Electrónica

Diseño lógico

Fundamentos en electrónica digital

Héctor Arturo **Flórez** Fernández

ediciones de la U
conocimiento a su alcance
www.edicionesdelau.com

Capítulo 1
Sistemas Numéricos

Los sistemas numéricos son un concepto fundamental para el estudio de la electrónica digital. Dentro de los sistemas numéricos que se pretenden utilizar se encuentran el sistema decimal, el sistema octal, el sistema binario y el sistema hexadecimal. Cada uno de estos sistemas tiene una base, la cual indica la cantidad de símbolos del sistema.

1.1 SISTEMA DECIMAL

El sistema numérico decimal, es un sistema base 10 debido a que tiene 10 símbolos los cuales son los números del 0 al 9. Para este sistema a cada símbolo se le denomina ***dígito*** y es el más comúnmente usado en la vida cotidiana, ya que por medio de éste podemos representar cualquier cantidad numérica estándar. Cuando se trabaja números que se encuentran en diferentes sistemas, es aconsejable identificar la base del número colocándola como subíndice. Por ejemplo, si se desea representar el número 25 en decimal se coloca: 25_{10}

Al tener 10 dígitos, se pueden formar números de diferentes cantidades. Un número con una sola cifra, tiene 10 posibles cantidades que van de 0 a 9. Si un número tiene dos cifras, entonces se pueden tener hasta 100 posibles cantidades.

Cada cifra dentro de un número tiene un valor que es comúnmente llamado como ***peso.*** La cifra de menor peso, siempre es la cifra ubicada a la derecha del número. Los pesos de cada cifra incrementan de forma exponencial en donde la base es la cantidad de símbolos del sistema y el exponente es la posición de cada símbolo.

Entonces la cantidad de un número se obtiene de la siguiente forma:

$$348 = 8 * 10^0 + 4 * 10^1 + 3 * 10^2$$
$$348 = 8 * 1 + 4 * 10 + 3 * 100$$
$$348 = 8 + 40 + 300$$

La posición de cada dígito en un número decimal indica la magnitud de la cantidad representada. Los pesos para los números enteros con potencias positivas de diez que aumentan de derecha a izquierda comenzando por 10^0. Para los números fraccionarios, los pesos son potencias negativas de diez que aumentan de izquierda a derecha comenzando por 10^{-1}.

... 10^2 10^1 10^0 . 10^{-1} 10^{-2} ...

Por ejemplo, la representación del número decimal 568,25 como suma de valores de cada dígito es:

$$568{,}25 = 5 * 10^2 + 6 * 10^1 + 8 * 10^0 + 2 * 10^{-1} + 5 * 10^{-2}$$
$$568{,}25 = 5 * 100 + 6 * 100 + 8 * 100 + 2 * 0{,}1 + 5 * 0{,}01$$
$$568{,}25 = 500 + 60 + 8 + 0{,}2 + 0{,}05$$

1.2 SISTEMA BINARIO

El sistema numérico binario es un sistema base 2 debido a que tiene 2 símbolos los cuales son los números 0 y 1. En este sistema a un símbolo se le denomina ***bit***. Es aconsejable identificar la base del número colocándola como subíndice. Por ejemplo, si se desea representar el número 100110 en binario se coloca: 100110_2

Para formar cantidades binarias es necesario tener en cuenta que sólo hay 2 símbolos en el sistema. Con base en esto, un número con una sola cifra solo puede tener 2 posibles combinaciones. Un número con 2 cifras puede tener 4 posibles combinaciones. Esto es debido a que los pesos para los números binarios se calculan con potencias positivas de base 2 que aumentan de derecha a izquierda comenzando por 2^0.

Cada número binario tiene su equivalente decimal. Por ejemplo si se tiene números binarios de 2 bits se pueden lograr las siguientes combinaciones:

Tabla 1.1 Código binario de 2 bits

Decimal	Binario
0	00
1	01
2	10
3	11

Con 3 bits se puede lograr las siguientes combinaciones:

Tabla 1.2 Código binario de 3 bits

Decimal	Binario
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Entonces se puede observar que la cantidad de combinaciones que se pueden obtener en binario corresponde a 2^n donde n es el número de bits. Con base en lo anterior, con 4 bits se pueden obtener 16 combinaciones, con 5 bits 32 combinaciones y así sucesivamente.

Igual que el sistema decimal, cada bit tiene un peso. Los pesos para los números binarios son potencias positivas base dos que aumentan de derecha a izquierda comenzando por 2^0.

