

Representación de datos en la computadora

Bits

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

9 de septiembre de 2022



La memoria de la computadora

- 1 La memoria de la computadora
- 2 Representación de datos numéricos
- 3 Representación de datos alfanuméricos
- 4 Bibliografía

Temas

- 1 La memoria de la computadora
 - Bits

Bits

- La unidad de información en la computadora es el *bit*.
- Un bit puede tener uno de dos valores: verdadero/falso ó 0/1
- Su implementación física puede consistir en:
 - Corrientes que pasan o no pasan a través de un transistor,
 - orientaciones magnéticas perpendiculares en una cinta,
 - corrientes presentes o ausentes en condensadores,
 - otros sistemas donde dos estados sean fácilmente distinguibles.

Palabra

- Una palabra es una cadena de bits que es manejada en conjunto por la máquina.
- El tamaño de palabra histórico más tradicional es el *byte*: una palabra de 8 bits.
Ej.

10011010

- Actualmente se utilizan palabras de 16, 32 y 64 bits.

La memoria de la computadora

- La memoria de la computadora es como un muy largo arreglo de palabras^[1].
- Cada palabra tiene una dirección asociada.

Dirección		Valor
0	00000000	11010110
1	00000001	00011000
2	00000010	01011000
3	00000011	10001010
4	00000100	10111000
5	00000101	00011010
6	00000110	01001000
... 7	11111111	00011000

[1]Estrictamente pueden ser elementos divisores o múltiplos de una palabra

Interpretación

- Para realizar cálculos, es necesario interpretar las cadenas de ceros y unos como diferentes tipos de datos:
 - Numéricos
 - Alfanumérico
 - Tipos compuestos como: objetos.
 - Código
 - etc.

Representación de datos numéricos

- 1 La memoria de la computadora
- 2 Representación de datos numéricos
- 3 Representación de datos alfanuméricos
- 4 Bibliografía

Temas

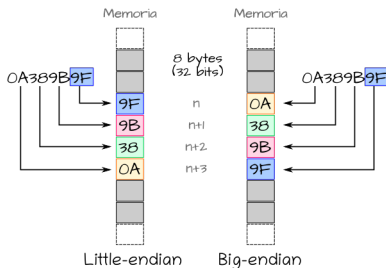
- 2 Representación de datos numéricos
 - Enteros en la computadora
 - Racionales en la computadora

Números grandes

Para representar números más grandes de lo que cabe en una palabra, se utilizan localidades de memoria adyacentes.

Little endian “Comienzo por el extremo pequeño”. El *byte* menos significativo se escribe primero.

Big endian “Comienzo por el extremo grande”. El *byte* más significativo se escribe primero.



Endianness

La nomenclatura de los criterios little-endian y big-endian proviene de la novela “Los viajes de Gulliver” de Jonathan Swift, hace referencia a una sociedad donde había dos grupos enemistados, uno sostenía que los huevos duros se tenían que empezar a comer por el extremo grande (big end) o otros por el pequeño (little end).

Wikipedia

* Un anillo

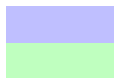
Un anillo cumple con 7 propiedades: Se definen dos operaciones, la suma $+$ y el producto \cdot .

- ① $a + b = b + a$
- ② $(a + b) + c = a + (b + c) = (a + c) + b$
- ③ $\exists 0_{\mathbb{Z}} | a + 0 = a \forall a \in \mathbb{Z}$
- ④ $\forall a \in \mathbb{Z}, \exists (-a) | a + (-a) = 0$
- ⑤ $ab = ba$
- ⑥ $a(bc) = (ab)c$
- ⑦ $\exists 1_{\mathbb{Z}} | a \cdot 1_{\mathbb{Z}} = a$
- ⑧ ~~inverso multiplicativo~~
- ⑨ $a(b + c) = ab + ac$

* \mathbb{Z}_4 el reloj de 4 horas

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

.	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1



Neutro aditivo

Neutro multiplicativo

Eje de simetría, garantiza conmutatividad

Enteros con signo

- Se requiere utilizar un bit para indicar el signo del número.
- Los sistemas más populares para representar enteros con signo:

$$\begin{array}{c} s \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{0} \mid \boxed{1} \mid \boxed{0} \mid \boxed{1} \mid \boxed{1} \mid \boxed{0} \mid \boxed{0} \mid \boxed{1} \end{array} = 89$$

- 1 Signo magnitud.

