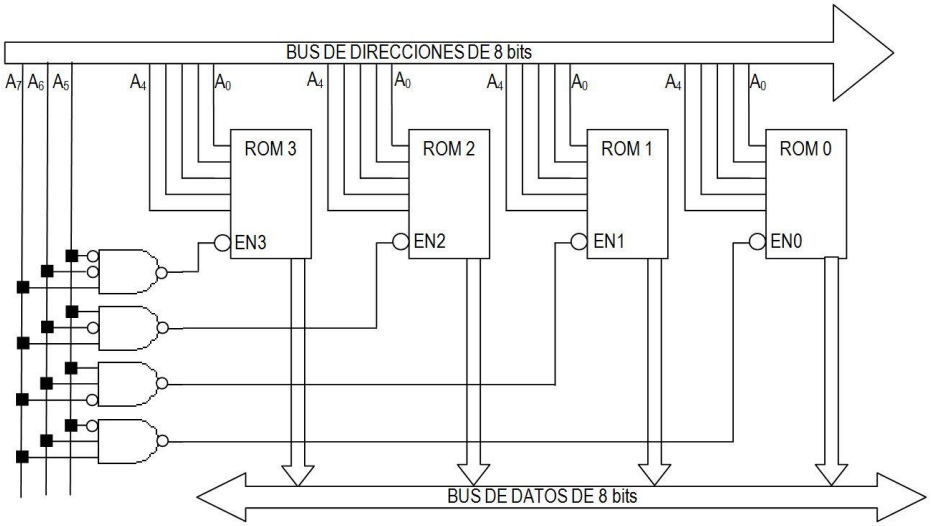
COMPUTER STRUCTURE
UNIT 6.- MEMORY MAPS. REAL NUMBERS OPERATIONS

6.1. A memory map includes 4 ROM chips, as shown in the figure. The address decoding circuit is included in the figure.



a) Maximum number of bytes for this system:

2^8 Bytes = 256 Bytes

b) Using the following table, write in hex and binary the highest and lowest address of each ROM chip.

		ROM 3	ROM 2	ROM 1	ROM 0
HIGH ADDRESS	**Hexa**	**9F**	**BF**	**7F**	**DF**
	Bin	**1001 1111**	**1011 1111**	**0111 1111**	**1101 1111**
LOW ADDRESS	**Hexa**	**80**	**A0**	**60**	**C0**
	Bin	**1000 0000**	**1010 0000**	**0110 0000**	**1100 0000**

6.2. The memory map of a microprocessor system is shown in the table. Please, design the address decoder. For the RAM memory, 2 kBytes chips are used which include R/W control signal and Chip Selection (CSx). The ROM chips are 4 kBytes each one with a single control signal CSx. The I/O peripherals are treated as RAM memory, with a single selection control signal for the I/O block. Design the control signals CSx of the address decoder in the following cases:

a) Using complete mapping: **a1)** supposing active high CSx ('1') and **a2)** supposing active low CSx ('0')

b) Using incomplete mapping: **b1)** supposing active high CSx ('1') y **b2)** supposing active low CSx ('0')

Address (hexa)		**Type**	**Size**	**Selection bit**	
Begin	**End**			**(1)**	**(2)**
0000	07FF	System RAM	2 kBytes	CS1	/CS1
0800	0FFF	User RAM	2 kBytes	CS2	/CS2
Not used zone					
B000	BFFF	I/O peripherals	4 kBytes	CS5	/CS5
Not used zone					
E000	EFFF	Tables ROM	4 kBytes	CS3	/CS3
F000	FFFF	Program ROM	4 kBytes	CS4	/CS4

SOLUTION:

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	CS1	CS2	CS3	CS4	CS5	/CS1	/CS2	/CS3	/CS4	/CS5
0	0	0	0	0	1	0	0	0	0	0	1	1	1	1
0	0	0	0	1	0	1	0	0	0	1	0	1	1	1
1	1	1	0	X	0	0	1	0	0	1	1	0	1	1
1	1	1	1	X	0	0	0	1	0	1	1	1	0	1
1	0	1	1	X	0	0	0	0	1	1	1	1	1	0

a1)

$CS1 = \overline{A_{15}}\ \overline{A_{14}}\ \overline{A_{13}}\ \overline{A_{12}}\ \overline{A_{11}}$

$CS2 = \overline{A_{15}}\ \overline{A_{14}}\ \overline{A_{13}}\ \overline{A_{12}}\ A_{11}$

$CS3 = A_{15}\ A_{14}\ A_{13}\ \overline{A_{12}}$

$CS4 = A_{15}\ A_{14}\ A_{13}\ A_{12}$

$CS5 = A_{15}\ \overline{A_{14}}\ A_{13}\ A_{12}$

a2)

$\overline{CS1} = A_{15} + A_{14} + A_{13} + A_{12} + A_{11}$

$\overline{CS2} = A_{15} + A_{14} + A_{13} + A_{12} + \overline{A_{11}}$

$\overline{CS3} = \overline{A_{15}} + \overline{A_{14}} + \overline{A_{13}} + A_{12}$

