Examen de Arquitectura de Computadoras

20 de febrero de 2024

Instrucciones:

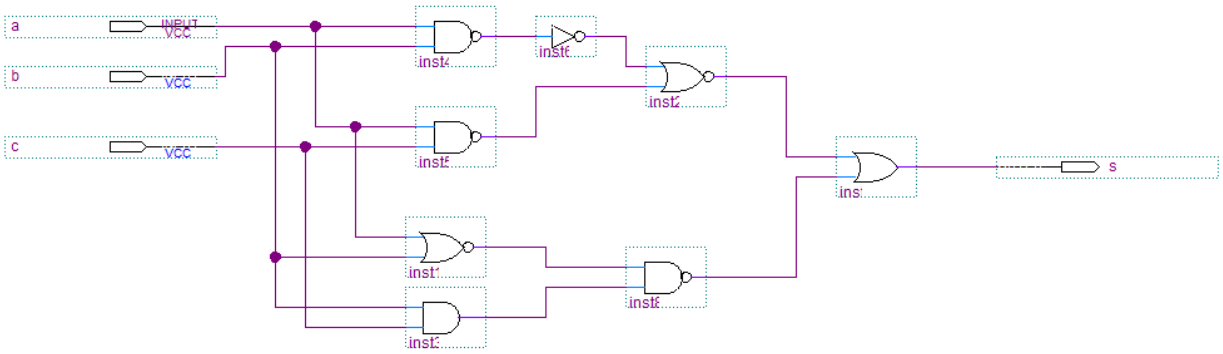
- Indique su nombre, apellido y número de cédula en todas las hojas que entregue.
- Las hojas deben estar numeradas y en la primer hoja debe escribirse el total de hojas entregadas.
- Apague su celular. No puede utilizar material ni calculadora.
- La duración del examen es de tres horas, incluyendo el tiempo para completar sus datos.
- Para aprobar debe contestar correctamente un ejercicio entero y dos preguntas.

Pregunta 1

a) Describa los pasos necesarios para codificar una palabra arbitraria de 4 bits de datos **a4a3a2a1** mediante códigos de Hamming de 4 bits de datos suponiendo que luego será transmitida por un canal con ruido.
b) Suponga ahora que recibe una palabra transmitida de acuerdo a la parte a). Describa los pasos necesarios para decodificar la palabra y determinar si es válida y si no lo es describa los pasos para determinar cuál sería la palabra corregida.

Pregunta 2

Calcule una expresión booleana mínima en dos niveles usando solamente compuertas OR, AND y NOT para la salida S dada por el siguiente circuito:



Pregunta 3

a) En el contexto de jerarquía de memoria, ¿puede un reemplazo producirse como consecuencia de un HIT en cache? ¿Y de un MISS? Justifique.
b) Explique las estrategia de reemplazo LRU.

Pregunta 4

Describa el potencial problema que aparece cuando se conectan varias señales de solicitud de interrupción de controladores de E/S a una misma entrada de pedido de interrupción de la CPU que detecta por flanco.

Ejercicio 1

Se desea implementar un circuito para controlar el portón de un garaje que puede subir y bajar. El mismo cuenta con un motor para mover el portón que se puede encender y apagar, y además indicar en que sentido trabaja (para arriba o para abajo), un control remoto con un solo botón, un sensor que detecta la presión de dicho botón y un sensor que indica si el portón está completamente arriba o completamente abajo.

Cuando el portón está cerrado (abajo) y se presiona el botón se debe encender el motor en el sentido de subida y mantenerlo así hasta que el sensor indique que está completamente arriba. En ese momento se debe apagar el motor. Cuando está completamente abierto (arriba) y se presiona el botón el comportamiento es el inverso con el motor bajando.

Si se presiona el botón mientras el portón está subiendo o bajando se considera que puede haber una emergencia por lo que el motor se deberá detenerse por un tick (un ciclo) de clock y luego comenzar a funcionar nuevamente en el sentido contrario.

