

SOLUCIONES

P1 (1,5 pts).- Dada la tabla de estados adjunta, se pide, justificando la respuesta:

a) Señalar si se trata de un circuito de Moore o de Mealy. **b)** Implementar la ecuación de salida Z_1 , utilizando un multiplexor 4-1 y el mínimo número de puertas lógicas posible y **c)** Implementar las ecuación de salida Z_0 utilizando un decodificador de 4-16 con las salidas activas en bajo.

Nota: Utilizar los circuitos facilitados con las entradas señaladas. Indicar necesariamente los valores de las entradas y de las salidas en cada caso utilizadas.

Estado actual $Q_1^n Q_0^n$	Estado siguiente/salida ($Q_1^{n+1} Q_0^{n+1}/Z_1 Z_0$)			
	$X_1 X_0 = 00$	$X_1 X_0 = 01$	$X_1 X_0 = 10$	$X_1 X_0 = 11$
00	01/00	00/10	01/10	00/01
01	10/00	00/00	10/00	00/00
10	10/01	11/11	10/01	11/10
11	01/10	00/10	00/10	00/10

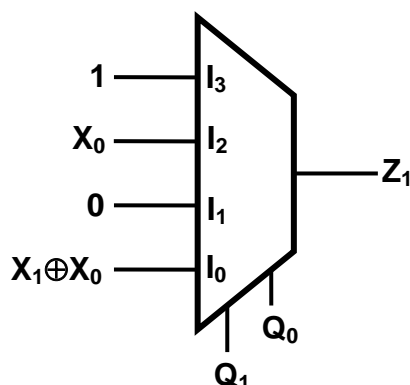
RESPUESTA:

a) La tabla representa una máquina de estados de**MEALY**.....

Justificación: **Las salidas son función del estado y de las entradas como se observa para el caso de los estados 00 y 10.**

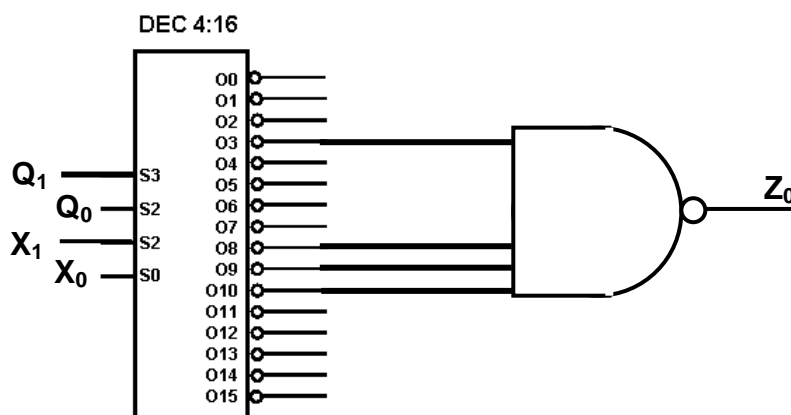
b) Implementar la salida Z_1 utilizando un multiplexor 4:1 y el mínimo número de puertas lógicas posible.