$$\begin{array}{c} s \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{1} \mid \boxed{1} \mid \boxed{0} \mid \boxed{1} \mid \boxed{1} \mid \boxed{0} \mid \boxed{0} \mid \boxed{1} \end{array} = -89$$

- 2 Corrimiento c.

$$\begin{array}{c} 2^7 \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{0} \mid \boxed{0} \mid \boxed{1} \mid \boxed{0} \mid \boxed{0} \mid \boxed{1} \mid \boxed{1} \mid \boxed{0} \end{array} \begin{array}{l} -127 \\ = -89 \end{array}$$

$$\begin{array}{c} s \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{0 \mid 1 \mid 0 \mid 1 \mid 1 \mid 0 \mid 0 \mid 1} = 89 \end{array}$$

④ Complemento a uno C_1^N .

$$\begin{array}{c} s \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{1 \mid 0 \mid 1 \mid 0 \mid 0 \mid 1 \mid 1 \mid 0} = -89 \end{array}$$

⑤ Complemento a dos $C_2^N = C_1^N + 1$.

$$\begin{array}{c} s \quad 2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0 \\ \boxed{1 \mid 0 \mid 1 \mid 0 \mid 0 \mid 1 \mid 1 \mid 1} = -89 \end{array}$$

- Permite sumar directamente.
- Sólo hay una representación para el cero.

Complementos a $b - 1$ y b

Se puede generalizar la noción de complemento.

$$C_{b-1}(r_b) = (b - 1_k \dots b - 1_1) - r_b \quad (1)$$

$$C_b(r_b) = (10_k \dots 0_1) - r_b \quad (2)$$

Por ejemplo:

$$C_9(193842_{10}) = 999999 - 193842 = 806157$$

$$C_{10}(193842_{10}) = 1000000 - 193842 = 806158$$

¿Quién es quién?

x_2	x_{10}	$SM(x)$	$C(x) - 3$	$C_1(x)$	$C_2(x)$
000	0	0	-3	0	0
001	1	1	-2	1	1
010	2	2	-1	2	2
011	3	3	0	3	3
100	4	-0	1	-3	-4
101	5	-1	2	-2	-3
110	6	-2	3	-1	-2
111	7	-3	4	-0	-1

4 Representaciones con $-(-x) = x$

x_{10}	x_2	SM(x)		C(x) - 3		$C_1(x)$		$C_2(x)$	
0	000	-0	100	0	011	-0	111	0	000
1	001	-1	101	-1	010	-1	110	-1	111
2	010	-2	110	-2	001	-2	101	-2	110
3	011	-3	111	-3	000	-3	100	-3	101
4	100	0	000	1	100	3	011	-4	100
5	101	1	001	2	101	2	010	3	011
6	110	2	010	3	110	1	001	2	010
7	111	3	011	4	111	0	000	1	001

Suma en \mathbb{Z}

Nótese lo que sucede con cada representación cuando se suman números positivos con negativos (nótese que sólo hay 3 bits disponibles)

$$3 + (-2) = \quad (3)$$

$$SM(x) = 011 + 110 = 1001 = 001 = 1 \quad \checkmark$$

$$C(x) = 110 + 001 = 111 = 4 \quad \times$$

$$C_1(x) = 011 + 101 = 1000 = 000 = 0 \quad \times$$

¡Desbordamiento! (Overflow)

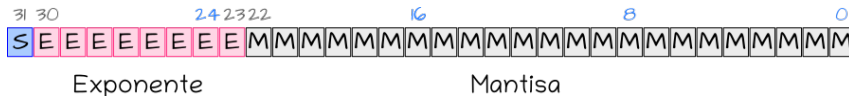
$$C_2(x) = 011 + 110 = 1001 = 001 = 1 \quad \checkmark$$

Temas

- 2 Representación de datos numéricos
 - Enteros en la computadora
 - Racionales en la computadora

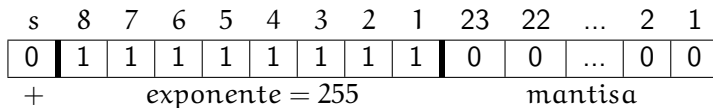
Números racionales

- Los números reales, en general, no pueden ser almacenados explícitamente.
- Sólo se usan números racionales.
- En Java se tienen dos opciones:
 - float 32 bits: 1 signo + 8 exponente + 23 mantisa
 - double 64 bits: 1 signo + 11 exponente + 52 mantisa (análogo a float, pero con más bits)

