Examen de Arquitectura de Computadoras

29 de julio de 2013

Instrucciones:

- Indique su nombre, apellido y número de cédula en todas las hojas que entregue.
- Escriba las hojas de un solo lado.
- Las hojas deben estar numeradas y en la primer hoja debe escribirse el total.
- Empiece cada ejercicio en una hoja nueva.
- No puede utilizar material ni calculadora.
- **Apague su celular.**
- La duración del examen es de tres horas. En dicho tiempo debe también completar sus datos. Solo se contestarán dudas de letra. No se aceptan preguntas en los últimos 30 minutos del examen.
- Para aprobar debe contestar correctamente un ejercicio entero y dos preguntas.

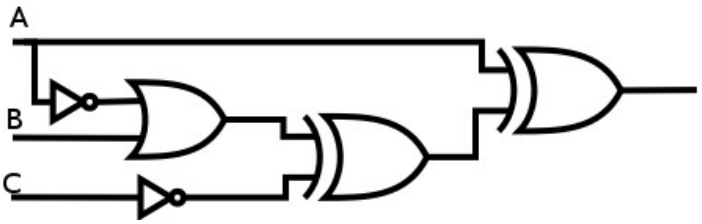
Pregunta 1

En la dirección de memoria [DS:BX] está guardado el entero 23.

- Sabiendo que el valor de DS es 0x3010 y el de BX 0xAA22, en qué posición de memoria física está guardado el entero?
- Especifique el valor de ES para que la dirección [ES:0x12] coincida con la dirección del entero guardado.
- Explique por qué no es posible especificar un valor para ES de forma tal que [ES:0x4] coincida con la dirección física del dato guardado.

Pregunta 2

¿Cuál es el propósito de utilizar los diagramas de Karnaugh en la construcción de circuitos?
¿Cuál es la tabla de verdad para la función implementada por el siguiente circuito?



Pregunta 3

Explique el propósito del vector de interrupciones.

Pregunta 4

Describa el propósito de los registros IP y FLAGS.
¿El registro de FLAGS es visible, parcialmente visible u oculto? Justifique su respuesta.

Solución Preguntas:

Pregunta 1

El cálculo para la dirección de memoria física es el resultado de multiplicar por 16 (o hacer un shift de cuatro a la izquierda) el valor del registro de segmento y sumarle el desplazamiento. En este caso, el resultado es 0x3AB22.

El valor de ES debe ser tal que ES * 16 + 0x12 = 0x3AB22, o sea, ES = 0x3AB1

Como los últimos 4 bits de la dirección de memoria física solo se ven afectados por el valor del desplazamiento, no existe valor posible de ES que permita generar la dirección 0x3AB22 si el valor del desplazamiento es 0x4 (más genéricamente: no hay forma de generar 0x3AB22 con un desplazamiento que no termine en 2).

Pregunta 2

Los diagramas de Karnaugh permiten expresar en forma mínima una función en dos niveles de compuertas. Se utiliza en el proceso de síntesis de un circuito justamente por su capacidad de lograr circuitos mínimos en dos niveles.

A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Pregunta 3

El vector de interrupciones es una estructura de datos que mantiene la asociación entre los códigos de las interrupciones del sistema y las direcciones de las rutinas de atención para las mismas. Entonces, al recibir el procesador una interrupción, determina la dirección de comienzo del código a ejecutar para esa interrupción consultando al vector de interrupciones.

Pregunta 4

El registro IP mantiene la dirección de la instrucción que se está ejecutando actualmente (o la siguiente instrucción a ejecutar, dependiendo de la arquitectura).

El registro de FLAGs mantiene el estado del procesador así como el último resultado de las banderas de las operaciones lógico-aritméticas. Se trata de un registro parcialmente visible.

