1 PROBLEMAS DE SISTEMAS LÓGICOS COMBINACIONALES

Problemes d'àlgebra de Boole

1.1 Juan se ha examinado de tres asignaturas. Sus amigos han visto los resultados y se los han comentado:

- Has aprobado Mates o Física - dice uno de los amigos.
- Has suspendido Química o Mates - dice otro.
- Has aprobado sólo dos asignaturas - dice el tercero.

¿Qué asignaturas ha aprobado y cuál ha suspendido Juan?

Nota: en lenguaje natural, las conjunciones "os" suelen ser "exclusivas".

1.2 Determina qué pares de funciones son equivalentes, utilizando las propiedades del álgebra de Boole y/o mediante tablas de verdad:

a) F(x,y,z,w)=x'z'w' + xy'z'w' + yz'w + x'yw + xyzw + x'yzw' + y'zw'

G(x,y,z,w)=y'w' + x'y + yw

b) F(x,y,z,w)=x'y' + yw + x'w

G(x,y,z,w)=x'y'z'w' + x'z'w + yz'w + xyw + x'z

1.3 Dada la función $x\bar{y} + \bar{x}y = z$, demuestra que.

1.4 Demuestra algebraicamente que las funciones $F=\sum_4 m(1,2,4,7,8,11,13,14)$ y $G = A \oplus B \oplus C \oplus D$ son equivalentes.

1.5 Usando sólo una puerta NOR del número de entradas que haga falta y las puertas NOT que creas necesarias, diseña un circuito que realice una AND de tres entradas.

Problemes de Representació i codificació de les dades

1.6 Realitzeu els canvis de base que s'indiquen a continuació:

a) $EF02_{16}$ a binari natural

b) 235_{10} a hexadecimal

c) 0100111_2 a decimal

d) $FA12_{16}$ a decimal

e) 1345_{10} a binari natural

f) 1001011001100_2 a hexadecimal

1.7 ¿Cuál es el rango de números representables con 6 bits codificados en binario natural? ¿Y en Complemento a 2 (Ca2)?

1.8 Contesteu les següents preguntes:

a) Representeu en complement a 2 amb 8 bits els següents números: 79, -12 i 145.

b) Apliqueu l'algorisme de canvi de signe als números 79 i –12 per tal d'obtenir la representació del –79 i del 12

1.9 ¿Qué significa el término desbordamiento *(overflow)*? ¿Qué significa el término acarreo *(carry)*? Indica la relación que hay entre *overflow* y *carry* si los números son:

a) naturales.

b) enteros en complemento a dos.

Realiza la suma directamente en binario:

i) 0110 + 0111

ii) 1110 + 1010

iii) 0001 + 1100

Indica en cada caso cuándo el resultado no es representable con 4 bits suponiendo que los números son:

c) naturales codificados en binario con cuatro bits.

d) enteros codificados en Ca2 con cuatro bits.

1.10 Suposant que treballem en complement a 2 i els números es representen amb 6 bits,

a) feu la resta bit a bit de A=011010 i B = 001110 (C=A-B).

b) Calculeu la resta anterior però ara a partir de la suma d'A més -B.

c) Quines diferències trobeu entre l'operació feta a l'apartat a) i la de l'apartat b)?

1.11 Contesta a las siguientes preguntas:

a) Representa en complemento a 2 usando 6 bits los números 18 y -24.

b) Dados los números 010110 y 110101, representados en en Ca2: ¿Qué números decimales representan?

c) Realiza la resta 6-1 (en 4 bits) utilizando el método tradicional (6-1) y después utilizando el método de sumar el complementario (6+(-1)) representado en Ca2. ¿Qué diferencias encuentras?

1.12 Realitza la suma dels següents números naturals, representats en les bases indicades, amb el mateix nombre de dígits amb els que s'expressen. Indica si es produeix transport (*Carry*) i/o sobreeiximent (overflow).

a) 001010 y 100011 en base 2

b) 8B y 3A en base 12

c) ABC y A28 en base 16

1.13 Donada la suma següent:

$$\begin{array}{r} 011100 \\ \underline{+111000} \\ 1010100 \end{array}$$

Suposant que tant els operands com el resultat són números de 6 bits, digueu quins valors tenen els operands i el resultat en base 10 en els casos següents:

a) considerant que els números estan codificats en binari natural.

b) considerant que els números estan codificats en complement a 2.

1.14 Realitzeu les següents operacions amb números de 8 bits representats en complement a 2:

a) $63_{10} + (-120_{10})$

b) $63_{10} + 120_{10}$

El resultat s'ha d'expressar també en 8 bits i complement a 2. Indiqueu si es produeix sobreeiximent (overflow) i/o transport (*Carry*).

Problemes de especificació de taules de veritat

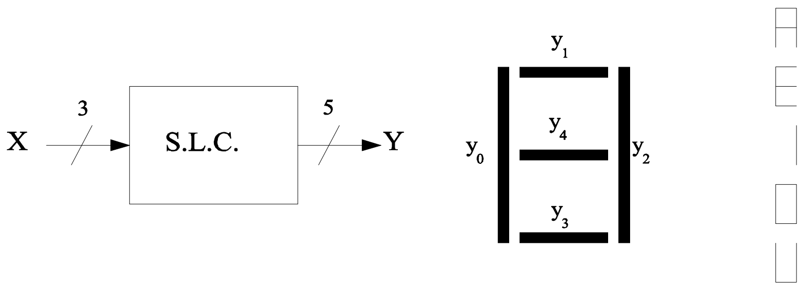
1.15 Haz la tabla de verdad de un circuito combinacional que implemente:

a) un multiplicador de dos números naturales de 2 bits.

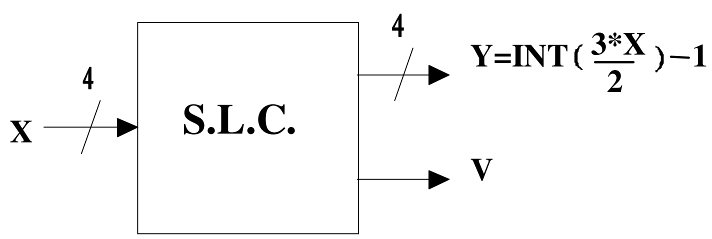
b) un multiplicador de dos números enteros de 2 bits representados en complemento a dos.

1.16 Haz la tabla de verdad de un sumador completo de un bit (tres bits de entrada y dos de salida)

1.17 Haz la tabla de verdad de un sistema que visualice las letras vocales en un *display* de 5 segmentos como el de la figura. Las vocales han de iluminarse en el visualizador según la siguiente codificación de X: A con 0, E con 1, I con 2, O con 3 y U con 4. No es posible que aparezcan en la entrada valores de X superiores a 4.



1.18 Realiza la tabla de verdad de un sistema lógico combinacional que, dado un número entero X de cuatro bits codificado en signo/magnitud, dé como salida otro número entero Y codificado de la misma forma, según la formula indicada en la figura:



El sistema dispondrá de un bit V que valdrá 1 cuando el resultado no se pueda representar con cuatro bits.

Nota: La función INT devuelve la parte entera por defecto.

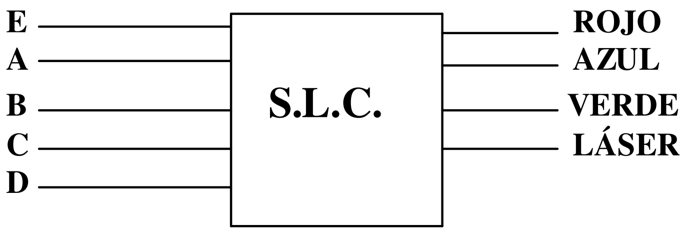
1.19 El popular conjunt musical "Nedant i guardant la roba" ens ha encarregat el disseny d'una part de la il·luminació de l'espectacle de la seva nova gira mundial. En particular, desitgen sincronitzar diferents grups de llums amb la música, per a la qual cosa disposen d'una bateria sintetitzada que envia quatre senyals a través dels seus sensors S1, S2, S3 i S4.

A partir del valor d'aquests quatre sensors, el sistema genera diferents funcions de sortida (F1, F2 i F3) que s'activen de la següent manera:

- F1 s'activa quan el sensor S4 està activat i hi ha un nombre imparell d'uns en S1, S2 i S3 (el zero es considera un número parell).
- F2 s'activa quan S4 està activat i hi ha algun altre '1' present a les entrades.
- F3 s'activa en cas que hi hagi com a mínim dos zeros presents a les entrades i el senyal F1 estigui desactivat.

Fes la taula de veritat del circuit.

1.20 La iluminación de una discoteca se compone de una combinación de luces de tres colores (rojo, azul y verde) y de un rayo láser. Se quiere diseñar un sistema combinacional que controle estas luces a partir de cuatro conmutadores (A, B, C, D) y de un interruptor general E.

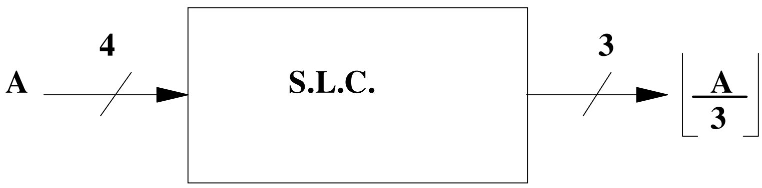


El funcionamiento del sistema es el siguiente:

- El interruptor general E hace que el sistema actúe (cuando vale '1') o que el sistema esté desconectado (cuando vale '0').
- La luz roja se enciende siempre que esté pulsado el conmutador A, o siempre que esté pulsado B pero no lo esté C.
- La luz azul no se enciende si A no está conectado o si lo están B o D.
- La luz verde se enciende si no está pulsado C, o si no lo está A y sí B, o si está pulsado D.
- El rayo láser se pone en funcionamiento cuando hay un número impar de luces encendidas.

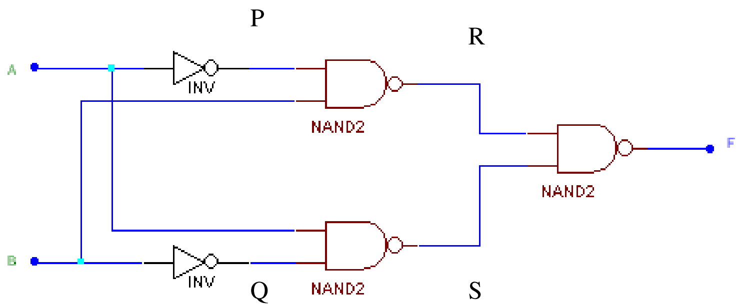
Realiza la tabla de verdad del circuito.

1.21 Haz la tabla de verdad de un sistema lógico combinacional que, dado un número de entrada de cuatro bits, que es un entero representado en complemento a dos, genere la parte entera por defecto de su división entre tres.

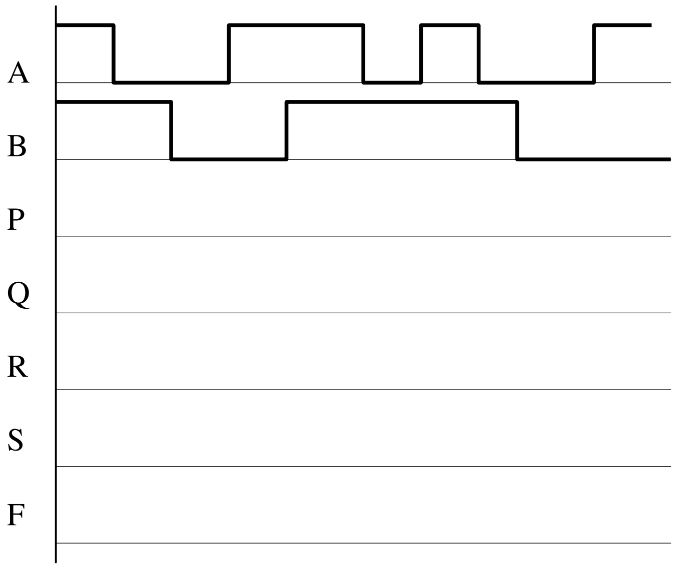


1.22 Contesteu les següents preguntes:

a) Feu l'anàlisi (doneu la taula de veritat i l'expressió algebraica) del següent circuit:



b) Omple el cronograma adjunt, suposant que no hi ha retards a les portes



1.23 Minimiza las funciones siguientes mediante mapas de Karnaugh:

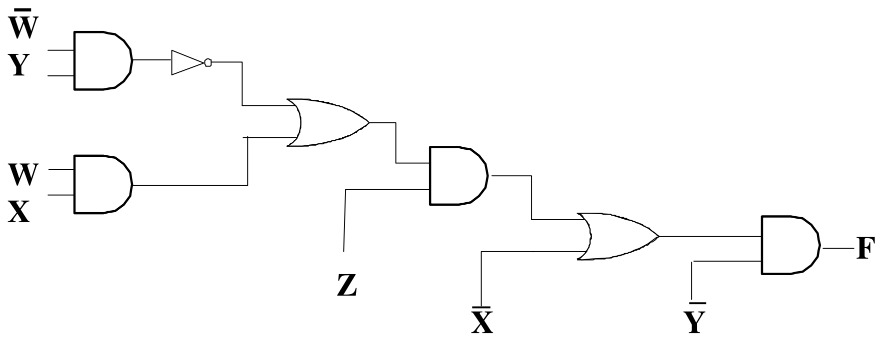
a) F(x,y,z,w)=x'y'z'w' + x'yz'w' + xy'zw' + xyzw + x'y'z'w + x'yzw

b) F(x,y,z,w)= Σ_4m (2,3,4,5,11,15)

