SOLUCIONES COMENTADAS AL EXAMEN DE ESTRUCTURAS DE LOS COMPUTADORES. FEBRERO DE 1.998

1º) Realizar un procedimiento en ensamblador que lea un número de tres cifras decimales como máximo y las convierta en el correspondiente número hexadecimal. Se supone que el número resultado se dejará en la variable *NumHex*.

Nota: El máximo número decimal será 255d, es decir tres cifras decimales, con lo que su valor hexadecimal será el FFh.

SOLUCIÓN:

```
; ********************************************************************************************
;
; PROCEDIMIENTO:      ConvertirAHexadecimal
; OBJETIVOS:          Convierte el valor de cada cada cifra en Numero al correspondiente
;                     valor hexadecimal
;
; Parámetros que pasa:    NINGUNO.
; Parámetros que recibe:  NINGUNO.
;
; ********************************************************************************************
;
ConvertirAHexadecimal PROC NEAR
        PUSH AX             ; Guardamos el contenido de los registros que vamos a emplear.
        PUSH BX
        PUSH CX
        PUSH DX
        MOV NumHex, 0000    ; Ponemos NumHex a cero.
        XOR AX, AX          ; Inicializamos a cero los registros que vamos a usar.
        XOR BX, BX
        XOR CX, CX
        XOR DX, DX
        LEA BX, Numero      ; Ponemos en DS:BX la cadena Numero, se supone que la hemos
        INC BX              ; leído previamente con la función 0Ah de la NT 21h
        MOV CL, [BX]        ; Inicializamos el contador al número de términos leídos menos uno.
        DEC CX
        JZ UnaCifra         ; Si es una cifra decimal es equivalente a hexadecimal.
        JS Final            ; Si no se ha leído nada terminará.
Bucle:
        INC BX
        XOR AX, AX
        MOV AL, [BX]        ; Ponemos en AL la primera cifra del número.
        SUB AL, 30h         ; Le restamos 30h para convertir al valor númerico
        ADD AX, NumHex      ; Sumamos a esta cifra el número que tenemos en NumHex

        PUSH CX
        XOR CX, CX
        MOV CL, 0Ah         ; Mulitplicamos por 10d (0Ah) el valor anterior.
        MUL CX
        POP CX
        MOV NumHex, AX      ; Llevamos el resultado a NumHex
        LOOP Bucle          ; Y repetimos tantas veces como términos menos uno.

        INC BX
        XOR AX, AX
        MOV AL, [BX]        ; Llevamos el último término a AL.
        SUB AL, 30h         ; Convertimos al valor numérico.
        ADD NumHex, AX      ; Sumamos al contenido de NumHex
        JMP Final           ; Hemos terminado.
```

```
UnaCifra:
        XOR AX, AX
        INC BX
        MOV AL, [BX]        ; Accedemos al elemento leído
        SUB AL, 30h         ; Corregimos a su valor numérico.
        MOV NumHex, AX      ; Guardamos el resultado.
Final:
        POP DX          ; Recuperamos el contenido que tenían los registros antes de
        POP CX          ; llamar al procedimiento.
        POP BX
        POP AX
        RET
ConvertirAHexadecimal ENDP
```

2º) Se tiene una máquina con la siguiente representación en coma flotante:

- Exponente 8 bits en complemento a 1
- Mantisa 8 bits en complemento a 1, sin emplear la técnica del bit implícito.

a) Calcular el rango para dicha representación.

b) Si se supone que recibimos el siguiente número en coma flotante, protegido mediante código Hamming, ver si el número llega correctamente o no.

1 0010 0010 1011 0000 0101

c) En caso de que llegue correctamente calcular su valor suponiendo el sistema de representación del enunciado

d) Si en el caso de coma fija la base en lugar de ser binaria fuese base 4, estuviese representada en complemento restringido a la base (equivalente al complemento a 1) y se contara con 8 bits.
d1) ¿Cómo pasarías de base dos a base 4?
d2) ¿Con que dígito o dígitos indicarías el signo?
d3) ¿Cual sería el signo para positivos y negativos?