$$Z_1 = /Q_1/Q_0 X_1/X_0 + Q_1 Q_0/X_0 + Q_1 Q_0 X_1 + Q_1/Q_0 X_0$$

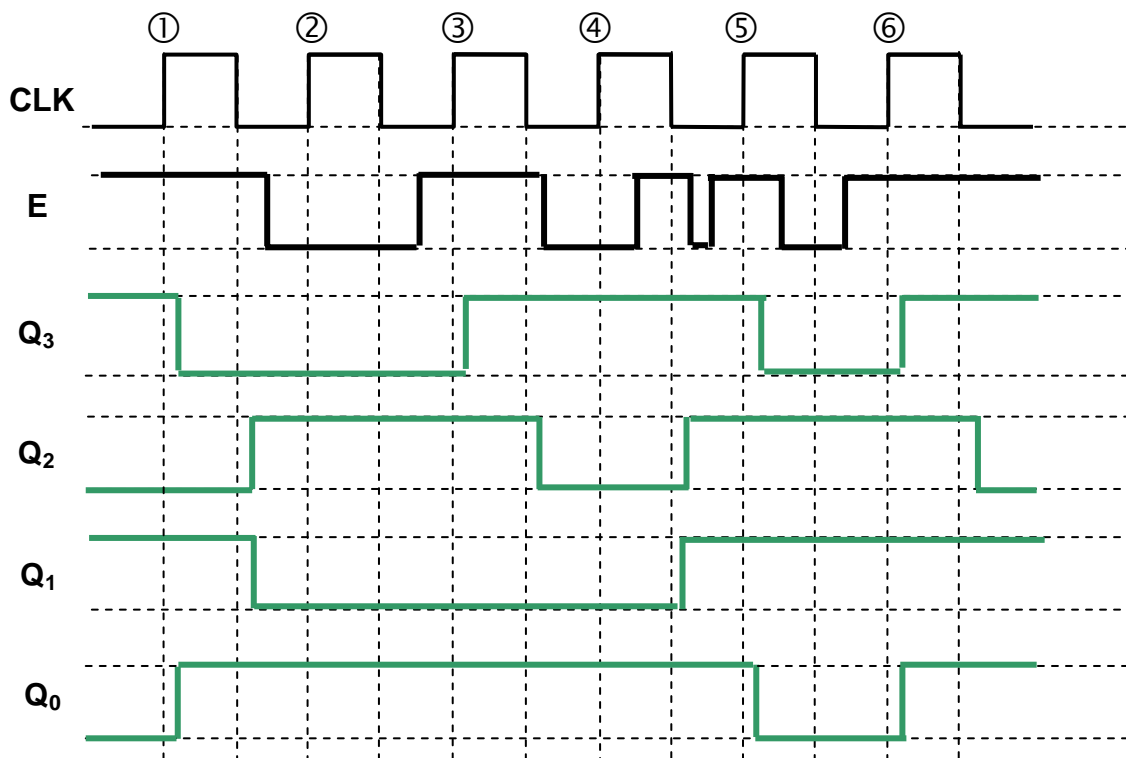
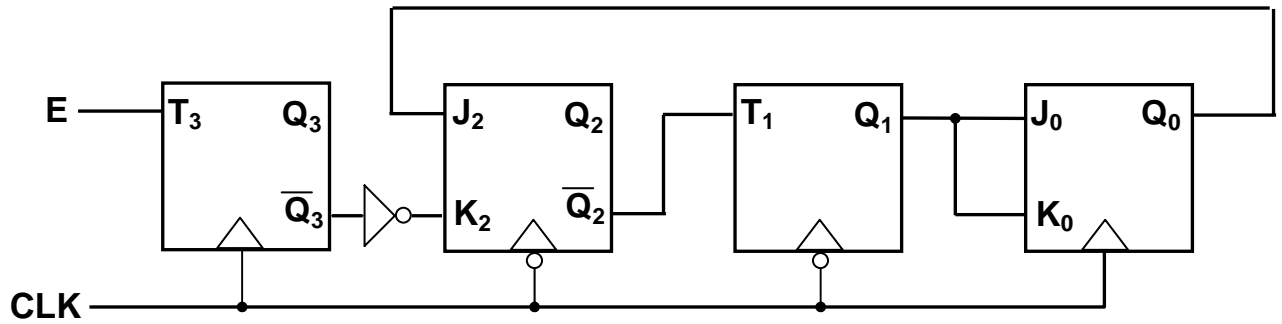


c) Implementar la salida Z_0 utilizando un decodificador 4:16 con las salidas negadas.

$$Z_0 = /Q_1/Q_0 X_1 X_0 + Q_1/Q_0 /X_1/X_0 + Q_1/Q_0 /X_1 X_0 + Q_1/Q_0 X_1/X_0 = \Sigma m(3,8,9,10)$$



