

Sistemas de Numeración y Representación numérica. Aritmética Binaria

Sistemas de numeración

☐ Sistema de numeración

Conjunto de símbolos y reglas de combinación de dichos símbolos que permiten representar los números enteros y/o fraccionarios.

Sistemas de Numeración

❑ Sistemas de numeración no posicionales

- Son los más primitivos.
- Usaban los dedos de la mano para representar la cantidad cinco y después hablaba de cuántas manos se tenía.
- Usaban cuerdas con nudos para representar cantidad.

Sistemas de Numeración

❑ Sistemas de numeración semi posicionales

- El sistema de los números romanos por ejemplo
- Es muy complejo diseñar algoritmos de uso general (por ejemplo, para sumar, restar, multiplicar o dividir).
- En el número romano XCIX (99 decimal) los numerales X (10 decimal) del inicio y del fin de la cifra equivalen siempre al mismo valor, sin importar su posición dentro de la cifra

Sistemas Posicionales

- ❑ El número de símbolos permitidos en un sistema de numeración posicional se conoce como base del sistema de numeración.
- ❑ Si un sistema de numeración posicional tiene base b significa que disponemos de b símbolos diferentes para escribir los números

Sistemas Posicionales

□ En un sistema de numeración posicional **de base b** , la representación de un número se define a partir de la regla:

$$(...a_3a_2a_1a_0.a_{-1}a_{-2}a_{-3}...)_b = ... + a_2b^2 + a_1b^1 + a_0b^0 + a_{-1}b^{-1} + a_{-2}b^{-2} + a_{-3}b^{-3} + ...$$

□ Donde:

○ $b \in \mathbb{Z}$, y $b > 1$ cuando $a_i \in \mathbb{Z}$ de enteros y $0 \leq a_i < b$

○ El punto entre los dígitos a_0 y a_{-1} se denomina punto fraccionario. Cuando $b = 10$ se lo llama punto decimal y cuando $b = 2$, punto binario.

Bases de Interés

- ❑ **Sistema Decimal:** Es el sistema de numeración utilizado en la vida cotidiana, cuya base es diez, utilizando los símbolos 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9 .
- ❑ **Sistema Binario:** los dos símbolos utilizados son el 0 y el 1, los que reciben el nombre de bit (**binarydigit**).
- ❑ **Sistema Octal:** de base 8, los símbolos utilizados son 0, 1, 2, 3, 4, 5, 6, 7.
- ❑ **Sistema Hexadecimal:** de base 16, los símbolos utilizados son 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Expresiones Generales

❑ Decimal

$$N = \sum_{i=-k}^n d_i \cdot 10^i$$

❑ Binario

$$N = \sum_{i=-k}^n d_i \cdot 2^i$$

❑ Octal

$$N = \sum_{i=-k}^n d_i \cdot 8^i$$

❑ Hexadecimal

$$N = \sum_{i=-k}^n d_i \cdot 16^i$$

Ejemplos

$$243.51_{10} = 2 * 10^2 + 4 * 10^1 + 3 * 10^0 + 5 * 10^{-1} + 1 * 10^{-2}$$

$$212_3 = 2 * 3^2 + 1 * 3^1 + 2 * 3^0 = 23_{10}$$

$$10110_2 = 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 0 * 2^0 = 22_{10}$$

