049	1	081	Q	113	q
018	↕	DC2	050	2	082	R	114	r
019		DC3	051	3	083	S	115	s
020	#	DC4	052	4	084	T	116	t
021	\$	NAK	053	5	085	U	117	u
022	■	SYN	054	6	086	V	118	v
023	↕	ETB	055	7	087	W	119	w
024	↕	CAN	056	8	088	X	120	x
025	↕	EM	057	9	089	Y	121	y
026	↑	SUB	058	:	090	Z	122	z
027	←	ESC	059	;	091	[123	{
028	(cursor right)	FS	060	<	092	\	124	
029	(cursor left)	GS	061	=	093]	125	}
030	(cursor up)	RS	062	>	094	^	126	~
031	(cursor down)	US	063	?	095	_	127	␣

< 32

Ejemplo: tabla ASCII (7 bits)

distancia mayúsculas-minúsculas

ASCII value	Character	Control character	ASCII value	Character	ASCII value	Character	ASCII value	Character
000	(null)	NUL	032	(space)	064	@	096	a
001	☺	SOH	033	!	065	A	097	á
002	☻	STX	034	"	066	B	098	b
003	♥	ETX	035	#	067	C	099	c
004	♦	EOT	036	\$	068	D	100	d
005	♣	ENQ	037	%	069	E	101	e
006	♠	ACK	038	&	070	F	102	f
007	(beep)	BEL	039	'	071	G	103	g
008	■	BS	040	(072	H	104	h
009	(tab)	HT	041)	073	I	105	i
010	(line feed)	LF	042	*	074	J	106	j
011	(home)	VT	043	+	075	K	107	k
012	(form feed)	FF	044	,	076	L	108	l
013	(carriage return)	CR	045	-	077	M	109	m
014	♪	SO	046	.	078	N	110	n
015	☼	SI	047	/	079	O	111	o
016	▲	DLE	048	0	080	P	112	p
017	▴	DC1	049	1	081	Q	113	q
018	↕	DC2	050	2	082	R	114	r
019		DC3	051	3	083	S	115	s
020	#	DC4	052	4	084	T	116	t
021	\$	NAK	053	5	085	U	117	u
022	■	SYN	054	6	086	V	118	v
023	↕	ETB	055	7	087	W	119	w
024	↕	CAN	056	8	088	X	120	x
025	↕	EM	057	9	089	Y	121	y
026	↑	SUB	058	:	090	Z	122	z
027	←	ESC	059	::	091	[123	{
028	(cursor right)	FS	060	<	092	\	124	
029	(cursor left)	GS	061	=	093]	125	}
030	(cursor up)	RS	062	>	094	^	126	~
031	(cursor down)	US	063	?	095	_	127	☑

$$97-65=32$$

Ejemplo: tabla ASCII (7 bits)

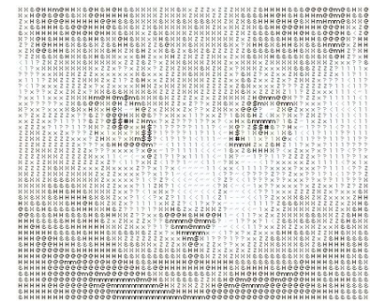
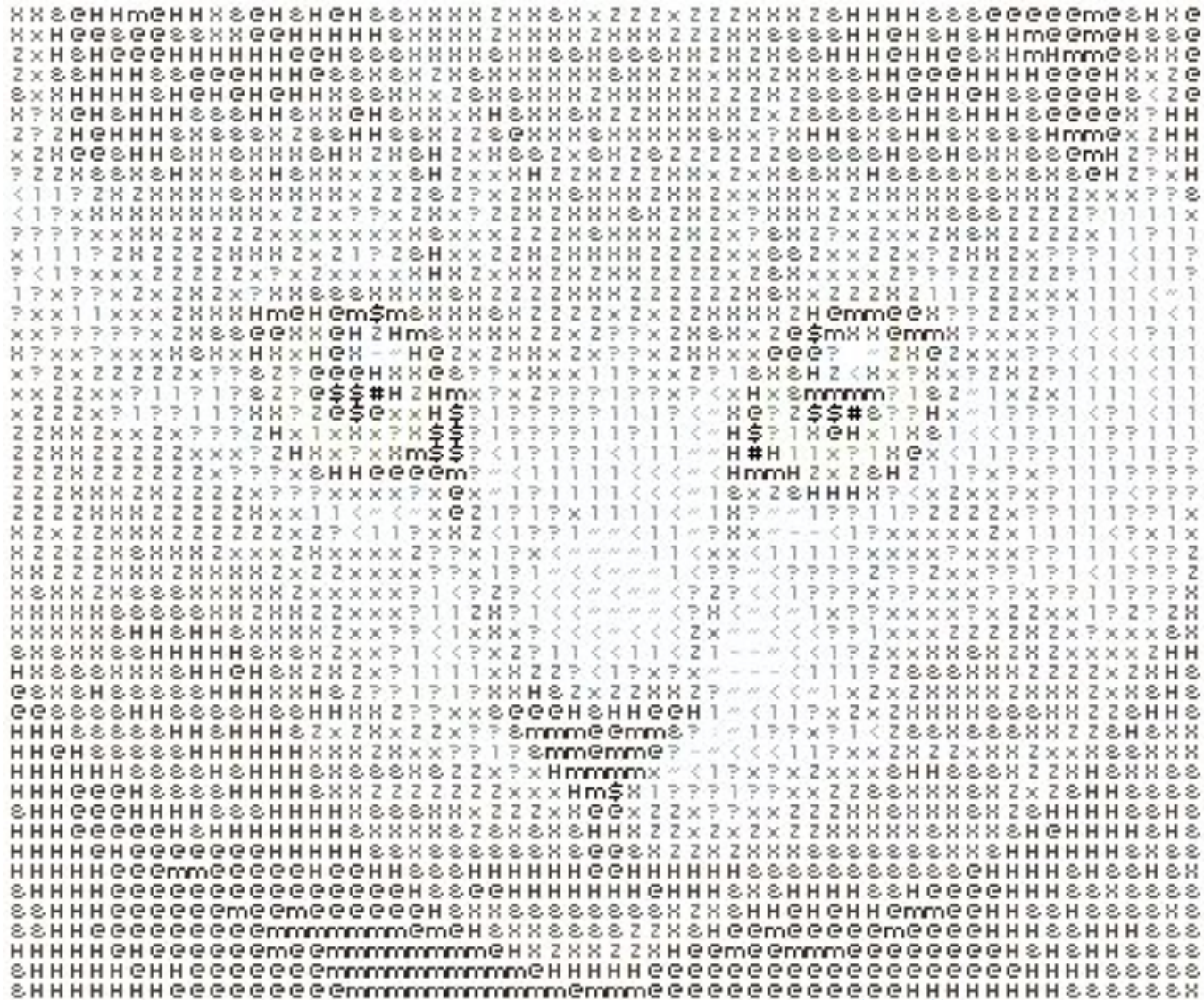
conversión de un número a carácter

ASCII value	Character	Control character	ASCII value	Character	ASCII value	Character	ASCII value	Character
000	(null)	NUL	032	(space)	064	@	096	
001	☺	SOH	033	!	065	A	097	a
002	☻	STX	034	"	066	B	098	b
003	♥	ETX	035	#	067	C	099	c
004	♦	EOT	036	\$	068	D	100	d
005	♣	ENQ	037	%	069	E	101	e
006	♠	ACK	038	&	070	F	102	f
007	(beep)	BEL	039	'	071	G	103	g
008	■	BS	040	(072	H	104	h
009	(tab)	HT	041)	073	I	105	i
010	(line feed)	LF	042	*	074	J	106	j
011	(home)	VT	043	+	075	K	107	k
012	(form feed)	FF	044	,	076	L	108	l
013	(carriage return)	CR	045	-	077	M	109	m
014	♪	SO	046	.	078	N	110	n
015	☼	SI	047	/	079	O	111	o
016	▼	DLE	048	0	080	P	112	p
017	▲	DC1	049	1	081	Q	113	q
018	↕	DC2	050	2	082	R	114	r
019		DC3	051	3	083	S	115	s
020	#	DC4	052	4	084	T	116	t
021	\$	NAK	053	5	085	U	117	u
022	■	SYN	054	6	086	V	118	v
023	↕	ETB	055	7	087	W	119	w
024	↕	CAN	056	8	088	X	120	x
025	↕	EM	057	9	089	Y	121	y
026	↑	SUB	058	:	090	Z	122	z
027	←	ESC	059	;	091	[123	{
028	(cursor right)	FS	060	<	092	\	124	
029	(cursor left)	GS	061	=	093]	125	}
030	(cursor up)	RS	062	>	094	^	126	~
031	(cursor down)	US	063	?	095	_	127	☑

Copyright 1998, JimPrice.Com Copyright 1982, Loading Edge Computer Products, Inc.

$$6+48=54$$

Curiosidad: Visualización 'gráfica' con caracteres

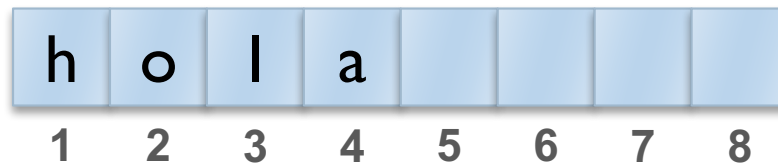


<http://www.typorganism.com/asciomatic/>

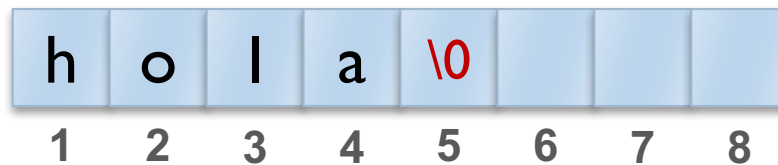
Cadenas de caracteres

1000	00110011
1001	01101100
	...
1008	10100011

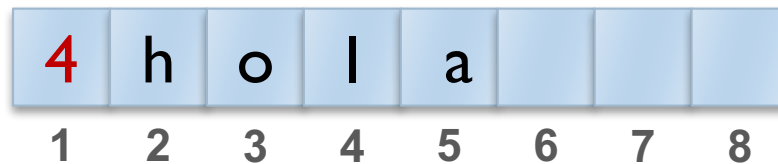
1. Cadenas de longitud fija:



2. Cadenas de longitud variable con separador:



3. Cadenas de longitud variable con longitud en cabecera:



Contenidos

1. Introducción

1. Objetivo
2. Motivación
3. Sistemas posicionales

2. Representaciones

1. Alfanuméricas: caracteres y cadenas
2. **Numéricas: naturales y enteras**
3. Numéricas: coma fija
4. Numéricas: coma flotante: estándar IEEE 754

Representación numérica

- ▶ **Clasificación de números reales:**
 - ▶ Naturales: 0, 1, 2, 3, ...
 - ▶ Enteros: ... -3, -2, -1, 0, 1, 2, 3,
 - ▶ Racionales: fracciones ($5/2 = 2,5$)
 - ▶ Irracionales: $2^{1/2}$, π , e, ...
- ▶ **Conjuntos infinitos y espacio de representación finito:**
 - ▶ Imposible representar todos
- ▶ **Características de la representación usada:**
 - ▶ Elemento representado:
Natural, entero, ...
 - ▶ Rango de representación:
Intervalo entre el menor y mayor n^o representable
 - ▶ Resolución de representación:
Diferencia entre un n^o representable y el siguiente.
Representa el máximo error cometido. Puede ser cte. o variable.

Sistemas de representación binarios más usados

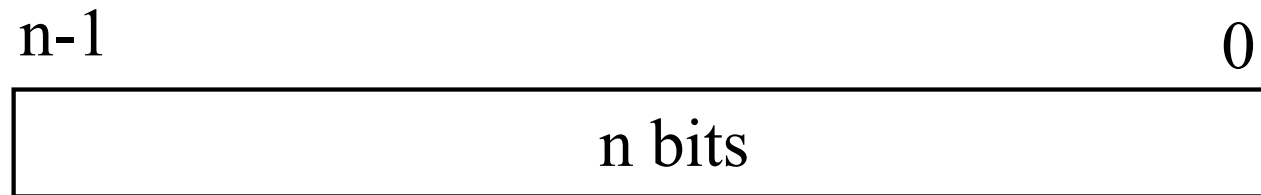
- A. Coma fija sin signo o binario puro naturales

- B. Signo magnitud
- C. Complemento a uno (Ca 1) enteros
- D. Complemento a dos (Ca 2)
- E. Exceso $2^{n-1}-1$

- F. Coma flotante: Estándar IEEE 754 reales

Coma fija sin signo o binario puro [naturales]

- ▶ Sistema posicional con base 2 y sin parte fraccionaria.



$$V(X) = \sum_{i=0}^{n-1} 2^i \cdot x_i$$

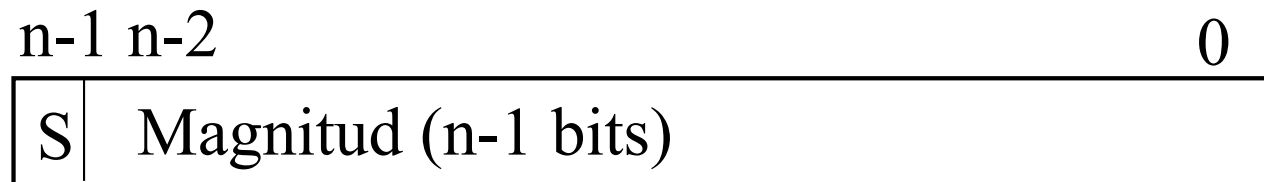
- Rango de representación: $[0, 2^n - 1]$
- Resolución: 1 unidad

Ejemplo comparativo (3 bits)

Decimal	Binario Puro
+7	111
+6	110
+5	101
+4	100
+3	011
+2	010
+1	001
+0	000
-0	N.D.
-1	N.D.
-2	N.D.
-3	N.D.
-4	N.D.
-5	N.D.
-6	N.D.
-7	N.D.

Coma fija con signo o signo magnitud [enteros]

- ▶ Se reserva un bit (S) para el signo (0 \Rightarrow +; 1 \Rightarrow -)



$$\begin{array}{l}
 \text{Si } x_{n-1} = 0 \quad V(X) = \sum_{i=0}^{n-2} 2^i \cdot X_i \\
 \text{Si } x_{n-1} = 1 \quad V(X) = -\sum_{i=0}^{n-2} 2^i \cdot X_i
 \end{array}
 \left| \Rightarrow V(X) = (1 - 2 \cdot x_{n-1}) \cdot \sum_{i=0}^{n-2} 2^i \cdot X_i
 \right.$$

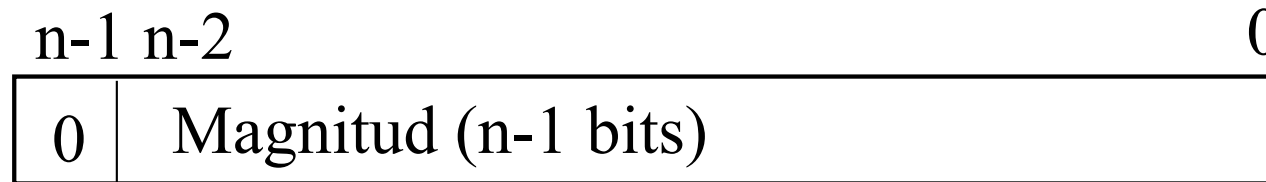
- Rango de representación: $[-2^{n-1} + 1, 2^{n-1} - 1]$
- Resolución: 1 unidad
- Ambigüedad del 0

Ejemplo comparativo (3 bits)

Decimal	Binario Puro	Signo magnitud
+7	111	N.D.
+6	110	N.D.
+5	101	N.D.
+4	100	N.D.
+3	011	011
+2	010	010
+1	001	001
+0	000	000
-0	N.D.	100
-1	N.D.	101
-2	N.D.	110
-3	N.D.	111
-4	N.D.	N.D.
-5	N.D.	N.D.
-6	N.D.	N.D.
-7	N.D.	N.D.

