Tema 10: Representación de información en la computadora

Existen 10 tipos de personas. Las que saben leer binario y las que no"

Dos de los aspectos más importantes que se presentan en Informática, relacionados con la información, es cómo *representarla* y cómo *materializarla* o *registrarla físicamente.*

En la representación al interior de las computadoras, se consideran principalmente, dos tipos de información: **texto y números.** Cada uno de ellos presenta características diferentes.

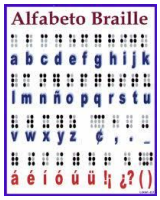
El objetivo de este tema es comprender los procesos que **transforman** la información externa a la computadora en patrones de bits almacenables y procesables por los elementos internos de la misma.

Representación de la información

Cuando se pretende plasmar la información de una forma transmisible y más o menos permanente, se debe disponer de un soporte físico adecuado, el cual contenga a la información. Existe una variedad de soportes físicos y algunos muy modernos, pero un medio que sigue en plena vigencia es la ESCRITURA.

Han evolucionado los métodos, pero el fundamento sigue siendo el mismo: **poner en la secuencia conveniente una serie de símbolos escogidos dentro de un conjunto predefinido**.

La información se representa en base a **cadenas de símbolos**. En base a un alfabeto convencional cualquiera, sobre el que se establece un acuerdo cultural de entendimiento entre el que escribe y el que lee, podemos representar cualquier información compuesta de palabras y cantidades numéricas.

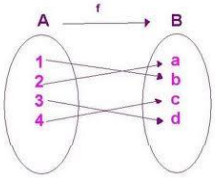


Un *alfabeto* es un conjunto de símbolos elementales, fijado por acuerdo cultural, en base a los cuales se forma la información. Cualquier alfabeto se fija arbitrariamente, y esto es muy importante. Gracias a este concepto la Informática ha logrado el tratamiento automático de la información con máquinas.

No es necesario que el alfabeto que usa una máquina en su interior sea el mismo que utiliza el hombre que la ha construido y la maneja, basta con que la transformación de los símbolos internos a los externos o viceversa se efectúe de una manera sencilla, de ser posible automáticamente por la propia máquina.

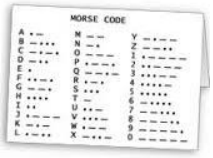
Codificación de la información

Cuando una información representada en un alfabeto **A** se transcribe a un segundo alfabeto **B,** se dice que ha sido **codificada**.



El caso más sencillo es cuando ambos alfabetos tienen la misma cantidad de símbolos y a cada símbolo del primer alfabeto le corresponde un símbolo del segundo alfabeto (correspondencia biunívoca o biyectiva).

Otro caso es cuando el segundo alfabeto dispone de un número de símbolos menor que el alfabeto de origen. En este caso no se puede recurrir a una correspondencia de símbolos uno a uno y será necesario transcribir (codificar) cada símbolo del conjunto **A1** con una combinación de símbolos del conjunto **A2.**



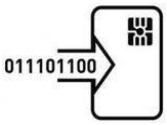
Un ejemplo es el sistema de codificación Morse. Este código dispone tan solo de dos elementos: el **punto** y la **raya**. Esto se debió a razones técnicas, dado que distinguir más de dos niveles de pulsación (corto = punto; largo = raya) hubiera sido totalmente inoperante, los mensajes hubieran estado sometidos a una enorme cantidad de subjetivismo y malas interpretaciones.

Existen razones que determinan la necesidad de que la información sea codificada y ellas son:

a) Posibilitar la transmisión automática de la información.

b) Necesidad de abreviar la escritura.

c) Hacer secreta e ininteligible la información que se codifica. Se trata de hacer críptico un mensaje plasmándolo en un sistema de codificación que el emisor y el receptor conocen pero que un posible interceptor desconocerá.



Codificar significa transformar unos datos de su representación actual a otra representación predefinida y preestablecida, que podrá ser tan arbitraria y convencional como se quiera, pero que deberá tener en cuenta el soporte físico sobre el cual se mantienen los datos, así como los procesos a los cuales se los deberá someter y, también, si será necesario transmitirlos a través de ciertos canales físicos de comunicación.

Sistemas de codificación binarios

Cuando los símbolos de un alfabeto **A1** son transcritos a un alfabeto **A2** que sólo tiene dos símbolos, se dice que es **sistema de codificación binario**.