Sistema Binario

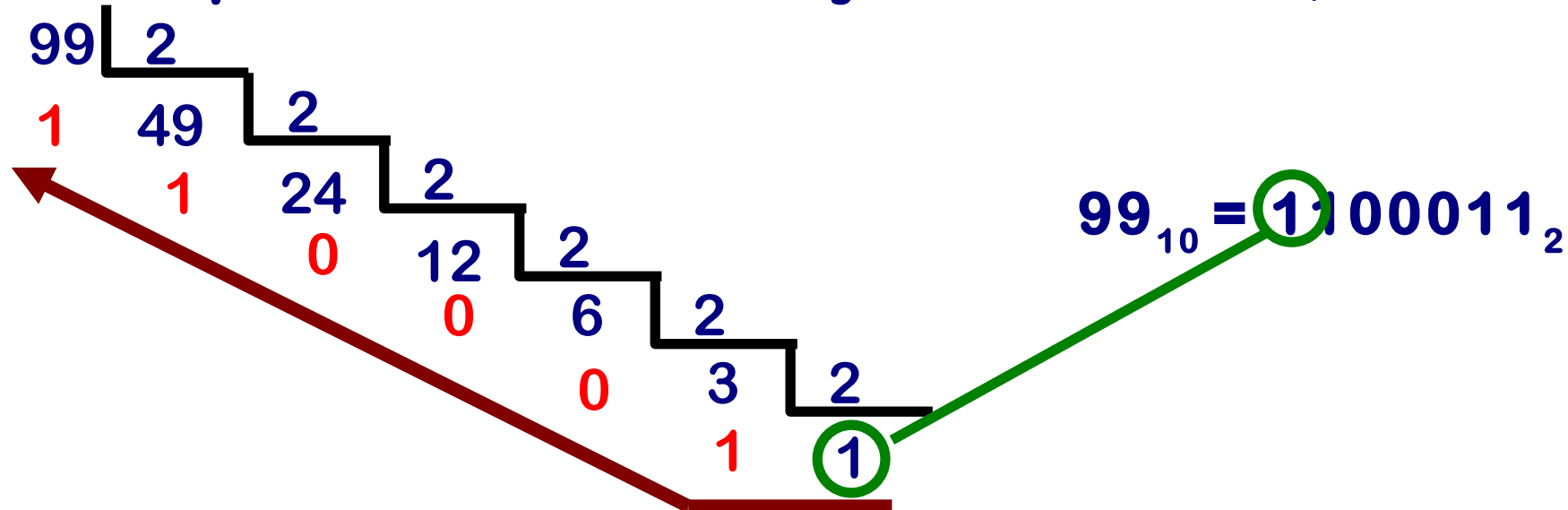
- ❑ **Bit: (B**inary Digit) Un bit es un dígito binario. Como tal, puede tener 2 valores posibles, 1 y 0. Como los circuitos de una computadora pueden asumir 2 estados, los bits se utilizan para representar el estado de los circuitos. Y siendo uno de estos circuitos la unidad mínima de almacenamiento que posee una computadora, el bit será la mínima unidad de representación.
- ❑ **Byte:** En términos generales, un byte es un conjunto de bits. En el presente, se entiende como byte al conjunto de 8 bits.
- ❑ **Palabra:** Una palabra es el conjunto de bits que pueden ser accedidos por la CPU en un requerimiento de lectura/escritura.

Representación en las cuatro bases

Decimal	Binario	Octal	Hexadecimal	Decimal	Binario	Octal	Hexadecimal
00	0000	00	00	09	1001	11	09
01	0001	01	01	10	1010	12	0A
02	0010	02	02	11	1011	13	0B
03	0011	03	03	12	1100	14	0C
04	0100	04	04	13	1101	15	0D
05	0101	05	05	14	1110	16	0E
06	0110	06	06	15	1111	17	0F
07	0111	07	07	16	10000	20	10
08	1000	10	08				

Métodos de cambio de base

- ❑ Divisiones sucesivas.
- ❑ Se divide el número a convertir por la base a convertir, hasta que el cociente de un número menor que dicha base. Ej: 99 decimal, a binario

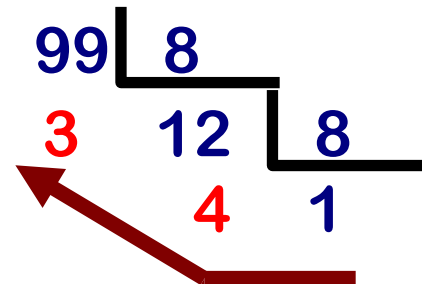


Otros ejemplos de Divisiones sucesivas

❑ El mismo número 99 a base 16:

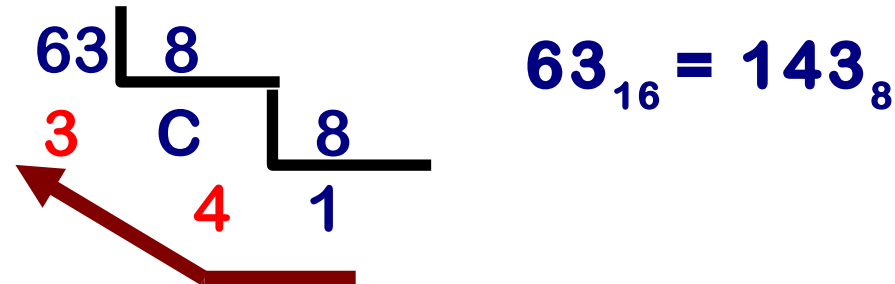
$$\begin{array}{r|l} 99 & 16 \\ \hline & 6 \\ \hline 3 & \end{array} \quad 99_{10} = 63_{16}$$


