Nivel de Lógica Digital: Aritmética en base "b"

Ing. Ronald Caravaca Mora

IF4000 - Arquitectura de Computadores
Informática Empresarial

21 de agosto de 2021

Introducción I

1. El actual periodo tecnológico se conoce como “era digital”.
2. Una característica de los sistemas digitales es su capacidad para manipular elementos discretos de información.
3. Las primeras computadoras digitales se usaron para efectuar cálculos numéricos. En este caso, los elementos discretos que se usaron fueron los dígitos. De ahí el término **digital**.
4. Los sistemas digitales electrónicos actuales, las señales emplean sólo dos valores discretos, por lo que decimos que son binarios. Un dígito binario, llamado bit (binary digit), tiene dos valores: 0 y 1.

Introducción II

5. El sistema numérico que se utiliza es el binario.
6. Tanto los datos como las instrucciones están codificados en un sistema binario.
7. Los bits se pueden agrupar en pequeños conjunto de 8 bits o de 16 bits, para cual se utilizan los sistemas octal y hexadecimal.

Sistemas de notación posicional I

Los sistemas de notación posicional están formados por un conjunto de n cantidad de símbolos, cuya combinación representa valores diferentes. El concepto de criterio posicional es que cada dígito tiene un **peso** distinto según el lugar que ocupa; el peso es la base elevada a la posición que ocupa dentro del número. La suma de cada dígito multiplicado por su **peso** permitirá obtener el valor final del número.

En todo sistema posicional, dada una base B se tienen B dígitos posibles, de 0 a $B - 1$.

10

B - 1 = 0 a X

10 - 1 = 0 a 9

Sistemas de notación posicional II

Al símbolo con menor peso se le conoce como: ***"menos significativo"***, mientras que al símbolo con mayor peso se le conoce como: ***"mas significativo"***.

En el caso del sistema binario se tiene:

- Bit menos significativo o LSB (Less Significant Bit).
- Bit mas significativo o MSB (Most Significant Bit).

Expresión generalizada

Se expande el número en base B en la siguiente serie de potencias:

base

$$N = d_n \cdot B^n + d_{n-1} \cdot B^{n-1} + \ldots + d_1 \cdot B^1 + d_0 \cdot B^0$$

digitos

en cualquier base.

Expresión generalizada

Se expande el número en base B en la siguiente serie de potencias:

$$N = d_n \cdot B^n + d_{n-1} \cdot B^{n-1} + \ldots + d_1 \cdot B^1 + d_0 \cdot B^0$$

Por ejemplo, el sistema decimal es un sistema de notación posicional formado por 10 símbolos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Entonces, el número 1023 estará conformado por la suma de las siguientes cantidades:

m	c	d	u
10^3	10^2	10^1	10^0
1	0	2	3_{10}

Expresión generalizada

Se expande el número en base B en la siguiente serie de potencias:

$$N = d_n \cdot B^n + d_{n-1} \cdot B^{n-1} + \ldots + d_1 \cdot B^1 + d_0 \cdot B^0$$

Por ejemplo, el sistema decimal es un sistema de notación posicional formado por 10 símbolos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Entonces, el número 1023 estará conformado por la suma de las siguientes cantidades:

$$\begin{array}{cccc} 10^3 & 10^2 & 10^1 & 10^0 \\ 1 & 0 & 2 & 3_{10} \end{array}$$

digito base

$$1 \cdot 10^{3} + 0 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 = 1000 + 0 + 20 + 3 = 1023_{10}$$

3 2 1 0

Sistema binario o base 2

Es un sistema de notación posicional, formado por dos símbolos (0,1) a los que se denomina bits. También se les llama:

0	**1**
OFF	ON
Falso	Verdadero
No	Si

Sistema binario o base 2

Es un sistema de notación posicional, formado por dos símbolos (0,1) a los que se denomina bits. También se les llama:

0	**1**
OFF	ON
Falso	Verdadero
No	Si

Ejemplo:

$$1 \quad 1 \quad 0 \quad 1 \quad 1_2$$

Sistema binario o base 2

Es un sistema de notación posicional, formado por dos símbolos (0,1) a los que se denomina bits. También se les llama:

0	**1**
OFF	ON
Falso	Verdadero
No	Si

Ejemplo:

$$1\ 1\ 0\ 1\ 1_2$$

base

$$1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 16 + 8 + 0 + 2 + 1 = 27_{10}$$

Sistema octal o base 8

Es un sistema posicional formado por ocho símbolos (0, 1, 2, 3, 4, 5, 6, 7); el peso de cada cifra son las potencias sucesivas de 8.

