Examen de Arquitectura de Computadoras

28 de julio de 2020

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

Implemente un contador binario ascendente módulo 8 utilizando flip-flops J-K.

Pregunta 2

En la dirección de memoria ES:[BX] está guardado el entero -12.
• Sabiendo que el valor de ES es 0x1021 y el de BX 0xAA33, indique la posición de memoria física en la que está guardado el entero.
• Especifique el valor de DS para que la dirección DS:[0xB3] acceda al entero.

Pregunta 3

Utilice el metodo de Karnaugh para minimizar la función de 4 variables definida por la expresión:

f(a, b, c, d) = a'bc' + a'c + abcd + bc'

Pregunta 4

Explique cómo se derermina un hit para una caché totalmente asociativa y para una caché asociativa por conjuntos de 4 vías.

Problema 1

Se pretende utilizar una ROM para calcular la suma de dos números naturales (sin signo) representados en BCD de 2 dígitos y representar el resultado en binario.

Se pide:

a) Especificar tamaño de la ROM, entradas/salidas y construirla disponiendo de compuertas básicas y ROMs de 16Kx16. Implementar en alto nivel el código de carga de la ROM construida.

b) Compilar el código de carga de la parte anterior en Intel 8086. Asuma que la ROM comienza en la posición 0 del segmento extra.

Problema 2

Dada la situación de emergencia sanitaria y la reapertura de las grandes superficies comerciales bajo techo a las que concurren un número elevado de personas, se propone el desarrollo de COVIDFREE, un sistema que permite automatizar el monitoreo de este tipo de establecimientos.

Para la solución se utilizan diversos sensores. El objetivo en la primera versión es detectar las siguientes situaciones: exceso de personas en el establecimiento y personas con temperatura corporal más alta que la permitida para permanecer en lugares cerrados (37.4°).

Para contar la cantidad de personas que se encuentran en el establecimiento, se dispone de contadores en cada una de las tres entradas del mismo. Los contadores de personas cuentan tanto la cantidad de gente que entra como la que sale. Cuando dispone de información actualizada cada contador genera una interrupción, que invoca a la rutina ***isr_contadores()***, dejando en el byte de E/S de solo lectura en la dirección CONTADORES la información codificada de la siguiente manera: en los 2 bits más significativos está el número de puerta, en los siguientes 5 bits el valor absoluto de la cantidad y en el bit menos significativo el signo de la cantidad. Por ejemplo, si en el intervalo que se mide entraron dos personas y salieron 3, entonces el conteo es -1. En cambio, si entraron dos personas y salió una, entonces el conteo que se devuelve es +1.

Existen monitores en distintos lugares para mostrar el conteo actual de personas. En la palabra de E/S de solo escritura en la dirección PANTALLAS se escribe un entero con el valor a mostrar. Se debe actualizar este valor cada 5 segundos.

En caso de que la máxima cantidad de personas permitida (MAXPERSONAS) sea excedida, se debe prender una alarma a los efectos que el personal de seguridad tome las medidas pertinentes. La alarma debe sonar durante diez segundos y detenerse. Luego de esto se debe esperar 30 segundos, y si la cantidad de personas sigue excediendo el máximo, entonces debe volver a sonar durante 10 segundos y así sucesivamente hasta que la cantidad de gente sea menor o igual al máximo permitido. Esta alarma se enciende escribiendo un 1 en el bit menos significativo del byte de E/S de solo escritura en la dirección ALARMA_MAX (un 0 en dicho bit apaga la alarma).

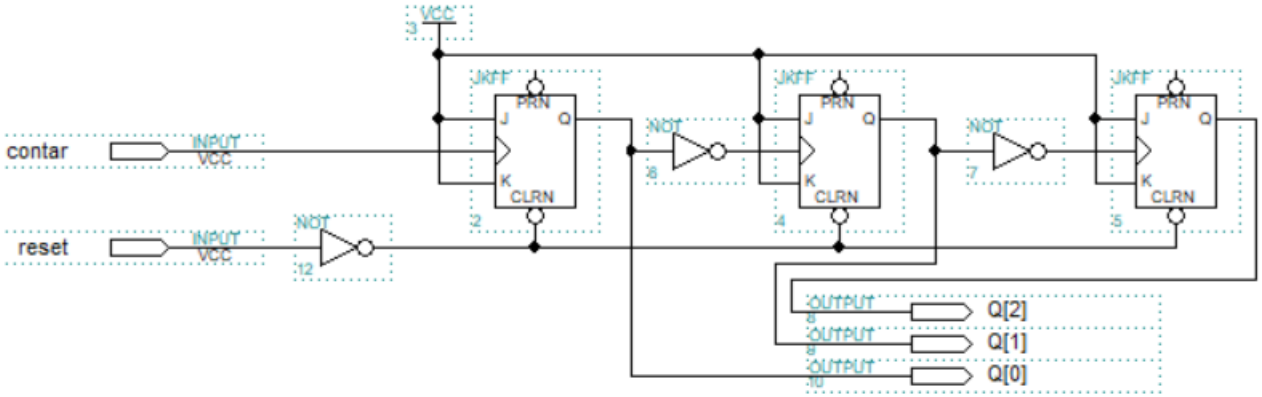
Además se controla la temperatura de las personas durante su permanencia en el establecimiento. Para esto se dispone de cámaras que detectan la temperatura. Cuando una cámara detecta que una persona en su campo de visión sobrepasa la temperatura aceptada, se debe emitir una alarma y además dar aviso a los puestos de control para que los guardias atiendan la situación. Cuando las cámaras detectan que se sobrepasa la temperatura, generan una interrupción que invoca a la rutina de interrupción ***isr_camaras()***, dejando su identificador en los 7 bits más significativos del byte de E/S de solo lectura en la dirección CAMARAS. La alarma se emite escribiendo un 1 en el bit menos significativo del byte de E/S de solo escritura ALARMA_TEMP (se apaga escribiendo 0). El aviso a los puestos está activo mientras esté en 1 el bit menos significativo del byte de E/S de solo escritura en la dirección AVISAR_PUESTO, colocando en los bits más significativos el identificador de la cámara que realizó la detección. Luego que se recibe el aviso de una cámara, los próximos avisos de esa cámara se ignoran durante 1 minuto. que es el tiempo que se estima que los guardias tardan en encontrar a la persona en estado febril y acompañarla fuera de las instalaciones. Al finalizar el minuto de debe apagar el aviso.