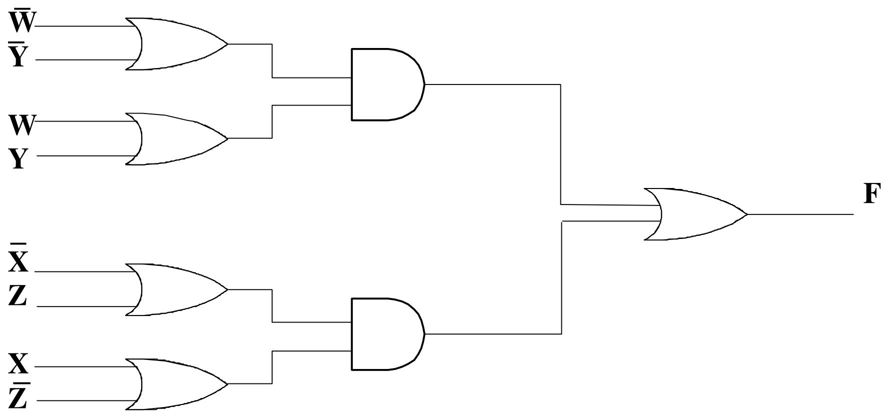
c) F(x,y,z,w)= xz' + y'z + x'y

1.24 Analiza los circuitos combinacionales siguientes y expresa en forma de suma de minterms la función que implementan. Sintetiza, si es posible, un circuito más sencillo que implemente la misma función:

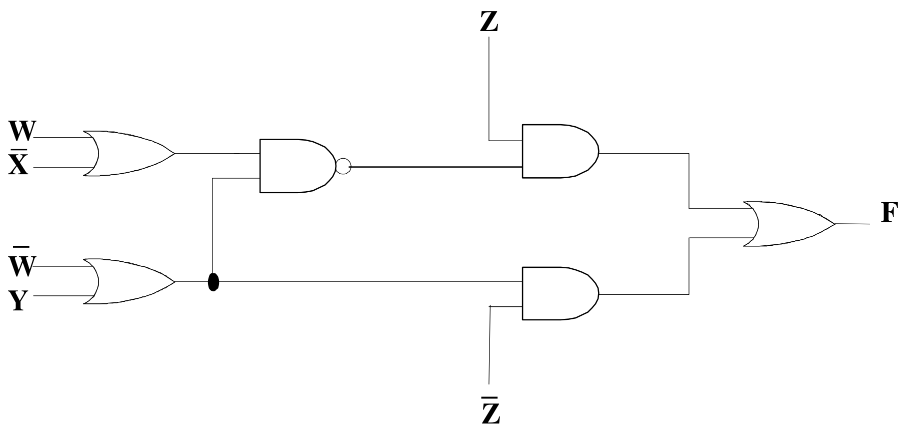
a)



b)



c)



1.25 Dada la función lógica siguiente:

$$f(a,b,c,d) = \sum_{4} m(0,4,5,10,13) + \sum_{4} d(2,6,7,8,9,12)$$

a) Sintetízala en dos niveles sin minimizarla.

b) Sintetízala en dos niveles con el mínimo número de puertas.

1.26 Dada la función lógica siguiente:

$$z = \overline{a}\,\overline{b} + \overline{a}\,b\,\overline{d} + \overline{a}\,d + b\,c\,d + a\,c\,d$$

a) Escribe su tabla de verdad.

b) Minimiza la función mediante un mapa de Karnaugh.

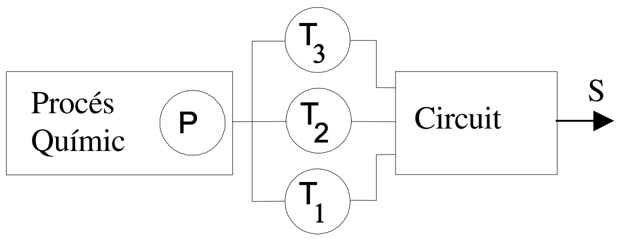
c) Implementa la función usando el mínimo número de puertas

1.27 Un procés químic té tres indicadors de temperatura sobre un punt P determinat, les sortides dels quals T_1, T_2, i T_3, adopten dos nivells de tensió segons la temperatura sigui menor, major o igual que unes determinades constants t_1, t_2 i t_3 respectivament ($t_1 < t_2 < t_3$).

$$T_1 = 1 \text{ si } t > t_1 \quad T_1 = 0 \text{ si } t \leq t_1$$

Les sortides T_2 i T_3 tenen el mateix comportament.

Sintetitza un circuit combinacional que generi una sortida S que valgui 1 si quan la temperatura a P estigui entre t_1 i t_2 o bé sigui superior a t_3. Realitza la implementació amb portes de forma mínima.



1.28 Juan tiene una planta muy delicada, por eso necesita un sistema de riego con control de la temperatura y de la humedad. El sistema combinacional tiene tres entradas y dos salidas.

Entradas:

- Un sensor de temperatura (T): se pone a '1' si la temperatura de la tierra supera un límite prefijado t0.
- Dos sensores de humedad de la tierra (H0 y H1): se ponen a '1' cuando la humedad de la tierra supera los límites h0 y h1, respectivamente. El límite h0 es inferior al límite h1 (h0<h1).

Salidas:

- *Regar* (R): activa el riego de la planta cuando se pone a '1'.
- *Calentar* (E): activa el calentamiento de la tierra cuando se pone a '1'.

Las especificaciones del sistema son:

- La planta se riega siempre que la tierra se seca, es decir, siempre que no se supera el límite h0.
- También se riega cuando la temperatura T supera el límite t0 y la humedad de la tierra es inferior a h1.
- La tierra de la planta se calienta cuando la temperatura T es inferior a t0 y la humedad es superior a h0.

Contesta los siguientes apartados:

a) Encuentra la tabla de verdad de las funciones R y E.

b) Implementa la función E de forma mínima a dos niveles.

c) Implementa la función R de forma mínima a dos niveles.

Problemes bàsics de Blocs Combinacionals

1.29 Implementa un Full Adder usando:

- Un multiplexor para la salida C_{i+1}
- Un decodificador y una puerta OR para la salida S_i

1.30 El popular conjunt musical "Nedant i guardant la roba" ens ha encarregat el disseny d'una part de la il·luminació de l'espectacle de la seva nova gira mundial. En particular, desitgen sincronitzar diferents grups de llums amb la música, per a la qual cosa disposen d'una bateria sintetitzada que envia quatre senyals a través dels seus sensors S1, S2, S3 i S4.

A partir del valor d'aquests quatre sensors, el sistema genera diferents funcions de sortida (F1, F2 i F3) que s'activen de la següent manera:

- F1 s'activa quan el sensor S4 està activat i hi ha un nombre imparell d'uns en S1, S2 i S3 (el zero es considera un número parell).
- F2 s'activa quan S4 està activat i hi ha algun altre '1' present a les entrades.
- F3 s'activa en cas que hi hagi com a mínim dos zeros presents a les entrades i el senyal F1 estigui desactivat.

Contesta els següents apartats:

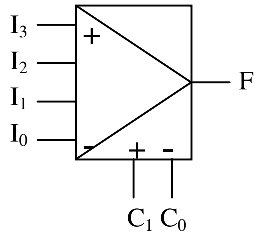
a) Implementa la funció F1 de forma mínima.

b) Implementa la funció F2 utilitzant un multiplexor 16-1.

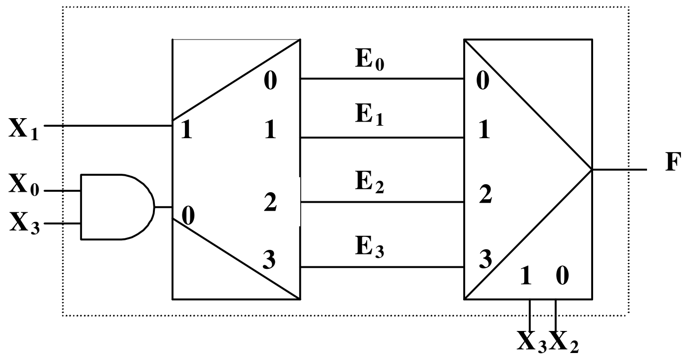
c) Implementa la funció F3 utilitzant un descodificador 4-16.

d) Implementa el sistema complet amb una memòria ROM. Indica la grandària i el contingut.

1.31 Dado un multiplexor como el de la figura, escribe la expresión algebraica de F.



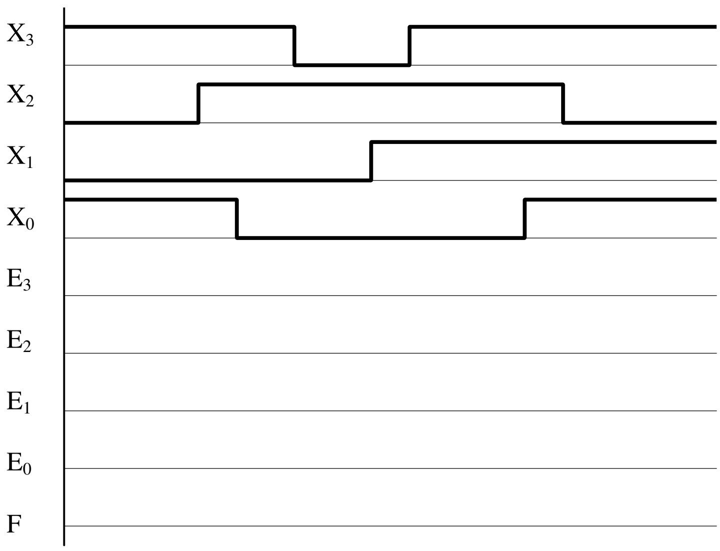
1.32 Donat el següent circuit:



a) Troba les expressions algebraiques d'E_i (i=0,1,2,3) en funció d'x_3, x_1 i x_0 (no fa falta que les minimitzis).

b) Escriu la taula de veritat d'$F(x_3,x_2,x_1,x_0)$.

c) Omple el següent cronograma:



1.33 Dissenya els següents circuits:

a) Dissenya un multiplexor de 16 canals, utilitzant 2 multiplexors de 8 canals i les portes lògiques que facin falta.

b) Dissenya un multiplexor de 16 canals, utilitzant únicament multiplexors de 4 canals (no usis cap porta lògica addicional).

c) Dissenya un multiplexor de 4 canals usant un descodificador de 2 a 4 i les portes lògiques que facin falta.

1.34 Dissenya els següents circuits:

a) Dissenya un descodificador de 3 entrades/8 sortides utilitzant dos descodificadors de 2 entrades/4 sortides i les portes lògiques que facin falta.

b) Dissenya un descodificador de 4 entrades/16 sortides utilitzant únicament cinc descodificadors de 2 entrades/4 sortides.

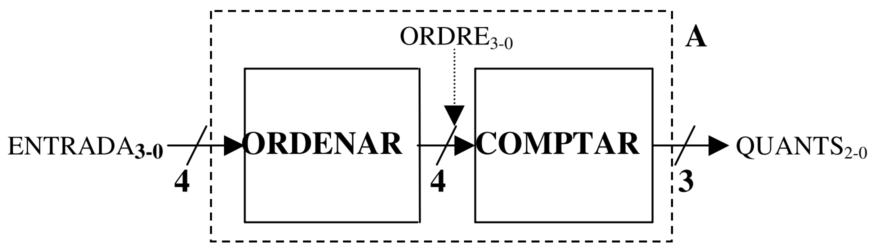
1.35 Dissenya un sistema combinacional a nivell de blocs que a partir de 3 bits d'entrada X_2 X_1 X_0 (corresponents als dígits 0 al 7 codificats en binari natural) obtingui tres bits de sortida Z_2 Z_1 Z_0, codificats en binari natural, que representin la funció $Z=(3 \cdot X) \bmod 8$.

Realitza el disseny utilitzant únicament:

a) Un descodificador de 3 entrades/8 sortides i un codificador de 8 entrades/3 sortides

b) Un sumador de 4 bits. Per aquest cas, recorda que $3 \cdot X = 2 \cdot X + X$

1.36 Dissenya el sistema combinacional de la figura, destinat a comptar el nombre d'uns que té un vector binari *entrada*$_{3\text{-}0}$ de quatre bits.