Problema 1

Sea *F(m, n) = N* la función que mapea un número binario ***n*** de ***m*** bits en la palabra de código de Gray *N* de 16 bits asociada a dicho número. Para este problema consideraremos que el rango de **m** estará comprendido entre 1 y 15 inclusive.
La función F(m, n) puede definirse recursivamente de la siguiente manera:

F(1, 0) = 0
F(1, 1) = 1
F(m, n) = F(m-1, n), para $0 <= n < 2^{m-1}$
F(m, n) = 2^{m-1} + F(m-1, 2^m – n – 1), para $2^{m-1} <= n < 2^m$

Por ejemplo `F(3,3) = 0000000000000010` y `F(3,7) = 0000000000000100`

Se pide:

a) Implemente en alto nivel un procedimiento **recursivo en m** que cargue en memoria a partir de la dirección *GRAY* la función F(m, n) para 1 <= m <= 15. Dicho procedimiento debe generar y almacenar los 2^m valores de F(m, n) a partir de la direccion *GRAY*.
El procedimiento deberá tener el siguiente prototipo:

```
void genera_gray(short m);
```

Compile en assembler 8086 el procedimiento. Los argumentos del procedimiento se retiran del stack y se debe preservar el valor de los registros. La dirección *GRAY* se encuentra a partir del desplazamiento 0x0000 del segmento ES.

b) Calcule el consumo de stack en función de **m**.
Indique como utilizaría la tabla una vez cargada para obtener el valor de *F(10, 257)*

Solución Problema 1

a)

```
void genera_gray(short m)
{
     if(m == 1)
     {
          GRAY[0] = 0;
          GRAY[1] = 1;
     }else
     {
          genera_gray(m-1);
          int bitM = 1 << (m-1);
          int nPos = bitM;
          for(int pos = nPos; pos > 0; nPos++)
          {
               pos--;
               GRAY[nPos] = bitM | GRAY[pos];
          }
     }
}
```

Assembler 8086:

```
genera_gray proc
     ; guardo contexto
     push BP
     mov BP, SP
     push CX
     push DX
     push DI
     push SI
     ; obtego parametros del stack
     mov CX, [BP+4] ; CX = m
     cmp CX, 1
     jne recursivo
     ; paso base
     mov WORD PTR ES:[0], 0
     mov WORD PTR ES:[2], 1
     jmp fin
recursivo:
     dec CX
     push CX
     call genera_gray ; genera_gray(m-1)
     mov DX, 1
     shl DX, CL     ; DX = bitM
     mov DI, DX
     add DI, DI     ; DI = dirNPos
     mov SI, DI     ; SI = dirPos
for:
     cmp SI, 0
     jz fin
     dec SI
     dec SI         ; dirPos--
     mov CX, ES:[SI]
     or CX, DX
     mov ES:[DI], CX ; GRAY[nPos] = bitM | GRAY[pos]
     inc DI
     inc DI         ; dirNPos++
     jmp for
fin:
     ; restauro contexto
     mov CX, [BP+2]
     mov [BP+4], CX
     pop SI
     pop DI
     pop DX
     pop CX
     pop BP
     add SP, 2
     ret
genera_gray endproc
```

b)

Para el cálculo del consumo de stack se debe separar en 2 casos:

- Paso base (m=1): parámetro, dir retorno, BP, CX, DX, DI, SI. Es decir 7 palabras = 14 bytes
- Paso recursivo (m > 1): Cada paso recursivo consume exactamente lo mismo que el paso base (7 palabras = 14 bytes). Por lo tanto para m > 1 se tiene consumo: Paso_base + (m-1) * Paso_recursivo = m * 7 palabras = m * 14 bytes

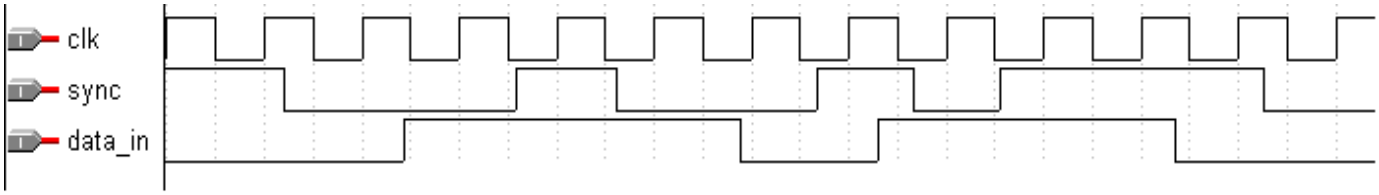
Por lo tanto el consumo de stack es **m * 14 bytes**

