Solució Problema 1:

Efectuar les següents operacions aritmètiques **indicant clarament tots els càlculs realitzats**.

a) Escriure en binari en *complement a dos* (Ca2) els següents números fent servir el menor nombre de bits. Indica el nombre de bits necessari per cada apartat. (objectiu 5.4).

a1.1) Per passar directament el número **5C9** en base 15 a base binària podem anar dividint per 2 successivament tenint en compte que tots els valors de les divisions estan representats en base 15.
El primer que faríem (és útil però no necessari) és representar com seria la taula de multiplicar per 2 en base 15 que ens servirà per poder fer les divisions fàcilment i per després fer-les.

base 15	
2·0	0
2·1	2
2·2	4
2·3	6
2·4	8
2·5	A
2·6	C
2·7	E
2·8	11
2·9	13
2·A	15
2·B	17
2·C	19
2·D	1B
2·E	1D

```
5C9 |_2__
1C   2DC |_2__
 19  0D   16D |_2__
  0   1C   1D  AE |_2__
       1    0  0E  57 |_2__
                0  17  2B |_2__
                    0  0B  15 |_2__
                        1   0  A |_2__
                               0  5 |_2__
                                  1  2 |_2__
                                     0  1
```

$5C9_{15}= 10100100010_2$ però com es tracta d'un número positiu en *complement a dos* i el valor obtingut comença per **1** és precís afegir un bit de signe addicional . Aquest bit ha de ser **0** per indicar de que es tracta d'un número positiu. Per tant per poder representar aquest número necessitem 12 bits.

$$\mathbf{5C9_{15}= 010100100010_2}$$

a1.2) Si no sabem passar directament de base 15 a binari en complement a 2, podem primer passar el número $5C9_{15}$ a base decimal i després d'aquesta a binari mitjançant divisions successives per 2.

Per passar a decimal fem el sumatori corresponent

$$V=\sum_{i=0}^{2} x_i \cdot 15^i = 5\cdot 15^2 + 12\cdot 15^1 + 9\cdot 15^0 = 5\cdot 225 + 12\cdot 15 + 9\cdot 1 = 1125 + 180 + 9 = 1314$$

$5C9_{15}= 1314_{10}$ ara només falta passar a binari en complement a dos fent divisions per dos successivament.

```
1314 |_2__
   0 657 |_2__
       1 328 |_2__
           0 164 |_2__
               0  82 |_2__
                   0 41 |_2__
                      1 20|_2__
                         0 10|_2__
                            0 5|_2__
                              1 2|_2__
                                0 1
```

$1314_{10}= 10100100010_2$ però com es tracta d'un número positiu en *complement a dos* i el valor obtingut comença per **1** és precís afegir un bit de signe addicional . Aquest bit ha de ser **0** per indicar de que es tracta d'un número positiu. Per tant per poder representar aquest número necessitem 12 bits.

$$\mathbf{5C9_6= 1314_{10}= 010100100010_2}$$

a2.1) Aquest problema es resolt de manera anàloga a l'anterior. Per passar directament el número **-345** en base 6 a base binària podem anar dividint per 2 successivament tenint en compte que tots els valors de les divisions estan representats en base 6.
Així, calcularem quant val en binari en *complement a dos* el valor positiu del nombre que volem calcular i després fer un canvi de signe.
El primer que faríem (és útil però no necessari) és representar com seria la taula de multiplicar per 2 en base 6 que ens servirà per poder fer les divisions fàcilment i després fer-les.

```
base 6          345 |2
2·0|  0         14   152 |2
2·1|  2          05   12  54 |2
2·2|  4           1    0  14  25 |2
2·3| 10                    0  05  12 |2
2·4| 12                        1   0  4 |2
2·5| 14                               0  2|2
                                         0 1
```

$345_6 = 10001001_2$ però com es tracta d'un número positiu en *complement a dos* i el valor obtingut comença per **1** es precís afegir un bit de signe addicional . Aquest bit ha de ser **0** per indicar de que es tracta d'un número positiu. Per tant per poder representar aquest número necessitem 9 bits.

$$\mathbf{345_6 = 010001001_2}$$

El següent pas és fer el canvi de signe (canviarem els zeros per uns i els uns per zeros, i sumar 1).

```
345₆ =   010001001
         101110110   ← Canvi de bits
       +         1   ← Sumem 1
         101110111
```

Per tant $\mathbf{-345_6 = 101110111_2}$

a2.2) Si no sabem passar directament de base 6 a binari en complement a 2, podem primer passar el número 345_6 a base decimal i després d'aquesta a binari mitjançant divisions successives per 2.

