Examen 2022/23-2

Asignatura	Código	Fecha	Hora inicio
Fundamentos de computadores	75.562	1/7/2023	10:00

Este enunciado también corresponde a las siguientes asignaturas:

- 81.518 - Fundamentos de computadores

Ficha técnica del examen

- No es necesario que escribas tu nombre. Una vez resuelta la prueba final, solo se aceptan documentos en formato .doc, .docx (Word) y .pdf.
- Comprueba que el código y el nombre de la asignatura corresponden a la asignatura de la que te has matriculado.
- Tiempo total: **2 horas** Valor de cada pregunta: **P1:20%; P2:35%; P3:35%; P4:10%**
- ¿Puede consultarse algún material durante el examen? **NO** ¿Qué materiales están permitidos? **Ninguno**
- ¿Puede utilizarse calculadora? **NO** ¿De qué tipo? **NINGUNO**
- Si hay preguntas tipo test, ¿descuentan las respuestas erróneas? **NO** ¿Cuánto?
- Indicaciones específicas para la realización de este examen: **Razonad todas las respuestas. Las respuestas sin justificar no serán puntuadas.**

Examen 2022/23-2

Asignatura	Código	Fecha	Hora inicio
Fundamentos de computadores	75.562	1/7/2023	10:00

Enunciados

PROBLEMA 1 [20%]

a) **[10%]** Dados los números *A* = 0101001 y *B* = 1010011, que representan dos números naturales expresados en binario natural con 7 bits, calculad *A* + *B* usando el mismo formato. Explicad si se produce o no desbordamiento y por qué razón. Indicad cuál es el rango del resultado de esta operación expresándolo en decimal.

Con números naturales podemos hacer la suma directamente. Así pues:

```
          1  1     ← acarreo
    0  1  0  1  0  0  1  ← A
+   1  0  1  0  0  1  1  ← B
-------------------------
    1  1  1  1  1  0  0
```

Como no hay acarreo final, no hay desbordamiento, y el resultado final es: 1111100.

El rango de representación de los números naturales es [0..2^n-1]. El rango de representación del resultado de esta operación es el mismo puesto que podemos obtener estos mismos valores extremos con múltiples combinaciones de los operandos de entrada. Por lo tanto, en este caso que tenemos números de 7 bits (n=7) el rango es [0..127].

b) **[5%]** Dado el número *C* = 1001101 que representa un número natural expresado en binario con 7 bits, indicad su valor en base 16.

Puesto que podemos transformar directamente un número binario natural a un número hexadecimal (base 16), no hace falta obtener su valor decimal. Tan solo tenemos que hacer grupos de cuatro bits de derecha a izquierda y sustituir cada grupo de 4 bits por su valor correspondiente:

```
100    1101   ← binario
 4      D     ← *hex.
```

Así pues, el resultado es $4D_{(16}$.

c) **[5%]** Dado el formato de coma flotante siguiente:

S	Exponente			Mantisa		
12	11		8	7		0

Donde:

- El bit de signo (posición 12), *S*, vale 0 para cantidades positivas y 1 para negativas.
- El exponente se representa en exceso a 8, con 4 bits de la posición 11 a la 8.
- La mantisa está normalizada en la forma 1,*X*. Se representa con bit implícito y con 8 bits de la posición 7 a la 0.

Examen 2022/23-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Fundamentos de computadores	75.562	1/7/2023	10:00

Representad el número -57,3$_{(10}$ en este formato de coma flotante.

