DAM/DAW

PROGRAMACIÓN

02 Anexo 1

Representación de la Información

Rev: 3.2

fran@iessanclemente.net

Indice

Representación de la información (I)

Los computadores representan la información usando dos dígitos, 0 y 1, es decir usando el sistema binario. Los dígitos de este sistema de denominan bits (*binary digits*) y con ellos se representan los siguientes tipos de datos:

- **Números**, que pueden ser de dos tipos:
  - Enteros, es decir, números enteros positivos y negativos
  - En coma flotante. Números reales con cifras decimales
- **Caracteres** alfabéticos y signos de puntuación

Se denomina Byte (B) a la unidad de información base utilizada en computación. Se corresponde con un conjunto ordenado de bits, generalmente 8 bits (octeto) y se creó originalmente para indicar la cantidad más pequeña de datos que una computadora podía manejar. Se utiliza como unidad base para indicar capacidades de almacenamiento y tasas de procesamiento y transferencia de datos

Representación de la información (II)

Los prefijos empleados para los múltiplos de bits y bytes son los prefijos del SI (base 10: 10^3, 10^6, 10^9, ...) y los prefijos binarios (base 2: 2^{10}, 2^{20}, 2^{30}, ...). Los primeros se usan más con tasas de transmisión, almacenamiento en disco,... mientras que los segundos en relación a memorias (RAM, ROM,...)

S.I. (decimal)			Binario		
kilobyte	kB	10^3	**kibi**byte	KiB	2^{10}
megabyte	MB	10^6	**mebi**byte	MiB	2^{20}
gigabyte	GB	10^9	**gibi**byte	GiB	2^{30}
terabyte	TB	10^{12}	**tebi**byte	TiB	2^{40}
petabyte	PB	10^{15}	**pebi**byte	PiB	2^{50}
exabyte	EB	10^{18}	**exbi**byte	EiB	2^{60}
zettabyte	ZB	10^{21}	**zebi**byte	ZiB	2^{70}
yottabyte	YB	10^{24}	**yobi**byte	YiB	2^{80}

Representación de la información (III)

Números enteros

- La representación binaria de números naturales se realiza de igual forma a como se hace en el decimal, pero empleando potencias de 2 (base[1] 2 dígitos: 0,1) en lugar de potencias de 10 (base 10: 0,1,...,9)
- Un código binario de n-dígitos permite representar 2^n números decimales positivos, desde 0 a 2^n-1
- El dígito correspondiente al exponente menor de la base es el bit menos significativo (LSB) y el del exponente mayor, el bit más significativo (MSB)

Número decimal (base-10): d**1425**
$1*10^3 + 4*10^2 + 2*10^1 + 5*10^0$
$= 1000 + 400 + 20 + 5 = 1425$