Para obtener el valor de *F(10, 257)* se debe leer el contenido del índice 257-ésimo de la tabla GRAY, es decir el contenido de la dirección **ES:[514]**

Problema 2

Se desea transmitir datos desde un dispositivo que emite en serie y otro que recibe en paralelo. El dispositivo transmisor envía los datos utilizando el siguiente protocolo:

Los datos se envían a través de la señal **data_in** en ráfagas de 8 bits. Luego de un reset, la señal **sync** vale 1 y antes de comenzar cada byte baja a cero durante un período completo de reloj. A partir de ahí cada bit válido en **data_in** se indica con un 1 en la señal **sync** durante el flanco ascendente del reloj. Luego de cada bit transmitido, **sync** baja a cero durante uno o más períodos de reloj. Una transmisión de ejemplo se presenta a continuación:

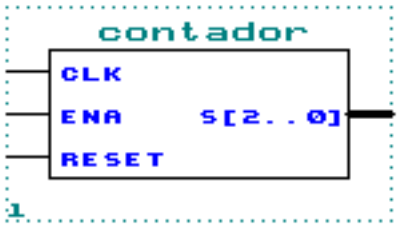


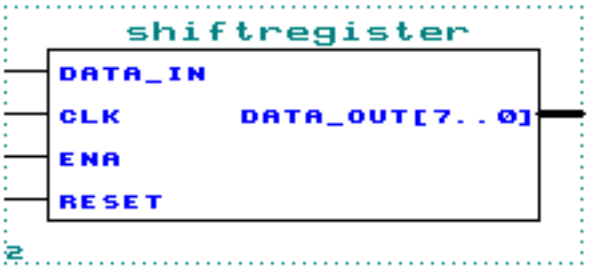
El sistema a implementar deberá leer los datos según el protocolo anterior y presentarlos en la salida **data_out[7..0]**. Se deberá indicar que hay un byte listo en dicha salida con un pulso a 1 de un período de duración en la señal **ready**.

Para la construcción del sistema se dispone de los siguientes bloques:

- Registro de desplazamiento (shift register) de 8 bits con habilitación (se captura el dato si **ena** vale 1).

- Circuito contador circular de 0 a 7 con habilitación. Mientras la señal **ena** valga 1, las salidas **s[2..0]** mostrarán los valores 0->1->2->...->7->0->1 en cada período de reloj. Si la señal **ena** vale 0, la salida permanece incambiada.