❑ Y convertido a octal

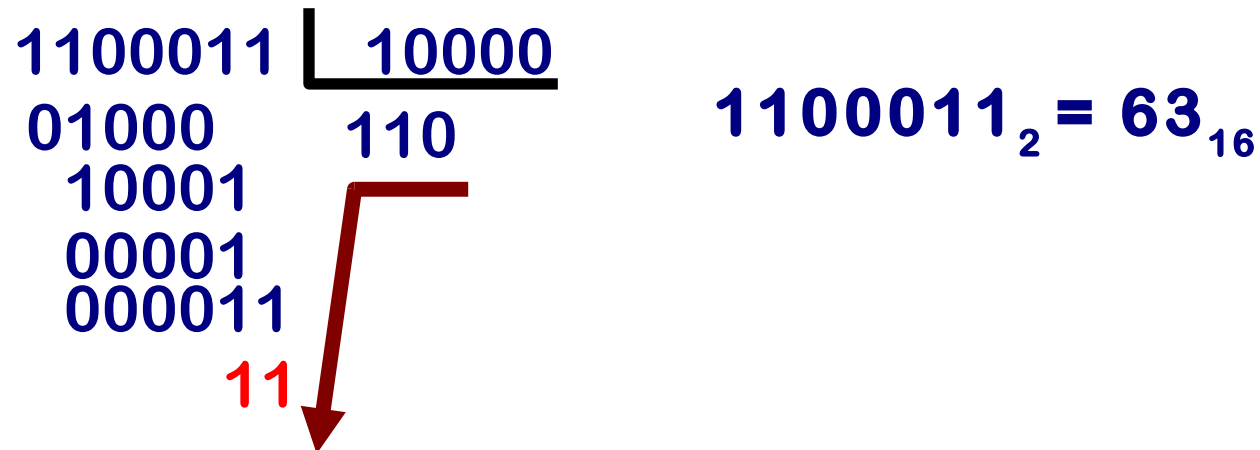
$$\begin{array}{r|l} 99 & 8 \\ \hline & 12 \\ \hline 3 & \end{array} \quad \begin{array}{r|l} 12 & 8 \\ \hline & 4 \\ \hline 1 & \end{array} \quad 99_{10} = 143_8$$


Mas ejemplos

□ El 63 hexadecimal (es decir el 99 decimal) a octal



□ El 1100011_2 (es decir 99_{10}) a hexadecimal



Métodos de cambio de base

❑ Restas sucesivas

$$\begin{array}{r} 99 \\ - 64 \quad 2^6 \times 1 \\ \hline 35 \\ - 32 \quad 2^5 \times 1 \\ \hline 3 \\ - 0 \quad 2^4 \times 0 \\ \hline 3 \\ - 0 \quad 2^3 \times 0 \\ \hline 3 \\ - 0 \quad 2^2 \times 0 \\ \hline 3 \\ - 2 \quad 2^1 \times 1 \\ \hline 1 \end{array}$$

$$\begin{array}{r} 99 \\ - 96 \quad 16^1 \times 6 \\ \hline 3 \end{array}$$

$$\begin{array}{r} 99 \\ - 64 \quad 8^2 \times 1 \\ \hline 35 \\ - 32 \quad 8^1 \times 4 \\ \hline 3 \end{array}$$

Conversiones rápidas

❑ Binario a octal:

- Dividiendo el número en grupos de tres bits se pasa en forma directa traduciendo cada grupo a decimal (al ser tres bits nunca tendremos un número mayor de 7)

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

❑ Octal a binario

- Cada dígito octal genera tres bits en binario

Conversiones rápidas

❑ Binario a hexadecimal:

- Dividiendo el número en **nibbles** (grupos de cuatro bits) se pasa en forma directa traduciendo cada grupo a decimal (cuando se tiene 10, 11, 12, etc se coloca A,B,etc)

❑ Ej

101 0010 1110 1010 1001 1100 0011 0000 0111₂
5 2 E A 9 C 3 0 7₁₆

Números Negativos

❑ Para representar un número con signo se pueden utilizar diversas notaciones:

- Signo y Magnitud
- Complemento a 1
- Complemento a 2
- Binario Desplazado

Signo y Magnitud

- ❑ Se destina el bit mas significativo (MSB) para el signo: 0 para números positivos y 1 para números negativos.
- ❑ El resto de los bits contiene la magnitud, es decir el valor absoluto.

$$01111111_2 = +127_{10}$$

$$00000000_2 = 0_{10}$$

$$10000000_2 = -0_{10}$$

$$11111111_2 = -127_{10}$$

Complemento a 1

- ❑ Para representar el número es necesario invertir cada bit por su complemento (1 en 0 y viceversa)
- ❑ Cada número negativo es el C(1) del positivo

$$01111111_2 = +127_{10}$$

$$00000000_2 = 0_{10}$$

$$11111111_2 = -0_{10}$$

$$10000000_2 = -127_{10}$$

- ❑ Al sumar dos números de distinto signo hay que sumar el acarreo del MSB para no tener errores.