$\overline{CS4} = \overline{A_{15}} + \overline{A_{14}} + \overline{A_{13}} + \overline{A_{12}}$

$\overline{CS5} = \overline{A_{15}} + A_{14} + \overline{A_{13}} + \overline{A_{12}}$

b1)

$CS1 = \overline{A_{15}}\ \overline{A_{11}}$

$CS2 = \overline{A_{15}}\ A_{11}$

$CS3 = A_{14}\ \overline{A_{12}}$

$CS4 = A_{14}\ A_{12}$

$CS5 = \overline{A_{14}}\ A_{12}$

b2)

$\overline{CS1} = A_{15} + A_{11}$

$\overline{CS2} = A_{15} + \overline{A_{11}}$

$\overline{CS3} = A_{14} + A_{12}$

$\overline{CS4} = A_{14} + \overline{A_{12}}$

$\overline{CS5} = A_{14} + \overline{A_{12}}$

6.3. La tabla adjunta describe el mapa de memoria de un sistema ordenador. Se pide, justificando cada una de las respuestas,
a) La máxima capacidad de direccionamiento del sistema.
b) Completar la tabla adjunta.
c) Realizar el decodificador de direcciones que genera las señales de selección (activas a nivel bajo) con mapeo incompleto.
d) Indicar las direcciones de memoria en las que podría colocar una nueva pastilla RAM de 128 kBytes para que esté alineada.

Pastilla	Tamaño	Selector	Dirección inicial	Dirección final
ROM	**16 kBytes**	/CS1	10000_{16}	$13FFF_{16}$
RAM SUP	64 kBytes	/CS2	**30000_{16}**	$3FFFF_{16}$
RAM USER 1	256 kBytes	/CS3	80000_{16}	**$BFFFF_{16}$**
RAM USER 2	**128 kBytes**	/CS4	60000_{16}	$7FFFF_{16}$
I/O	**32 kBytes**	/CS5	$F0000_{16}$	$F7FFF_{16}$

SOLUTION:

a) Si las direcciones son de 20 bits, la capacidad máxima de direccionamiento será de: 2^{20} = 1 MByte.

b) Solución en la tabla del enunciado

c)

A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	/CS1	/CS2	/CS3	/CS4	/CS5
0	0	0	1	0	0	0	1	1	1	1
0	0	1	1	X	X	1	0	1	1	1
1	0	X	X	X	X	1	1	0	1	1
0	1	1	X	X	X	1	1	1	0	1
1	1	1	1	0	X	1	1	1	1	0

$\overline{CS1} = A_{19} + A_{17}$

$\overline{CS2} = A_{19} + A_{18} + \overline{A_{17}}$

$\overline{CS3} = \overline{A_{19}} + A_{18}$

$\overline{CS4} = A_{19} + \overline{A_{18}}$

$\overline{CS5} = \overline{A_{19}} + \overline{A_{18}}$

d) Puede ubicarse en dos posiciones:
- **Entre las posiciones 40000_{16} y la $5FFFF_{16}$**
- **Entre las posiciones $C0000_{16}$ y la $DFFFF_{16}$**

6.4. A system has a 20-bits address bus. There are five block in the memory map, as shown in the table.
a) Fill in the table. The block MEM4 occupies the consecutive positions to MEM1. MEM5 can go into any aligned block.
(**Note:** There can be multiple right solutions)
b) Calculate the minimum selection equations of each block.

SOLUTION:

a)

Block	Selection bit	Size	Initial address	Final address
MEM1	/CS1	**128 kBytes**	20000_{16}	$3FFFF_{16}$
MEM2	/CS2	64 kBytes	90000_{16}	**$9FFFF_{16}$**
MEM3	/CS3	32 kBytes	**$E0000_{16}$**	$E7FFF_{16}$
MEM4	/CS4	256 kBytes	**40000_{16}**	**$7FFFF_{16}$**
MEM5	/CS5	128 kBytes	**00000_{16}** **$A0000_{16}$** **$C0000_{16}$**	**$1FFFF_{16}$** **$BFFFF_{16}$** **$DFFFF_{16}$**

b) The minimum selection equations depend on the MEM5 position. Minimum selection equations imply incomplete address.

$\overline{CS1} = A_{19} + A_{18} + \overline{A_{17}}$ $\quad$ $\overline{CS1} = A_{19} + A_{18}$ $\quad$ $\overline{CS1} = A_{19} + A_{18}$

$\overline{CS2} = \overline{A_{19}} + A_{18}$ $\quad$ $\overline{CS2} = \overline{A_{19}} + A_{18} + A_{17}$ $\quad$ $\overline{CS2} = \overline{A_{19}} + A_{18}$

$\overline{CS3} = \overline{A_{19}} + \overline{A_{18}}$ $\quad$ $\overline{CS3} = \overline{A_{19}} + \overline{A_{18}}$ $\quad$ $\overline{CS3} = \overline{A_{19}} + \overline{A_{18}} + \overline{A_{17}}$