SOLUCIÓN:

a) Números en coma flotante. Los números reales se representan según el formato $V(x) = M \times 2^e$. Es decir, una mantisa, multiplicada por 2 elevado a un exponente. Nos piden que calculemos el rango de dicha representación sin emplear la técnica del bit implícito.
Rango para el exponente:
Nos indican que el exponente viene expresado en complemento a 1 y que tenemos un total de 8 bits para su representación.
El rango para un exponente en signo-magnitud es:

Positivos: $[\ 0, 2^{n-1}-1\ ] = [\ 0, 2^{8-1}-1\ ] = [\ 0, 2^7-1\ ] = [\ 0, 127\ ]$
Negativos: $[\ -0, -(2^{n-1}-1)\ ] = [\ -0, -(2^{8-1}-1)\ ] = [\ -0, (2^7-1)\ ] = [\ -0, -127\ ]$

Rango para la mantisa:

Nos indican que la mantisa es fraccionaria,está normalizada y no se emplea la técnica del bit implícito. Tenemos 8 bits para representarla, y se emplea complemento a 1.

Los números en complemento a 1 se encontrarán normalizados cuando:

Positivos: ,01 xxx …xx

Negativos: ,10 xxx …xx

En nuestro caso, si tenemos 8 bits, dos indicarán la normalización y seis se pueden poner con las combinaciones de ceros y de unos que deseemos.

Para los positivos la menor y la mayor mantisa son, respectivamente:

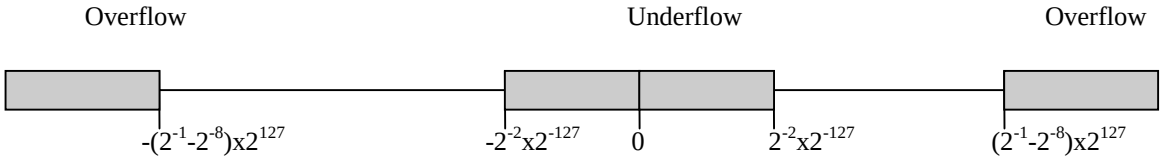
Menor mantisa: $,01\ 000\ 000 = 2^{-2}$

Mayor mantisa: $,0111\ 111 = 2^{-1}-2^{-8}$

Y para los negativos, como el rango es simétrico en complemento a 1 tendre las mantisas:

-2^{-2} y $-(2^{-1}-2^{-8})$

Con lo que la recta real queda de la forma:



b) Si se supone que recibimos el siguiente número en coma flotante, protegido mediante código Hamming, ver si el número llega correctamente o no.

1 0010 0010 1011 0000 0101

Nos indican que el número siguiente está codificado según el código de Hamming y se nos pide que comprobemos si el número ha sido recibido de forma correcta.

En el diagrama inferior se han resaltado en negrita los bits de paridad de Hamming. Que ocupan las posiciones correspondientes a las potencias de dos: 1, 2, 4, 8, 16 … A partir de las cuales se descomponen los demás números tal y como sigue:

1 = paridad

2 = paridad

3 = 2 + 1

4 = paridad

5 = 4 + 1

6 = 4 + 2

7 = 4 + 2 + 1

8 = paridad

9 = 8 + 1

10 = 8 + 2

11 = 8 + 2 + 1

12 = 8 + 4

13 = 8 + 4 + 1

14 = 8 + 4 + 2

15 = 8 + 4 +2 +1

16 = paridad

17 = 16 + 1

18 = 16 + 2

19 = 16 + 2 + 1

20 = 16 + 4

21 = 16 + 4 + 1

1	**0**	0	1	0	0	0	**1**	0	1	0	1	1	0	0	**0**	0	0	1	0	1
$\mathbf{b_1}$	$\mathbf{b_2}$	b_3	$\mathbf{b_4}$	b_5	b_6	b_7	$\mathbf{b_8}$	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}	$\mathbf{b_{16}}$	b_{17}	b_{18}	b_{19}	b_{20}	b_{21}

El $\mathbf{b_1}$ protegerá a todos aquellos valores en cuya descomposición aparezca un 1. El $\mathbf{b_2}$ protege´ra todos aquellos valores en los que se encuentra un 2 en su descomposición y así sucesivamente.

Los bits de paridad (resaltados en negrita en la parte superior) protegen a los bits siguientes:
$\mathbf{b_1} = b_3 \oplus b_5 \oplus b_7 \oplus b_9 \oplus b_{11} \oplus b_{13} \oplus b_{15} \oplus b_{17} \oplus b_{19} \oplus b_{21} = 1$ *correcto.*
$\mathbf{b_2} = b_3 \oplus b_6 \oplus b_7 \oplus b_{10} \oplus b_{11} \oplus b_{14} \oplus b_{15} \oplus b_{18} \oplus b_{19} = 0$ *correcto.*
$\mathbf{b_4} = b_5 \oplus b_6 \oplus b_7 \oplus b_{12} \oplus b_{13} \oplus b_{14} \oplus b_{15} \oplus b_{20} \oplus b_{21} = 1$ *correcto.*
$\mathbf{b_8} = b_9 \oplus b_{10} \oplus b_{11} \oplus b_{12} \oplus b_{13} \oplus b_{14} \oplus b_{15} = 1$ *correcto.*
$\mathbf{b_{16}} = b_{17} \oplus b_{18} \oplus b_{19} \oplus b_{20} \oplus b_{21} = 0$ *correcto.*