... 2^2 2^1 2^0

1.3 SISTEMA OCTAL

El sistema numérico octal es un sistema base 8 debido a que tiene 8 símbolos los cuales son los números del 0 al 7. Es necesario identificar la base del número colocándola como subíndice. Por ejemplo, si se desea representar el número 256 en octal se coloca: 256_8. La cantidad de combinaciones que se pueden obtener en octal corresponde a 8^n donde n es el número de símbolos.

Igual que el sistema decimal, cada símbolo tiene un peso. Los pesos para los números octales son potencias positivas base ocho que aumentan de derecha a izquierda comenzando por 8^0.

... 8^2 8^1 8^0

1.4 SISTEMA HEXADECIMAL

El sistema numérico hexadecimal es un sistema base 16 debido a que tiene 16 símbolos los cuales son los números del 0 al 9 y de la letra A a la F. Es nece-

sario identificar la base del número colocándola como subíndice. Por ejemplo, si se desea representar el número 95B en binario se coloca: $95B_{16}$. La cantidad de combinaciones que se pueden obtener en hexadecimal corresponde a 16^n donde n es el número de símbolos.

Igual que el sistema decimal, cada símbolo tiene un peso. Los pesos para los números hexadecimales son potencias positivas base dieciséis que aumentan de derecha a izquierda comenzando por 16^0.

... 16^2 16^1 16^0

Cada símbolo en hexadecimal tiene una equivalencia en los demás sistemas que se presentan en la siguiente tabla:

Tabla 1.3 Equivalencias entre sistemas

Decimal	Binario	Octal	Hexadecimal
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

1.5 CÓDIGO BCD

Vale la pena mencionar el concepto de código BCD en este capítulo, debido a que este código tiene fuerte relación con el sistema binario y el sistema decimal.

BCD es la sigla de *Binary Coded Decimal*. Significa Decimal codificado binario. Este código tiene una única función que es representar únicamente los símbolos decimales en binario. Con base en lo anterior, el código BCD no es equivalente a código base 2. El código BCD es el de la siguiente tabla:

Tabla 1.4 Código BCD

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Entonces si se desea representar un valor decimal de más de un dígito en BCD, se toma el código BCD de cada dígito y se colocan en el peso correspondiente.

Por ejemplo, si se desea representar el número 45_{10} en BCD quedaría 0100 0101. El 0100_2 equivale a 4_{10} y el 0101_2 equivale a 5_{10}.

1.6 CÓDIGO GRAY

El código Gray es un código con una característica fundamental. La variación entre una combinación y otra es de solo un bit. Esta característica es bastante importante en muchas aplicaciones como detección y corrección de errores en sistemas de comunicaciones digitales, de allí la importancia del mismo.

Para construir el código se debe iniciar con los símbolos del código binario en el bit de menor peso, es decir con el 0 y 1. Después se refleja el código resultante, es decir, que se continúa el código colocando el 1 y 0, luego se coloca el bit de siguiente peso con la mitad de posiciones en 0 y la otra mitad en 1.

00
01
11
10

Después de esto se vuelve a reflejar el código obteniendo:

000
001
011
010
110
111
101
100

Finalmente se obtiene el código Gray (tabla 1.5) de 4 bits.

Tabla 1.5 Código Gray

Binario	Gray
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

1.7 CONVERSIÓN ENTRE SISTEMAS

Es posible convertir un número que se encuentra en un sistema a cualquier otro sistema. Es decir, si tenemos un número decimal, podemos representarlo en binario, octal y hexadecimal.

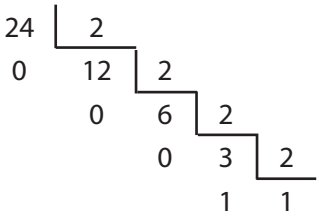
1.7.1 Conversión Decimal-Binario

Esta conversión consiste en obtener un número binario equivalente de un número decimal. Hay dos métodos comúnmente conocidos.

Método de divisiones sucesivas

Este método consiste en tomar el número decimal y dividirlo en 2 (debido a que la base del sistema binario es 2). La división se repite hasta que el cociente sea menor que el divisor. Para este caso en particular, debido a que el divisor es 2 entonces se hace divisiones hasta que el cociente sea 1. Al finalizar el número binario se construye tomando el último cociente y se toma los residuos de las divisiones desde el último hasta el primer residuo.

Por ejemplo si tomamos el número decimal 24_{10} le aplicamos el método quedaría de la siguiente forma:



Finalmente el valor binario sería 11000_2.

