Examen de Arquitectura de Computadoras

19 de julio de 2019

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

Explique las dos filosofías de diseño para la Unidad de Control y sus principales diferencias.

Pregunta 2

a) Explique para qué se utilizan los diagramas de Karnaugh en la construcción de circuitos.
b) Minimice la siguiente función booleana:

$$f(a,b,c,d) = \sigma(0,1,2,4,5,6,8,10,11,12,14,15)$$

Pregunta 3

Explique cómo se realiza la comunicación de la CPU con los controladores de E/S y describa los dos tipos de arquitecturas de E/S respecto a la forma de acceso desde la CPU.

Pregunta 4

Considere una CPU 8086 con una memoria caché de 64 KB, con 1024 conjuntos y función de correspondencia asociativa de 4 vías. Indique como se interpreta una dirección de memoria para determinar el hit o el miss de un acceso.

Problema 1

Se desea controlar un sistema de clasificación de paquetes de encomienda que consta de una cinta transportadora que posee cuatro estaciones de procesamiento.

La primer estación dispone de un lector de código de barras a ambos lados de la cinta transportadora. La segunda estación posee un mecanismo de giro, que permite rotar 90° el paquete (para el caso que el código de barras que identifica el paquete esté en alguna de las dos caras que no pudieron ser escaneadas en la primer estación). La tercer estación dispone de una nueva pareja de lectores de códigos de barra. La cuarta y ultima estación posee un mecanismo de desvío que permite dirigir la caja al destino “nacional” o “internacional” en función de lo indicado en el código de barras.

Si no se pudo leer el código de barras en ninguna de las dos estaciones correspondientes, no se debe accionar el desvío.

Un esquema del sistema con sus componentes se presenta en la Figura 1.

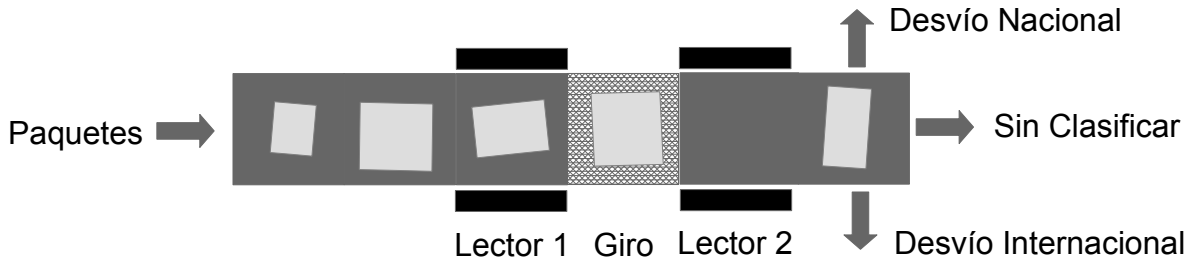

Figura 1: Esquema del sistema de clasificación de paquetes de encomienda

Existe un sistema de sensores que indican que un paquete está alineado en alguna estación. Se asume que si un paquete está alineado en una estación, los demás paquetes que estén sobre la cinta también estarán alineados en sus respectivas estaciones (puede darse que no haya paquetes en todas las estaciones) y que solo ocurre una interrupción por alineado por vez (por ejemplo en la figura hay paquete en la estación Lector 1, en Giro y en Desvío, cuando el paquete en Lector 1 esté alineado también lo estará el de Giro y el de Desvió y eso generará una sola invocación a alineado()).

Los lectores y el mecanismo de desvío pueden funcionar con la cinta en movimiento, pero no así el mecanismo de giro, que solo puede funcionar con la cinta detenida. El mecanismo de giro dispone de sensores que determinan que ha logrado girar el paquete 90°, generando una interrupción. Si el tiempo de funcionamiento del mecanismo de giro supera los 5 segundos se debe asumir que, o no hay paquete a girar, o existió algún problema por el cual no se pudo girar y el sistema debe proseguir.