El motivo para utilizar este alfabeto de codificación es de tipo técnico. Existen dificultades al usar dispositivos físicos que puedan detectar, con el debido grado de precisión, más de dos estados claramente diferenciados, en cualquier circunstancia y frente a cualquier posible perturbación. Se debe recurrir, por lo tanto, a dispositivos físicos **biestables** (presentan dos estados físicos diferenciados en forma clara y estable). Por ejemplo:

Corriente eléctrica: Distinguir entre diez o más niveles de voltaje o intensidad es complicado, en cambio, distinguir entre dos extremos de *pasa / no pasa* corriente, es más sencillo y económico.

Intensidad de la luz: Es muy difícil discernir a simple vista entre varias intensidades de luz, sin embargo, se pueden identificar claramente dos situaciones extremas: *luz apagada / luz encendida.*

Sentido de la magnetización: De la misma manera, diferenciar entre los diferentes valores que puede asumir un campo magnético es más difícil que diferenciar entre una magnetización *norte-sur* y su contraria.

Ejemplo: Un anillo de ferrita es capaz de adoptar dos estados de magnetización distintos y estables, se puede emplear para almacenar una información elemental: sólo hay que asignar arbitrariamente los valores "0" y "1" a esos estados

Códigos de representación de la información en las computadoras

Los datos ó cualquier información que se maneja internamente en un sistema informático están representados por números binarios según un método preestablecido.

En esta asignatura se estudiará solamente la representación de texto y de números, en particular, los números enteros, y sus distintos métodos de representación, que se muestran en la Tabla 1.

Tabla 1: Códigos de representación de información

TEXTO	▪ **EBCDIC** ▪ **ASCII** ▪ **UNICODE**	
NÚMEROS	**Enteros**	▪ Módulo y Signo ▪ Complemento a 1 ▪ Complemento a 2 ▪ Exceso a 2 elevado a N-1
	Reales (Coma Flotante)	IEEE754: simple precisión y doble precisión.

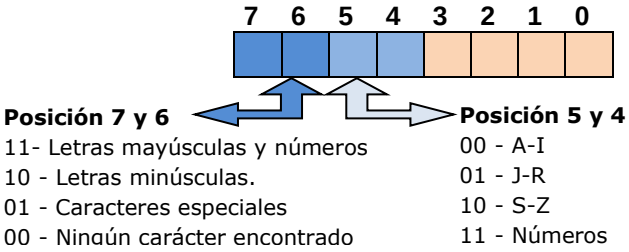
1. Representación de TEXTO

La computadora utiliza codificación de caracteres de texto para almacenar los datos alfanuméricos, tales como, nombre y apellido, dirección, etc. Cuando se presiona una tecla cualquiera en la computadora, ésta convierte el impulso eléctrico que proviene del teclado en un conjunto de bits. Para realizar esta transformación se utilizaron y se utilizan distintos códigos.

1.1. EBCDIC: Código de Intercambio Decimal Codificado en Binario Extendido

Es un sistema de codificación de 8 bits, donde cada carácter se representa como una cadena de 8 dígitos binarios y hay un total de $\mathbf{2^8}$ (256) caracteres a disposición. En la actualidad es poco utilizado pero es interesante para comprender le método de codificación.

Cada carácter codificado o **byte,** se divide normalmente en **cuatro bits de zona** (**7 6 5 4**), y **cuatro bits numéricos** (**3 2 1 0**), de pesos: 8-4-2-1 (el valor decimal resultante del producto del dígito por la base 2 elevado a la posición del dígito). En la Tabla 2 se muestran los 256 símbolos o caracteres codificados en EBCDIC.



Ejemplos:

Letra **A** mayúscula = 1100 0001
Letra **k** minúscula = 1001 0010
Número **5** = 1111 0101

Bits numéricos

Bits zona

Tabla 2: Código EBCDIC de 8 bits

bits 3210 / 7654 0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	DEL	DS		SP	&	-						{	}	\	0
0001	SOH	DC1	SOS						a	j	~		A	J		1
0010	STX	DC2	FS	SYN					b	k	s		B	K	S	2
0011	ETX	DC3							c	l	t		C	L	T	3
0100	PF	RES	BYP	PN					d	m	u		D	M	U	4
0101	HT	NL	LF	RS					e	n	v		E	N	V	5
0110	LC	BS	EOB	UC					f	o	w		F	O	W	6
0111	DEL	IL	ESC	EOT					g	p	x		G	P	X	7
1000		CAN							h	q	y		H	Q	Y	8
1001	RLF	EM						\	i	r	z		I	R	Z	9
1010	SMM	CC	SM		¢	!	¦	:								
1011	VT				.	$	'	#								
1100	FF	IFS		DC4	<	*	%	@								
1101	CR	IGS	ENQ	NAK	(	)	_	´								
1110	SO	IRS	ACK		+	;	>									
1111	SI	IUS	BEL	SUB	|	¬	?	"								