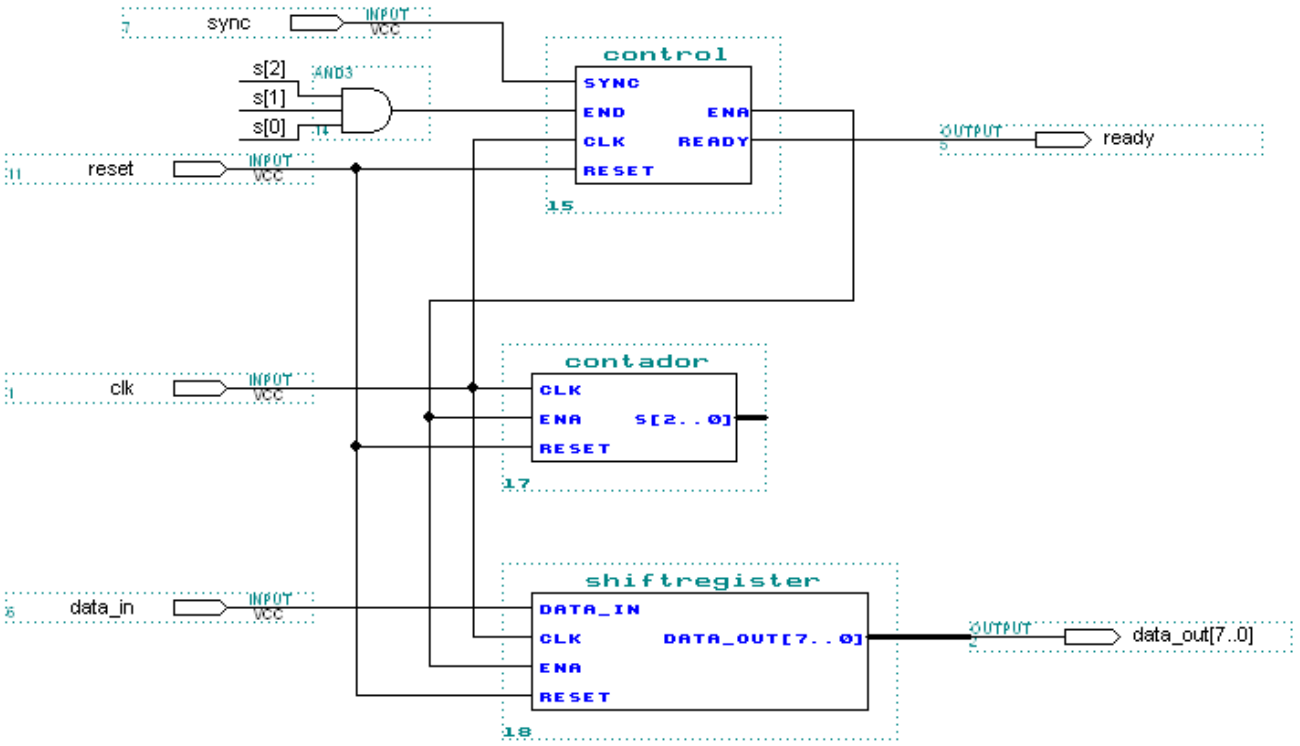
Se pide:

Utilizando la metodología del curso, construir un circuito que controle la señal **ready,** la señal **ena** del contador y la señal **ena** del registro de desplazamiento. Conectar dicho circuito adecuadamente con los otros bloques provistos para que el sistema resultante funcione según lo descrito anteriormente.

Nota: como la señal **sync** es propensa a presentar espurios, **NO** es aceptable una solución donde se utilice dicha señal como reloj.