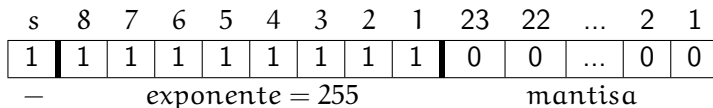


Precisión sencilla (float)

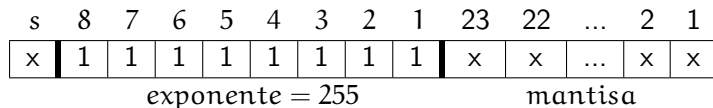
- ∞



- $-\infty$

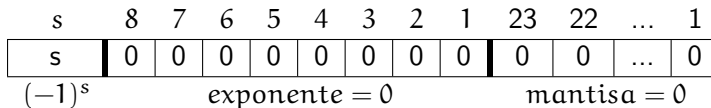


- NaN - no es un número



Precisión sencilla 2

- ± 0 .



Precisión sencilla 3 - normalizados

- $(-1)^s 2^{e-127} (1.f)$

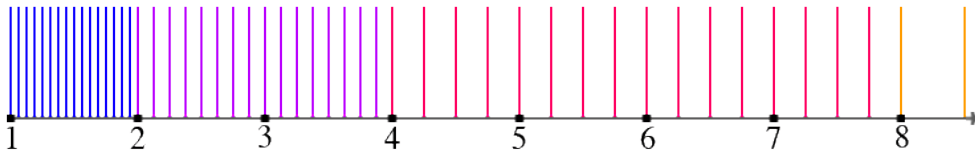
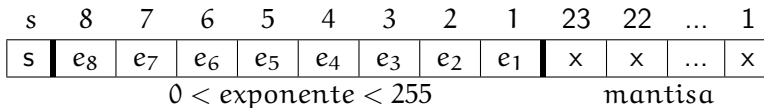


Figura: <http://ridiculousfish.com/blog/posts/float.html>

Precisión sencilla 4 - no normalizados

- $(-1)^s 2^{-126} (0.f)$ Números no normalizados.

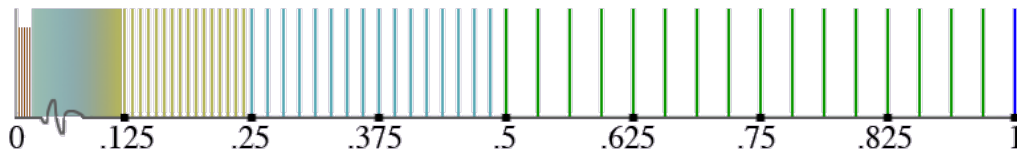
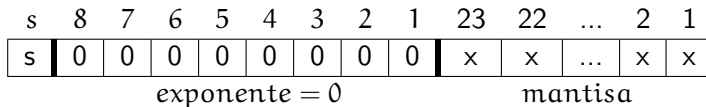


Figura: <http://ridiculousfish.com/blog/posts/float.html>

Floats

	signo (1 bit)	exponente (8 bits)	mantisa (23 bits)	
Cero	0	00000000	00000000000000000000000	0
Uno	0	11111111	00000000000000000000000	1
Menos uno	1	11111111	00000000000000000000000	-1
No normalizado más pequeño	*	00000000	00000000000000000000001	$\pm 1.4E-45 = \pm 2^{-126} \times 2^{-23} = 2^{-149}$ <code>Float.intBitsToFloat(i);</code>
No normalizado más grande	*	00000000	11111111111111111111111	$\pm 1.75E-38 = \pm 2^{-126} \times (1-2^{-23})$ <code>int num = (int) Math.pow(2, 23)-1;</code> <code>Float.intBitsToFloat(num)</code>
Normalizado más pequeño	*	00000001	00000000000000000000000	$\pm 1.75E-38 = \pm 2^{1-127} \times 1 = 2^{-126}$
Normalizado más grande	*	11111110	11111111111111111111111	$\pm 3.4E38 = \pm 2^{254-127} \times (2-2^{-23}) = 2^{127} \times (2-2^{-23})$
Infinito positivo	0	11111111	00000000000000000000000	∞ <i>Usar la serie geométrica</i>
Infinito negativo	1	11111111	00000000000000000000000	$-\infty$
Indefinido	*	11111111	~0	NaN