Según el formato dado, analizamos primero el número separando la parte entera y la parte fraccionaria:

1. Pasamos la parte entera a binario, aplicando el método de la división entera:

57	=	28 *· 2	+	**1**
28	=	14 *· 2	+	**0**
14	=	7 *· 2	+	**0**
7	=	3 *· 2	+	**1**
3	=	1 *· 2	+	**1**
1	=	0 *· 2	+	**1**

$57_{(10}$ = $111001_{(2}$

2. Para la parte fraccionaria aplicamos el método correspondiente. Teniendo en cuenta que es con bit explícito, con truncamiento y que en la parte entera ya tenemos 6 bits, hay que calcular 3 bits para la parte fraccionaria :

0,3	*· 2 =	0,6	+	**0**
0,6	*· 2 =	0,2	+	1
0,2	*· 2 =	0,4	+	**0**

$0,3_{(10}$ ≈ $0,010_{(2}$

Juntamos la parte entera y la fraccionaria: $111001,010_{(2}$

Para normalizar la mantisa hay que mover la coma 5 posiciones hacia la izquierda:

$$111001,010_{(2} = 1,11001010_{(2} *\cdot 2^5$$

Finalmente, identificamos cada campo:

- Signo: negativo, *S*=1.
- Exponente: 5. Hay que representarlo en exceso a 8. Por lo tanto, tenemos que sumarle el exceso, 5 + 8 = 13, que en base 2 es $1101_{(2}$.
- Mantisa: $1,11001010_{(2}$. Como que lo tenemos que representar con bit implícito eliminamos el 1 de la parte entera. Así pues, la mantisa será 11001010. Como la mantisa tiene disponibles 8 posiciones, y ya hemos truncado la parte fraccionario teniendo en cuenta el número total de bits, no hace falta añadir ningún 0.

El número en el formato solicitado es:

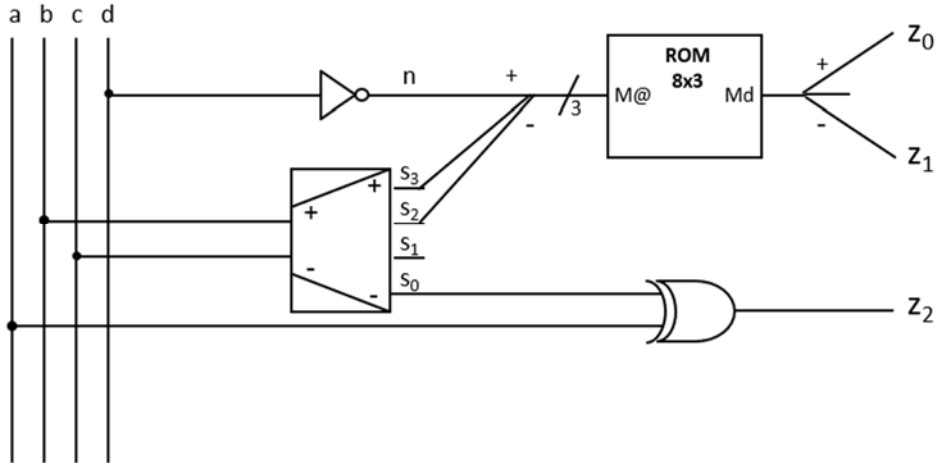
1	1101	11001010

Examen 2022/23-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Fundamentos de computadores	75.562	1/7/2023	10:00

PROBLEMA 2 [35%]

a) **[10%]** Dado el circuito lógico combinacional siguiente:



Donde el contenido de la memoria ROM, especificado en hexadecimal, es:

M@	[Md]
0	4
1	6
2	3
3	1
4	7
5	0
6	5
7	2

Completad la tabla de verdad que especifica las salidas z_0, z_1, z_2 en función de las entradas *a, b*, *c* y *d*. Calculad previamente los valores intermedios (incluyendo la dirección de entrada en la ROM, M@, en decimal) indicados en el circuito y añadidos a la tabla de verdad siguiente:

a	b	c	d	n	s_3	s_2	s_1	s_0	z_0	z_1	z_2
0	0	0	0								
0	0	0	1								
0	0	1	0								
0	0	1	1								
0	1	0	0								
0	1	0	1								
0	1	1	0								
0	1	1	1								
1	0	0	0								
1	0	0	1								
1	0	1	0								
1	0	1	1								
1	1	0	0								
1	1	0	1								

