Códigos Binarios

BIN	OCT	DEC	HEX	BCD	EXC-3	GRAY
0 0 0 0	0	0	0	0000	0011	0 0 0 0
0 0 0 1	1	1	1	0001	0100	0 0 0 1
0 0 1 0	2	2	2	0010	0101	0 0 1 1
0 0 1 1	3	3	3	0011	0110	0 0 1 0
0 1 0 0	4	4	4	0100	0111	0 1 1 0
0 1 0 1	5	5	5	0101	1000	0 1 1 1
0 1 1 0	6	6	6	0110	1001	0 1 0 1
0 1 1 1	7	7	7	0111	1010	0 1 0 0
1 0 0 0	10	8	8	1000	1011	1 1 0 0
1 0 0 1	11	9	9	1001	1100	1 1 0 1
1 0 1 0	12	10	A	1 0000	10 00011	1 1 1 1
1 0 1 1	13	11	B	1 0001	10 00100	1 1 1 0
1 1 0 0	14	12	C	1 0010	10 00101	1 0 1 0
1 1 0 1	15	13	D	1 0011	10 00110	1 0 1 1
1 1 1 0	16	14	E	1 0100	10 00111	1 0 0 1
1 1 1 1	17	15	F	1 0101	10 01000	1 0 0 0

Binario Codificado en Decimal (BCD)

- En el código BCD [Binary Coded Decimal = Decimal Codificado en Binario], cada dígito decimal está representado por un grupo de 4-bits, a esta agrupación se la denomina "quad".
- Cada quad tiene 4-bits [con ponderaciones: 8, 4, 2 y 1] con 10 valores permisibles de 0 a 9.
- En la codificación BCD, los quads con valores superiores a 9 [1010, 1011, 1100, 1101, 1110, 1111] no están permitidos, por tanto, nunca se usan en BCD.
- De modo que para representar el número 12_{10} en BCD sería 10010_{BCD}. Al código BCD se lo utiliza principalmente en diferentes tipos de medidores de panel, por ejemplo en voltimetros digitales.

Código Exceso de 3

- Puede decirse que el código exceso de 3 es una modificación del código BCD, puesto que el primero se forma añadiendo 3 al código BCD.
- Eventualmente se lo utiliza en lugar del BCD debido a que posee ventajas en algunas operaciones aritméticas.
- La tabla anterior muestra el código exceso de 3 y su equivalente BCD.

Código de Gray (Reflejado)

- Es un código binario en el que la posición del bit no tiene significación numérica [ponderación]; sin embargo, cada código de Gray corresponde a un mismo número decimal.
- Fácilmente se lo puede transformar a su equivalente binario.
- En la tabla anterior se presentan los códigos de Gray y binario natural para los números del 0 hasta el 15.
- Después se hace una comparación entre los dos códigos para determinar las relaciones que permitan convertir el uno en el otro y viceversa.

Código de Gray (Reflejado)

- Como puede verse en esta tabla, en el código de Gray, cuando el valor de un número cambia, la transición de un código al siguiente implica el cambio de un solo dígito a la vez.
- Por observación de la tabla, puede decirse que la conversión del código de Gray al código binario se realiza de la siguiente manera: El bit correspondiente al extremo izquierdo ["MSB"] es el mismo tanto en el código de Gray como en el binario; al continuar hacia la derecha, si el siguiente bit de Gray es "1", entonces el próximo bit binario es el complemento del anterior bit binario. Pero si el siguiente bit de Gray en "0", entonces el próximo bit binario es la copia del bit binario anterior.

Código de Gray (Reflejado)

Para reducir la probabilidad de que un circuito digital malinterprete una entrada cambiante, se desarrolló el código Gray como una manera de representar una secuencia de números. El aspecto único del código Gray es que, entre dos números sucesivos en la secuencia sólo un bit cambia. La tabla muestra la traducción entre el valor del código binario de tres bits y el código Gray. Para convertir de binario a Gray sólo hay que empezar en el bit más significativo y usarlo como el MSB de Gray, como muestra la figura 2-2(a). Después se compara el MSB binario con el siguiente bit binario (B1). Si son iguales, entonces G1 = 0; si son distintos, entonces G1 = 1. Para encontrar G0 se compara B1 con B0.

