ELECTRÓNICA DIGITAL. **CONTROL 2. Curso 2019-2020**

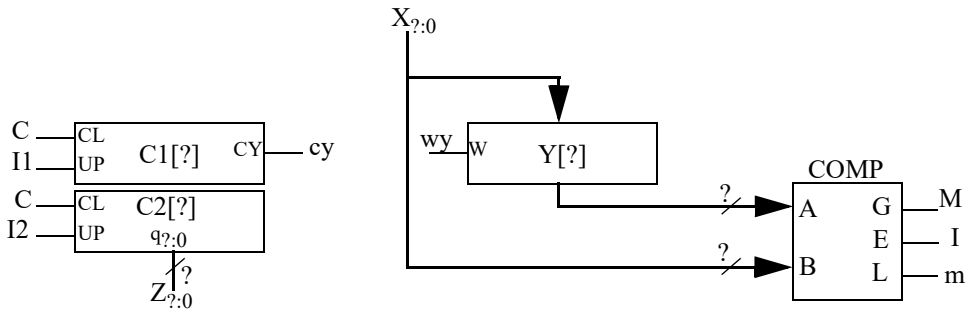
Apellidos:.....**SOLUCIÓN**.....

Nombre:.....

1	2	3	4

Duración 1:00 h.

1.- La UD de la figura corresponde a sistema digital que recibe 32 datos sin signo por un bus de entrada de 16 bits, X. Se desea contar cuántos de ellos son menores que un cierto valor umbral, Y. El sistema mostrará el resultado de la cuenta por sus salidas Z. Su funcionamiento es el siguiente: tras recibir la señal de comienzo Xs, el sistema capturará cada dato cuando se active una señal externa DV (Dato Válido). El primer dato que recibe es el valor umbral Y, seguido por los 32 datos en sí a comparar (cada uno con su correspondiente validación con DV).



a) Dimensione los registros y buses y describa a nivel RT los componentes.

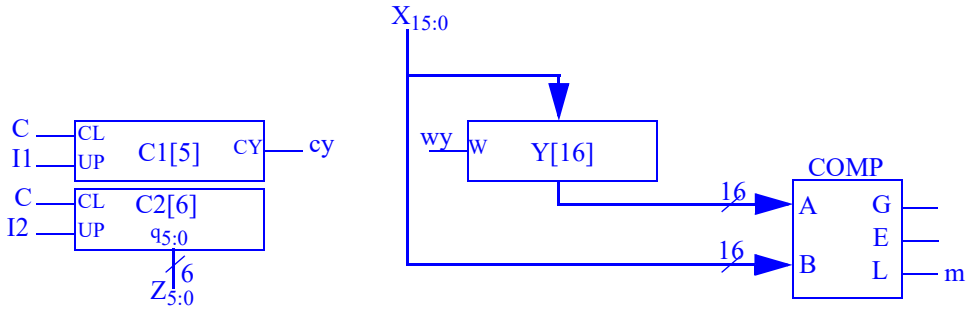
b) Proponga una carta ASM de Datos y de Control.

NOTA: Hacer variaciones con el número de bits de datos, el número de datos y la comparación.

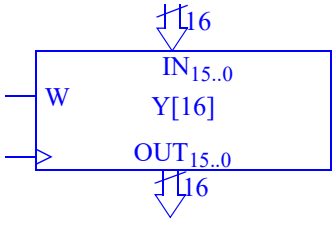
SOLUCIÓN

a) El bus de entrada es de 16 bits, lo que lleva a que el registro Y también lo sea, así como las entradas del comparador. C1 tendrá 5 bits ya que se usará para contar el número de datos que se reciben (32). C2 necesita un bit más ya que se usará para contar el número de datos que satisfacen la condición. Nótese que el rango de dicha magnitud es 33 (6 bits) ya que de los 32 números recibidos, entre 0 y 32 pueden cumplir la condición.