Se pide: diseñar y dibujar un circuito secuencial en base a flip flops tipo D y compuertas básicas. El sentido del motor se codifica con un 1 para subir y 0 para bajar. El sensor del portón marca 1 si el portón está completamente arriba o completamente abajo y 0 en caso contrario. **Debe** utilizarse la metodología del curso en la resolución de este problema.

Ejercicio 2

Considere las siguientes estructuras de datos para almacenar árboles de enteros sin signo.

```
struct nodo_arbol {
      unsigned short dato;
      nodo_arbol * hijo_izq;
      nodo_arbol * hijo_der;
} arbol[N];
```

y el siguiente seudo-código de la función que determina la cantidad de elementos menores, iguales y mayores a un entero dado un árbol dado.

```
menor, igual, mayor cuento_comparo(nodo_arbol * rama, unsigned short elem) {
  unsigned short int elems_menor=0, elems_igual=0, elems_mayor=0;

  if (rama != NULL) {
    if (rama->dato < elem)
      elems_menor++
    else if (rama->dato == elem)
      elems_igual++
    else
      elems_mayor++;

    elems_menor, elems_igual, elems_mayor =+ cuento_comparo(rama→hijo_izq, elem);
    elems_menor, elems_igual, elems_mayor =+ cuento_comparo(rama→hijo_der, elem);
  }

  return(elems_menor, elems_igual, elems_mayor);
}
```

Se pide:
a) Escribir la función ***cuento_comparo*** en ensamblador 8086. Considere que la estructura del árbol está totalmente contenida en el segmento apuntado por el registro ES y el puntero es relativo a dicho segmento. Los argumentos se pasan y devuelven por el stack. La forma de invocar es del estilo:

PUSH “arbol”
PUSH “elem”
CALL cuento_comparo
POP “mayor”
POP “igual”
POR “menor”

b) Calcular el máximo valor de “N” para que la función pueda ejecutar siempre en una máquina 8086, justificando cual es la restricción que prevalece.

Respuesta Pregunta 1

a) El código de Hamming de 4 bits de datos utiliza 3 bits de paridad (p3,p2,p1) para detectar y corregir errores de un solo bit.
Se calculan los bits de paridad de forma que la distancia resultante del código sea 3 de la siguiente manera:

p1 = a4 XOR a2 XOR a1	p2 = a4 XOR a3 XOR a1	p3 = a4 XOR a3 XOR a2

Puede verificarse que para dos palabras del código original de distancia 1 al menos 2 bits de paridad cambiarán y para dos palabras del código original de distancia 2 al menos 1 bit de paridad cambiará.
Se codifica la palabra en el siguiente orden: **a4a3a2p3a1p2p1**

b) Para decodificar la palabra se debe primero calcular los síndromes(s3,s2,s1) de la palabra codificada de la siguiente manera:

s1 = p1 XOR a4 XOR a2 XOR a1	s2 = p2 XOR a4 XOR a3 XOR a1	s3 = p3 XOR a4 XOR a3 XOR a2

Es decir que se le aplica el XOR entre la paridad recibida y la paridad recalculada para los bits de dato recibidos.
Cuando no hay errores ambos coinciden y por lo tanto los tres síndromes valen 0. En ese caso la palabra recibida será a4a3a2a1
En caso que alguno de los síndromes no sea 0, se considera que ha habido 1 único error y en ese caso S=**s3s2s1** representa en binario la posición de la palabra recibida donde hay un error y para corregirla basta con invertir el valor de dicho bit. Puede verificarse evaluando las expresiones de los síndromes que si el error está en a1 entonces S=011, para a2 S=101, para a3 S=110 y para a4 S=111 por lo que en cada uno de esos casos la palabra corregida.
Por lo tanto la palabra decodificada será:

S=s3s2s1	Palabra decodificada
011	a4a3a2!a1
101	a4a3!a2a1
110	a4!a3a2a1
111	!a4a3a2a1
demás casos	a4a3a2a1

Respuesta Pregunta 2

La tabla de verdad del circuito es la sigiuente:

a	b	c	f(a,b,c)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Como se puede ver, la expresión booleana mínima en dos niveles no incluye conectivas binarias y es f(a,b,c) = 1.