$\overline{CS4} = A_{19} + \overline{A_{18}}$ $\quad$ $\overline{CS4} = A_{19} + \overline{A_{18}}$ $\quad$ $\overline{CS4} = A_{19} + \overline{A_{18}}$

$\overline{CS5} = A_{19} + A_{18} + A_{17}$ $\quad$ $\overline{CS5} = \overline{A_{19}} + A_{18} + \overline{A_{17}}$ $\quad$ $\overline{CS5} = \overline{A_{19}} + \overline{A_{18}} + A_{17}$

6.5. Se tiene un micro de 16 bits de direcciones con E/S mapeada en memoria. Se sabe que la E/S necesita 1 kposiciones y que deben ser las direcciones más altas. Aparte hay otras tres zonas de memoria: una Flash de 32 kBytes, una RAM de usuario de 16 kBytes y una RAM de sistema de 8 kBytes. En la medida de lo posible, todas las zonas de memoria deberán estar alineadas. Se pide:

a) Calcular las direcciones del mapa para cada zona de memoria.

b) Realizar el circuito decodificador de direcciones que genere las señales de selección de cada zona de memoria, activas por nivel alto, con mapeo incompleto.

c) Indicar el porcentaje de memoria utilizada.

SOLUTION:

a)

Zona	Tamaño	Selector	Dirección inicial	Dirección final
E/S	1 kBytes	CS1	FC00	FFFF
Sin uso	7 kBytes	--	E000	FBFF
RAM sistema	8 kBytes	CS2	C000	DFFF
RAM Usuario	16 kBytes	CS3	8000	BFFF
Flash	32 kBytes	CS4	0000	7FFF

b)

$$CS1 = A_{15} \cdot A_{14} \cdot A_{13}$$
$$CS2 = A_{15} \cdot A_{14} \cdot \overline{A_{13}}$$
$$CS3 = A_{15}\ \overline{A_{14}}$$
$$CS4 = \overline{A_{15}}$$

c) 57 kBytes utilizados, 7 kBytes sin utilizar, 64 kBytes en total: 89% utilizada

6.6. El sistema de memoria utiliza bloques, todos alineados y ubicados físicamente en tres circuitos (chips) diferentes. El bloque de E/S se ubica en los últimos 4 Mbytes del mapa de memoria, la RAM para los datos es de 64 Mbytes y está ubicada en las primeras direcciones de memoria. Por último, la RAM de código es cuatro veces mayor que la de datos y está ubicada en la primera zona libre alineada por encima de la RAM de datos.

Con los datos facilitados, se pide:

a) Señale, justificando brevemente la respuesta, las direcciones inicial y final para el bloque de E/S.

La dirección final es 0xFFFFFFFF; 4Mbytes = 2^{22} bytes => la dirección final conserva los 10 primeros bits y combina el resto. Por tanto, la dirección inicial 0xFFC00000

b) Señale, justificando brevemente la respuesta, la dirección final para la RAM de datos.

Si la dirección inicial es 0x00000000, 64Mbytes = 2^{26} bytes => la dirección final conserva los 6 primeros bits y combina el resto. **Por tanto la dirección final es 0x03FFFFFF**

c) Señale, justificando brevemente la respuesta, el tamaño y las direcciones inicial y final para la RAM de código.

El tamaño es de 64 Mbytes*4 = 256 Mbytes = 2^{28} bytes, y la primera dirección posible, es 0x10000000. La dirección final conserva los 4 primeros bits y combina el resto, por tanto la dirección inicial 0x1FFFFFFF

d) Suponiendo un mapeo completo señale, justificando brevemente la respuesta, la ecuación de selección para la RAM de código ($\overline{\textbf{CS3}}$), activa en bajo.

Los 4 bits fijos que definen el bloque RAM pedido son A_{31}, A_{30}, A_{29} y A_{28} y deben valer '0, 0, 0 y 1' respectivamente.

Por tanto, la ecuación que selecciona este bloque es $\overline{CS3} = A_{31} + A_{30} + A_{29} + \overline{A_{28}}$

COMPUTER STRUCTURE
UNIT 6.- MEMORY MAPS. REAL NUMBERS OPERATIONS

6.7. Se tiene un sistema basado en un microprocesador con el mapa de direcciones indicado. Se conoce que el micro es capaz de manejar un bus de direcciones de 24 bits.

Componente	Capacidad	Dirección inicial	Dirección final
Periférico 1	1 Mbyte	x000000	**x0FFFFF**
Periférico 2	2 Mbytes	**X600000**	x7FFFFF
Periférico 3	1 kbyte	x400000	**X4003FF**
Memoria RAM	**8 Mbytes**	x800000	xFFFFFF

Se pide:
1. Completar el mapa de direcciones.

COMPUTER STRUCTURE
UNIT 6.- MEMORY MAPS. REAL NUMBERS OPERATIONS

6.8. A MIPS-based system includes a ROM block for the firmware, a RAM block and two blocks for peripherals. The address bus is 32 bits wide. The following table includes partial information about its memory map.