Se cuenta con las siguientes interrupciones:

- ***alineado()****: s*e ejecuta cada vez hay paquete(s) alineado(s) en alguna(s) estación(es).
- ***girocompleto()***: se ejecuta cuando el mecanismo de giro completa su tarea.
- ***timer()***: se ejecuta a una frecuencia de 1 kHz.

También se cuenta con los siguientes puertos:

- Puertos de E/S de 8 bits de solo escritura en la direcciones ***LECTOR1*** *y* ***LECTOR2***, en cuyo bit menos significativo se dispara el respectivo lector de código de barras de la estación correspondiente al escribir un 1 en el mismo. La lectura es instantánea luego del disparo.
- Puertos de E/S de 16 bits de solo lectura en las direcciones ***LECTURA1*** y ***LECTURA2*** que informan el código de barras leído por los correspondientes lectores (0 si no hay lectura).
- Puerto de E/S de 8 bits de solo escritura en la dirección ***CONTROL***, en cuyo bit menos significativo se activa el motor de la cinta (1 = encendido), en su bit 7 (bit más significativo) se activa el desvío hacia “nacional” y en el bit 6 hacia “internacional” (1 = activado). El bit 4 controla el mecanismo de giro (1 = encendido).

Se pide: implementar en un lenguaje de alto nivel, preferentemente C, todas las rutinas necesarias para el control del sistema de clasificación, mediante un microprocesador dedicado. Se cuenta con la rutina ***destino(código)*** que devuelve 0 si el destino correspondiente al código es “*nacional*” y 1 en caso que sea “*internacional*”. El sistema deberá ser lo mas rápido posible, por lo que solo detendrá la cinta en los casos en que se requiera.

Problema 2

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

```
int arbol[MAX_LARGO];
unsigned char mapaDeBits[MAX_LARGO / 8];

int sumaArbol(int indice){
     int indiceMapa = indice / 8;
     int offsetMapa = indice % 8;
     int suma = 0;
     if (mapaDeBits[indiceMapa] & (1 << offsetMapa)){
          // si es válido el árbol
          int sumaIzq = sumaArbol(2 * indice + 1);
          int sumaDer = sumaArbol(2 * indice + 2);
          suma = sumaIzq + sumaDer + arbol[indice];
     }
     return suma;
}
```

En la representación propuesta, el nodo con índice i en el arreglo 'arbol' tiene sus hijos en los bytes con índices 2i + 1 (para el hijo izquierdo) y 2i + 2 (para el hijo derecho). A su vez, el arreglo 'mapaDeBits' indica en el bit i-ésimo si la entrada correspondiente del árbol es válida.

Se pide:

a) Compilar a ensamblador 8086. Considere que las variables arbol y mapaDeBits se encuentran disponibles a partir de las direcciones DS:SI y DS:DI respectivamente, y que el parámetro se pasa por stack. Se debe preservar el valor de todos los registros.
b) Calcular el consumo máximo de stack.

Solución Pregunta 1

Las dos filosofías de diseño para la Unidad de Control son la de *lógica cableada* y la de *lógica microprogramada*.

El diseño de la CPU en base a lógica cableada corresponde a un circuito secuencial, construido en base a un diagrama de estados que contemple todos los estados posibles en función de las distintas instrucciones y sus variantes. Se deben tomar como salidas todas las señales de control necesarias y como entradas los bits del código binario de las instrucciones. El método permite una implementación con la mejor velocidad de los circuitos, pero tiene poca flexibilidad ya que cualquier cambio en el diseño del set de instrucciones de la CPU hace necesario reescribir el diagrama de estados y, por ende, cambiar todo el circuito lógico que lo implementa.

La filosofía de lógica microprogramada propone que la CU se construya en base a un autómata más sencillo que ejecute el ciclo de instrucción de cada instrucción de la CPU siguiendo en secuencia un conjunto de microinstrucciones para cada instrucción particular. Para cada instrucción existe un microprograma (una secuencia lógica de microinstrucciones que establece el orden de los eventos necesarios para ejecutar la instrucción en la CPU). La CPU tiene un retardo en la velocidad de proceso, porque el microprograma se almacena en una ROM interna de la CPU y cada microinstrucción debe ser leída de la ROM, pero resulta mucho más simple modificar el set de instrucciones, ya que es suficiente cambiar el contenido de la ROM del microprograma.