TABLA
Equivalencia entre el código binario de tres bits y el código Gray.

B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Código de Gray (Reflejado)

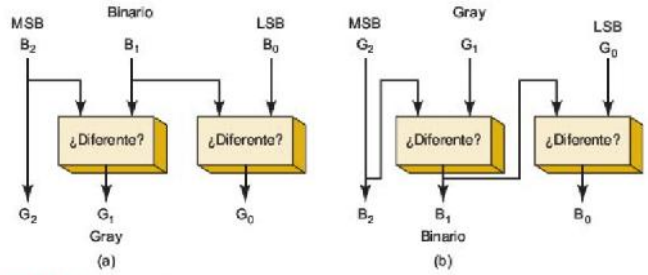
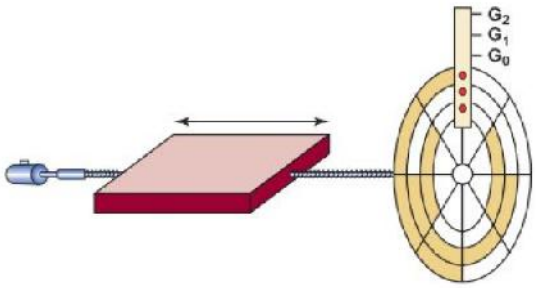


FIGURA Conversión de (a) binario a Gray y de (b) Gray a binario.

La figura muestra la conversión del código Gray a binario. Observe que el MSB en Gray siempre es el mismo que el MSB en binario. El siguiente bit binario se encuentra comparando el bit *binario* a la izquierda con el *bit correspondiente en código Gray*. Los bits similares producen un 0 y los bits distintos un 1. La aplicación más común del código Gray es en los codificadores de posición de eje, como muestra la figura . Estos dispositivos producen un valor binario que representa la posición de un eje mecánico giratorio. Un codificador de eje práctico utiliza mucho más de tres bits y divide la rotación en mucho más de ocho segmentos, por lo que puede detectar incrementos de rotación mucho más pequeños.

Código de Gray (Reflejado)

FIGURA Un codificador de eje de ocho posiciones y tres bits.



Byte, Nibble y Palabra

Bytes

La mayoría de las microcomputadoras maneja y almacena datos binarios e información en grupos de ocho bits, por lo que una cadena de ocho bits tiene un nombre especial: **byte**. Un byte consiste de ocho bits y puede representar cualquier tipo de datos o de información.

Nibbles

A menudo los números binarios se descomponen en grupos de cuatro bits, como hemos visto con los códigos BCD y las conversiones a números hexadecimales. En los primeros días de los sistemas digitales surgió un término para describir un grupo de cuatro bits. Como abarca la mitad de un byte, se le denominó **nibble**.

Palabras

Los términos bit, nibble y byte representan un número fijo de dígitos binarios. A medida que los sistemas han ido creciendo a través de los años, también ha crecido su capacidad (¿apetito?) de manejar datos binarios. Una **palabra** es un grupo de bits que representa una cierta unidad de información. El tamaño de la palabra depende del tamaño de la ruta de datos en el sistema que utiliza la información. El **tamaño de palabra** puede definirse como el número de bits en la palabra binaria con el que opera un sistema digital. Por ejemplo, tal vez la computadora en su horno de microondas sólo pueda manejar un byte a la vez. Tiene un tamaño de palabra de ocho bits. Por otro lado, la computadora personal en su escritorio puede manejar ocho bytes a la vez, por lo que tiene un tamaño de palabra de 64 bits.

