Examen de Arquitectura de Computadoras

26 de julio de 2021

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.**
- **Deben escanearse todas las hojas en orden para generar el pdf a entregar.**

Problema 1

Se desea construir un circuito que controle una baliza para automóvil. La baliza dispone de un único botón (***botón***) y un conjunto de LEDs, los cuales son controlados con una única salida (***luz***). Si el botón está presionado, la entrada ***botón*** vale 1 y 0 en caso contrario. Un 1 en la salida ***luz*** enciende los LEDs, mientras que un 0 los apaga.

La baliza tiene tres modos de funcionamiento: apagada, encendida e intermitente. La baliza inicia en modo apagada, y cambia de modo, en el orden indicado, apretando y soltando el botón. El cambio de modo ocurre al momento de apretar el botón. En el modo apagado, la luz debe permanecer apagada, en el modo encendido la luz debe permanecer encendida, y en el modo intermitente la luz se debe encender y apagar, según el valor de la entrada ***reloj_luz*** (onda cuadrada).

El circuito cuenta con una entrada de reloj (***clk***), cuya frecuencia se asume mucho mayor a la de la entrada que controla la intermitencia (***reloj_luz)***, y con una entrada de reset (***reset***) activa por nivel bajo.

Se pide:

Diseñar y dibujar el circuito que controla la luz, **utilizando la metodología del curso**. Se dispone de flip-flops tipo D y compuertas lógicas.

Problema 2

Se desea utilizar una ROM para implementar un comparador de números enteros representados en 8 bits. El operando A está representado en *Valor Absoluto y Signo*, mientras que el operando B está representado en *Complemento a 1*.
La salida del comparador será el mayor de A y B, representado en *Complemento a 2* de 8 bits.

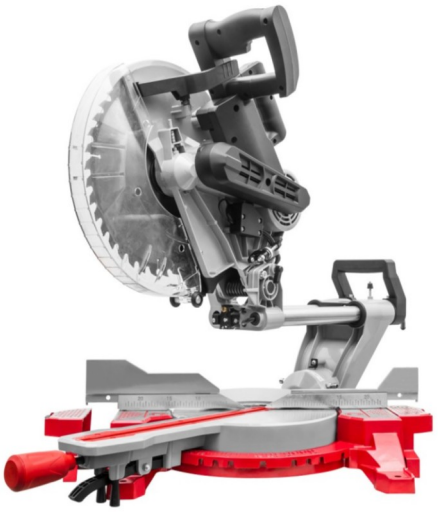
Se pide:

a) Construir la ROM necesaria, a partir de ROMs de 16 K x 16 y compuertas básicas.

b) Escribir en un lenguaje de alto nivel (preferentemente C) el programa de carga de la ROM construída para que funcione como el comparador descripto.

Problema 3

Se desea incrementar la seguridad de una sierra ingletadora para uso hogareño, a través de un sistema de control basado en un procesador dedicado.



Estas sierras incluyen un disco circular de corte que gira a gran velocidad, y un protector móvil que envuelve la sierra, para evitar que el disco entre en contacto con las personas u otros objetos durante la realización del corte.

La sierra cuenta con dos botones que deben presionarse simultáneamente para poner en marcha el motor, el cual se apaga inmediatamente si se libera alguno de ellos. Se considera que los botones se presionan simultáneamente, si los instantes en que se aprietan no difieren en más de 150 ms. El estado de los botones (1 presionado y 0 liberado) puede consultarse en los bits menos significativos del byte de E/S de solo lectura en la dirección ***STATUS***. El motor de la sierra se controla (1 enciende y 0 apaga) con el bit menos significativo del byte de E/S de solo escritura en la dirección ***CONTROL***.

Una vez encendido el motor de la sierra el usuario la coloca en posición para hacer el corte. Se dispone de un sensor que detecta cuando la protección hace tope con el material a cortar, y un sensor de distancia que devuelve la distancia en mm al material a cortar.

Debe controlarse que la distancia al material a cortar no varíe más de ***TOL_MM*** mm respecto de la distancia de referencia, que es la que existe al momento del primer tope de la protección con el material a cortar.

El sensor de tope interrumpe al procesador, invocando a la rutina ***tope()***, toda vez que el protector de la sierra entra en contacto con un objeto. La distancia medida por el sensor de distancia puede consultarse en la palabra de E/S de solo lectura en la dirección ***DISTANCE***.