Solución Pregunta 2

a) Los diagramas de Karnaugh se utilizan para minimizar expresiones lógicas mediante un método sistemático. En la construcción de circuitos es una herramienta utilizada, ya que al modelar circuitos con funciones lógicas es posible minimizarlas para construir un circuito equivalente (que ante las mismas entradas presente las mismas salidas), lo cual permite ahorrar compuertas para su construcción.

b)

$f(a,b,c,d) = \sigma(0,1,2,4,5,6,8,10,11,12,14,15)$

a	b	c	d	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1

1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

ab\cd	00	01	11	10
00	1	1	0	1
01	1	1	0	1
11	1	0	1	1
10	1	0	1	1

$F = \bar{a}\,\bar{c} + ac + \bar{d}$

Solución Pregunta 3

La comunicación de la CPU con los controladores de E/S se realiza a través de posiciones de memoria especiales de estos dispositivos, que son accesibles también para la CPU. Existen dos tipos de arquitecturas de E/S respecto a la forma de acceso desde la CPU:
1. La CPU dispone de un espacio de direcciones reservado para la E/S, que es accedido por instrucciones especiales (in y out). Un ejemplo de este enfoque es la familia de procesadores Intel.
2. La CPU accede a los controladores de E/S como si se trataran de posiciones normales de memoria, utilizando para tales fines cualquier instrucción que acceda a memoria. Un ejemplo de este diseño es la familia de procesadores SPARC y otros diseños RISC.

Solución Pregunta 4

Para determinar el hit o el miss el subsistema de caché descompone la dirección accedida por la CPU como:

TAG - CONJUNTO - BYTE

Dado que tenemos 64 KB y 4 vías, cada vía dispone de 64 / 4 = 16 KB. Al tener 1024 conjuntos, cada vía tendrá 1024 líneas de caché, por lo que el tamaño de la línea es de 16 bytes.
Precisamos 4 bits para identificar el BYTE, 10 bits para el CONJUNTO y dado que una CPU 8086 direcciona 1 MB de memoria, dispone de 20 bits en total, por lo que los 6 más significativos serán el TAG.

Solución Problema 1

Mantendremos un array de cuatro elementos para indicar el código de barras que está en la correspondiente estación. Si es 0 indicará que no se ha leído. En función de los valores del código se tomarán las acciones correspondientes en las estaciones de Giro, Lector 2 y Desvío. El array se desplazará hacia el final en cada alineación y se cargará el elemento inicial con la lectura del Lector 1.

```
#define TRUE              1
#define FALSE             0
#define NACIONAL          0x80
#define INTERNACIONAL     0x40
#define GIRAR             0x10
#define NOTGIRAR          0xEF
#define CINTA             0x01
#define TIMEOUTGIRO       5000

unsigned char girado;
int tics;
int codigos[4];

void interrupt girocompleto() {

     girado = TRUE;
}

void interrupt timer() {

     tics--;
}

void interrupt alineado() {

     unsigned char vcontrol;
     int i, nuevo_codigo;

     vcontrol = 0;
     if (codigos[3] != 0)
          if (destino(codigos[3] == 1) vcontrol = vcontrol | INTERNACIONAL;
          else vcontrol = vcontrol | NACIONAL;
     if (codigos[2] == 0) {
          out(LECTOR2, 0x01);
          codigos[2] == in(LECTOR2);
     }
     out(LECTOR1, 0x01);
     nuevo_codigo == in(LECTOR1);
     if(codigos[1] == 0) {
          vcontrol = vcontrol | GIRAR;
          tics = TIMEOUTGIRO;
          girado = FALSE;
          out(CONTROL, vcontrol);
          enable();  // se requiere para permitir timer() y girocompleto()
          while(!(girado) && (tics > 0));
          vcontrol = (vcontrol & NOTGIRAR) | CINTA;
          out(CONTROL, vcontrol);
```

```
      }
      else {
            vcontrol = vcontrol | CINTA;
            out(CONTROL, vcontrol);
      }
      for(i = 3; i > 0; i--) codigos[i] = codigos[i – 1];
      codigos[0] = nuevo_codigo;
}

void main() {

      int i;

      // instalo rutinas de interrupción
      for(i = 0; i < 4; i++) codigos[i] = 0;
      out(control, CINTA);        // arranco la cinta transportadora
      enable();
      while (TRUE);
}
```