Per passar a decimal fem el sumatori corresponent

$$V = -\sum_{i=0}^{2} x_i \cdot 6^i = 3\cdot 6^2 + 4\cdot 6^1 + 5\cdot 6^0 = -(3\cdot 36 + 4\cdot 6 + 5\cdot 1) = -(108+24+5) = -137$$

$-345_6 = -137_{10}$ ara només falta passar a binari en complement a dos. Per passar un número decimal negatiu a binari en *complement a dos* fem servir les següents transformacions:

Z → N → B		Si $x \geq 0 \Rightarrow x_e = x$
x → x_e → $x_{n-1} \ldots x_0$		Si $x < 0 \Rightarrow x_e = x + 2^n$

Passem el número enter -137_{10} al numero natural corresponent fent servir la transformació anterior.
$x=-137$
Com que x és inferior a 0, el número natural que li correspon es el $x_e = -137+2^n$.

Per saber quin valor ha de prendre n hem de saber quin es el nombre mínim de bits amb els que podem representar el valor -137_{10}. Hem de calcular el rang.

Rang enters: $-2^{n-1} \leq x \leq 2^{n-1}-1$

Per $n=8$, el rang es $-128 \leq x \leq 127$
Per $n=9$, el rang es $-256 \leq x \leq 255$

Podem observar que amb 8 bits el valor -137 no és representable i necessitarem 9 bits. Per tant $n=9$.

Aleshores $x_e = -137+2^9 = -137+512 = 375$

Ara ja podem passar el número 375_{10} a binari mitjançant divisions successives per 2 i obtindrem el número que buscàvem.

```
375 |_2_
  1 187 |_2_
      1  93 |_2_
          1  46 |_2_
              0  23 |_2_
                  1 11 |_2_
                     1  5|_2_
                        1 2|_2_
                          0 1
```

$375_{10} = 101110111_2$

Per tant $\mathbf{-345_6 = -137_{10} = 101110111_2}$

b) Defineix breument com es pot detectar un *overflow* en una suma en *Complement a 2* (Ca2) observant els nombres a sumar i el resultat. (objectiu 5)

Quan el resultat d'una operació de suma és incorrecte es diu que s'ha produït un sobreeiximent o *overflow*.

Per determinar si el resultat de la operació és correcte a partir de la representació es pot fer servir la *regla dels signes* : si els dos operands són del mateix signe (observant els bits de més pes) i el resultat és de signe contrari, l'operació és incorrecta i per tant es produeix *overflow*.

També es pot fer servir la *regla dels carrys*: si la XOR de l'últim *carry* amb el *carry* de sortida dóna 1, l'operació és incorrecta. Observeu que el *carry* de sortida només es calcula per poder utilitzar la regla dels *carrys*. En *Ca2*, el *carry* de sortida no té cap sentit.

c) Representa en binari en *Complement a 2* (Ca2) els nombres implicats en les següents operacions (justificant com ho has fet). Realitza les operacions treballant amb les representacions en Ca2, amb el nombre de bits que se senyala. Indica també si algun dels nombres no és representable. Indica en cada cas en decimal el valor del resultat obtingut amb el nombre de bits senyalat i si es produeix *overflow* o no. (objectiu 5.2 i 5.5)

c1) (-4) + (-1), en 3 bits.

La representació en Ca2, en 3 bits, de –4 és 100_2.
La representació en Ca2, en 3 bits, de –1 és 111_2.

Fem l'operació corresponent

```
   100
 + 111
  1011
```

Per determinar si el resultat de l'operació és correcte fem servir la *regla dels signes*. En aquest cas se sumen 2 números negatius i el resultat és positiu, per tant el resultat és incorrecte i es produeix *overflow*.

El valor decimal del resultat obtingut en 3 bits és $011_2 = 3_{10}$

c2) 3 + 8, en 4 bits.

La representació en Ca2, en 4 bits, de 3 és 0011_2.
El valor 8 no es pot representar en 4 bits en Ca2. El rang de valors representables per a enters és $-2^{n-1} \leq x \leq 2^{n-1}-1$.
Per tant en 4 bits el valor que podem representar són $-8 \leq x \leq 7$.

En aquesta operació no té sentit parlar de resultat ni de *overflow*.

c3) 6 – (-4), en 4 bits.

La representació en Ca2, en 4 bits, de –6 és 0110_2.
La representació en Ca2, en 4 bits, de –4 és 1100_2.