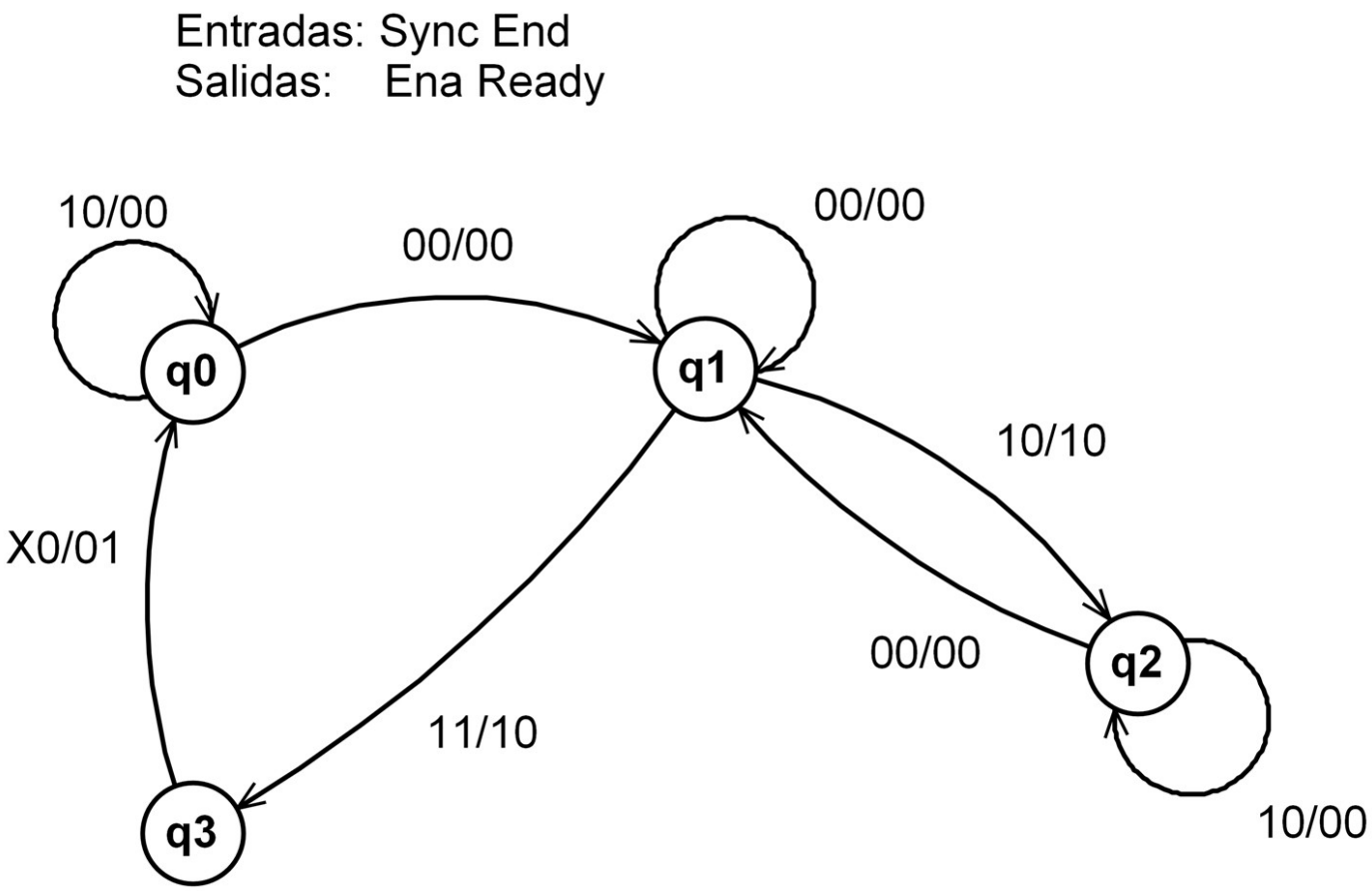
Solución Problema 2:

Circuito principal:



El bloque control será entonces construido siguiendo la metodología del curso.

El diagrama de estados queda de la siguiente manera:



Donde los estados tienen la siguiente semántica:

q0: Inicio
q1: Espero flanco subida
q2: Espero flanco bajada
q3: Ready a 1

En base a este diagrama de estados construimos la tabla de estados:

	Entradas		Próximo estado	Salidas	
Estado	Sync	End		Ena	Ready
Q0	0	0	Q1	0	0
	0	1	X	X	X
	1	0	Q0	0	0
	1	1	X	X	X
Q1	0	0	Q1	0	0
	0	1	X	X	X
	1	0	Q2	1	0
	1	1	Q3	1	0
Q2	0	0	Q1	0	0
	0	1	X	X	X
	1	0	Q2	0	0

	1	1	X	X	X
Q3	0	0	Q0	0	1
	0	1	X	X	X
	1	0	Q0	0	1
	1	1	X	X	X

Dado que son 4 estados, bastan 2 bits para codificarlos. Por lo tanto usaremos dos flip flops (tipo D) para guardar el estado.

Q0: 00
Q1: 01
Q2: 10
Q3: 11

Usando la ecuación de los FF tipo D, pasamos a la tabla de verdad:

Estado		Entrada		Prox. Est.		Salidas	
e1	e0	Sync	End	d1	d0	ena	ready
0	0	0	0	0	1	0	0
0	0	0	1	X	X	X	X
0	0	1	0	0	0	0	0
0	0	1	1	X	X	X	X
0	1	0	0	0	1	0	0
0	1	0	1	X	X	X	X
0	1	1	0	1	0	1	0
0	1	1	1	1	1	1	0
1	0	0	0	0	1	0	0
1	0	0	1	X	X	X	X
1	0	1	0	1	0	0	0
1	0	1	1	X	X	X	X
1	1	0	0	0	0	0	1
1	1	0	1	X	X	X	X
1	1	1	0	0	0	0	1
1	1	1	1	X	X	X	X