Otra forma de obtener este resultado es notar que la ecuación para el NAND inferior del circuito es:

!(!(a+b) . bc) = (de morgan) !!(a+b) + !(b.c) = (de morgan, doble negación) a + b + !b + !c = 1 (neutro del OR)

Respuesta Pregunta 3

a) Un hit nunca puede dar como resultado un reemplazo. Las acciones luego de que ocurre un hit en caché son simplemente devolver el valor buscado a la CPU desde la caché.

Luego de que ocurre un MISS, la caché solicita el bloque que contiene el dato buscado al nivel inferior de la jerarquía, y al recibirlo debe colocarlo en alguna línea de la caché. En caso de que todas las líneas candidatas estén ocupadas, se debe elegir una según la estrategia de reemplazo utilizada. En conclusión, un reemplazo se produce como consecuencia de un MISS en caché.

b) El algoritmo LRU selecciona para reemplazar, dentro de las líneas posibles, la que tenga el bloque que haya sido accedido hace más tiempo.

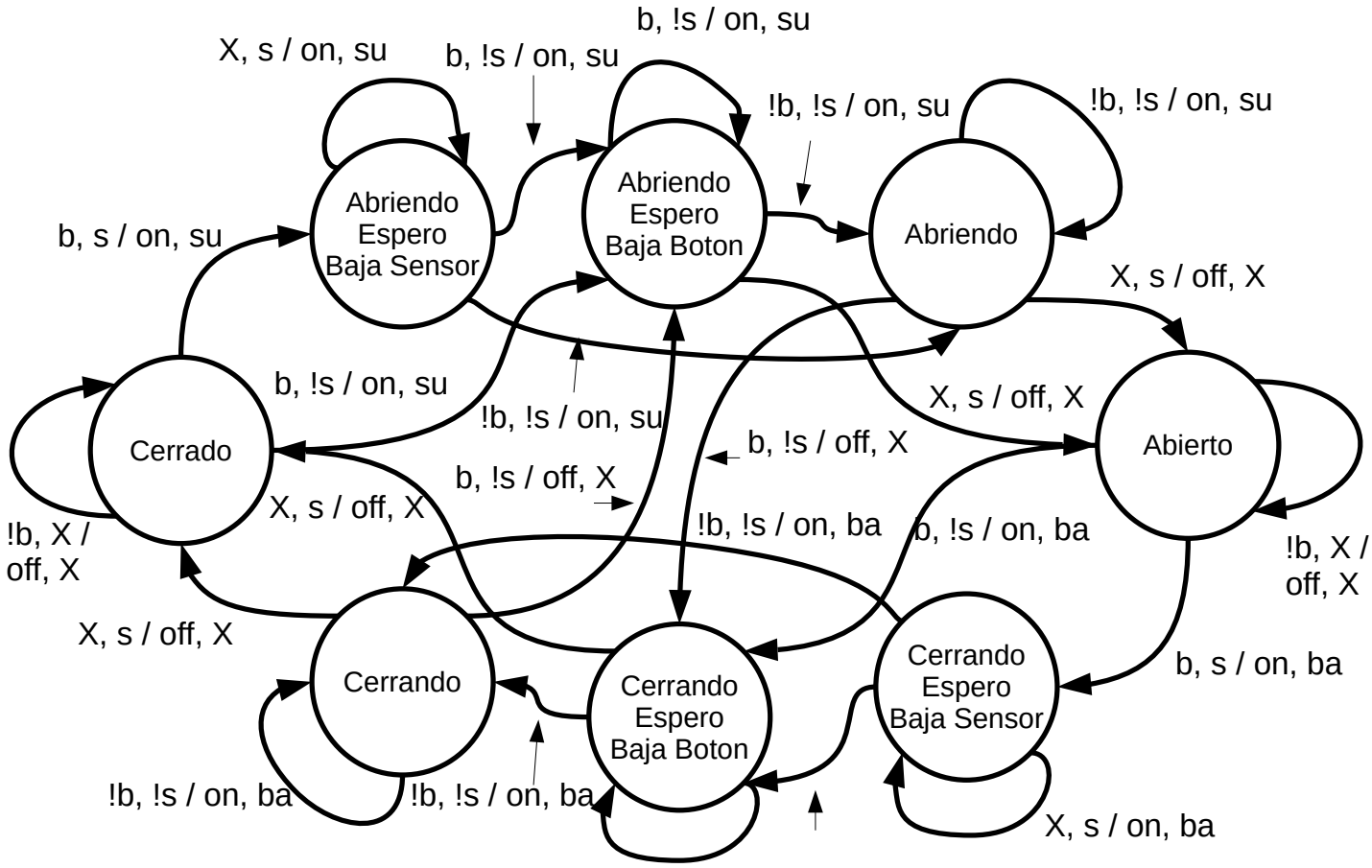
Respuesta Pregunta 4

Cuando tenemos varios controladores conectados en “paralelo” (mediante un OR cableado) a una entrada de pedido de interrupción que detecta por flanco el problema que aperece es que hay pedidos que no van a ser detectados y si no se elabora una estrategia apropiada en la rutina de atención a la interrupción se corre el riesgo de dejar bloqueado el sistema de detección.
Esto ocurre porque el controlador de E/S que solicita una interrupción subirá su salida de pedido de 0 a 1, y suponiendo que en ese momento no hay ningún otro controlador solictando se generará un flanco en la entrada de pedido de interrupción y el pedido será detectado. Pero si otro controlador sube su salida para solicitar interrupción antes que el primero hay sido atendido (y por consiguiente bajado su salida a 0), al ser un OR entre las señales de pedido, la entrada de pedido de interrupción seguirá en 1 y ningún flanco habrá sido generado y por tanto para la CPU el segundo pedido no existe.
Si se abandonara la rutina de atención enseguida de atender al primero que solicitó, el segundo seguiría manteniendo en 1 la entrada de pedido y por tanto bloqueando la detección de todo nuevo pedido, ya que no se detectarían flancos.

Solución Ejercicio 1

Las entradas del circuito son el sensor del botón (que puede estar apretado o no) y el sensor (que también puede estar detectando o no). La salida es el control del motor codificado con dos señales. La primera indica si está prendido o no y la segunda indica el sentido (sube o baja).