Complemento a uno (a la base menos uno) [enteros] (1 / 3)

- ▶ **Número positivo:**
se representa en binario puro con n-1 bits



$$V(X) = \sum_{i=0}^{n-1} 2^i \cdot x_i = \sum_{i=0}^{n-2} 2^i \cdot x_i$$

- Rango de representación (+): $[0, 2^{n-1} - 1]$
- Resolución: 1 unidad

Complemento a uno (a la base menos uno) [enteros] (2/3)

▶ **Número negativo:**

- ▶ Se complementa a la base menos uno
- ▶ El número $X < 0$ se representa como $2^n - X - 1$

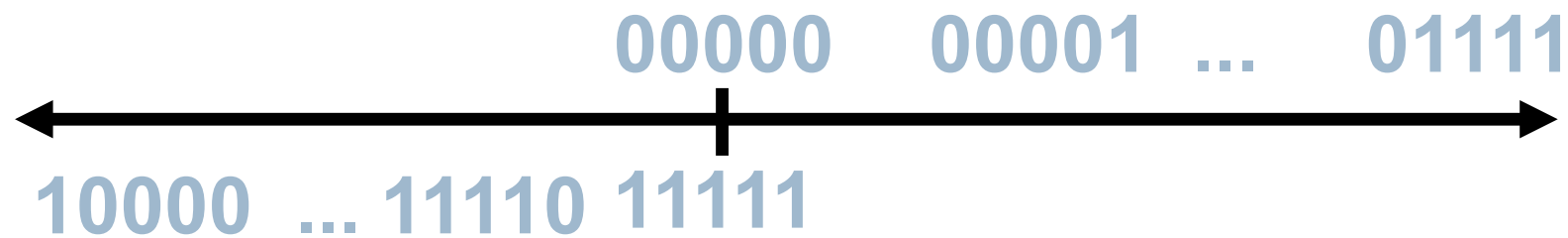


$$V(X) = -2^n + \sum_{i=0}^{n-1} 2^i \cdot y_i + 1$$

- Rango de representación (-): $[-(2^{n-1}-1), -0]$
- Resolución: 1 unidad

Complemento a uno

- ▶ Los números positivos tienen un 0 en el bit más significativo



- ▶ Los números negativos tienen un 1 en el bit más significativo

Complemento a uno (a la base menos uno) [enteros] (3/3)

Truco: $C a 1 (X) = X$

$C a 1 (-X) =$ cambiar los 1 por 0 y los 0 por 1

- ▶ Ejemplo: Para $n=4 \Rightarrow$ el valor $+3_{10} = 0011_2$
- ▶ Ejemplo: Para $n=4 \Rightarrow$ el valor $-3_{10} = 1100_2$
 - ▶ - \Rightarrow 1 (bit signo y también parte de magnitud)
 - ▶ $C a 1(3) \Rightarrow 2^4 - 0011_2 - 1 = 2^4 - 3 - 1 = 12 \Rightarrow 1100_2$

- Rango de representación: $[-2^{n-1}+1, 2^{n-1}-1]$
- Resolución: 1 unidad
- El 0 tiene doble representación (+0 y -0)
- Rango simétrico

Ejemplo comparativo (3 bits)

Decimal	Binario Puro	Signo magnitud	Complemento a uno
+7	111	N.D.	N.D.
+6	110	N.D.	N.D.
+5	101	N.D.	N.D.
+4	100	N.D.	N.D.
+3	011	011	011
+2	010	010	010
+1	001	001	001
+0	000	000	000
-0	N.D.	100	111
-1	N.D.	101	110
-2	N.D.	110	101
-3	N.D.	111	100
-4	N.D.	N.D.	N.D.
-5	N.D.	N.D.	N.D.
-6	N.D.	N.D.	N.D.
-7	N.D.	N.D.	N.D.

Ejemplo

- ▶ Para $n = 5$ bits
- ▶ ¿Cómo se representa $X = 5$?
 - ▶ Como es positivo, en binario puro
 - ▶ 00101
- ▶ ¿Cómo se representa $X = -5$?
 - ▶ Como es negativo, se complementa el valor 5 (00101)
 - ▶ 11010
- ▶ ¿Cuál es el valor de 00111 en complemento a 2?
 - ▶ Como es positivo, su valor es directamente 7
- ▶ ¿Cuál es el valor de 11000 en complemento a 2?
 - ▶ Como es negativo, se complementa y se obtiene 00111 (7)
 - ▶ El valor es -7

Aritmética en complemento a uno

- ▶ Sumas y restas se realizan de igual forma
- ▶ Para $n = 5$ bits
- ▶ Sea $X = 5$
 - ▶ En complemento a uno = 00101
- ▶ Sea $Y = 7$
 - ▶ En complemento a uno = 00111
- ▶ ¿ $X + Y$?

X	=	00101
Y	=	<u>00111</u> +
$X+Y$	=	01100
- ▶ El valor de 01100 en complemento a uno es 12

Aritmética en complemento a uno

- ▶ Para $n = 5$ bits
- ▶ Sea $X = -5$
 - ▶ En complemento a uno = complemento de 00101 : 11010
- ▶ Sea $Y = -7$
 - ▶ En complemento a uno = complemento de 00111 : 11000
- ▶ ¿ $X + Y$?

$$-X = 11010$$

$$-Y = \underline{11000}$$

$$-(X+Y) = 110010$$

desprecia

Se produce un acarreo, se suma y se

$$\begin{array}{r} \underline{\quad\quad\quad} 1 \\ 10011 \end{array}$$

- ▶ El valor de 10011 en complemento a uno es el valor negativo de su complemento $-01100 = -12$

¿Porqué se desprecia el acarreo y se suma al resultado?

- ▶ $-X$ se representa como $2^n - X - 1$
- ▶ $-Y$ se representa como $2^n - Y - 1$
- ▶ $-(X + Y)$ se representa como $2^n - (X+Y) - 1$

- ▶ Cuando sumamos directamente $-X - Y$ se obtiene

$$-X = 2^n - X - 1$$

$$-Y = \underline{2^n - Y - 1}$$

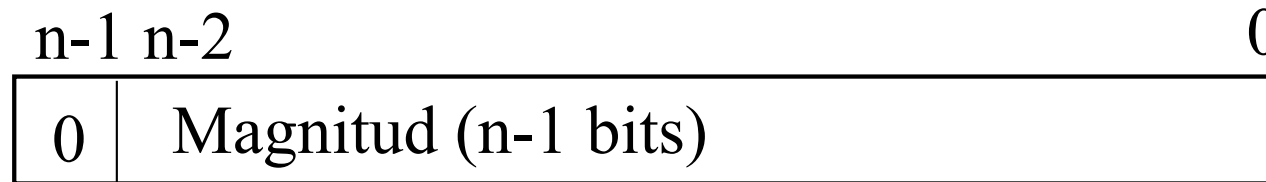
$$-(X+Y) = \textcircled{2^n} + 2^n - (X + Y) - 2$$

Se corrige el resultado sumando el acarreo (2^n)
y despreciándolo

$$\Rightarrow 2^n - (X + Y) - 1$$

Complemento a dos (complemento a la base) [enteros] (1 / 3)

- ▶ **Número positivo:**
se representa en binario puro con n-1 bits



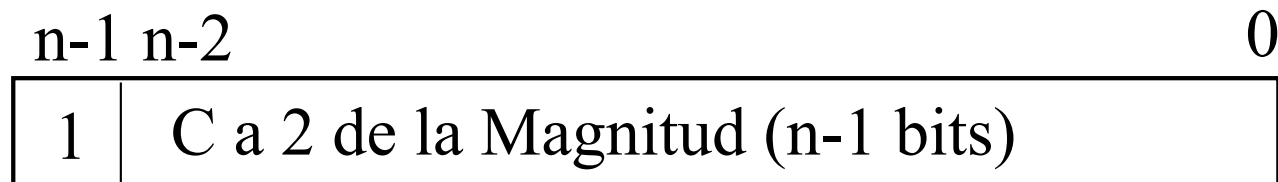
$$V(X) = \sum_{i=0}^{n-1} 2^i \cdot x_i = \sum_{i=0}^{n-2} 2^i \cdot x_i$$

- Rango de representación (+): $[0, 2^{n-1} - 1]$
- Resolución: 1 unidad

Complemento a dos (complemento a la base) [enteros] (2/3)

▶ **Número negativo:**

- ▶ Se complementa a la base
- ▶ El número $X < 0$ se representa como $2^n - X$



$$V(X) = -2^n + \sum_{i=0}^{n-1} 2^i \cdot y_i$$

- Rango de representación (-): $[-2^{n-1}, -1]$
- Resolución: 1 unidad

Complemento a dos (complemento a la base) [enteros] (3/3)

$$\begin{aligned}\text{Truco: } C a 2 (X) &= X \\ C a 2 (-X) &= C a 1 (X) + 1\end{aligned}$$

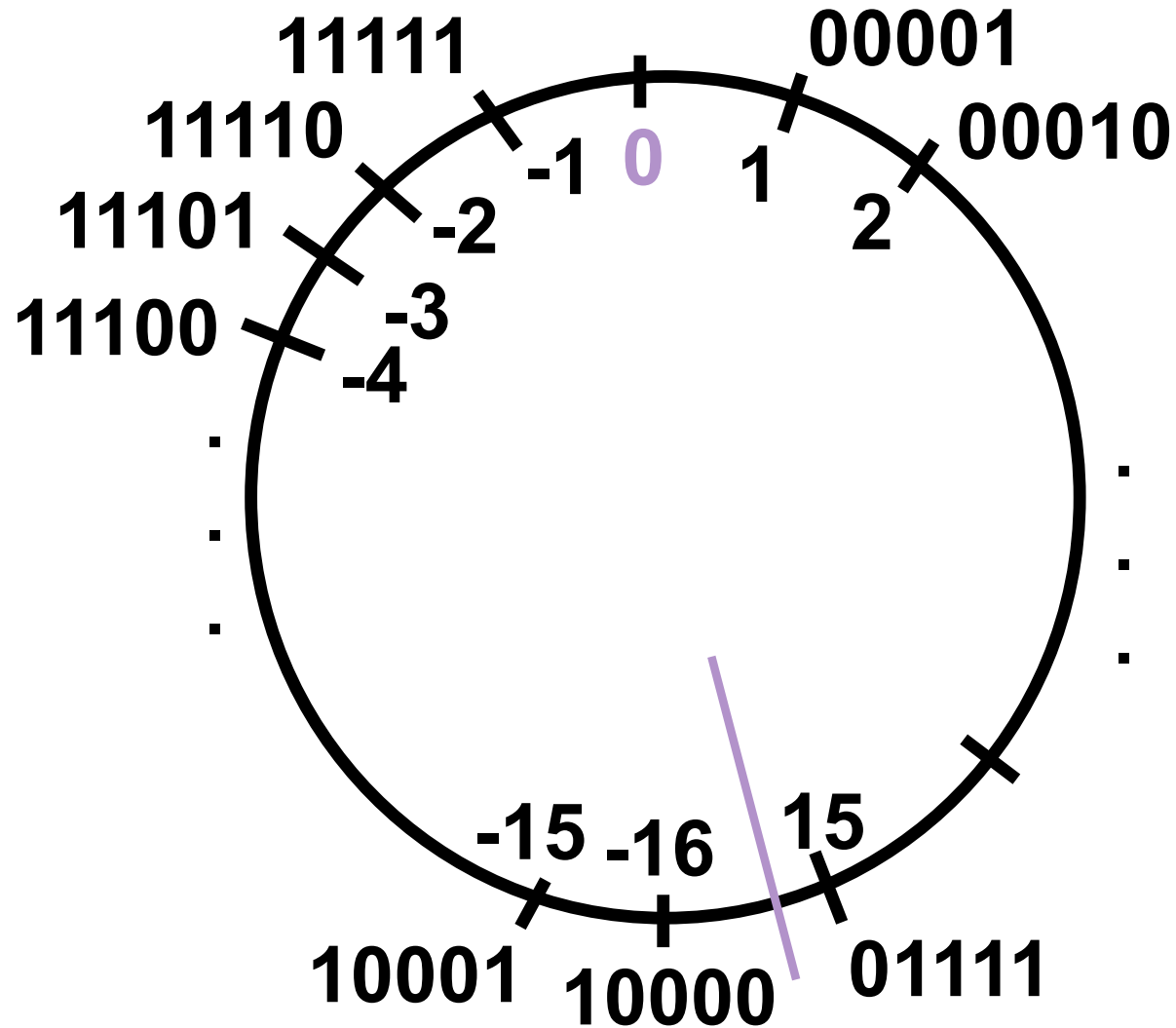
- ▶ Ejemplo: Para $n=4 \Rightarrow +3 = 0011_2$
- ▶ Ejemplo: Para $n=4 \Rightarrow -3 = 1101_2$
 - ▶ $1 \Rightarrow -$ (bit signo y también parte de magnitud)
 - ▶ $C a 2 (3) = C a 2(0011_2) = 2^4 - 3 = 13 \Rightarrow 1101_2$

- Rango de representación: $[-2^{n-1}, 2^{n-1}-1]$
- Resolución: 1 unidad
- El 0 tiene una única representación (No $\exists -0$)
- Rango asimétrico

Ejemplo comparativo (3 bits)

Decimal	Binario Puro	Signo magnitud	Complemento a uno	Complemento a dos
+7	111	N.D.	N.D.	N.D.
+6	110	N.D.	N.D.	N.D.
+5	101	N.D.	N.D.	N.D.
+4	100	N.D.	N.D.	N.D.
+3	011	011	011	011
+2	010	010	010	010
+1	001	001	001	001
+0	000	000	000	000
-0	N.D.	100	111	N.D.
-1	N.D.	101	110	111
-2	N.D.	110	101	110
-3	N.D.	111	100	101
-4	N.D.	N.D.	N.D.	100
-5	N.D.	N.D.	N.D.	N.D.
-6	N.D.	N.D.	N.D.	N.D.
-7	N.D.	N.D.	N.D.	N.D.

Complemento a dos



2^{N-1} no negativos

2^{N-1} negativos

Un cero

Complemento a dos para 32 bits

$$\begin{aligned} 0000 \dots 0000 \ 0000 \ 0000 \ 0000 \ 0000_{\text{dos}} &= 0_{(10)} \\ 0000 \dots 0000 \ 0000 \ 0000 \ 0001_{\text{dos}} &= 1_{(10)} \\ 0000 \dots 0000 \ 0000 \ 0000 \ 0010_{\text{dos}} &= 2_{(10)} \\ &\dots \\ 0111 \dots 1111 \ 1111 \ 1111 \ 1101_{\text{dos}} &= 2,147,483,645_{(10)} \\ 0111 \dots 1111 \ 1111 \ 1111 \ 1110_{\text{dos}} &= 2,147,483,646_{(10)} \\ 0111 \dots 1111 \ 1111 \ 1111 \ 1111_{\text{dos}} &= 2,147,483,647_{(10)} \\ \hline 1000 \dots 0000 \ 0000 \ 0000 \ 0000_{\text{dos}} &= -2,147,483,648_{(10)} \\ 1000 \dots 0000 \ 0000 \ 0000 \ 0001_{\text{dos}} &= -2,147,483,647_{(10)} \\ 1000 \dots 0000 \ 0000 \ 0000 \ 0010_{\text{dos}} &= -2,147,483,646_{(10)} \\ &\dots \\ 1111 \dots 1111 \ 1111 \ 1111 \ 1101_{\text{dos}} &= -3_{(10)} \\ 1111 \dots 1111 \ 1111 \ 1111 \ 1110_{\text{dos}} &= -2_{(10)} \\ 1111 \dots 1111 \ 1111 \ 1111 \ 1111_{\text{dos}} &= -1_{(10)} \end{aligned}$$

Aritmética en complemento a dos

- ▶ Sumas y
- ▶ Para $n = 5$ bits
- ▶ Sea $X = 5$
 - ▶ En complemento a dos = 00101
- ▶ Sea $Y = 7$
 - ▶ En complemento a uno = 00111
- ▶ ¿ $X + Y$?
$$\begin{array}{rcl} X & = & 00101 \\ Y & = & \underline{00111} + \\ X+Y & = & 01100 \end{array}$$
- ▶ El valor de 01100 en complemento a uno es 12
- ▶ restas de igual forma

Aritmética en complemento a dos

- ▶ Para $n = 5$ bits
- ▶ Sea $X = -5$
 - ▶ En complemento a dos = complemento de 00101 : $11010 + 1 = 11011$
- ▶ Sea $Y = -7$
 - ▶ En complemento a uno = complemento de 00111 : $11000 + 1 = 11001$
- ▶ ¿ $X + Y$?

$$-X = 11011$$

$$-Y = \underline{11001} +$$

$$-(X+Y) = 110100 \quad \text{Se produce un acarreo: se desprecia}$$

- ▶ El valor es 10100 . Su valor en complemento a dos = el valor negativo de su complemento a dos = complemento a uno: $01011 + 1 = 01100 = >-12$

Aritmética en complemento a dos

- ▶ Para $n = 5$ bits
- ▶ Sea $X = 8$
 - ▶ En complemento a dos = 01000
- ▶ Sea $Y = 9$
 - ▶ En complemento a uno = 01001

- ▶ ¿ $X + Y$?

$$X = 01000$$

$$Y = \underline{01001} +$$

$$X+Y = 10001$$

- ▶ Se obtiene un negativo \Rightarrow desbordamiento

Aritmética en complemento a dos

- ▶ Para $n = 5$ bits
- ▶ Sea $X = -8$
 - ▶ En complemento a dos = complemento de 01000 : $10111 + 1 = 11000$
- ▶ Sea $Y = -9$
 - ▶ En complemento a uno = complemento de 01001 : $10110 + 1 = 10111$
- ▶ ¿ $X + Y$?

$$-X = 11000$$

$$-Y = \underline{10111} +$$

$$-(X+Y) = 10111 \quad \text{Se produce un acarreo: se desprecia}$$

- ▶ El valor 01111 , como es positivo \Rightarrow desbordamiento

¿Porqué se desprecia el acarreo?

