Andy Mendoza

Operaciones Fundamentales en Binario

El sistema de numeración binario, que están conformado solo por 0 y 1, es el pilar fundamental de los sistemas informáticos. Debido a su simplicidad y estar relacionado directamente con el estado de los dispositivos electrónicos (encendido o apagado), el sistema binario permite representar datos, realizar operaciones lógicas y controlar procesos digitales [1].

El sistema binario emplea el mismo principio de valor posicional que el sistema decimal, con la diferencia de que en lugar de usar la base 10, el sistema binario utiliza la base 2 conformada por 0 y 1. Para determinar el valor de cada posición de un número binario, se multiplica la potencia de la base que en este caso es 2 [2]. A continuación, en las Tablas 1 y 2 se muestran los valores decimales vinculados a cada posición de un número binario tanto en su parte entera como fraccionaria:

Tabla 1. Potencias Positivas

Potencias positivas	
$\mathbf{2^0}$	1
$\mathbf{2^1}$	2
$\mathbf{2^2}$	4
$\mathbf{2^3}$	8
$\mathbf{2^4}$	16
$\mathbf{2^5}$	32
$\mathbf{2^6}$	64
$\mathbf{2^7}$	128
$\mathbf{2^8}$	256

Tabla 2. Potencias negativas

Potencias negativas	
$\mathbf{2^{-1}}$	0.5
$\mathbf{2^{-2}}$	0.25
$\mathbf{2^{-3}}$	0.125
$\mathbf{2^{-4}}$	0.0625
$\mathbf{2^{-5}}$	0.03125
$\mathbf{2^{-6}}$	0.015625
$\mathbf{2^{-7}}$	0.0078125
$\mathbf{2^{-8}}$	0.00390625
$\mathbf{2^{-9}}$	0.001953125

En la Tabla 3 se muestra la equivalencia de un número del sistema decimal en sistema binario:

Tabla 3. Equivalencia de números del sistema decimal en el sistema binario

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

Importancia del sistema binario en el procesamiento de datos en computadoras

La representación binaria de los datos no solo es clave para guardar y enviar información digital, sino que es muy importante para mejorar el procesamiento dentro de los sistemas informáticos. Zhabin y Zhabina [3] demostraron que usar versiones del sistema binario, como los sistemas binarios redundantes que emplean los dígitos -1, 0 y 1, pueden llegar a acelerar considerablemente los cálculos en computadoras con arquitecturas paralelas.

Estas computadoras conectadas entre sí procesan los datos dígito por dígitos, comenzando por los más significativos, lo que se traduce en un menor tiempo de espera y en una mejora del rendimiento en tiempo real. Además, al trasmitir datos de forma binaria, se reduce el consumo de recursos de hardware como pines y conexiones internas [3]. Esto demuestra que el sistema binario puede ser una herramienta importante en el desarrollo de arquitecturas eficiente y que cuenten con un alto rendimiento para el procesamiento de datos.

Operaciones aritméticas básicas en el sistema binario

En el sistema de numeración binario se pueden realizar las operaciones aritméticas básicas siguiendo pequeñas reglas en cada operación. Estas operaciones son fundamentales en el procesamiento de datos, ya que permiten a las computadoras realizar cálculos utilizando los dígitos 0 y 1 [4].

Suma de números binarios

La suma de números binarios es un proceso importante en la aritmética digital, está basada en un conjunto de reglas simples que se derivan del sistema binario [4], como se puede observar en la Tabla 4:

Tabla 4. Reglas de la suma de binarios

Bit A	Bit B	Resultado	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Cuando sumamos 1 + 1 en binario, el resultado es 0 y se produce un acarreo o carry de 1 a la siguiente posición de mayor orden. Es similar cuando realizamos una suma en el sistema decimal que al obtener un número mayor que 9, por ejemplo 6 + 6 = 12, dejamos el 2 y llevamos el 1 a la siguiente posición. El carry es esencial en la arquitectura de los sumadores lógicos en procesadores digitales [4].

Ejemplo:

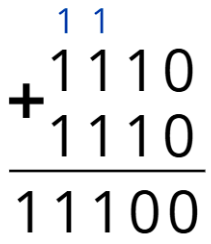

Figura 1. Suma de binarios

Resta de números binarios

Al igual que en la suma de binarios, la resta de binarios se basa en un conjunto específico de reglas del sistema binario [4], las cuales se pueden visualizar en la Tabla 5:

Tabla 5. Reglas de la resta de binarios

Bit A	Bit B	Resultado	Borrow
0	0	0	0
1	0	1	0
1	1	0	0
0	1	1	1

Cuando se realiza una resta binaria y tenemos la operación 0 – 1, se debe prestar un 1 o borrow de la siguiente posición más significativa. El borrow en binario es de suma importancia, ya que los circuitos encargados de la resta lo gestionan. Su correcta implementación es de vital importancia no solo para realizar operaciones aritméticas con números negativos, sino que también son esenciales para el diseño y funcionamiento eficiente de los microprocesadores que utilizan el complemento a dos [4].

