Ejercicios resueltos

Realizar la siguiente resta y expresar el resultado en hexadecimal:

$A6_h - 95_d =$

La modalidad que emplearemos para resolver las restas es la utilizada por el microprocesador: la resta como suma del complemento a dos del sustraendo (o simplemente suma por complemento a dos).

Paso 1: Operandos.

i. Pasaje de $A6_h$ a binario.

Recordemos que cada dígito hexadecimal se expresa como los cuatro dígitos binarios equivalentes. Así:

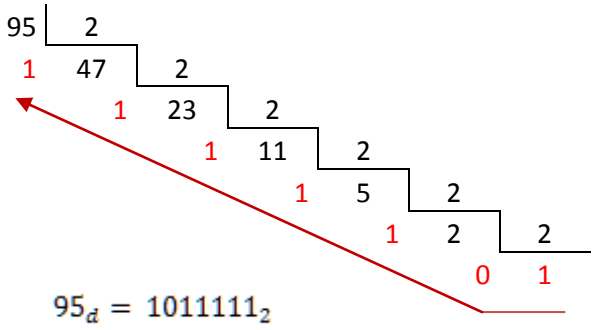
A: 1010

6: 0110

$A6_h = 10100110_b$

ii. 95_d a binario.

Por divisiones sucesivas por dos resulta:



$95_d = 1011111_2$

iii. Del sustraendo (en este caso 95_d) debe obtenerse el complemento a dos (Ca2), para poder realizar la resta.

En este caso además debemos tomar en cuenta que la cantidad de bits del sustraendo y el minuendo debe ser la misma, por esto completamos con un cero a la izquierda de 1011111_2 **antes** de obtener el Ca2.

$$\text{Ca2}(01011111_2) = Ca1(01011111_2) + 1$$

Resolvemos el Ca2 de un número como el Ca1 del mismo, más 1. Para obtener el Ca1 de un número invertimos bit a bit 0 por 1 y 1 por 0.

$$\text{Ca2}(01011111_2) = Ca1(01011111_2) + 1$$

$$10100000_2 + 1$$

$$10100001_2$$

Paso 2: La operación.

Se realiza la suma entre el minuendo (primer valor) y el complemento a dos del sustraendo (segundo valor).

		1							Acarreo
	1	0	1	0	0	1	1	0	1º valor - minuendo
	1	0	1	0	0	0	0	1	2º valor - Ca2 del sustraendo
~~1~~	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**RESULTADO**
i	h	g	f	e	d	c	b	a	

Debe tenerse en cuenta que si de la suma de los dos bits más significativos (en este caso la columna h) resultase un acarreo, el mismo se descarta.

Paso 3: Expresar el resultado.

La consigna especifica que el resultado debe expresarse en hexadecimal. Como el mismo está dado en binario, tomaremos de a cuatro estos bits y los reemplazaremos por el dígito hexadecimal correspondiente.

0111: 7

0100: 6

entonces:

$$\mathbf{01000111_2 = 67_h}$$

Rta: $\mathbf{67_h}$

Realizar la siguiente suma y expresar el resultado en hexadecimal:

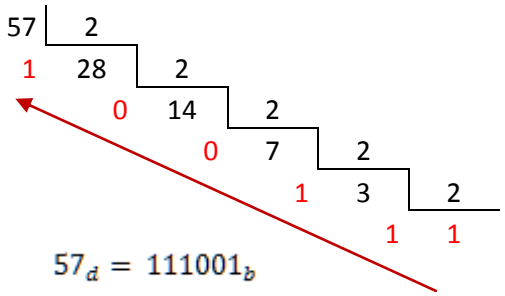
$$57_d + 313_4 + 1110011_b =$$

Como dice la consigna, debemos sumar estas tres cifras que se encuentran en bases 10, 4 y 2 respectivamente y luego dar el resultado en base 16.

Paso 1: Operandos.

Para poder realizar la suma tenemos que pasar los tres números a una misma base. Por una cuestión de sencillez y por ser la base en la que el procesador realiza las operaciones, elegiremos el sistema binario.

i. 57_d a binario, mediante divisiones sucesivas por 2:



$57_d = 111001_b$

ii. 313_4 a binario, pasando dígito a dígito de base 4 a base 2:

Base 4:	3	1	3
Base 2:	11	01	11

$313_d = 110111_b$

Nótese:

1) Que así como un dígito hexadecimal se expresa como cuatro dígitos binario y un dígito octal lo hace como tres bits, un dígito en base 4 se corresponde con dos de base dos, de acuerdo a la siguiente tabla:

Base 4	Bin
0	00
1	01
2	10
3	11

2) También puedo aplicar el método general que consiste en pasar de base *n* (en este caso 4) a base 10 y luego de base 10 a base 2.

Paso 2: Suma.

i. Alineamos a la derecha los tres números.