Códigos Alfanuméricos

Además de los datos numéricos, una computadora debe ser capaz de manejar información no numérica. En otras palabras, una computadora debe reconocer códigos que representen letras del alfabeto, signos de puntuación y otros caracteres especiales, además de los números. A estos códigos se les denomina **códigos alfanuméricos**. Un código alfanumérico completo debe incluir las 26 letras minúsculas, las 26 letras mayúsculas, los 10 dígitos numéricos, 7 signos de puntuación y de 20 a 40 caracteres adicionales, como +, /, #, %, *, y así sucesivamente. Podemos decir que un código alfanumérico representa a todos los caracteres y funciones que se encuentran en el teclado de una computadora.

Código ASCII

El código alfanumérico más utilizado es el **Código estándar estadounidense para el intercambio de información (ASCII)**. Este código es de siete bits, por lo cual tiene $2^7 = 128$ código posibles. Más que suficiente para representar todos los caracteres estándar del teclado, así como las funciones de control tales como retorno de carro (RETURN) y avance de línea (LINEFEED). La tabla muestra un listado del código ASCII estándar de siete bits. La tabla proporciona los equivalentes en hexadecimal y decimal. Para obtener el código binario de siete bits para cada carácter hay que convertir el valor hexadecimal en binario.

Código ASCII

TABLA 2-4 Código ASCII estándar.

Caracter	HEX	Decimal	Caracter	HEX	Decimal	Caracter	HEX	Decimal	Caracter	HEX	Decimal
NUL (nulo)	0	0	Espacio	20	32	@	40	64	`	60	96
Inicio del encabezado	1	1	!	21	33	A	41	65	a	61	97
Inicio del texto	2	2	"	22	34	B	42	66	b	62	98
Fin del texto	3	3	#	23	35	C	43	67	c	63	99
Fin de la transmisión	4	4	$	24	36	D	44	68	d	64	100
Consulta	5	5	%	25	37	E	45	69	e	65	101
Reconocimiento	6	6	&	26	38	F	46	70	f	66	102
Timbre	7	7	'	27	39	G	47	71	g	67	103
Retroceso	8	8	(	28	40	H	48	72	h	68	104
Tabulación horizontal	9	9	)	29	41	I	49	73	i	69	105
Avance de línea	A	10	*	2A	42	J	4A	74	j	6A	106
Tabulación vertical	B	11	+	2B	43	K	4B	75	k	6B	107
Avance de hoja de impresi	C	12	,	2C	44	L	4C	76	l	6C	108
Retorno de carro	D	13	-	2D	45	M	4D	77	m	6D	109
Mayúsculas desactivadas	E	14	.	2E	46	N	4E	78	n	6E	110
Mayúsculas activadas	F	15	/	2F	47	O	4F	79	o	6F	111
Escape de enlace de datos	10	16	0	30	48	P	50	80	p	70	112
Control directo 1	11	17	1	31	49	Q	51	81	q	71	113
Control recib 2	12	18	2	32	50	R	52	82	r	72	114
Control cabra	13	19	3	33	51	S	53	83	s	73	115
Control cabre	14	20	4	34	52	T	54	84	t	74	116
ACK (reconocimiento) negativo	15	21	5	35	53	U	55	85	u	75	117
Sincronía en estado cilíra	16	22	6	36	54	V	56	86	v	76	118
Fin de Bloque de transmisión	17	23	7	37	55	W	57	87	w	77	119
Cancelar	18	24	8	38	56	X	58	88	x	78	120
Fin co c medi	19	25	9	39	57	Y	59	89	y	79	121
Sustituir	1A	26	:	3A	58	Z	5A	90	z	7A	122
Escape	1B	27	;	3B	59	[	5B	91	{	7B	123
Separador de formas	1C	28	<	3C	60	\	5C	92	|	7C	124
Separador de grupos	1D	29	=	3D	61	]	5D	93	}	7D	125
Separador de registro	1E	30	>	3E	62	^	5E	94	~	7E	126
Separador de unidades	1F	31	?	3F	63	_	5F	95	Suprimir	7F	127

Código EBCDIC

- [Extended Binary-Coded Decimal Interchange Code]. Es un código que usa 8 dígitos binarios para representar un carácter simple, dando un máximo posible de 256 caracteres.
- Es utilizado como un sistema de código en muchos computadores.
- El código EBCDIC es simplemente el código BCD extendido a 8-bits.

