Diseño Digital Moderno

Sistemas númericos y códigos

M.I. Bryan Emmanuel Alvarez Serna

Facultad de Ingeniería
Universidad Nacional Autónoma de México

“Siempre parece imposible hasta que se hace”

Nelson Mandela

Bases número

Son un sistema de número que usan conjuntos específicos de dígitos y reglas para representar valores numéricos.

Base decimal

- Base: 10.
- Dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9.

Base binaria

- Base: 2.
- Dígitos: 0 y 1.

Bases número

Base octal

- Base: 8.
- Dígitos: 0, 1, 2, 3, 4, 5, 6 y 7.

Base hexadecimal

- Base: 16.
- Dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F.

Nota: A, B, C, D, E y F corresponden al 10, 11, 12, 13, 14 y 15, respectivamente.

Bases número

Ejemplos

- Decimal

$$312 = 3 \times 10^2 + 1 \times 10^1 + 2 \times 10^0 = 300 + 10 + 2 = 312$$

- Binario

$$101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 4 + 0 + 1 = 5$$

- Octal

$$567_8 = 5 \times 8^2 + 6 \times 8^1 + 7 \times 8^0 = 320 + 48 + 7 = 375$$

- Hexadecimal

$$93_{16} = 9 \times 16^1 + 3 \times 16^0 = 144 + 3 = 147$$

Bases numéricas

DEC	BIN	OCT	HEX
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Conversiones

Decimal a binario

Para representar un número decimal x se debe dividir entre 2 y tomar el residuo como el valor binario.

Ejemplo

Convertir 93 a binario.

93	÷ 2	
46	1	LSB
23	0	
11	1	
5	1	
2	1	
1	0	
	1	MSB

Tomando el bit MSB primero, queda lo siguiente

$$93_{10} = 101\ 1101$$

Conversiones

Ejemplos

Convertir 178 a binario.

178	÷ 2	
89	0	LSB
44	1	
22	0	
11	0	
5	1	
2	1	
1	0	
	1	MSB

$178_{10} = 101\ 1101$

Convertir 23 a binario.

23	÷ 2	
11	1	LSB
5	1	
2	1	
1	0	
	1	MSB

$23_{10} = 1\ 0111$

Conversiones

Binario a decimal

Tomar el número binario, multiplicar por base 2 y sumar los resultados. Hay que considerar al bit de la derecha como el LSB.

Ejemplos

Convertir 101011 a decimal.

$$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 32 + 8 + 2 + 1 = 43$$

Convertir 11111 a decimal.

$$1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 16 + 8 + 4 + 2 + 1 = 31$$

Conversiones

Decimal a octal

Para representar un número decimal x se debe dividir entre 8 y tomar el residuo como el valor octal. Es un procedimiento similar a la conversión a binario, pero con diferente base.

Ejemplo

Convertir 46 a octal.

46	÷ 8	
5	6	LSB
	5	MSB

Tomando el bit MSB primero, obtenemos

$$46_{10} = 56_8$$

Conversiones

Ejemplos

Convertir 200 a octal.

200	÷ 8	
25	0	LSB
3	1	
	3	MSB

$200_{10} = 310_8$

Convertir 1234 a octal.

1234	÷ 8	
154	2	LSB
19	2	
2	3	
	2	MSB

$1234_{10} = 2322_8$

Conversiones

Octal a decimal

Se usa el mismo procedimiento que en la conversión binaria, pero se cambia la base por 8.

Ejemplos

Convertir 76_8 a decimal.

$$7 \times 8^1 + 6 \times 8^0 = 56 + 6 = 62$$

Convertir 3160_8 a decimal.

$$3 \times 8^3 + 1 \times 8^2 + 6 \times 8^1 + 0 \times 8^0 = 1536 + 64 + 48 + 0 = 1648$$

Conversiones

Decimal a hexadecimal

Se usa el mismo procedimiento que en la conversión binaria y octal, pero con base 16. Es importante recordar que los valores 10, 11, 12, 13, 14 y 15 decimal corresponden a los valores A, B, C, D, E y F hexadecimal.

Ejemplo

Convertir 845 a hexadecimal.

845	÷ 16	
52	(13) D	LSB
3	4	
	3	MSB

Tomando el bit MSB primero, obtenemos

$$845_{10} = 34D_{16}$$

Conversiones

Ejemplos

Convertir 789 a hexadecimal.

789	÷ 16	
49	5	LSB
3	1	
	3	MSB

$789_{10} = 315_{16}$

Convertir 3570 a hexadecimal.