Método de suma de pesos

Este método consiste en calcular el número binario equivalente al número decimal dado mediante la suma de los pesos binarios que dan como resultado el número decimal. Es necesario tener en cuenta que los pesos binarios van de 2^0 hasta 2^n donde n es el número de bits. Esto es equivalente a decir que los pesos son: 1, 2, 4, 8, 16, 32, etc.

Entonces, una forma práctica de aplicar el algoritmo es encontrar el peso mayor en binario para el número dado.

Por ejemplo, si tenemos el número 25_{10} entonces debemos encontrar un peso que no exceda esa cantidad. Si se escoge el peso 5, $2^5 = 32$ y excede el 25. Entonces el peso adecuado a escoger es el 4.

$$2^4 = 16$$
$$25 - 16 = 9$$

Aplicamos el mismo algoritmo para el valor resultante que es el 8.

$$2^3 = 8$$
$$9 - 8 = 1$$

Aplicamos el mismo algoritmo para el valor resultante que es el 1. Nos damos cuenta que $2^2 = 4$ excede el 1, $2^1 = 2$ excede el 1, entonces:

$$2^0 = 1$$
$$1 - 1 = 0$$

Finalmente nos damos cuenta que debe intervenir el peso 4, 3 y 0. Eso significa que para estos pesos el valor es 1. Para los pesos 2 y 1 el valor debe ser 0.

2^4	2^3	2^2	2^1	2^0
1	1	1	1	1

$$16 + 8 + 0 + 0 + 1 = 25$$

Entonces el número binario equivalente al número 25 decimal es 11001_2.

1.7.2 Conversión Binario-Decimal

Esta conversión consiste en obtener un número decimal equivalente de un número binario.

Método de suma de pesos

Este método consiste en calcular el valor de cada peso del número binario y hacer la sumatoria de sus resultados. El resultado de la sumatoria equivaldrá al número decimal.

Por ejemplo si tenemos el número 11011010_2 hacemos las siguientes operaciones:

1	1	0	1	1	0	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
$128 * 1 = 128$	$64 * 1 = 64$	$32 * 0 = 0$	$16 * 1 = 16$	$8 * 1 = 8$	$4 * 0 = 0$	$2 * 1 = 2$	$1 * 0 = 0$

$$128 + 64 + 16 + 8 + 2 = 218$$

1.7.3 Conversión Decimal-Octal

Esta conversión consiste en obtener un número octal equivalente de un número decimal. Hay dos métodos comúnmente conocidos.

Método de divisiones sucesivas

Este método consiste en tomar el número decimal y dividirlo en 8 (debido a que la base del sistema octal es 8). La división se repite hasta que el cociente sea menor que el divisor. Al finalizar el número octal se construye tomando el último cociente y se toma los residuos de las divisiones desde el último hasta el primer residuo.

Por ejemplo si tomamos el número decimal 95_{10} le aplicamos en método quedaría de la siguiente forma:

95	8	
7	11	8
	3	1

Finalmente el valor octal sería 137_8.

Método de suma de pesos

Este método consiste en calcular el número octal equivalente al número decimal dado mediante la suma de los pesos octales que dan como resultado el número decimal. Es necesario tener en cuenta que los pesos octales van de 8^0 hasta 8^n donde n es el número de cifras del número octal. Esto es equivalente a decir que los pesos son: 1,8,64,512, etc. El peso de cada símbolo se debe multiplicar por el símbolo correspondiente. Ese resultado es el que se sumará para obtener el número octal.

Entonces, una forma práctica de aplicar el algoritmo es encontrar el peso mayor en octal para el número dado.

Por ejemplo si tenemos el número 95_{10} entonces debemos encontrar un peso que multiplicado por un símbolo octal, no exceda esa cantidad. Si se escoge el peso 3, 8^3=512 y excede el 95, entonces el peso adecuado a escoger es el 2.

$$8^2 = 64$$
$$64 * 1 = 64$$
$$95 - 64 = 9 = 3$$

Aplicamos el mismo algoritmo para el valor resultante que es el 31.

$$8^1 = 8$$
$$8 * 3 = 24$$
$$31 - 24 = 7$$

Aplicamos el mismo algoritmo para el valor resultante que es el 7.

$8^0 = 1$
$1 * 7 = 7$
$7 - 7 = 0$

Entonces:

$8^2 \quad 8^1 \quad 8^0$
$64 \quad 8 \quad 1$
$64 * 1 + 8 * 3 + 1 * 7 = 95$

Entonces el número octal equivalente al número 95 decimal es 137_8.

1.7.4 Conversión Octal-Decimal

Esta conversión consiste en obtener un número decimal equivalente de un número octal.