Fem l'operació corresponent restant directament.

```
   0110
 - 1100
  11010
```

Si no sabem restar directament podem fer l'operació amb una suma canviant el segon operand de signe (cal recordar que si ho fem amb aquest mètode el *carry* de la suma és incorrecte i és sempre el contrari al que obtindríem si fèssim la resta directament).
La representació en Ca2, en 4 bits, de 4 (canvi de signe del –4) és 0100_2.
Fem l'operació de suma.

```
   0110
 + 0100
  01010
```

Invertim el valor del *carry* i obtenim que el resultat de la operació és 1010_2 i el *carry* de sortida es 1.

Per determinar si el resultat de l'operació és correcte tornem a fer servir la *regla dels signes*. En aquest cas es sumen 2 números positius i el resultat és negatiu, per tant el resultat és incorrecte i es produeix *overflow*.

El valor decimal del resultat obtingut en 4 bits és $1010_2 = -6_{10}$

c4) (-3) – (-5),en 5 bits

La representació en Ca2, en 5 bits, de –3 és 11101_2.
La representació en Ca2, en 5 bits, de –5 és 11011_2.

Fem l'operació corresponent restant directament.

```
   11101
 - 11011
  000010
```

Si no sabem restar directament podem fer l'operació amb una suma canviant el segon operand de signe.
La representació en Ca2, en 5 bits, de 5 (canvi de signe del –5) és 00101_2.
Fem l'operació de suma.

```
   11101
 + 00101
  100010
```

Invertim el valor del *carry* i obtenim que el resultat de la operació és 00010_2 i el *carry* de sortida es 0.

Per determinar si el resultat de l'operació és correcte tornem a fer servir la *regla dels signes*. En aquest cas es sumen 2 números de diferent signe i per tant el resultat sempre és correcte i no es produeix *overflow*.

El valor decimal del resultat obtingut en 5 bits és $00010_2 = 2_{10}$

Solució Problema 2:

a) Es tracta d'un sistema combinacional amb quatre entrades i una sortida:
La acció segura de l'enunciat suposa que no es donarà mai la situació D=0 i A=1.
per altra banda, la sortida ha de valer "1" pels següents casos:

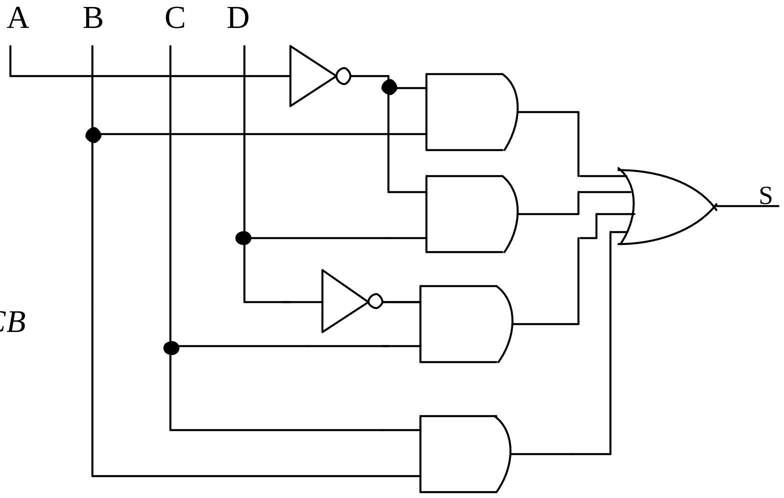
1. A=0 i B=1
2. C=1 i A=0
3. D=1 i A=0
4. B=1, C=1 i A=1

Entrades				**sortida**
A	B	C	D	S
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	X
1	0	0	1	0
1	0	1	0	X
1	0	1	1	0
1	1	0	0	X
1	1	0	1	0
1	1	1	0	X
1	1	1	1	1

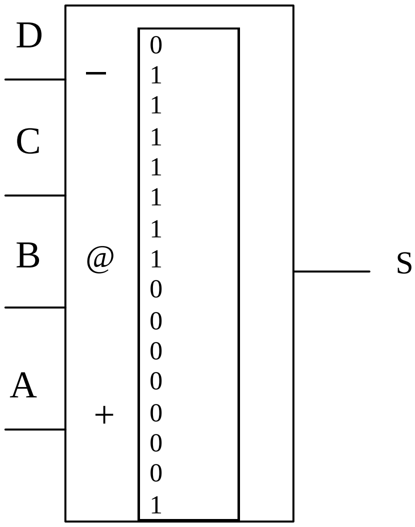
b)