Código EBCDIC

HEX				MSD–	0	1	2	3	4	5	6	7
	BITS			b7	0	0	0	0	0	0	0	0
				b6	0	0	0	0	1	1	1	1
				b5	0	0	1	1	0	0	1	1
LSD	b3	b2	b1	b0 b4	0	1	0	1	0	1	0	1
0	0	0	0	0	NUL	DLE	DS		SP	&	-	
1	0	0	0	1	SOH	DC1	SOS					
2	0	0	1	0	STX	DC2	FS	SYN				
3	0	0	1	1	ETX	DC3						
4	0	1	0	0	PF	RES	BYP	PN				
5	0	1	0	1	HT	NL	LF	RS				
6	0	1	1	0	LC	BS	EOB ETB	UC				
7	0	1	1	1	DEL	IL	PRE ESC	EOT				
8	1	0	0	0		CAN						
9	1	0	0	1	RLF	EM						\
A	1	0	1	0	SMM	CC	SM		¢	!	|	:
B	1	0	1	1	VT				.	$	'	#
C	1	1	0	0	FF	IFS		DC4	<	*	%	@
D	1	1	0	1	CR	IGS	ENQ	NAK	(	)	_	´
E	1	1	1	0	SO	IRS	ACK		+	;	>	=
F	1	1	1	1	SI	IUS	BEL	SUB	¦	¬	?	"

NUL Null
SOH Start of Heading
STX Start of Text
ETX End of Text

RLF Reverse Line Feed
SMM Start of Manual Message
VT Vertical Tabulation
FF Form Feed
CR Carriage Return
SO Shift Out
SI Shift In
DLE Data Link Escape
DC1 Device Control 1
DC2 Device Control 2
DC3 Device Control 3
RES Restore
NL New Line
BS Backscape
IL Idle
CAN Cancel
EM End of Medium
CC Cursor Control
IFS Interchange File Separator
IGS Interchange Group Separator
IRS Interchange Record Separator
IUS Interchange Unit Separator

PF Punch Off
HT Horizontal Tab
LC Lower Case
DEL Delete

DS Digit Select Start
SOS of Significance
FS Field Separator
BYP Bypass
LF Line Feed
EOB/ETB End of B lock/End Transmission Block
PRE/ESC Prefix/Escape
SM Set Mode
ENQ Enquiry
ACK Ackmowledge
BEL Bell
SYN Synchronous Idle
PN Pench On
RS Reader Stop
UC Upper Case
EOT End of Transmission
DC4 Device Control 4
NAK Negative Acknowledge
SUB Substitute
SP Space

Código EBCDIC

HEX	BITS			MSD→	8	9	A	B	C	D	E	F
				b7	1	1	1	1	1	1	1	1
				b6	0	0	0	0	1	1	1	1
				b5	0	0	1	1	0	0	1	1
LSD	b3	b2	b1	b0 b4	0	1	0	1	0	1	0	1
0	0	0	0	0					{	}	\	0
1	0	0	0	1	a	j	~		A	J		1
2	0	0	1	0	b	k	s		B	K	S	2
3	0	0	1	1	c	l	t		C	L	T	3
4	0	1	0	0	d	m	u		D	M	U	4
5	0	1	0	1	e	n	v		E	N	V	5
6	0	1	1	0	f	o	w		F	O	W	6
7	0	1	1	1	g	p	x		G	P	X	7
8	1	0	0	0	h	q	y		H	Q	Y	8
9	1	0	0	1	i	r	z		I	R	Z	9
A	1	0	1	0								
B	1	0	1	1								
C	1	1	0	0								
D	1	1	0	1								
E	1	1	1	0								
F	1	1	1	1								