Sistema octal o base 8

Es un sistema posicional formado por ocho símbolos (0, 1, 2, 3, 4, 5, 6, 7); el peso de cada cifra son las potencias sucesivas de 8.

Ejemplo:

$$2\ 7\ 5_8$$

Sistema octal o base 8

Es un sistema posicional formado por ocho símbolos (0, 1, 2, 3, 4, 5, 6, 7); el peso de cada cifra son las potencias sucesivas de 8.

Ejemplo:

$$275_8$$

$$2 \cdot 8^2 + 7 \cdot 8^1 + 5 \cdot 8^0 = 128 + 56 + 5 = 189_{10}$$

Base

Sistema hexadecimal o base 16

En general se abrevia con las siglas ''Hexa", "H", "h" y algunos casos "x", es un sistema posicional formado por dieciséis símbolos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) que representan los valores decimales de 0 a 15; el peso de cada cifra son las potencias sucesivas de 16.

Sistema hexadecimal o base 16

En general se abrevia con las siglas "Hexa", "H", "h" y algunos casos "x", es un sistema posicional formado por dieciséis símbolos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) que representan los valores decimales de 0 a 15; el peso de cada cifra son las potencias sucesivas de 16.

Ejemplo:

$$1 \;\; A \;\; 2_{16}$$

Sistema hexadecimal o base 16

En general se abrevia con las siglas ”Hexa“, “H”, “h” y algunos casos “x”, es un sistema posicional formado por dieciséis símbolos (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) que representan los valores decimales de 0 a 15; el peso de cada cifra son las potencias sucesivas de 16.

Ejemplo:

$$1\,A\,2_{16}$$

base

$$1 \cdot 16^2 + 10 \cdot 16^1 + 2 \cdot 16^0 = 256 + 160 + 2 = 418_{10}$$

Expresión generalizada con punto fraccionario

Que pasa si queremos expresar el número $\underbrace{11010}\!,\underbrace{11}_2$ en base 10?

Expresión generalizada con punto fraccionario

Que pasa si queremos expresar el número $11010{,}11_2$ en base 10?

$$N = \overbrace{d_n \cdot B^n + d_{n-1} \cdot B^{n-1} + \ldots + d_1 \cdot B^1 + d_0 \cdot B^0}^{\text{Primero Parte}} + d_{-1} \cdot B^{-1} +$$

$$+d_{-2} \cdot B^{-2} + \ldots + d_{-m} \cdot B^{-m}$$

segunda parte

Expresión generalizada con punto fraccionario

Que pasa si queremos expresar el número $11010{,}11_2$ en base 10?

$$N = d_n \cdot B^n + d_{n-1} \cdot B^{n-1} + \ldots + d_1 \cdot B^1 + d_0 \cdot B^0 + d_{-1} \cdot B^{-1} + \\ + d_{-2} \cdot B^{-2} + \ldots + d_{-m} \cdot B^{-m}$$

base

decimal

$$N = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}$$

$$N = 16 + 8 + 0 + 2 + 0 + \frac{1}{2} + \frac{1}{2^2} \rightarrow \text{decimal}$$

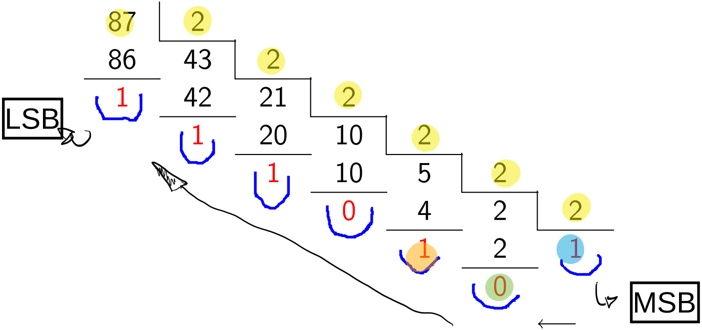
$$N = 26{,}75_{10}$$

Conversión entre bases I

1. Para convertir cualquier número de cualquier base B a base 10, se expande el número usando la expresión generalizada con series de potencias como se explicó anteriormente.
2. La conversión inversa, de base 10 a base B se efectúa dividiendo el número y todos sus cocientes sucesivos entre B y acumulando los residuos.
3. Si el número tiene punto fraccionario, será necesario separar la parte entera de la parte fraccionaria, pues cada parte se convierte de manera distinta.