Block	Selection bit	Size	Initial address	Final address
ROM	CS1	1 GByte	0x00000000	0x3FFFFFFF
RAM	$\overline{CS2}$	1 Gbyte	0x80000000	0xBFFFFFFF
Peripheral 1	$\overline{CS3}$	256 Mbytes	0x40000000	0x4FFFFFFF
Peripheral 2	CS4	128 Mbytes	0xC0000000	0xC7FFFFFF

a. Fill in the table with the missing information. The RAM block is 1 GB but can go into any aligned position.

b. Calculate the equations for CS1 and $\overline{CS3}$ using complete mapping.

CS1 = $\overline{A_{31}}$ $\overline{A_{30}}$
$\overline{CS3}$ = A_{31} + $\overline{A_{30}}$ + A_{29} + A_{28}

c. Calculate the simplified equations (incomplete mapping) for $\overline{CS2}$, $\overline{CS3}$ and CS4, which must be compatible with other equations for the rest of the blocks.

Note: For the address bits use the following notation. $A_0 \ldots A_{31}$ where A_0 is the least significant bit and A_{31} is the most significant bit.

$\overline{CS2}$ = $\overline{A_{31}}$ + A_{30}
$\overline{CS3}$ = A_{31} + $\overline{A_{30}}$
CS4 = $A_{31} \cdot A_{30}$

6.9. Se tiene un sistema basado en un microprocesador con el mapa de direcciones indicado. Se conoce que el micro es capaz de manejar un bus de direcciones de 18 bits.

Componente	Señal de selección	Capacidad	Dirección inicial	Dirección final
Memoria 1	CS1	64 kbytes	**0x20000**	**0x2FFFF**
Memoria 2	CS2	128 kbytes	**0x00000**	**0x1FFFF**
Periféricos	CS3	32 kbytes	**0x30000**	**0x37FFF**

Se sabe que el componente “memoria 1” incluye entre sus direcciones válidas la dirección 0x2BEBE.

También se sabe que todos los bloques son alineados y contiguos. Con esta información se pide:

1. Completar la tabla del mapa de direcciones, poniendo las direcciones solicitadas en hexadecimal.

2. Indique las ecuaciones óptimas de las señales de selección (CS1, CS2 y CS3) activas por nivel alto sabiendo que se realiza mapeo incompleto. Para ello, nombre a los bits de dirección A_0, A_1, A_2…, etc, siendo A_0 el bit menos significativo.

El bloque “memoria 1” mueve 16 bits (4 nibbles), por lo que al ser alineado los únicos bits fijos son los 2 bits correspondientes al primer nibble. Por tanto ocupa las posiciones 0x2XXXX.

El bloque “memoria 2” ocupa la mitad de la memoria, así que para estar alineado sus únicas posibilidades sería de la 0x00000 a la 0x1FFFF, o de la 0x20000 a la 0x3FFFF. La segunda opción no puede ser por estar ocupada por el bloque anterior.

En cuanto al bloque “periféricos”, al ser contiguo a los bloques existentes tiene que empezar en 0x30000 (todas las direcciones más bajas a esa ya están ocupadas).

2. Sólo se puede minimizar, respecto al mapeo completo, la señal de selección CS3. Las otras quedan igual con mapeo completo o incompleto. La solución es:

$CS1 = A_{17} \cdot \overline{A_{16}}$ **$CS2 = \overline{A_{17}}$** **$CS3 = A_{17} \cdot A_{16}$**

6.10. **a)** Escribir utilizando 4 dígitos hexadecimales y siguiendo la notación en complemento a 2 para 16 bits los siguientes números con signo dados en decimal + 367_{10}, - 512_{10}, + 457_{10} y - 331_{10}
b) Utilizando notación binaria en coma fija (8 bits para la parte entera y 4 para la fraccionaria), escribir el valor más próximo para representar los números decimales +$137{,}34_{10}$ y $235{,}63_{10}$
c) Utilizando notación binaria en coma flotante con el estándar IEEE-754, escribir el valor más próximo para representar los números decimales: - $234{,}625_{10}$, + $347{,}3125_{10}$, - $518{,}75_{10} \times 10^2$, - $325{,}120_{10} \times 10^3$ y - $145{,}3125_{10}$
d) Los números dados utilizan la notación binaria en coma flotante con el estándar IEEE-754, señalar cuáles son los valores en decimal que estos números representan:
$01000011101011001000000000000000_2$, $3DE15000_{16}$ y $443BA800_{16}$

SOLUTION:

a) + $367_{10} = 016F_{16}$

- $512_{10} = FE00_{16}$

+ $457_{10} = 01C9_{16}$

- $331_{10} = FEB5_{16}$

b) $137{,}34_{10} = 10001001, 0101_2$

$235{,}63_{10} = 1110\ 1011{,}1010_2$

c) - $234{,}625_{10} = 1\ 1000011\ 01101010101000000000000_2 = C36AA000_{16}$

+ $347{,}3125_{10} = 0\ 1000011\ 10101101101010000000000_2 = 43ADA800_{16}$

- $518{,}75_{10} \times 10^2 = 1\ 10001110\ 10010101010001100000_2 = C74AA300_{16}$