Ejemplo:

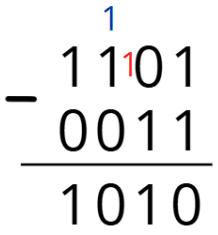

Figura 2. Resta de binarios

Multiplicación de números binarios

En la multiplicación binaria, se suma el multiplicando según el valor de cada bit del multiplicador, y se aplica un desplazamiento a la izquierda por posición [4]. En la Tabla 6 se muestran sus reglas:

Tabla 6. Reglas de la multiplicación de binarios

Bit A	Bit B	Producto
0	0	0
0	1	0
1	0	0
1	1	1

En la multiplicación de binarios no se utiliza directamente el carry o el borrow, sin embargo, los carries si son necesarios y deben ser manejados por el sistema durante la suma de los productos parciales [4].

Ejemplo:

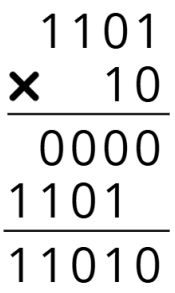

Figura 3. Multiplicación de binarios

División de números binarios

La división de números binarios se fundamenta en comparar segmentos del dividendo con el divisor, restar y desplazar bits [4]. En la Tabla 7 de detallan sus reglas básicas:

Dividendo	Divisor	Cociente	Resto
1	1	1	0
0	1	0	0
1	0	-	-
0	0	-	-

Tabla 7. Reglas de la división de binarios

Si el divisor es menor o igual que el conjunto de bits actual del dividendo, se anota un 1 en el cociente y se resta, caso contrario, se anota un 0 y se toma el siguiente bit del dividendo. Al restar se debe manejar el borrow cuando se lo requiera [4].

Ejemplo:

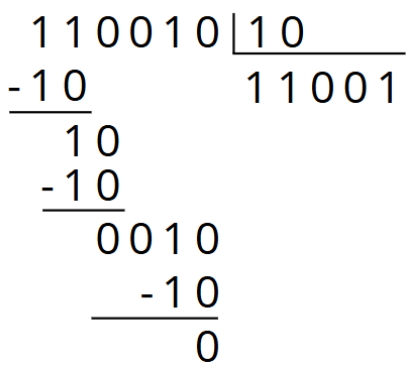

Figura 4. División de binarios

Códigos de Representación Numérica y No Numérica

Los sistemas digitales utilizan códigos binarios estandarizados para la representación de datos numéricos y no numéricos. Natarajan [5] explica que códigos como BCD y ASCII permiten transformar datos numéricos y alfanuméricos en secuencias binarias, adecuadas para el procesamiento por circuitos digitales. Estos códigos también simplifican el diseño de la lógica digital y permiten la interoperabilidad entre sistemas digitales y de comunicación.

Código ASCCI

El código ASCII (American Estándar Code for Information Interchange) es un método esencial y muy utilizado para representar caracteres como valores numéricos en sistemas computacionales. Funciona asignando un código numérico único a cada carácter permitiendo su procesamiento digital [5].

En [6], Park demuestra la rapidez con la que el código ASCI) convierte caracteres no numéricos, que luego se utilizan en el algoritmo DTW para el reconocimiento de texto en imágenes. Aunque tenga una gran velocidad y fácil implementación, puede presentar dificultades cuando los caracteres son parecidos como el 5 y la S.

En la Tabla 8 se puede visualizar el código ASCII:

Tabla 8. Código ASCII

Caracteres ASCII de control		Caracteres ASCII imprimibles						ASCII extendido							
00	NULL	**32**	Espacio	**64**	@	**96**	`	**128**	Ç	**160**	á	**192**	└	**224**	Ó
01	SOH	**33**	!	**65**	A	**97**	a	**129**	ü	**161**	í	**193**	┴	**225**	ß
02	STX	**34**	“	**66**	B	**98**	b	**130**	é	**162**	ó	**194**	┬	**226**	Ô
03	ETX	**35**	#	**67**	C	**99**	c	**131**	â	**163**	ú	**195**	├	**227**	Ò
04	EOT	**36**	$	**68**	D	**100**	d	**132**	ä	**164**	ñ	**196**	─	**228**	õ
05	ENQ	**37**	%	**69**	E	**101**	e	**133**	à	**165**	Ñ	**197**	┼	**229**	Õ
06	ACK	**38**	&	**70**	F	**102**	f	**134**	å	**166**	ª	**198**	ã	**230**	µ
07	BEL	**39**	‘	**71**	G	**103**	g	**135**	ç	**167**	º	**199**	Ã	**231**	þ
08	BS	**40**	(	**72**	H	**104**	h	**136**	ê	**168**	¿	**200**	╚	**232**	Þ
09	HT	**41**	)	**73**	I	**105**	i	**137**	ë	**169**	®	**201**	╔	**233**	Ú
10	LF	**42**	*	**74**	J	**106**	j	**138**	è	**170**	¬	**202**	╩	**234**	Û
11	VT	**43**	+	**75**	K	**107**	k	**139**	ï	**171**	½	**203**	╦	**235**	Ù
12	FF	**44**	,	**76**	L	**108**	l	**140**	î	**172**	¼	**204**	╠	**236**	ý
13	CR	**45**	-	**77**	M	**109**	m	**141**	ì	**173**	¡	**205**	═	**237**	Ý
14	SO	**46**	.	**78**	N	**110**	n	**142**	Ä	**174**	«	**206**	╬	**238**	¯
15	SI	**47**	/	**79**	O	**111**	o	**143**	Å	**175**	»	**207**	¤	**239**	´
16	DEL	**48**	0	**80**	P	**112**	p	**144**	É	**176**	░	**208**	ð	**240**	-
17	DC1	**49**	1	**81**	Q	**113**	q	**145**	æ	**177**	▒	**209**	Ð	**241**	±
18	DC2	**50**	2	**82**	R	**114**	r	**146**	Æ	**178**	▓	**210**	Ê	**242**	‗
19	DC3	**51**	3	**83**	S	**115**	s	**147**	ô	**179**	│	**211**	Ë	**243**	¾
20	DC4	**52**	4	**84**	T	**116**	t	**148**	ö	**180**	┤	**212**	È	**244**	¶
21	NAK	**53**	5	**85**	U	**117**	u	**149**	ò	**181**	Á	**213**	ı	**245**	§
22	SYN	**54**	6	**86**	V	**118**	v	**150**	û	**182**	Â	**214**	Í	**246**	÷
23	ETB	**55**	7	**87**	W	**119**	w	**151**	ù	**183**	À	**215**	Î	**247**	¸
24	CAN	**56**	8	**88**	X	**120**	x	**152**	ÿ	**184**	©	**216**	Ï	**248**	°
25	EM	**57**	9	**89**	Y	**121**	y	**153**	Ö	**185**	╣	**217**	┘	**249**	¨
26	SUB	**58**	:	**90**	Z	**122**	z	**154**	Ü	**186**	║	**218**	┌	**250**	·
27	ESC	**59**	;	**91**	[	**123**	{	**155**	ø	**187**	╗	**219**	█	**251**	¹
28	FS	**60**	<	**92**	\	**124**	|	**156**	£	**188**	╝	**220**	▄	**252**	³
29	GS	**61**	=	**93**	]	**125**	}	**157**	Ø	**189**	¢	**221**	¦	**253**	²
30	RS	**62**	>	**94**	^	**126**	~	**158**	×	**190**	¥	**222**	Ì	**254**	■
31	US	**63**	?	**95**	_	**127**	DEL	**159**	ƒ	**191**	┐	**223**	▀	**255**	nbsp

Conversión de texto a binario utilizando código ASCII

Para convertir un texto a binario separamos cada carácter y tomamos su equivalencia en el código ASCII que es un número decimal, luego este número lo convertimos a binario. ASCII utiliza 7 bits, pero a menudo se lo represente con 8 bits añadiendo un bit a la izquierda que por lo general es 0 [6].

Ejemplo:

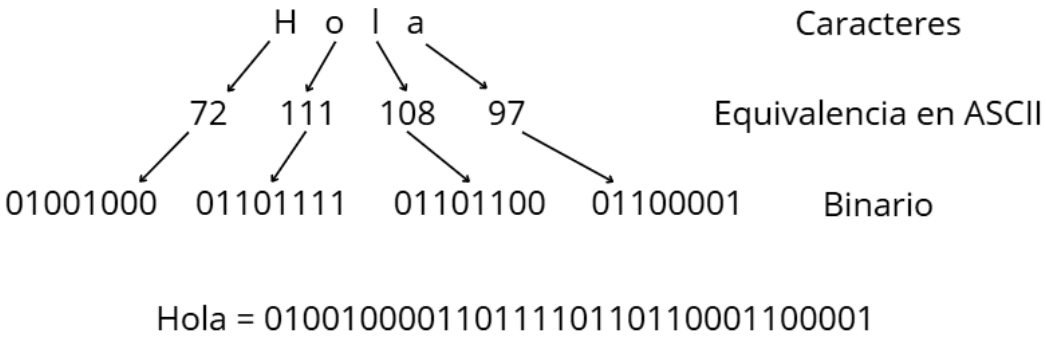

Figura 10. Conversión de texto a binario

Conversión de binario a texto utilizando código ASCII

Para convertir un número binario a texto, separamos en grupo de 8 bits de izquierda a derecha, en caso de que no se completen los 8 bits podemos descartar el grupo incompleto, o añadir ceros a la derecha. Luego convertimos cada grupo a sistema decimal y buscamos su equivalencia en el código ASCII [6].

Ejemplo:

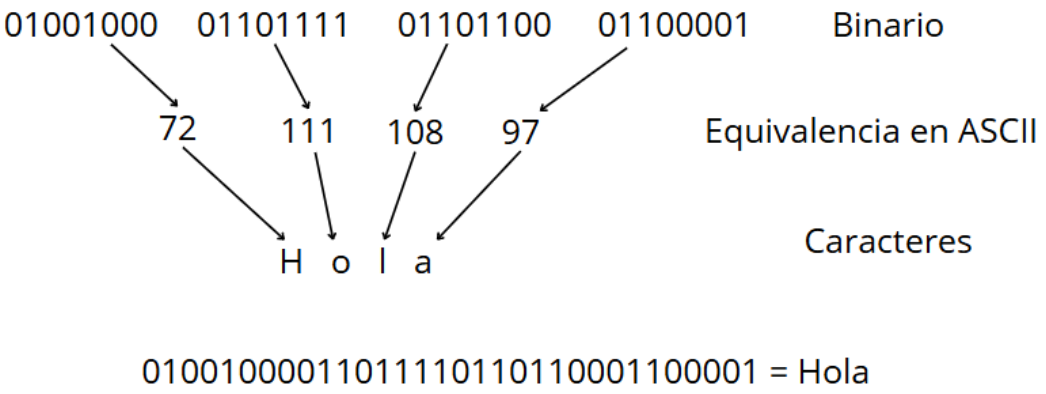

Figura 11. Conversión de binario a texto

BCD

El código BCD (Binary Coded Decimal) es un método muy utilizado para representar números decimales en formato binario. Es útil en contextos donde la exactitud decimal es esencial, como en calculadoras, relojes digitales y sistemas financieros. Este método asigna un bloque de 4 bits binarios a cada dígito decimal, del 0 al 9 [5]. Por ejemplo, 22 en BCD sería 0001 0110.

Según Natarajan [5], cada conjunto de 4 bits en BCD tiene un valor posicional fijo siguiendo el esquema 8421. Destaca que lo valores del 10 al 15 en binario no son válidos en la codificación BCD, lo que garantiza que los circuitos digitales interpreten los datos decimales de forma precisa y predecible por los circuitos digitales.

Bibliografía

[1] H. Li, “Binary System: Foundation of Modern Computing,” *Mathematica Eterna*, vol. 14, no. 1, 2024, doi: 10.35248/1314-3344.24.14.206.

[2] J. Salido Tercero, *Lógica digital y tecnología de computadores. Un enfoque práctico mediante simulación con Logisim*. España: Ediciones de la Universidad de Castilla-La Mancha, 2023. doi: 10.18239/manuales_2023.26.00.

[3] V. Zhabin and V. Zhabina, “Methods of On-Line Computation Acceleration in Systems with Direct Connection between Units,” in *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Kyiv: IEEE, May 2020, pp. 356–362. doi: 10.1109/DESSERT50317.2020.9125082.

[4] A. C. Jha, “Positional Number System,” *NUTA Journal*, vol. 7, no. 1–2, pp. 1–9, Dec. 2020, doi: 10.3126/nutaj.v7i1-2.39924.

[5] D. Natarajan, “Number Systems and Binary Codes,” in *Fundamentals of Digital Electronics*, vol. 623, Bengaluru: Springer International Publishing, 2020, ch. 5, pp. 95–119. doi: 10.1007/978-3-030-36196-9.

[6] H. J. Park, “A method to convert non-numeric characters into numerical values in dynamic time warping for string matching,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2660–2665, Jun. 2021, doi: 10.11591/ijece.v11i3.pp2660-2665.