En caso que la distancia varíe fuera del rango de seguridad debe habilitarse el freno mecánico y encender la luz de alarma que debe titilar cada un segundo. El freno mecánico se controla (1 frenado y 0 liberado) con el bit 2 del byte de E/S ***CONTROL*** y la luz con el bit 4 del mismo puerto. En caso de habilitarse el freno mecánico el usuario no podrá usar la sierra hasta tanto no la apague y encienda de nuevo.

Se dispone de un reloj externo de 100 Hz que interrumpe al procesador invocando a la rutina ***timer()***.

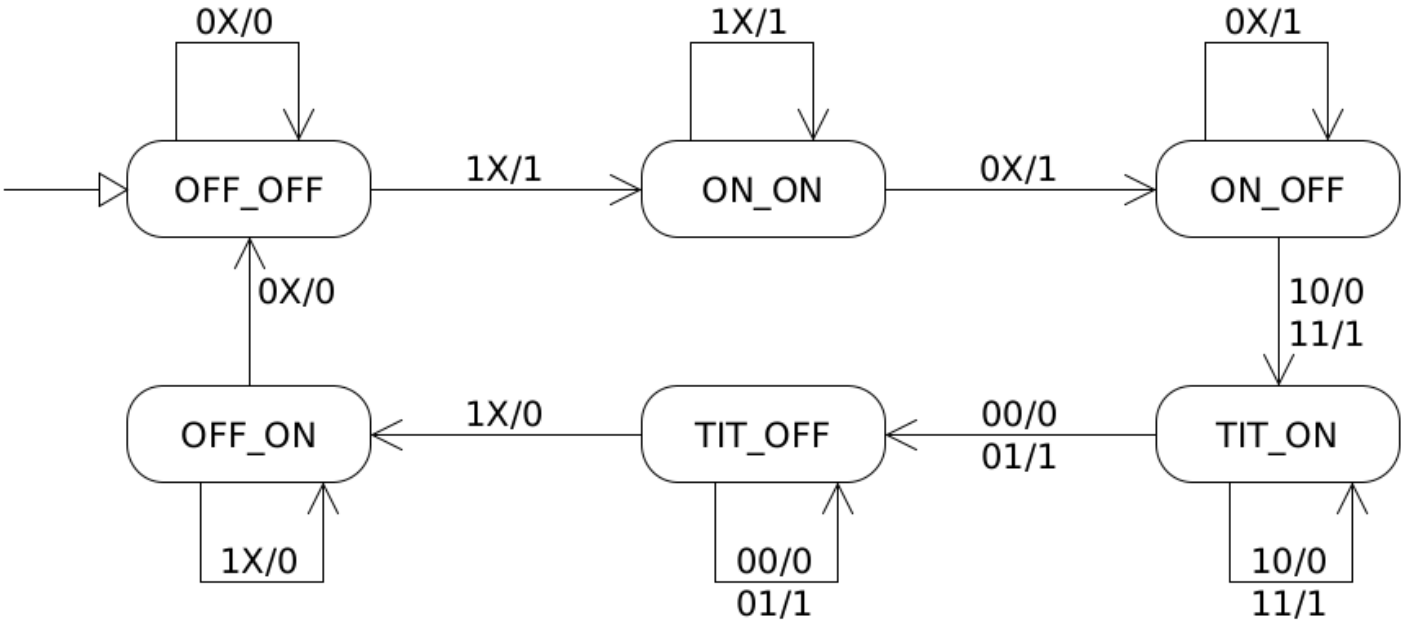
Se pide:

Diseñar una solución a la realidad planteada y escribir en un lenguaje de alto nivel (preferentemente C) todas las rutinas necesarias.

Escribir en assembler 8086 la instalación de las rutinas de interrupción sabiendo que la interrupción del sensor de tope tiene el identificador 5 y ***tope()*** se encuentra en la dirección 0x54321, y que para la interrupción del timer el identificador es 6 y la dirección de ***timer()*** es 0x6789A.

Solución Problema 1

Entradas/Salida: botón reloj_luz / luz



Estado Actual	Entradas		Próximo Estado	Salida
	boton	luz_reloj		luz
OFF_OFF	0	0	OFF_OFF	0
OFF_OFF	0	1	OFF_OFF	0
OFF_OFF	1	0	ON_ON	1
OFF_OFF	1	1	ON_ON	1
ON_ON	0	0	ON_OFF	1
ON_ON	0	1	ON_OFF	1
ON_ON	1	0	ON_ON	1
ON_ON	1	1	ON_ON	1
ON_OFF	0	0	ON_OFF	1
ON_OFF	0	1	ON_OFF	1
ON_OFF	1	0	TIT_ON	0
ON_OFF	1	1	TIT_ON	1

TIT_ON	0	0	TIT_OFF	0
TIT_ON	0	1	TIT_OFF	1
TIT_ON	1	0	TIT_ON	0
TIT_ON	1	1	TIT_ON	1
TIT_OFF	0	0	TIT_OFF	0
TIT_OFF	0	1	TIT_OFF	1
TIT_OFF	1	0	OFF_ON	0
TIT_OFF	1	1	OFF_ON	0
OFF_ON	0	0	OFF_OFF	0
OFF_ON	0	1	OFF_OFF	0
OFF_ON	1	0	OFF_ON	0
OFF_ON	1	1	OFF_ON	0

6 estados, se precisan 3 flip flops para codificarlos.