Los diagramas de Karnaugh correspondientes resultan:

e1 e0 \ sync end	00	01	11	10
00	0	X	X	0
01	0	X	1	1
11	0	X	X	1
10	0	X	X	0

d1 = sync . e0

e1 e0 \ sync end	00	01	11	10
00	1	X	X	0
01	1	X	1	0
11	1	X	X	0
10	0	X	X	0

d0 = end + !sync . !e1 + !sync . e0

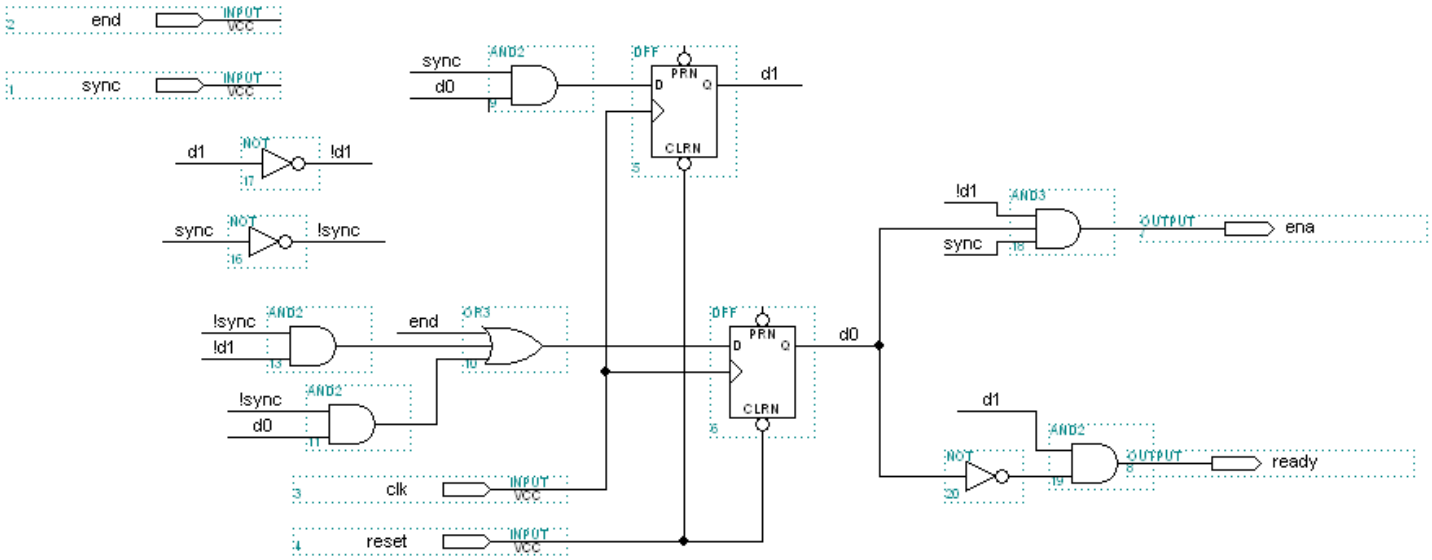
e1 e0 \ sync end	00	01	11	10
00	0	X	X	0
01	0	X	1	1
11	0	X	X	0
10	0	X	X	0

ena = !e1 . e0 . sync

e1 e0 \ sync end	00	01	11	10
00	0	X	X	0
01	0	X	0	0
11	0	X	X	0
10	1	X	X	1

ready = e1 . !e0

El circuito resultante es el siguiente:


end
INPUT
VCC
sync
INPUT
VCC
sync
d0
AND2
DFF
PRN
D
Q
CLRN
d1
d1
NOT
!d1
sync
NOT
!sync
!d1
AND3
sync
OUTPUT
ena
!sync
!d1
AND2
end
OR3
!sync
d0
AND2
DFF
PRN
D
Q
CLRN
d0
d1
AND2
NOT
OUTPUT
ready
clk
INPUT
VCC
reset
INPUT
VCC