El bloc *ORDENAR* ordena el vector binari *entrada*$_{3\text{-}0}$, col·locant tots els uns a la dreta i tots els zeros a l'esquerra. Per exemple:

si *entrada* = 0101   llavors *ordre* = 0011

si *entrada* = 0000   llavors *ordre* = 0000

El bloc COMPTAR genera a la sortida *quants*$_{2\text{-}0}$ la codificació binària de la quantitat d'uns del vector binari *ordre*$_{3\text{-}0}$. Per exemple:

si *ordre* = 0011   llavors *quants* = 010 (2)

si *ordre* = 0000   llavors *quants* = 000 (0)

Respon els següents apartats:

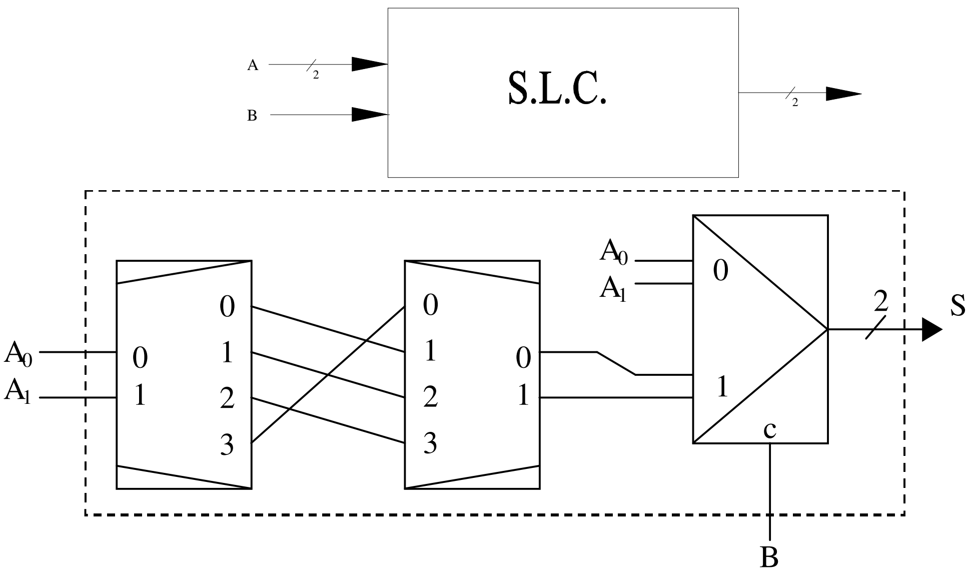
a) Fes la taula de veritat del bloc *ORDENAR* y expressa en forma de suma de productes les seves funcions de sortida *ordre*$_{3\text{-}0}$. Realitza el disseny sense minimitzar i utilitzant portes lògiques.

b) Dissenya el bloc *COMPTAR* utilitzant només blocs combinacionals (exceptuant memòria ROM), de la grandària que faci falta.

c) Dissenya el sistema combinacional A amb una memòria ROM (indicant la seva grandària i contingut).

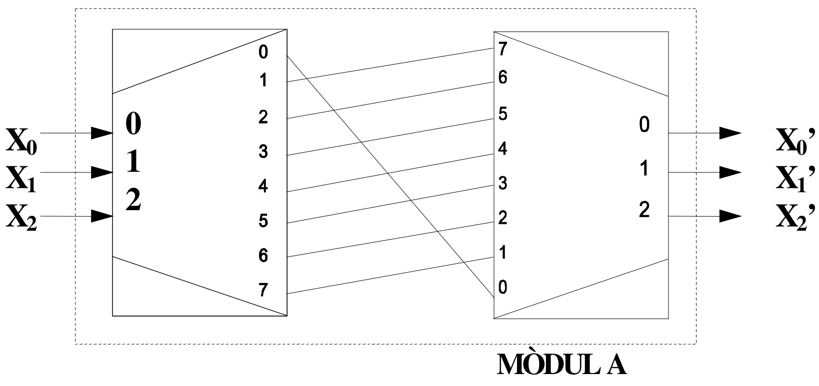
Problemes avançats de blocs combinacionals

1.37 Quina funció realitza el següent circuit si interpretem les entrades i la sortida com nombres codificats en binari natural?

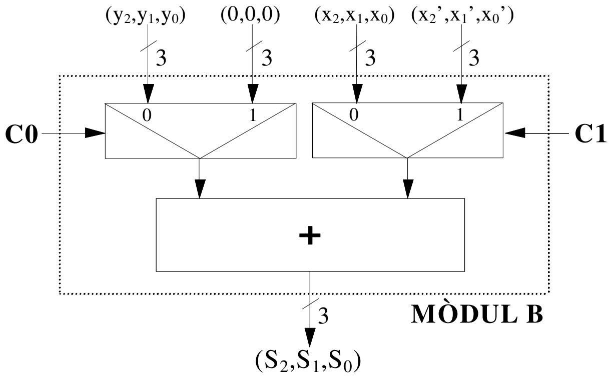


1.38 Contesta les següents preguntes:

a) Què fa el mòdul A, si X=(x2,x1,x0) és un nombre enter codificat en complement a dos?



b) Què fa el mòdul B, si X=(x2,x1,x0) i X'=(x2',x1',x0') són els mateixos números utilitzats a l'apartat anterior i que Y=(y2,y1,y0) és també un número enter codificat en complement a dos?



1.39 Implementa la función y=(x+1)mod 8 con decodificadores y codificadores suponiendo que trabajamos con números de 3 bits.

1.40 Contesta a los siguientes apartados:

a) Implementa un desplazador de 1 bit a la izquierda para números de 4 bits usando un multiplexor de buses de dos canales de 4 bits. El bit de menor peso del resultado se ha de poner a cero.

b) ¿A qué operación aritmética equivale este desplazamiento?

c) Indica cuándo se produciría *overflow* para el caso de que la entrada sea un número natural o sea un número entero representado en complemento a dos

1.41 Contesta a los siguientes apartados:

a) Implementa un desplazador programable de números de 8 bits. El desplazador tiene una entrada de un bit, c, que le indica el número de bits que se desplazan a la izquierda. Los bits de menor peso del resultado se ha de poner a cero.

b) ¿A qué operación aritmética equivale este desplazamiento?

c) Indica cuándo se produciría *overflow* para el caso de que la entrada sea un número natural o entero en complemento a dos

Si el desplazamiento fuese a la derecha, en lugar de a la izquierda:

d) ¿A qué operación aritmética equivaldría ahora este desplazamiento?

e) ¿Qué valor deberían tomar los bits de mayor peso en el caso de que los números desplazados fuesen naturales o enteros representados en complemento a dos?

f) Indica cuándo se produciría *overflow* para el caso de que la entrada sea un número natural o entero en complemento a dos

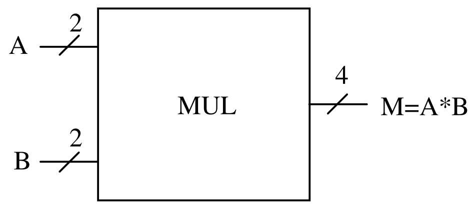
1.42 Un codificador con prioridad cíclica es un codificador con n entradas de control adicionales que le indican cuál de las 2^n entradas de datos es la de máxima prioridad para ser codificada a la salida. A partir de ahí, la prioridad es cíclica. Es decir, si n=3 y las entradas de control codifican un 4, el orden en que deben explorarse las entradas de datos para codificar la salida es: 4, 3, 2, 1, 0, 7, 6, 5.

a) Implementa un codificador de prioridad cíclica para n=1

b) Idem para n=2

c) Idem para n=3

1.43 Es disposa d'un circuit combinacional que implementa un multiplicador de dos nombres enters, de



2 bits, representats en complement a dos.

a) Escriu la taula de veritat del bloc MUL.

b) Realitza amb blocs combinacionals un sistema que calculi:

$$(a^4 + a^3 + a^2)$$

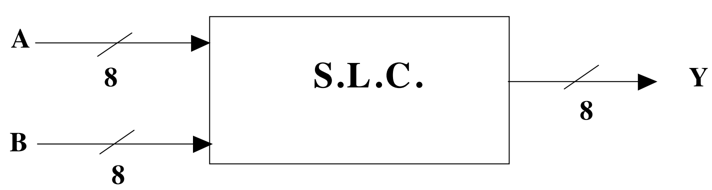
on 'a' és un nombre enter de dos bits representat en complement a dos. Es poden usar blocs de qualsevol número de bits (especifica de quants), inclòs el de l'apartat a, que no cal dissenyar. Una possible solució només requereix un sumador i dos multiplicadors.

1.44 Realiza la función (3.x) mod 8 usando únicamente un sumador de 4 bits

1.45 Usando sumadores y restadores, diseña un circuito que permita comparar dos números naturales de 4 bits, y nos diga, mediante tres bits independientes, si A>B, A=B y A<B

1.46 Diseña un circuito que obtenga el máximo de dos números naturales de 4 bits

1.47 Diseña a nivel de bloques un sistema combinacional que haga la función Y=*INT* (0.75**MAX* (A,B)), donde la función *INT* calcula la parte entera por defecto y la función *MAX* calcula el máximo de los dos números naturales.



Hazlo de 3 maneras distintas, considerando:

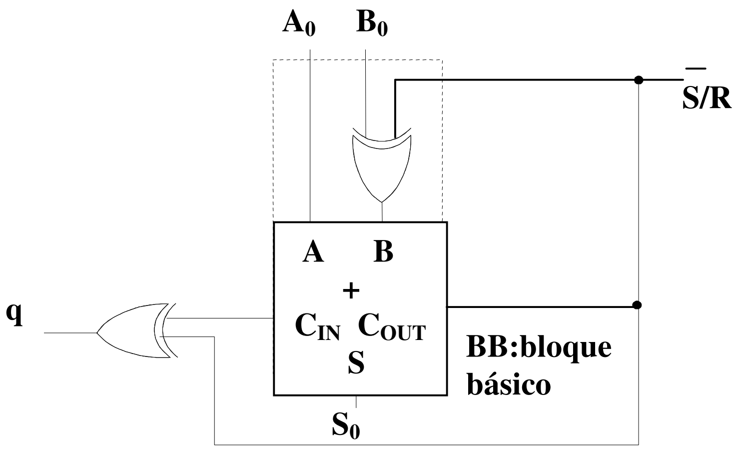
a) 0.75= ½ + ¼

b) 0.75 =(2+1)/3

c) 0.75 = 1 – ¼

Compara los tres diseños

1.51 Dado el circuito combinacional de la figura:



Contesta los siguientes apartados:

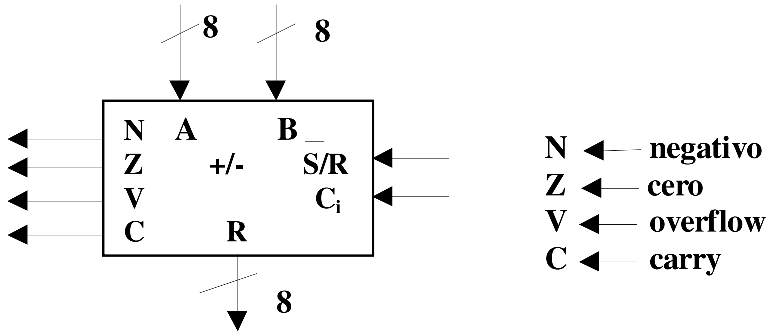
a) Obtén la tabla de verdad del bloque básico BB.

b) Analiza el funcionamiento del circuito completo y especifica qué operaciones aritméticas hace en función de la señal de control $\overline{s} / r$.

c) Diseña, utilizando el bloque básico BB anterior, un circuito combinacional que haga las mismas operaciones pero con n bits.

1.52 Diseña un sumador/restador de ocho bits de números naturales, codificados en binario natural, y de números enteros, codificados en Complemento a dos. Además del resultado, ha de proporcionar cuatro bits de condición: N, Z, V y C. Este sistema responde al bloque siguiente:

$\overline{S}/R$	R
0	$A+B+C_i$
1	$A+B-C_i$



Cuando $\overline{S}/R$ vale '0' ha de sumar los dos operandos de entrada (A y B) más el *carry* de entrada (Ci), y cuando vale '1' ha de hacer la resta de los dos operandos de entrada (A - B). El bit de condición N se activa si el resultado R, interpretado en Complemento a dos, es negativo. El bit Z se activa si el resultado es cero. El bit V se activa si el resultado no es correcto (interpretando los operandos A y B como números enteros). El bit C se activa si hay *carry* del bit más significativo (en el caso de la suma) o *borrow* hacia el bit más significativo (en el caso de la resta).

Contesta los siguientes apartados:

a) Indica por qué el sumador/restador no necesita una entrada que le indique cómo están codificados los números de entrada A y B.

b) Implementa el sumador/restador utilizando únicamente sumadores completos de un bit y puertas de cualquier tipo y número de entradas.