Con lo que el número es correcto.

c) En caso de que llegue correctamente calcular su valor suponiendo el sistema de representación del enunciado

Como el número ha llegado de manera correcta debemos calcular su valor. Para ello, lo primero que debemos hacer es eliminar los bits de paridad del código Hamming, los cuales no forman parte del número.

		0		0	0	0		0	1	0	1	1	0	0		0	0	1	0	1
$\mathbf{b_1}$	$\mathbf{b_2}$	b_3	$\mathbf{b_4}$	b_5	b_6	b_7	$\mathbf{b_8}$	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}	$\mathbf{b_{16}}$	b_{17}	b_{18}	b_{19}	b_{20}	b_{21}

De está manera nos quedan únicamente los dígitos que se corresponden con un número en coma flotante de las característcas del enunciado. Es decir, de la forma:

0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	1

Exponente en Signo-Magnitud (8 bits)

Mantisa en Complemento a 1 (8 bits) Sin bit implícito.

Los números reales se representan según el formato $V(x) = M \times 2^e$

Si empezamos por el exponente, veremos que está representado en complemento a 1. Mirando el bit más significativo) veremos que indica que es negativo. Y su valor es 5.
Luego el exponente e = 5.

La mantisa es fraccionaria, y está representada en complemento a 1. Al empezar por ,10 nos indica que se trata de un valor negativo. Para poder calcular su valor, volvemos a complementar el número, con lo que obtenemos el $,01111010 = (2^{-1} - 2^{-5}) + 2^{-7}$.

Juntando todas las piezas obtendré:

$$V(x) = M \times 2^e = -((2^{-1} - 2^{-5}) + 2^{-7}) \times 2^5 = -15{,}25$$

d) Si en el caso de coma fija la base en lugar de ser binaria fuese base 4, estuviese representada en complemento restringido a la base (equivalente al complemento a 1) y se contara con 8 bits.

d1) ¿Cómo pasarías de base dos a base 4?

Juntaría los bits de dos en dos.

d2) ¿Con que dígito o dígitos indicarías el signo?

Como tengo 8 bits, tendré los dígitos 0, 1, 2, 3 en base 4 (en binario 00, 01, 10, 11). Para seguir consecuentemente con los números en complemento a 1, escogería los que empiecen por 0 para los positivos y los que empiecen por uno para los negativos. Es decir, los positivos empezarán por 0 y 1, mientras que los negativos por 2 y 3.

d3) ¿Cual sería el rango para positivos y negativos?

Los positivos irán desde el 0000, hasta el 1333.

Los negativos irán desde el 2000 hasta el 3333.

EXAMEN DE ESTRUCTURAS DE LOS COMPUTADORES.
6 DE FEBRERO DE 1.998. PROBLEMAS (6 PUNTOS)

3º) La CPU de la Figura 1 cuenta con un ancho de palabra de 16 bits. Se quiere dotar a esa CPU de una memoria con las siguientes características:
640 k x 16 de memoria ROM.
384 k x 16 de memoria RAM.
La ROM debe estar después que la RAM.

a) Diseñar la memoria con el menor número de pastillas sabiendo que disponemos de las siguientes pastillas:

Pastillas de memoria RAM	Pastillas de memoria ROM
64 k x 1	32 k x 1
512 k x 8	64 k x 1
128 k x 8	128 k x 8

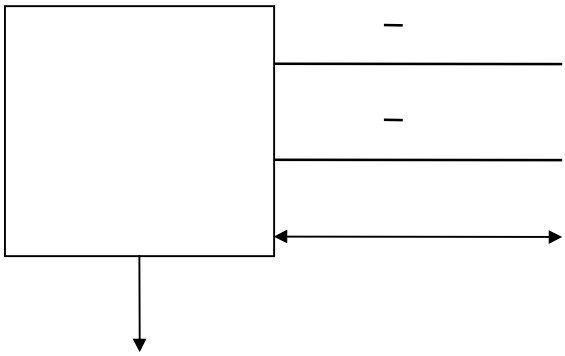


Figura 1

SOLUCIÓN:

1. Comprobar que nos piden algo que realmente se pueda hacer.

Para poder comprobarlo, debemos fijarnos en el número de bits que tenemos en el bus de direcciones, y ver que con ese número de bits, podemos direccionar la memoria que se nos pide.