1.2. ASCII: Código Estándar Americano para el Intercambio de Información

El código ASCII básico utiliza 7 bits para representar los caracteres. Como otros códigos de formato de representación de caracteres, es un método para una correspondencia entre cadenas de bits y una serie de símbolos (alfanuméricos y otros), permitiendo de esta forma la comunicación entre dispositivos digitales. El código de caracteres ASCII, o una extensión compatible, se usa casi en todas las computadoras.

Existen otras versiones ampliadas de este código que utilizan 8 bits y respetan los códigos normalizados del ASCII básico, aprovechando las combinaciones no usadas para representar símbolos adicionales. Entre ellas se encuentran los códigos ISO 8859-n, donde n es el número que identifica el juego de los nuevos caracteres introducidos dependiendo de los lenguajes. Por ejemplo, la norma, ISO 8859-1, también denominada ISO-Latin1, se proyectó para América y Europa occidental e incluye vocales con acentos, tildes, diéresis y otras letras latinas no usadas en los países anglosajones.

El código ASCII reserva los primeros 32 códigos (numerados del 0 al 31 en decimal) para **caracteres de control**: códigos pensados para controlar los dispositivos. Por el avance tecnológico la mayoría de ellos ya no se utiliza, excepto LF (decimal 10) representa la función "nueva línea" (*line feed*) que hace que una impresora avance el papel, el CR que indica el “retorno de carro” (tecla *Enter*), y el símbolo ESC (decimal 27) representa la tecla "Escape" que se utiliza para cancelar acciones.

Los códigos del 33 al 126 se conocen como **caracteres imprimibles**, y representan letras, dígitos, signos de puntuación y varios símbolos. El código 32 el carácter espacio, designa al espacio entre palabras, y se produce cuando se oprime la barra espaciadora de un teclado.

En la tabla 3 podemos ver que cada carácter tiene una representación interna dada por un conjunto determinado de bits, que también pueden expresarse en decimal y hexadecimal, siguiendo las reglas de conversión entre sistemas. Por ejemplo: al carácter ‘**J**’ le corresponde el código ASCII decimal **74** que en hexadecimal se representa como **4A**, y en binario como **01001010**, como se puede comprobar en la tabla.

El decimal 74 se obtiene sumando los valores decimales de la intersección de la fila y la columna en la que se encuentra el carácter “j”, esto es, al decimal 64 de la fila 5 se le suma el decimal 10 de la columna 11.

Tabla 3: Código ASCII de 7 bits

Bits zona

Bits numéricos

Hex	Dec	Bits	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
		Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0	0000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
10	16	0001	DEL	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
20	32	0010	SP	!	“	#	$	%	&	‘	(	)	*	+	’	-	.	/
30	48	0011	0	1	2	3	4	5	6	7	8	9		;	<	=	>	?
40	64	0100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	80	0101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	96	0110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	112	0111	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	DEL

Caracteres ASCII Extendido:
El código ASCII de 7 bits contiene 128 caracteres y contiene todos lo necesarios para escribir en idioma inglés. En 1986, se modificó el estándar para agregar nuevos caracteres latinos, necesarios para la escritura de textos en otros idiomas, como por ejemplo el español, así fue como se agregaron los caracteres que van del ASCII 128 al 255, denominados ASCII Extendido.

En la tabla 4 se puede ver que las letras propias del español, como la ñ, se codifican como decimal 164 y 165, respectivamente.

Las letras acentuadas como la á minúscula se representa con el decimal 160, la í es 161, la ó es 162 y la ú es 163.