1.53 Contesta los siguientes apartados:

a) Usando un único sumador de 4 bits, diseña una ALU con salida *Out* de 4 bits que, a partir de 2 entradas de control S_1 y S_0, realice las siguientes operaciones sobre dos números A y B de 4 bits:

- S_1=0, S_0=0: Out=A+B
- S_1=0, S_0=1: Out=A-B
- S_1=1, S_0=0: Out=A
- S_1=1, S_0=1: Out=-A

Para el diseño debes usar únicamente un sumador con propagación de acarreo de 4 bits y las puertas lógicas que consideres necesarias

b) Añade a la ALU diseñada en el apartado anterior la lógica necesaria para calcular los siguientes bits de condición:

- O': *Overflow* si los números de entrada son naturales
- O: *Overflow* si los números de entrada son enteros
- N: Signo si los números de entrada son enteros. N=1 si el resultado de la operación es negativo, y cero en caso contrario.
- Z: Z=1 si el valor de *Out* es 0, y vale 0 en caso contrario

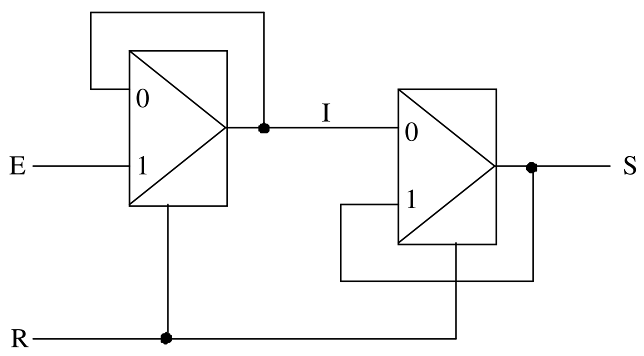
1.54 Usando un único sumador de 4 bits, diseña una ALU con salida *Out* de 4 bits que, a partir de 3 entradas de control S_2, S_1 y S_0, realice las siguientes operaciones sobre dos números A y B de 4 bits:

- S_2= 0, S_1=x, S_0=x: B
- S_2= 1, S_1=0, S_0=0: Out=A+B
- S_2= 1, S_1=0, S_0=1: Out=A-B
- S_2= 1, S_1=1, S_0=0: Out=B >> 1
- S_2= 1, S_1=1, S_0=1: Out=A and B (bit a bit)

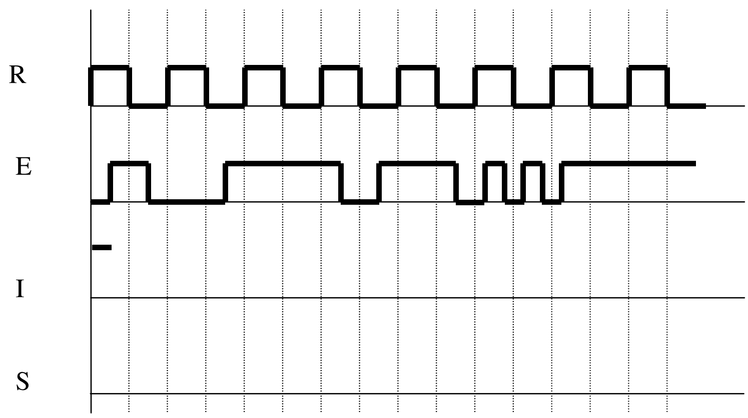
2 PROBLEMAS DE SISTEMAS LÓGICOS SECUENCIALES

Cronogrames de sistemes seqüencials

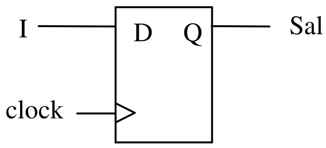
2.1 Donat el circuit següent:

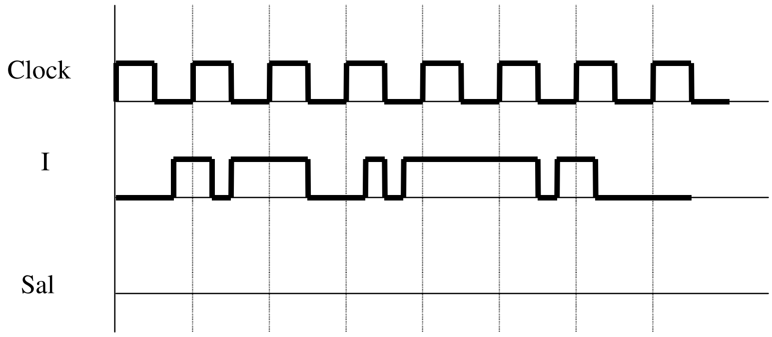


Dibuixa el valor dels senyals I i S al cronograma, suposant que el senyal I val inicialment 1.

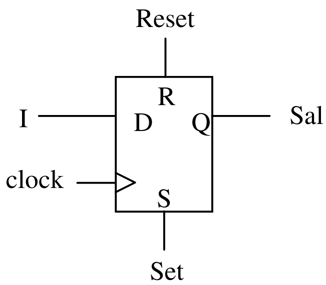


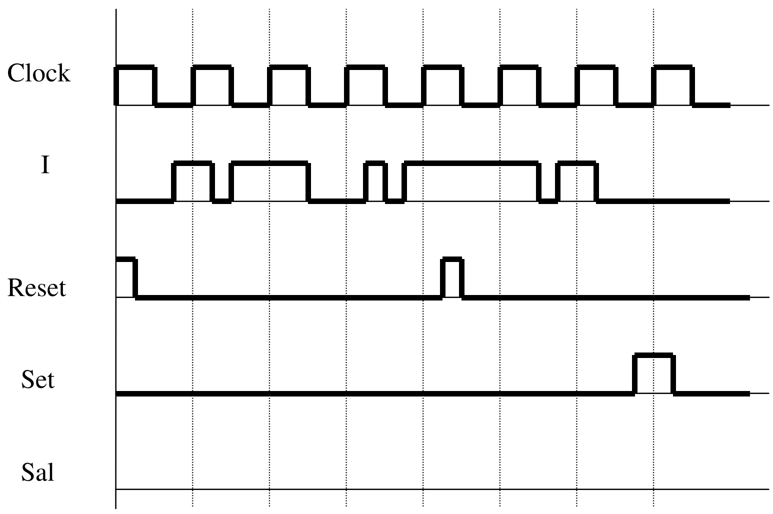
2.2 Completa el cronograma que correspon al circuit descrit a la figura. Inicialment, suposarem que la sortida Q val zero.



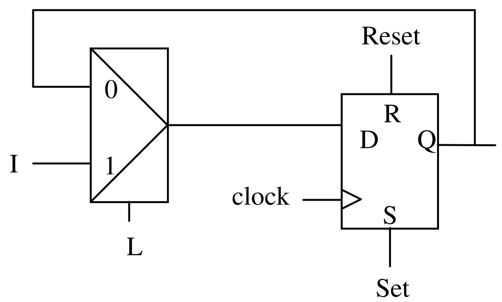


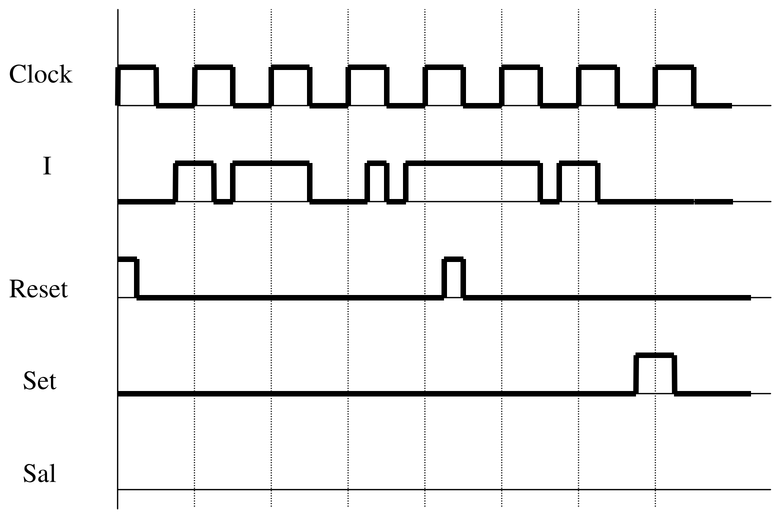
2.3 Completa el cronograma que correspon al circuit descrit a la figura. Inicialment, suposarem que la sortida Q val zero.



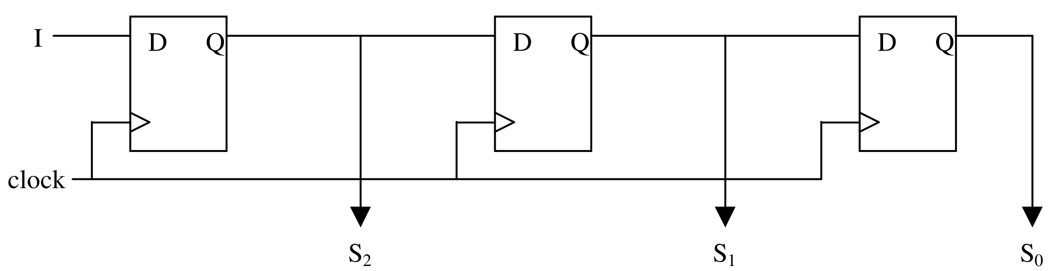


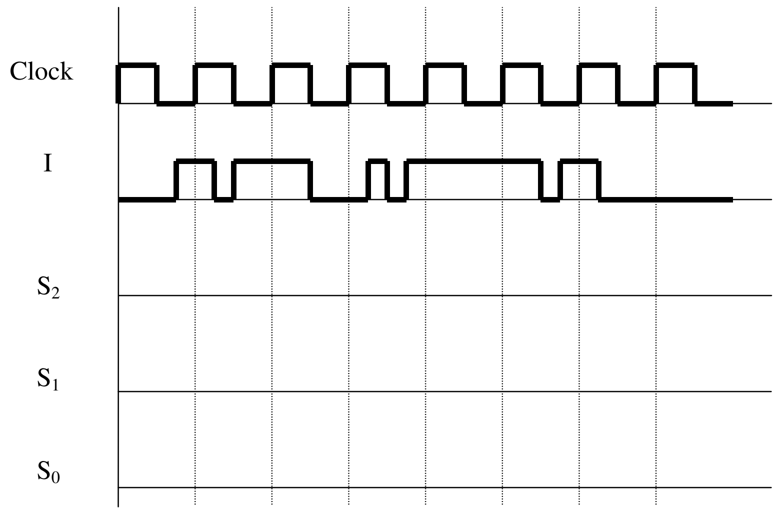
2.4 Completa el cronograma que correspon al circuit descrit a la figura. Inicialment, suposarem que la sortida Q val zero.



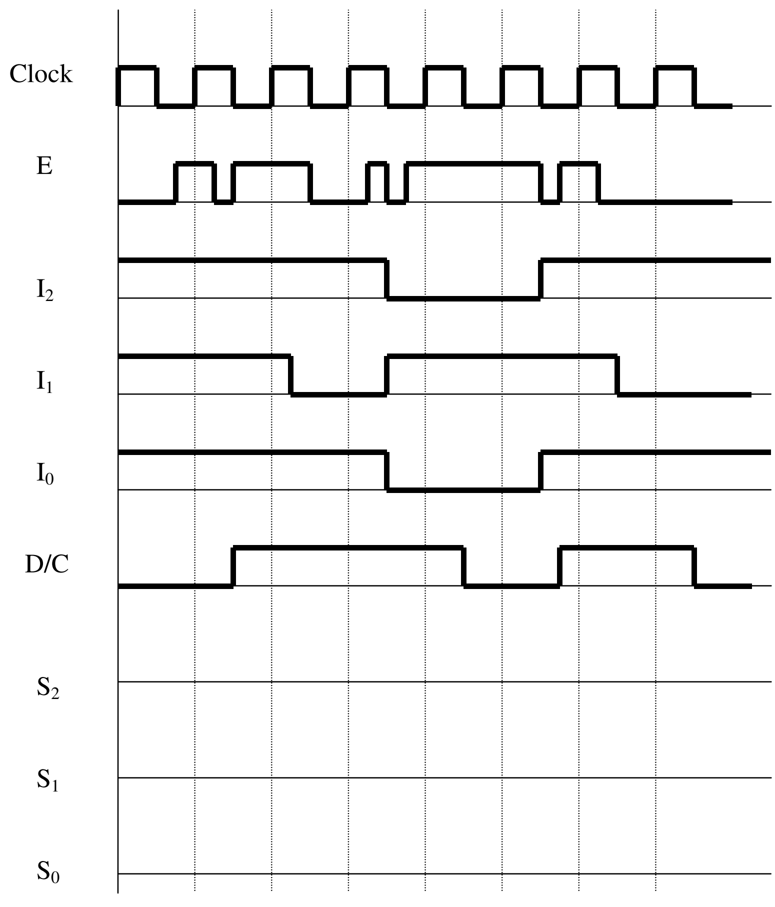


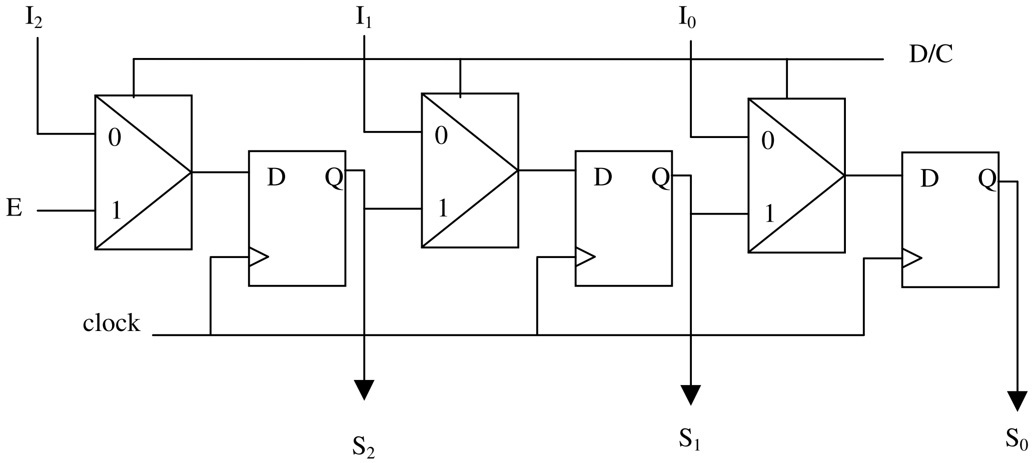
2.5 Completa el cronograma que correspon al circuit descrit a la figura. Inicialment, suposarem que las sortides Q de tots els biestables valen zero.