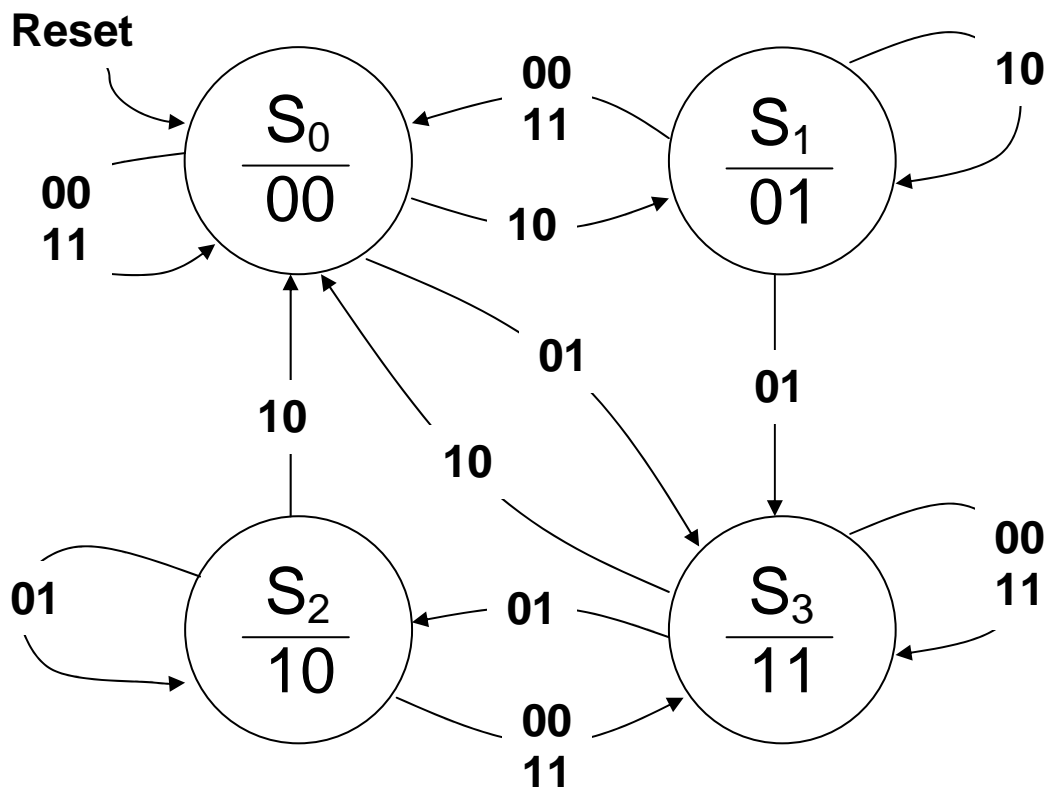
P2 (1 pto).- Se pide completar el cronograma de la figura adjunta, sabiendo que en el estado inicial los valores de los biestables son: $Q_3 = Q_1 = 1$ y $Q_2 = Q_0 = 0$



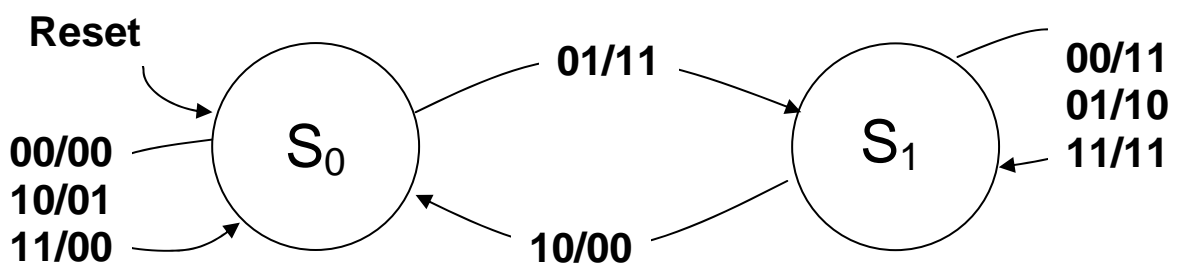
P3 (2 ptos).- Se quiere diseñar un circuito secuencial que funcione como un restador serie de n bits. Partiendo de una señal de Reset asíncrono que inicializa toda la lógica secuencial a "0", el circuito escribe en serie (síncronamente con el reloj) los bits de los dos operandos A_i (minuyendo) y B_i (sustraendo) por sus dos entradas X_2 y X_1 respectivamente. El circuito debe suministrar tanto el bit de la resta R_i como el de acarreo C_{i+1} por sus dos salidas Z_1 y Z_2 respectivamente. Se supone que la resta continua indefinidamente hasta que se recibe un nuevo Reset por la entrada asíncrona, momento en el que la introducción de nuevos operandos se entiende como el comienzo de una nueva resta.

Se pide diseñar el diagrama de estados del circuito en sus dos versiones Moore y Mealy, utilizando los estados necesarios de los diagramas adjuntos, añadiendo más si fuera necesario o dejando en blanco los que no se necesiten.