	10	10	1	1	1	1		Acarreos
		1	1	1	0	0	1	1º valor
		1	1	0	1	1	1	2º valor
	1	1	1	0	0	1	1	3º valor
1	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**RESULTADO**
h	g	f	e	d	c	b	a	

ii. Al sumar los bits de la columna a, nos encontramos con 1+1+1, cuyo resultado es 11_2 (o 3 en base 10); tal como haríamos en una suma "común", en base 10, anotamos el dígito de la derecha y acarreamos (o "nos llevamos") el de la izquierda.

iii. En la columna b nos tenemos que agregar el acarreo a los otros tres dígitos, por lo que nos queda 1+0+1+1, que nuevamente resulta 11_2. Procedemos igual que con la columna a.

iv. En la columna *e* debemos sumar tres unos más el acarreo de la columna *d* con lo que obtenemos 100_2 como resultado. ¿Y cómo hacemos? Nuevamente anotamos el dígito menos significativo como resultado de la columna y acarreamos 10_2(2 en base 10) a la columna *f*.
Nótese que también hubiera podido acarrear 0 (el segundo dígito de 100) a la columna *f* y 1 (el dígito más significativo) a la columna *g*.

Paso 3: Resultado.

El resultado es 11100011_2, pero según la consigna debe ser expresado en sistema hexadecimal.
Nuevamente recurrimos a la equivalencia directa que hay entre dígitos de bases que son potencias de 2 (bases 4, 8 o 16 por ejemplo) y dígitos binarios. En el caso del pasaje de binario a hexadecimal, 4bits se corresponden con un dígito hexadecimal.

Base 2:	1110	0011
Base 16:	E	3

Rta: $E3_h$

Representar en Ca2 el número -22_{10}, mediante 8 bits.

En el sistema de representación de enteros llamada *complemento a 2* o *Ca2* se siguen los siguientes criterios:

I. Si el número a representar es *mayor o igual* a 0 (cero), el valor se expresa como *binario sin signo*. Por ejemplo: $96_{10} \rightarrow 01100000$ (disponiendo de 8 bits).

II. Los números negativos son representados como el Ca2 de su valor absoluto: $96_{10} \rightarrow \text{Ca2}(01100000) = \text{Ca1}(01100000) + 1 = 10011111 + 1 = 10100000$
(nuevamente para 8 bits).

Teniendo en cuenta estos criterios podemos resolver el ejercicio propuesto:

Al ser -22 < 0, aplicamos II:

El valor absoluto, es decir 22, pasado a binario es 10110_2; entonces -22 se expresará como el *Ca2* de 00010110 (utilizando 8 bits):

$Ca2(00010110) = Ca1(00010110) + 1 = 11101001 + 1 = 11101010$

Indicar qué valores representan los siguientes bytes en complemento a dos:

1. 01010101
2. 10101010

1. El número comienza con 0 (cero), por lo que se puede determinar que es positivo. En tal caso calculamos el valor "normalmente":

01010101_2

$01010101_2 = 1 \times 2^0 + 1 \times 2^2 + 1 \times 2^4 + 1 \times 2^6 = 1 + 4 + 16 + 64 = 85$

2. En este caso observamos que el dígito más significativo es 1 (uno); esto indica que se trata de un número negativo cuyo valor absoluto está expresado como un complemento a dos.

10101010
$\rightarrow -Ca2(10101010) = -(Ca1(10101010) + 1) = -(01010101 + 1) =$
$-(01010110) = -(1 \times 2^1$
$+1 \times 2^2 + 1 \times 2^4 + 1 \times 2^6) = -(2 + 4 + 16 + 64) = -86$

Representar en Exceso a 2^{n-1} en un byte el número decimal -45.

La notación en exceso consiste en sumar un valor "conveniente" (por ejemplo 127) a los números de un intervalo (por ejemplo de -127 a 128) para poder representarlos (o codificarlos) mediante valores que sólo pueden ser positivos (por ejemplo de 0 a 255, para un byte). Esto también se puede ver como una función $f(x) = x + exceso$, en este caso: $f(x) = x + 127$

De esta manera -127 se relaciona con 0, -126 con 1, -125 con 2, -124 con 3, etc.

Valor	Represent.	Repr. Bin.
-127	0	00000000
-126	1	00000001
-125	2	00000010
-124	3	00000011
⋮	⋮	⋮
0	127	01111111
1	128	10000000
2	129	10000001
⋮	⋮	⋮
126	253	11111101
127	254	11111110

128	255	11111111

Normalmente se desea que el rango de valores que se puedan representar sea lo más simétrico posible. Por esto los excesos más habituales, para *n* bits, son 2^{n-1} y $2^{n-1} - 1$; por ejemplo, para 8 bits serían Exceso 127 y Exceso 128.

En el caso concreto de la consigna se indica que los bits son 8 y que el exceso es 2^{n-1} por lo que estamos hablando de exceso 128. Esto significa que la cantidad a sumar para representar los valores será 128 (la función será $f(x) = x + 128$).

Se desea representar -22, así que $f(-22) = -22 + 128 = 106 =$ 01101010

Indicar qué valor representa el byte 00110011 en exceso 127.

En este ejercicio se debe hacer el proceso inverso al del anterior, es decir pasar de un valor codificado o representado en exceso 127 al valor decimal correspondiente.

i. Por una cuestión de simplicidad pasamos a decimal el valor codificado. El byte 00110011 expresado en decimal es igual a 51.
ii. Este valor, 51, está "excedido" en 127, o sea que para poder averiguar el valor *representado* o *codificado* debemos restarle 127.
    51 - 127 = -76.

El valor representado es, entonces, -76.