2.6 Completa el cronograma que correspon al circuit descrit a la figura. Inicialment, suposarem que las sortides Q de tots els biestables valen zero. Descriu en poques paraules que fa el circuit en funció del senyal D/C.

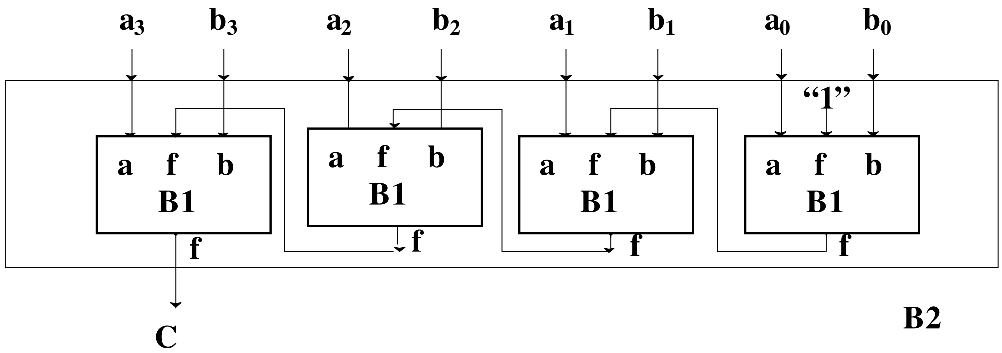




2.7 El bloc combinacional B2 és un comparador de dos números binaris A i B. Concretament,

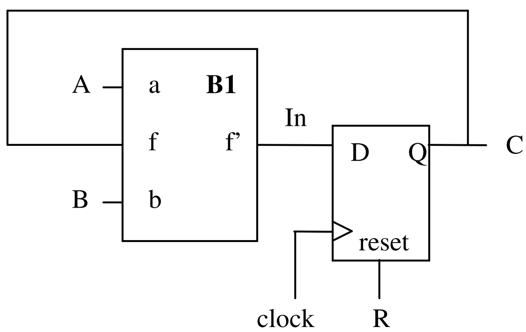
C = 0 si A és menor que B

C = 1 si A és més gran o igual que B

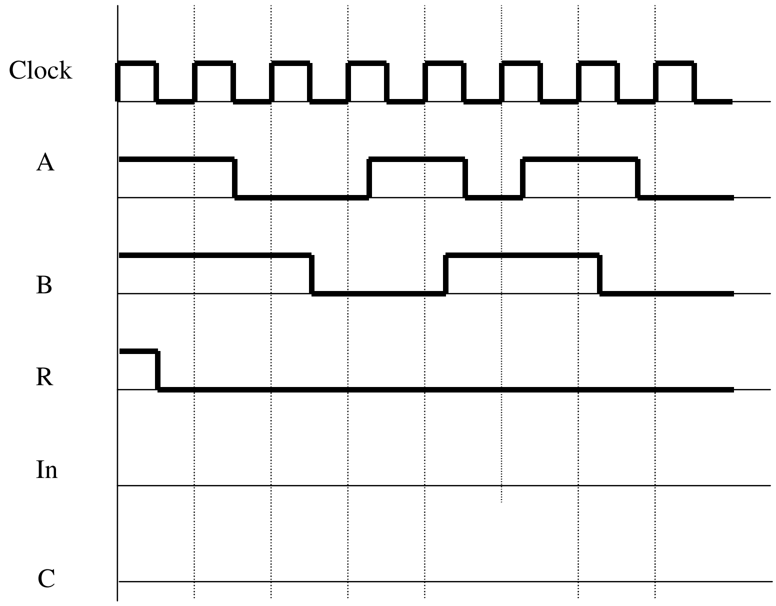


a) Veient com s'ha construït el bloc B2, dedueix la taula de veritat del bloc B1.

b) Utilitzant el bloc B1 s'ha construït aquest altre circuit. Descriu quina funció pot realitzar aquest circuit



c) Completa el cronograma que correspon al circuit de l'apartat b).



Especificació de Sistemes Lògics Seqüencials mitjançant grafs d´estats.

2.8 Especifica (fes el graf d'estats) d'un sistema lògic seqüencial que generi cíclicament la seqüència de números 0,1,3,4,7, codificats en binari.

2.9 Especifica un sistema lògic seqüencial que, en funció d'una entrada X, generi per la sortida les següents seqüències de números (codificats en binari):

X = 0 => Sortida = 0,5,7,11,14,0,5,7,...

X = 1 => Sortida = 7,11,14,7,11,...

2.10 Diseña un reconocedor *de secuencia* con salida Z=1 durante un ciclo de reloj al detectar por la línea serie de entrada X la secuencia '0110'.

2.11 Especifica un sistema lògic seqüencial que posi la sortida a 1 durant un cicle quan hagin arribat per l'entrada dos o més 0s consecutius, o bé quatre o més 1s consecutius. Inicialment la sortida ha de valer 0. Abans de fer el disseny, fes un cronograma amb l'exemple i comprova que la sortida Y esta un cicle desplaçada con respecte a la entrada X. ¿Quin és el motiu?

Per exemple:

```
X   10001011111011...
Y   00011000001100...
```

(s'entén que els bits entren d'esquerra a dreta)

2.12 Un sistema lògic seqüencial rep a l'entrada dígits decimals ([0..9]) codificats en BCD (un per cicle). Quan ha arribat la seqüència 121, la sortida es posa a 1 durant un cicle. Dibuixa el graf d'estats del sistema.

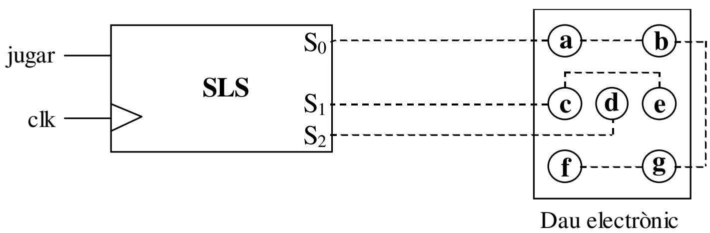
2.13 Diseña un sistema secuencial que reconozca las secuencias de tres bits '101' y '010', y detecte el posible solapamiento entre ellas. Cuando se detecta una de estas secuencias se activa un bit de salida a 1.

2.14 Especifica un sistema lògic seqüencial en què la seqüència 101 a l'entrada (d'un bit) provoqui que la sortida es posi a 1, i la seqüència 001 a l'entrada provoqui que la sortida es posi a 0. Inicialment la sortida ha de valer 1.

2.15 Un sistema secuencial tiene tres posibles símbolos de entrada A, B y C. Dibuja el grafo de estados de dicho circuito sabiendo que es capaz de detectar, con un '1' en la salida, que han entrado un número par de símbolos A o un número impar de símbolos B. Utiliza la codificación de entrada: A=00, B=10, C=01.

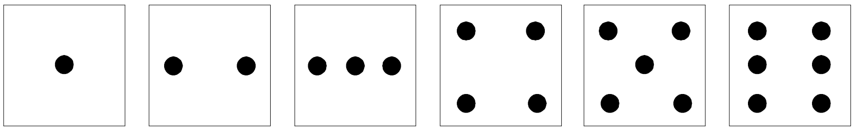
Prevé también un posible error a la entrada si se presenta la combinación '11', en cuyo caso el sistema volverá al estado inicial.

2.16 Es vol dissenyar un sistema lògic seqüencial per a tirar un dau. Les sortides del sistema estaran connectades als punts d'un "dau electrònic" de la manera següent:



Dau electrònic

Les combinacions possibles del dau són les següents:



Mentre el senyal d´entrada jugar estigui activat, el sistema generarà cíclicament les 6 combinacions del dau. Quan el senyal jugar estigui desactivat, no es produirà cap transició. Així, quan un jugador deixi de polsar el senyal jugar veurà al dau la combinació que li ha sortit (se suposa que la freqüència del rellotge és molt alta i no es poden arribar a distingir les combinacions intermitges).

Dibuixa el graf d'estats del sistema.

2.17 Es vol dissenyar el comandament del control remot d'un cotxe de joguina. El comandament té dos botons: E i D.

Si el cotxe està parat, en polsar-se qualsevol botó es posa en moviment: gira a l'esquerra si es polsa E, gira a la dreta si es polsa D, i tira endavant si es polsen tots dos alhora.

Mentre el cotxe està en moviment, si es polsa E

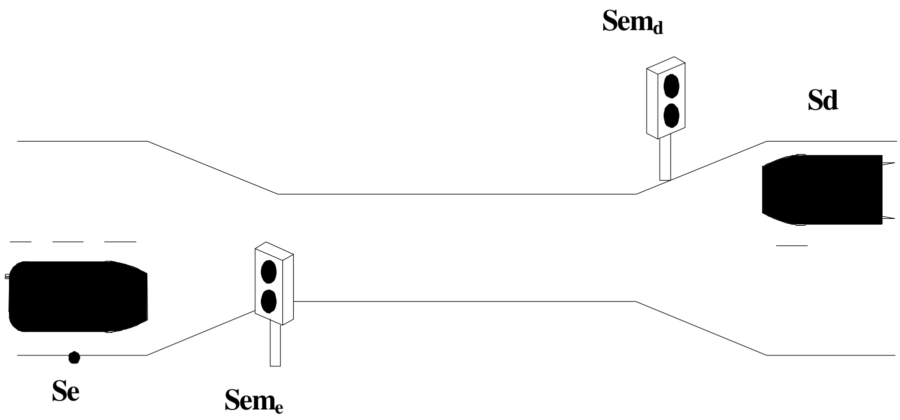
- girarà a l'esquerra si anava recte,
- anirà recte si estava girant a la dreta,
- seguirà girant a l'esquerra si ja ho feia

i similarment quan es polsi D. Quan es polsin tots dos botons alhora es pararà.

Especifica un sistema lògic seqüencial que rebi com a entrada 2 senyals connectats als botons E i D (1: polsat; 0: no polsat) i generi dos senyals Z1 i Z0 que governaran el cotxe, segons la taula següent:

Acció del cotxe	**Z1**	**Z0**
Girar a la dreta	0	0
Girar a l'esquerra	0	1
Parar	1	0
Moure's endavant	1	1

2.18 Dibuixa el graf d'estats d'un sistema lògic seqüencial que controli els semàfors d'un pont pel qual només poden passar cotxes en un sentit.



A cada extrem del pont hi ha:

- un sensor que posa el senyal *Sd* (o *Se*) a 1 quan hi ha un cotxe.
- un semàfor que es posa vermell quan li arriba un 0 pel senyal sem_d (o sem_e) i es posa verd quan li arriba un 1.

El sistema ha de funcionar de la manera següent:

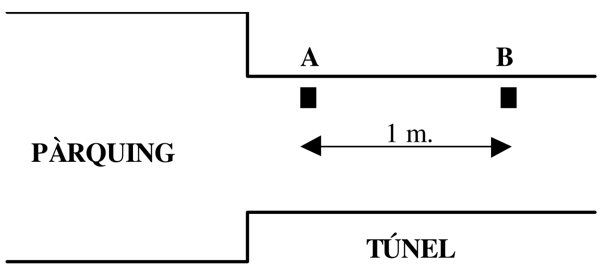
- Els semàfors es posen verds de forma alternada durant 2 cicles de rellotge com a mínim
- Després del segon cicle de rellotge, el semàfor que està verd s'hi queda en verd fins que no arribi un cotxe per l'altre extrem.

2.19 Un pàrquing necessita comptar el nombre de cotxes que hi ha en cada moment.

Els cotxes entren i surten del pàrquing pel mateix túnel, en el qual només hi cap un cotxe. Al túnel hi ha dos sensors, A i B, separats un metre, de manera que es pot saber si un cotxe està entrant o sortint segons l'ordre en què s'activin els sensors (se suposa que tots els cotxes medeixen més d'un metre i que entre un cotxe i el següent hi ha més d'un metre).

