ELECTRÓNICA DIGITAL. **CONTROL 2. Curso 2023-2024**

Apellidos:.....**SOLUCIÓN**.....

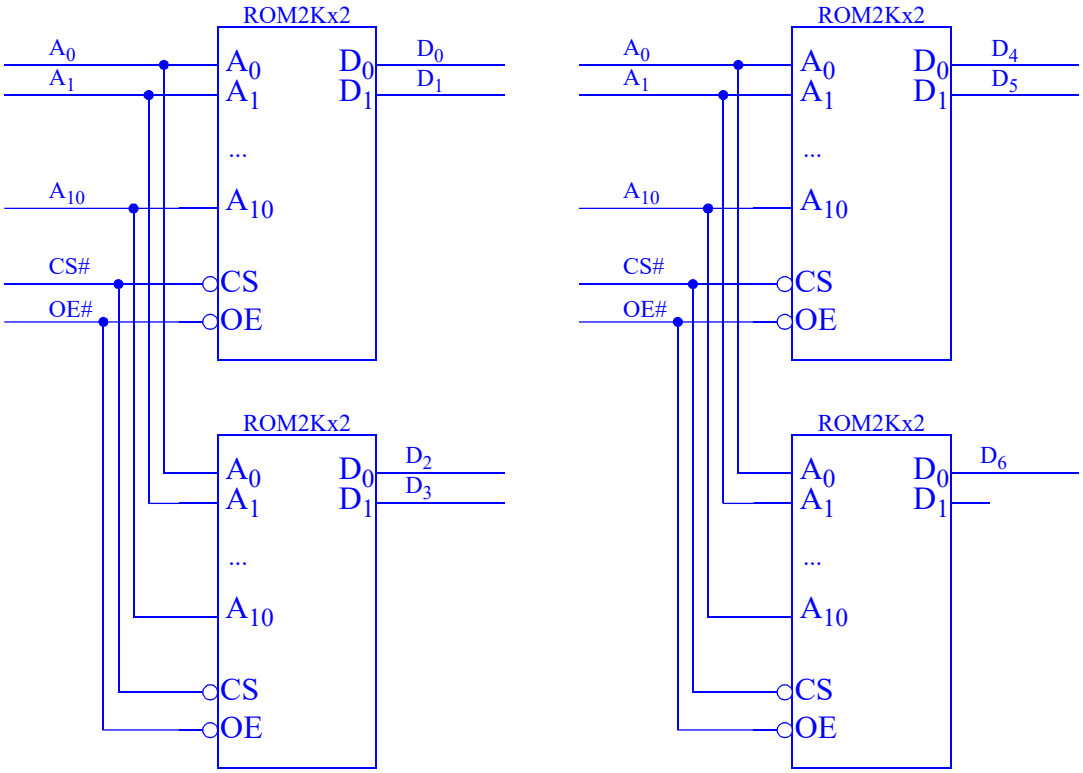
Nombre:.....

1	2	3	4

1.- **[2 puntos]** Jerarquía de memorias de un computador. En qué consiste y para qué sirve.

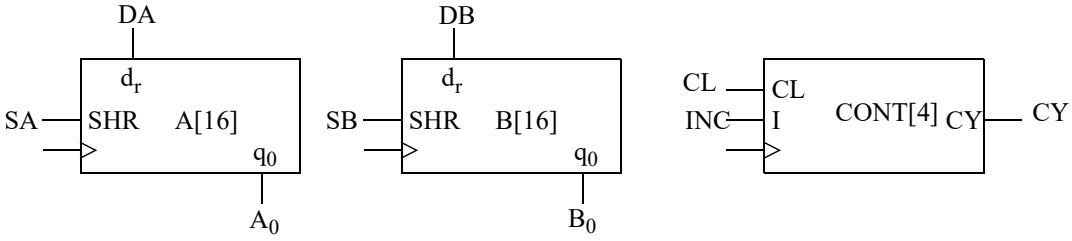
2.- **[1 punto]** Dispone de chips de ROM de 2Kx2. Implemente un módulo ROM de 2Kx7