Examen 2022/23-2

Asignatura	Código	Fecha	Hora inicio
Fundamentos de computadores	75.562	1/7/2023	10:00

1	1	1	0		
1	1	1	1		

En primer lugar, para completar las columnas de los valores intermedios, buscamos sus funciones lógicas:

- n = d'
- s_3 = bc
- s_2 = bc'
- s_1 = b'c
- s_0 = b'c'
- z_2 = s_0 (xor) a

Y evaluamos estas funciones para cada fila de la tabla de verdad:

a	b	c	d	n	s_3	s_2	s_1	s_0	z_0	z_1	z_2
0	0	0	0	1	0	0	0	1			1
0	0	0	1	0	0	0	0	1			1
0	0	1	0	1	0	0	1	0			0
0	0	1	1	0	0	0	1	0			0
0	1	0	0	1	0	1	0	0			0
0	1	0	1	0	0	1	0	0			0
0	1	1	0	1	1	0	0	0			0
0	1	1	1	0	1	0	0	0			0
1	0	0	0	1	0	0	0	1			0
1	0	0	1	0	0	0	0	1			0
1	0	1	0	1	0	0	1	0			1
1	0	1	1	0	0	0	1	0			1
1	1	0	0	1	0	1	0	0			1
1	1	0	1	0	0	1	0	0			1
1	1	1	0	1	1	0	0	0			1
1	1	1	1	0	1	0	0	0			1

Finalmente, para evaluar las salidas z0 y z1 debemos tener en cuenta el contenido de la ROM y sus entradas. De esta forma, podemos expresar la tabla del contenido de la ROM en función de las conexiones de este circuito, es decir:

M@ →	n s_3 s_2	[Md] →	z_0 – z_1
0	0 0 0	4	1 ~~0~~ 0
1	0 0 1	6	1 ~~1~~ 0
2	0 1 0	3	0 ~~1~~ 1
3	0 1 1	1	0 ~~0~~ 1
4	1 0 0	7	1 ~~1~~ 1
5	1 0 1	0	0 ~~0~~ 0
6	1 1 0	5	1 ~~0~~ 1
7	1 1 1	2	0 ~~1~~ 0

Y ahora ya podemos rellenar las columnas que faltan de la tabla de verdad utilizando esta última tabla, tomando como referencia para cada fila los valores de las columnas n, s_3 y s_2:

Examen 2022/23-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Fundamentos de computadores	75.562	1/7/2023	10:00

a	b	c	d	n	s_3	s_2	s_1	s_0	z_0	z_1	z_2
0	0	0	0	1	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0	1	1	0	1
0	0	1	0	1	0	0	1	0	1	1	0
0	0	1	1	0	0	0	1	0	1	0	0
0	1	0	0	1	0	1	0	0	0	0	0
0	1	0	1	0	0	1	0	0	1	0	0
0	1	1	0	1	1	0	0	0	1	1	0
0	1	1	1	0	1	0	0	0	0	1	0
1	0	0	0	1	0	0	0	1	1	1	0
1	0	0	1	0	0	0	0	1	1	0	0
1	0	1	0	1	0	0	1	0	1	1	1
1	0	1	1	0	0	0	1	0	1	0	1
1	1	0	0	1	0	1	0	0	0	0	1
1	1	0	1	0	0	1	0	0	1	0	1
1	1	1	0	1	1	0	0	0	1	1	1
1	1	1	1	0	1	0	0	0	0	1	1

b) **[10%]** Dada la tabla de verdad siguiente:

a	b	c	d	f
0	0	0	0	X
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	X
0	1	0	1	0
0	1	1	0	X
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	X
1	1	1	1	X

Sintetizad de manera mínima a dos niveles la función *f* mediante el método de Karnaugh. No hace falta que implementéis el circuito.