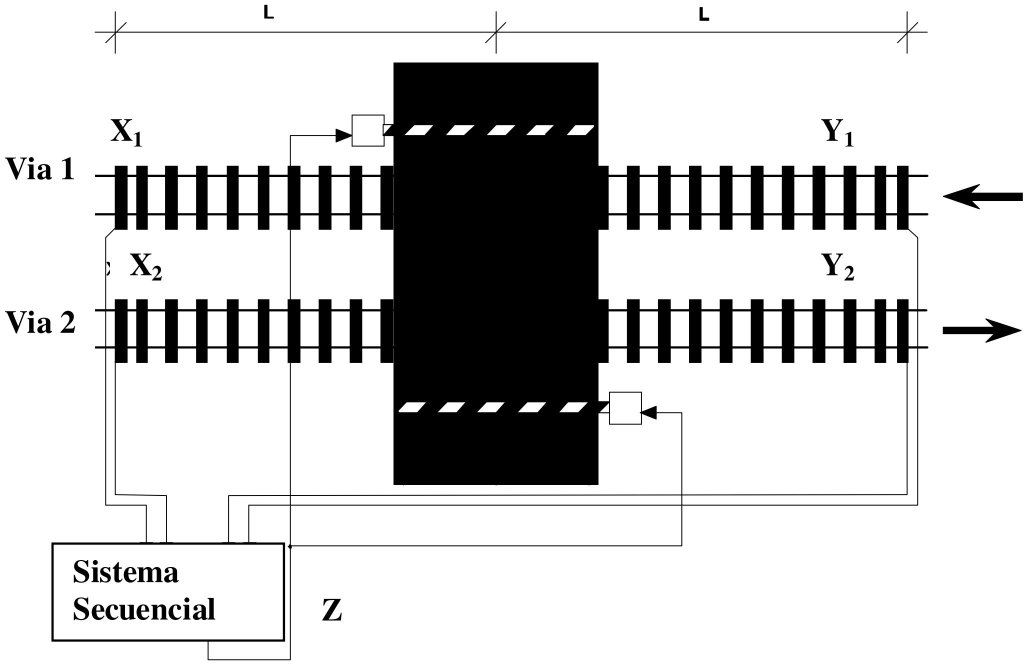
Quan un cotxe ha entrat totalment al pàrquing s'incrementa el nombre de cotxes aparcats, i quan n'ha sortit totalment es decrementa. Un cotxe pot fer marxa enrera en qualsevol moment.

Dos cotxes no es trobaran mai de cara al túnel. Tampoc no hi haurà mai peatons.



Dissenya un sistema lògic seqüencial que generi els senyals COMPTAR i DESCOMPTAR a partir dels senyals A i B que envien els sensors.

2.20 Se desea diseñar un sistema secuencial para controlar un paso a nivel con barreras como el que se muestra en la figura 1.

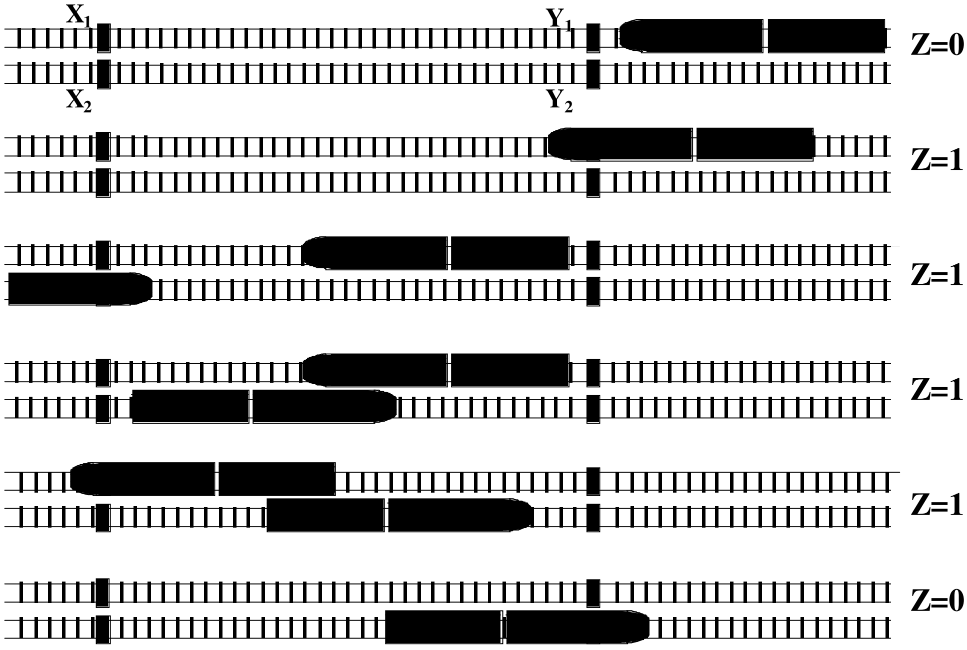


2.20.1 Figura 1

Los trenes siempre circulan en los sentidos indicados en la figura 1. Se han colocado sobre las vías cuatro sensores (X_1, X_2, Y_1, Y_2) que envían una señal de valor '1' mientras un tren pasa por encima de él.

El sistema tiene como salida la señal Z, que toma el valor '1' cuando las barreras están cerradas y '0' cuando están abiertas. Las barreras se han de cerrar (Z=1) cuando un tren encuentre el primer sensor (Y_1 para la vía 1 y X_2 para la vía 2), y se han de abrir (Z=0) cuando encuentre el segundo sensor. En la figura 2 puede verse un ejemplo.

Nota: la longitud de los trenes es menor que L (un mismo tren no puede estar activando simultáneamente Y_i y X_i, i=1,2) y los trenes que circulan por la misma vía están separados por una distancia mayor que 2L, por lo que las combinaciones '1x1x' y 'x1x1' están prohibidas.



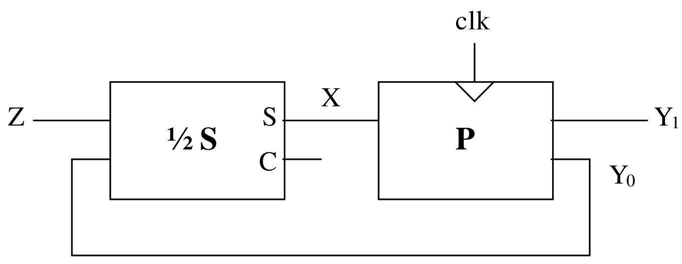
2.20.2 Figura 2

a) Realiza el grafo de estados para la vía 1.

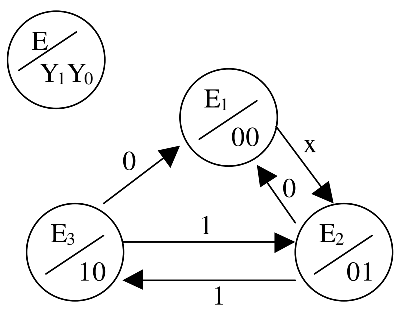
b) Realiza el grafo de estados para las dos vías.

c) ¿Cómo se podría resolver, de forma fácilmente escalable, el problema para *n* vías, con $n \geq 5$?

2.21 Dibuixa el graf d'estats del sistema de la figura (a), sabent que P és un sistema lògic seqüencial que es comporta segons el graf de la figura (b).



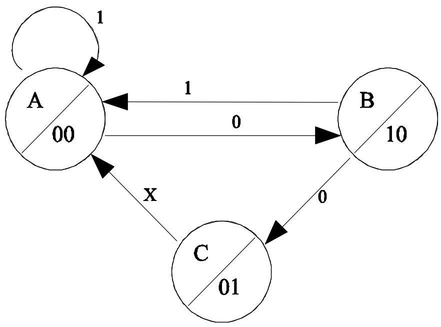
(a)



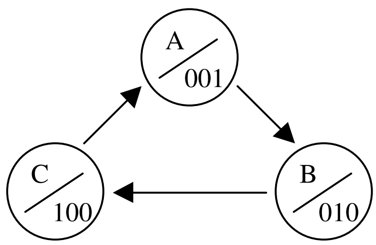
(b)

2.22 Implementa els sistema lògic seqüencials que descriuen cadascun dels grafs d´estats següents. En cada cas, fes-ho codificant els estats per tal de minimitzar el nombre de biestables.

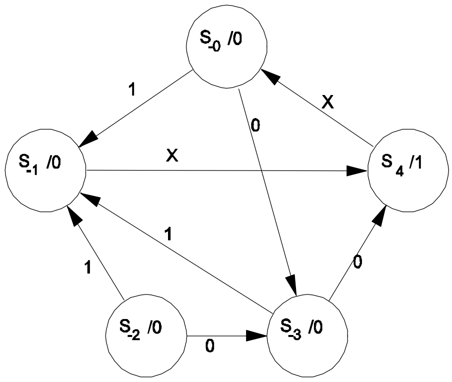
a)



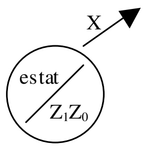
b)

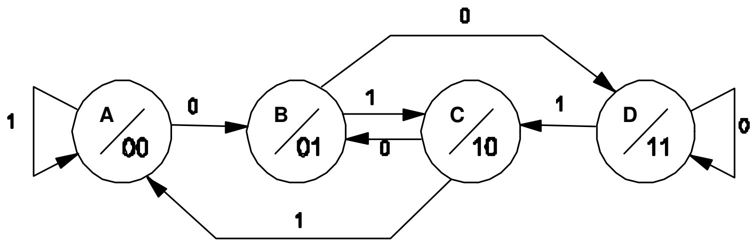


c)

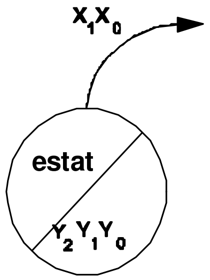


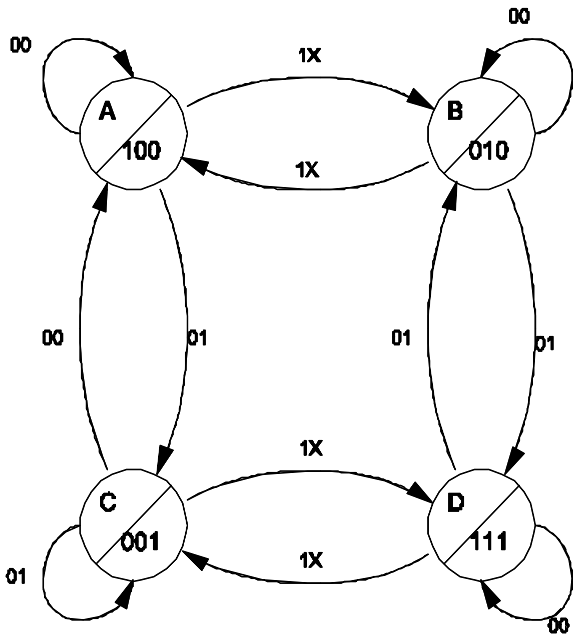
2.23 Implementa el sistema lògic seqüencial següent, usant el menor nombre possible de biestables i una memòria ROM.





2.24 Fes la síntesi del sistema lògic seqüencial següent, usant 2 biestables i el menor nombre possible de portes.





2.25 Diseña el sistema *secuencial* propuesto en el problema 2.13 usando biestables D y una memoria ROM del tamaño necesario. Indica el tamaño y contenido de la ROM.

2.26 Fes el disseny del sistema seqüencial modelat pel graf d'estats del problema 2.8 utilitzant portes lògiques i una memòria ROM de la mida que calgui. Indica clarament la grandària i contingut de la ROM. Introdueix un senyal asíncron INICIO que faci que el sistema comenci la seqüència generant un 1 per la seva sortida.

2.27 Considera el grafo de estados del problema 2.10.

a) Implementa el sistema con el menor número posible de biestables tipo D y las puertas lógicas necesarias.

b) Implementa el sistema con un registro de desplazamiento de longitud adecuada y las puertas que consideres necesarias.

c) Compara las dos soluciones, considerando:

- la facilidad en el diseño,
- el coste,
- la facilidad de modificar cualquier parámetro del diseño.

2.28 Dissenya i implementa amb el menor nombre possible de biestables un sistema lògic seqüencial que funcioni com un comptador reversible mòdul 5. Comptarà endavant o endarrera en funció d'un senyal d'entrada.

X = 0 : comptar endavant

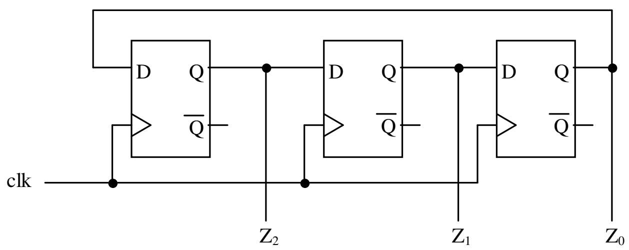
X = 1 : comptar endarrera

La sortida es codificarà segons aquesta taula:

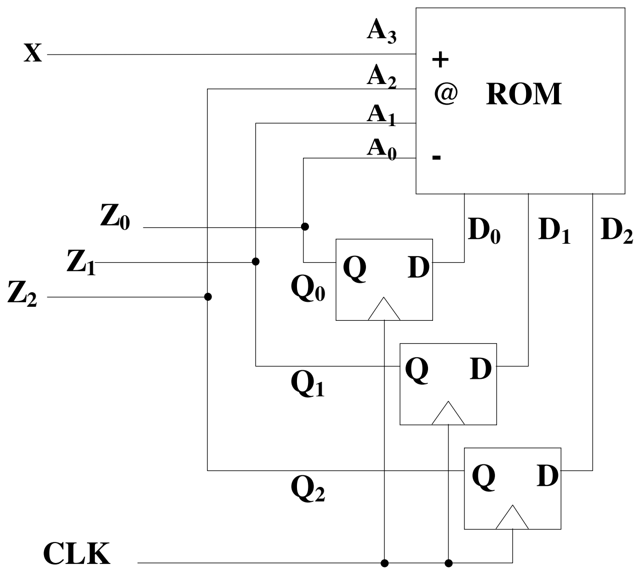
	Sortida
	0000
	0001
	0011
	0111
	1111

Utilitza una memòria ROM per a implementar les funcions de transició, i dos blocs combinacionals (diferents d'una ROM) per a implementar les sortides.