SOLUCIÓN



CRITERIO DE CORRECCIÓN

3.- **[4 puntos]** Para la unidad de datos de la figura:



a) Describa a nivel RT el registro A. SHR es una señal de desplazamiento a la derecha, d_r es la entrada de datos serie y q_0 es la salida del bit menos significativo.

b) Describa a nivel RT el contador CONT. CL es una señal de puesta a cero síncrona e I es una señal de incremento; CY es salida de carry.

c) Obtenga las cartas ASM de datos y control de una UC que haga la instrucción A ← B/2 cuando I=0 y B ← A*2 cuando I=1.

ELECTRÓNICA DIGITAL. **CONTROL 2. Curso 2023-2024**

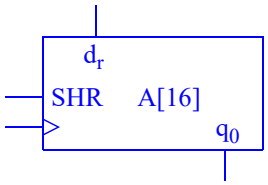
Apellidos:.....**SOLUCIÓN**.....

Nombre:.....

3	4

SOLUCIÓN

a) Registro A:



SHR	A←	q_0=
0	A	$[A_0]$
1	SHR (A, d_r)	$[A_0]$

b) Registro CONT:

CL I	CONT←	CY=
1 -	0	
0 1	CONT+1	1 sii [CONT]=15
0 0	CONT	

c) $\bar{I}$: A ← B/2; I: A ← B*2

Para dividir por 2 un número sólo hay que desplazarlo a la derecha un bit. Dado que los registros A y B tienen dicha capacidad, el principal problema va a ser copiar B/2 a A usando la UD disponible.

La copia se hace mediante un bucle de 16 iteraciones que desplace B a la derecha, lea el bit saliente y lo introduzca en A por la izquierda mediante otro desplazamiento a la derecha.

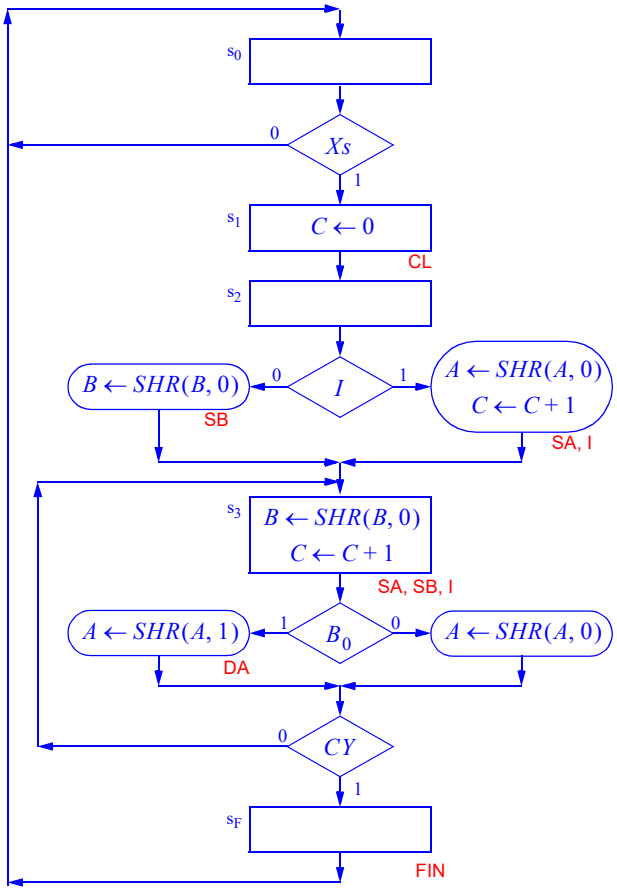
La multiplicación por 2 se hace desplazando a la izquierda el registro, pero puesto que dicha operación no está disponible, se procederá en orden inverso. Primero se desplaza A un bit a la derecha introduciendo un 0 por la izquierda y se incrementa el contador en 1. Después se copiarán los 15 lbs de B en A usando el procedimiento de copia descrito anteriormente.