Tabla 4: Código ASCII extendido de 8 bits

128	Ç	144	É	160	á	176	░	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	│	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	169	_	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	█	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

Ingreso de caracteres por teclado:
No todos los símbolos que necesitamos utilizar se encuentran disponibles en el teclado, o el teclado tiene una configuración diferente. En este caso, se puede ingresar códigos ASCII utilizando su valor decimal, de la siguiente manera:

- Para ingresar el símbolo corchete "[": Presionar la tecla "Alt" en el teclado y el número "91" (sin dejar de presionar Alt).
- Para ingresar la letra "Ñ", presionar la tecla ALT y el número "165"

1.3. UNICODE

Este código fue propuesto por un consorcio de empresas y entidades con el objetivo de representar texto de muy diversas culturas.

Los códigos anteriores presentan varios inconvenientes, tales como:

- Los símbolos son insuficientes para representar los caracteres especiales que requieren numerosas aplicaciones.
- Los símbolos y códigos añadidos en las versiones ampliadas a 8 bits no están normalizados.
- Los lenguajes escritos de diversas culturas orientales, como la china, japonesa y coreana se basan en la utilización de ideogramas o símbolos que representan palabras, frases o ideas completas, siendo, por tanto, inoperantes los códigos que sólo codifican letras individuales.

Unicode está reconocido como estándar **ISO/IEC 10646**, y presenta las siguientes propiedades:

- *Universalidad*, trata de cubrir la mayoría de lenguajes escritos: 16 bits → 65.536 símbolos.
- *Unicidad*, a cada carácter se le asigna exactamente un único código.
- *Uniformidad*, ya que todos los símbolos se representan con un número mínimo de bits (16).

Unicode trata los caracteres alfabéticos, ideográficos y símbolos de forma equivalente, lo que significa que se pueden mezclar en un mismo texto sin la introducción de marcas o caracteres de control. La tabla 5 muestra un esquema de cómo se han asignado los códigos Unicode.

Tabla 5: Esquema de asignación de códigos en Unicode

Zona	Códigos		Símbolos codificados	N° de caracteres
A	0000	0000 00FF	Código ASCII Latín-1	256
			Otros alfabetos	7.936
		2000	Símbolos generales y caracteres fonéticos chinos, japoneses y coreanos	8.192
I	4000		Ideogramas	24.576
O	A000		Pendiente de asignación	16.384
R	E000 FFFF		Caracteres locales y propios de los usuarios Compatibilidad con otros códigos	8.192

UTF-8 (8-bit Unicode Transformation Format) es un formato de codificación de caracteres Unicode e ISO 10646 utilizando símbolos de longitud variable. Se usa mucho en la web debido a que contiene todos los caracteres necesarios para la representación HTML del código de una página.

Video: Representación digital de textos - Alberto Prieto Espinosa
https://www.youtube.com/watch?v=86f47DCTMms&t=488s

2. Representación de Números

Sistemas de representación restringidos a n bits

Las computadoras disponen de registros y buses de tamaños específicos que limitan la cantidad de bits disponibles para la representación de los datos. Es habitual mencionar que el sistema trabaja con un número finito de **n** bits**,** denominado "**palabra".** La palabra puede ser de 8, 16, 32 bits. Por lo tanto, **las representaciones que se utilizan tienen limitaciones**, y los **cálculos están siempre sujetos a aproximaciones** y por ende **a errores**.

Para caracterizar los sistemas de representación y compararlos, se definen dos parámetros importantes:

Capacidad de representación:
Es la cantidad de combinaciones distintas que se pueden representar. Por ejemplo, si el sistema está restringido a 5 bits, son 2^5 combinaciones de unos y ceros que permiten representar 32 elementos diferentes.

Rango:
El rango de un sistema está dado por el número mínimo y el número máximo representables. Por ejemplo, en binario sin signo con cinco dígitos el rango es [0, 31], donde "0" es el menor valor a representar y "31" el máximo.

Magnitud y signo: La notación habitual que se usa para representar cantidades numéricas con signo consiste en añadir un símbolo adicional para diferenciar un número positivo de uno negativo. Ej. 5 y -5.
En una computadora, la traducción directa de esta notación consiste en considerar a uno de los bits del número como bit de signo, por ejemplo, señalando a un número como positivo con un bit 0 y a un número negativo con un bit 1. Ej. 0111 (7) y 1111 (-7). Por tanto, la representación tiene: bit de signo y la magnitud del número.

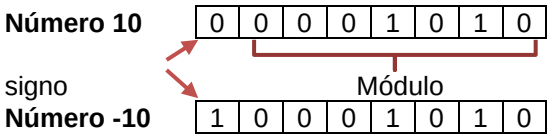
En general los números utilizados por la computadora se dividen en dos grandes grupos: los **enteros**[1] y los **reales**. Para los números reales se utiliza el método de *coma flotante*.