Solución Alternativa

Esta solución alternativa no habilita interrupciones dentro de la interrupción alineado() (que es una acción que tiene cierto riesgo si no se hace adecuadamente).

```
#define NACIONAL 128
#define INTERNACIONAL 64
#define SIN_CLASIFICAR 0
#define ACTIVA 1
#define GIRO 16

void main()
{
  // instalar rutinas de interrupción
  espera=0; paso1=1; paso2=0; paso3=0;
  sentido = SIN_CLASIFICAR;
  out(CONTROL, ACTIVA);
  enable();
  while(1){}
}

void interrupt alineado()
{
  tics = 0;
  if(!paso1)
     espera = 1;

  if(!paso2)
  {
     out(LECTOR2, 1);
     paso2 = in(LECTURA2);
  }
  out(CONTROL, espera ? GIRO : ACTIVA|sentido);
```

```
  if(paso3)
     sentido = destino(paso3) ? INTERNACIONAL : NACIONAL;
  else
     sentido = SIN_CLASIFICAR;
  out(LECTOR1, 1);
  paso3=paso2;
  paso2=paso1;
  paso1 = in(LECTURA1);
}

void interrupt girocompleto()
{
  if(espera)
  {
     out(CONTROL, ACTIVA|sentido);
     espera = 0;
  }
}

int tics, sentido, paso1, paso2, paso3;
char espera;

void interrupt timer()
{
  if(espera)
  {
     tics++;
     if(tics == 5000)
     {
          out(CONTROL, ACTIVA|sentido);
          espera = 0;
     }
  }
}
```

Solución Problema 2

a)

```
SUMAPROC
        PUSH BP
        MOV BP, SP
        PUSH AX
        PUSH BX
        PUSH CX

        MOV CL, 0x03
        MOV BX, [BP + 4]
        SHR BX, CL              // BX = índice mapa

        // CX =offset mapa
        MOV CX, [BP + 4]
        AND CL, 0x07

        // genero máscara
        MOV CH, 0x01
        SHL CH, CL

        XOR AX,AX
        AND CH, DS:[DI+BX]
        JZ FIN

        // llamada recursiva derecha
        MOV BX, [BP + 4]
        SHL BX, 1
        INC BX
        PUSH BX
        CALL SUMA
        POP AX

        // llamada recursiva izquierda
        INC BX
        PUSH BX
        CALL SUMA
        POP CX
        ADD AX, CX

        // suma del árbol
        MOV BX, [BP + 4]
        SHL BX,1
        ADD AX, DS:[SI + BX]

FIN:    MOV [BP +4], AX
```

```
        POP CX
        POP BX
        POP AX
        POP BP
        RET
ENDPROC
```

b)

La altura del árbol indica cuál será el máximo número de llamadas recursivas. Para un MAX_LARGO dado, la altura del árbol se calcula como ceil(log_2(MAX_LARGO)).

Si N es la altura del árbol, el programa incurrirá como máximo en un paso base y N llamadas recursivas. Tanto el paso base como el recursivo consume 2 bytes por parámetros, 2 bytes por IP y 8 bytes por contexto.

Por lo tanto, el máximo consumo de stack es:

max_stack = 12 * (ceil (log_2(MAX_LARGO)) + 1) bytes