- ▶ $-X$ se representa como $2^n - X$
- ▶ $-Y$ se representa como $2^n - Y$
- ▶ $-(X + Y)$ se representa como $2^n - (X+Y)$

- ▶ Cuando sumamos directamente $-X - Y$ se obtiene

$$-X = 2^n - X$$

$$-Y = \underline{2^n - Y}$$

$$-(X+Y) = \textcircled{2^n} + 2^n - (X + Y)$$

Se corrige el resultado despreciando el acarreo

$$\Rightarrow 2^n - (X + Y)$$

Desbordamientos en complemento a dos

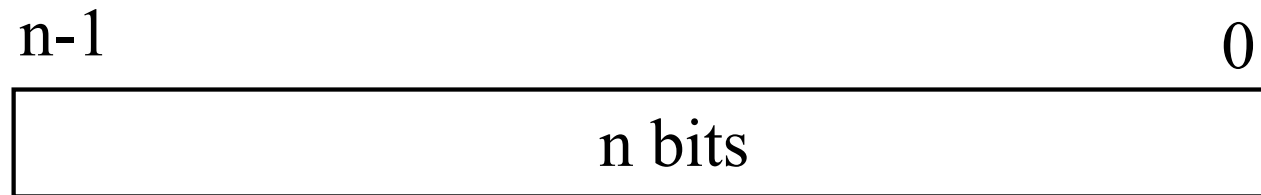
- ▶ Suma de dos negativos \Rightarrow positivo
- ▶ Suma de dos positivos \Rightarrow negativo

Extensión de signo en complemento a dos

- ▶ ¿Cómo pasar de n bits a m bits, siendo $n < m$?
- ▶ Ejemplo:
 - ▶ $n = 4, m = 8$
 - ▶ Si $X = 0110$ con 4 bits $\Rightarrow X = 00000110$ con 8 bits
 - ▶ Si $X = 1011$ con 4 bits $\Rightarrow X = 11111011$ con 8 bits

Representación en Exceso $2^{n-1}-1$ [enteros]

- ▶ El valor X con n bits se representa como $X + 2^{n-1}-1$
- ▶ Se denomina sesgo a la cantidad $2^{n-1}-1$



$$V(X) = \sum_{i=0}^{n-1} 2^i \cdot x_i - (2^{n-1} - 1)$$

- Rango de representación: $[-(2^{n-1} - 1), 2^{n-1}]$
- Resolución: 1 unidad
- No existe ambigüedad con el 0

Ejemplo comparativo (3 bits)

Decimal	Binario Puro	Signo magnitud	Complemento a uno	Complemento a dos	Exceso 3
+7	111	N.D.	N.D.	N.D.	N.D.
+6	110	N.D.	N.D.	N.D.	N.D.
+5	101	N.D.	N.D.	N.D.	N.D.
+4	100	N.D.	N.D.	N.D.	111
+3	011	011	011	011	110
+2	010	010	010	010	101
+1	001	001	001	001	100
+0	000	000	000	000	011
-0	N.D.	100	111	N.D.	N.D.
-1	N.D.	101	110	111	010
-2	N.D.	110	101	110	001
-3	N.D.	111	100	101	000
-4	N.D.	N.D.	N.D.	100	N.D.
-5	N.D.	N.D.	N.D.	N.D.	N.D.
-6	N.D.	N.D.	N.D.	N.D.	N.D.
-7	N.D.	N.D.	N.D.	N.D.	N.D.

Ejercicio

Indique la representación de los siguientes números, razonando brevemente su respuesta:

1. **-32** en complemento a uno con **6 bits**
2. **-32** en complemento a dos con **6 bits**
3. **-10** en signo magnitud con **5 bits**
4. **+14** en complemento a dos con **5 bits**

Ejercicio (solución)

1. Con 6 bits **no es representable** en CI:
[$-2^{6-1}+1, \dots, -0, +0, \dots, 2^{6-1}-1$]
2. CI + 1 -> **100000**
3. Signo=1, magnitud=1010 -> **11010**
4. Positivo -> CI=C2=SM -> **01110**

Ejercicio

- ▶ Usando 5 bits para representarlo, haga las siguientes sumas en complemento a uno:
 - a) $4 + 12$
 - b) $4 - 12$
 - c) $-4 - 12$

Ejercicio (Solución)

► Usando 5 bits en complemento a uno:

a) $4 + 12$

00100

01100

10000 \Rightarrow se obtiene un negativo $\Rightarrow -15 \Rightarrow$ overflow

Ejercicio (Solución)

► Usando 5 bits en complemento a uno:

b) 4 - 12

$$\begin{array}{r} 00100 \\ 10011 \\ \hline 10111 \Rightarrow -8 \end{array}$$

Ejercicio (Solución)

► Usando 5 bits en complemento a uno:

c) $-4 - 12$

11011

10011

101110 \Rightarrow se obtiene un negativo \Rightarrow overflow

Contenidos

1. Introducción

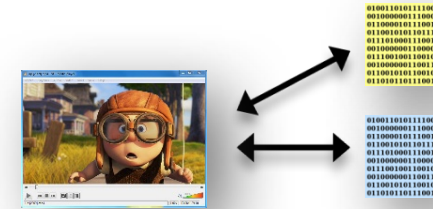
1. Objetivo
2. Motivación
3. Sistemas posicionales

2. Representaciones

1. Alfanuméricas: caracteres y cadenas
2. Numéricas: naturales y enteras
3. **Numéricas: coma fija**
4. Numéricas: coma flotante: estándar IEEE 754

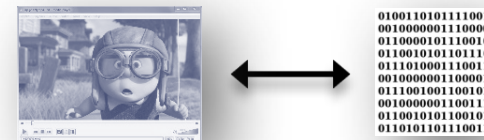
Recordatorio: necesitaremos...

- ▶ Conocer **posibles representaciones**:

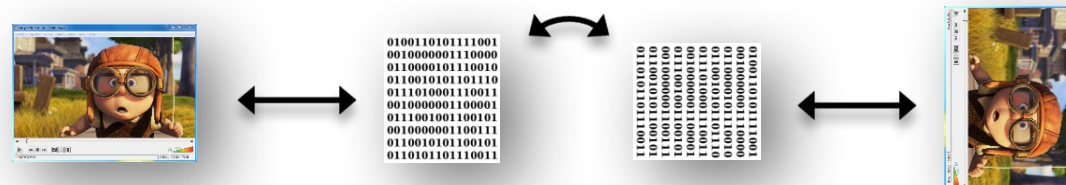


- ▶ Conocer las **características** de las mismas:

- ▶ Limitaciones



- ▶ Conocer **cómo operar** con la representación:



Otras necesidades de representación

- ▶ ¿Cómo representar?

- ▶ Números muy grandes: $30.556.926.000_{(10)}$

- ▶ Números muy pequeños: $0.0000000000529177_{(10)}$

- ▶ Números con decimales: 1,58567

Ejemplo de fallo...

- ▶ Explosión del **Ariane 5** (primer viaje)
 - ▶ Enviado por ESA en junio de 1996
 - ▶ Coste del desarrollo:
10 años y 7000 millones de dólares
 - ▶ Explotó 40 segundos después de despegar, a 3700 metros de altura.
 - ▶ Fallo debido a la pérdida total de la información de altitud:
 - ▶ El software del sistema de referencia inercial realizó la conversión de un valor real en coma flotante de 64 bits a un valor entero de 16 bits. El número a almacenar era mayor de 32767 (el mayor entero con signo de 16 bits) y se produjo un fallo de conversión y una excepción.



Coma fija [racionales]

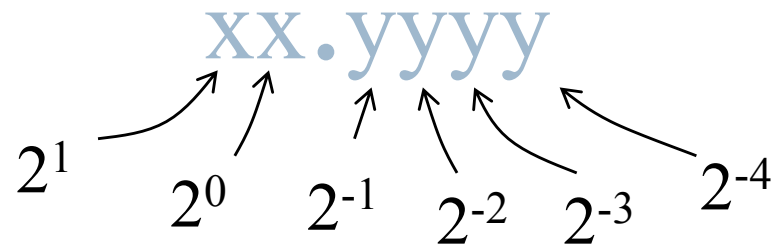
- ▶ Se fija la posición de la coma binaria y se utilizan los pesos asociados a las posiciones decimales

- ▶ Ejemplo:

$$1001.1010 = 2^4 + 2^0 + 2^{-1} + 2^{-3} = 9,625$$

Representación de fracciones con representación binaria en coma fija

- ▶ Ejemplo de representación con 6 bits:



$$10,1010_{(2)} = 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-3} = 2.625_{10}$$

Asumiendo esta coma fija, el rango sería:

- 0 a 3.9375 (casi 4)

Potencias negativas

i	2⁻ⁱ	
0	1.0	1
1	0.5	1/2
2	0.25	1/4
3	0.125	1/8
4	0.0625	1/16
5	0.03125	1/32
6	0.015625	
7	0.0078125	
8	0.00390625	
9	0.001953125	
10	0.0009765625	

Contenidos

1. Introducción

1. Objetivo
2. Motivación
3. Sistemas posicionales

2. Representaciones

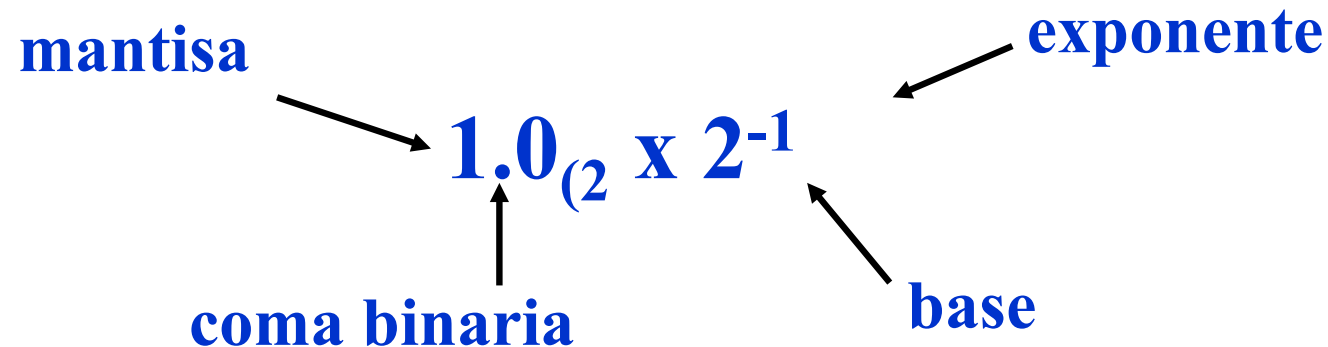
1. Alfanuméricas: caracteres y cadenas
2. Numéricas: naturales y enteras
3. Numéricas: coma fija
4. **Numéricas: coma flotante: estándar IEEE 754**

Notación científica decimal

The diagram shows the scientific notation 9.12×10^{25} . The number 9.12 is underlined and labeled 'mantisa' with an upward arrow. The '10' is labeled 'base' with a gray arrow pointing to it. The '25' is labeled 'exponente' with an upward arrow. The 'x' is between the mantisa and the base.

- ▶ Cada número lleva asociado una mantisa y un exponente
- ▶ Notación científica decimal usada: notación normalizada
 - ▶ Solo un dígito distinto de 0 a la izquierda del punto
- ▶ Se adapta el número al **orden de magnitud** del valor a representar, trasladando la *coma decimal* mediante el exponente

Notación científica en binario



- ▶ Forma normalizada: Un 1 (solo un dígito) a la izquierda de la coma
 - ▶ Normalizada: 1.0001×2^{-9}
 - ▶ No normalizada: 0.0011×2^{-8} , 10.0×2^{-10}

Estándar IEEE 754 [racionales]



- ▶ Estándar para coma flotante usado en la mayoría de los ordenadores.
- ▶ **Características** (salvo casos especiales):
 - ▶ Exponente: en exceso con sesgo $2^{\text{num_bits_exponente} - 1} - 1$
 - ▶ Mantisa: signo-magnitud, normalizada, con bit implícito
- ▶ Diferentes **formatos**:
 - ▶ **Precisión simple**: 32 bits (signo: 1, exponente: 8 y mantisa: 23)
 - ▶ **Doble precisión**: 64 bits (signo: 1, exponente: 11 y mantisa: 52)
 - ▶ **Cuádruple precisión**: 128 bits (signo: 1, exponente: 15 y mantisa: 112)

Números normalizados

- ▶ En este estándar los números a representar tienen que estar **normalizados**. Un número normalizado es de la forma:

- ▶ $1,bbbbbb \times 2^e$

- ▶ mantisa: $1,bbbbbb$ (siendo $b = 0, 1$)
- ▶ 2 es la base del exponente
- ▶ e es el exponente

Normalización y bit implícito

▶ Normalización

Para normalizar la mantisa se ajusta el exponente para que el bit más significativo de la mantisa sea 1

▶ Ejemplo: $100100000000000000000000 \times 2^3$ (ya está normalizado)

▶ Ejemplo: ~~000~~ $100000000010101 \times 2^3$ (no lo está)
 $100000000010101000 \times 2^0$ (ahora sí)

▶ Bit implícito

Una vez normalizado, dado que el bit más significativo es 1, **no** se almacena para dejar espacio para un bit más (aumenta la precisión)

▶ Así se puede representar mantisas con un bit más

Estándar IEEE 754 de precisión simple [racionales]



S es el signo (1 bit)

E es el exponente (8 bits)

M es la mantisa (23 bits)

- ▶ El valor se calcula con la siguiente expresión (salvo casos especiales):

$$\mathbf{N = (-1)^S \times 2^{E-127} \times 1.M}$$

donde:

S = 0 indica número positivo, S = 1 indica número negativo

$0 < E < 255$ (E=0 y E=255 indican casos especiales)

0000000000000000000000000000 ≤ M ≤ 11111111111111111111111111111111

Estándar IEEE 754 de precisión simple [racionales]

► Existencia de casos especiales:

$$(-1)^s * 0.\text{mantisa} * 2^{-126}$$

Exponente	Mantisa	Valor especial
0 (0000 0000)	0	+/- 0 (según signo)
0 (0000 0000)	No cero	Número NO normalizado
255 (1111 1111)	No cero	NaN (0/0,...)
255 (1111 1111)	0	+/-infinito (según signo)
1-254	Cualquiera	Número normalizado (no especial)

$$(-1)^s * 1.\text{mantisa} * 2^{\text{exponente}-127}$$

Ejemplos (incluyen casos especiales)

S	E	M	N
1	00000000	000000000000000000000000	-0 (Excepción 0) E=0 y M=0.
1	01111111	000000000000000000000000	$-2^0 \times 1.0_2 = -1$
0	10000001	111000000000000000000000	$+2^2 \times 1.111_2 = +2^2 \times (2^0+2^{-1}+2^{-2}+2^{-3}) = +7.5$
0	11111111	000000000000000000000000	∞ (Excepción ∞) E=255 y M=0
0	11111111	100000000000000000000001	NaN (Not a Number, como la raíz cuadrada de un número negativo) E=255 y M \neq 0.

Ejercicio

- a) Calcular el valor correspondiente al número
0 10000011 110000000000000000000000
dado en coma flotante según norma 754 de simple precisión

Ejercicio (solución)

- a) Calcular el valor correspondiente al número
0 10000011 110000000000000000000000
dado en coma flotante según norma 754 de simple precisión

- a) Bit de signo: $0 \Rightarrow (-1)^0 = +1$
b) Exponente: $10000011_2 = 131_{10} \Rightarrow E - 127 = 131 - 127 = 4$
c) Mantisa: $110000000000000000000000 \Rightarrow 1 \times 2^{-1} + 1 \times 2^{-2} = 0,75$

Por tanto el valor decimal del n° es $+1 \times 2^4 \times 1,75 = +28$

Ejercicio

- b) Expresar según norma IEEE 754 de simple precisión el n° -9

Ejercicio (solución)

b) Expresar según norma IEEE 754 de simple precisión el nº -9

$$-9_{10} = -1001_2 = -1001_2 \times 2^0 = -1,001_2 \times 2^3 \text{ (mantisa normalizada)}$$

a) Bit de signo: negativo \Rightarrow S=1

b) Exponente: $3+127$ (exceso) = 130 \Rightarrow 10000010

c) Mantisa: 1,001 (bit impl.) \Rightarrow 001000000000000000000000

Por tanto $-9 = 1 \ 10000010 \ 001000000000000000000000$

Estándar IEEE 754 de precisión simple [racionales]

▶ Rango de magnitudes representables (sin considerar el signo):

▶ Menor normalizado:

$$2^{-127} \times 1.000000000000000000000000_2$$

▶ Mayor normalizado:

$$2^{254-127} \times 1.111111111111111111111111_2$$

▶ Menor no normalizado:

$$2^{-126} \times 0.000000000000000000000001_2$$

▶ Mayor no normalizado:

$$2^{-126} \times 0.111111111111111111111111_2$$

$$(-1)^s * 0.\text{mantisa} * 2^{-126}$$

Exponente	Mantisa	Valor especial
0	≠ 0	No normalizado
1-254	cualquiera	normalizado

$$(-1)^s * 1.\text{mantisa} * 2^{\text{exponente}-127}$$

Estándar IEEE 754 de precisión simple [racionales]

▶ Rango de magnitudes representables (sin considerar el signo):

▶ Menor normalizado:

$$2^{-127} \times 1.000000000000000000000000_2 = 2^{-126}$$

▶ Mayor normalizado:

$$2^{254-127} \times 1.111111111111111111111111_2 = 2^{127} \times (2 - 2^{-23}) = 2^{128} \times (1 - 2^{-24})$$

▶ Menor no normalizado:

$$2^{-126} \times 0.000000000000000000000001_2 = 2^{-149}$$

▶ Mayor no normalizado:

$$2^{-126} \times 0.111111111111111111111111_2 = 2^{-126} \times (1 - 2^{-23})$$

Truco:

$$\begin{array}{rcl}
 & 1.111111111111111111111111_2 & = X \\
 + & 0.000000000000000000000001_2 & = 2^{-23} \\
 \hline
 & 10.000000000000000000000000_2 & = 2 \\
 & & X = 2 - 2^{-23}
 \end{array}$$

Estándar IEEE 754 de precisión simple [racionales]

▶ Rango de magnitudes representables (sin considerar el signo):

▶ Menor normalizado:

$$2^{-127} \times 1.000000000000000000000000_2 = 2^{-126} = 2^{-127} \times 0.5$$