2.1. Representación de enteros

Existen diferentes formas para representar un número entero en una computadora. Hay que tener en cuenta que almacenar un número entero requiere que no sólo se almacenen los dígitos sino también el signo, o por lo menos que se utilice una forma de almacenamiento que permita reconocer los enteros positivos y los negativos. Los métodos son:

2.1.1. Módulo y signo (MS)

El bit que está situado más a la izquierda representa el signo, y su valor será de **0** para el positivo (+) y de **1** para el negativo (-). El resto de los bits (N-1) representan el módulo del número.



El rango de representación para un tamaño de palabra de N bits es:

$$-2^{N-1} + 1 <= X <= 2^{N-1} - 1$$

Para el caso de 8 bits, el rango se calcula reemplazando el tamaño de la palabra en la expresión:

$-2^{8-1} + 1 <= X <= 2^{8-1} - 1$

$-2^{7} + 1 <= X <= 27 - 1$

$-128 + 1 <= X <= 128 - 1$

$-127 <= X <= 127$

La siguiente tabla muestra los rangos de valores para tamaños de palabra de hasta 4 byte.

1 byte (8 bits)	**-127**	**<= X <=**	**+127**
2 byte (16 bits)	**-32767**	**<= X <=**	**+32767**
4 byte (32 bits)	**-2147483647**	**<= X <=**	**+2147483647**

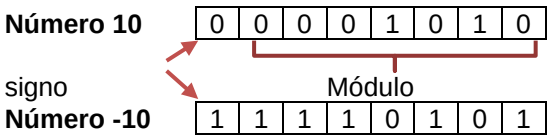
Presenta la ventaja de tener rango simétrico y la desventaja de tener dos representaciones del 0.

+0 = 0 0000000

–0 = 1 0000000

2.1.2. Complemento a 1 (C-1)

En este método, el bit de la izquierda corresponde al signo, **0** para el positivo y **1** para el negativo. Para los números positivos el resto de los bits (N-1) representan el módulo del número. El negativo de un número positivo se obtiene complementando todos los dígitos del módulo (cambiando ceros por unos y viceversa) y colocando el 1 en el bit de signo.



El rango de representación es de:

$$-2^{N-1} + 1 <= X <= 2^{N-1} - 1$$

Los rangos de valores para tamaños de palabra de hasta 4 byte.

1 byte (8 bits)	**-127**	**<= X <=**	**+127**
2 byte (16 bits)	**-32767**	**<= X <=**	**+32767**
4 byte (32 bits)	**-2147483647**	**<= X <=**	**+2147483647**

[1] Los números enteros comprenden a los números naturales, sus opuestos (negativos) y el cero.

Este sistema posee la ventaja de rango simétrico y la desventaja de tener dos representaciones del 0.

+ 0 = 0 0000000
– 0 = 1 11111111

2.1.3. Complemento a 2 (C-2)

Este sistema de representación utiliza el bit de la izquierda para el signo, correspondiendo el 0 para el positivo y el 1 para el negativo. Para los números positivos el resto de los bits (N-1) representan el módulo del número. El negativo de un número positivo se obtiene en dos pasos:

1°) Se complementa el módulo del número positivo en todos sus bits (cambiando ceros por unos y viceversa) incluido el bit de signo. Es decir, se realiza el "complemento a 1".

2°) Al resultado obtenido en el primer paso se le suma 1 (en binario) **despreciando el último acarreo si existe**. Ejemplo:

Número 10		0	0	0	0	1	0	1	0
complemento		**1**	**1**	**1**	**1**	**0**	**1**	**0**	**1**
	+								**1**
Número -10	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**0**

Un caso particular es el número -128, no puede ser representado siguiendo este método, y se le asigna la representación por convención.

El rango de representación es asimétrico, lo que constituye su mayor inconveniente y viene dado por:

$$-2^{N-1} <= x <= 2^{N-1} - 1$$

Se puede observar que el término de la izquierda no se suma el 1 dado que no existe el cero negativo.