El mapa de Karnaugh de la función *f* es el siguiente:

Examen 2022/23-2

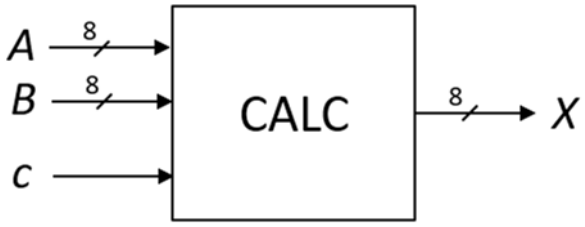
Asignatura	Código	Fecha	Hora inicio
Fundamentos de computadores	75.562	1/7/2023	10:00

cd \ ab	00	01	11	10
00	X	X	1	0
01	0	0	0	0
11	0	1	X	1
10	1	X	X	0

Y obtenemos esta expresión mínima:

f = bd' + bc + acd + a'd'

c) **[15%]** Se quiere diseñar un circuito combinacional llamado *CALC* con la estructura siguiente:

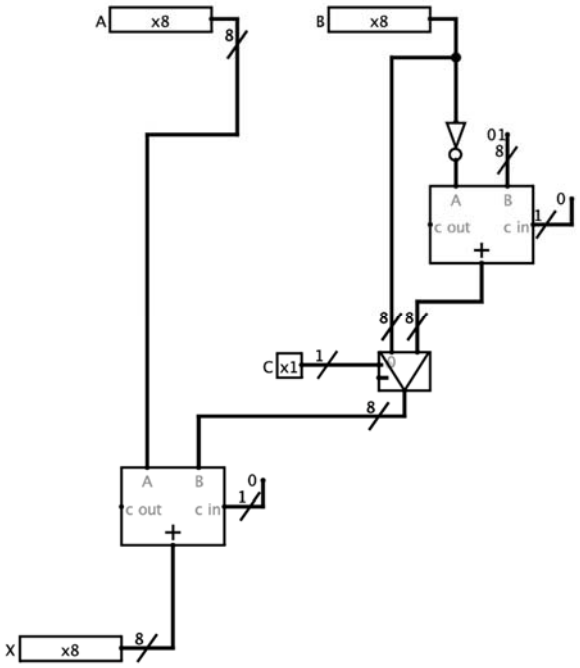


Las entradas *A y B* representan números enteros de 8 bits codificados en Ca2, y la entrada *c* es una señal de un bit. La salida *X* también es de 8 bits y representa igualmente un valor en Ca2. Cuando *c* vale 0 la salida *X* tiene que valer la suma de *A más B*, es decir X = (*A* + *B*), y cuando *c* vale 1 la salida *X* tiene que valer la resta de *A menos B, es decir, X* = (*A* – *B*).
Diseñad el circuito *CALC* usando bloques y puertas combinacionales, y especificando claramente la dimensión de todos los buses utilizados. No hay que controlar si hay desbordamiento.

Las dos operaciones que se tienen que implementar se pueden redefinir como A+B en el caso de que C=0 y A+(-B) en el caso de que C=1. Es decir, el circuito se puede implementar como un sumador cuyo primer operando es la entrada A y que el segundo operando se seleccione en función del valor de C. Si C=0 se tiene que seleccionar directamente el valor de entrada B. Si C=1, primero tenemos que cambiar de signo la entrada B, es decir, complementar los bits de B y sumarle 1. Para seleccionar uno o el otro caso, utilizaremos un MUX con un bit de selección donde conectaremos la entrada C.