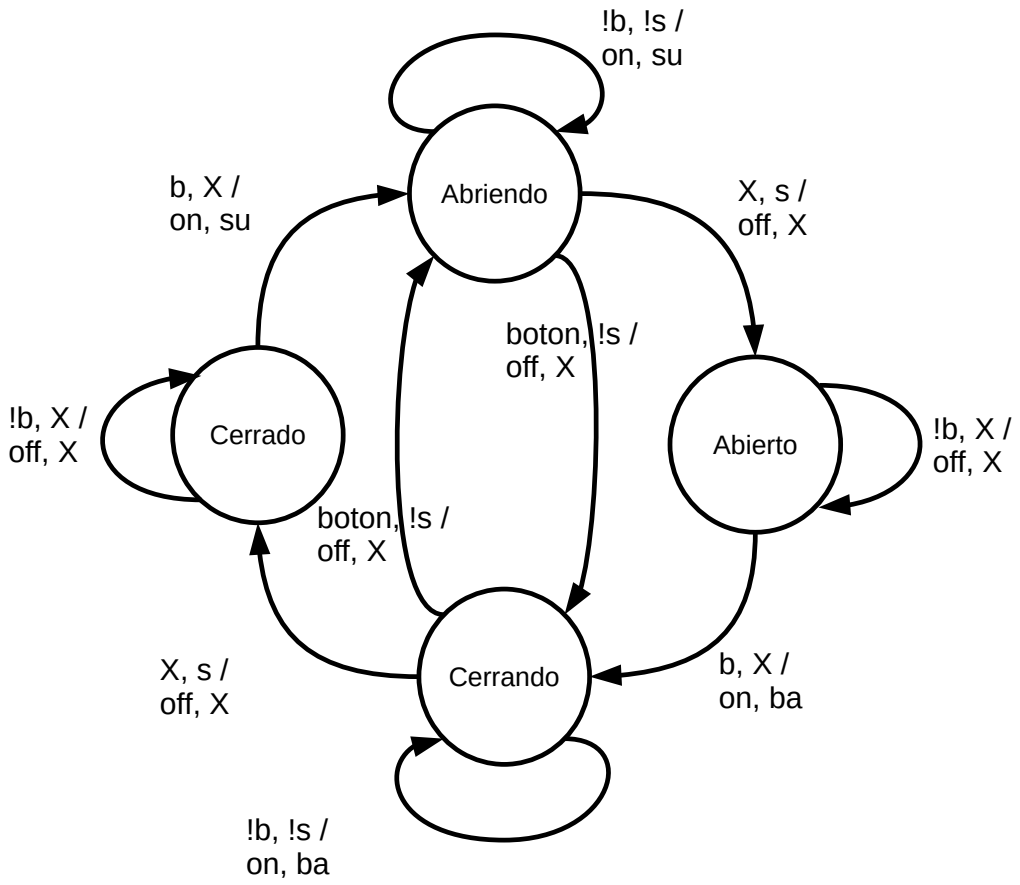
El diagrama de estados es el siguiente:



Esta máquina modela apropiadamente dos aspectos que, si bien no estaban expuestos explícitamente en la letra, son inherentes al funcionamiento de los dispositivos que generan las entradas, en particular en relación con la frecuencia del reloj del circuito que regula los momentos en que se muestrean los valores de dichas entradas:

- cuando se detecta que el botón está apretado hay que accionar, pero luego esperar que el usuario lo deje de apretar. Si no lo hacemos enseguida interpretaremos la entrada del botón como el accionar de emergencia y entraremos en un loop de cambio de sentido del movimiento del portón.
- el sensor del portón en una posición extrema no necesariamente cambia en forma instantanea, ya que el motor no arranca en esa manera (tiene una cierta inercia) y puede no generar el movimiento suficiente del portón para que el sensor cambie su valor en menos de un periodo del reloj del circuito. Por tanto también es necesario esperar que cambie, porque sino nuestra máquina de estados pordría entender que el portón llegó a cerrarse o abrirse en un periodo de reloj, lo que no tiene sentido físico.

Al momento de corregir el examen, no se tuvo en cuenta el modelado correcto del sensor y respecto al botón, se aceptaron soluciones que implícitamente asumieran que la señal del botón bajaba en menos de un periodo de reloj, dando lugar a una máquina de estados como la siguiente:



Volviendo a la solución completa, construimos la tabla de estados a partir del diagrama:

Estado	Boton	Sensor	Proximo Estado	Motor	Sentido
Cerrado	!boton	X	Cerrado	off	X
	boton	!sensor	Abriendo Espero Baja Sensor	on	sube
	boton	sensor	Abriendo Espero Baja Boton	on	sube
Abriendo Espero Baja Sensor	X	sensor	Abriendo Espero Baja Sensor	on	sube
	!boton	!sensor	Abriendo Espero Baja Boton	on	sube
	boton	!sensor	Abriendo	on	sube
Abriendo Espero Baja Boton	boton	X	Abriendo Espero Baja Boton	on	sube
	!boton	!sensor	Abriendo	on	sube
	!boton	sensor	Abierto	off	X
Abriendo	!boton	!sensor	Abriendo	on	sube
	X	sensor	Abierto	off	X
	boton	!sensor	Cerrando Espero Baja Boton	off	X
Abierto	!boton	X	Abierto	off	X
	boton	!sensor	Cerrando Espero Baja Sensor	on	baja
	boton	sensor	Cerrando Espero Baja Boton	on	baja
Cerrando Espero Baja Sensor	X	sensor	Cerrando Espero Baja Sensor	on	baja
	!boton	!sensor	Cerrando Espero Baja Boton	on	baja
	boton	!sensor	Cerrando	on	baja
Cerrando Espero Baja Boton	boton	X	Cerrando Espero Baja Boton	on	baja
	!boton	!sensor	Cerrando	on	baja
	!boton	sensor	Cerrado	off	X
Cerrando	!boton	!sensor	Cerrando	on	baja
	X	sensor	Cerrado	off	X
	boton	!sensor	Abriendo Espero Baja Boton	off	X