Ejemplos

- $+1.375 \times 2^{43}$

s	8	7	6	5	4	3	2	1	23	22	21	20	...
0	1	0	1	0	1	0	1	0	0	1	1	0	...
exponente = $170 - 127 = 43$									mantisa = 1.011_2 $= 1.375$				

- -1.375×2^{-85}

s	8	7	6	5	4	3	2	1	23	22	21	20	...
1	0	0	1	0	1	0	1	0	0	1	1	0	...
exponente = $42 - 127 = -85$									mantisa = 1.011_2 $= 1.375$				

Representación de datos alfanuméricos

- 1 La memoria de la computadora
- 2 Representación de datos numéricos
- 3 Representación de datos alfanuméricos**
- 4 Bibliografía

Temas

- 3 Representación de datos alfanuméricos
 - ASCII
 - Unicode

ASCII (original 8 bits)

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

Source: www.LookupTables.com

ASCII (extendido 16 bits)

128	Ç	144	É	160	á	176	⋮	192	⋈	208	⋈	224	α	240	≡
129	ü	145	æ	161	í	177	⋮	193	⊥	209	⊥	225	β	241	±
130	é	146	Æ	162	ó	178	⋮	194	⊥	210	⊥	226	Γ	242	≥
131	â	147	ô	163	ú	179		195	⊥	211	⊥	227	π	243	≤
132	ä	148	ö	164	ñ	180	⊥	196	—	212	⊥	228	Σ	244	∫
133	à	149	ò	165	ñ	181	⊥	197	⊥	213	⊥	229	σ	245	∫
134	â	150	û	166	²	182	⊥	198	⊥	214	⊥	230	μ	246	+
135	ç	151	ù	167	°	183	⊥	199	⊥	215	⊥	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	⊥	200	⊥	216	⊥	232	Φ	248	°
137	ë	153	Ö	169	⌈	185	⊥	201	⊥	217	⊥	233	⊙	249	.
138	è	154	Û	170	⌋	186	⊥	202	⊥	218	⊥	234	Ω	250	.
139	ï	155	◊	171	½	187	⊥	203	⊥	219	■	235	δ	251	√
140	î	156	£	172	¼	188	⊥	204	⊥	220	■	236	∞	252	∞
141	ì	157	¥	173	¡	189	⊥	205	=	221	■	237	φ	253	²
142	Ä	158	£	174	«	190	⊥	206	⊥	222	■	238	ε	254	■
143	Å	159	ƒ	175	»	191	⊥	207	⊥	223	■	239	∩	255	

Source: www.LookupTables.com

GUI con ASCII



Temas

- 3 Representación de datos alfanuméricos
 - ASCII
 - Unicode

Unicode

- Estándar de codificación de caracteres para textos de múltiples lenguas, técnicos y en lenguas muertas.
- Unicode incluye todos los caracteres de uso común en la actualidad y continúa expandiéndose.
- ¡Ya incluye emoticonos!
- Es el tipo de codificación utilizado por Java.

Unicode - UTF



Unicode define tres formas de codificación bajo el nombre UTF (Unicode transformation format: formato de transformación Unicode):

- UTF-8: codificación orientada a bytes con símbolos de longitud variable (de 1 a 4 bytes por carácter Unicode).
 - UTF-8 ahorrará espacio de almacenamiento para textos en caracteres latinos.
 - Los caracteres ideográficos usan 3 bytes en UTF-8, los textos chinos, japoneses o coreanos ocupan más espacio cuando se representan en UTF-8.
- UTF-16: codificación de 16 bits de longitud variable optimizada para la representación del plano básico multilingüe (BMP).
- UTF-32: codificación de 32 bits de longitud fija, y la más sencilla de las tres.

Bibliografía

- 1 La memoria de la computadora
- 2 Representación de datos numéricos
- 3 Representación de datos alfanuméricos
- 4 Bibliografía**

Bibliografía I

-  Galaviz Casas, José (s.f.). *Sistemas numéricos y representación de números en una computadora*. Departamento de Matemáticas, Facultad de Ciencias, UNAM. URL: <http://mmc.geofisica.unam.mx/acl/MaterialCursos/SistemasNumericos/SistemasNumericosyRepresentacionDeNumerosEnUnaComputadora.pdf>.
-  Ridiculous Fish (26 de sep. de 2005). *Float*. URL: <https://ridiculousfish.com/blog/posts/float.html>.

Licencia

Creative Commons
Atribución-No Comercial-Compartir Igual