Unidad de datos

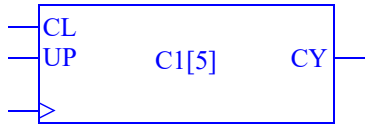


Registro Y:



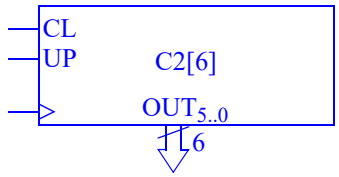
W	Y←	OUT=
0	Y	[Y]
1	IN	[Y]

Contador C1:



CL UP	C1←	CY=
1 - 0 0 0 1	0 C1 C1+1	1 sii [C1]=31

Contador C2:



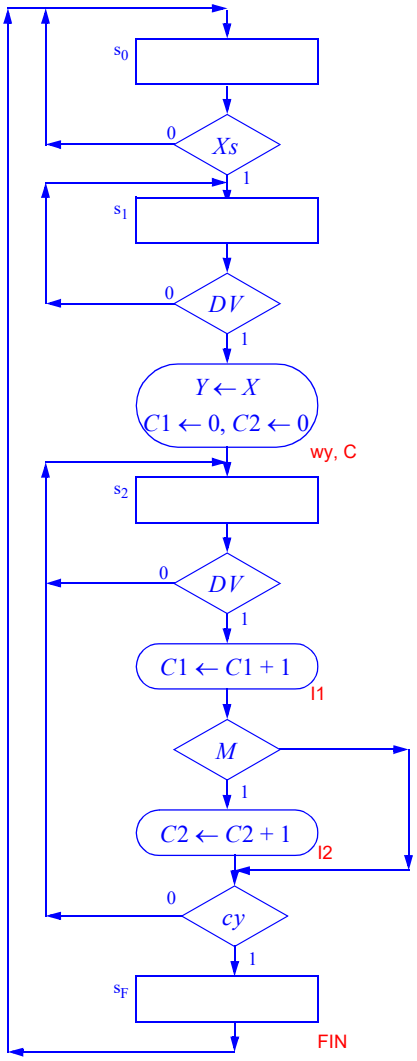
CL UP	C2←	OUT=
1 - 0 0 0 1	0 C2 C2+1	[C2]

Comparador:

G=	E=	L=
1 sii A>B	1 sii A=B	1 sii A<B

b) Carta ASM de la unidad de datos (y de control en rojo). El algoritmo es directo. Después de esperar en s_0 la activación de Xs, el estado s_1 espera la activación de DV. Cuando ocurre, se borran ambos contadores y se carga el dato de entrada X en el registro Y. En el estado s_2 se entra en un bucle doble a la espera de la activación de DV y, cuando ocurre, se incrementa C1 para contar un nuevo dato y, si

es menor que Y, incrementar C2. El bucle se repite mientras C1 no llegue a su valor límite.



CRITERIO CORRECCIÓN

a) Descripción RT: 40% (2 puntos dimensionamiento y 2 puntos cada registro)
b) Cartas ASM: 60% (de datos 80%, de control 20%)

2.- Considere la subrutina siguiente escrita en ensamblador del CS3:

```
;-------------------------------------------------------------------------------
; MulU                                                                  v1.0
;
; Multiplicación algebraica de 2 números de 8 bit sin signo.
; Entradas:
;   R0 y R1 dos números de 8 bits sin signo, A y B.
; Salidas:
;   R1:R0 = AxB resultado de 16 bits sin signo, donde R1 es el MSB y R0 el LSB.
;-------------------------------------------------------------------------------
MulU:       mov       r4, r1     ; 1:
            ldi       r2, 0      ; 2:
            ldi       r3, 8      ; 3:
MuluBuc:    ror       r4         ; 4:
            brcs      MuluSum    ; 5:
            jmp       MuluNoSum  ; 6:
MuluSum:    add       r2, r0     ; 7:
MuluNoSum:  ror       r2         ; 8:
            ror       r1         ; 9:
            sub       r3, 1      ;10:
            brzs      MuluFin    ;11:
            jmp       MuluBuc    ;12:
MuluFin:    mov       r0, r1     ;13:
```

```
            mov     r1, r2      ;14:
            ret                 ;15:
```

a) Analice e indique qué hace cada instrucción. Use comentarios significativos.

b) Obtenga el código máquina en hexadecimal de las dos primeras instrucciones.