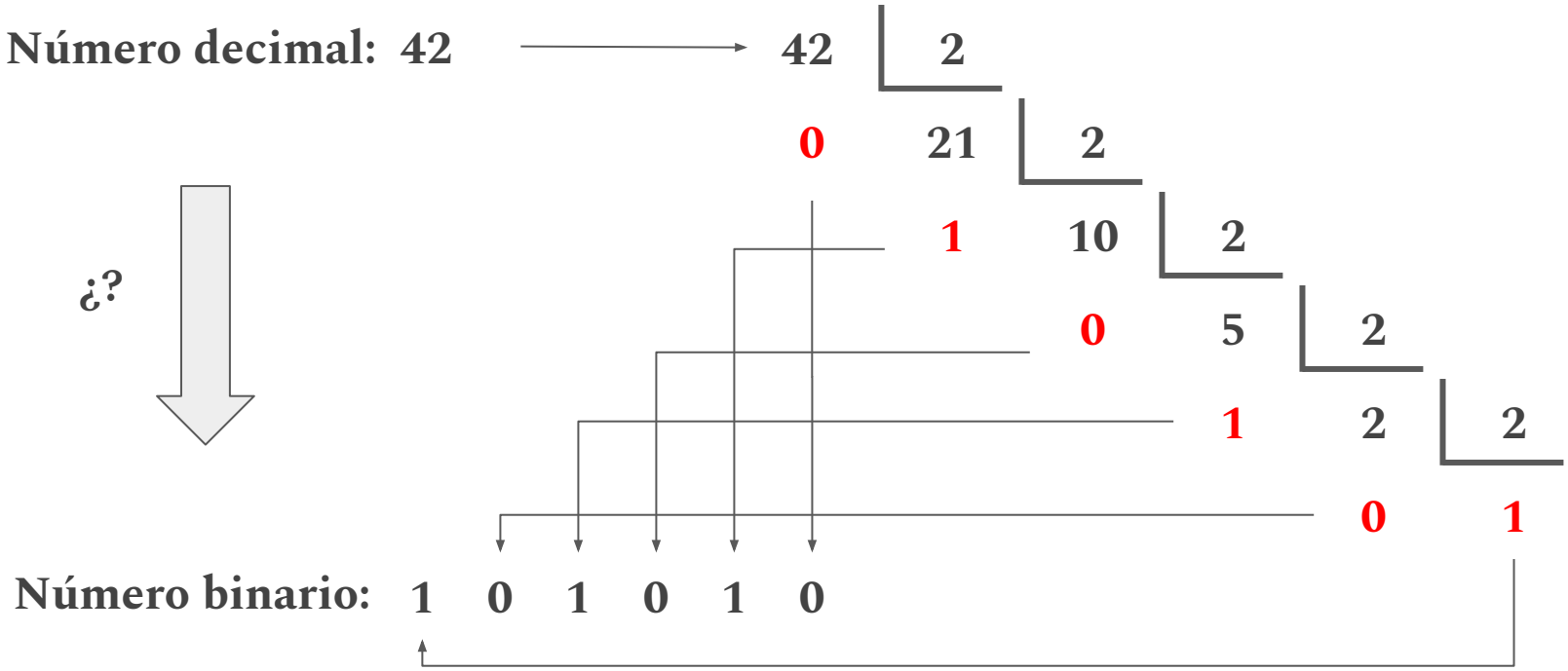
Número binario (base-2): 0b**10011101**
$1*2^7 + 0*2^6 + 0*2^5 + 1*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0$
$= 128 + 16 + 8 + 4 + 1 = 157$

[1] La base del sistema de numeración indica de cuántos dígitos (símbolos) diferentes se compone

Representación de la información (IV)

Conversión decimal → binario

Para realizar la conversión desde decimal a otra base, en este caso binaria, se realizan sucesivas divisiones enteras entre la nueva base (2) y cada resto resultante se anota como un nuevo dígito del número en la nueva base, empezando desde la posición menos significativa. El último cociente obtenido será el dígito más significativo del número resultante.



Representación de la información (V)

Bases octal y hexadecimal

- Para representar números binarios es habitual utilizar los sistemas octal (base 8, dígitos 0,1,...,7) y hexadecimal (base 16, dígitos 0,..., 9, A, B,..., F)
- Para pasar de binario a octal, se agrupan los bits de 3 en 3

binario	octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

binario → octal

0b11000001

↓

011.000.001

↓

3 0 1

↓

0o301

octal → decimal

0o301

$3*8^2 + 0*8^1 + 1*8^0$

$= 3*64 + 1 = 193$

Representación de la información (VI)

- Para pasar de binario a hexadecimal, se agrupan los bits de 4 en 4

bin	dec	hex		bin	dec	hex
0000	0	0		1000	8	8
0001	1	1		1001	9	9
0010	2	2		1010	10	A
0011	3	3		1011	11	B
0100	4	4		1100	12	C
0101	5	5		1101	13	D
0110	6	6		1110	14	E
0111	7	7		1111	15	F

binario → hexadecimal

0b11000001

↓

1100.0001

↓

12 1

↓

0xC1

hexadecimal → decimal

0xC1

$C*16^1 + 1*16^0$

$= 12*16 + 1 = 193$

- Para pasar de decimal a octal o hexadecimal, podemos realizar divisiones sucesivas por la base correspondiente (8 ó 16), pero es más cómodo realizar el paso intermedio a binario y, posteriormente, a octal o hexadec

Representación de la información (VII)

Números enteros con signo

- ¿Cómo podemos representar los signos + y - en un mundo binario?
- Fácil!! Como sólo hay dos signos, basta con dedicar un bit para representar el signo (normalmente el bit más significativo)
- **Representación signo-magnitud**

Supongamos un código binario de 3 dígitos...

positivos		negativos	
binario	decimal	binario	decimal
000	0	100	-0
001	1	101	-1
010	2	110	-2
011	3	111	-3