3570	÷ 16	
223	2	LSB
13	(15) F	
	(13) D	MSB

$3570_{10} = DF2_{16}$

Conversiones

Hexadecimal a decimal

Es el mismo procedimientos que converisón a binario y octal, pero usando base 16.

Ejemplos

Convertir 5043_{16} a decimal.

$$5 \times 16^3 + 0 \times 16^2 + 4 \times 16^1 + 3 \times 16^0 = 20480 + 0 + 64 + 3 = 20547$$

Convertir $7AF_{16}$ a decimal.

$$7 \times 16^2 + A \times 16^1 + F \times 16^0 = 1792 + 160 + 15 = 1967$$

Conversiones

Binario a octal

Agrupar de izquierda a derecha el número binario en 3 y poner su equivalente en octal.

Ejemplos

Convertir 101011 a octal.

$$101\ 011 = 5\ 3 = 53_8$$

Convertir 10111010 a octal.

$$10\ 111\ 010 = 010\ 111\ 010 = 2\ 7\ 2 = 272_8$$

Conversiones

Binario a hexadecimal

Hacer grupos de 4 bits y poner su valor equivalente en hexadecimal. EL agrupamiento se hace de izquierda a derecha.

Ejemplos

Convertir 10101111 a hexadecimal.

$$1010\ 1111 = A\ F = AF_{16}$$

Convertir 11101101101 a hexadecimal.

$$111\ 0110\ 1101 = 0111\ 0110\ 1101 = 7\ 6\ D = 76D_{16}$$

Convertir 11011 a hexadecimal.

$$1\ 1011 = 0001\ 1011 = 1\ B = 1B_{16}$$

Conversiones

¿Para qué me sirve? 🤔

BIN	OCT	HEX	DEC
0101	5	5	5
1110 1010 0011	7243	EA3	3747
1110 1010 1101 0011	165 323	EAD3	60115

En resumen

Las conversiones de bases sirven para simplificar el manejo de datos y optimizar el diseño de un sistema digital. Sin embargo, todas las bases son útiles.

Suma

El manejo de operaciones binarias básicas como suma, resta, multiplicación y división, es fundamental para el análisis y diseño de circuitos digitales.

Suma

Se realiza bit a bit según la siguiente tabla

	Res	Carry
$0+0$	0	0
$0+1$	1	0
$1+0$	1	0
$1+1$	0	1

El carry o acarrero también se conoce como carry out.

Ejemplos

$$\begin{array}{cccc} & 1 & & \\ & & 1 & 0 \\ + & & 1 & 1 \\ \hline & 1 & 0 & 1 \end{array} \qquad \begin{array}{ccccc} & 1 & & & 1 \\ & & 1 & 1 & 0 & 1 \\ + & & 1 & 0 & 0 & 1 \\ \hline & 1 & 0 & 1 & 1 & 0 \end{array}$$

Resta

Resta

La resta es una operación complicada en los sistemas digitales, ya que la mayoría de estos están formados únicamente por sumadores. Además, la resta necesita "pedir prestado" o *borrow* en ciertas condiciones.

Ejemplo

Para comprender mejor el problema de *borrow*, intentemos resolver la siguiente resta.

$$\begin{array}{rrrr} & 1 & 0 & 1 \\ - & 1 & 1 & 1 \\ \hline \end{array} \quad \text{que es equivalente a} \quad \begin{array}{rr} & 5 \\ - & 7 \\ \hline - & 2 \end{array}$$

Resta

Ejemplo

$$\begin{array}{rrrr} & 1 & 0 & 1 \\ - & 1 & 1 & 1 \\ \hline & & & 0 \end{array} \quad \rightarrow \quad \begin{array}{rrrr} & 0 & 1 & 1 \\ - & 1 & 1 & 1 \\ \hline & ? & 0 & 0 \end{array}$$

En conlclusión

Por lo tanto, por este método es imposible resolver esta resta. El resultado en decimal es -2, sin embargo, en binario el resultado se expresa como 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111 1111.

Resta

Para solucionar este problema de la resta, se puede usar la suma con números negativos; es decir

Ejemplo

$$\begin{array}{r} 9 \\ -\ 3 \\ \hline 6 \end{array} \qquad \text{es igual a} \qquad \begin{array}{r} 9 \\ +\ -3 \\ \hline 6 \end{array}$$

Pero... ¿Es posible tener un número binario negativo? 😨

Resta

Para solucionar este problema de la resta, se puede usar la suma con números negativos; es decir

Ejemplo

$$\begin{array}{r} 9 \\ -\ 3 \\ \hline 6 \end{array} \quad \text{es igual a} \quad \begin{array}{r} 9 \\ +\ -3 \\ \hline 6 \end{array}$$

Pero... ¿Es posible tener un número binario negativo? 😨

Si, si se puede 😈

Resta

Para convertir números binarios positivos a negativos se tienen aplicar complementos, por ello, es necesario entender los siguientes complementos.