Complemento a 2. C(2)

- ❑ $C(2) = 1 + C(1)$
- ❑ Elimina la ambigüedad del “0 signado”
- ❑ No requiere ajuste con acarreo en la suma de números de diferente signo.

$$00000000_2 = 0_{10}$$

$$01111110_2 = +126_{10}$$

$$01111111_2 = +127_{10}$$

$$10000000_2 = -128_{10}$$

$$10000001_2 = -127_{10}$$

$$11111111_2 = -1_{10}$$

Binario Desplazado

- ❑ Se suma al valor signado el valor absoluto de la mitad del rango. Ej: 8 bits: Suma 127

$$00000000_2 = -127_{10}$$

$$01111110_2 = -1_{10}$$

$$01111111_2 = 0_{10}$$

$$10000000_2 = +1_{10}$$

$$11111111_2 = +128_{10}$$

Resumen de números signados

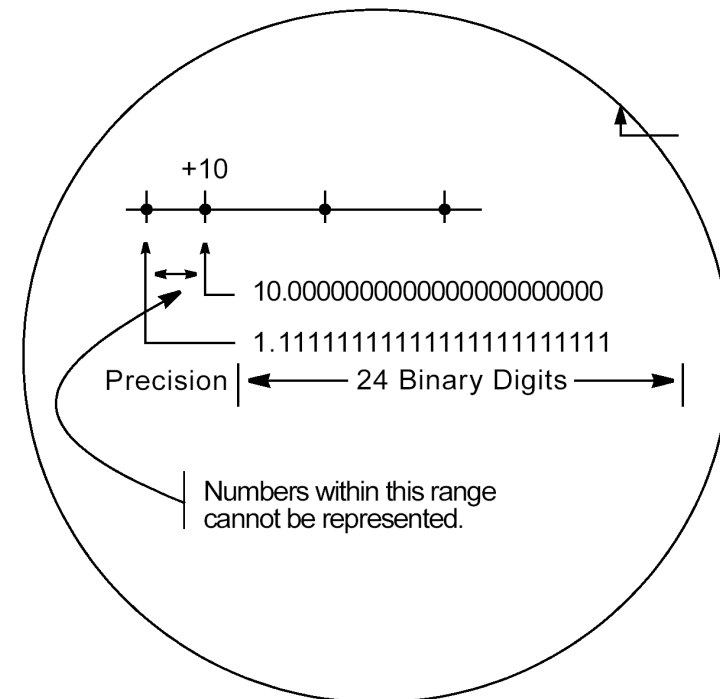
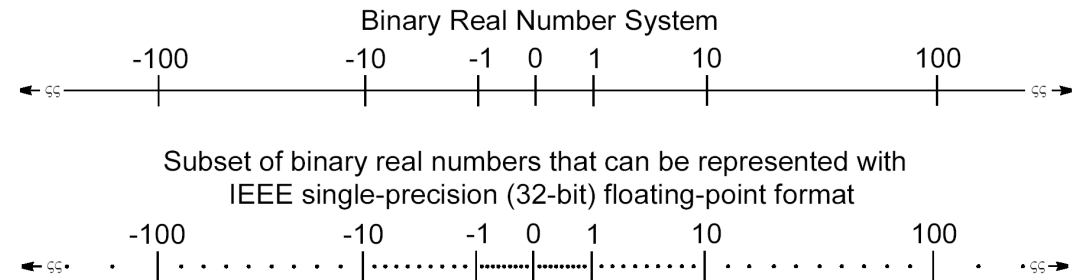
Decimal	Entero positivo	Signo y magnitud	C(1)	C(2)	Binario Desplazado
8	1000	n/a	n/a	n/a	1111
7	111	111	111	111	1110
6	110	110	110	110	1101
5	101	101	101	101	1100
4	100	100	100	100	1011
3	11	11	11	11	1010
2	10	10	10	10	1001
1	1	1	1	1	1000
(+)0	0	0	0	0	0111
(-)0	n/a	1000	1111	n/a	n/a
-1	n/a	1001	1110	1111	0110
-2	n/a	1010	1101	1110	0101
-3	n/a	1011	1100	1101	0100
-4	n/a	1100	1011	1100	0011
-5	n/a	1101	1010	1011	0010
-6	n/a	1110	1001	1010	0001
-7	n/a	1111	1000	1001	0000
-8	n/a	n/a	n/a	1000	n/a