Método de suma de pesos

Este método consiste en calcular el valor de cada peso del número octal, multiplicarlo por la cifra octal correspondiente y hacer la sumatoria de sus resultados. El resultado de la sumatoria equivaldrá al número decimal.

Por ejemplo si tenemos el número 7315_8 hacemos las siguientes operaciones:

7	3	1	5
8^3	8^2	8^1	8^0
$512 * 7 = 3584$	$64 * 3 = 192$	$8 * 1 = 8$	$1 * 5 = 5$

$3584 + 192 + 8 + 5 = 3789$

1.7.5 Conversión Decimal-Hexadecimal

Esta conversión consiste en obtener un número hexadecimal equivalente de un número decimal.

Método de divisiones sucesivas

Este método consiste en tomar el número decimal y dividirlo en 16 (debido a que la base del sistema hexadecimal es 16). La división se repite hasta que el cociente sea menor que el divisor. Al finalizar el número hexadecimal se construye tomando el último cociente y se toma los residuos de las divisiones desde el último hasta el primer residuo. Se debe tener en cuenta que el último cociente o residuos pueden ser números decimales de 10 a 15. Para estos casos, se hace la equivalencia mostrada en la tabla 1.3.

Por ejemplo si tomamos el número decimal 200_{10} le aplicamos el método quedaría de la siguiente forma:

200	16
8	12

Siguiendo el algoritmo, se toma el último cociente y se toma los residuos desde el último hasta el primero. Entonces el valor hexadecimal sería $C8_{16}$.

Método de suma de pesos

Este método consiste en calcular el número hexadecimal equivalente al número decimal dado mediante la suma de los pesos hexadecimales que dan como resultado el número decimal. Es necesario tener en cuenta que los pesos hexadecimales van de 16^0 hasta 16^n donde n es el número de cifras del número. Esto es equivalente a decir que los pesos son: 1,16,256, etc. El peso de cada símbolo se debe multiplicar por el símbolo del mismo peso. Ese resultado es el que se sumará para obtener el número hexadecimal.

Entonces, una forma práctica de aplicar el algoritmo es encontrar el peso mayor en hexadecimal para el número dado.

Por ejemplo si tenemos el número 200_{10} entonces debemos encontrar un peso que multiplicado por un símbolo octal, no exceda esa cantidad. Si se escoge el peso 2, 16^2=256 y excede el 200. Entonces el peso adecuado a escoger es el 1.

$$16^1 = 16$$
$$16 * 12 = 192$$
$$200 - 192 = 8$$

Aplicamos el mismo algoritmo para el valor resultante que es el 8.

$$16^0 = 1$$
$$1 * 8 = 8$$
$$8 - 8 = 0$$

Entonces:

$$16^1 \quad 16^0$$
$$12 \quad 8$$
$$16 * 12 + 1 * 8 = 200$$

Teniendo en cuenta que los valores resultantes se encuentran entre 10 y 15 se realiza la equivalencia, entonces el resultado es $C8_{16}$.

1.7.6 Conversión Hexadecimal-Decimal

Esta conversión consiste en obtener un número decimal equivalente de un número hexadecimal.

Método de suma de pesos

Este método consiste en calcular el valor de cada peso del número hexadecimal multiplicarlo por la cifra octal correspondiente y hacer la sumatoria de sus resultados. El resultado de la sumatoria equivaldrá al número decimal.

Por ejemplo, si tenemos el número $45B6_{16}$ hacemos las siguientes operaciones

4	5	B	6
16^3	16^2	16^1	16^0
$4096 * 4 = 16384$	$256 * 5 = 1280$	$16 * 11 = 176$	$1 * 6 = 6$

$$16384 + 1280 + 176 + 6 = 17846$$

1.7.7 Conversión Binario-Octal

Esta conversión consiste en obtener un número octal equivalente de un número binario.

El método indica que se debe hacer grupos de tres bits en el número binario, de derecha a izquierda. Cada grupo de tres bits representa un símbolo octal, debido a que el sistema octal tiene 8 símbolos y con 3 bits se pueden tener 8 combinaciones. Si el último grupo de la izquierda no tiene 3 bits, se rellena con ceros a la izquierda. Se puede entonces utilizar la siguiente tabla de equivalencia.

Tabla 1.6. Equivalencias binario - octal

Binario	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Con base en lo anterior, si se tiene el número binario 1001010111_2 el equivalente octal sería:

001	001	010	111
1	1	2	7

Entonces el número octal es 1127_8

1.7.8 Conversión Octal-Binario

Esta conversión consiste en obtener un número binario equivalente de un número octal.