El bus de direcciones tiene las líneas desde la A_0 hasta la A_{19}, en total 20 bits. Nosotros necesitamos direccionar 640k palabras de memoria RAM y 384k palabras de memoria ROM, luego debemos de ser capaces de direccionar 640k + 384k de memoria total, RAM y ROM.

Para que seamos capaces de poder direccionar 640k + 384k = 1024k, necesitaremos 20 bits (2^n ### 1024k, de donde n = 20). Por lo tanto, vemos que con los 20 bits del bus de direcciones **si** que podemos direccionar la memoria que se nos pide.

Tenemos que comprobar que el valor del bus de datos también sea correcto. Como queremos que cada posición de memoria almacene una palabra (16 bits), necesitaremos que el bus de datos nos soporte el ancho de la palabra.

El bus de datos tiene las líneas desde D_0 hasta D_{15}, en total 16 bits. Por tanto, también podemos acceder a posiciones de memoria en las que se almacenen 16 bits.

EXAMEN DE ESTRUCTURAS DE LOS COMPUTADORES.
6 DE FEBRERO DE 1.998. PROBLEMAS (6 PUNTOS)

2. Calcular el menor número de módulos de memoria que nos harán falta.

Para la memoria RAM.

Disponemos de los siguientes módulos de memoria.

a) 64k x 1
b) 128k x 8
c) 512k x 8

384k x 16 = 6 x 16 = 96 módulos de 64k x 1
64k x 1

384k x 16 = 3 x 2 = 6 módulos de 128k x 8
128k x 8

384k x 16 = 0,75 x 2 ≅ 1 x 2 = 2 módulos de 512k x 8
512k x 8

Para la memoria ROM.

Disponemos de los siguientes módulos de memoria.

a) 32k x 1.
b) 64k x 1.
c) 128k x 8.

640k x 16 = 20 x 16 = 320 módulos de 32k x 1
32k x 1

640k x 16 = 10 x 16 = 160 módulos de 64k x 1
64k x 1

640k x 16 = 5 x 2 = 10 módulos de 128k x 8
128k x 8

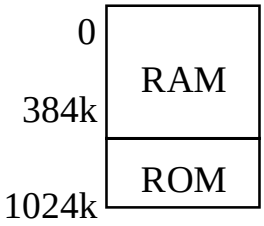
Entonces necesitaremos 2 módulos de 512k x 8 para la memoria RAM y 10 móduloa de 128k x 8 para la memoria R0M. No obstante nos sobran posiciones de la memoria RAM.

EXAMEN DE ESTRUCTURAS DE LOS COMPUTADORES.
6 DE FEBRERO DE 1.998. PROBLEMAS (6 PUNTOS)

3. Diseñar el mapa de memoria.

Para diseñar el mapa de memoria, aunque no nos indican nada, haremos que la ROM esté después que la RAM, tal y como reflejamos en la siguiente figura.

Para direcciónar 512k de RAM necesitaremos 512k = 2^{19} => 19 bits: A_0-A_{18}. Pero yo solamente voy a emplear 384k. Es decir 512k -128k = 384k. Estamos empleando la cuarta parte de la pastilla. Es decir que desde las posiciones 384k-512k no se usan. Con lo que emplearíamos las direciones 000 … 0 a 001 … 1, 010 … 0 a 011 … 1 y 100 … 0 a 101 … 1. Las direciones que empiecen por 11x … x no se usarán.
Para direccionar 128k de ROM necesitaremos 128k = 2^{17} => 17 bits: A_0- A_{16}.