Números de precisión finita

- ❑ En un computador la cantidad de dígitos disponibles para representar un número siempre será limitada.
- ❑ No podemos por ejemplo almacenar el rango de los números enteros, ya que se extiende desde $-\infty$ hasta $+\infty$
- ❑ A los números que podemos almacenar en un computador se los denomina por lo tanto: ***números de precisión finita.***

Números Reales

- ❑ El rango de los números reales comprende desde $-\infty$ hasta $+\infty$.
- ❑ Los registros de un procesador tienen resolución finita.
- ❑ Por lo tanto un computador solo puede representar un sub conjunto de \mathcal{R} .
- ❑ Además, no es solo un tema de magnitud sino de resolución.



Representación binaria de Números Reales

- ❑ En general podemos formalizar la representación de un número real expresado en los siguientes formatos:
 - Punto Fijo
 - Punto Flotante

Representación binaria en Punto Fijo con signo

❑ Se representan mediante una expresión del tipo:

$$(a_n a_{n-1} \dots a_0 . a_{-1} a_{-2} \dots a_{-m})_2 = (-1)^s \cdot (a_n 2^n + \dots + a_0 2^0 + a_{-1} 2^{-1} + a_{-2} 2^{-2} + \dots + a_{-m} 2^{-m})$$

❑ donde

○ **s** es el signo: 0 si el número es positivo y 1 si es negativo

○ **a_i** ∈ enteros y $0 \leq a_i \leq 1$, para todo $i = -m, \dots, -1, 0, 1, \dots, n$

❑ Distancia entre dos números consecutivos es 2^{-m} .

❑ Deja de ser un rango continuo de números para transformarse en un rango discreto

Notación científica

- ❑ Para el caso de los números reales se trabaja en ***notación científica***.

$$n = \pm f * 10^e$$

$$-725.832 = -7.25832 \cdot 10^2 = -725.832 \times 10^0$$

$$3.14 = 0.314 * 10^1 = 3.14 * 10^0$$

$$0.000001 = 0.1 * 10^{-5} = 1.0 * 10^{-6}$$

$$1941 = 0.1941 * 10^4 = 1.941 * 10^3$$

- ❑ Para unificar la representación se recurre a la ***notación científica normalizada***, en donde

$$0.1 \leq f < 1, \text{ y } e \text{ es un entero con signo}$$

Notación científica en binario

- ❑ En el sistema binario la expresión de un número en notación científica normalizada es

$$n = \pm f^* 2^e$$

- ❑ en donde $0.5 \leq f < 1$ y e es un entero con signo

Representación en Punto Flotante

- ❑ Se representan con los pares de valores (m, e) , denotando:

$$(m, e) = m \cdot b^e$$

- m llamado *mantisa*, y que representa un número fraccionario
- e , llamado *exponente*, al cual se debe elevar la base numérica (b) de representación para obtener el valor real

Representación en Punto Flotante

☐ Mantisa y exponente pueden representarse:

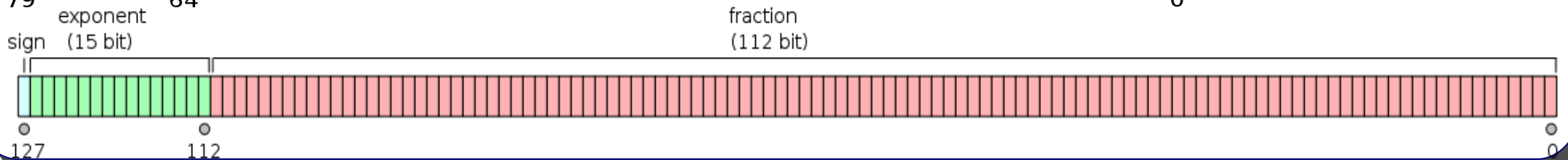
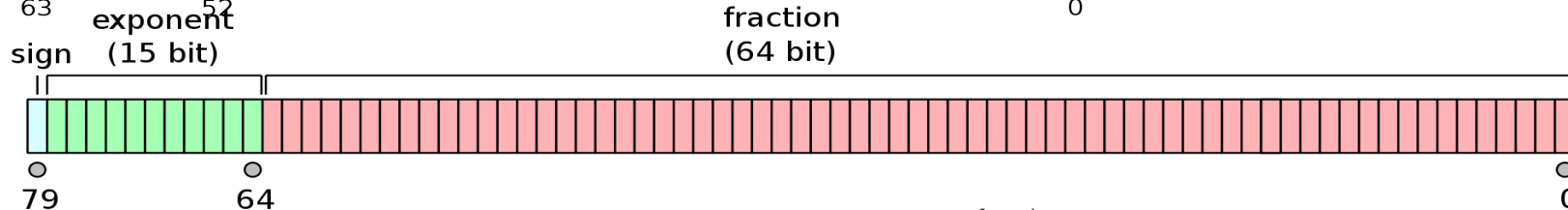
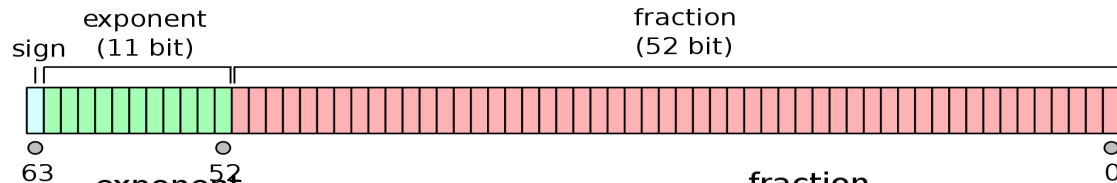
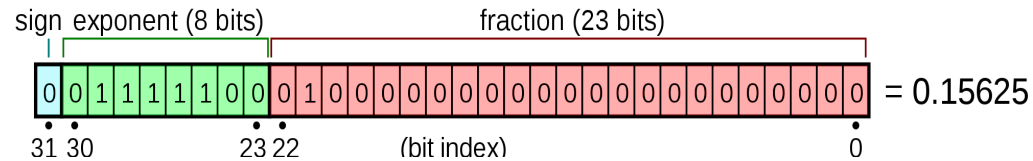
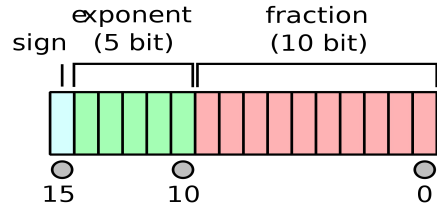
- ☐ con signo
- ☐ sin signo
- ☐ con notación complemento
- ☐ con notación exceso m.

☐ Para que las representaciones sean únicas, la mantisa deberá estar normalizada.

Punto Flotante: Formato IEEE 754

- ❑ IEEE (Institute of Electrical and Electronics Engineers.)
- ❑ El Standard IEEE 754 para punto flotante binario es el mas ampliamente utilizado. En este Standard se especifican los formatos para 32 bits, 64 bits, y 80-bits.
- ❑ En 2008 se introdujeron un formato de 16 bits y el de 80 fue reemplazado por uno de 128 bits (IEEE 754-2008)

Formatos IEEE 754



IEEE 754: Rangos

Data Type	Length	Precision (Bits)	Approximate Normalized Range	
			Binary	Decimal
Single Precision	32	24	2^{-126} to 2^{127}	1.18×10^{-38} to 3.40×10^{38}
Double Precision	64	53	2^{-1022} to 2^{1023}	2.23×10^{-308} to 1.79×10^{308}
Double Extended Precision	80	64	2^{-16382} to 2^{16383}	3.37×10^{-4932} to 1.18×10^{4932}

Representación de caracteres

- ❑ Además de representar números es necesario manipular mensajes y guardar información alfabética en un computador
- ❑ Por tal motivo se requiere poder guardar caracteres alfabéticos.
- ❑ Su representación sin duda alguna será a través de números binarios pero será imprescindible poder codificarlos mediante algún Standard

ASCII

- ☐ **American Standard Code for Information Interchange**
- ☐ Hay 95 caracteres ASCII imprimibles, numerados del 32 al 126.
- ☐ Es un código de caracteres basado en el alfabeto latino tal como se usa en inglés moderno y en otras lenguas occidentales.

ASCII

- ❑ Fue creado en 1963 por el Comité Estadounidense de Estándares (ASA, conocido desde 1969 como el Instituto Estadounidense de Estándares Nacionales, o ANSI) como una refundación o evolución de los conjuntos de códigos utilizados entonces en telegrafía.
- ❑ En 1967, se incluyeron las minúsculas, y se redefinieron algunos códigos de control para formar el código conocido como **US-ASCII**.