Complemento a 1

Consiste en negar o intercambiar los valores; es decir, los ceros cambiarlos por unos y viceversa.

Ejemplos

Aplicar complemento a 1 al número 1011 0101.

$$1011\ 0101 = 0100\ 1010$$

Negar el número 1100 1111 0001.

$$1100\ 1111\ 0001 = 0011\ 0000\ 1110$$

Resta

Complemento a 2

Para aplicar este complemento se deben seguir los siguientes pasos.

1. Aplicarle complemento a 1 al número.
2. Sumar 1 al resultado.

Ejemplo

Aplicar complemento a 2 al número 101101.

1. Aplicar complemento a 1.

$$101101 = 010010$$

2. Sumar 1 al valor 010010.

$$\begin{array}{cccccccc} & 0 & 1 & 0 & 0 & 1 & 0 \\ + & & & & & & 1 \\ \hline & 0 & 1 & 0 & 0 & 1 & 1 \end{array}$$

Resta

Ejemplo

Aplicar complemento a 2 al número 1010 0100.

1. 1010 0100 = 0101 1011.
2.

$$
\begin{array}{cccccccccc}
 & & & & & & 1 & 1 & \\
 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
+ & & & & & & & & 1 \\
\hline
 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0
\end{array}
$$

Un método más rápido.

1. Ubicar el primer 1 de izquierda a derecha.

   1 0 1 0 0 **1** 0 0

   ↑

2. Reescribir los valores hasta el primer 1 y después aplicar complemento 1 al resto.

   0 1 0 1 1 1 0 0

Resta

Para convetir un número en negativo se debe usar un bit de signo, que es el MSB. Este bit se debe marcar para poder identificarlo.
La finalidad de tener un número negativo, es que se puede aplicar una suma para resolver el porblema de *borrow* que tiene la resta.

Ejemplo

Si tenemos la resta 24 - 13 la podemos plantear como la suma de 24 + (-13). Para esto necesitamos hacer lo siguiente.

1. Plantear la suma y tener el mismo número de dígitos en el minuendo y el sustraendo.

$$\begin{array}{rrrrrr} & 1 & 1 & 0 & 0 & 0 \\ - & 0 & 1 & 1 & 0 & 1 \\ \hline \end{array} \qquad \begin{array}{rr} & 24 \\ - & 13 \\ \hline \end{array}$$

Resta

2 Agregar el bit de signo de los números, considerar que 1 es para negativos y 0 para positivos.

$$\begin{array}{rcccccc} & 0 & 1 & 1 & 0 & 0 & 0 \\ - & 0 & 0 & 1 & 1 & 0 & 1 \\ \hline \end{array}$$

3 Aplicar complemento a 2 al sustraendo y aplicar la suma.

$$\begin{array}{rcccccc} & \cancel{1} & 1 & & & & \\ & 0 & 1 & 1 & 0 & 0 & 0 \\ + & 1 & 1 & 0 & 0 & 1 & 1 \\ \hline & 0 & 0 & 1 & 0 & 1 & 1 \end{array}$$

El resultado es 1011 u 11 en decimal.