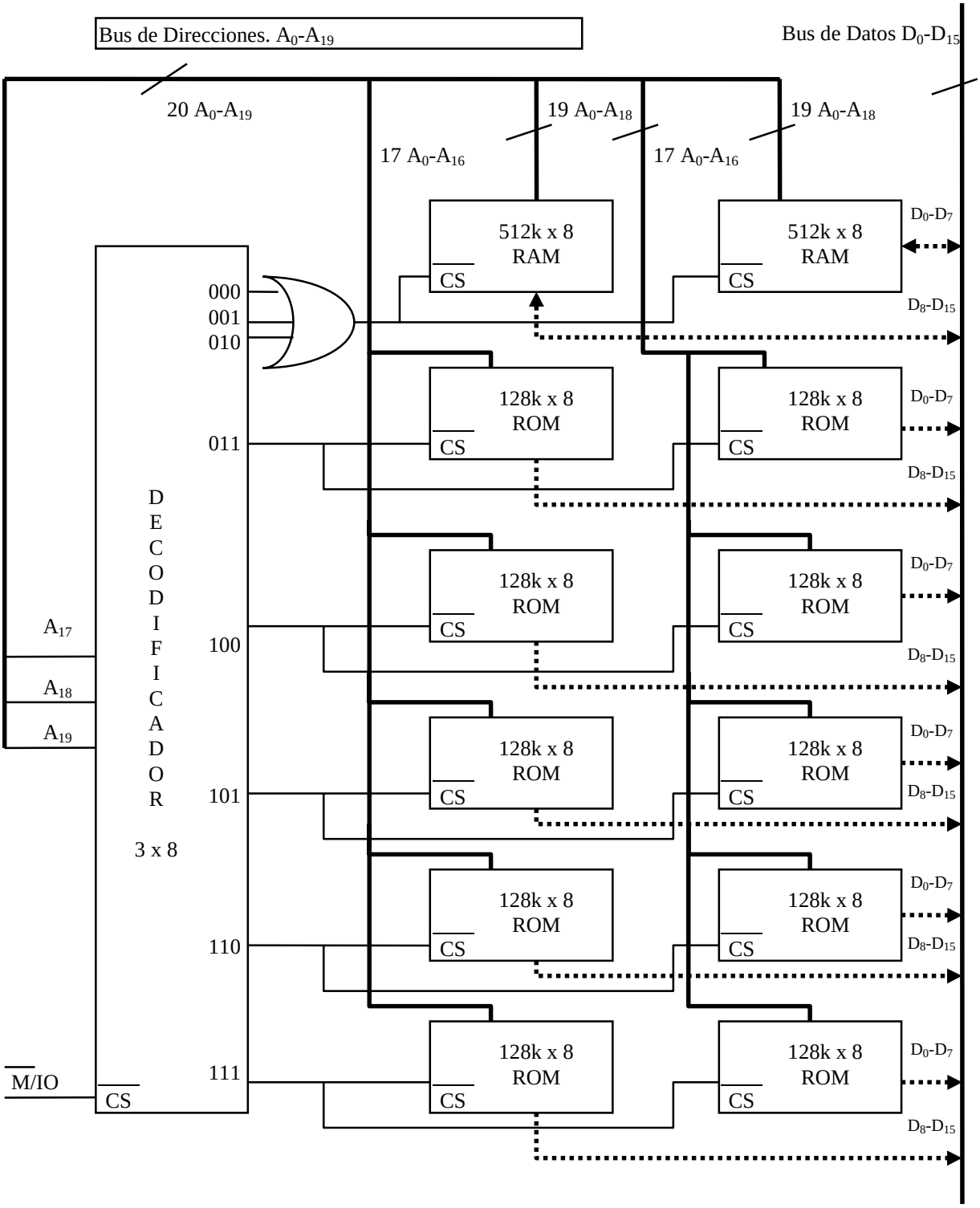


Además, para direccionar la memoria que se nos pide, necesitábamos 20 bits.

A_{19}	A_{18}	A_{17}	A_{16}	A15	.....	A_1	A_0		
0	**0** **1**	**0** **0**	0 1	0 1	.....	0 1	0 1	Mínima dirección. Máxima dirección.	Fila de pastillas: RAM (0k-384k)
0	1	1	0 1	0 1	.....	0 1	0 1	Mínima dirección. Máxima dirección.	1ª fila de pastillas: ROM (384k-512k)
1	0	0	0 1	0 1	.....	0 1	0 1	Mínima dirección. Máxima dirección.	2ª fila de pastillas: ROM (512k-640k)
1	0	1	0 1	0 1	.....	0 1	0 1	Mínima dirección. Máxima dirección.	3ª fila de patillas: ROM (640k-768k)
1	1	0	0 1	0 1	.....	0 1	0 1	Mínima dirección. Máxima dirección.	4ª fila de patillas: ROM (768k-896k)
1	1	1	0 1	0 1	.....	0 1	0 1	Mínima dirección. Máxima dirección.	5ª fila de patillas: ROM (896k-1024k)

EXAMEN DE ESTRUCTURAS DE LOS COMPUTADORES.
6 DE FEBRERO DE 1.998. PROBLEMAS (6 PUNTOS)

4. Dibujar el esquema.



5. Especificarlas líneas que sean necesarias.

Las líneas que debemos escoger son:

- L/E lectura o escritura en las pastillas RAM.