Codificación de estados:

OFF_OFF - 000

ON_ON - 001

ON_OFF - 010

TIT_ON - 011

TIT_OFF - 100

OFF_ON - 101

Tabla de Verdad

Estado Actual - q(n)			Entrada		Próximo Estado – q(n+1) = d(n)			Salida
q2	q1	q0	boton	luz_reloj	d2	d1	d0	luz
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	1	1
0	0	0	1	1	0	0	1	1
0	0	1	0	0	0	1	0	1
0	0	1	0	1	0	1	0	1
0	0	1	1	0	0	0	1	1
0	0	1	1	1	0	0	1	1
0	1	0	0	0	0	1	0	1
0	1	0	0	1	0	1	0	1
0	1	0	1	0	0	1	1	0
0	1	0	1	1	0	1	1	1
0	1	1	0	0	1	0	0	0
0	1	1	0	1	1	0	0	1
0	1	1	1	0	0	1	1	0
0	1	1	1	1	0	1	1	1
1	0	0	0	0	1	0	0	0
1	0	0	0	1	1	0	0	1
1	0	0	1	0	1	0	1	0
1	0	0	1	1	1	0	1	0
1	0	1	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0
1	0	1	1	0	1	0	1	0
1	0	1	1	1	1	0	1	0
1	1	0	0	0	X	X	X	X
1	1	0	0	1	X	X	X	X
1	1	0	1	0	X	X	X	X
1	1	0	1	1	X	X	X	X
1	1	1	0	0	X	X	X	X
1	1	1	0	1	X	X	X	X
1	1	1	1	0	X	X	X	X
1	1	1	1	1	X	X	X	X

Mapas K (b = boton, rel = reloj_luz)

q2 = 0

q1q0\ b,rel	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	0	0
10	0	0	0	0

q2 = 1

q1q0\ b,rel	00	01	11	10
00	1	1	1	1
01	0	0	1	1
11	X	X	X	X
10	X	X	X	X

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

q2 = 0

q1q0\ b,rel	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	0	0	1	1
10	1	1	1	1

q2 = 1

q1q0\ b,rel	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	X	X	X	X

d1 = !q2 !q1 q0 !b + q1 !q0 + q1 b

q2 = 0

q1q0\ b,rel	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

q2 = 1

q1q0\ b,rel	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	X	X	X	X
10	X	X	X	X