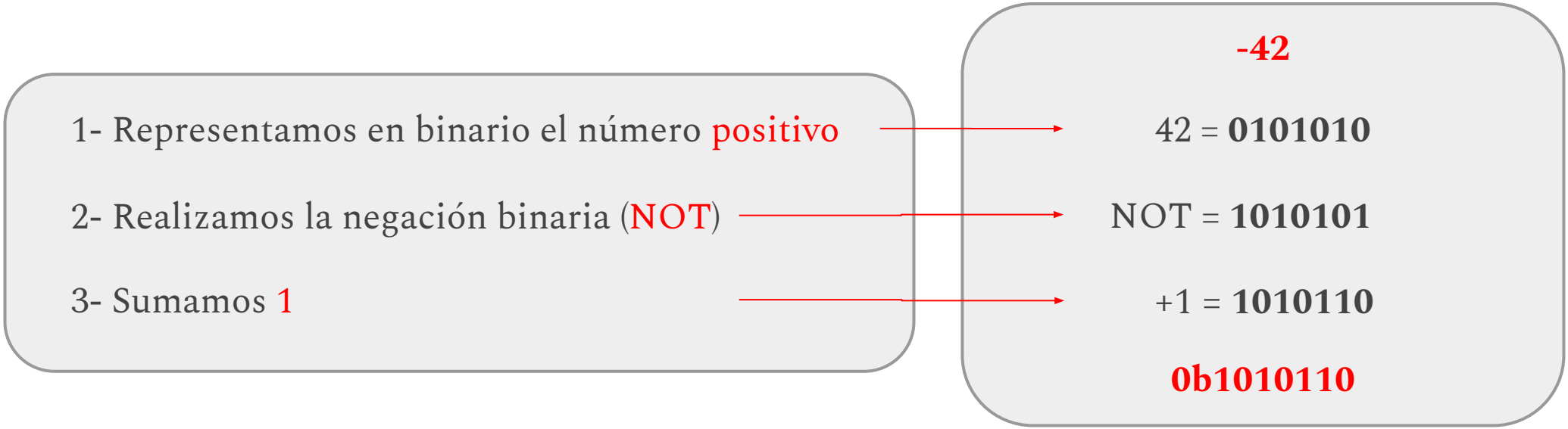
PROBLEMAS!!
representación doble del 0
¿y si hago 1 + (-1)?
= -2 !!!!

Representación de la información (VIII)

- **Representación en complemento a 2**

Para evitar las ambigüedades de la representación signo-magnitud y facilitar los cálculos, se ha adoptado universalmente el sistema de representación en complemento a 2

- ¿Cómo obtener la representación en comp-2 de un número negativo?



- Para pasar un binario negativo en comp-2 a decimal, hacemos 2) y 3), pasamos a decimal y le ponemos signo negativo

Representación de la información (IX)

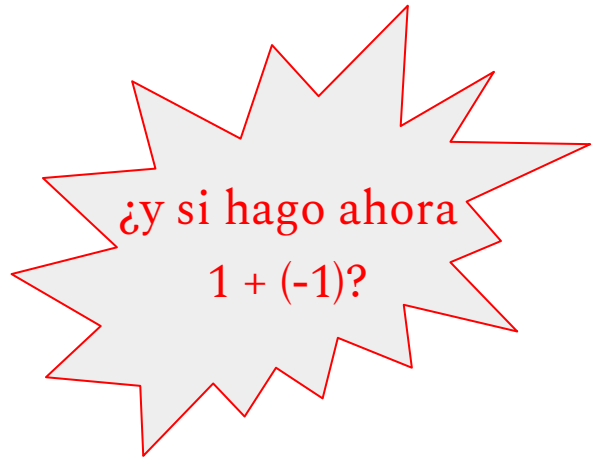
- Dado que perdemos uno de los bits para representar el signo, un código binario de n-dígitos permite representar 2^{n-1} números decimales positivos, desde 0 a 2^n-1, y otros tantos negativos, desde -1 a -2^n

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

a) código binario 4-bits (sin negativos)

bin	dec		bin	dec
0000	0		1000	-8
0001	1		1001	-7
0010	2		1010	-6
0011	3		1011	-5
0100	4		1100	-4
0101	5		1101	-3
0110	6		1110	-2
0111	7		1111	-1

b) código binario 4-bits (con negativos comp-2)



Representación de la información (X)

Algunas operaciones aritmético-lógicas binarias...