Los rangos de valores para tamaños de palabra de hasta 4 byte son:

1 byte (8 bits)	**-128**	**<= X <=**	**+127**
2 byte (16 bits)	**-32768**	**<= X <=**	**+32767**
4 byte (32 bits)	**-2147483648**	**<= X <=**	**+2147483647**

La principal ventaja es la de tener una única representación del cero:

En el caso de palabras de 8 bits tendríamos:

Número 0		0 0 0 0 0 0 0 0
Número -0	*Primer paso*	1 1 1 1 1 1 1 1
	Segundo paso	1 1 1 1 1 1 1 1
	+	1
		1 0 0 0 0 0 0 0 0

El último acarreo se desprecia, por lo tanto, el 0 y el -0 tienen una sola representación.

2.1.4. Exceso a 2 elevado a N-1

Este método no utiliza ningún bit para el signo, todos los bits se utilizan para representar el valor del número. La idea es sumarles a todos los valores *un off-set* o *corrimiento*, excediendo el valor cero por $\mathbf{2^{(n-1)}}$, que se denomina “exceso”, donde **n** es la cantidad de bits que se utilizará para almacenar el número. El número entero en exceso $\mathbf{N_{(E)}}$ = **Exceso** + **N**, donde Exceso = $\mathbf{2^{(n-1)}}$ y N es el número a representar.

La idea del desplazamiento o «exceso» es que todos los bits 0 (valor 0) correspondan al máximo número negativo y todos los bits 1 (valor 255) al máximo número positivo

Número -128 → (-128 + 128) = 0: **0 0 0 0 0 0 0 0**

Número 127 → (127 + 128) = 255: **1 1 1 1 1 1 1 1**

Ejemplo: Para representar el número 10 con n = 8 bits el exceso es de 2^7 = 128, con lo que el número 10 se representa como 10 + 128 = 138; para el caso de -10 tendremos -10 + 128 = 118.

Número 10: 10001010

Número -10: 01110110

en este caso, el 0 tiene una única representación, que para 8 bits corresponde a:

Número 0 (0 + 128): 10000000

El rango de representación es asimétrico y viene dado por: $\mathbf{-2^{N-1} <= X <= 2^{N-1} -1}$

1 byte (8 bits)	**-128**	**<= X <=**	**+127**
2 byte (16 bits)	**-32768**	**<= X <=**	**+32767**
4 byte (32 bits)	**-2147483648**	**<= X <=**	**+2147483647**

Desbordamiento (Overflow)

Si tenemos 8 bits para representar un entero, tendremos un rango de representación de 0 a 255. ¿Qué sucede si se quiere representar el valor 256? NO SE PUEDE, se requieren 9 bits.

Se conoce como Overflow (desbordamiento) a la condición que se produce al sumar dos números a través de un determinado método de representación y una determinada cantidad de bits, cuyo resultado no puede ser representado utilizando la misma cantidad de bits, obteniéndose un resultado erróneo.

Sucede principalmente cuando se realiza una suma, y el resultado obtenido es mayor al máximo número representable en el sistema.

Ejemplo de overflow:

Decimal	Operación en módulo y signo	Decodificación
52	00110100	52
+ 97	+ 01100001	+ 97
149	10010101	-21

Dado que la suma supera el valor máximo de 127 para este método de representación en 8 bits, la ALU hace la suma colocando en el bit de signo una parte del resultado, produciendo un resultado inválido. La ALU posee una bandera o *flag* para indicar este tipo de errores.

Conclusiones sobre los sistemas de representación de enteros

El método más utilizado en la actualidad para la representación de enteros es el **Complemento a 2**, ello se debe a la facilidad de efectuar las sumas y restas con esta representación, porque en todos los casos, las operaciones se resuelven con sumas.

Este método reduce la complejidad de los circuitos de la unidad aritmética lógica, dado que no son necesarios circuitos específicos para restar.

En la tabla 6 se muestran los límites aproximados de valores enteros representables con distintas longitudes de palabras:

Tabla 6: Límites o rangos de representación para distintas longitudes de palabras

Longitud de palabra	Límite superior N (max)	Límite inferior N (min)	
		Complemento a 1	Complemento a 2
8	127	-127	-128
16	32.767	-32.767	-32.768
32	2.147.483.649	-2.147.483.649	-2.147.483.650
64	$9{,}223372 * 10^{18}$	$-9{,}223372 * 10^{18}$	$-9{,}223372 * 10^{18} -1$

Si como resultado de las operaciones, se obtiene un número fuera de los límites o rango, se dice que se ha producido un **desbordamiento u overflow**. La figura 1 muestra gráficamente el rango de representación de los números enteros.