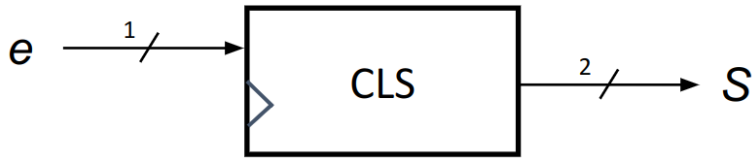
Examen 2022/23-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Fundamentos de computadores	75.562	1/7/2023	10:00



PROBLEMA 3 [35%]

a) **[17.5%]** Se quiere diseñar el grafo de estados de un circuito lógico secuencial (CLS) que se comporte como un detector de secuencias **con solapamiento** y que tenga las entradas y salidas que se muestran a continuación:



El circuito tendrá que detectar si a la entrada *e* (de un bit) se ha dado la secuencia 01x1, donde el bit *x* puede ser tanto 0 como 1. Al detectar la secuencia, si el bit x era 0, entonces la señal de salida *S* tomará el valor *S* = 01 durante un ciclo de reloj. Si, por el contrario, el bit *x* era 1, entonces la señal de salida *S* tomará el valor *S* = 10. En cualquier otra circunstancia, la señal de salida *S* tomará el valor *S* = 00.

A continuación, se muestra un ejemplo del funcionamiento del circuito, en que cada columna corresponde a un ciclo de reloj. Con gris se destaca las secuencias reconocidas en la entrada *e* y los valores para la salida *S* distintos de 00, y con resaltado negro los valores que representan el bit *x* en las secuencias 01x1:

Examen 2022/23-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Fundamentos de computadores	75.562	1/7/2023	10:00

entrada *e*	1	**0**	**1**	**0**	**1**	**1**	**1**	0	1	0	0	1	0
salida s_1	0	0	0	0	0	**0**	0	**1**	0	0	0	0	0
salida s_0	0	0	0	0	0	**1**	0	**0**	0	0	0	0	0

Diseñad el grafo de estados del circuito CLS, especificando claramente el significado de cada estado.

Para conseguir el funcionamiento deseado, el circuito debe tener los siguientes estados:

Estado	Descripción	Salida
INI	Estado inicial.	00
Z	El valor de la entrada *e* era 0.	00
ZU	Se ha detectado la secuencia 01 en la entrada *e*	00
ZUU	Se ha detectado la secuencia 011 en la entrada *e*	00
ZUZ	Se ha detectado la secuencia 010 en la entrada *e*	00
ZUUU	Se ha detectado la secuencia 0111 en la entrada *e*	10
ZUZU	Se ha detectado la secuencia 0101 en la entrada *e*	01

Partiendo del estado inicial *INI*, haremos una transición al estado *Z* solo si la entrada *e* es 0, que es el primer valor de la secuencia a detectar (01x1). Una vez en *Z*, haremos una transición hacia el estado *ZU* solo si la entrada es uno, que quiere decir que se ha detectado la secuencia 01. En caso contrario, nos mantenemos en el estado *Z*. A partir de este estado (*ZU*) haremos una transición hacia *ZUU* o *ZUZ*, en función del valor de la entrada *e*, y que implica tener en cuenta el valor x de la secuencia (01x1).