ASCII

- ❑ Utiliza 7 bits para representar los caracteres.
- ❑ Inicialmente empleaba un bit adicional (bit de paridad) para detectar errores en la transmisión.
- ❑ En la actualidad define códigos para 33 caracteres no imprimibles, de los cuales la mayoría son caracteres de control obsoletos que tienen efecto sobre como se procesa el texto, más otros 95 caracteres imprimibles que les siguen en la numeración (empezando por el carácter espacio).

ASCII

- ❑ Casi todos los sistemas informáticos actuales utilizan el código ASCII o una extensión compatible para representar textos y para el control de dispositivos que manejan texto.

Tablas ASCII

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`	128	80	Ç	160	A0	á	192	C0	Ł	224	E0	α
1	01	Start of heading	33	21	!	65	41	A	97	61	a	129	81	ù	161	A1	í	193	C1	ł	225	E1	β
2	02	Start of text	34	22	"	66	42	B	98	62	b	130	82	é	162	A2	ó	194	C2	ṽ	226	E2	Γ
3	03	End of text	35	23	#	67	43	C	99	63	c	131	83	â	163	A3	ú	195	C3	ṽ	227	E3	π
4	04	End of transmit	36	24	\$	68	44	D	100	64	d	132	84	ä	164	A4	ñ	196	C4	—	228	E4	Σ
5	05	Enquiry	37	25	%	69	45	E	101	65	e	133	85	å	165	A5	Ñ	197	C5	†	229	E5	σ
6	06	Acknowledge	38	26	&	70	46	F	102	66	f	134	86	ä	166	A6	*	198	C6	†	230	E6	μ
7	07	Audible bell	39	27	'	71	47	G	103	67	g	135	87	ç	167	A7	°	199	C7	‡	231	E7	τ
8	08	Backspace	40	28	(72	48	H	104	68	h	136	88	ê	168	A8	¿	200	C8	Ł	232	E8	φ
9	09	Horizontal tab	41	29)	73	49	I	105	69	i	137	89	ë	169	A9	ƒ	201	C9	ŕ	233	E9	θ
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j	138	8A	è	170	AA	ŕ	202	CA	Ł	234	EA	Ω
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k	139	8B	ì	171	AB	½	203	CB	ŕ	235	EB	δ
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l	140	8C	í	172	AC	¾	204	CC	‡	236	EC	∞
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m	141	8D	î	173	AD	ı	205	CD	=	237	ED	ø
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n	142	8E	Ï	174	AE	«	206	CE	†	238	EE	ε
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o	143	8F	Ä	175	AF	»	207	CF	±	239	EF	∏
16	10	Data link escape	48	30	0	80	50	P	112	70	p	144	90	É	176	B0	☐	208	D0	Ł	240	FO	≡
17	11	Device control 1	49	31	1	81	51	Q	113	71	q	145	91	æ	177	B1	☐	209	D1	ŕ	241	F1	±
18	12	Device control 2	50	32	2	82	52	R	114	72	r	146	92	Æ	178	B2	☐	210	D2	ŕ	242	F2	≥
19	13	Device control 3	51	33	3	83	53	S	115	73	s	147	93	ô	179	B3		211	D3	Ł	243	F3	≤
20	14	Device control 4	52	34	4	84	54	T	116	74	t	148	94	ö	180	B4		212	D4	Ł	244	F4	
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u	149	95	ò	181	B5		213	D5	ŕ	245	F5	
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v	150	96	û	182	B6		214	D6	ŕ	246	F6	÷
23	17	End trans. block	55	37	7	87	57	W	119	77	w	151	97	ù	183	B7		215	D7	†	247	F7	≈
24	18	Cancel	56	38	8	88	58	X	120	78	x	152	98	ÿ	184	B8		216	D8	†	248	F8	°
25	19	End of medium	57	39	9	89	59	Y	121	79	y	153	99	Ö	185	B9		217	D9	ŕ	249	F9	•
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z	154	9A	Û	186	BA		218	DA	ŕ	250	FA	·
27	1B	Escape	59	3B	;	91	5B	[123	7B	{	155	9B	ø	187	BB		219	DB	■	251	FB	√
28	1C	File separator	60	3C	<	92	5C	\	124	7C		156	9C	£	188	BC		220	DC	■	252	FC	∂
29	1D	Group separator	61	3D	=	93	5D]	125	7D	}	157	9D	¥	189	BD		221	DD	■	253	FD	ˆ
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~	158	9E	€	190	BE		222	DE	■	254	FE	■
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□	159	9F	f	191	BF		223	DF	■	255	FF	□