Codificamos de la siguiente manera:

boton: 0 (sin apretar), 1 (apretado)
sensor: 0 (sin detectar), 1 (detectando)
motor: 0 (off), 1 (on)
sentido: 0 (baja), 1 (sube)
estados:
Cerrado → 0 0 0
Abriendo Espero Baja Sensor → 0 0 1
Abriendo Espero Baja Boton → 0 1 0
Abriendo → 0 1 1
Abierto → 1 0 0
Cerrando Espero Baja Sensor → 1 0 1
Cerrando Espero Baja Boton → 1 1 0
Cerrando → 1 1 1

De esta manera la tabla de transiciones y salidas queda así:

Q2n Q1n Q0n	Boton	Sensor	Q2n+1 Q1n+1 Q0n+1	Motor	Sentido
0 0 0	0	X	0 0 0	0	X
	1	0	0 0 1	1	1
	1	1	0 1 0	1	1
0 0 1	X	1	0 0 1	1	1
	0	0	0 1 0	1	1
	1	0	0 1 1	1	1
0 1 0	1	X	0 1 0	1	1
	0	0	0 1 1	1	1
	0	1	1 0 0	0	X
0 1 1	0	0	0 1 1	1	1
	X	1	1 0 0	0	X
	1	0	1 1 0	0	X
1 0 0	0	X	1 0 0	0	X
	1	0	1 0 1	1	0
	1	1	1 1 0	1	0
1 0 1	X	1	1 0 1	1	0
	0	0	1 1 0	1	0
	1	0	1 1 1	1	0
1 1 0	1	X	1 1 0	1	0
	0	0	1 1 1	1	0
	0	1	0 0 0	0	X
1 1 1	0	0	1 1 1	1	0
	X	1	0 0 0	0	X
	1	0	0 1 0	0	X

Dado que vamos a utilizar flip-flops tipo D, las tablas de verdad serán:

Q2n	Q1n	Q0n	Boton	Sensor	D2n	D1n	D0n	Motor	Sentido
0	0	0	0	0	0	0	0	0	X
0	0	0	0	1	0	0	0	0	X
0	0	0	1	0	0	0	1	1	1
0	0	0	1	1	0	1	0	1	1
0	0	1	0	0	0	1	0	1	1
0	0	1	0	1	0	0	1	1	1
0	0	1	1	0	0	1	1	1	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	0	0	1	1	1	1
0	1	0	0	1	1	0	0	0	X
0	1	0	1	0	0	1	0	1	1
0	1	0	1	1	0	1	0	1	1
0	1	1	0	0	0	1	1	1	1
0	1	1	0	1	1	0	0	0	X
0	1	1	1	0	1	1	0	0	X
0	1	1	1	1	1	0	0	0	X
1	0	0	0	0	1	0	0	0	X
1	0	0	0	1	1	0	0	0	X
1	0	0	1	0	1	0	1	1	0
1	0	0	1	1	1	1	0	1	0
1	0	1	0	0	1	1	0	1	0
1	0	1	0	1	1	0	1	1	0
1	0	1	1	0	1	1	1	1	0
1	0	1	1	1	1	0	1	1	0
1	1	0	0	0	1	1	1	1	0
1	1	0	0	1	0	0	0	0	X
1	1	0	1	0	1	1	0	1	0
1	1	0	1	1	1	1	0	1	0
1	1	1	0	0	1	1	1	1	0
1	1	1	0	1	0	0	0	0	X
1	1	1	1	0	0	1	0	0	X
1	1	1	1	1	0	0	0	0	X