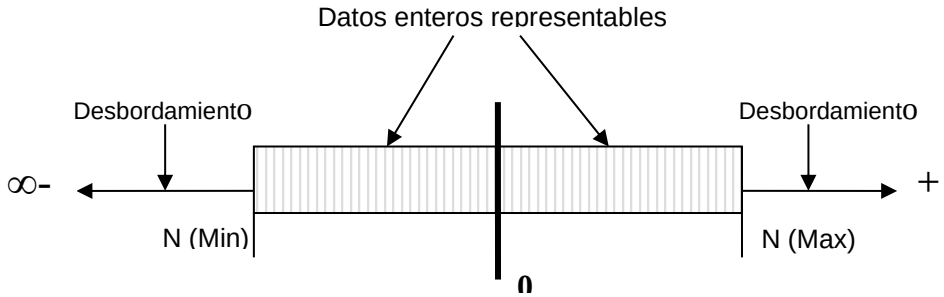


Figura 1: Rango de posibles representaciones de enteros

Video: Representación digital de números enteros - Alberto Prieto Espinosa
https://www.youtube.com/watch?v=pSPuGiQ0vdE&t=368s

2.2. Coma Flotante (Números Reales)

La representación de coma flotante, es una forma de notación científica usada en los procesadores (CPU) y otros componentes de procesamiento, con la cual se pueden representar números reales extremadamente grandes y pequeños de una manera muy eficiente y compacta, y con la que se pueden realizar operaciones aritméticas. El estándar para la representación en coma flotante es el IEEE 754.

Notación científica

La notación científica es un recurso matemático empleado para simplificar cálculos y representar en forma concisa números muy grandes o muy pequeños. Para hacerlo se usan potencias de la base del sistema En notación científica estándar, los números se expresan de la forma:

$$N = +- m\ E^{+-p} = +- m * b^{+- p}$$

Donde **m** es un número real**, b** es la base del sistema y **p** es un número entero, cuyo signo indica si la coma se desplaza a la derecha **(+)** o a la izquierda **(-).**

Ejemplos:

a) $-246{,}36_{10} = -2{,}4636\ E+2 = -2{,}4636 * 10^{2}$

b) $82000000000_{10} = 8{,}2\ E+10 = 8{,}2 * 10^{10}$

c) $0{,}00003 = 3{,}0\ E-5 = 3 * 10^{-5}$

d) $-1010000_{2} = -1{,}01\ E+6 = -1{,}01 * 2^{6}$

e) $0{,}0000012 = 1{,}0\ E-6 = 1{,}0 * 2^{-6}$

Normalización IEEE 754

Existen muchas maneras de representar números en formato de punto flotante. Cada uno tiene características propias en términos de rango, precisión y cantidad de elementos que pueden representarse. En un esfuerzo por mejorar la portabilidad de los programas y asegurar la uniformidad en la exactitud de las operaciones en este formato, el IEEE (Instituto de Ingeniería Eléctrica y Electrónica de Estados Unidos) creó un estándar que especifica cómo deben representarse los números en coma flotante con simple precisión (32 bits) o doble precisión (64 bits), y también cómo deben realizarse las operaciones aritméticas con ellos.

Precisiones usuales en IEEE754:

a) ***Simple Precisión (32 bits).*** El primer bit es el bit de signo (S), los siguientes 8 son los bits del exponente (E) y los restantes 23 son la mantisa (M).

Signo	Exponente	Mantisa
31	**30 - 23**	**22 - 0**

Exponente: Se destinan 8 bits para el exponente. Se utiliza la representación de entero en Exceso. El exceso se suma al exponente original, y el resultado es el que se almacena: E = EO + E

Mantisa: Utiliza 23 bits. Dado que la normalización implica un bit implícito, son 24 bits efectivos. La normalización toma la forma **1, bb..b**, donde **bb..b** representa los 23 bits de la mantisa que se almacenan. Cuando la mantisa se **normaliza** situando la coma decimal a la derecha del bit más significativo, dicho bit siempre vale **1**. Por tanto, se puede prescindir de él (bit implícito), y tomar en su lugar un bit más de la mantisa para aumentar la precisión del número.

b) ***Doble precisión (64 bits).*** El primer bit es el bit de signo (S), los siguientes 11 son los bits del exponente (E) y los restantes 52 son la mantisa (M).