Cada símbolo octal equivale a un valor binario de 3 bits, debido a que el sistema octal tiene 8 símbolos y con 3 bits se pueden tener 8 combinaciones. Se puede utilizar la tabla 1.6 como equivalencias.

Con base en lo anterior, si se tiene el número octal 7516_8 el equivalente binario sería:

7	5	1	6
111	101	001	110

Entonces el número binario es 111101001110_2

1.7.9 Conversión Binario-Hexadecimal

Esta conversión consiste en obtener un número hexadecimal equivalente de un número binario.

El método indica que se debe hacer grupos de cuatro bits en el número binario, de derecha a izquierda. Cada grupo de cuatro bits representa un símbolo hexadecimal, debido a que el sistema hexadecimal tiene 16 símbolos y con 4 bits se pueden tener 16 combinaciones. Si el último grupo de la izquierda no tiene 4 bits, se rellena con ceros a la izquierda. Se puede entonces utilizar la siguiente tabla de equivalencia.

Tabla 1.7 Equivalencias binario - hexadecimal

Binario	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4

0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Con base en lo anterior, si se tiene el número binario 1101110010111_2 el equivalente hexadecimal sería:

0001	1011	1001	0111
1	*B*	9	7

Entonces el número hexadecimal es $1B97_{16}$

1.7.10 Conversión Hexadecimal-Binario

Esta conversión consiste en obtener un número binario equivalente de un número hexadecimal.

Cada símbolo hexadecimal equivale a un valor binario de 4 bits, debido a que el sistema hexadecimal tiene 16 símbolos y con 4 bits se pueden tener 16 combinaciones. Se puede utilizar la tabla 1.7 como equivalencias

Con base en lo anterior, si se tiene el número octal $B45A_{16}$ el equivalente binario sería:

B	4	5	*A*
1011	0100	0101	1010

Entonces el número binario es 1011010001011010_2

1.8 COMPLEMENTO A 1 Y COMPLEMENTO A 2

El complemento a 1 y complemento a 2 de números representados en sistemas diferentes al decimal, es supremamente importante, porque gracias a este concepto se puede representar números negativos. Estos conceptos son usados en computación para hacer operaciones aritméticas con números negativos.

1.8.1 Complemento a 1

El complemento a 1, consiste en obtener el valor que le hace falta a un símbolo para llegar al símbolo máximo de la base. Para el caso particular del sistema binario, el complemento a la base consiste en cambiar ceros por unos y unos por ceros, debido a que solo hay estos dos símbolos en el sistema.

El complemento a 1 del número binario 101101010_2 sería:

101101010
010010101

El complemento a 1 del número octal 27610_8 sería:

27610
50167

El complemento a 1 del número hexadecimal $A4B2_{16}$ sería:

A4B2
5B4D

El complemento a 1 del número decimal 846_{10} sería:

846
153

1.8.2. Complemento a 2

El complemento a 2, consiste en obtener el complemento a 1 y sumarle 1.

El complemento a 1 del número binario 101101010_2 sería:

101101010
010010101 + 1 = 010010110

El complemento a 1 del número octal 27610_8 sería:

27610
50167 + 1 = 50170

El complemento a 1 del número hexadecimal $A4B2_{16}$ sería:

A4B2
5B4D + 1 = *5B4E*

El complemento a 1 del número decimal 846_{10} sería:

846
153 + 1 = 154

1.9 OPERACIONES ARITMÉTICAS DE DIFERENTES SISTEMAS

Las operaciones aritméticas son fundamentales para cualquier tipo de proceso en las máquinas. Dentro de las operaciones aritméticas más comunes se encuentran la ***suma*** y ***resta***. Otras operaciones aritméticas que se basan además de la suma y resta son la ***multiplicación y división***.

En este capítulo se trabajarán las operaciones aritméticas suma y resta para los sistemas binario, octal y hexadecimal y se trabajarán las operaciones multiplicación y división únicamente para el sistema binario, debido a que para hacer multiplicación y división en otros sistemas es necesario conocer las tablas de multiplicar de dichos sistemas.

1.9.1 Suma en Binario

Existen unas reglas básicas para realizar la suma en binario. Como el sistema binario tiene dos símbolos, entonces sólo existen cuatro posibles sumas con dos cantidades de un bit. Ellas son:

0 + 0 = 0 con acarreo = 0
0 + 1 = 1 con acarreo = 0
1 + 0 = 1 con acarreo = 0
1 + 1 = 1 con acarreo = 1

El acarreo es un valor que debe ser utilizado en el siguiente peso del número binario.