	NOT
0	**1**
1	**0**

AND	0	1
0	**0**	**0**
1	**0**	**1**

OR	0	1
0	**0**	**1**
1	**1**	**1**

XOR	0	1
0	**0**	**1**
1	**1**	**0**

SUMA:

0 + 0 = 0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 0, acarreo 1

ej: 3 + 2

```
  0011  (3)
+ 0010  (2)
--------
  0101  (5)
```

RESTA en comp-2: suma de complementario

ej: 5 - 2 = 5 + (-2)

```
  0101   (5)
+ 1110  (-2)
---------
(1)0011  (3)
```

Desplazamiento lógico
(rellenamos con 0)

0 1 1 0 ← **SLL** (Shift Left Logical)
0 0 1 1
→ **0 0 0 1** **SRL** (Shift Right Logical)

multiplicar y dividir por potencias de 2

Desplazamiento aritmético
(en SRA rellenamos con signo)

0 1 1 0 ← **SLA** (Shift Left Arithm)
1 0 1 1
→ **1 1 0 1** **SRA** (Shift Right Arithm)

Representación de la información (XI)

Números decimales

- Para la representación binaria de números decimales, existen dos esquemas: coma fija y punto flotante
- **Números decimales de coma fija en comp-2**

El formato de coma fija asume que hay un cierto número de bits que representan la parte decimal y, por tanto, un punto decimal implícito en una determinada posición

Supongamos un código binario de 16 dígitos, de los cuales 4 son decimales, ¿qué se podría representar?

signo	parte entera	parte decimal
bit 15	bits 14-4	bits 3-0
signo + 11 bits: [-2048, 2047]		prec: 0.0625 (1/16)

Representación de la información (XII)

- **Ejemplo de binario coma fija a decimal**

Supongamos el número 0b0011001101011011 del código anterior de 16 bits...

Número binario: **0011001101011011**

Signo: **0**011001101011011 → (+)