Moore



Mealy



ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES I. FINAL 11-6-07

P4 (2 ptos).- Un sistema con un bus de direcciones de 20 bits, dispone de un mapa de memoria de cinco elementos tal y como indica la tabla adjunta. Se pide:

- Completar la tabla con los valores adecuados. Se conoce que el dispositivo MEM4, ocupa las posiciones de memoria consecutivas a MEM1 y que MEM5 puede ocupar zonas distintas (**Nota:** Se considerará válida cualquiera de las zonas)
- Calcular las ecuaciones de selección mínimas para las cinco pastillas.

RESPUESTA:

a)

Pastilla	Bit selección	Capacidad	Dirección inicial	Dirección final
MEM1	/CS1	128 kbytes	20000 ₁₆	3FFFF ₁₆
MEM2	/CS2	64 kbytes	90000 ₁₆	9FFFF ₁₆
MEM3	/CS3	32 kbytes	E0000 ₁₆	E7FFF ₁₆
MEM4	/CS4	256 kbytes	40000 ₁₆	7FFFF ₁₆
MEM5	/CS5	128 kbytes	00000 ₁₆ A0000 ₁₆ C0000 ₁₆	1FFFF ₁₆ BFFFF ₁₆ DFFFF ₁₆

b)

$$/CS1 = A_{19} + A_{18} + /A_{17}$$

$$/CS2 = /A_{19} + A_{18}$$

$$/CS3 = /A_{19} + /A_{18}$$

$$/CS4 = A_{19} + /A_{18}$$

$$/CS5 = A_{19} + A_{18} + A_{17}$$

$$/CS1 = A_{19} + A_{18}$$

$$/CS2 = /A_{19} + A_{18} + A_{17}$$

$$/CS3 = /A_{19} + /A_{18}$$

$$/CS4 = A_{19} + /A_{18}$$

$$/CS5 = /A_{19} + A_{18} + /A_{17}$$

$$/CS1 = A_{19} + A_{18}$$

$$/CS2 = /A_{19} + A_{18}$$

$$/CS3 = /A_{19} + /A_{18} + /A_{17}$$

$$/CS4 = A_{19} + /A_{18}$$

$$/CS5 = /A_{19} + /A_{18} + A_{17}$$

ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES I. FINAL 11-6-07

P5 (2 ptos).- A partir del código en ensamblador que se expone más abajo responder a las siguientes preguntas:

- ¿Cuál es el contenido de la pila justo **antes** de la ejecución de la instrucción **call func**.
- ¿Cuál es el contenido de la pila justo **antes** de la ejecución de la instrucción **jmp %15+4, %r0**.
- ¿Cuál es el valor del contador de programa (%pc) justo **después** de la ejecución de la instrucción **jmp %15+4, %r0**.

Consideraciones:

- El puntero de pila (%sp = %r14) se encuentra inicializado a la dirección 0xF8000000.
- El contenido de la pila se encuentra inicializado a 0x00000000.
- Responder en los recuadros habilitados para tal efecto en el reverso de la hoja. Indicar **claramente** la posición del puntero de pila en cada momento. Rellenar el contenido de **todas** las posiciones de memoria indicadas en los recuadros (véase como ejemplo la tabla-1).
- El valor de la constante MASK es distinto en cada examen, y se haya como las cuatro últimas cifras del DNI del alumno expresado en base 16. Por ejemplo, si el DNI fuera 123.456.789, el valor de la constante MASK sería 0x6789.

```
mask      .equ 0x6789      ! ultimas 4 cifras del DNI
buffer    .equ 3000        ! Direccion de comienzo del buffer
          .begin
          .org 2048
          sethi mask, %r1   ! mask -> (%r1 <= mask & 10x0`s)
          srl %r1, 10, %r1  ! %r1 >> 10
          addcc %sp, -4, %sp ! (%sp) - 4 -> (%sp)
          st %r1, %sp       ! (%r1) -> M[(%sp)]
          ld [address], %r2 ! M[address] -> %r2
          addcc %sp, -4, %sp ! (%sp) - 4 -> (%sp)
          st %r2, %sp       ! (%r2) -> M[(%sp)]
          call func         ! Llamada a subrutina
          ld %sp, %r1       ! M[(%sp)] -> (%r1)
          addcc %sp, 4, %sp  ! (%sp) + 4 -> (%sp)

address: buffer
          .org buffer
          0x0000FFFF
          0x00004444
          0x00001234
          0x00007866
          0x00000000

func:     ld %sp, %r3       ! M[(%sp)] -> (%r3)
          addcc %sp, 4, %sp ! (%sp) + 4 -> (%sp)
          ld %sp, %r4       ! M[(%sp)] -> (%r4)
          st %r15, %sp      ! (%r15) -> M[(%sp)]
          addcc %sp, 0, %r5 ! (%sp) -> (%r5)
          addcc %sp, -12, %sp ! (%sp) - 12 -> (%sp)

test:     ld %r3, %r1       ! M[(%r3)] -> (%r1)
          andcc %r1, %r1, %r0 ! (%r1) and (%r1)
          be done
          xorcc %r1, %r4, %r1 ! (%r1) xor (%r4) -> (%r1)
          st %r1, %sp       ! (%r1) -> M[(%sp)]
          addcc %sp, 4, %sp ! (%sp) + 4 -> (%sp)
          addcc %r3, 4, %r3 ! (%r3) + 4 -> (%r3)
          ba test

done:     ld %r5, %r15      ! M[(%r5)] -> (%r15)
          jmp %r15 + 4, %r0 ! retorno de la subrutina
          .end
```