La suma binaria de los sumandos $101101_2 + 1011_2$ es:

$$\begin{array}{cccccc} & 1 & 1 & 1 & 1 & \\ 1 & 0 & 1 & 1 & 0 & 1 \\ & & 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 \end{array}$$

En la suma del bit de menor peso, se tiene 1+1, el resultado es 0 con un acarreo que se coloca en el siguiente peso, dando como resultado total 111000_2

1.9.2 Suma en Octal

Para hacer la suma en octal es importante tener en cuenta que existen solo 8 símbolos que van del 0 al 7. Entonces si una suma, supera el 7, existe entonces un acarreo debido a que se excede la base. Por ejemplo si se tiene 5 + 6 enton-

ces a 5 se le agrega 6, sin embargo si a 5 se le agrega 2, llega al valor máximo de la base, si se le agrega 3, existe un acarreo y el resultado sería entonces 0. Por consiguiente 5 + 6 = 13, donde 1 es el acarreo y 3 el resultado.

Se puede representar también de la siguiente forma:

- Tomamos los símbolos del sistema octal
  0 1 2 3 4 5 6 7
- Nos ubicamos en el primer sumando que es 5
  0 1 2 3 4 <u>5</u> 6 7
- Adicionamos el segundo sumando
  3 4 5 6 1 2
  0 1 2 3 4 <u>5</u> 6 7

Se puede notar que al sumarle a 5 un 6, se excede la base y el resultado es 3, por consiguiente la suma da como resultado 13_8

La suma octal de los sumandos $3462_8 + 413_8$ es:

$$\begin{array}{r} 1 \\ 3\ 4\ 6\ 2 \\ \quad 4\ 1\ 3 \\ \hline 4\ 0\ 7\ 5 \end{array}$$

El resultado es 4075_8

1.9. 3 Suma en hexadecimal

Para hacer la suma en hexadecimal es importante tener en cuenta que existen 16 símbolos que van del 0 al 9 y de la A a la F. Entonces si una suma, supera la F, existe entonces un acarreo debido a que se excede la base. Por ejemplo si se tiene A + 8 entonces a A se le agrega 8. Por consiguiente A + 8 = 12, donde 1 es el acarreo y 2 el resultado.

Se puede representar también de la siguiente forma:

- Tomamos los símbolos del sistema hexadecimal

0 1 2 3 4 5 6 7 8 9 *A B C D E F*

- Nos ubicamos en el primer sumando que es A

0 1 2 3 4 5 6 7 8 9 <u>*A*</u> *B C D E F*

- Adicionamos el segundo sumando

6 7 8 1 2 3 4 5
0 1 2 3 4 5 6 7 8 9 <u>*A*</u> *B C D E F*

Se puede notar que al sumarle a A un 8, se excede la base y el resultado es 2, por consiguiente la suma da como resultado 12_{16}

La suma octal de los sumandos $A463_{16}$ + 13E16 es:

$$\begin{array}{cccc} & & 1 & \\ A & 4 & 6 & 3 \\ & 1 & 3 & E \\ \hline A & 5 & A & 1 \end{array}$$

El resultado es $A4A1_{16}$

1.9.4 Resta en Binario

Para realizar la resta en binario se requiere seguir un algoritmo descrito por los siguientes pasos:

- Se iguala el número de cifras en el minuendo y el sustraendo añadiendo ceros a la izquierda del número con menos cifras.
- Se conserva el minuendo.
- Se complementa a 1 el sustraendo.
- Se suma.
- Si existe acarreo, se suma el acarreo. La existencia del acarreo indica que el resultado es positivo.
- Si no existe acarreo, se complementa a 1 el resultado. La no existencia de acarreo indica que el resultado es negativo.

Con base en la anterior descripción, la resta $10110101_2 - 10001010_2$ es:

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

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

El resultado es 101011_2 positivo

Ahora por ejemplo la resta $10110101_2 - 11001010_2$ es:

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

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

El resultado es 10101_2 negativo

1.9.5 Resta en octal

Para realizar la resta en octal se aplica el mismo algoritmo de resta binaria.