P. entera: **0011001101011**011

$1*2^9 + 1*2^8 + 1*2^5 + 1*2^4 + 1*2^2 + 1*2^0$
$= 821$

P. decimal: **0011001101011011**

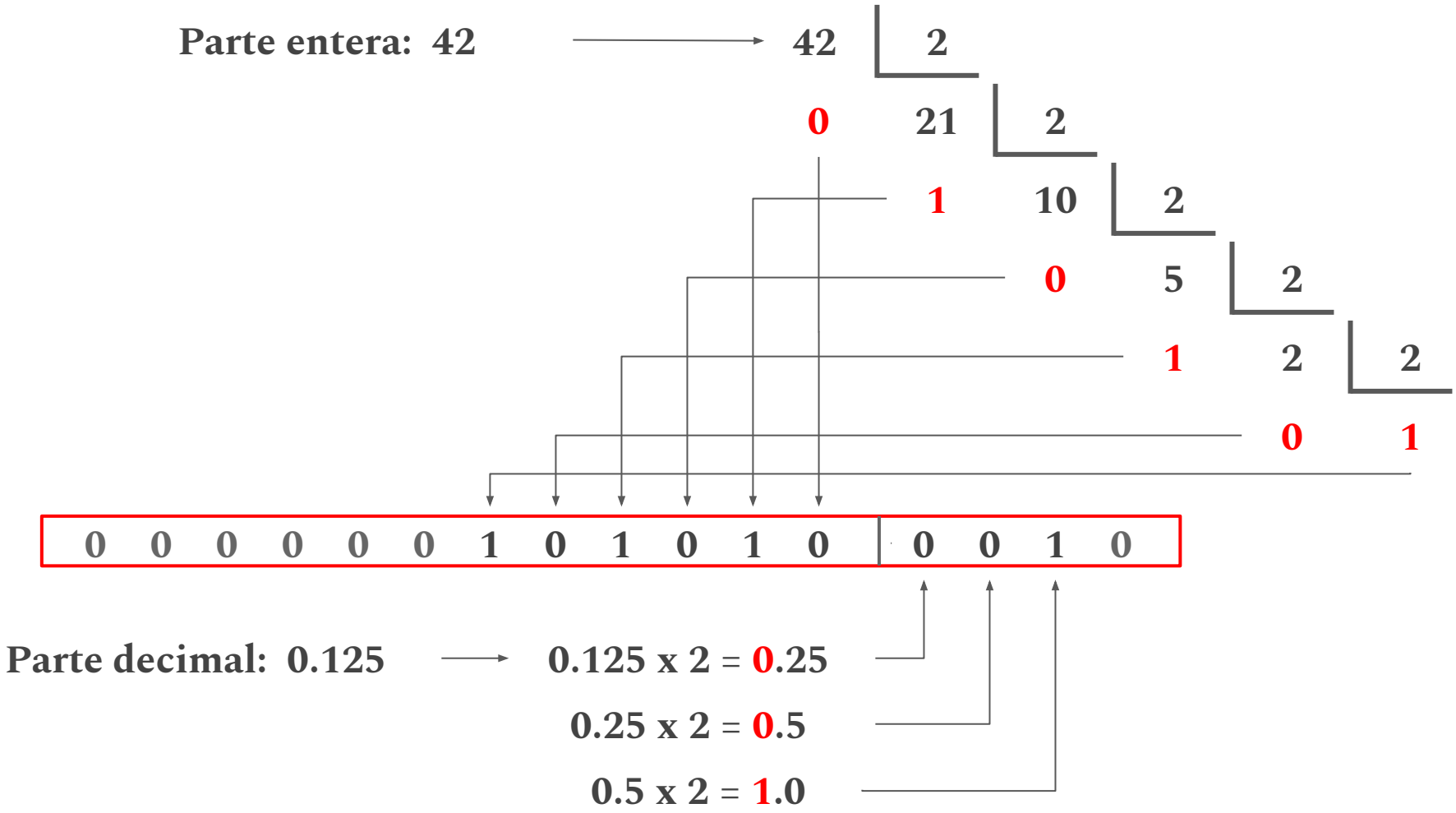
$1*2^{-1} + 0*2^{-2} + 1*2^{-3} + 1*2^{-4}$
$= 0.5 + 0.125 + 0.0625 = 0.6875$

Número decimal: **821.6875**

Representación de la información (XIII)

- **Ejemplo de decimal a binario coma fija**

Supongamos el número 42.125 y lo pasamos al código anterior de 16 bits...



Representación de la información (XIV)

Número decimales en coma flotante

Los números en coma fija permiten representar números con una parte decimal relativamente grande. Por ejemplo, los 4 bits del código anterior permiten una precisión máxima de 2^{-4}, es decir, 0.0625. Pero, ¿y si necesitamos representar números más pequeños? ¿por ejemplo $2.5x10^{-6}$?

La representación en punto flotante usa el formato:

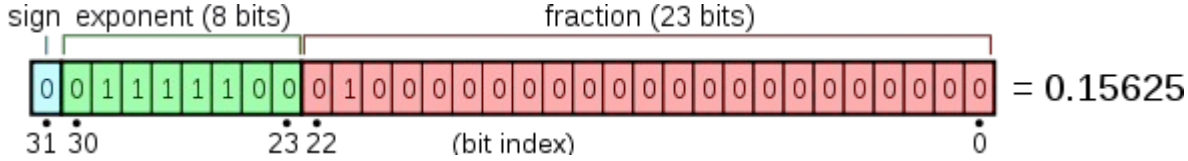
$sM \times B^{sE}$

s: signo
M: mantisa o fracción decimal
B: base, en este caso 2 (binario)
E: exponente

$sM \times B^{sE}$

Por ejemplo, el estándar IEEE754 define el formato binary32 como:

s: 1 bit
M: 24 bits (23 almacenados)
E: 8 bits (sesgo de 127, [127, -126])



➢ Calculadora IEEE754: https://www.h-schmidt.net/FloatConverter/IEEE754.html

Representación de la información (XV)

- **Ejemplo de *binary32* a decimal**

  Supongamos el número 0x41C80000...
  que en binario es 0100 0001 1100 1000 0000 0000 0000 0000

S	Exp	Mantisa
0	100 0001	1100 1000 0000 0000 0000 0000

  signo: 0 → positivo (+)

  exponente: 100 0001 1 = 131 → 131 - 127 = 4

  (se añade un 1 correspondiente a 2^0)

  mantisa: 1 100 1000 0000 0000 0000 0000

  $1*2^0 + 1*2^{-1} + 1*2^{-4} = 1 + 0.5 + 0.0625 = 1.5625$

  El número resultante es ($sMx2^E$): $+1.5625 \times 2^4 =$ **25**

Representación de la información (XVI)