Resumiendo, se inicializa el contador a 0 y, si I=0, se desplaza B a la derecha completando con 0. Es decir, $B=0B_{15}B_{14}B_{13}B_{12}B_{11}B_{10}B_9B_8B_7B_6B_5B_4B_3B_2B_1$. Después se hace un bucle de 16 iteraciones en el que se desplaza B a la derecha completando con 0 y A completando con B_0 con lo que $A=0B_{15}B_{14}B_{13}B_{12}B_{11}B_{10}B_9B_8B_7B_6B_5B_4B_3B_2B_1$. Si I=1, se desplaza A a la derecha completando con 0 y se incrementa C en una unidad. Después se hace un bucle de 15

Apellidos:.....**SOLUCIÓN**.....

Nombre:.....

iteraciones en el que se desplaza B a la derecha completando con 0 y A completando con B_0 con lo que A=$B_{14}B_{13}B_{12}B_{11}B_{10}B_9B_8B_7B_6B_5B_4B_3B_2B_1B_0$0.



CRITERIO DE CORRECCIÓN

a) 10%

b) 10%

c) 80%

4.- **[3 Puntos]** Se desea introducir en el CS3 la instrucción `CPSE Rd, N`, que compara el registro `Rd` con la constante `N` y salta (esquiva) la siguiente instrucción si son iguales.

a) Proponga un código de operación e indique qué formato de instrucción usará.

b) Indique el código máquina de `CPSE R3, 8`.

c) Indique la secuencia de microoperaciones de esta instrucción (especifique tanto las transferencias a nivel RT como las señales que activaría la UC). Si necesitara modificar la unidad de datos, explique cómo lo haría, describiéndolo adecuadamente.

SOLUCIÓN

a) Se podría usar cualquier código de los que están libres en el CS3, como 9, C, D, E, 10, 11, 16, 19, 1C, 1D y 1E (todos en hexadecimal). Para facilitar la posterior decodificación, convendría que el código fuera similar a alguna de las instrucciones ya implementadas con funcionamiento parecido. En nuestro caso, como es una instrucción que compara y salta, debería parecerse a la de comparación (CP, CoOp B) o salto condicional (BRxx, CoOp 6). Como el 16 está libre, podría ser una buena elección. El formato de instrucción a usar es el B, ya que se necesita albergar un registro y un número de 8 bits en inmediato. El formato es

Apellidos:.....**SOLUCIÓN**.....

Nombre:.....

similar y se usa para las instrucciones de salto, pero no alberga un número en inmediato, sino la condición y dirección del salto.

b) Código máquina: el código de operación es 16 (bits 11 a 15), Rd es 3 (bits 8 a 10) y N es 8 (bits 0 a 7).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	0	0	1	1	0	0	0	0	1	0	0	0

Luego el código resultante es $B308.

c) No es necesario modificar la unidad de datos para implementar esta instrucción. La secuencia de microoperaciones de la fase de ejecución sería:

micro op	Transferencias	Señales
1	$REG[IR_{10..8}]\text{-}IR_{7..0}$	OP_3, OP_1, Ws, INM
2	Z: PC ← PC+1	Z: I_{PC}

Primero se calcula la diferencia entre el registro y la constante que se alberga en los 8 bits menos significativos del IR, pero no se guarda el resultado. Sólo es necesario guardar los flags de salida de la ALU en el SR (Ws=1 y Wac=0). En la segunda microoperación se comprueba el bit Z y en caso de valer 1, se incrementa el PC para saltarse la siguiente instrucción.

Esta instrucción altera el contenido de los bits del SR. Se podría evitar esto realizando un pequeño cambio en la unidad de datos, dando salida al bit Z de la ALU para que sea accesible por al UC (en la actualidad la UC sólo puede acceder al bit Z del SR). Si se hiciera este cambio, la secuencia de microoperaciones se simplificaría, al poder hacerlo todo en un ciclo:

micro op	Transferencias	Señales
1	$REG[IR_{10..8}]\text{-}IR_{7..0}$, Z_{ALU}: PC ← PC+1	OP_3, OP_1, INM, Z_{ALU}: I_{PC}

CRITERIO DE CORRECCIÓN