- $325{,}120_{10} \times 10^3 = 1\ 1001000\ 10011110110000000000000 = C89EC000_{16}$

- $145{,}3125_{10} = 1\ 10000110\ 00100010101000000000000_2 = C3115000_{16}$

d) $01000011101011001000000000000000_2 = 345{,}00_{10}$

$3DE15000_{16} = 0\ 01111011\ 110\ 0001\ 0101000000000000 = 0{,}1100$

$443BA800_{16} = 0\ 10001000\ 01110111010100000000000 = 750{,}625_{10}$

6.11. Given the decimal number N = - 954,625, write it in hexadecimal using the following formats:
a. Fixed point with sign/magnitude with 16 bits in total, using 4 of them for the fractional part:

N = - $954{,}625_{10} = 1011\ 1011\ 1010.1010_2$ => **$N = BBA,A_{16}$**

b. Fixed point in two's complement with 16 bits in total, using 4 of them for the fractional part:

N = - $954{,}625_{10} = 1100\ 0100\ 0101.0110_2$ => **$N = C45,6_{16}$**

c. IEEE-754 single precision (32 bits):

Sign: negative, 1

$001110111010.1010 = 1.1101110101010 * 2^9$

Biased exponent => 9 + 127 = 136 = 10001000

Mantissa => 1.1101110101010; the '1' before the point is not stored (implicit), so the 23 b mantissa is

=> 110 1110 1010 1000 0000 0000

Final result: **$N = - 954{,}625_{10} = C46EA800_{16}$**

6.12. Two numbers A and B, in hexadecimal, are both fractional and positive numbers. One of them is represented in single precision IEEE-754 while the other is represented in unsigned fixed point. Place the point of the fixed point one as needed for obtaining A = B.

A: $81CA8000_{16}$ **B:** $4501CA80_{16}$

Solution:

As both are positive, the one starting with '1' can not be in IEEE-754. Therefore, A is in unsigned fixed point format.

We analyse B. The exponent is 10001010. Once unbiased (subtracting 127) we get that the exponent is e = 11, so $B \geq 2^{11}$.

There two ways of going on, both valid:

a) Go on analysing B. The mantissa is: 1.000 0001 1100 1010 1000 0000, so the number is that number shifted 11 times to the left: 1000 0001 1100.1010 1000 00001000 = $81C.A8000_{16}$

b) We know $B \geq 2^{11}$. Therefore, the msb in A has to be 2^{11} which means that the integer part goes from 2^{11} to 2^0, 12 bits in total.

Therefore A = $81C.A8000_{16}$

6.13. Se pide el resultado de la resta R = A – B, de los números escritos en hexadecimal, A = 0x803F0000 y B=0xC1E00000. El número A está escrito en notación en complemento a 2, mientras que el número B está escrito en coma flotante según el estándar IEEE-754.

a) Obtenga el número B en decimal.
b) Realice la operación de resta y señale el resultado en hexadecimal con notación en complemento a 2. Indique, justificando la respuesta, si el valor obtenido es un número positivo o negativo.

SOLUTION**:**

a) El valor en decimal del número B = 1 10000011 11000000000000 escrito en CF es:

Signo s = 1, negativo.

Exponente $10000011_2 = 131_{10}$; eliminado el sesgo (127) => Exponente = 131-127 = 4.

Mantisa: 11000000000000; añadiendo el 1 implícito => Mantisa = 1,11000

El número es negativo cuyo valor absoluto es $|B| = 1{,}1100 \times 2^4 = 11100_2$.

En decimal B = - 28_{10}

b) Para restar sumamos al número A de 32 bits, el complemento a 2 de B (substraendo) también con 32 bits. Como B es negativo su C2 es positivo e igual al valor de su valor absoluto.

A = 1000 0000 0011 1111 0000 0000 0000
C2(B) = 0000 0000 0000 0000 0000 0001 1100
A + C2(B) = 1000 0000 0011 1111 0000 0001 1100

R = A – B = $803F01C_{16}$

Es un número negativo puesto que está en C2 y el msb es un '1'

6.14. We have two real numbers X and Y, both in hexadecimal. The first one, X = 0x635A, is in fixed point using two's complement with 4 bits for the fractional part. The second one, Y = 0xC2B8A000, is in floating point using single precision IEEE-754.

a) Convert Y to the same representation used for X.

Convert Y into fixed point using two's complement with 4 bits for the fractional part. First we get the fields of Y according to IEEE-754. Y = 0xC2B8A000 => 1 10000101 01110001010000000000000.

sign (1b) : s = 1. It is a negative number

exponent (8b): exp = 1000 0101 = 133. Removing the 127 bias, the exponent is 6.

mantissa (23b): m = 01110001010000000000000. Adding the implicit '1', M = 1.0111000101.

$| Y | = 1.0111000101 * 2^6 = 1011100.0101$.