c) Suponga que R0 contiene los dos dígitos más significativos de su DNI y que R1 contiene los 2 menos significativos (Ejemplo, si DNI=12,345.678-K, R0=12, R1=78). Rellene el siguiente formulario (en binario) en el proceso de análisis (no olvide considerar el carry en las instrucciones de desplazamiento). Indique el valor de cada registro justo antes de la instrucción 4 (iteración 0) y justo después de la instrucción 10 (iteración i). Añada las líneas que necesite y compruebe que el algoritmo funciona:

Iteración	R4	R3	R2 (antes de 8)	R2	R1
0					
1					
...					

d) Describa qué papel juega cada registro en la subrutina.

e) Escriba un trozo de código en ensamblador que cargue sus A y B personales en R0 y R1 respectivamente y llame a `MulU`.

SOLUCIÓN

a) Se ha comentado cada línea en el propio código

```
MulU:       mov     r4, r1      ;R4 = B
            ldi     r2, 0       ;R2 = acumulador (MSB de resultado)
            ldi     r3, 8       ;R3 es contador de iteraciones
MuluBuc:    ror     r4          ;Obtener bit de B (lsb a msb)
            brcs    MuluSum     ;...es 1. Sumar
            jmp     MuluNoSum   ;...es 0. No sumar. Notar que C=0
MuluSum:    add     r2, r0      ;Suma
MuluNoSum:  ror     r2          ;Desplaza MSB (entra C por la izquierda)
            ror     r1          ;Desplaza LSB (entra el lsb del anterior)
            sub     r3, 1       ;Contabilizar bit procesado
            brzs    MuluFin     ;Terminado? Sí, salir
            jmp     MuluBuc     ;...no. Otra iteración más
MuluFin:    mov     r0, r1      ;Dejar LSB en R0
            mov     r1, r2      ;Dejar MSB en R1
            ret                 ;Volver al llamador
```

b) Código máquina:

mov r4, r1; 01111 100 ----- 001 = 0111 1100 ---- -001 = $7C01

ldi r2, 0; 11111 010 00000000 = 1111 1010 0000 0000 = $FA00

c) Supongamos DNI=12,345.678. R0=12=0000 1100_2. R1=78=0100 1110_2.

Iteración	R4	R3	R2 (antes de ROR)	R2 (después de ROR)	R1
Antes bucle	0100 1110	8	0000 0000	0000 0000	0100 1110
1	0010 0111	7	0000 0000	0000 0000	0010 0111
2	0001 0011	6	0000 1100	0000 0110	0001 0011
3	0000 1001	5	0001 0010	0000 1001	0000 1001
4	0000 0100	4	0001 0101	0000 1010	1000 0100
5	0000 0010	3	0000 1010	0000 0101	0100 0010
6	0000 0001	2	0000 0101	0000 0010	1010 0001
7	0000 0000	1	0000 1110	0000 0111	0101 0000
8	0000 0000	0	0000 0111	0000 0011	1010 1000

El resultado es R1:R0=0000 0011 1010 1000 = 936 = 12 x 78. Funciona.

d) R0: Se mantiene todo el tiempo con el valor A. Sólo al final alberga la parte baja del resultado.

R1: Inicialmente es B, pero va almacenando los bits que se van obteniendo del resultado, que se van introduciendo por la izquierda conforme salen los bits de B por la derecha. Al final alberga la parte baja del resultado.

R2: Se inicializa a 0 como acumulador de la parte alta del resultado, que es el par R2:R1

R3: Contador de bits procesados. Se inicializa a 8 y cuando llega a 0, se termina.

R4: Se inicializa a B y se usa para ir obteniendo los bits de B uno a uno, empezando por el menos significativo.

e)

```
ldi       r0, 12    ;R0=A
ldi       r1, 78    ;R1=B
call      Mulu      ;Multiplicar
```

CRITERIO CORRECCIÓN

a) Comentarios del código: 10% (cada línea, 0'5 comentario obvio, 0'66 buen comentario)
b) Código máquina: 20%
c) Tabla de ejecución: 40% (1 punto por AyB y 1 punto por línea correcta)
d) Función registros: 20%
e) Código llamada: 10%