Conversión entre bases II

Ejemplo 1. Si queremos convertir el número 87_{10} a base 2.



$$87_{10} = 1010111_2$$

Conversión entre bases III

Ejemplo 2. Si queremos convertir el número 87_{10} a base 8.

$$
\begin{array}{r|l}
87 & 8 \\ \hline
\end{array}
\qquad
\begin{array}{ccc}
87 & 8 & \\
80 & 10 & 8 \\
7 & 8 & 1 \\
 & 2 &
\end{array}
$$

$\longleftarrow$

$$87_{10} = 127_8$$

Conversión entre bases IV

Ejemplo 3. Si queremos convertir el número 87_{10} a base 16.

$$\begin{array}{r|l} 87 & 16 \\ \cline{2-2} 80 & 5 \\ \cline{1-1} 7 & \end{array}$$

$$\longleftarrow$$

$$87_{10} = 57_{16}$$

Conversión entre bases V

Ejemplo 4. Si queremos convertir el número 418_{10} a base 16.

$$\begin{array}{r|l l}
418 & 16 & \\ \cline{2-2}
416 & 26 & \multicolumn{1}{|l}{16} \\ \cline{1-1} \cline{3-3}
\color{red}{2} & 16 & \color{red}{1} \quad 1 \\ \cline{2-2}
3 & \color{red}{10} \quad A & \\
 & \longleftarrow &
\end{array}$$

$$418_{10} = \cancel{1102_{16}}$$

$$418_{10} = 1A2_{16}$$

1 A 3

Conversión entre bases VI

Ejemplo 5. Si queremos convertir el número $0{,}6875_{10}$ a base 2.

$$
\begin{aligned}
0{,}6875 \cdot 2 &= 1{,}375 \\
0{,}375 \cdot 2 &= 0{,}75 \\
0{,}75 \cdot 2 &= 1{,}5 \\
0{,}5 \cdot 2 &= 1{,}0
\end{aligned}
$$

MSB

LSB

$$0{,}6875_{10} = ,1011_2$$

Conversión entre bases VII

Ejemplo 8. Si queremos convertir el número $0{,}513_{10}$ a base 8.

$$
\begin{aligned}
0{,}513 \cdot 8 &= 4{,}104 \\
0{,}104 \cdot 8 &= 0{,}832 \\
0{,}832 \cdot 8 &= 6{,}656 \\
0{,}656 \cdot 8 &= 5{,}248 \\
0{,}248 \cdot 8 &= 1{,}984 \\
0{,}984 \cdot 8 &= 7{,}872 \\
&\vdots
\end{aligned}
$$

$$0{,}513_{10} = 0{,}406517\ldots_8 \tag{1}$$

Conversión entre bases VIII

Como practica:

La conversión de números decimales que tienen tanto parte entera como parte fraccionaria se efectúa convirtiendo por separado las dos partes y combinando después las dos respuestas. Por tanto, desarrolle la conversión de los siguientes números:

- Convertir el número $41{,}23_{10}$ a base 2.
- Convertir el número $153{,}56_{10}$ a base 8.

1

Solución:

41	2				
40	20	2			
1	20	10	2		
	0	10	5	2	
		0	4	2	2
			1	2	1
				0	

$\longleftarrow$

$$41_{10} = 101001_2$$

$$0{,}23 \cdot 2 = 0{,}46$$

$$0{,}46 \cdot 2 = 0{,}92$$

$$0{,}92 \cdot 2 = 1{,}84$$

$$0{,}84 \cdot 2 = 1{,}68$$

$$0{,}68 \cdot 2 = 1{,}36$$

$$0{,}36 \cdot 2 = 0{,}72$$

$$0{,}72 \cdot 2 = 1{,}44$$

$$\vdots$$

$$41{,}23_{10} = 101001{,}0011101\ldots_2$$

Solución:

$$
\begin{array}{r|l|l}
153 & 8 & \\
\cline{2-2}
152 & 19 & 8 \\
\cline{1-1}\cline{3-3}
\color{red}{1} & 16 & \color{red}{2} \\
\cline{2-2}
 & \color{red}{3} &
\end{array}
$$