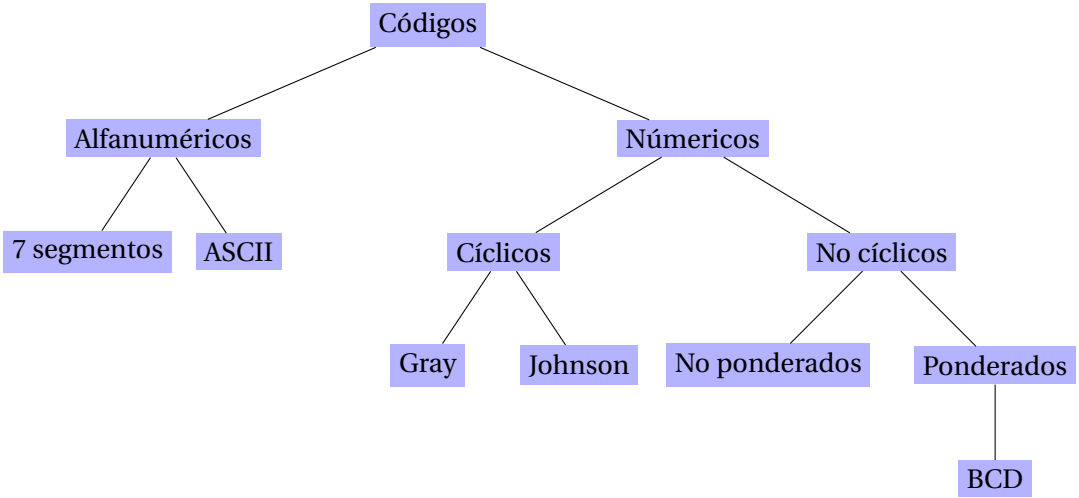
Nota

Si al final de la suma aparece un carry, no se debe tomar en cuenta.

Multiplicación y división

- La multiplicación y la división en bases numéricas no decimales tienen principios similares a los de la aritmética decimal, pero con diferentes conjuntos de dígitos y reglas de conversión.
- La base depende de la aplicación y puede simplificar el diseño de circuitos digitales.

Códigos


Códigos
Alfanuméricos
Númericos
7 segmentos
ASCII
Cíclicos
No cíclicos
Gray
Johnson
No ponderados
Ponderados
BCD

Código Gray

DECIMAL	BCD	GRAY 3 BITS	GRAY 2 BITS
0	000	000	00
1	001	001	01
2	010	011	11
3	011	010	10
4	100	110	
5	101	111	
6	110	101	
7	111	100	

BCD: Binary-Coded Decimal / Decimal codificado en binario

Código Gray

Conversión de Gray a BCD.

1. Se mantiene el MSB.
2. Se suma el bit adyacente y se desprecia el carry.

Ejemplo

Covertir el número 101_{Gray} a BCD.

1. Se mantiene el 1 del MSB.

$$\begin{array}{ccc} 1 & 0 & 1 \\ 1 & & \end{array}$$

2. Se suman el resultado con el bit adyancente (de forma cruzada).

$$\begin{array}{ccc} 1 & 0 & 1 \\ 1 & 1+0 & \end{array} \qquad \begin{array}{ccc} 1 & 0 & 1 \\ 1 & 1 & 1+1 \end{array} \qquad \begin{array}{ccc} 1 & 0 & 1 \\ 1 & 1 & 0 \end{array} \qquad 101_{Gray} = 110$$

Código Gray

Conversión de BCD a Gray.

1. Se mantiene el MSB.
2. Se suma el bit adyacente de forma directa y se desprecia el carry.

Ejemplo

Convertir el número 111 a Gray.

1. Mantener el MSB.

$$1 \quad 1 \quad 1 \qquad\qquad 1$$

2. Sumar con el bit adyacente de forma directa e ignorar el carry.

$$1 \quad 1 \quad 1 \qquad\qquad 1 \quad 1+1 \quad 1+1 \qquad\qquad 1 \quad 0 \quad 0$$

$$111 = 100_{Gray}$$

Código ASCII

ASCII (American Standard Code for Information Interchange).

ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*
1	1	1	1	*[START OF HEADING]*
2	2	10	2	*[START OF TEXT]*
3	3	11	3	*[END OF TEXT]*
4	4	100	4	*[END OF TRANSMISSION]*
5	5	101	5	*[ENQUIRY]*
6	6	110	6	*[ACKNOWLEDGE]*
7	7	111	7	*[BELL]*
8	8	1000	10	*[BACKSPACE]*
9	9	1001	11	*[HORIZONTAL TAB]*
10	A	1010	12	*[LINE FEED]*
11	B	1011	13	*[VERTICAL TAB]*
12	C	1100	14	*[FORM FEED]*
13	D	1101	15	*[CARRIAGE RETURN]*
14	E	1110	16	*[SHIFT OUT]*
15	F	1111	17	*[SHIFT IN]*
16	10	10000	20	*[DATA LINK ESCAPE]*
17	11	10001	21	*[DEVICE CONTROL 1]*
18	12	10010	22	*[DEVICE CONTROL 2]*
19	13	10011	23	*[DEVICE CONTROL 3]*
20	14	10100	24	*[DEVICE CONTROL 4]*
21	15	10101	25	*[NEGATIVE ACKNOWLEDGE]*
22	16	10110	26	*[SYNCHRONOUS IDLE]*
23	17	10111	27	*[END OF TRANS. BLOCK]*
24	18	11000	30	*[CANCEL]*
25	19	11001	31	*[END OF MEDIUM]*
26	1A	11010	32	*[SUBSTITUTE]*
27	1B	11011	33	*[ESCAPE]*
28	1C	11100	34	*[FILE SEPARATOR]*
29	1D	11101	35	*[GROUP SEPARATOR]*
30	1E	11110	36	*[RECORD SEPARATOR]*
31	1F	11111	37	*[UNIT SEPARATOR]*
32	20	100000	40	*[SPACE]*
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	**0**
49	31	110001	61	**1**
50	32	110010	62	**2**
51	33	110011	63	**3**
52	34	110100	64	**4**
53	35	110101	65	**5**
54	36	110110	66	**6**
55	37	110111	67	**7**
56	38	111000	70	**8**
57	39	111001	71	**9**
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	**A**
66	42	1000010	102	**B**
67	43	1000011	103	**C**
68	44	1000100	104	**D**
69	45	1000101	105	**E**
70	46	1000110	106	**F**
71	47	1000111	107	**G**
72	48	1001000	110	**H**
73	49	1001001	111	**I**
74	4A	1001010	112	**J**
75	4B	1001011	113	**K**
76	4C	1001100	114	**L**
77	4D	1001101	115	**M**
78	4E	1001110	116	**N**
79	4F	1001111	117	**O**
80	50	1010000	120	**P**
81	51	1010001	121	**Q**
82	52	1010010	122	**R**
83	53	1010011	123	**S**
84	54	1010100	124	**T**
85	55	1010101	125	**U**
86	56	1010110	126	**V**
87	57	1010111	127	**W**
88	58	1011000	130	**X**
89	59	1011001	131	**Y**
90	5A	1011010	132	**Z**
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	**a**
98	62	1100010	142	**b**
99	63	1100011	143	**c**
100	64	1100100	144	**d**
101	65	1100101	145	**e**
102	66	1100110	146	**f**
103	67	1100111	147	**g**
104	68	1101000	150	**h**
105	69	1101001	151	**i**
106	6A	1101010	152	**j**
107	6B	1101011	153	**k**
108	6C	1101100	154	**l**
109	6D	1101101	155	**m**
110	6E	1101110	156	**n**
111	6F	1101111	157	**o**
112	70	1110000	160	**p**
113	71	1110001	161	**q**
114	72	1110010	162	**r**
115	73	1110011	163	**s**
116	74	1110100	164	**t**
117	75	1110101	165	**u**
118	76	1110110	166	**v**
119	77	1110111	167	**w**
120	78	1111000	170	**x**
121	79	1111001	171	**y**
122	7A	1111010	172	**z**
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	*[DEL]*

Código 2 entre 5

También conocido como ITF, siglas en inglés de *Interleaved 2 of 5*. Es un código utilizado para la detección de errores en transmisiones digitales.

Características

1. Debe haber dos bits en 1 entre los cinco existentes.
2. No existe codificación para el 0, como tradicionalmente se conoce; ie, 00000.
3. El peso para los 4 primeros bits son 1, 2, 3 y 6.
4. El MSB se utiliza para completar la paridad.

DEC	2-5
1	11000
2	10100
3	10010
6	10001

Código 2 entre 5

Los números restantes se obtienen mediante sumas.

DEC	Suma	2-5
4	3 + 1	01010
5	3 + 2	00110
7	6 + 1	01001
8	6 + 2	00101
9	6 + 3	00011

Nota

Si existe un acarreo en el último bit de la suma se ignora, para no afectar la paridad.

Representación del cero

El cero se puede representar como 2 + 1, ya que el 3 tiene su propio valor de peso.

0	01100

Tarea 1

Resolver los siguientes ejercicios.

1. Convertir el número decimal 429 a binario, octal y hexadecimal.
2. Suponiendo una base llamada *trinaria* que tiene como base al número 3 y se compone de los dígitos 0, 1 y 2. Convertir el valor *trinario* 1202_3 a decimal y el decimal 853 a *trinario*.
3. Realizar la suma binaria de 327_8 con $1D8_{16}$ y representar el resultado en binario.
4. Utilizar complemento a 2 para realizar la siguiente resta.

$$\begin{array}{cccccccc} & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ - & & 1 & 1 & 1 & 1 & 0 & 0 \\ \hline \end{array}$$

5. Obtener todos los valores posibles del código Gray de 4 bits a partir de las conversiones de los valores decimales del 0 al 9 en BCD.