Ejemplo Tabla-1: Contenido de la pila **antes** de comenzar la ejecución del programa:

0xF7FFFFF0	0x00000000
0xF7FFFFF4	0x00000000
0xF7FFFFF8	0x00000000
0xF7FFFFFC	0x00000000

%r14 → 0xF8000000 0x00000000

Respuesta apartado a)

	0xF7FFFFFF0	0x00000000
	0xF7FFFFFF4	0x00000000
%r14 →	0xF7FFFFFF8	0x00000BB8
	0xF7FFFFFFC	0x00006789
	0xF8000000	0x00000000

Respuesta apartado b)

	0xF7FFFFFF0	0x00009876
	0xF7FFFFFF4	0x000023CD
	0xF7FFFFFF8	0x000075BD
	0xF7FFFFFFC	0x00001FEF
%r14 →	0xF8000000	0x00000000

Respuesta apartado c)

%PC 0x00001FF3

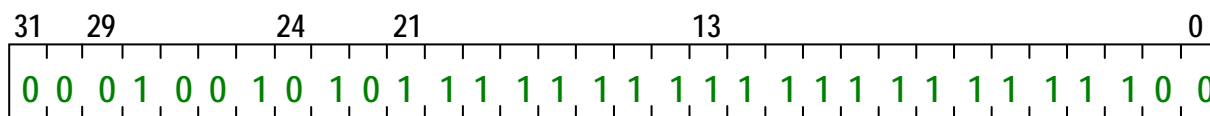
ESTRUCTURA Y TECNOLOGÍA DE COMPUTADORES I. FINAL 11-6-07

P6 (1,5 ptos).- Se quiere añadir la instrucción **BNE** (Branch on Not Equal) al juego de instrucciones del ARC. Esta instrucción permite comprobar si el resultado de la última operación es distinto de cero. De esta forma, se realizará un salto siempre y cuando la bandera Z = 0.

A continuación se muestra cómo quedaría el nuevo juego de instrucciones:

op	Tipo	op2	Instr.	[rd] = cc	Instr.	op3	Instr.
00	sethi/branch	010	branch	00001	be	000000	ld
01	call	100	sethi	00101	bcs	000100	st
10	ALU			00110	bneg	010000	addcc
11	Memoria			00111	bvs	010001	andcc
				01000	ba	010010	orcc
				01001	bne	010110	orncc
						100110	srl
						111000	jmpl

a) Hallar la codificación de la instrucción **bne** para que realice un salto que **retrase cuatro instrucciones**.



b) Se pide completar la zona punteada con el pseudocódigo RTL, como el empleado en clase de teoría, para las últimas tres fases del ciclo de ejecución que sirva para describir la nueva instrucción **bne**.

IF {ir[31:30] = 00} (sethi, be, bcs, bneg, bvs, ba y bne):

② IF (ir[24:22]=010) THEN

```
IF {(ir[28]=...1. and ir[25] =. 0 ..(cond=1)} THEN [rt1] <= ext_sig(ir[21:0]x4)
```

③- - - - -; En esta fase no se hace nada

④ IF (ir[24:22]=010) THEN

```
IF {(ir[28]=1 and ir[25]=0) or (cond=1)} THEN ...[pc]... <= [pc] + [rt1]
```

```
ELSE .....[pc] <= [pc] + 4 .....
```

ELSIF (ir[24:22]= ...100...) THEN

```
[RD] <= ir[21:0]&(0000000000), [pc] <= [pc] + 4
```