$$\longleftarrow$$

$$153,\cancel{56}_{10} = 231_8$$

8

$$0{,}56 \cdot \cancel{2} = {\color{red}4},48$$

$$0{,}48 \cdot \cancel{2} = {\color{red}3},84$$

$$0{,}84 \cdot \cancel{2} = {\color{red}6},72$$

$$0{,}72 \cdot \cancel{2} = {\color{red}5},76$$

$$0{,}76 \cdot \cancel{2} = {\color{red}6},08$$

$$0{,}08 \cdot \cancel{2} = {\color{red}0},64$$

$$0{,}64 \cdot \cancel{2} = {\color{red}5},12$$

$$\vdots$$

$$153{,}56_{10} = 231{,}4365605\ldots_{8}$$

Representación de números

Números con diferente base

Decimal (base 10)	Binario (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	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

Representación de números

4 2 1

$$
\begin{array}{ccccccccccl}
10 & 110 & 001 & 101 & \underline{011} & ,111 & 100 & 000 & 110_2 & = 26153,7406_8 \\
2 & 6 & 1 & 5 & 3 & ,7 & 4 & 0 & 6 &
\end{array}
$$

$$
\begin{array}{ccccccl}
10 & 1100 & 0110 & 1011 & ,1111 & 0010 & = 2C6B,F2 \\
2 & C & 6 & B & ,F & 2 &
\end{array}
\qquad (2)
$$

Representación de números

Pontecia de 2	Valor en binario	Valor en decimal
2^0	1	1
2^1	10	2
2^2	100	4
2^3	1000	8
2^4	10000	16
2^5	100000	32
2^6	1000000	64

Representación de números

Pontecia de 2	Valor en binario	Valor en decimal
2^7	10000000	128
2^8	100000000	256
2^9	1000000000	512
2^{10}	10000000000	1024
2^{11}	100000000000	2048
2^{12}	1000000000000	4096
2^{13}	10000000000000	8182

Suma de números binarios

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \text{ y acarreo } 1$$

Suma de números binarios

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0 \textbf{ y acarreo 1}$$

Ejemplos: Sumar en binario los números 8_{10} y 3_{10} y los números 11_{10} y 5_{10} con 4 bits.

$$\begin{array}{r} 1000 \\ 0011 \\ +\underline{} \\ 1011 \end{array} \qquad \begin{array}{r} 1011 \\ 0101 \\ +\underline{} \\ 10000 \end{array}$$

→ Overflow

Representación de números negativos

La representación de números negativos se puede hacer de dos maneras:

1. Signo y magnitud
2. Complemento

Signo y magnitud

Se utiliza el bit mas significativo (MSB) para codificar el signo, y los bits restantes para la magnitud.

Decimal	Magnitud con signo
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
−0	1000
−1	1001
−2	1010
−3	1011
−4	1100
−5	1101
−6	1110
−7	1111
−8	—

Signo y magnitud

Se utiliza el bit mas significativo (MSB) para codificar el signo, y los bits restantes para la magnitud.

Ejemplo:
Con 4 bits puedo representar 16 números; los positivos del 0 a 15, o los negativos de -7 al -0 y lo positivos del 0 al 7.

0 → positivo

1 → negativo

Decimal	Magnitud con signo
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
−0	1000
−1	1001
−2	1010
−3	1011
−4	1100
−5	1101
−6	1110
−7	1111
−8	—

Signo y magnitud

Se utiliza el bit mas significativo (MSB) para codificar el signo, y los bits restantes para la magnitud.

Ejemplo:
Con 4 bits puedo representar 16 números; los positivos del 0 a 15, o los negativos de -7 al -0 y lo positivos del 0 al 7.

Problemas:

1. Complejidad de implementación.
2. Doble representación del 0.

No se usa

Decimal	Magnitud con signo
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
−0	1000
−1	1001
−2	1010
−3	1011
−4	1100
−5	1101
−6	1110
−7	1111
−8	—

Números en complemento de base

Dado un número N en base B que tiene n dígitos, el complemento a $(B-1)$ de N se define como $(B^n - 1) - N$.

Números en complemento de base

Dado un número N en base B que tiene n dígitos, el complemento a $(B-1)$ de N se define como $(B^n - 1) - N$.