As Y is negative, we do the two's complement using 4 bits for the fractional part and 16 bits in total:

| Y | = 0000 0101 1100 .0101=> Y = 1111 1010 0011 .1011 = 0xFA3B

The result is Y=0xFA3B

b) Do the addition Add = X + Y, putting the result in fixed point using two's complement with 4 bits for the fractional part and 16 bits in total. Indicate whether the result is correct or not, justifying your answer.

As both numbers and the result are in the same fixed point format, the addition is straight forward. Add = X+Y

X = 0110 0011 0101 ,1010
\+ Y = 1111 1010 0011 ,1011
Add = 0101 1101 1001 ,0101

The result is **Add=0x5D95 which is correct**. In two's complement the addition of two numbers with different signs can not overflow.

c) Do the subtraction Sub = X - Y, putting the result in fixed point using two's complement with 4 bits for the fractional part and 16 bits in total. Indicate whether the result is correct or not, justifying your answer.

We convert the subtraction to an addition doing the two's complement of the subtrahend.
Y = 1111 1010 0011 .1011 => (- Y) = 0000 0101 1100 .0101

X = 0110 0011 0101 .1010
\+ -(Y) = 0000 0101 1100 .0101
Sub = 0110 1001 0001 .1111

The result is **Sub=0x691F which is correct.** This subtraction does not produce overflow, as we can check seeing that the equivalent addition of two positive numbers lead also to a positive result.

6.15. Se pide multiplicar dos números A y B, dados en hexadecimal y guardados en memoria según el estándar IEEE-754. Se pide el resultado en el mismo estándar y en hexadecimal.
a) $A = 2E3E0000_{16}$ y $B = C2230000_{16}$ **b)** $A = B53F0000_{16}$ y $B = D8430000_{16}$
Nota: se recomienda eliminar los ceros no significativos de la mantisa para facilitar los cálculos y añadir los que sean necesarios en el resultado final.

SOLUTION:

a) A = 0 (s) 010 1110 0 (e) 011 1110 0000 0000 0000 0000 (m)
B = 1 (s) 100 0010 0 (e) 010 0011 0000 0000 0000 0000 (m)

Para facilitar los cálculos eliminamos los ceros de las mantisas
A = 0 (s) 010 1110 0 (e) 011 1110 (m) B = 1 (s) 100 0010 0 (e) 010 0011 (m)

Signo resultado => XOR de los signos de A y B => s =1. El resultado es negativo.

Exponente resultado => Suma de los exponentes de A y B. Normalizado (restar 127).

e = 01011100 + 10000100 - 01111111 = 01100001

Mantisa => Producto de las mantisas de A y B (7 b) añadiendo en ambas el 1 implícito como bit más significativo (8b). Las mantisas son números **sin signo**, su producto será:

$1,0111110 * 1,0100011 = 10111110*10100011*2^{-14}$

Nota: Las mantisas son siempre números positivos, la operación producto será:

```
      10111110
    x 10100011
0000000010111110
000000010111110
00010111110
010111110
0111100011111010
```

El resultado es 0001,1110 0011 1110 10. En este caso no es necesario normalizar el resultado, por lo tanto el exponente no cambia.

Como mantisa normalizada, nos quedamos con los bits después de la coma, añadiendo ceros hasta 23 cifras.

La mantisa del resultado es m => 111 0001 1111 0100 0000

El resultado final es:

$$2E3E0000_{16} * C2230000_{16} = 10110000111100011111010000000000 = B0F1F400_{16}$$

b) A = 1 (s) 011 0101 0 (e) 011 1111 0000 0000 0000 0000 (m)
B = 1 (s) 101 1000 0 (e) 100 0011 0000 0000 0000 0000 (m)

Para facilitar los cálculos eliminamos los ceros de las mantisas
A = 1 (s) 011 0101 0 (e) 011 1111 (m)
B = 1 (s) 101 1000 0 (e) 100 0011 (m)

Signo del resultado => XOR de los signos de A y B. s => $1 \oplus 1 = 0$ => El resultado es POSITIVO.

Exponente resultado => Suma de los exponentes de A y B. Normalizado (restar 127).

e => 01101010 + 10110000 - 01111111 = 1001 1011

Mantisa => Producto de las mantisas de A y B (7 b) añadiendo en ambas el 1 implícito como bit más significativo (8b).

$1,0111110 * 1,0100011 = 10111111*11000011 *2^{-14}$

```
      10111111
    x 11000011
0000000010111111
000000010111111
0000111111
010111111
1001000101111101
```

El resultado es $10,01000101111101 = 1,001000101111101 \times 2^1$

Es necesario normalizar el resultado, por lo tanto al exponente obtenido, se le debe sumar +1.

e => 1001 1011 + 1 = 1001 1100

Como mantisa normalizada, nos quedamos con los bits después de la coma, añadiendo ceros hasta 23 cifras.