▶ Mayor normalizado:

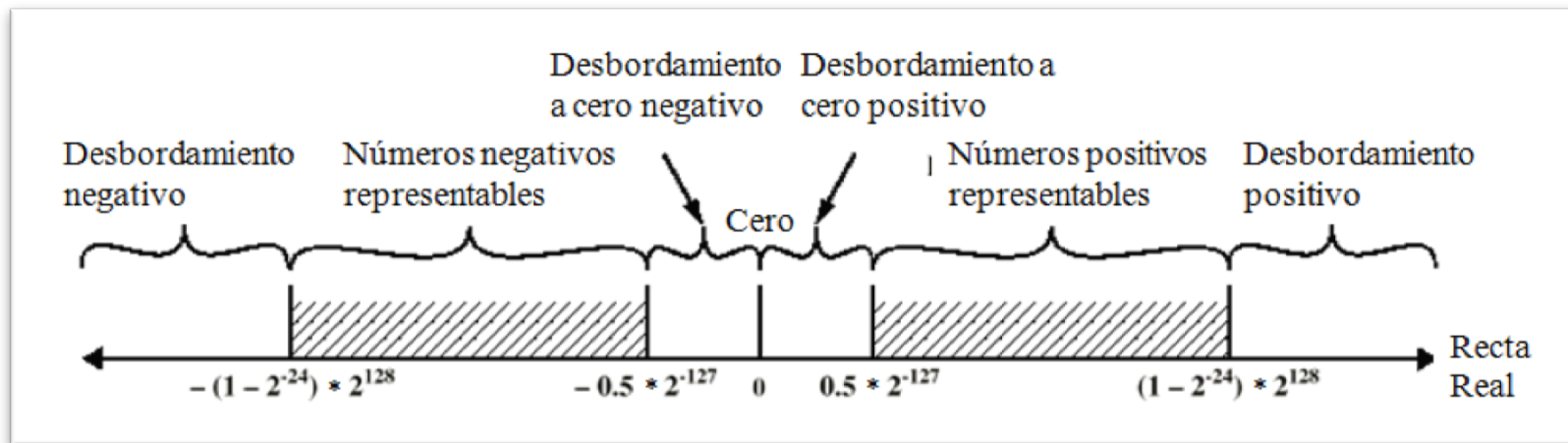
$$2^{254-127} \times 1.111111111111111111111111_2 = 2^{127} \times (2 - 2^{-23}) = 2^{128} \times (1 - 2^{-24})$$

▶ Menor no normalizado:

$$2^{-126} \times 0.000000000000000000000000_2 = 2^{-149}$$

▶ Mayor no normalizado:

$$2^{-126} \times 0.111111111111111111111111_2 = 2^{-126} \times (1 - 2^{-23})$$



Ejercicio

- ▶ ¿Cuántos números de *floats* (coma flotante de simple precisión) hay entre el 1 y el 2 (no incluido)?

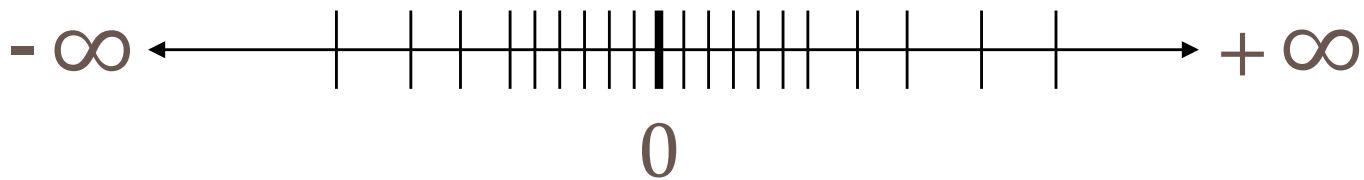
- ▶ ¿Cuántos números de *floats* (coma flotante de simple precisión) hay entre el 2 y el 3 (no incluido)?

Ejercicio (solución)

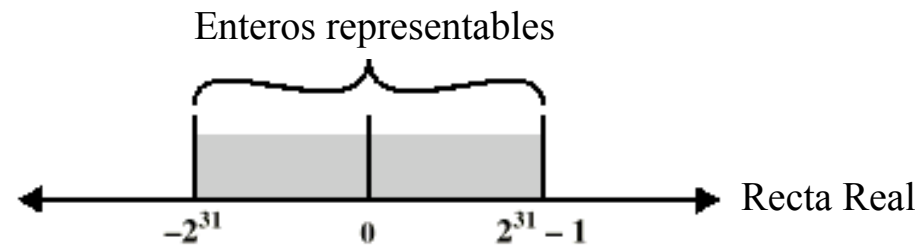
- ▶ ¿Cuántos números de *floats* (coma flotante de simple precisión) hay entre el 1 y el 2 (no incluido)?
 - ▶ $1 = 1,00000000000000000000000000000000 \times 2^0$
 - ▶ $2 = 1,00000000000000000000000000000000 \times 2^1$
 - ▶ Entre 1 y 2 hay 2^{23} números
- ▶ ¿Cuántos números de *floats* (coma flotante de simple precisión) hay entre el 2 y el 3 (no incluido)?
 - ▶ $2 = 1,00000000000000000000000000000000 \times 2^1$
 - ▶ $3 = 1,10000000000000000000000000000000 \times 2^1$
 - ▶ Entre 2 y 3 hay 2^{22} números

Números representables

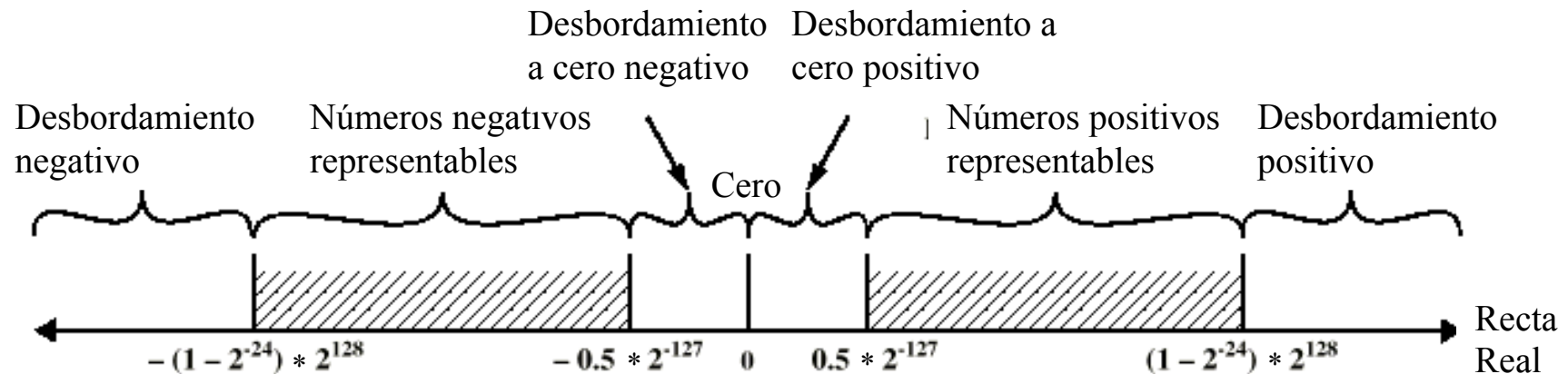
- ▶ Resolución variable:
Más denso cerca de cero, menos hacia el infinito



Números representables



(a) Enteros en complemento a dos

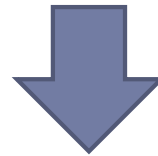


(b) Números en coma flotante

Ejemplo 1 imprecisión

0,4 →

0	01111101	10011001100110011001101
---	----------	-------------------------



3.9999998×10^{-1}

0,1 →

0	01111011	10011001100110011001100
---	----------	-------------------------



9.9999994×10^{-2}

Ejemplo 2

imprecisión

- ▶ ¿Cómo realiza C una división?

t2.c

```
#include <stdio.h>

int main ( )
{
    float a ;

    a = 3.0/7.0 ;
    if (a == 3.0/7.0)
        printf("Igual\n") ;
    else printf("No Igual\n") ;
    return (0) ;
}
```

Ejemplo 2

imprecisión

- ▶ ¿Cómo realiza C una división?

```
                                t2.c
#include <stdio.h>

int main ( )
{
    float a ;

    a = 3.0/7.0 ;
    if (a == 3.0/7.0)
        printf("Igual\n") ;
    else printf("No Igual\n") ;
    return (0) ;
}
```

```
$ gcc -o t2 t2.c
$ ./t2
No Igual
```


Ejemplo 2

imprecisión

- ▶ ¿Cómo realiza C una división?

```
                                t2.c
#include <stdio.h>

int main ( )
{
    float a ;
    a = 3.0/7.0 ;
    if (a == 3.0/7.0)
        printf("Igual\n") ;
    else printf("No Igual\n") ;
    return (0) ;
}
```

float

double

```
$ gcc -o t2 t2.c
$ ./t2
No Igual
```

Ejemplo 3

imprecisión

- ▶ La propiedad asociativa no siempre se cumple
¿ $a + (b + c) = (a + b) + c$?

t1.c

```
#include <stdio.h>

int main ( )
{
    float x, y, z ;

    x = 10e30;  y = -10e30;  z = 1;
    printf("(x+y)+z = %f\n", (x+y)+z);
    printf("x+(y+z) = %f\n", x+(y+z));

    return (0);
}
```

Ejemplo 3

imprecisión

- ▶ La propiedad asociativa no siempre se cumple
¿ $a + (b + c) = (a + b) + c$?

```
                                t1.c
#include <stdio.h>

int main ( )
{
    float x, y, z ;

    x = 10e30;  y = -10e30;  z = 1;
    printf("(x+y)+z = %f\n", (x+y)+z);
    printf("x+(y+z) = %f\n", x+(y+z));

    return (0);
}
```

```
$ gcc -o t1 t1.c
$ ./t1
(x+y)+z = 1.000000
x+(y+z) = 0.000000
```

Redondeo

- ▶ El redondeo elimina cifras menos significativas de un número para obtener un valor aproximado.
- ▶ **Tipos** de redondeo:
 - ▶ Redondeo **hacia $+\infty$**
 - ▶ Redondeo “hacia arriba”: $2.001 \rightarrow 3$, $-2.001 \rightarrow -2$
 - ▶ Redondeo **hacia $-\infty$**
 - ▶ Redondea “hacia abajo”: $1.999 \rightarrow 1$, $-1.999 \rightarrow -2$
 - ▶ **Truncar**
 - ▶ Descarta los últimos bits: $1.299 \rightarrow 1.2$
 - ▶ Redondeo **al más cercano**
 - ▶ $2.4 \rightarrow 2$, $2.6 \rightarrow 3$, $-1.4 \rightarrow -1$

Redondeo

- ▶ El redondeo **supone ir perdiendo precisión.**
- ▶ El redondeo **ocurre:**
 - ▶ **Al pasar a una representación con menos representables:**
 - ▶ Ej.: Un valor de doble a simple precisión
 - ▶ Ej.: Un valor en coma flotante a entero
 - ▶ **Al realizar operaciones aritméticas:**
 - ▶ Ej.: Después de sumar dos números en coma flotante (al usar dígitos de guarda)

Dígitos de guarda

- ▶ Se utilizan **dígitos de guarda** para mejorar la precisión: internamente se usan dígitos adicionales para operar.
- ▶ Ejemplo: $2,65 \times 10^0 + 2,34 \times 10^2$

	SIN dígitos de guarda	CON dígitos de guarda
1.- igualar exponentes	$0,02 \times 10^2$ $+ 2,34 \times 10^2$	$0,0265 \times 10^2$ $+ 2,3400 \times 10^2$
2.- sumar	$2,36 \times 10^2$	$2,3665 \times 10^2$
3.- redondear	$2,36 \times 10^2$	$2,37 \times 10^2$

Operaciones en coma flotante

- ▶ **Sumar**

- ▶ **Restar**

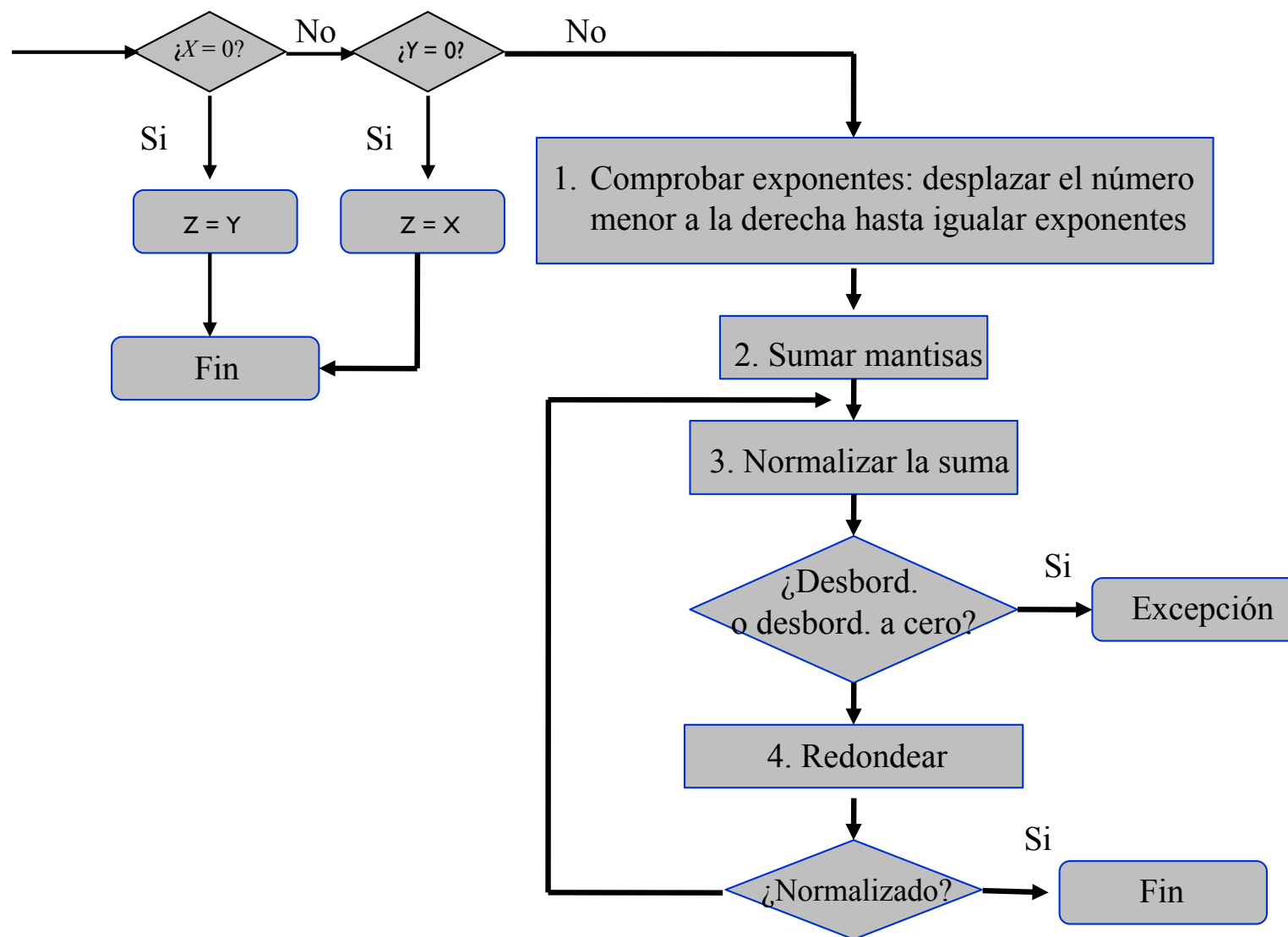
1. Comprobar valores cero.
2. Igualar exponentes (desplazar número menor a la derecha).
3. Sumar/restar las mantisas.
4. Normalizar el resultado.