En el caso de números decimales, $B = 10$ y $B - 1 = 9$, así que el complemento a **nueve** de N es $(10^n - 1) - N$. En este caso, 10^n representa un número que consiste en un uno seguido de n ceros y $10^n - 1$ es un número representado por n nueves. Por ejemplo, si $n = 4$, tenemos $10^4 = 10000$ y $10^4 - 1 = 9999$.

Números en complemento de base

Dado un número N en base B que tiene n dígitos, el complemento a $(B-1)$ de N se define como $(B^n - 1) - N$.

base decimal

En el caso de números decimales, $B = 10$ y $B - 1 = 9$, así que el complemento a **nueve** de N es $(10^n - 1) - N$. En este caso, 10^n representa un número que consiste en un uno seguido de n ceros y $10^n - 1$ es un número representado por n nueves. Por ejemplo, si $n = 4$, tenemos $10^4 = 10000$ y $10^4 - 1 = 9999$.

6 dígito 6 9's

Ejemplo:

El complemento a nueve de 546700 es 999999 − 546700 = 453299.
El complemento a nueve de 012398 es 999999 − 012398 = 987601.

Números en complemento de base

Dado un número N en base B que tiene n dígitos, el complemento a $(B-1)$ de N se define como $(B^n - 1) - N$.

En el caso de números decimales, $B = 10$ y $B - 1 = 9$, así que el complemento a **nueve** de N es $(10^n - 1) - N$. En este caso, 10^n representa un número que consiste en un uno seguido de n ceros y $10^n - 1$ es un número representado por n nueves. Por ejemplo, si $n = 4$, tenemos $10^4 = 10000$ y $10^4 - 1 = 9999$.

Ejemplo:

6 digitos   6 digitos

El complemento a nueve de 546700 es 999999 − 546700 = 453299.
El complemento a nueve de 012398 es 999999 − 012398 = 987601.

El complemento a (B-1) de los números octales y hexadecimales se obtiene restando cada dígito a 7 y F (15 decimal), respectivamente.

Números en complemento de base

En el caso de los números binarios, $B = 2$ y $B - 1 = 1$, así que el complemento a **uno** de N es $(2^n - 1) - N$. Por ejemplo, si $n = 4$, tenemos $2^4 = 10000_2$ y $2^4 - 1 = 1111_2$. Así, el complemento a uno de un número binario se obtiene restando cada dígito a uno. Sin embargo, al restar dígitos binarios a 1 podemos tener $1 - 0 = 1$ o bien $1 - 1 = 0$, lo que hace que el bit cambie de 0 a 1 o de 1 a 0. Por tanto, el complemento a uno de un número binario se forma cambiando los unos a ceros y los ceros a unos. He aquí algunos ejemplos numéricos:

Números en complemento de base

En el caso de los números binarios, $B = 2$ y $B - 1 = 1$, así que el complemento a **uno** de N es $(2^n - 1) - N$. Por ejemplo, si $n = 4$, tenemos $24 = 10000_2$ y $24 - 1 = 1111_2$. Así, el complemento a uno de un número binario se obtiene restando cada dígito a uno. Sin embargo, al restar dígitos binarios a 1 podemos tener $1 - 0 = 1$ o bien $1 - 1 = 0$, lo que hace que el bit cambie de 0 a 1 o de 1 a 0. Por tanto, el complemento a uno de un número binario se forma cambiando los unos a ceros y los ceros a unos. He aquí algunos ejemplos numéricos:

no cambia 0, 1

Ejemplo:
El complemento a uno de 1011000 es 0100111.
El complemento a uno de 0101101 es 1010010.

Complemento a 2

El complemento a B de un número N de n dígitos en base B se define como $B^n - N$, para $N \neq 0$, y 0 para $N = 0$. Si comparamos con el complemento a $(B-1)$, veremos que el complemento a B se obtiene sumando 1 al complemento a $(B-1)$, ya que $B^n - N = [(B^n - 1) - N] + 1$.

Cuando $B = 2$, se tiene que el **complemento a 2** se obtiene sumando un 1 al complemento a uno.