Signo	Exponente	Mantisa
63	**62 - 52**	**51 - 0**

Los mecanismos de representación del exponente y la mantisa son similares al descripto en precisión simple.

Valores límites: Con toda representación se obtienen unos valores máximos y mínimos representables que para precisión simple son:

Número Mayor (N max)	$3{,}402823466.\ 10^{38}$
Número menor normalizado (N min, nor)	$1{,}2.\ 10^{-38}$
Número menor denormalizado (N min, den)	$1{,}1401.\ 10^{-45}$

En la figura 2 se muestra el rango de representación de los números reales. Obsérvese que los números reales que cumplen las siguientes condiciones no pueden ser representados:

- Los números comprendidos entre –N(min,den) y N(Min,den) con N distinto de cero. Si como resultado de una operación el número N está incluido en esa zona, se dice que se produce un **agotamiento** (*underflow*).
- Los números menores que –N(max) y mayores que N(Max) con N distinto de infinito positivo o negativo. Si como resultado de una operación el número N cae en esa zona, se dice que se produce un **desbordamiento** *(overflow).*

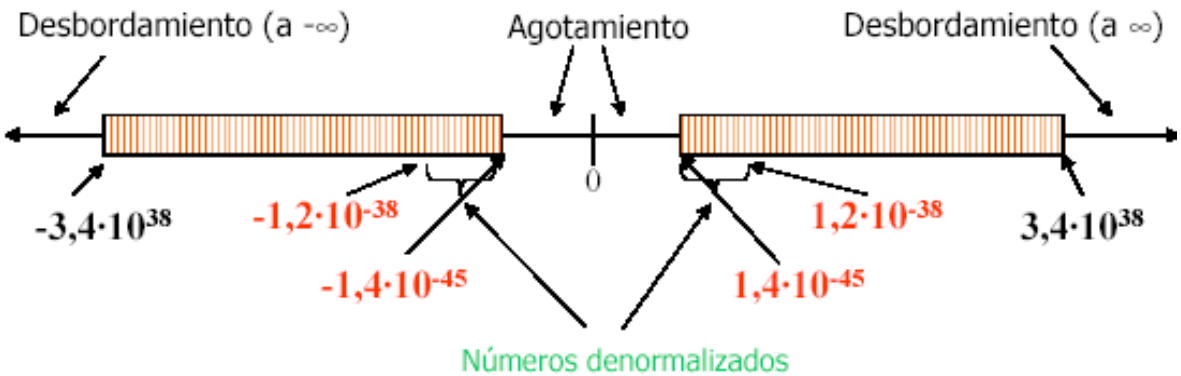


Figura 2: Rango de posibles representaciones de números reales

Observaciones finales

En importante que el programador tenga en cuenta cómo se almacenan los números en la computadora, ya que se pueden presentar problemas inherentes a la forma en que se representan los números (con un número limitado de bits).

Dificultades:

- Por la obtención, en resultados intermedios, de **números excesivamente pequeños.** Esto puede ocurrir por restar dos números extremadamente próximos entre sí o por la división entre números en los que el divisor es mucho mayor que el dividendo. En estos casos puede perderse la precisión de los cálculos o producirse un **agotamiento**.
- Por la obtención de **resultados numéricos excesivamente altos**, es decir por **desbordamiento**. Esto ocurre, por ejemplo, al dividir un número por otro mucho menor que él o al efectuar sumas o productos sucesivos con números muy elevados.
- En la comparación de dos números. Hay que tener en cuenta que cada dato real en la computadora representa a infinitos números reales (un intervalo de la recta real), por lo que en general una mantisa decimal no puede representarse exactamente con *n* bits, con lo que genera un error **"de representación".**
- Esto da lugar a problemas al comparar si un número es igual a otro (sobre todo si estos números se han obtenido por cálculos o procedimientos distintos), ya que la computadora considera que dos números son iguales únicamente si son iguales todos sus bits. La detección de igualdad se debe hacer con números enteros o considerando que dos números son iguales si la diferencia entre ellos es menor que un valor dado.
- Una consecuencia de lo dicho anteriormente es que, la suma y multiplicación de datos de tipo real no siempre cumplen las propiedades asociativas y distributivas, se pueden obtener resultados distintos dependiendo del orden en que se realizan las operaciones.

Video: Representación digital de números reales - Alberto Prieto Espinosa
https://www.youtube.com/watch?v=L2YUAIWXlco&t=38s