A continuación se realizan los diagramas de Karnaugh para obtener una expresión mínima en dos niveles para las salidas del circuito y entradas de los FF (por temas de espacio se abrevia Q2n = q2, Q1n = q1, Q0n = q0, D2n = d2, D1n = d1, botón = b, sensor = s)

q2 = 0

q1q0\bs	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	1	1	1
10	0	1	0	0

D2 = !q2.!b.s.q1 + !q2.q1.q0.b + q2.!b.!s + q2.!q1 + q2.b.!q0

q2 = 1

q1q0\bs	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	0	0	0
10	1	0	1	1

q2 = 0

q1q0\bs	00	01	11	10
00	0	0	1	`0
01	1	0	0	1
11	1	0	0	1
10	1	0	1	1

q2 = 1

q1q0\bs	00	01	11	10
00	0	0	1	0
01	1	0	0	1
11	1	0	0	1
10	1	0	1	1

D1 = b.s.!q0 + q0.!s + q1.!s

q2 = 0

q1q0\bs	00	01	11	10
00	0	0	0	1
01	0	1	1	1
11	1	0	0	0
10	1	0	0	0

q2 = 1

q1q0\bs	00	01	11	10
00	0	0	0	1
01	0	1	1	1
11	1	0	0	0
10	1	0	0	0

D0 = q1.!b.!s + !q1.q0.s + !q1.b.!s

q2 = 0

q1q0\bs	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	0	0	0
10	1	0	1	1

q2 = 1

q1q0\bs	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	0	0	0
10	1	0	1	1

motor = q1.!b.!s + !q1.q0 + b.!q0

q2 = 0

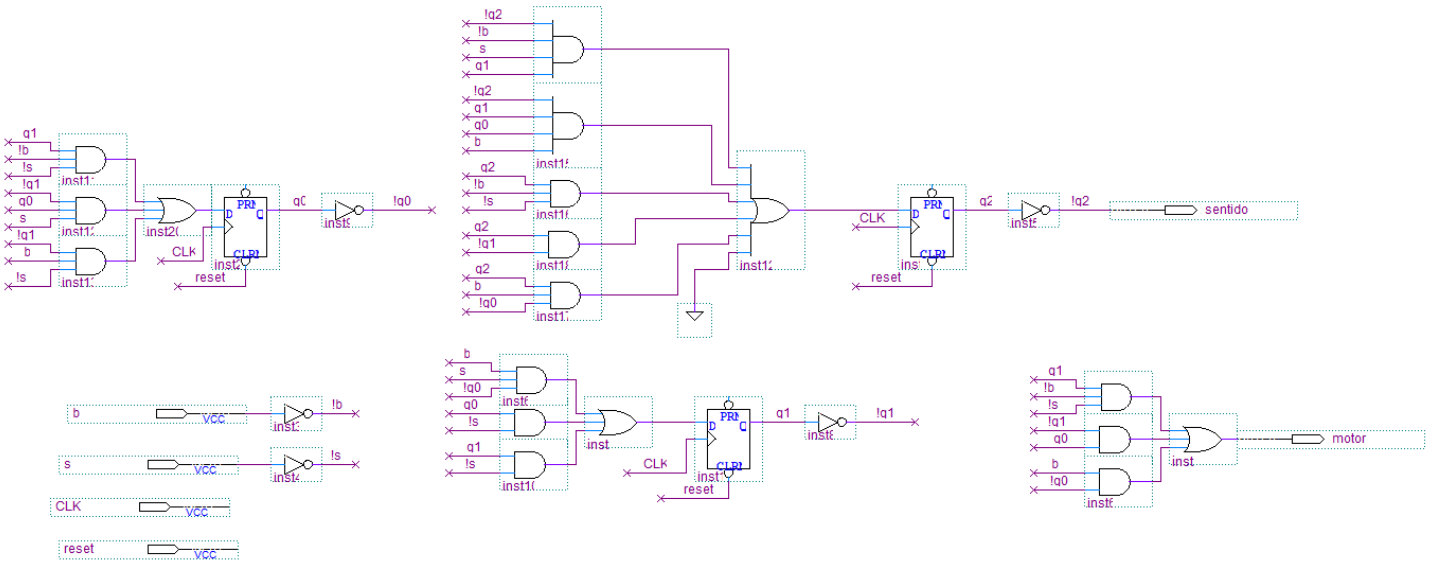
q1q0\bs	00	01	11	10
00	X	X	1	1
01	1	1	1	1
11	1	X	X	X
10	1	X	1	1