2.29 Fes l'anàlisi (dibuixa el graf d'estats) que descriu el funcionament del sistema lògic seqüencial següent.



2.30 Suposant que la memòria ROM de la figura (a) té el contingut que es mostra a la figura (b), dibuixa el graf d'estats del sistema lògic seqüencial de la figura (a).

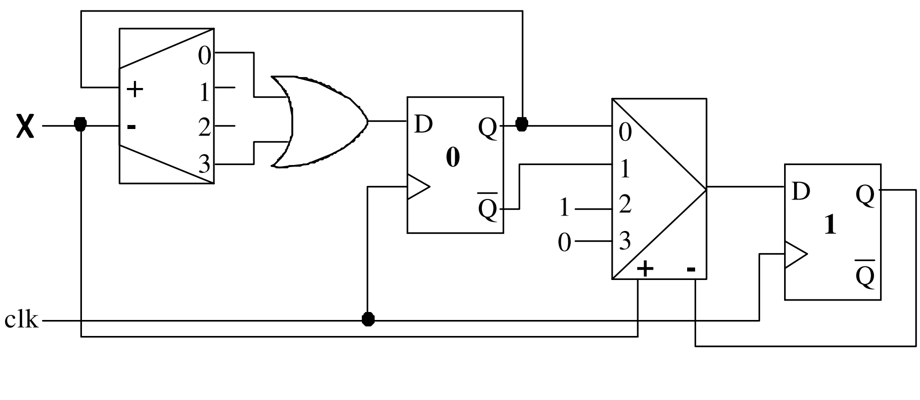


(a)

@	M[@]		
	D_2	D_1	D_0
0	0	0	1
1	0	1	0
2	0	1	1
3	1	0	0
4	1	0	1
5	0	0	0
6	0	0	0
7	0	0	0
8	1	0	1
9	0	0	0
10	0	0	1
11	0	1	0
12	0	1	1
13	1	0	0
14	0	0	0
15	0	0	0

(b)

2.31 Troba el graf d'estats del sistema lògic seqüencial que implementa el circuit següent:



3 Máquina Rudimentaria

Lenguaje máquina

3.1 Traduce a lenguaje ensamblador de la MR las siguientes instrucciones expresadas en hexadecimal.

a) 1006h
b) 4E20h
c) 80A1h
d) EA64h
e) C6F1h
f) E846h
g) A825h

3.2 Pasa a notación binaria y hexadecimal las siguientes instrucciones escritas en lenguaje ensamblador de la MR.

a) ADD R0, R1, R2
b) SUBI R4, #1, R5
c) BGE 43
d) BR 12
e) AND R1, R7, R6
f) ASR R4, R2

3.3 Indica cuál es el resultado de ejecutar las siguientes instrucciones, dando el contenido final de los registros y posiciones de memoria que se modifican y el valor de los indicadores de condición. Se supone para cada caso que el contenido inicial de los registros y posiciones de memoria es el siguiente:

R1=2454h R2=5656h
R3=FFFFh R4=0000h
R5=0002h

Memoria

Dirección	Contenido
00h	0339h
01h	63AFh
02h	EA00h
03h	3304h
04h	7834h
05h	54AAh

a) SUB R1,R2,R5

b) ADDI R3, #1, R1

c) LOAD 3(R5), R1

d) STORE R4, 1(R4)

e) ASR R2, R7

3.4 Indica qué operación realiza la siguiente secuencia de instrucciones.

D024h

C864h

D844h

3.5 La instrucción *SWAP a, b* está presente en algunos lenguajes de alto nivel. Esta instrucción intercambia los valores de las variables *a* y *b*. Escribe en lenguaje máquina de la MR un programa que se comporte como esta instrucción, suponiendo que las variables *a* y *b* se encuentran:

a) La variable *a* en la posición de memoria 60h y la variable *b* en el registro R2 de la Unidad de Proceso.

b) La variable *a* en la posición de memoria 25h y la variable *b* en la posición 1Dh.

3.6 Escribe en lenguaje máquina de la MR un programa que cuente el número de bits que valen 1 en un número NUM almacenado en memoria. Supón que el número NUM se encuentra en la dirección 03h y que las instrucciones se almacenan a partir de la dirección 04h.

3.7 Escribe en lenguaje máquina de la MR un programa que realice la división por 2^n de un número NUM. El exponente n es un número positivo menor de 16. Utiliza desplazamientos para realizar la división, y supón que:

- el número NUM se encuentra en la dirección 13h
- el exponente n la dirección 17h
- las instrucciones se almacenan a partir de la dirección 2Ah

3.8 Escribe en lenguaje máquina de la MR un programa que indique si un número NUM es par o impar. El número NUM esta en la posición 03h de memoria. Utiliza la instrucción AND para averiguar el valor del bit de menor peso (0 par, 1 impar).

El programa debe poner un 1 en la posición de memoria 02h si el número es impar, y un 0 si es par. Supón que las instrucciones están almacenadas a partir de la dirección 04h.

3.9 Traduce el siguiente programa, escrito en lenguaje máquina de la MR (hexadecimal), a lenguaje ensamblador. Indica qué operación realiza, suponiendo que la primera instrucción ejecutable se encuentra en la dirección 02h y que en las posiciones 00h y 01h hay datos.

Memoria	
Dirección	Contenido
00h	0009h
01h	XXXXh
02h	0800h
03h	D004h
04h	D824h
05h	8809h
06h	D224h
07h	DB09h
08h	8005h
09h	5001h

3.10 Indica qué operación realiza la siguiente secuencia de instrucciones

CA24h
D145h
C945h

3.11 Dado el siguiente fragmento de programa, escrito en algún lenguaje de alto nivel, tradúcelo a lenguaje máquina de la MR suponiendo que está almacenado a partir de la dirección de memoria 00h.

si R1>R2 ***entonces*** ejecutar la instrucción de la dirección 25h
sino seguir ejecutando en secuencia
fsi

3.12 Dado el siguiente fragmento de programa, escrito en algún lenguaje de alto nivel, tradúcelo a lenguaje máquina de la MR suponiendo que está almacenado a partir de la dirección de memoria A0h.

si (R1>R2) ***entonces*** ejecutar la instrucción de la dirección 25h
sino ejecutar la instrucción de la dirección 64h
fsi

3.13 Dado el siguiente fragmento de programa, escrito en algún lenguaje de alto nivel, tradúcelo a lenguaje máquina de la MR suponiendo que está almacenado a partir de la dirección de memoria 38h.

si (R1>R2) **o** (R3 < 0) ***entonces*** ejecutar la instrucción de la dirección 25h
sino ejecutar la instrucción de la dirección 64h
fsi

3.14 Dado el siguiente fragmento de programa, escrito en algún lenguaje de alto nivel, tradúcelo a lenguaje máquina de la MR suponiendo que está almacenado a partir de la dirección de memoria 22h.

> ***mientras*** (R1>R2) ***y*** (R3<=R4) ***hacer***
>
> R1 := R1 - 1;
>
> R3 := R3 + 1;
>
> ***Fmientras***

Unidad de proceso

3.15 Indica cómo afectaría a los siguientes elementos de la MR el aumentar de 8 a 16 el número de registros del Banco de Registros.

a) Tamaño de la instrucción

b) Anchura de la memoria

c) Buses internos de datos de la Unidad de Proceso

d) Registros internos de la Unidad de Proceso

3.16 Indica cómo afectaría a los siguientes elementos de la MR el aumentar el tamaño de la memoria hasta tener una memoria de 64 Kpalabras en lugar de 256 palabras.

a) Registro Contador de Programa (PC)

b) Registro de direcciones (R@)

c) Tamaño de la instrucción (Registro RI) y anchura de la memoria

d) Anchura del Banco de Registros y buses internos

3.17 Indica qué modificaciones habría que hacer en los siguientes elementos de la MR si se desease trabajar con números en complemento a dos de 32 bits, en lugar de hacerlo con números de 16 bits.

a) La ALU

b) La Memoria

c) El Banco de Registros

d) Los buses de datos internos de la Unidad de Proceso.

e) Los registros internos de la Unidad de Proceso

3.18 Realiza el diseño de la Unidad de Evaluación de la Condición de Salto.

3.19 Rediseña la ALU para que, además de los indicadores de condición Z y N, también calcule los indicadores:

* V: desbordamiento para enteros
* C: desbordamiento para naturales

3.20 Contesta a los siguientes apartados:

a) Rediseña la ALU para que un único sumador realice las operaciones de suma y de resta que necesitan las instrucciones ADD, SUB, ADDI y SUBI.

b) ¿Es necesario modificar la codificación de las instrucciones?

3.21 Queremos substituir la instrucción ASR de la MR por una instrucción que compare dos operandos fuente (CMP Rf1 Rf2) y modifique el indicador de condición Z de la manera que se indica a continuación, sin importar el valor que se guarde en el indicador de condición N:

* Z=1 si los dos operandos son iguales
* Z=0 si los dos operandos son distintos

a) Diseña la nueva ALU y rediseña el hardware que rodea la ALU.

b) ¿Qué implicaciones tiene el intercambiar la instrucción ASR por la instrucción CMP en el formato de las instrucciones?

Unidad de Control

3.22 Suponiendo que el tiempo de respuesta de los distintos elementos de la Unidad de Proceso es el siguiente:

- 10 ns. para cada multiplexor y decodificador
- 50 ns. para el sumador de direcciones y el incrementador conectado a la entrada del PC
- 40 ns. para el bloque que realiza la evaluación de la condición de salto
- 20 ns. para leer un registro del Banco de Registros
- 100 ns. el tiempo de respuesta de la ALU
- 100 ns. el tiempo requerido para leer o escribir un dato en la memoria RAM

Responde a las siguientes preguntas, despreciando el tiempo que pueda tardar en reaccionar la Unidad de Control.

a) Calcula el tiempo requerido para ejecutar cada una de las fases de cada instrucción (cada estado del grafo de estados reducido).

b) Indica la frecuencia máxima de reloj a la que puede funcionar la MR.

3.23 Suponiendo que se incluye un registro RMEM entre el multiplexor SELDAT y la memoria, que no afecte al camino entre la salida de la memoria y el registro IR, contesta a los siguientes apartados:

a) Diseña el nuevo grafo de estados para ejecutar las instrucciones de la MR de forma que se optimice el número de estados necesario para ejecutar la instrucción LOAD. Indica la tabla de salidas de los nuevos estados

b) Contesta a los apartados a) y b) del problema 3.22 utilizando el nuevo grafo de estados.

c) ¿Qué mejora supone (en tiempo de ejecución) incluir el registro RMEM en la Unidad de Proceso?

3.24 Queremos diseñar una máquina con el mismo juego de instrucciones que la MR:

a) Propón una codificación de instrucciones en el supuesto de que dicha máquina:

- tenga 16 registros de propósito general de 32 bits.
- trabaje con números en complemento a dos de 32 bits.
- tenga una memoria de 2^{22} palabras de 32 bits.

¿Cuál es el tamaño máximo del operando inmediato?

b) ¿Qué modificaciones habría que hacer en el resto de la Unidad de Proceso?

c) ¿Cómo se puede aumentar el número de instrucciones aritméticas hasta 64? ¿Qué repercusiones tendría este aumento en el resto de parámetros de las instrucciones?

d) ¿Se podrían añadir más instrucciones de salto sin modificar el resto de formatos de instrucción? ¿Cuántas y cuáles sugerirías, suponiendo que en la Unidad de Proceso se incluyesen los indicadores de condición C (acarreo) y V (desbordamiento)?

3.25 Contesta a las siguientes preguntas:

a) Diseña un Banco de Registros que posea dos puertos de lectura. El hecho de que un Banco de Registros tenga dos puertos de lectura implica que se pueden leer dos registros al mismo tiempo.

b) ¿Qué cambios en el diseño de la Unidad de Proceso de la MR provocaría el substituir su Banco de Registros por el que se propone en el apartado a)? ¿Tendría que cambiar el formato de las instrucciones?

3.26 Queremos añadir al juego de instrucciones de la MR una instrucción aritmética que multiplique por dos un operando fuente almacenado en un registro y deje el resultado en un registro destino (*MULDOS Rf, Rd*).

a) ¿Qué implicaciones tiene en el formato de las instrucciones el añadir esta instrucción?

b) Diseña la nueva ALU y rediseña el hardware que rodea la ALU.