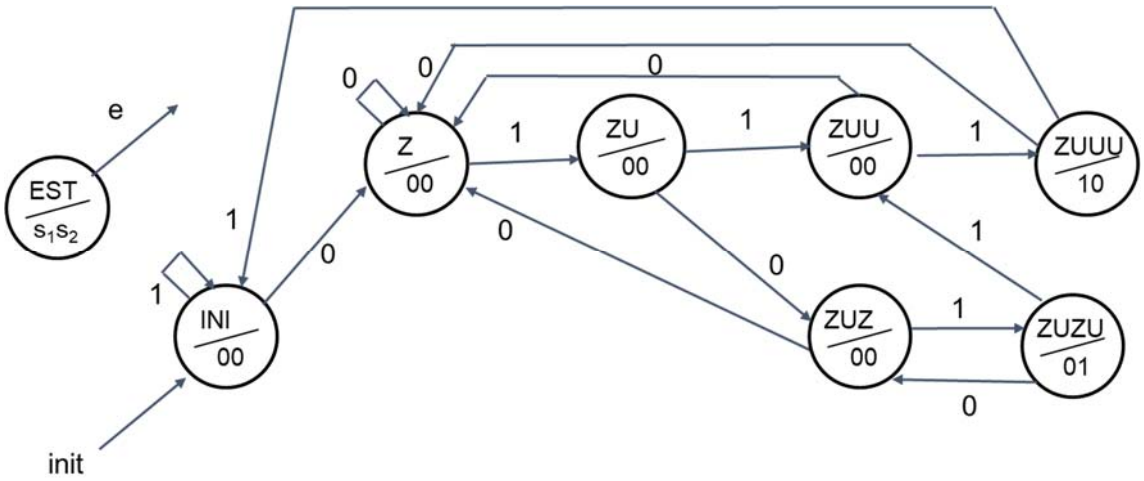
En el caso de realizar la transición hacia *ZUU,* la secuencia detectada hasta el momento es 011. Si el siguiente valor de *e* es 0, la secuencia detectada es 0110, que no cumple con la secuencia esperada (01x1). En este caso se evoluciona hacia el estado Z. Si el siguiente valor de *e* es 1, la secuencia detectada es 0111, que si cumple con la esperada. En este caso, se hace una transición hacia el estado *ZUUU*, donde el valor de la salida es 10. Una vez en el estado *ZUUU*, reiniciamos la detección de la secuencia, haciendo una transición hacia *INI* si el valor de *e* es 1 o hacia *Z* si el valor de *e* es cero.

De una forma similar a como hemos hecho a partir del estado *ZUU*, se puede realizar un razonamiento parecido a partir del estado *ZUZ,* donde la secuencia detectada hasta el momento es 010. Si el siguiente valor de *e* es 0, la secuencia detectada es 0100, que no cumple con la secuencia esperada (01X1). En este caso se evoluciona hacia el estado *Z* ya que hay detección con solapamiento. En cambio, si el siguiente valor de *e* es 1, la secuencia detectada es 0101, que si cumple con la secuencia esperada. En este caso, se hace una transición hacia el estado *ZUZU*, donde el valor de la salida es 01. Una vez en el estado *ZUZU*, hay que tener en cuenta que una nueva detección con solapamiento. Observamos como los dos últimos valores de la secuencia 0101 son 01, por tanto, en este caso haremos una transición hacia ZUU si el siguiente valor de *e* es 1 y hacia el estado ZUZ si *e* es 0.

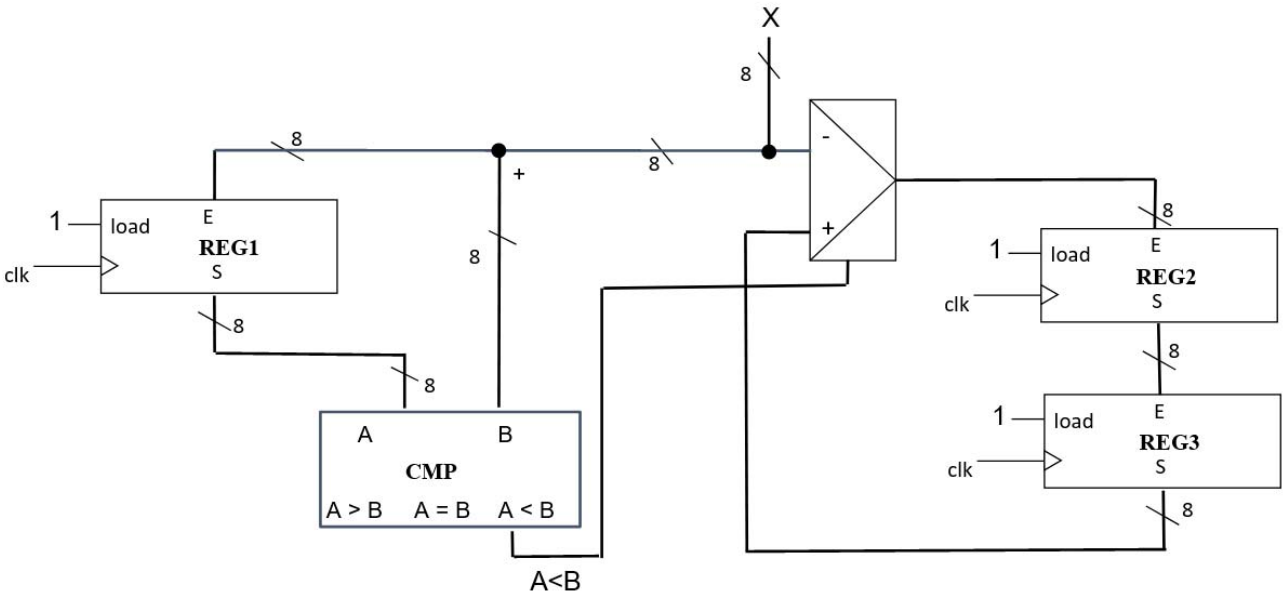
El siguiente grafo representa el comportamiento deseado:

Examen 2022/23-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Fundamentos de computadores	75.562	1/7/2023	10:00



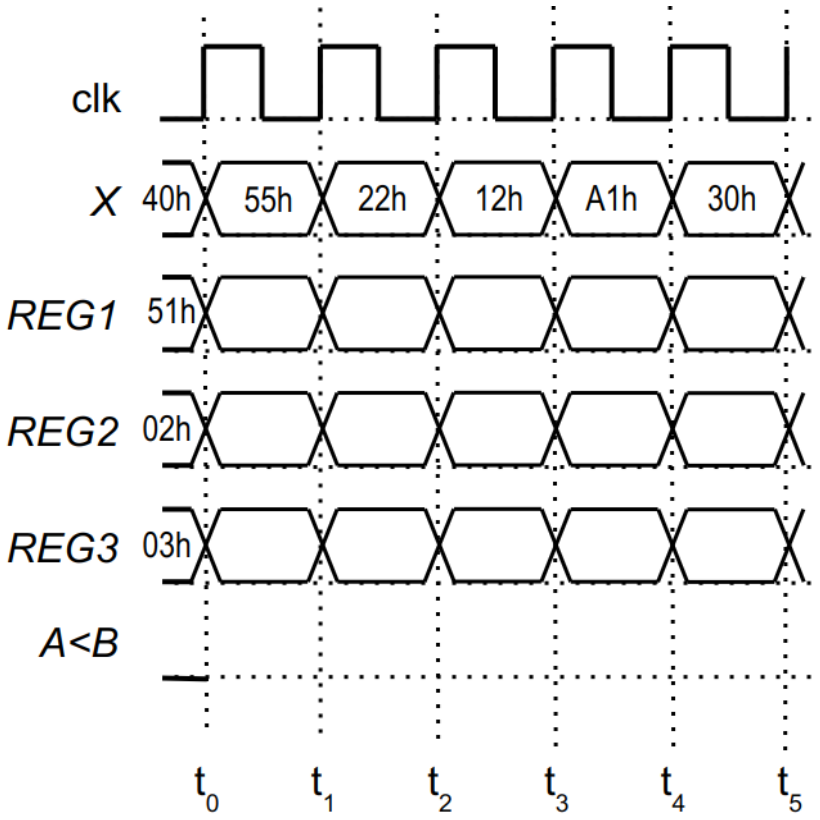
b) **[17.5%]** Dado el circuito secuencial siguiente:



Completad el cronograma siguiente. No hace falta que justifiquéis la respuesta.

Examen 2022/23-2

Asignatura	**Código**	**Fecha**	**Hora inicio**
Fundamentos de computadores	75.562	1/7/2023	10:00



Para rellenar el cronograma miramos en qué momentos se cargan los registros y con qué valores. En cuanto a los momentos, vemos que se cargan en cada flanco ascendente de reloj, ya que en la entrada *load* de los tres registros hay conectado un 1.