Entonces, la resta $7156_8 - 4326_8$ es:

$$\begin{array}{rcccc} & 7 & 1 & 5 & 6 \\ - & 4 & 3 & 2 & 6 \\ \hline \end{array}$$

$$\begin{array}{rcccc} 1 & & 1 & & \\ & 7 & 1 & 5 & 6 \\ + & 3 & 4 & 5 & 1 \\ \hline 1 & 2 & 6 & 2 & 7 \\ & & & + & 1 \\ \hline & 2 & 6 & 3 & 0 \end{array}$$

El resultado es 2630_8 positivo

Ahora por ejemplo la resta $3120_8 - 4033_8$ es:

$$\begin{array}{rcccc} & 3 & 1 & 2 & 0 \\ - & 4 & 0 & 3 & 3 \\ \hline \end{array}$$

$$\begin{array}{rcccc} & 1 & & & \\ & 3 & 1 & 2 & 0 \\ + & 3 & 7 & 4 & 4 \\ \hline 0 & 7 & 0 & 6 & 4 \\ \hline - & 0 & 7 & 1 & 3 \end{array}$$

El resultado es 713_8 negativo

1.9.6 Resta en hexadecimal

Para realizar la resta en octal se aplica el mismo algoritmo de resta binaria.

Entonces, la resta $A4B6_{16} - 4CD6_{16}$ es:

$$\begin{array}{cccc} A & 4 & B & 6 \\ - \; 4 & C & D & 6 \\ \hline \end{array}$$

$$\begin{array}{cccccc} 1 & & & & & \\ & A & 4 & B & 6 \\ + & B & 3 & 2 & 9 \\ \hline 1 & 5 & 7 & D & F \\ & & & + & 1 \\ \hline & 5 & 7 & E & 0 \end{array}$$

El resultado es $57E0_{16}$ positivo

Ahora por ejemplo la resta $B84_{16} - C1B3_{16}$ es:

$$\begin{array}{cccc} & B & 8 & 4 \\ - \; C & 1 & B & 3 \\ \hline \end{array}$$

$$\begin{array}{cccccc} & 1 & & 1 & & \\ & & B & 8 & 4 \\ + & 3 & E & 4 & C \\ \hline 0 & 4 & 9 & D & 0 \\ \hline - & B & 6 & 2 & F \end{array}$$

El resultado es $B62F_{16}$ negativo

1.9.7 Multiplicación en binario

Existen dos algoritmos para realizar la multiplicación en binario. Un algoritmo se basa en el algoritmo de multiplicación decimal y otro algoritmo se basa en rotaciones. Para cualquiera de los dos algoritmos es necesario conocer las tablas de multiplicar en binario, éstas son:

0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1

Algoritmo de multiplicación decimal para binario

Dado un multiplicando binario y un multiplicador binario, la multiplicación se realiza multiplicando el bit de menor peso del multiplicador con el multiplicando. Se realiza el algoritmo para todos los bits del multiplicador. Finalmente se realiza la suma de los resultados.

Por ejemplo la multiplicación de 110101_2 * 10011 es:

				1	1	0	1	0	1
			X		1	0	0	1	1
				1	1	0	1	0	1
			1	1	0	1	0	1	
		0	0	0	0	0	0		
	0	0	0	0	0	0			
1	1	0	1	0	1				
1	1	1	1	1	0	1	1	1	1

El resultado es 1111101111_2

Algoritmo de multiplicación binario por rotaciones

Dado un multiplicando binario y un multiplicador binario, la multiplicación por rotaciones procede con el siguiente algoritmo:

Se inicializa el resultado con 0 binario (La cantidad de bits, depende de la cantidad de bits mayor entre el multiplicando y multiplicador).

Se toma el primer bit del multiplicador de izquierda a derecha.

Si el bit tomado es 0, se debe rotar el resultado a la izquierda (para este caso en particular, rotar a la izquierda es equivalente a agregar un cero a la derecha).

Si el bit tomado es 1, se debe rotar el resultado a la izquierda y sumar el multiplicando.

Se repite el proceso desde el paso 2 hasta el 4 para el siguiente bit del multiplicador hasta el último bit de la derecha del multiplicador.

Por ejemplo la multiplicación de 110101_2 * 10011 es:

Primer paso: se inicializa el resultado con 0. Se toma el primer bit de derecha a izquierda del multiplicador. Como es 1 se rota el resultado a la derecha y se suma el multiplicando.

		1	1	0	1	0	1	
X			1	0	0	1	1	
	0	0	0	0	0	0	0	0
			1	1	0	1	0	1
			1	1	0	1	0	1

Segundo paso: se toma el siguiente bit de izquierda a derecha del multiplicador. Como es 0 se rota el resultado a la derecha.

		1	1	0	1	0	1	
X			1	0	0	1	1	
	0	0	0	0	0	0	0	
			1	1	0	1	0	1
			1	1	0	1	0	1	0

Tercer paso: se toma el siguiente bit de izquierda a derecha del multiplicador. Como es 0 se rota el resultado a la derecha.