d0 = b

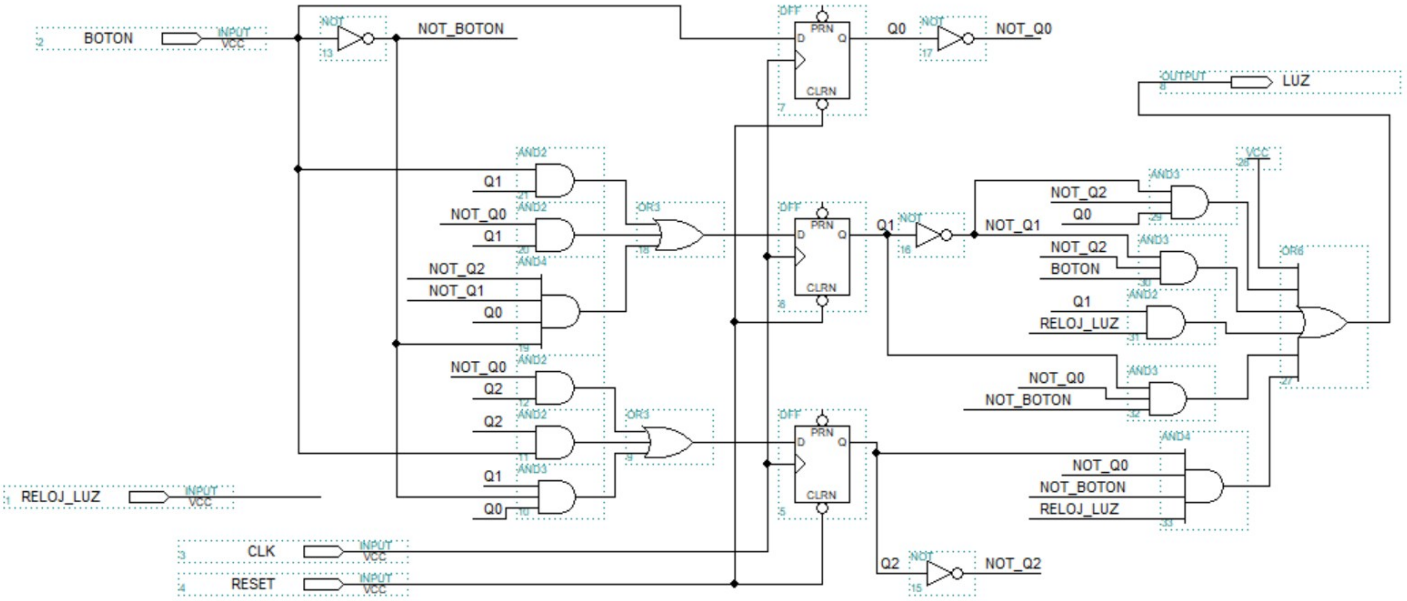
q2 = 0

q1q0\ b,rel	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	0	1	1	0
10	1	1	1	0

q2 = 1

q1q0\ b,rel	00	1	11	10
00	0	1	0	0
01	0	0	0	0
11	X	X	X	X
10	X	X	X	X

Luz = !q2 !q1 q0 + !q2 !q1 b + q1 rel + q1 !q0 !b + q2 !q0 !b rel

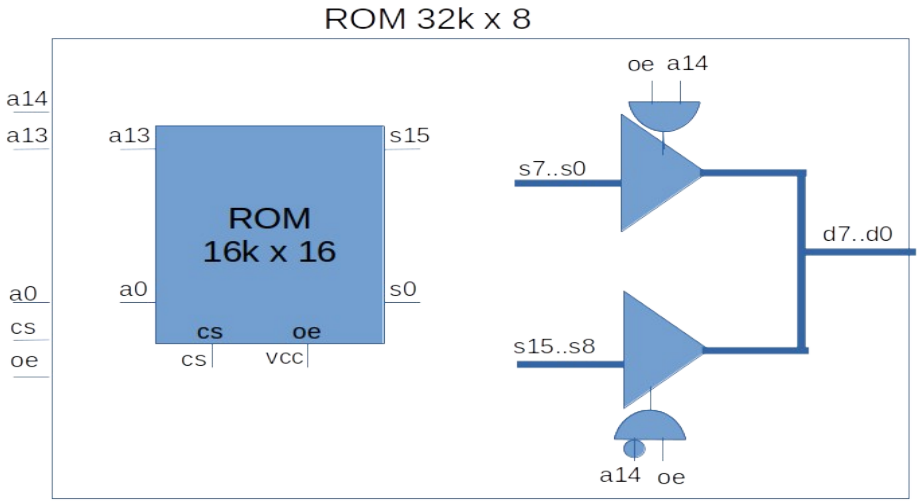

BOTON
INPUT
VCC
NOT
NOT_BOTON
Q0
NOT_Q0
OUTPUT
LUZ
Q1
NOT_Q0
Q1
NOT_Q2
NOT_Q1
Q0
NOT_Q0
Q2
Q2
Q1
Q0
NOT_Q2
Q0
NOT_Q1
NOT_Q2
BOTON
Q1
RELOJ_LUZ
NOT_Q0
NOT_BOTON
NOT_Q0
NOT_BOTON
RELOJ_LUZ
RELOJ_LUZ
CLK
RESET
Q2
NOT_Q2
PRN
D
Q
CLRN