- ▶ **Multiplicar**

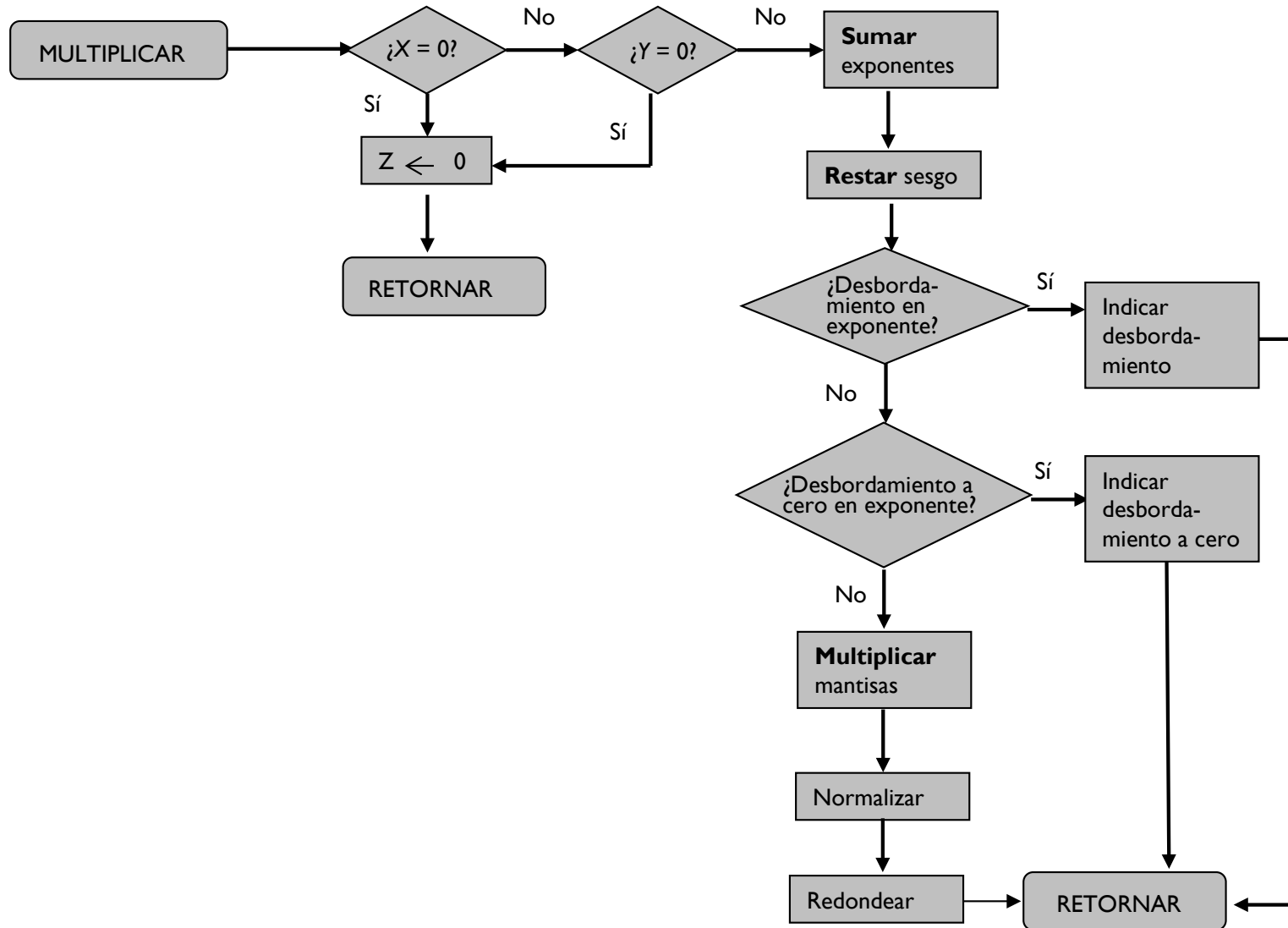
- ▶ **Dividir**

1. Comprobar valores cero.
2. Sumar/restar exponentes.
3. Multiplicar/dividir mantisas (teniendo en cuenta el signo).
4. Normalizar el resultado.
5. Redondear el resultado.

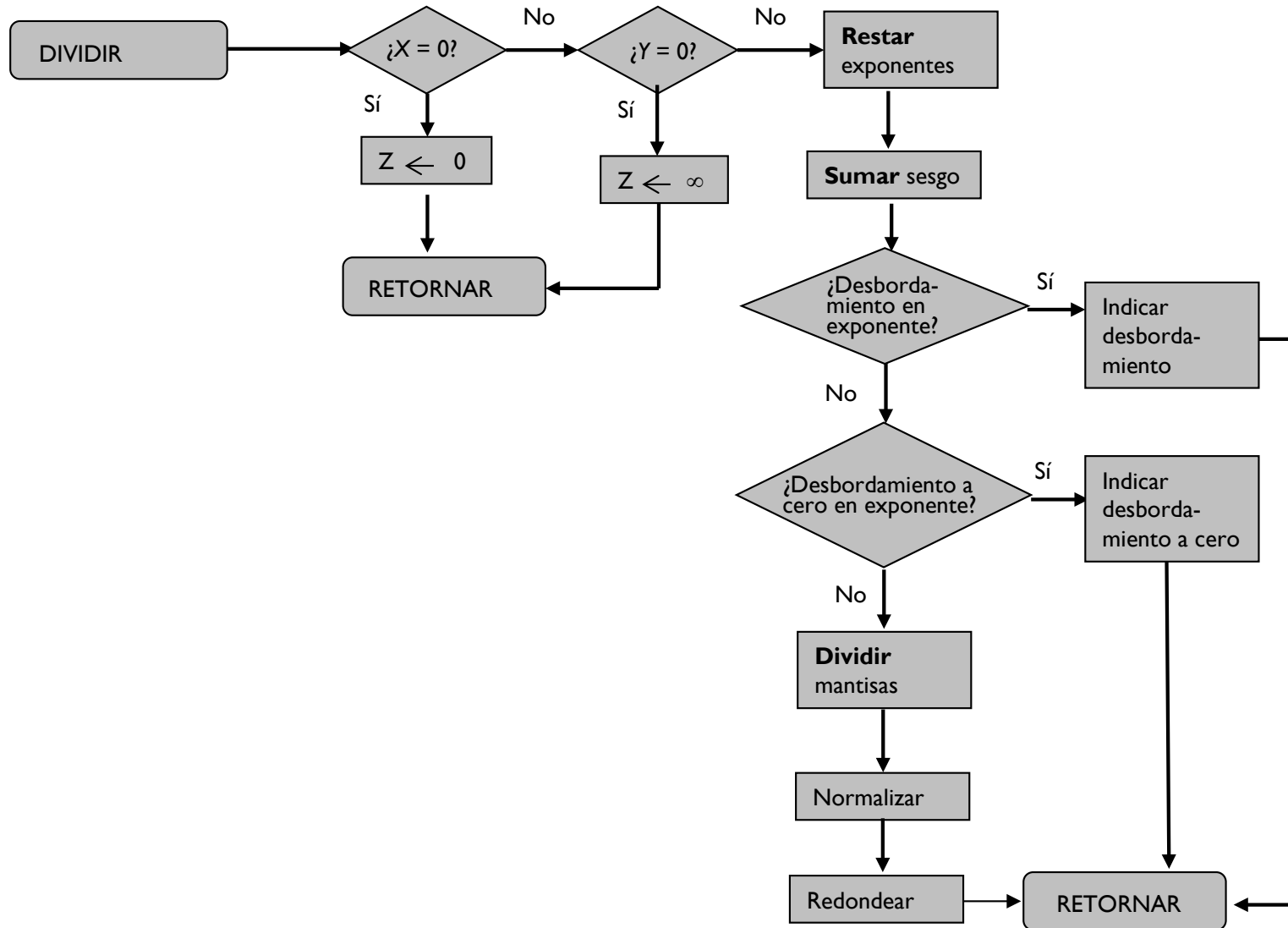
Suma y resta: $Z=X+Y$ y $Z=X-Y$



Multiplicación: $Z=X*Y$



División: $Z=X/Y$



Ejercicio

- ▶ Usando el formato IEEE 754, sumar 7,5 y 1,5 paso a paso

Solución

1) $7,5 + 1,5 =$

Pasar a binario

2) $1,111 * 2^2 + 1,1 * 2^0 =$

Igualar
exponentes

3) $1,111 * 2^2 + 0,011 * 2^2 =$

Sumar

4) $10,010 * 2^2 =$

5) $1,0010 * 2^3$

Ajustar
exponentes

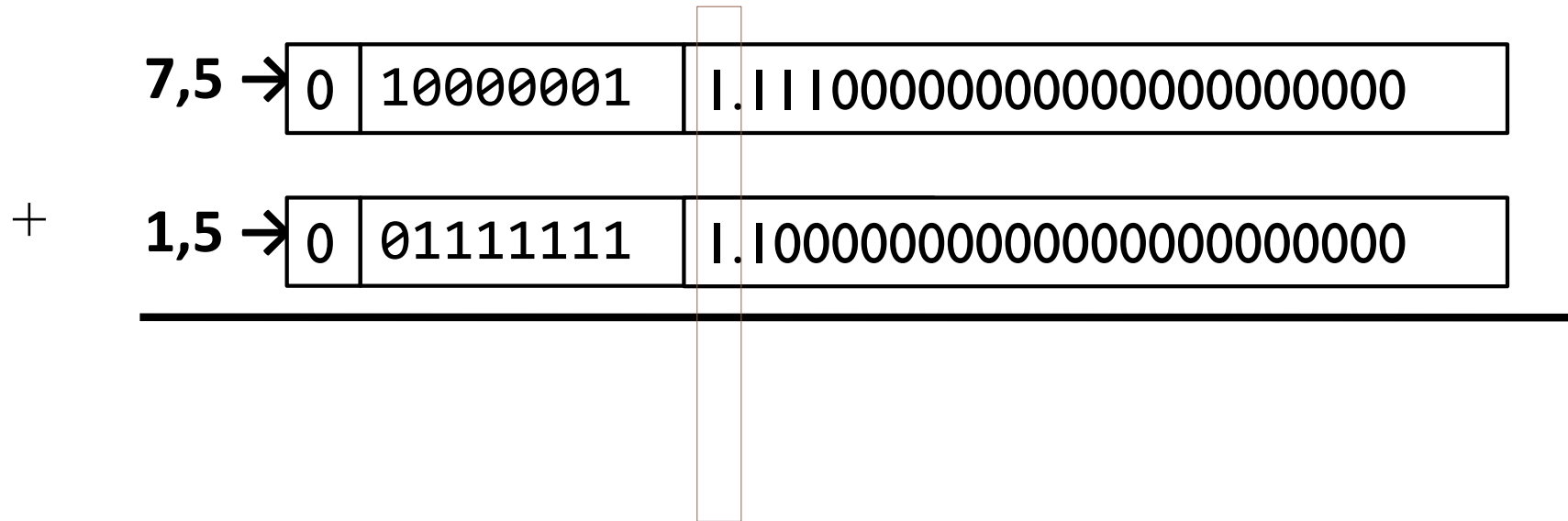
Solución

► Representación de los números

$$\begin{array}{r} 7,5 \rightarrow 0 \ 1000001 \ 111000000000000000000000 \\ + \ 1,5 \rightarrow 0 \ 01111111 \ 100000000000000000000000 \end{array}$$