La mantisa del resultado es m => 001 0001 0111 1101 0000 0000

El resultado final es:

$$P = A * B = B53F0000_{16} * D8430000_{16} = 0100\ 1110\ 0001\ 0001\ 0111\ 1101\ 0000\ 0000 = \mathbf{4E117D00_{16}}$$

6.16. Definimos un estándar propio para representar números en coma flotante, en todo similar al IEEE-754 pero con números de 16 bits y con un tamaño para signo/mantisa/exponente de 1/6/9
Se pide:

a) ¿Cuáles son los números mayor y menor representables dentro de la norma de este formato? Dejad los números indicados en binario (no hace falta dar su valor en decimal).

Mayor: (+) con exponente sesgado máximo de 62 (111110) (111111 se utiliza para valores especiales) y mantisa máxima (111111111) ⇒ 0 111110 111111111

Menor: (-) con exponente sesgado máximo de 62 (111110) y la mayor mantisa (111111111)
⇒ 1 111110 111111111

b) Efectuar la operación de suma (A+B) de los números guardados en el formato dado y mostrar el resultado en este mismo formato (en hexadecimal), donde A = 0x49FC y B = 0xC038

En binario: A = 0 100100 111111100 B = 1 100000 000111000

Se restan exponentes: eA-eB = 100100 – 100000 = 000100

Hay que igualar exponentes al mayor, así que es necesario desplazar mB cuatro lugares a la derecha:

mA = 1,111111100

mB = $1,000111000 \times 2^{-4} = 0,0001000111000$

A+B, en realidad es A-B por los signos:

```
 1,1111111000
-0,0001000111
 1,1110110001
```

Redondeo: La mantisa tiene 10 bits, uno más de los que permite el estándar definido ⇒ hay que eliminar el último bit ⇒ La norma de redondeo supone que si es un "1" el MSB de los bits a eliminar se suma "1" a la mantisa que permanece: 1,111011000 + 1 = 1,111011001

⇒ A + B = $0\ 100100\ 111011001_2 = 0x49D9$

6.17. Se tienen dos números escritos en hexadecimal A = 0x50C00000 en complemento a dos y B = 0x50C00000 en el estándar para número reales de coma flotante IEEE-754. Se pide el resultado de la operación R = A – B, también en hexadecimal y en el estándar para coma flotante señalado. Para calificar el problema es absolutamente necesario escribir el desarrollo del mismo señalando en cada caso de forma breve, los pasos para obtener el resultado pedido.

SOLUTION

Antes de operar ambos números deben estar en el mismo formato. Como se pide el resultado en coma flotante, se pasa A al estándar dado. A es un número positivo cuyo valor en formato para coma flotante sin componer es:

$$A = 0101\ 0000\ 1100\ 0000\ 0000\ 0000\ 0000\ 0000 = 1{,}01000011 * 2^{30}$$

Se descompone B según el estándar:

Signo positivo (1b): 0
Exponente (8b): 10100001 = 161 - 127 = 34
Mantisa completa (24b) = 1,10000000000000000000000
B = 0101 0000 1100 0000 0000 0000 0000 0000 = $1{,}1 * 2^{34}$

Para sumar y/o restar, se comparan los exponentes, se ajustan las mantisas y se opera.

Como Exp(A) - Exp(B) = - 4, se debe desplazar 4 unidades a la derecha la mantisa de A antes de restar. También se sabe que el resultado de la operación R es un número negativo.

(A) 00,000101000011	# La resta se convierte en suma si se #	(A) 00,000101000011 * 2^{34}
- (B) 01,100000000000	# complementa a dos el substraendo #	+ (-B) 10,100000000000 * 2^{34}
		(R) 10,100101000011 * 2^{34}

Como el resultado de operar las mantisas es negativo, antes de componerlo en el formato estándar IEEE-754 hay que calcular el valor absoluto y en su caso modificar el exponente.

$$R = 10{,}100101000011 * 2^{34} \quad => \quad |R| = 1{,}011010111101 * 2^{34}$$

Se compone R según el estándar:

Signo negativo (1b): 1
Exponente (8b): 34 +127 = 161 = 10100001
Mantisa incompleta (23b) = 011 0101 1110 1000 0000 0000

R = 1 10100001 01101011110100000000000 = 0xD0B5E800

R = A - B = 0xD0B5E800

6.18. Se tiene dos números binarios, A = 0xC3024 escrito en coma fija y complemento a dos con cuatro bits para la parte fraccionaria y B = 0xC3024000 escrito en coma flotante en el estándar IEEE-741. Se pide el resultado de la operación R = B – A, en binario y en hexadecimal con el mismo formato del número A.

Nota: Para facilitar los cálculos, elimine todos los bits no significativos que considere oportunos.

SOLUTION:

B (IEEE-741) => 1100 0011 0000 0010 0100 0000 0000 0000

- s (1b) = 1, signo negativo.
- exp (8b) 1000 0110 = 134; exp. sin sesgo 134 - 127 = 7 => exp = +7
- mantisa guardada, m (23b) = 000 0010 0100 0000…..; M = 1,000 0010 0100 0000…

  |B| = $1{,}000\ 0010\ 0100\ 0000 * 2^{7}$ = 1000 0010,0100 0000

COMPUTER STRUCTURE
UNIT 6.- MEMORY MAPS. REAL NUMBERS OPERATIONS

Para operar dos números ambos deben estar en el mismo tamaño y formato.