Algebra de Boole

- El Álgebra de Boole utiliza variables que tienen solo dos valores posibles, esto lo sintetizó Shannon usando ideas que inicialmente las expresó el matemático inglés: George Boole . A diferencia de las variables del álgebra común [que pueden tomar un número infinito de valores en un rango determinado], una variable booleana, por ejemplo A, puede tomar solamente 2 valores, que generalmente se los relaciona con VERDADERO y FALSO. Sin embargo, se les puede asignar otros valores, tal como: caliente/frío, macho/hembra, alto/bajo, etc.
- Para representar los 2 posibles valores de las variable booleanas se utilizan los símbolos 0 y 1. Generalmente A = 1 significa que A es VERDADERO en un sentido booleano, mientras que A = 0 indica que A es FALSO.

Algebra de Boole

- Entonces una variable booleana puede estar relacionada a algún ítem de información, por ejemplo, A = 1, significa que un interruptor asociado con A está abierto y A = 0 significa que el mismo interruptor está cerrado.
- Otra variable, B, puede relacionarse a la temperatura de una habitación, siendo VERDADERA cuando la temperatura exceda los 21°C y FALSA en otro caso o viceversa.
- Las variables booleanas no toman valores cuantitativos, pero pueden usarse para representar información cuantitativa. Por ejemplo, se pueden usar 4-variables booleanas para representar un número binario de 4-dígitos.

Algebra de Boole

- Cada variable puede estar relacionada a uno de los coeficientes del número binario, indicando que el coeficiente tiene un valor de 1 cuando la variable es VERDADERA y un valor 0 cuando es FALSA [o el inverso de esto].
- De esta manera las 16 posibles combinaciones pueden estar relacionadas a las cantidades 0-15 10 , que puede tomar el número binario. Conociendo los valores VERDADERO/FALSO de cada una de las variables, posibilitará el cálculo de la cantidad que ella representa.
- Para trabajar con variables booleanas, se utilizan operadores similares a los del álgebra común.
- A estos operadores booleanos comúnmente se los conoce como conectivos lógicos.

Proposiciones y Conectivos Lógicos

Proposición	Planteamiento de un teorema o de un problema que se debe demostrar o resolver.
Premisa	Supuesto material, no necesariamente válido lógicamente, a partir del que se infiere una conclusión.
Conectivo	Son los operadores [o compuertas] del álgebra de Boole, similares a los del álgebra común, y representan a los circuitos digitales más fundamentales. En este capítulo se describe su operación mediante el uso del álgebra de Boole. Se estudia cómo pueden combinarse entre sí varias compuertas para implementar circuitos lógicos más complejos.
Variable Booleana	Las variables booleanas sólo pueden tomar dos valores lógicos: "0" o "1". En un circuito lógico, una variable booleana puede representar ausencia o presencia de voltaje. En una proposición lógica, la variable booleana puede ser falsa o verdadera. En general sólo tienen dos opciones posibles.

Constantes y Variables Booleanas

El álgebra booleana difiere en gran medida del álgebra ordinaria, ya que a las constantes y variables booleanas sólo se les permite tener dos valores posibles: 0 y 1. Una variable booleana es una cantidad que puede ser (en distintas ocasiones) igual a 0 o a 1. Las variables booleanas se utilizan a menudo para representar el nivel de voltaje presente en un alambre o en las terminales de entrada/salida de un circuito. Por ejemplo, en cierto sistema digital el valor booleano 0 podría asignarse a cualquier voltaje en el intervalo de 0 a 0.8 V, mientras que el valor booleano 1 podría asignarse a cualquier voltaje entre 2 y 5 V.*

Por lo tanto, el 0 y el 1 booleanos no representan números reales, sino el estado de una variable de voltaje, o lo que se conoce como su **nivel lógico**. Se dice que un voltaje en un circuito digital está en el nivel 0 lógico o en el nivel 1 lógico, dependiendo de su valor numérico actual. En la lógica digital se utilizan otros términos más como sinónimos de 0 y 1. La tabla 3-1 muestra algunos de los más comunes. La mayor parte del tiempo utilizaremos las designaciones 0/1 y BAJO/ALTO.