q2 = 1

q1q0\bs	00	01	11	10
00	X	X	0	0
01	0	0	0	0
11	0	X	X	X
10	0	X	0	0

sentido = !q2

Circuito


q1
!b
!s
!q1
q0
s
!q1
b
!s
CLK
reset
!q0
!q2
!b
s
q1
q0
b
!s
!q1
!q0
CLK
reset
sentido
b
s
CLK
reset
vcc
!b
!s
!q0
q0
!s
q1
!s
CLK
reset
q1
!q1
q1
!b
!s
!q1
q0
b
!q0
motor

Solución Ejercicio 2

a)

```
#define NULL 0

cuento_comparo proc
  sub sp,2
  push bp
  mov bp, sp
  push ax
  push bx
  push cx
  push di

  xor ax, ax          ;contador menores
  xor bx. bx          ;contador iguales
  xor cx, cx          ;contador mayores

  mov di, [bp+8]      ;cargo el puntero al arbol
  cmp di, NULL
  je fin
  mov dx, es:[di]
  cmp dx, [bp+6] ;comparo la raíz con el elemento
  jb es_menor
  je es_igual
  inc cx
  jmp recursion

es_menor:
  inc ax
  jmp recursion

es_igual:
  inc bx

recursion:
  push es:[di+2]      ;cargo en el stack puntero a rama izq
  push [bp+6]         ;cargo en el stack elemento
  call cuento_comparo
  pop dx
  add cx, dx
  pop dx
  add bx, dx
  pop dx
  add ax, dx

  push es:[di+4]      ;cargo en el stack puntero a rama izq
  push [bp+6]         ;cargo en el stack elemento
  call cuento_comparo
  pop dx
  add cx, dx
  pop dx
  add bx, dx
  pop dx
  add ax, dx
```

```
fin:
  mov [bp+8], ax
  mov [bp+6], bx
  mov ax, [bp+4]
  mov [bp+4], cx
  mov [bp+2], ax

  pop di
  pop cx
  pop bx
  pop ax
  pop bp
  ret
cuento_comparo endp
```

b) El tamaño del arreglo de nodos (N) está limitado por las siguientes restricciones:

- la estructura debe caber en un segmento de 64 Kbytes. Dado que cada nodo tiene 6 bytes (2 de datos, 2 de puntero izquierdo y 2 de puntero derecho) y que la posición 0 no se puede usar,

  $6 * N \leq 65535$ (*) => $N \leq 10922$

- el tamaño del stack está limitado también a un segmento, por lo que se debe analizar el consumo del stack en el peor caso.
  En el peor caso el árbol degenera en una lista de N nodos. Para recorrer en este caso el árbol se necesitan N+1 llamadas. Todas las llamadas consumen nueve palabras (tres argumentos de entrada, dirección de retorno y cinco registros de contexto) de stack, es decir 18 bytes. Por tanto:

  $18 * (N+1) < 65536 \Rightarrow N < 3640$.

La restricción más significativa, en este caso, es la segunda, por lo que debe ser $N < 3640$.

(*) La posición 0 del segmento no puede usarse porque el 0 es usado como indicador de fin para los punteros. En esta oportunidad el código de alto nivel propuesto admite para generar la recursión que sus ramas hijas sean NULL.