Complemento a 2

complemento 9, 2 y 1

1101100		0110111	→ Número
0010011		1001000	→ Comp. a 1
0000001		0000001	→ Se suma 1
+		+	
0010100		1001001	→ Comp. a 2
↳ positivo		↳ negativo	

Resta de números binarios

La resta de dos números de n dígitos sin signo, $M - N$ en base B se efectúa así:

1. Sume el minuendo, M, al complemento a B del sustraendo, N. Esto da $M + (B^n - N) = M - N + B^n$.
2. Si $M \geq N$, la suma producirá un acarreo final, B^n, que puede desecharse; lo que queda es el resultado $M - N$. B
3. Si $M < N$, la suma no produce un acarreo final y es igual a r n-(N-M), que es el complemento a B de $(N - M)$. Para obtener la respuesta en una forma conocida, se toma el complemento a B de la suma y se le antepone un signo de menos.

Resta de números binarios

Dados los números binarios $X = 1010100$ y $Y = 1000011$, realizar las restas a) $X - Y$ y b) $Y - X$ empleando complementos a dos.

1010100 → Número X			1000011 → Número Y	
0111101 → Comp. a 2 de Y	calcular		0101100 → Comp. a 2 de X	calcular
+			+	
10010001			1101111	

acarreo ignorar

solo si $X > Y$

$Y < X$

Resta de números binarios

7 digitos

Dados los números binarios $X = 1010100$ y $Y = 1000011$, realizar las restas a) $X - Y$ y b) $Y - X$ empleando complementos a dos.

$$
\begin{array}{rl}
1010100 & \rightarrow \text{Número X} \\
0111101 & \rightarrow \boxed{\text{Comp. a 2 de Y}} \\
+\overline{} & \\
\cancel{1}\underbrace{0010001}_{\text{7 digitos}} &
\end{array}
\qquad
\begin{array}{rl}
1000011 & \rightarrow \text{Número Y} \\
0101100 & \rightarrow \text{Comp. a 2 de X} \\
+\overline{} & \\
1101111 &
\end{array}
$$

1010100 X

0101011 complemento

1

0101100

complemento 2 !!!

$$84 - 67 = 17\ (0010001_2)$$

$$67 - 84 = -17\ (1101111_2)$$

Decimal	Complemento a dos con signo	Complemento a uno con signo	Magnitud con signo
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

2

1 1
0101
1100
10001

1

0100
1011
1100

Desborde (Overflow)

- No pasa con números de diferente signo.
- Pero si pasa con numero de igual signo cuando resultado supera la cantidad de bits disponibles o el bit de signo del resultado no es consistente.

Desborde (Overflow)

- No pasa con números de diferente signo.
- Pero si pasa con numero de igual signo cuando resultado supera la cantidad de bits disponibles o el bit de signo del resultado no es consistente.

tiene que iniciar con 0

Ejemplo:

$$
\begin{array}{lll}
1101 \rightarrow -3 & 0101 \rightarrow 5 & 0111 \rightarrow 7 \\
1010 \rightarrow -6 & 0110 \rightarrow 6 & 0001 \rightarrow 1 \\
+\_\_\_\_\_ & +\_\_\_\_\_ & +\_\_\_\_\_ \\
10111 \rightarrow \boxed{+7} & 1011 \rightarrow -5 & 1000 \rightarrow -8
\end{array}
$$

Si se necesita

se necesita

se necesita

Desborde (Overflow)

Como se soluciona?

1. Analizando el bit de signo: Si ambos números tiene el mismo signo, debemos esperar que el resultado mantenga el mismo signo, si no es así, se levanta un bandera que le avise al procesador que hubo *overflow*.
2. Cuando hay un acarreo al sumar los bits MSB, se levanta un bandera que le avise al procesador que hubo *overflow*.

Códigos binarios I

Dígito decimal	Código BCD	Código 6-3-1-1	Código exceso 3	Código 2 de 5	Código Gray
0	0000	0000	0011	00011	0000
1	0001	0001	0100	00101	0001
2	0010	0011	0101	00110	0011
3	0011	0100	0110	01001	0010
4	0100	0101	0111	01010	0110
5	0101	0111	1000	01100	1110
6	0110	1000	1001	10001	1010
7	0111	1001	1010	10010	1011
8	1000	1011	1011	10100	1001
9	1001	1100	1100	11000	1000

Códigos binarios II

Binary	Gray code	One-hot
000	000	00000001
001	001	00000010
010	011	00000100
011	010	00001000
100	110	00010000
101	111	00100000
110	101	01000000
111	100	10000000

Preguntas?