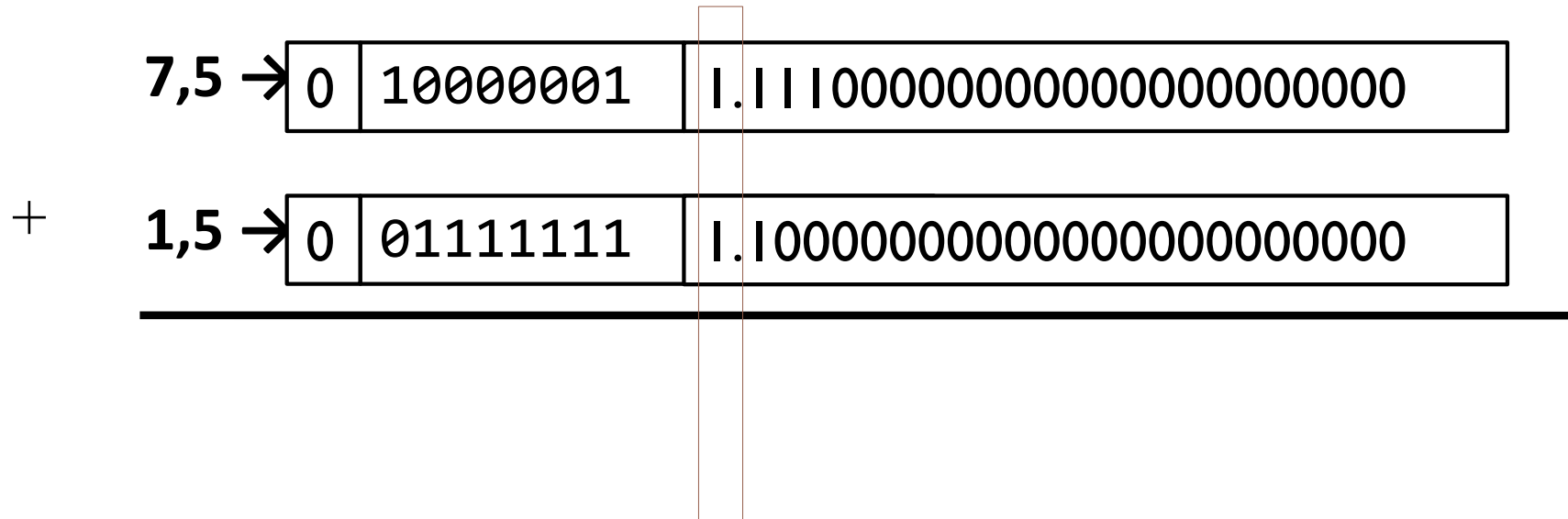
Solución

- ▶ Se separa exponentes y mantisas y se añade el bit implícito



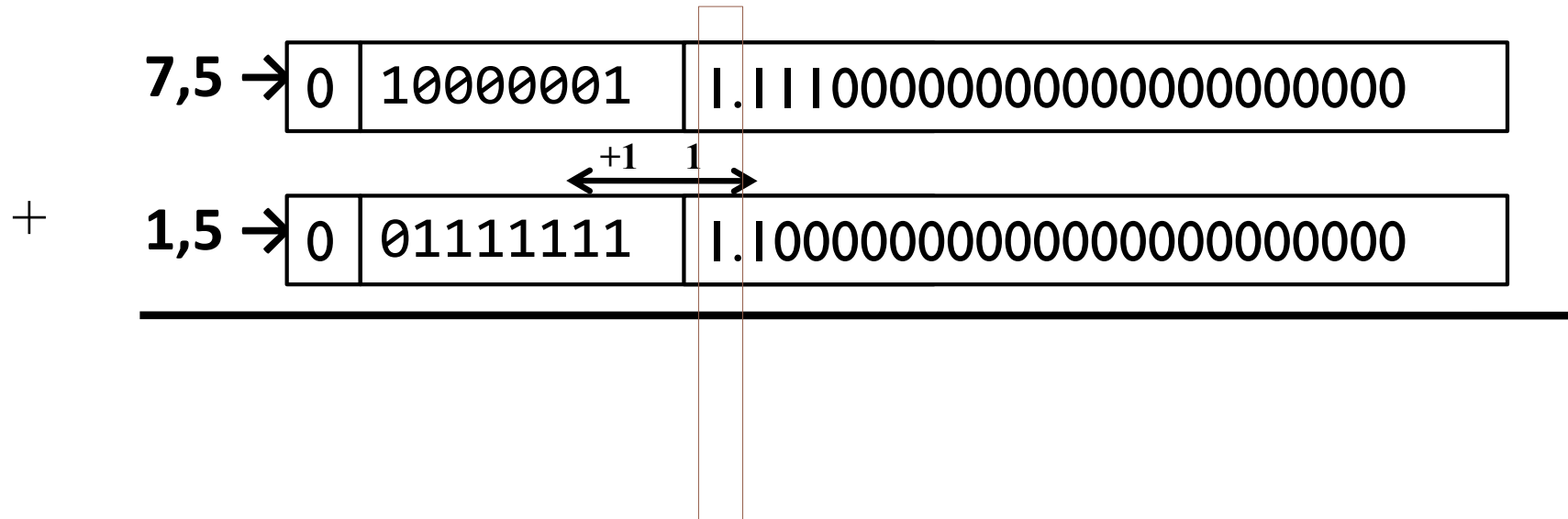
Solución

- ▶ Igualar exponentes



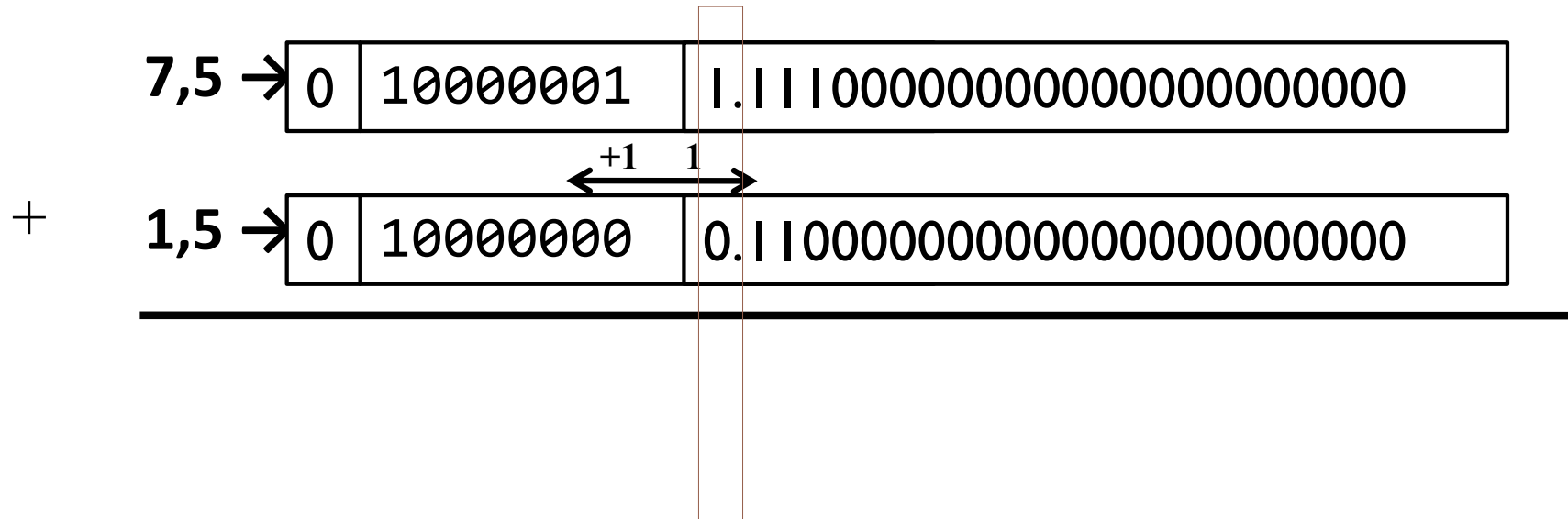
Solución

- ▶ Igualar exponentes



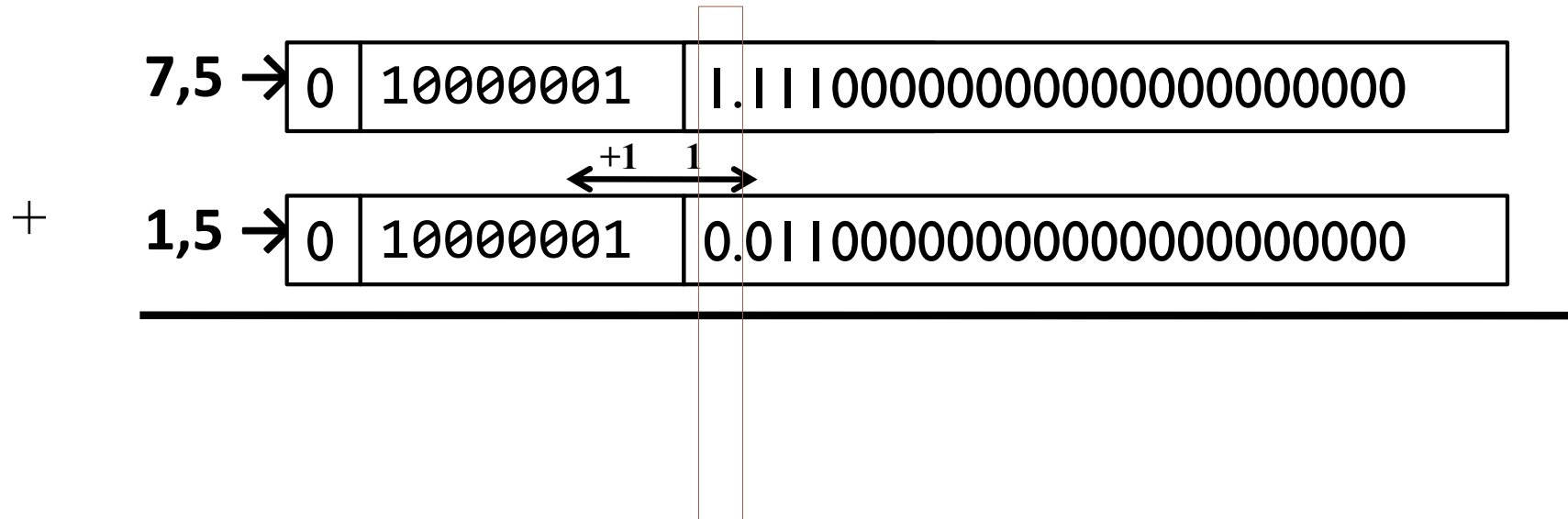
Solución

- ▶ Igualar exponentes



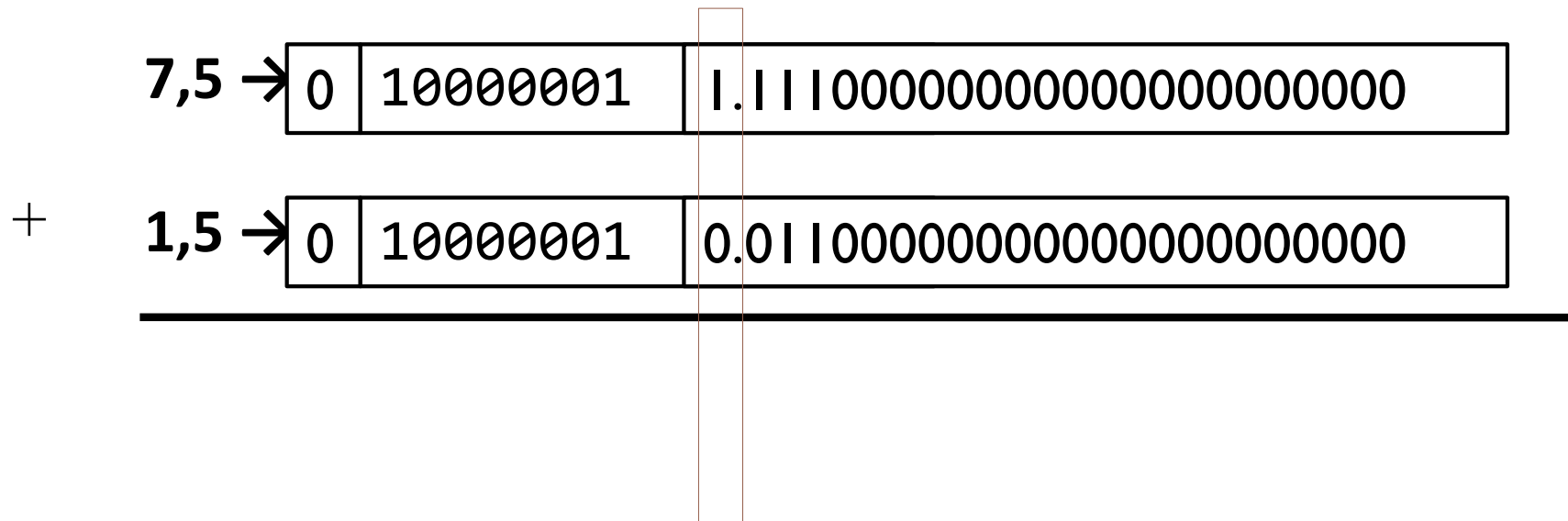
Solución

- ▶ Igualar exponentes



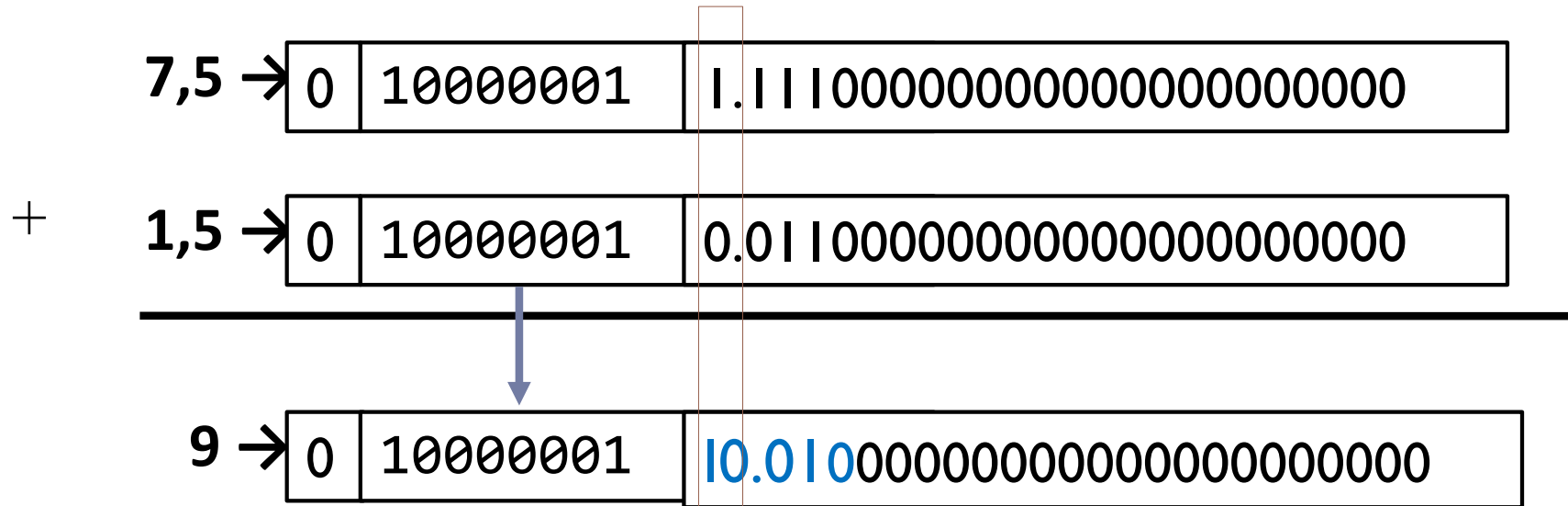
Solución

► Sumar mantisas



Solución

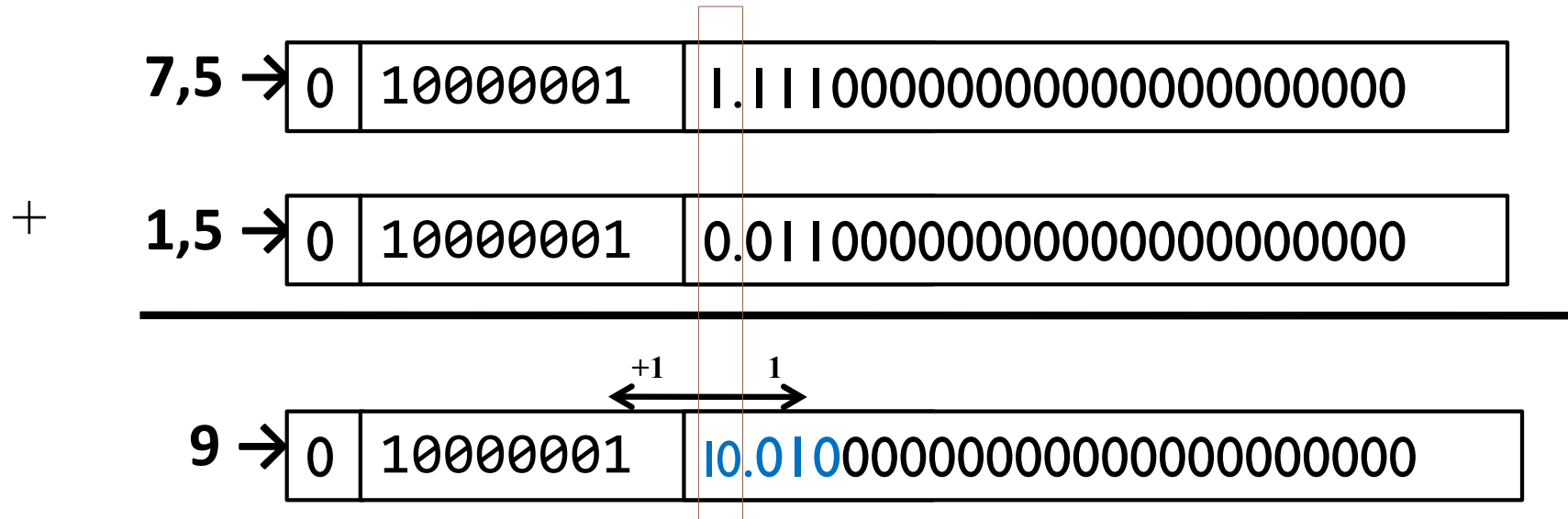
► Normalizar el resultado



Se produce un acarreo, mantisa no normalizada

Solución

► Normalizar el resultado



Solución

$$\begin{array}{r} 7,5 \rightarrow 0 \quad 10000001 \quad | \quad 1.1111000000000000000000000000 \\ + \quad 1,5 \rightarrow 0 \quad 10000001 \quad | \quad 0.0111000000000000000000000000 \\ \hline 9 \rightarrow 0 \quad 10000010 \quad | \quad 1.0010000000000000000000000000 \end{array}$$