Ejecución de instrucciones

3.27 Dado el siguiente programa, escrito en lenguaje ensamblador de la MR:

```
00h BR 03h
01h SUB R0, R0, R0
02h BR 05h
03h ADD R0, R0, R0
04h BGE 01h
```

Contesta a las siguientes preguntas:

a) ¿En qué orden se ejecutan las instrucciones?

b) ¿Cuántos ciclos tarda en ejecutarse el programa (suponiendo el grafo más simplificado para la Unidad de Control)?

c) Indica los diferentes valores que toma el PC en cada una de las fases de ejecución de las instrucciones durante la ejecución del programa.

d) Compara la secuencia de direcciones almacenada en el PC con la secuencia de direcciones correspondiente a la ejecución de instrucciones. ¿Se corresponden ambas secuencias? Justifica la respuesta.

3.28 Dado el siguiente bucle, escrito en lenguaje ensamblador de la MR:

```
00h SUB R3, R2, R0
01h BL 08h
02h SUB R4, R2, R0
03h BNE 06h
04h ADDI R2, #1, R2
05h BR 07h
06h SUB R4, #2, R4
07h BG 00h
```

Suponiendo que inicialmente R3=100_{10} y R2=49_{10}, contesta a los siguientes apartados:

a) Describe qué hace el cuerpo del bucle

b) ¿Cuántas veces se ejecuta el bucle?

3.29 ¿Qué se le indica a la Unidad de Proceso de la MR con estas señales de control?

a) Ld_PC = 1, Ld_IR = 1 y PC/@ = 0

b) Ld_PC = 1, Ld_IR = 1 y PC/@ = 1

c) Ld_RA = 1 y CRf= 1

d) ERd = 1

e) Ld_RZ = 1, Ld_RN = 1, ERd = 1, OPERAR = 1 y CRf = 2

3.30 Se desea diseñar la Unidad de Control de la MR como un sistema lógico secuencial cuya función de estado se realiza de forma mínima a tres niveles, y cuya función de salida se realiza con una memoria ROM. Escribe la tabla de transición y excitación suponiendo que se usan biestables D, y el contenido de la memoria ROM.

3.31 Contesta a los siguientes apartados:

a) ¿Por qué se ha de realizar la decodificación de la instrucción?

b) ¿Por qué se intenta reducir el número de estados de la Unidad de Control de la MR?

3.32 Si el Banco de Registros de la Unidad de Proceso de la MR poseyese dos puertos de lectura (ver problema 3.25):

a) ¿ Podría ser más corto el tiempo de ejecución de alguna instrucción?

b) ¿Mejoraría el rendimiento global de la MR durante la ejecución de instrucciones?.

3.33 Si la ALU de la MR, además de los indicadores de condición Z y N también calculara los indicadores:

- V: desbordamiento para enteros
- C: desbordamiento para naturales

sería interesante incluir más instrucciones de salto:

a) ¿Qué instrucciones de salto se podrían añadir?

b) ¿Sería necesario cambiar el formato de las instrucciones?

Programación en Lenguaje Ensamblador

3.34 Traduce el siguiente programa, escrito en un lenguaje de alto nivel, a lenguaje ensamblador de la MR y posteriormente a lenguaje máquina, suponiendo que:

- se almacena a partir de la dirección 00h
- las variables *a* y *b* se encuentran en las posiciones de memoria 00h y 01h respectivamente
- la primera instrucción ejecutable del programa está en la dirección 02h
- la instrucción *swap a, b* intercambia los valores de las variables *a* y *b*

programa ejemplo1;

var a, b: ***entero***;

a := 13;

b := 16;

mientras (a > 10) ***hacer***

a := a -1;

b := b +2;

fmientras;

si (a<b) ***entonces*** swap (a, b)

sino b :=a-1

fsi;

fprograma.

3.35 Traduce el siguiente programa, escrito en un lenguaje de alto nivel, a lenguaje ensamblador de la MR y posteriormente a lenguaje máquina, suponiendo que se almacena a partir de la dirección 00h.

```
programa ejemplo2;
var a, b, c : entero;
a := 13;
b := 1;
c := 0;
mientras (a > 10) o (b < 20) hacer
        si ( a < 11 ) y ( b > 18 ) entonces c := c + 3 fsi;
        a := a - b;
fmientras;
fprograma.
```

3.36 El siguiente programa calcula el máximo común divisor de dos números *a* y *b* según el algoritmo de restas de Euclides:

```
programa ejemplo3;
var a=5, b=15, mcd: entero;
mientras (a <> b) hacer
si (a>b)    entonces a:=a-b
            sino     b:=b-a
fsi;
fmientras;
mcd := a;
fprograma.
```

a) Tradúcelo a lenguaje ensamblador de la MR

b) Traduce el programa escrito en lenguaje ensamblador del apartado a) a lenguaje máquina, tanto en binario como en hexadecimal, suponiendo que:

- se almacena a partir de la dirección 0Fh
- las variables *a* y *b* se almacenan en las posiciones 0Fh y 10h respectivamente
- la variable *mcd* se almacena en la posición 11h
- la primera instrucción ejecutable del programa está en la dirección 12h.

3.37 Dado el siguiente fragmento de programa escrito en lenguaje ensamblador de la MR:

```
N=6
X=8
Y=12
.org 16
adrA:   .DW X,Y
        .DW 0, 0
adrB:   .DW N, X+Y, N+2
adrV:   .DW 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
        .DW 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

a) Indica el contenido de la tabla de símbolos.

b) Ensambla las siguientes instrucciones utilizando la anterior declaración de datos. Indica el valor final de los registros una vez ejecutadas las instrucciones.

- LOAD adrA+1(R0), R1
- LOAD adrB-4(R0), R2
- ADD R1, R2, R3
- STORE adrV+N(R2), R3

c) Sin añadir más etiquetas, escribe en lenguaje ensamblador de la MR un programa que:

- Asigne el valor N a la posición 14 del vector adrV.
- Asigne el contenido de la posición 10 del vector adrV, más el contenido de la posición 2*N de la memoria, a la posición X+Y del vector.

3.38 Escribe en un lenguaje de alto nivel un programa que calcule el cuadrado de un número positivo *a* mediante la suma acumulada *a* veces. Tradúcelo a lenguaje ensamblador de la MR y posteriormente a lenguaje máquina, suponiendo que se almacena a partir de la dirección 00h.

3.39 El siguiente algoritmo descrito en un lenguaje de alto nivel calcula el factorial de un número *a*. Tradúcelo a lenguaje ensamblador de la MR y posteriormente a lenguaje máquina, suponiendo que se almacena a partir de la dirección 00h. Indica el contenido de la tabla de símbolos.

Programa FACTORIAL;

var n, ind1, ind2, asumar, acum, fact: ***entero***;

n:= 9;

ind1:= n;

asumar := n;

acum:= 0;

mientras (ind1 > 2) ***hacer***

ind2 := ind1 - 1;

mientras (ind2 > 0) ***hacer***

acum := acum + asumar;

ind2 := ind2 - 1;

fmientras;

asumar := acum;

acum := 0;

ind1 := ind1 - 1;

fmientras;

fact := asumar;

fprograma.

3.40 Escribe en un lenguaje de alto nivel un programa que calcule el número de letras "*a*" de una frase terminada en punto. Traduce el programa a lenguaje ensamblador y lenguaje máquina de la MR, suponiendo que:

- El programa se encuentra almacenado a partir de la dirección 80h.
- La frase se encuentra almacenada a partir de la dirección 02h en posiciones consecutivas de memoria.
- Cada letra ocupa una posición de memoria y está codificada utilizando el código ASCII. En este sistema de codificación, cada símbolo se codifica con 7 bits; en este caso, en los 7 bits de menor peso de la posición de memoria que ocupa. Los 9 bits de mayor peso están a 0. La letra "*a*" se codifica mediante el número 61h, y el carácter "." mediante el número 2Eh.
- Los valores 61h y 2Eh se pueden almacenar en las direcciones 00h y 01h. La frase nunca tiene más de 125 letras, y siempre termina en ".".

3.41 Escribe en un lenguaje de alto nivel un programa que calcule la suma de los elementos de un vector al mismo tiempo que busca los elementos máximo y mínimo. Tradúcelo a lenguaje ensamblador de la MR y posteriormente a lenguaje máquina, suponiendo que los datos se almacenan a partir de la dirección 00h y las instrucciones a partir de la dirección 3Ah.

3.42 Escribe en un lenguaje de alto nivel un programa que sume, elemento a elemento, dos vectores A y B, dejando el resultado en un tercer vector C (C[i] := A[i] + B[i]). El tamaño de los vectores es de 10 elementos. Traduce el programa a lenguaje ensamblador de la MR y posteriormente a lenguaje máquina, suponiendo que los vectores se almacenan a partir de la dirección de memoria 00h y el programa se almacena a partir de la dirección 80h.

3.43 Escribe en lenguaje ensamblador de la MR las macros que permitan ejecutar las siguientes instrucciones, pertenecientes a los juegos de instrucciones de otros computadores:

a) clr Rd	Pone un 0 en el registro destino
b) clr A(Ri)	Pone un 0 en la posición de memoria M[A+(Ri)]
c) inc Rd	Incrementa en una unidad el valor del registro Rd
d) dec Rd	Decrementa en una unidad el valor del registro Rd
e) add3 Rf1, Rf2, Rd	Suma el contenido de 3 registros del Banco de Registros y deja el resultado en uno de ellos (Rd)
f) asl Rd	Desplaza un bit a la izquierda el registro Rd

g)	mov Rf, Rd	Copia el valor del registro Rf en el registro Rd
h)	mov Rf, A(Ri)	Copia el valor del registro Rf en la posición de memoria M[A+(Ri)]
i)	mov A(Ri), B(Rj)	Copia el valor de la posición de memoria M[A+(Ri)] en la posición de memoria M[B+(Rj)]
j)	swap Rf1, Rf2	Intercambia los valores de los registros Rf1 y Rf2
k)	swap A(Ri), B(Rj)	Intercambia los valores de las posiciones de memoria M[A+(Ri)] y M[B+(Rj)]
l)	asrn #n, Rd	Desplaza n bits a la derecha el registro Rd. El número n ha de ser mayor que 0 y menor que 16
m)	asln #n, Rd	Desplaza n bits a la izquierda el registro Rd. El número n ha de ser mayor que 0 y menor que 16
n)	inc A(Ri)	Incrementa el contenido de la posición de memoria M[A+(Ri)]

3.44 Escribe en un lenguaje de alto nivel un programa que calcule la suma de los elementos de un vector V que cumplan ser mayores que un cierto valor MÍNIMO y menores que un cierto valor MÁXIMO (MÍNIMO < V[i] < MÁXIMO). Tradúcelo a lenguaje ensamblador de la MR y posteriormente a lenguaje máquina, suponiendo que el programa se almacena a partir de la dirección 3Ah.

3.45 Traduce a lenguaje ensamblador de la MR el siguiente programa escrito en un lenguaje de alto nivel. El programa calcula el factorial de un número "a" (en el ejemplo, a=8):

programa FACTORIAL;

    var a, cont, factorial: ***entero***;

    a:= 8;

    cont := 1;

    factorial := 1;

    mientras (cont < A) ***hacer***

        cont := cont + 1;

        factorial := factorial * cont;

    fmientras;

    fprograma.

a) Diseña y usa la macro *mul $1, $2, $3*, que multiplica el contenido de dos registros ($1, $2) y deja el resultado en un tercer registro ($3). Los tres registros deben ser diferentes.

b) Haz el preensamblado (expansión de macros) y el postensamblado (creación de la tabla de símbolos y generación de código) del programa, suponiendo que el programa se almacena a partir de la dirección de memoria 00h.

3.46 Dado el siguiente programa escrito en lenguaje ensamblador de la MR:

```
N:          .DW 4
            RESUL:      .RW 1
                .DEF cmp $1, $2
                        SUB $1, $2, R0
                .ENDDEF
                .DEF cmp $1, $i2
                        SUBI $1, $i2, R0
                .ENDDEF
                .DEF clr $1
                        ADD R0, R0, $1
                .ENDDEF
                .DEF mov $1, $2
                        ADD R0, $1, $2
                        .ENDDEF
                        .DEF mov $i1, $2
                                ADDI R0, $i1, $2
                        .ENDDEF
                        .DEF mov $d1, $2
                                LOAD $d1, $2
                        .ENDDEF
                        .DEF mov $1, $d2
                                STORE $1, $d2
                        .ENDDEF
                        .DEF inc $1
                                ADDI $1, #1. $1
                        .ENDDEF
                        .DEF dec $1
                                SUBI $1, #1. $1
                        .ENDDEF
                        .BEGIN INI
INI:        MOV N(R0), R1
            MOV R1, R3
                        clr R4
                M1:     cmp R1, #2
                        BLE FM1
                        SUBI R1, #1, R2
                M2:     cmp R2, #0
                        BLE FM2
                        ADD R4, R3, R4
                        dec R2
```

```
        BR M2
FM2:    mov R4, R3
        clr R4
        dec R1
        BR M1
FM1:    mov R3, RESUL(R0)
        .END
```

a) Realiza el preensamblado (expansión de macros).

b) Realiza el postensamblado (generación de la tabla de símbolos y traducción a lenguaje máquina) suponiendo que el programa se almacena a partir de la posición 00h de memoria.

c) Traduce el programa a alto nivel, indicando qué operación realiza. Para ello, intenta analizar qué operación realiza cada macro.