		1	1	0	1	0	1		
X			1	0	0	1	1		
	0	0	0	0	0	0	0		
			1	1	0	1	0	1	
			1	1	0	1	0	1	0	0

Cuarto paso: se toma el siguiente bit de izquierda a derecha del multiplicador. Como es 1 se rota el resultado a la derecha y se suma el multiplicando.

		1	1	0	1	0	1			
X			1	0	0	1	1			
	0	0	0	0	0	0	0			
			1	1	0	1	0	1		
			1	1	0	1	0	1	0	0	0
						1	1	0	1	0	1
			1	1	1	0	1	1	1	0	1

Quinto paso: se toma el siguiente bit de izquierda a derecha del multiplicador. Como es 1 se rota el resultado a la derecha y se suma el multiplicando.

	1	1	0	1	0	1					
X		1	0	0	1	1					
	0	0	0	0	0	0	0				
		1	1	0	1	0	1				
		1	1	0	1	0	1	0	0	0	
					1	1	0	1	0	1	
		1	1	1	0	1	1	1	0	1	0
						1	1	0	1	0	1
		1	1	1	1	1	0	1	1	1	1

El resultado es 1111101111_2

1.9.8 División en binario

El algoritmo de división en binario más comúnmente utilizado se basa en el algoritmo de restas sucesivas. Este algoritmo debe seguir estos pasos:

Se iguala el número de bits en el minuendo y en el divisor agregando ceros a la izquierda en el número con menor cantidad de bits.

Se inicializa el cociente con 0.

Si el dividendo es mayor que el divisor, se le resta al dividendo el divisor y se incrementa el cociente. El resultado de la resta pasa a ser el nuevo dividendo para la siguiente resta.

Si el dividendo es menor que el divisor, la división ha terminado. El resultado está determinado por el cociente y el resultado de la última resta.

Se repite el paso 2 y 3 hasta que el dividendo sea menor que el divisor.

Por ejemplo la división de 11011_2 / 110 es:

Primer paso: como el dividendo es mayor que el divisor, se resta el dividendo menos el divisor. Se incrementa el cociente y el resultado de la resta pasa a ser el nuevo divisor.

11011	00110
11001	1
110100	
+1	
10101	

Segundo paso: como el dividendo es mayor que el divisor, se resta el dividendo menos el divisor. Se incrementa el cociente y el resultado de la resta pasa a ser el nuevo divisor.

```
 11011 | 00110
 11001   10
------
110100
   +1
------
 10101
 11001
------
101110
   +1
------
 01111
```

Tercer paso: como el dividendo es mayor que el divisor, se resta el dividendo menos el divisor. Se incrementa el cociente y el resultado de la resta pasa a ser el nuevo divisor.

```
 11011 | 00110
 11001   11
------
110100
   +1
------
 10101
 11001
------
101110
   +1
------
 01111
 11001
------
101000
   +1
------
 01001
```

Cuarto paso: como el dividendo es mayor que el divisor, se resta el dividendo menos el divisor. Se incrementa el cociente y el resultado de la resta pasa a ser el nuevo divisor.

```
 11011 | 00110
 11001 | 100
------
110100
   +1
------
 10101
 11001
------
101110
   +1
------
 01111
 11001
------
101000
   +1
------
 01001
 11001
------
100010
   +1
------
 00011
```

Quinto paso: como el dividendo es menor que el divisor, el resultado de la división es: 100_2 con residuo 11_2.

Ejercicios propuestos

1. Convertir de decimal a binario el número 1845.
2. Convertir de decimal a octal el número 2351.
3. Convertir de decimal a hexadecimal el número 5694.
4. Convertir de binario a decimal el número 10010101101.
5. Convertir de binario a octal el número 10010010100.
6. Convertir de binario a hexadecimal el número 110101100010.
7. Convertir de octal a decimal el número 7461.
8. Convertir de octal a binario el número 6541.
9. Convertir de hexadecimal a decimal el número A45F.
10. Convertir de hexadecimal a binario el número BC20.
11. Sumar en binario los números 101101 + 100110
12. Sumar en octal los números 7461 + 6201
13. Sumar en hexadecimal los números ACD + B49
14. Restar en binario los números 101101 - 100110
15. Restar en octal los números 7461 - 6201
16. Restar en hexadecimal los números ACD - B49
17. Multiplicar en binario los números 100101110 x 100101011
18. Dividir en binario los números 100101110 x 11011