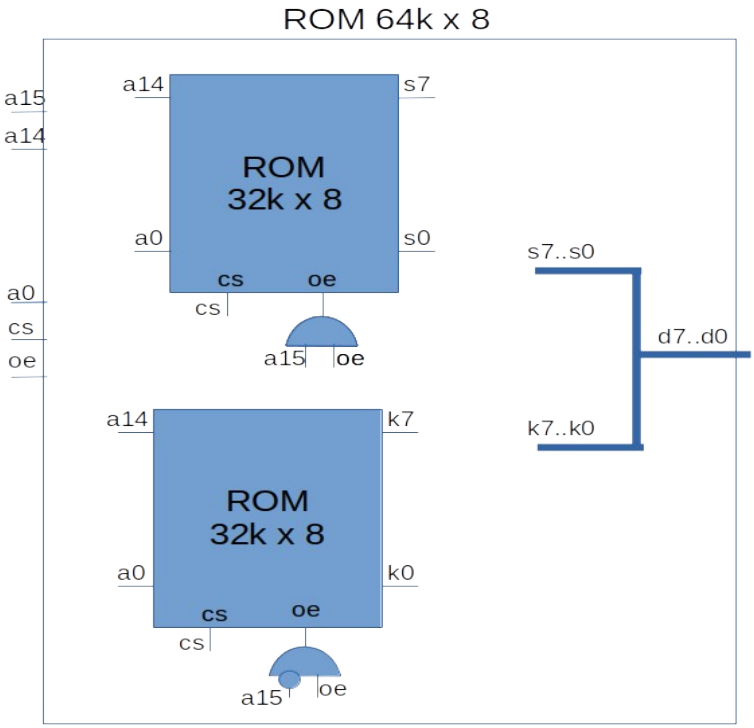
Solución Problema 2

Parte a)

Se requiere construir una ROM de 16 bits de entrada (dos operandos de 8 bits) y 8 bits de salida (un resultado de 8 bits) → 2^16 x 8 = 64k x 8.

Tenemos disponibles memorias de 16k x 16. El primer paso es transformar la memoria de 16k x 16 a 32k x 8, luego como segundo paso unificamos dos memorias de 32k x 8 y generamos una de 64k x 8.





Parte b)

```
char ROM[65536];
void cargarROM (){
      //recorro todas las direcciones
      for (unsigned int dir = 0; dir < 65536; dir++){
          //extraigo los dos operandos y los signos
          char op_A = dir >> 8;    // operando en v.a y signo.
          char op_B = dir & 0xFF;  // operando en complemento a 1
          char sg_A = op_A >> 7;
          char sg_B = op_B >> 7;
          if (sg_A == 1){
            op_A = op_A - 0x80;  // como es negativo, le quito el signo
            op_A = -op_A;        // cambio el signo en compl a 2
          }
          if (sg_B == 1) {
            op_B = op_B + 1;     // como es negativo, paso el compl a 1 a compl a 2
          }
          // ambos números están en compl a 2, los comparo con >
          ROM[dir] = (op_A > op_B)? op_A : op_B;
     }
}
```

Solución Problema 3

```
#define TIC_1_ SEC 100
#define TIC_2BUTTON 15

#define STATUS…
#define IZQ 1
#define DER 2
#define IZQDER 3

#define CONTROL…
#define MOTOR 1
#define FRENO 4
#define LUZ 16
#define APAGA 0
#define ENCIENDE 1

#define DISTANCE….
#define TOL_MM…

int tics;
char emergencia;
char tope;

void main() {
  int distancia_min, distancia_max;
  tics = 0;
  emergencia = 0;
  tope = 0;
  char presionado = 0, botones;
  // instalar ISR
  enable();
  while (true) {
    botones = in (STATUS) & IZQDER;
    switch (botones) {
    case IZQDER:
      break;
    case IZQ:
    case DER:
      if (presionado != botones) {
        tics = 0;
        presionado = botones;
      }
      break;
    default:
      tics = 0;
      presionado = 0;
    }
    if (tics > TIC_2BUTTON || botones != IZQDER)
      continue;

    out(CONTROL, ENCIENDE);
    tope = 0;
    distancia_max = 0;
```

```
    while (( in (STATUS) & IZQDER == IZQDER) {
  if (tope){
      int distancia = in (DISTANCE);
           if (!distancia_max) {
                distancia_max = distancia + TOL_MM; // se asume que entra en rango
                distancia_min = distancia - TOL_MM; // idem
           }
           if (distancia > distancia_max || distancia < distancia_min) {
                out(CONTROL, FRENO);
                tics = 0;
                emergencia = 1;
                while (1) {}
           }
  }
    }
    out(CONTROL, APAGA);
  }
}

void interrupt timer() {
  tics++;
  if (emergencia) {
    if (tics > TIC_1_ SEC)
      out(CONTROL, FRENO | LUZ);
    else
      out(CONTROL, FRENO);
    if (tics >= 2 * TIC_1_SEC)
      tics = 0;
  }
}

void interrupt tope() {
  tope = 1;
}
```

Assembler para la instalación

```
xor AX, AX
mov ES, AX

; instalar tope
mov word ptr ES:[20], 1     ; 5*4
mov ES:[22], 0x5432         ; 5*4+2

; instalar timer
mov word ptr ES:[24], 0xA   ; 6*4
mov ES:[26], 0x6789         ; 6*4 + 2
```