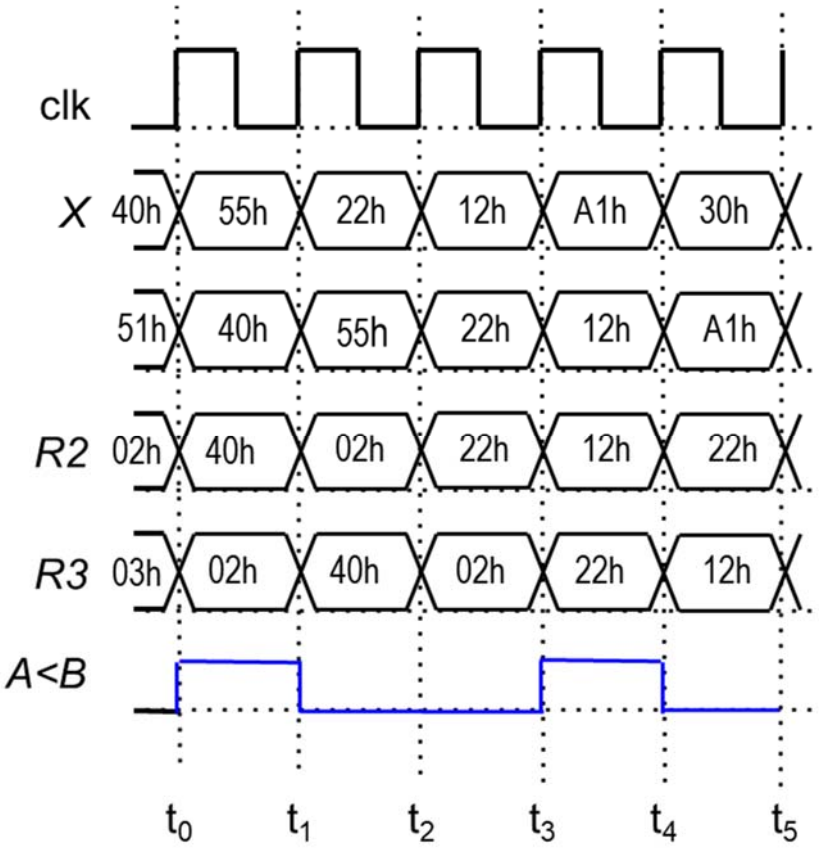
En cuanto a los valores que se cargan, vemos que en el registro *REG1* se guarda el valor de la entrada *X*. Por lo tanto, podemos escribir ya de entrada todos sus valores en el cronograma (en cada flanco se escribe el valor de *X* en el instante anterior al flanco). A *REG2* llega el valor propagado por el multiplexor y a *REG3* el valor de *REG2*. Por tanto, podremos escribir sus nuevos valores cuando hayamos razonado el comportamiento del multiplexor.

Para determinar el valor que se propaga por el multiplexor, observamos que la entrada de selección depende de la comparación entre el valor de *X* y de *REG1. Si X > REG1,* el multiplexor propaga a la salida el valor de *REG3*, en caso contrario se propaga el valor de *X.*

A partir de este razonamiento rellenamos el cronograma, que queda tal y como se muestra a continuación

Examen 2022/23-2

Asignatura	Código	Fecha	Hora inicio
Fundamentos de computadores	75.562	1/7/2023	10:00



PROBLEMA 4 [10%]

a) **[5%]** ¿Para qué se usa la memoria caché?

Para proporcionar a la CPU un acceso más rápido a la información de la memoria principal.

b) **[5%]** ¿Qué característica es más exclusiva de las arquitecturas de conjuntos de instrucciones (ISA) reducidos (RISC) respecto de los complejos (CISC)?

Tener dos tipos de instrucciones para acceder a datos en memoria: uno para lectura y otro para escritura.