Solución

- ▶ Se almacena el resultado eliminando el bit implícito

9 → 0 10000010 001000000000000000000000

Ejercicio

- ▶ Usando el formato IEEE 754, calcular $9 - 7.5$ paso a paso

Solución

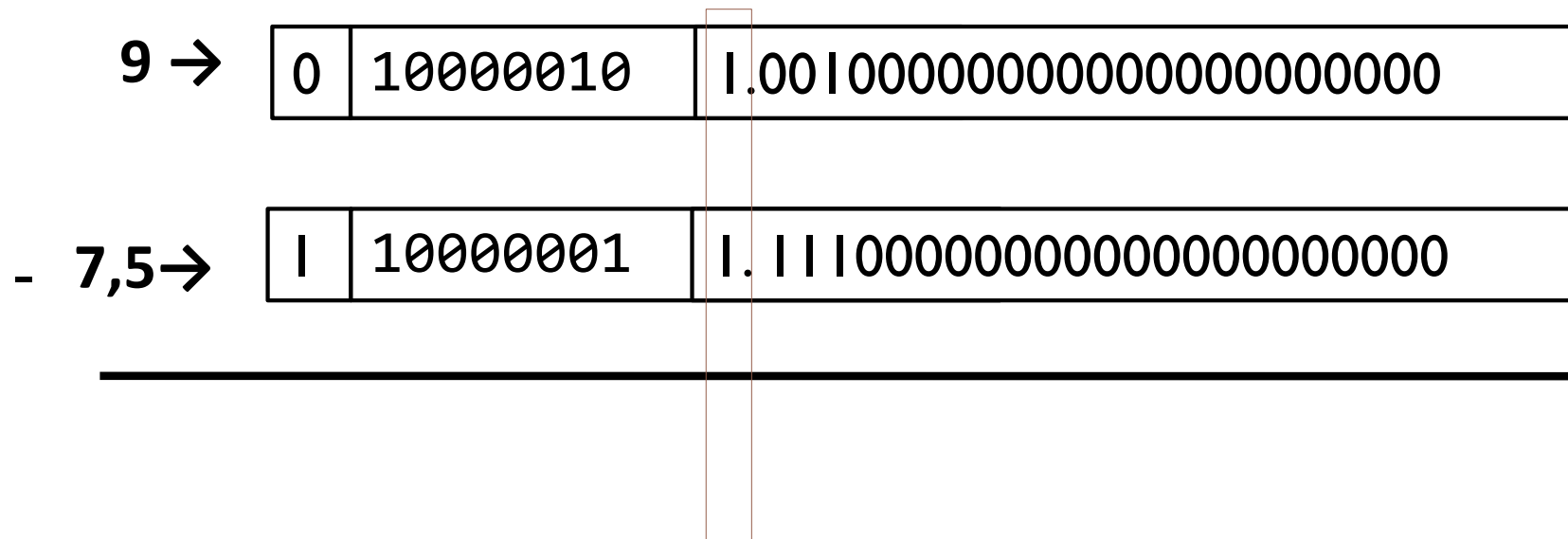
► Representación de los números

9 → 0 10000010 001000000000000000000000

- 7,5 → 1 10000001 111000000000000000000000

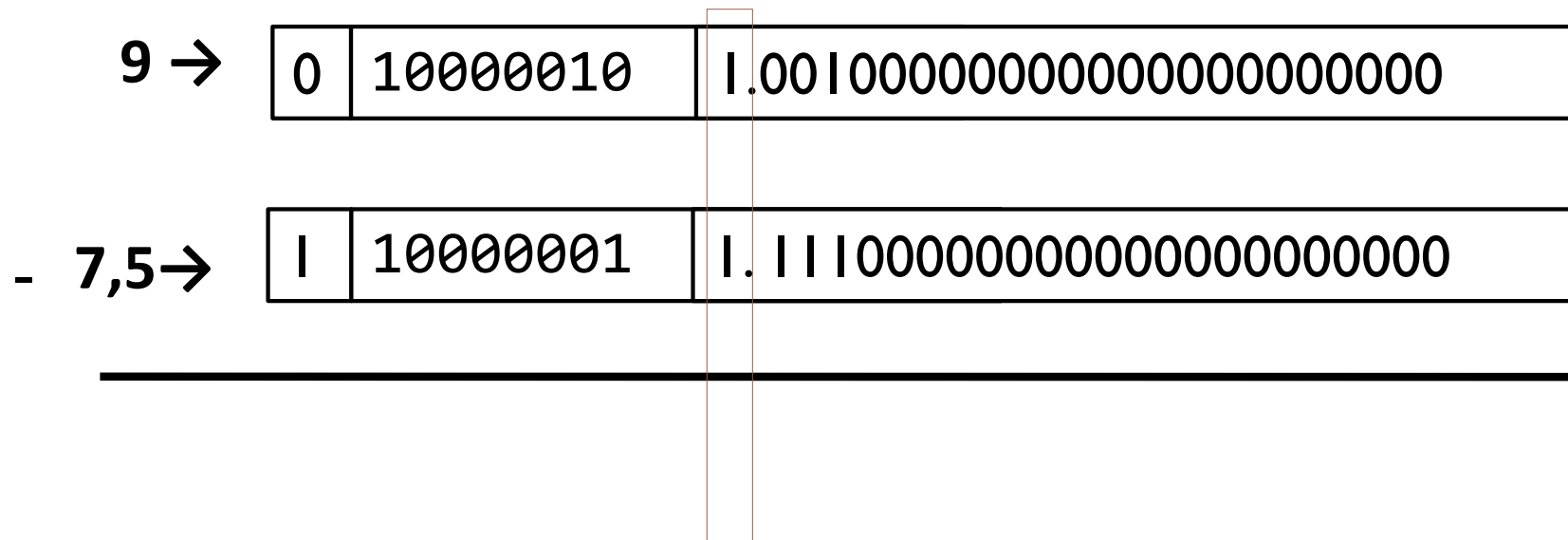
Solución

- ▶ Se separan exponentes y mantisas y se añade bit implícito



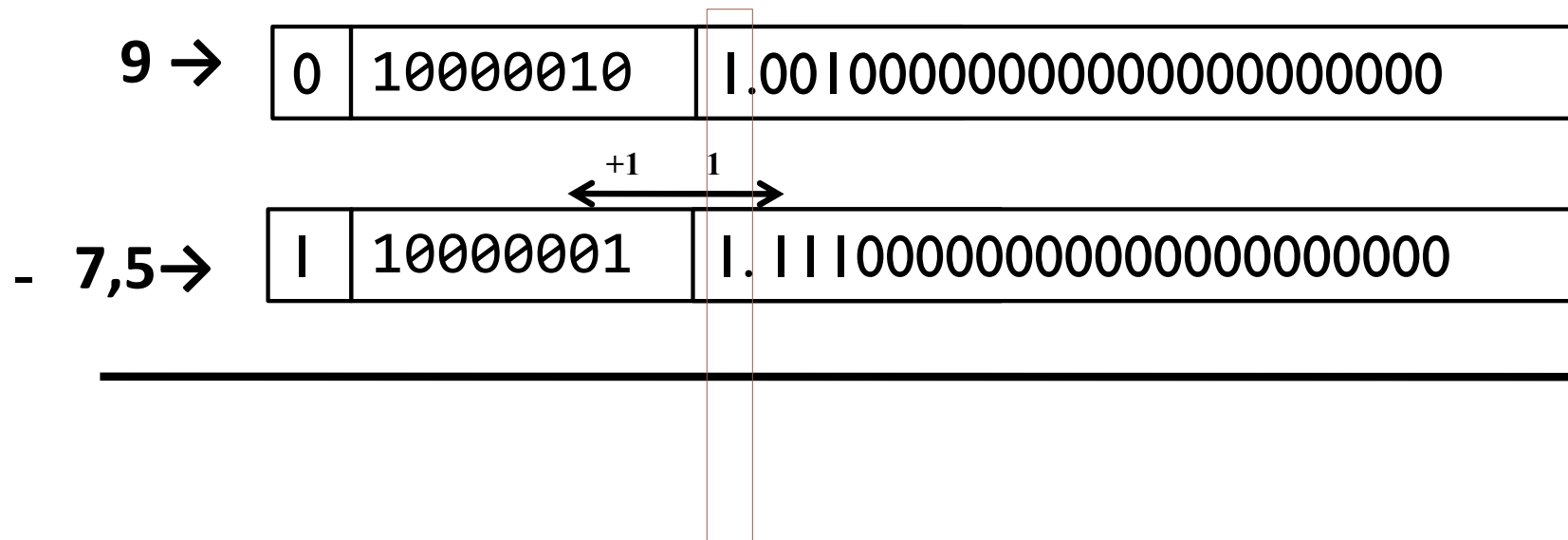
Solución

► Igualar exponentes



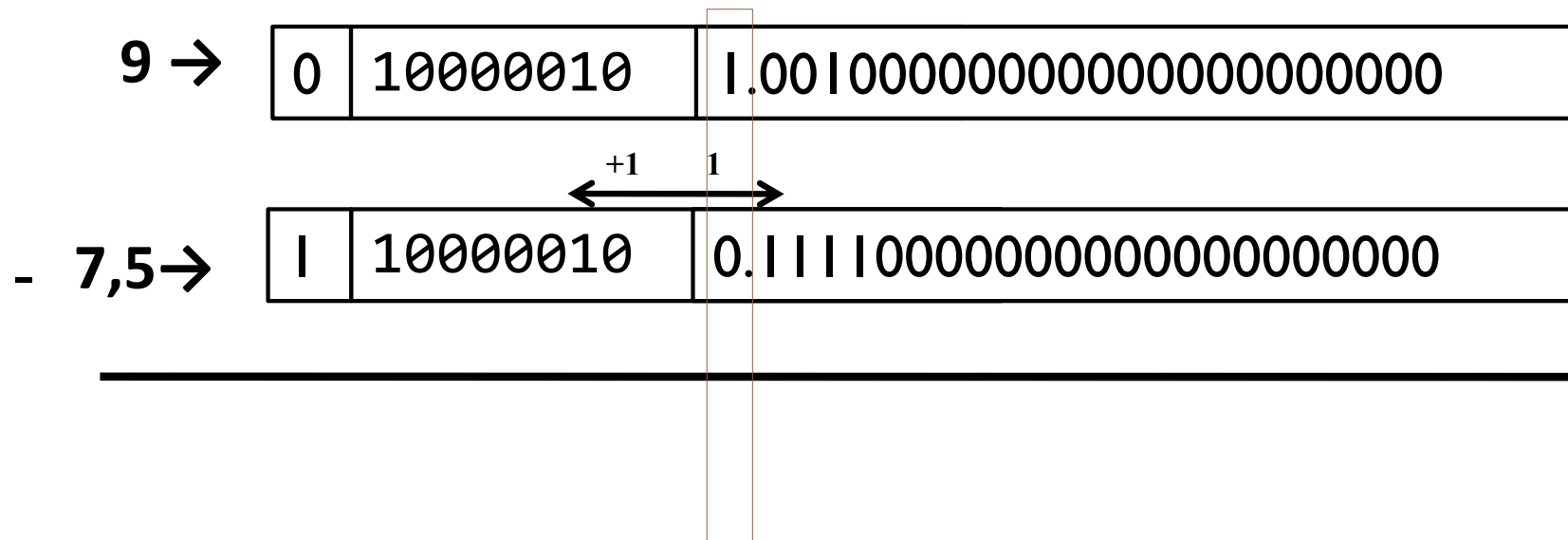
Solución

► Igualar exponentes



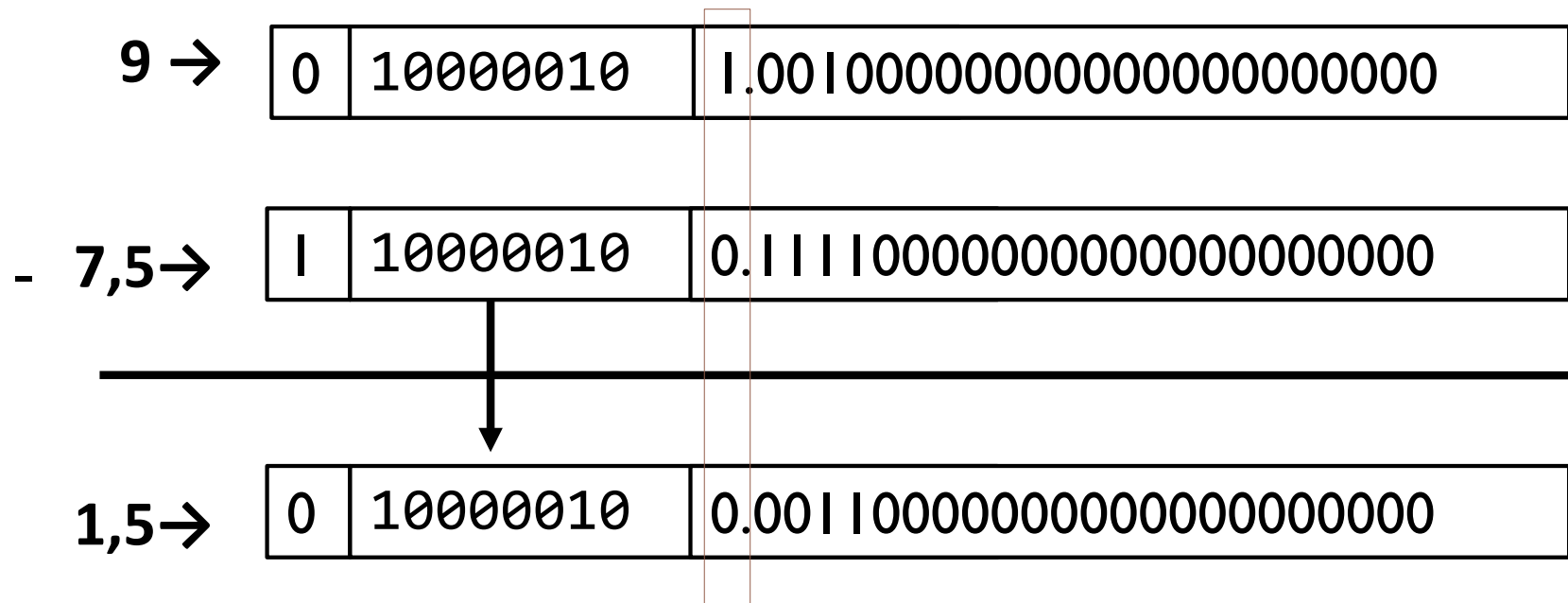
Solución

► Igualar exponentes



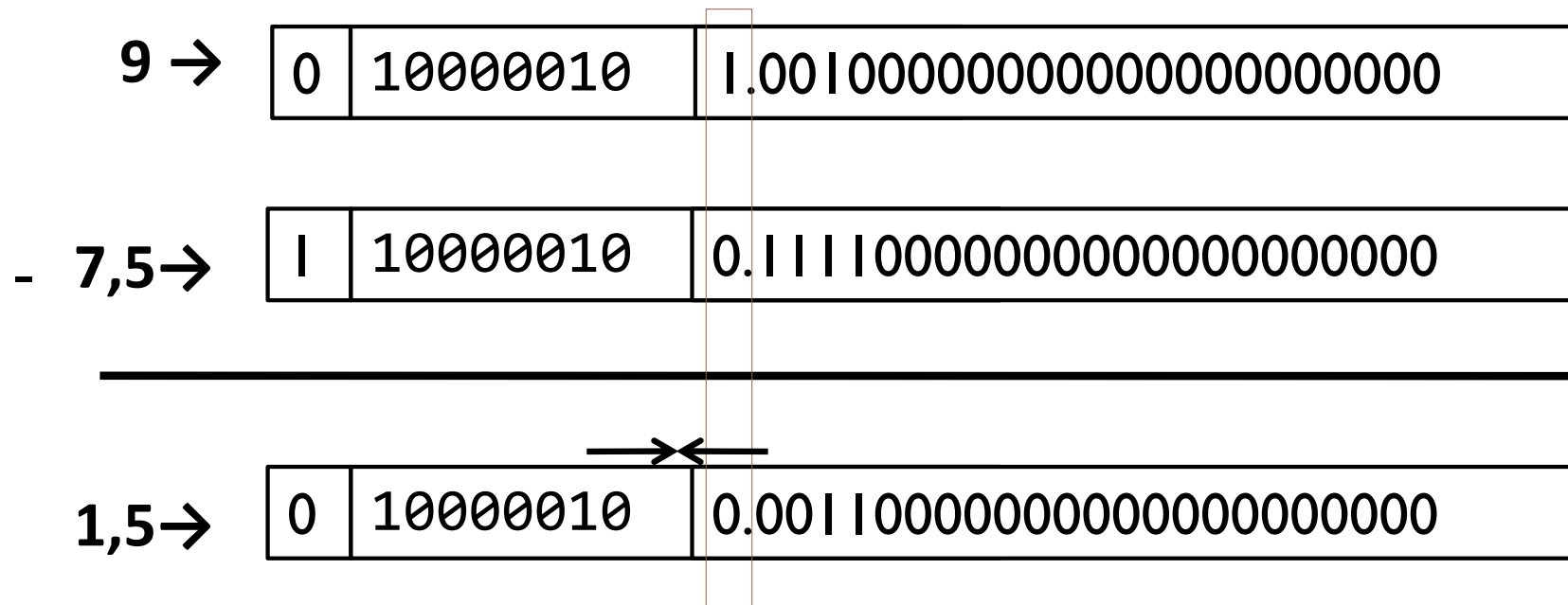
Solución

► Resta



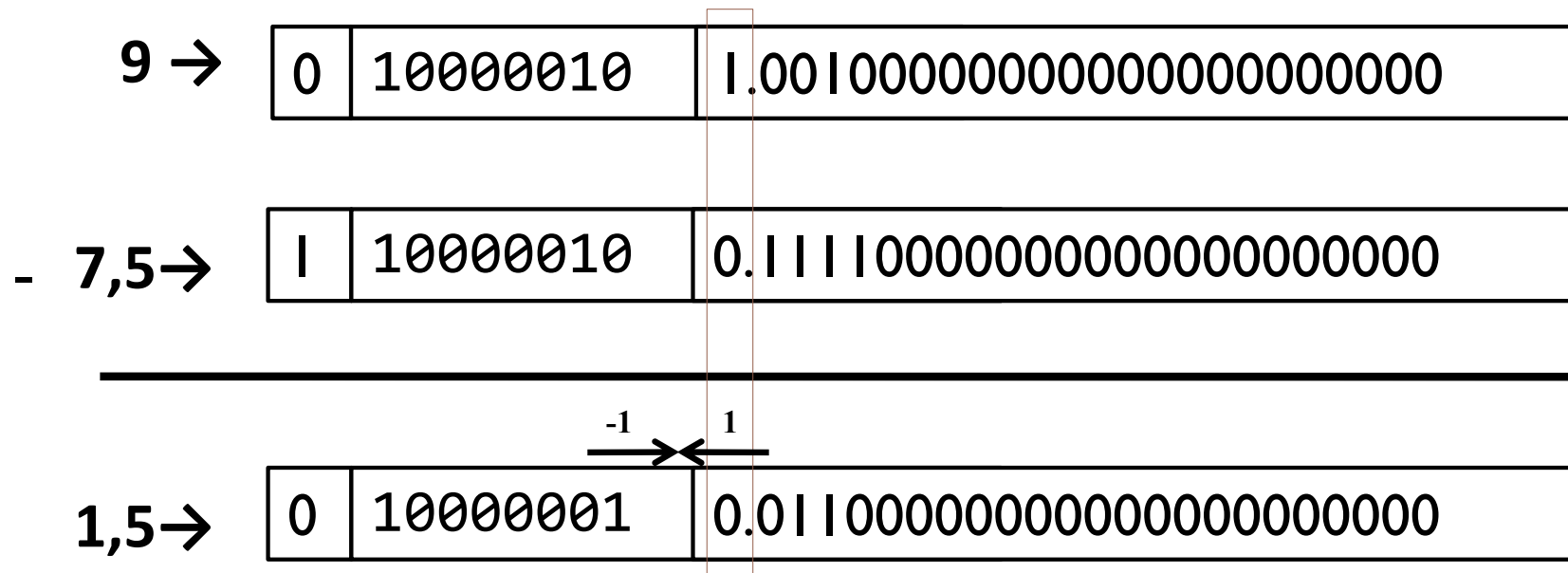
Solución

► Normalizar el resultado



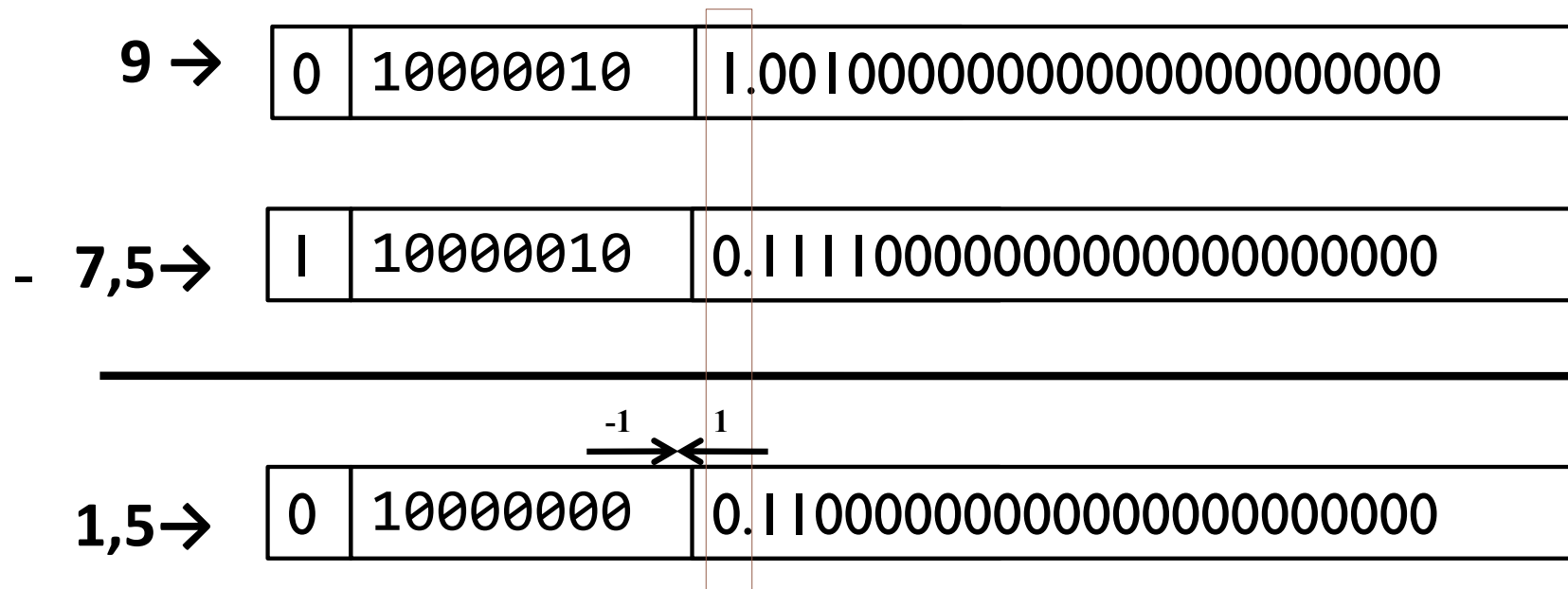
Solución

► Normalizar el resultado



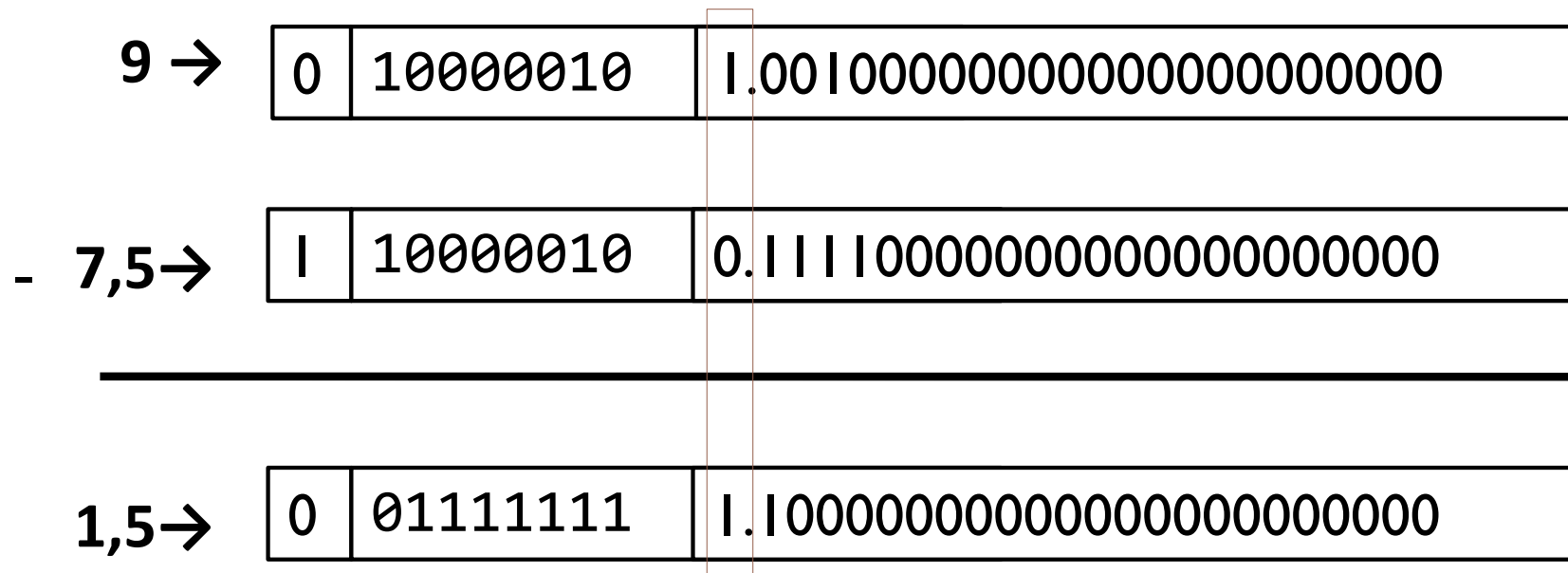
Solución

▶ Normalizar el resultado



Solución

► Normalizar el resultado



mantisa ya normalizada

Solución

- ▶ Se almacena el resultado definitivo eliminando el bit implícito

1,5 → 0 01111111 100000000000000000000000

Ejercicio

- ▶ Usando el formato IEEE 754, multiplicar 7,5 y 1,5 paso a paso

Solución

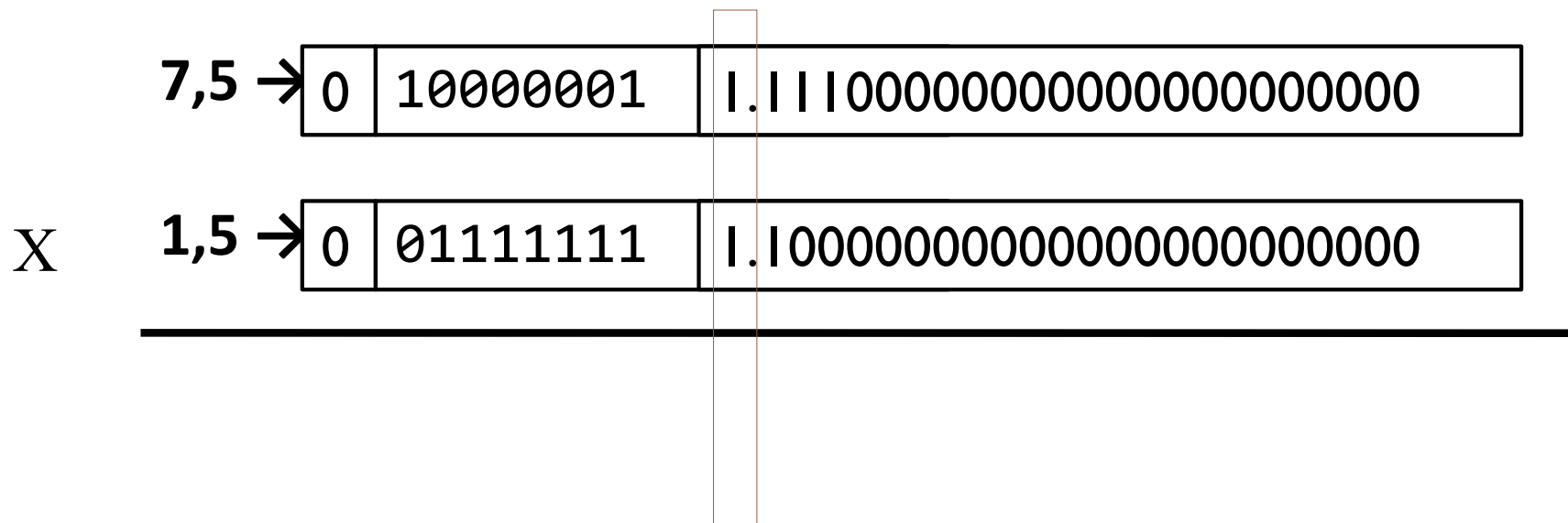
► Representación de los números

9 → 0 10000010 001000000000000000000000

- 7,5 → 1 10000001 111000000000000000000000

Solución

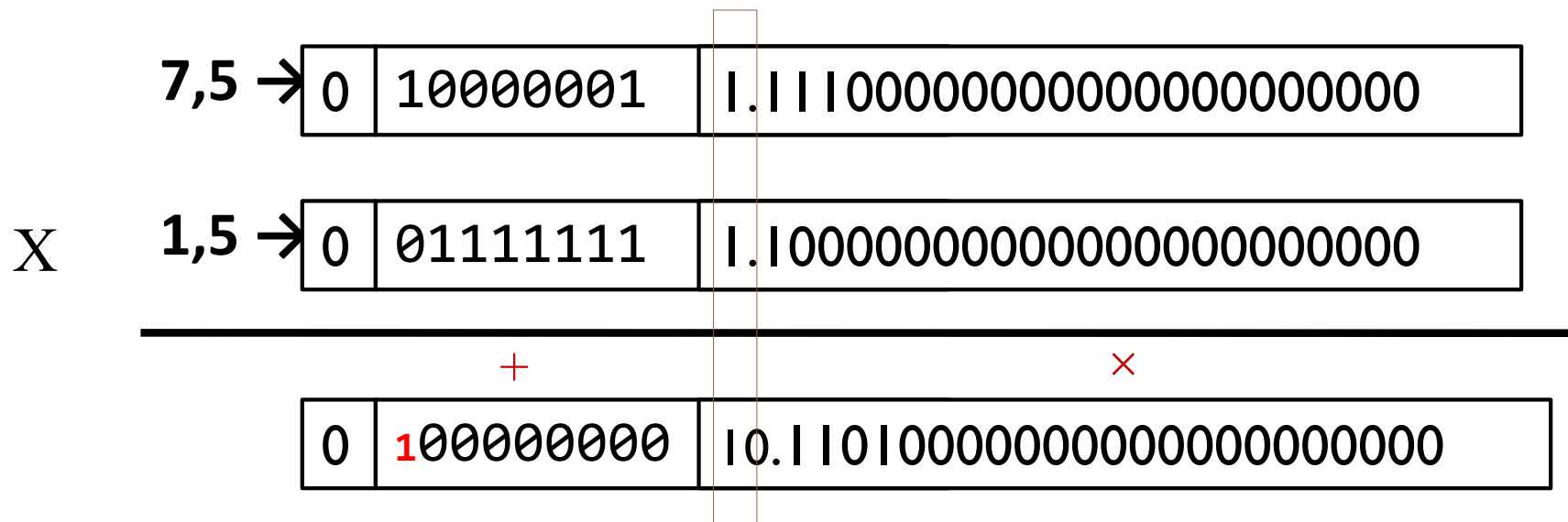
- ▶ Se separan exponentes y mantisas y se añade bit implícito



Se añade el bit implícito para operar

Solución

- ▶ Multiplicar: sumar exponentes y multiplicar mantisas



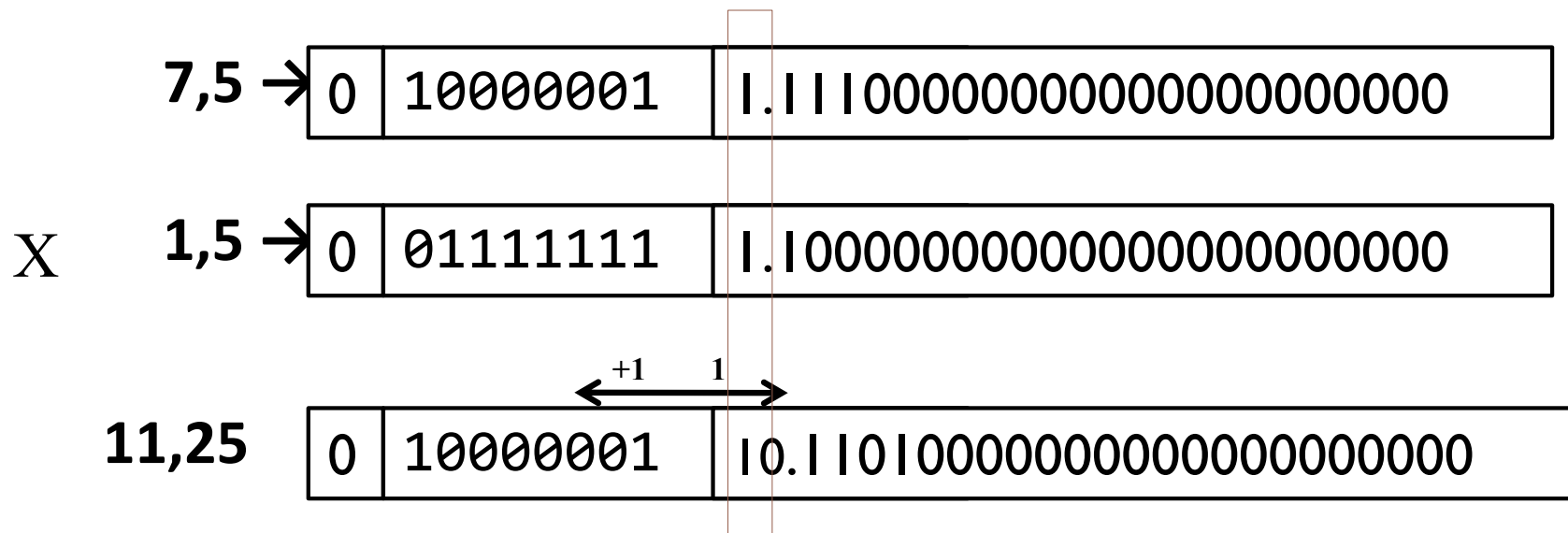
Solución

- ▶ Multiplicar: quitar el sesgo al exponente (hay dos)

$$\begin{array}{r} 7,5 \rightarrow \boxed{0 \mid 10000001 \mid 1.111000000000000000000000000000} \\ X \quad 1,5 \rightarrow \boxed{0 \mid 01111111 \mid 1.100000000000000000000000000000} \\ \hline \boxed{0 \mid 100000000 \mid 10.110100000000000000000000000000} \\ - \quad 01111111 \end{array}$$

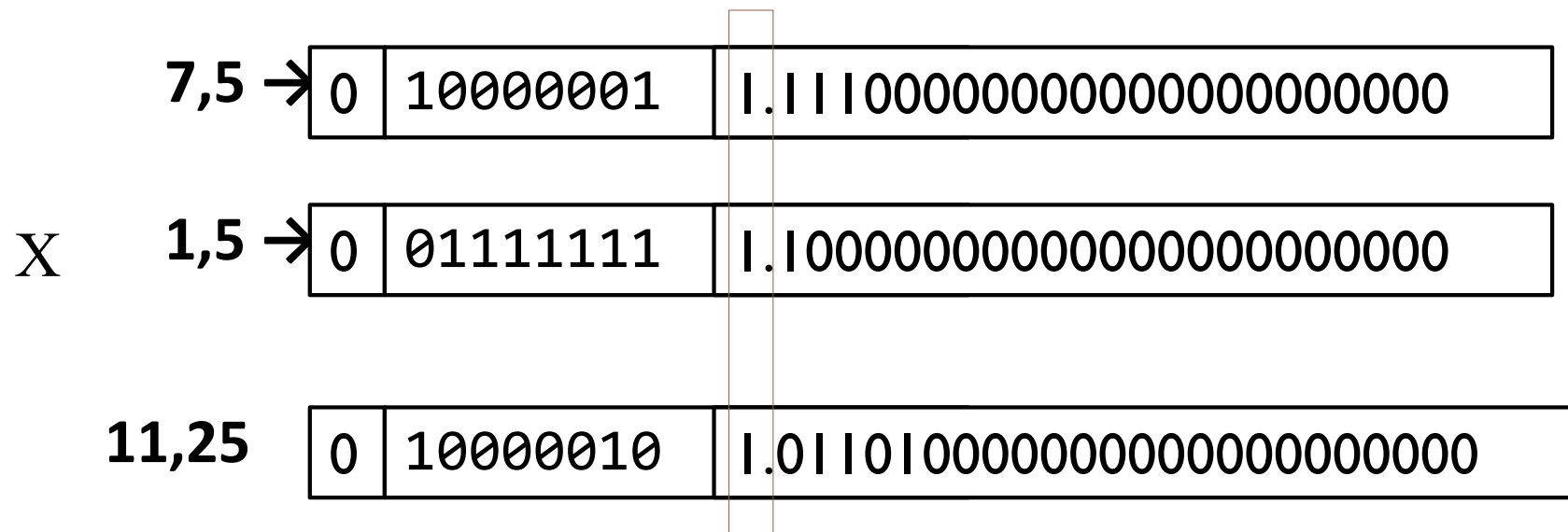
Solución

- ▶ Multiplicar: normalizar el resultado



Solución

► Resultado normalizado...



Solución