0 lógico	1 lógico
Falso	Verdadero
Apagado	Encendido
Bajo	Alto
No	Sí
Interruptor abierto	Interruptor cerrado

Operaciones Lógicas

Como dijimos en la introducción, el **álgebra booleana** es el medio para expresar la relación entre las entradas y las salidas de un circuito lógico. Las entradas se consideran variables lógicas cuyos niveles lógicos en cualquier momento determinan los niveles de salida. En todo el trabajo que veremos utilizaremos símbolos de letras para representar variables lógicas. Por ejemplo, la letra *A* podría representar una cierta entrada o salida de un circuito digital, y en un determinado momento debemos tener $A = 0$ o $A = 1$; alguno de los dos estados.

Como sólo dos valores son posibles, en realidad es muy sencillo trabajar con el álgebra booleana en comparación con el álgebra ordinaria. En el álgebra booleana no hay fracciones, decimales, números negativos, raíces cuadradas, raíces cúbicas, logaritmos, números imaginarios, etc. De hecho, en el álgebra booleana sólo hay *tres* operaciones básicas: *OR*, *AND* y *NOT*.

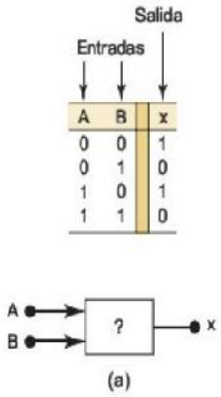
A estas operaciones básicas se les conoce como *operaciones lógicas*. Los circuitos digitales, llamados *compuertas lógicas*, pueden construirse a partir de diodos, transistores y resistencias conectados de manera que la salida del circuito sea el resultado de una operación lógica básica (*OR*, *AND*, *NOT*) que se lleva a cabo con las entradas. Utilizaremos primero el álgebra booleana para describir y analizar las compuertas lógicas básicas, y después para analizar y diseñar combinaciones de compuertas lógicas conectadas para formar circuitos lógicos.

Tablas de Verdad

- Una tabla de verdad es una herramienta para describir la forma en que la salida de un circuito lógico depende de los niveles lógicos presentes en las entradas del circuito.
- La figura muestra una tabla de verdad para un tipo de circuito lógico de dos entradas.
- La tabla lista todas las posibles combinaciones de niveles lógicos presentes en las entradas A y B, junto con el correspondiente nivel en la salida x.
- La primera entrada en la tabla muestra que cuando A y B se encuentran en el nivel 0, la salida x se encuentra en el nivel 1 o, de manera equivalente, en el estado 1. La segunda entrada muestra que cuando la entrada B se cambia al estado **1, de manera** que A= 0 y B = 1,la salida xse vuelve un 0. De manera similar la tabla muestra qué ocurre con el estado de salida para cualquier conjunto de condiciones de entrada.

Tablas de Verdad

FIGURA . Ejemplo de tablas de verdad para circuitos de (a) dos entradas, (b) tres entradas y (c) cuatro entradas.



A	B	C	x
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(b)

A	B	C	D	x
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

(c)

Tablas de Verdad

- La figura anterior muestra ejemplos de tablas de verdad para circuitos lógicos de tres y cuatro entradas.
- De nuevo, cada tabla lista todas las posibles combinaciones de niveles lógicos de las entradas a la izquierda, con el nivel lógico resultante para la salida x a la derecha. Desde luego que los valores reales para x dependerán del tipo de circuito lógico.
- Observe que hay 4 combinaciones para la tabla de verdad de dos entradas, 8 combinaciones para una tabla de verdad de tres entradas y 16 combinaciones para la tabla de verdad de cuatro entradas.
- El número de combinaciones de entrada será igual a 2^N para una tabla de verdad con N entradas. Observe también que la lista de todas las posibles combinaciones de entrada va de acuerdo con la secuencia de conteo binario, por lo que es fácil anotar todas las combinaciones sin que falte una.