A (compl. a 2, coma fija 4 bits fract.) => 1100 0011 0000 0010,0100

B (compl. a 2, coma fija 4 bits fract.) => 1111 1111 0111 1101,1100

La operación R = B – A, equivale a R = B + C2(A).

C2(A) => 0011 1100 1111 1101,1100

(B) 1111 1111 0111 1101,1100
\+ C2(A) 0011 1100 1111 1101,1100
(R) 0011 1100 0111 1011,1000

BINARIO:

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

HEXADECIMAL:

R =	**0x3C7B8**

6.19. Las operaciones de este ejercicio se deben realizar en binario y sin el uso de calculadoras. Hacer las operaciones en decimal sólo debe servir para comprobar el resultado obtenido.

a. Se tienen dos números de 8 bits, A = 0x58 y B = 0x8A ambos escritos en hexadecimal en un formato en complemento a 2. Se pide el resultado de la operación producto R = A·B. Realice la operación y escriba el resultado con 16 bits, también en hexadecimal y con formato en complemento a 2.

A: 0 1 0 1 1 0 0 0
x B: 1 0 0 0 1 0 1 0
R: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 1 0 0 0
0 0 0 0 0 0 1 0 1 1 0 0 0
1 1 0 1 0 1 0 0 0
1 1 0 1 0 1 1 1 0 1 1 1 0 0 0 0

R = **0xD770**

b. Dados dos números X = 0xEBF0, escrito en hexadecimal y con un formato en complemento a dos con 16 bits e Y = 0xC5A28000, escrito en hexadecimal y en el estándar IEEE-754 para coma flotante con 32 bits. Se pide, necesariamente razonando la respuesta cuál de los dos números, teniendo en cuenta el signo, es el mayor.

Ambos números son negativos, para compararlos los restamos y dependiendo del signo del resultado, se conocerá la respuesta solicitada. Para operarlos ambos números deben estar en el mismo formato. En la resolución se opta por poner ambos en coma flotante.

X = 1110101111110000 => | X | = 0001010000010000. Si X ≠ |X| => X < 0 => signo negativo, s = 1

| X | = 000101000001000 = 1,010000010000 * 2^{12}. Exp = 12 (sin sesgo) y M = 1,01000001

| X | = 1,01000001*2^{12}

Y= 1 10001011 01000101000000000000

Signo = 1 => Y es negativo.

Exp = 10001011 = 139 => Exp = 12 (sin sesgo)

m = 01000101000000000000. M = 1,01000101

$|Y| = 1{,}01000101 * 2^{12}$

No es necesario completar la resta, a la vista de los resultados ya se puede deducir que X > Y.

Ambos números tienen el mismo exponente (12) y la mantisa de Y es mayor que la de X, es decir | Y | > | X |. Como ambos números son negativos y la comparación debe incluir el signo, se deduce que la respuesta correcta es:

X > Y

6.20. Consider two numbers, A and B, written in hexadecimal using the standard IEEE-754. Calculate
a) $A = 45C80400_{16} + B = 4541C400_{16}$ **b)** $A = 45C80400_{16} - B = 4541C400_{16}$
Note: Remove the tailing zeros of the mantissa before making any operation.

a) A = 0 (s) 100 0101 1 (e) 100 1000 0000 0100 0000 0000 (m)
B = 0 (s) 100 0101 0 (e) 100 0001 1100 0100 0000 0000 (m)

$A = 1{,}1001000000001 \cdot 2^{12}$ $B = 1{,}1000001110001 \cdot 2^{11}$

exp(B)<exp(A), so B should be shifted to align the numbers

A= 1 , 1 0 0 1 0 0 0 0 0 0 0 0 1 0 $\cdot 2^{12}$

+B= 0 , 1 1 0 0 0 0 0 1 1 1 0 0 0 1 $\cdot 2^{12}$

R= 1 0 , 0 1 0 1 0 0 0 1 1 1 0 0 1 1 $\cdot 2^{12}$

The result should be normalized: $1{,}001010001110011 \cdot 2^{13}$. It does not need any rounding. Therefore,

- s (1b) = 0, positive sign.
- exp (8b) 13+127=140 = 1000 1100
- Stored mantissa, m (23b) = 001010001110011
- R = A + B = 01000110000101000111001100000000 = 0x46147300

b)

A= 1 , 1 0 0 1 0 0 0 0 0 0 0 0 1 0 $\cdot 2^{12}$

-B= 0 , 1 1 0 0 0 0 0 1 1 1 0 0 0 1 $\cdot 2^{12}$

R= 0 , 1 1 0 0 1 1 1 0 0 1 0 0 0 1 $\cdot 2^{12}$

The result should be normalized: $1{,}1001110010001 \cdot 2^{11}$. It does not need any rounding. Therefore,

- s (1b) = 0, positive sign.
- exp (8b) 11+127=138 = 1000 1010
- Stored mantissa, m (23b) = 1001110010001
- R = A + B = 010001010100111001000 1 = 0x454E4400