- ▶ Se almacena el resultado eliminando el bit implícito

11,25 0 10000010 011010000000000000000000

Evolución de IEEE 754

- ▶ 1985 – IEEE 754
- ▶ 2008 – IEEE 754-2008 (754+854)
- ▶ 2011 – ISO/IEC/IEEE 60559:2011 (754-2008)

Name	Common name	Base	Digits	E min	E max	Notes	Decimal digits	Decimal E max
binary16	Half precision	2	10+1	-14	+15	storage, not basic	3.31	4.51
binary32	Single precision	2	23+1	-126	+127		7.22	38.23
binary64	Double precision	2	52+1	-1022	+1023		15.95	307.95
binary128	Quadruple precision	2	112+1	-16382	+16383		34.02	4931.77
decimal32		10	7	-95	+96	storage, not basic	7	96
decimal64		10	16	-383	+384		16	384
decimal128		10	34	-6143	+6144		34	6144

Asociatividad

- ▶ La coma flotante no es asociativa

- ▶ $x = -1.5 \times 10^{38}, y = 1.5 \times 10^{38}, y z = 1.0$

- ▶ $x + (y + z) = -1.5 \times 10^{38} + (1.5 \times 10^{38} + 1.0)$
 $= -1.5 \times 10^{38} + (1.5 \times 10^{38}) = 0.0$

- ▶ $(x + y) + z = (-1.5 \times 10^{38} + 1.5 \times 10^{38}) + 1.0$
 $= (0.0) + 1.0 = 1.0$

- ▶ Las operaciones coma flotante no son asociativas

- ▶ Los resultados son aproximados

- ▶ 1.5×10^{38} es mucho más grande que 1.0

- ▶ $1.5 \times 10^{38} + 1.0$ en la representación en coma flotante sigue siendo 1.5×10^{38}

Conversión $\text{int} \rightarrow \text{float} \rightarrow \text{int}$

```
if (i == (int)((float) i)) {  
    printf("true");  
}
```

- ▶ **No siempre es cierto**
- ▶ Muchos valores enteros grandes no tienen una representación exacta en coma flotante
- ▶ ¿Qué ocurre con `double`?

Ejemplo

- ▶ El número 133000405 en binario es:
 - ▶ 111111011010110110011010101 (27 bits)
 - ▶ 111111011010110110011010101 $\times 2^0$
- ▶ Se normaliza
 - ▶ 1,11111011010110110011010101 $\times 2^{26}$
 - ▶ S = 0 (positivo)
 - ▶ e = 26 \rightarrow E = 26 + 127 = 153
 - ▶ M = 11111011010110110011010 (se pierden los 3 últimos bits)
- ▶ El número realmente almacenado es
 - ▶ 1,11111011010110110011010 $\times 2^{26} =$
 - ▶ 111111011010110110011010 $\times 2^3 = 133000400$

Conversión float \rightarrow int \rightarrow float

```
if (f == (float)((int) f)) {  
    printf("true");  
}
```

- ▶ **No siempre es cierto**
- ▶ **Los números con decimales no tienen representación entera**