AB / CD	00	01	11	10
00		1	X	X
01	1	1		
11	1	1	1	
10	1	1	X	X

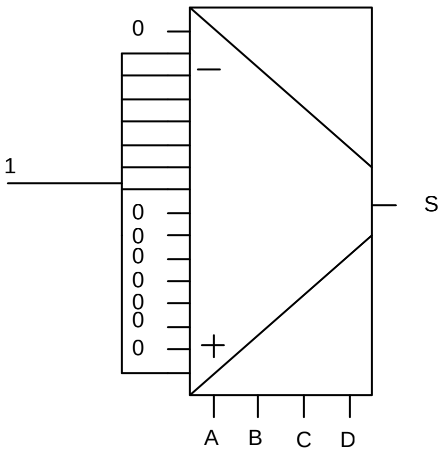
$$S = \overline{A}B + \overline{A}D + C\overline{D} + CB$$



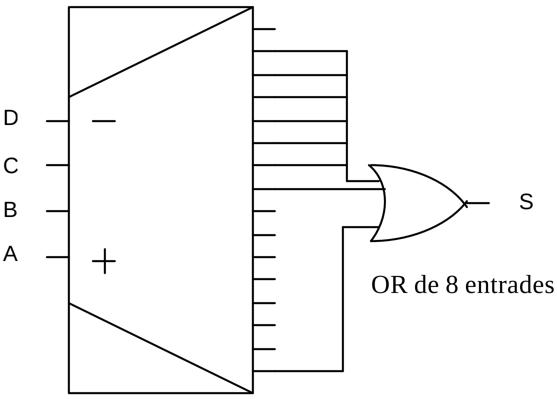
c) la ROM ha de tenir 4 entrades d'adreces i una sortida. Per tant, es necessita una memòria de 16 posicions d'un bit cadascuna, segons el dibuix següent.
Donat que en la memòria no es poden escriure "X", caldrà posar "0" o "1" indistintament, doncs mai s'accedirà a aquestes posicions de memòria.



d) Les quatre entrades de selecció correspondran a les quatre entrades del sistema combinacional ABCD i a les entrades de dades del multiplexor caldrà posar "1" o "0" depenent del valor de la taula de veritat del sistema combinacional. Igual que en l'apartat anterior, en aquells casos on la taula de veritat val "X" es pot posar "1" o "0" indistintament, doncs aquesta combinació mai serà seleccionada.



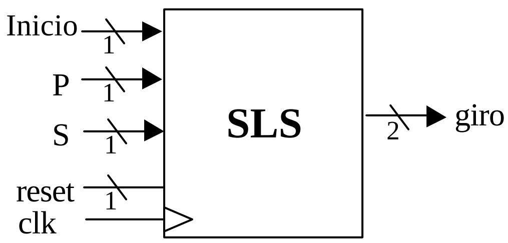
e) Cal connectar les entrades del decodificador a les entrades del sistema combinacional, i introduir en una porta OR de 8 entrades. Totes aquelles sortides del decodificador corresponents a l'activació de la sortida del sist. comb.



Solució Problema 3:

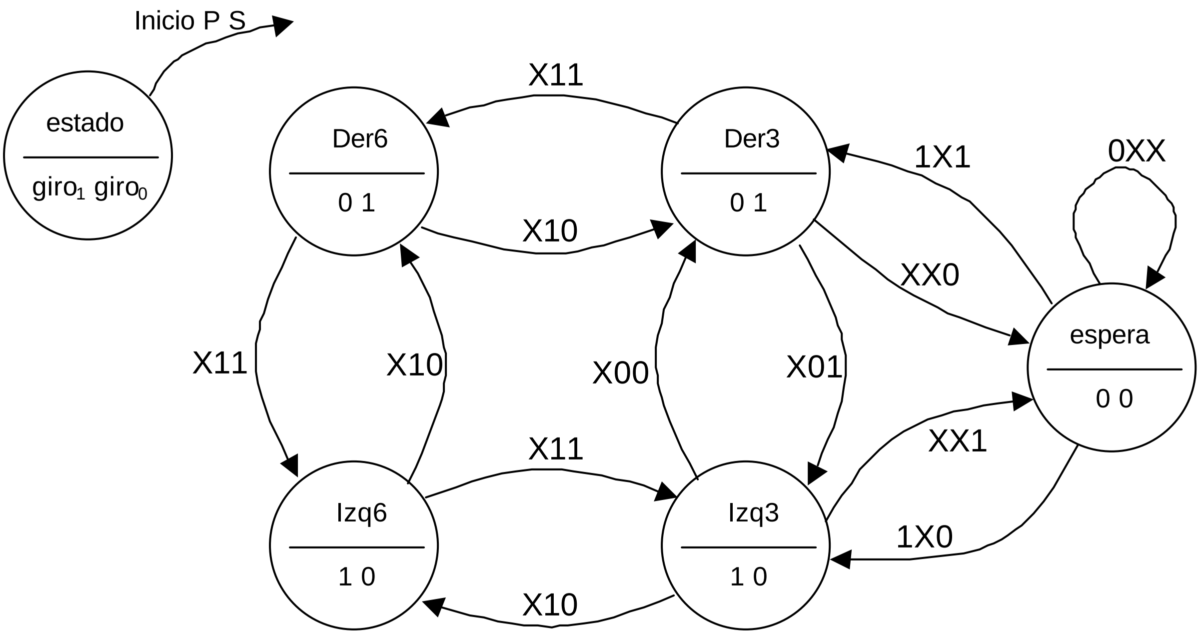
a) El tiempo de ciclo puede ser de 1 segundo, de esta manera el agujero de 3 cm se puede realizar en un estado, y el de 6 cm en dos estados.