ISO 8859-1

ISO-8859-1																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL
8x	PAD	HOP	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
9x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	SGCI	SCI	CSI	ST	OSC	PM	APC
Ax	NBSP	ı	€	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ò	ñ	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	

Unicode

- ❑ Estándar industrial cuyo objetivo es proveer el medio por el cual se codifique para uso informático un texto en cualquier forma e idioma.
- ❑ El establecimiento de Unicode ha involucrado un ambicioso proyecto para reemplazar los esquemas de codificación de caracteres existentes, muchos de los cuales están muy limitados en tamaño y son incompatibles con entornos multilingües.

Unicode

- ❑ La última versión comprende una tabla de mas de 107.000 caracateres.
- ❑ Incuye lenguajes como el mandarín, árabe, hebreo, y demás lenguajes de caracteres “no latinos”.
- ❑ Implementado en un número considerable de tecnologías recientes, que incluyen XML, Java y Sistemas Operativos modernos.

UTF-8

- ❑ **UTF-8 (8-bit Unicode Transformation Format)** es una norma de transmisión de longitud variable para caracteres codificados utilizando Unicode.
- ❑ Usa grupos de bytes para representar el estándar de Unicode para los alfabetos de muchos de los lenguajes del mundo.
- ❑ Especialmente útil para la transmisión sobre sistemas de correo de 8 bits.

UTF-8

- ❑ Usa 1 a 4 bytes por carácter, en función del símbolo Unicode. P.ej. se necesita un solo byte en UTF-8 para codificar los 128 caracteres US_ASCII en el rango U+0000 a U+007F Unicode.
- ❑ No es afectado al utilizar compresión de datos.
- ❑ El IETF requiere que todos los protocolos de Internet indiquen qué codificación utilizan para los textos y que UTF-8 esté entre las mismas.

UTF-8: Ventajas

- ❑ Puede codificar cualquier carácter.
- ❑ Ahorra espacio respecto de UTF-16 o UTF-32 que utilizan muchos caracteres ASCII de 7 bits.
- ❑ Una secuencia de bytes para un carácter jamás será parte de una secuencia más larga de otro carácter como lo hacían viejas codificaciones.

UTF-8: Ventajas

- ❑ El primer byte de una secuencia multi-byte es suficiente para determinar la longitud de una secuencia multi-byte. Esto hace muy simple extraer una subcadena de una cadena dada sin elaborar un análisis exhaustivo.
- ❑ Está diseñado para que los bytes codificados nunca tomen alguno de los valores de los caracteres especiales de ASCII, previniendo problemas de compatibilidad con librerías y sistemas operativos legacy como la ANSI C.

UTF-8: Ventajas

- ☐ Las cadenas en UTF-8 pueden ser ordenadas usando rutinas de ordenamiento estándar orientadas a byte.
- ☐ UTF-8 es el valor predeterminado para el formato XML.

UTF-8: desventajas

- ❑ Es de longitud variable; eso significa que diferentes caracteres toman secuencias de diferentes longitudes para codificar. La agudeza de esto podría ser disminuida, sin embargo, creando una interfaz abstracta para trabajar con cadenas UTF-8 y haciéndolo transparente al usuario.

UTF-8: desventajas

- ❑ Un analizador de UTF-8 mal escrito podría aceptar un número de diferentes representaciones pseudo-UTF-8 y convertirlas en la misma salida Unicode.
- ❑ Los caracteres ideográficos usan 3 bytes en UTF-8, pero sólo 2 en UTF-16. Así, los textos chinos/japoneses/coreanos usarán más espacio cuando sean representados en UTF-8.

UTF-16

- ❑ Es un código de caracteres que proporciona una forma de representar caracteres Unicode e ISO/IEC 10646 como una serie de palabras de 16 bits y 24 bits susceptibles de ser almacenados o transmitidos a través de redes de datos.
- ❑ Se halla oficialmente definido en el Anexo Q de la norma ISO/IEC 10646-1.

UTF-16

- ❑ Está descripta en el estándar Unicode (versión 3.0 u superior), al igual que en la RFC 2781 de la IETF
- ❑ Representa un carácter que ha sido asignado dentro del conjunto de los 65536 puntos del código unicode o ISO/IEC 10646 como un valor de código único equivalente al punto de código del carácter: por ejemplo, 0 para 0, FFFD hexadecimal para FFFD.