- **Ejemplo de decimal a *binary32***

  Supongamos el número 42.125_{10}

- Obtenemos la representación binaria de la parte entera y la decimal:

  0 0 0 0 0 0 1 0 1 0 1 0 . 0 0 1 0

- Desplazamos el punto decimal hasta la derecha del primer 1 multiplicando por potencias de **2** (izq→pot(+), drch→pot(-))

  0 0 0 0 0 0 1 . 0 1 0 1 0 0 0 1 0 x $\mathbf{2^5}$

- signo (42.125): positivo → 0
- mantisa (descartamos el 1) : 010 1000 1000 0000 0000 0000
- exponente: 5 → 5 + 127 = 132 → 1000 0100

S	Exp	Mantisa
0	1000 0100	010 1000 1000 0000 0000 0000

Representación de la información (XVII)

Representación de caracteres

- De igual modo a lo visto con los números, necesitamos un sistema de codificación que nos permita representar los caracteres (alfanuméricos y de control) mediante dígitos binarios de forma que puedan ser almacenados y procesados computacionalmente
- Existen multitud de esquemas de codificación. Algunos de los más representativos son:
  - ASCII, *American Standard Code for Information Interchange*
  - ISO 8859-1
  - UNICODE (UTF-8, UTF-16, UTF-32)

Representación de la información (XVIII)

ASCII

- Creado en 1963 por el Comité Estadounidense de Estándares como una evolución de los conjuntos de códigos utilizados entonces en telegrafía, se convirtió en un estándar *de facto* de los sistemas informáticos para representar textos y para el control de dispositivos que manejan texto como el teclado
- El código ASCII utiliza 7 bits para representar los caracteres. Inicialmente, se empleaba un bit adicional (bit de paridad) para detectar errores en la transmisión. Variantes posteriores emplearon ese bit adicional para añadir nuevos caracteres, muchas veces a costa de la compatibilidad entre diferentes sistemas informáticos
- Define códigos para 95 caracteres imprimibles y otros 32 caracteres no imprimibles, de los cuales la mayoría son caracteres de control que tienen efecto sobre cómo se procesa el texto

Representación de la información (XIX)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Representación de la información (XX)

ISO/IEC 8859-1

- También conocida como Alfabeto Latino n.º 1 o ISO Latín 1, es una norma de la ISO que define la codificación del alfabeto latino, incluyendo los diacríticos (como letras acentuadas, ñ, ç), y letras especiales (como ß, Ø), necesarios para la escritura de las lenguas originarias de Europa occidental
- Esta norma pertenece al grupo de juegos de caracteres de la ISO conocidos como ISO/IEC 8859 que se caracterizan por poseer la codificación ASCII en su rango inicial (128 caracteres) y otros 128 caracteres para cada codificación, con lo que en total utilizan 8 bits
- Los caracteres de ISO-8859-1 son además los primeros 256 caracteres del estándar ISO/IEC 10646 (Unicode)
- La norma ISO/IEC 8859-15 consistió en una revisión de la ISO 8859-1 incorporando, entre otros caracteres, el símbolo del Euro

Representación de la información (y XXI)

UNICODE

- Es un estándar de codificación de caracteres diseñado para facilitar el tratamiento informático, transmisión y visualización de textos en múltiples lenguajes (tabla de caracteres: https://unicode-table.com/es/)
- Incluye todos los caracteres de uso común en la actualidad. La versión 13 (2020) contenía 143 859 caracteres provenientes de alfabetos, sistemas ideográficos y colecciones de símbolos (matemáticos, técnicos, musicales, iconos...). Incluye tanto sistemas de escritura modernos (latino, árabe, braille, cirílico, griego, hanzi chino, kanji japonés,...), como escrituras históricas extintas (cuneiforme, fenicio, rúnico,...)
- Compatible hacia atrás con ASCII-7 o ISO 8859-1, define diferentes formas de codificación para diferentes arquitecturas:
  - UTF-8, de 8 bits y longitud variable
  - UTF-16, de 16 bits y longitud variable
  - UTF-32, de 32 bits y longitud fija