b) Esquema gráfico y la codificación del SLS cuyo grafo de estados se muestra en el apartado **c)** es:



movimiento de la broca	$giro_1$	$giro_0$
Parada	0	0
A derecha	0	1
A izquierda	1	0

c) Grafo de estados del SLS que controla el movimiento de la broca.



Solució Problema 4:

Codificación de estados

estado	Q_1	Q_0
A	0	0
B	1	1
C	1	0
D	0	1

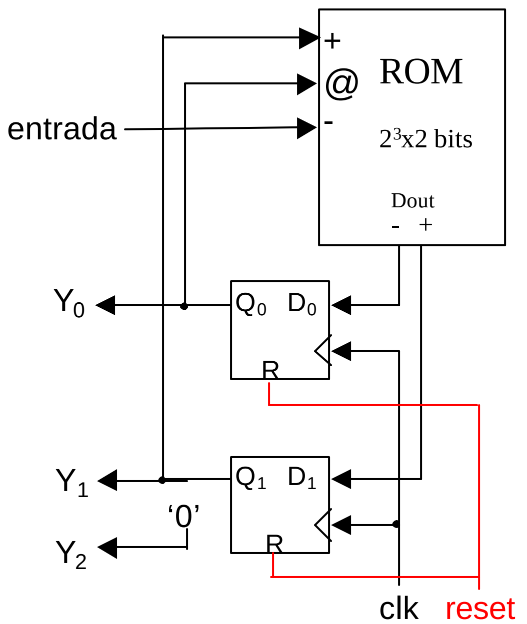
Tabla de transacción (contenido de la ROM)

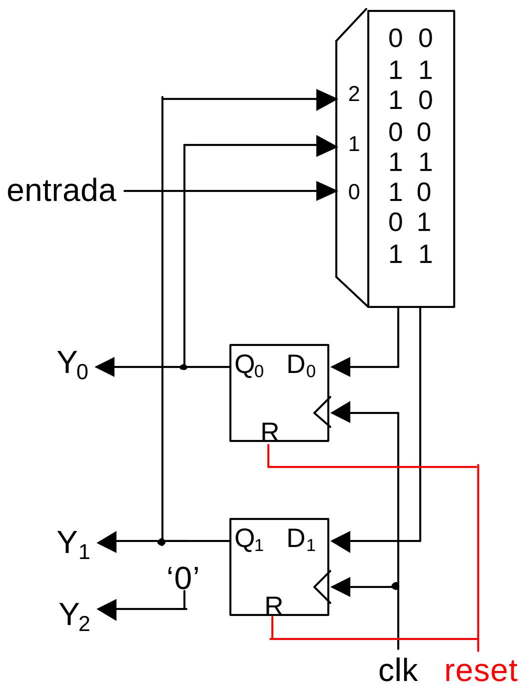
Q_1	Q_0	Entrada	Q_1^+	Q_0^+
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

$Q_1^+=D_1$
$Q_0^+=D_0$

Tabla de salidas

Q_1	Q_0	Y_2	Y_1	Y_0
0	0	0	0	0
0	1	0	0	1
1	0	0	1	0
1	1	0	1	1





La señal de reset se puede poner para inicializar el sistema.