Se dispone de un timer que interrumpe invocando a la rutina ***isr_timer()*** con una frecuencia de 10Hz.

Se pide: escribir en un lenguaje de alto nivel, preferentemente C, todas las rutinas necesarias para implementar el sistema descripto utilizando un procesador dedicado a la tarea.

Respuesta Pregunta 1

Con los flip-flops J-K contruímos flip-flops T para utilizar su propiedad de funcionar como divisor de frecuencia al encadenarlos de su salida a la entrada de reloj del siguiente.



Respuesta Pregunta 2

La dirección de memoria física se calcula como 0x1021 * 16 + 0xAA33 = 0x1AC43

Si DS:0xB3 = 0x1AC43, entonces DS * 16 + 0x00B3 = 0x1AC43

DS = 0x1AB9

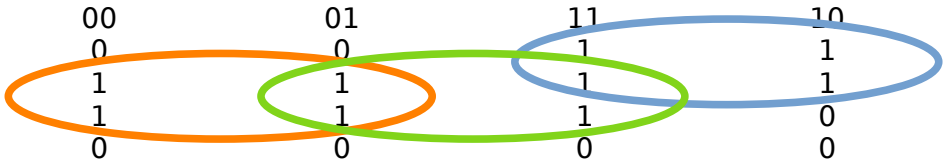
Respuesta Pregunta 3

Tabla de verdad:

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

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

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



f(a,b,c,d) = bc' + bd + a'c

Respuesta Pregunta 4

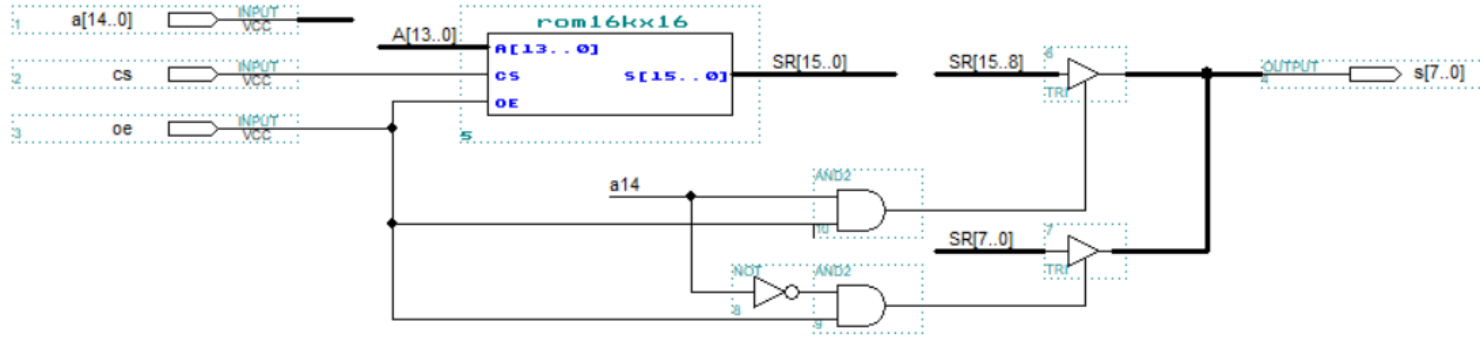
Para determinar un hit en una caché totalmente asociativa, se compara la sección *tag* de la dirección contra todos los tags almacenados en la caché. Si alguno de los tags coincide, hay hit.

Para determinar un hit en una caché asociativa por conjuntos de 4 vías, primero se debe acceder a las líneas del conjunto indicado en la sección *conjunto* de la dirección (bits del medio), comparando el *tag* de la dirección con los tags de las líneas del caché del conjunto mencionado, si alguno coincide, se trata de un hit.

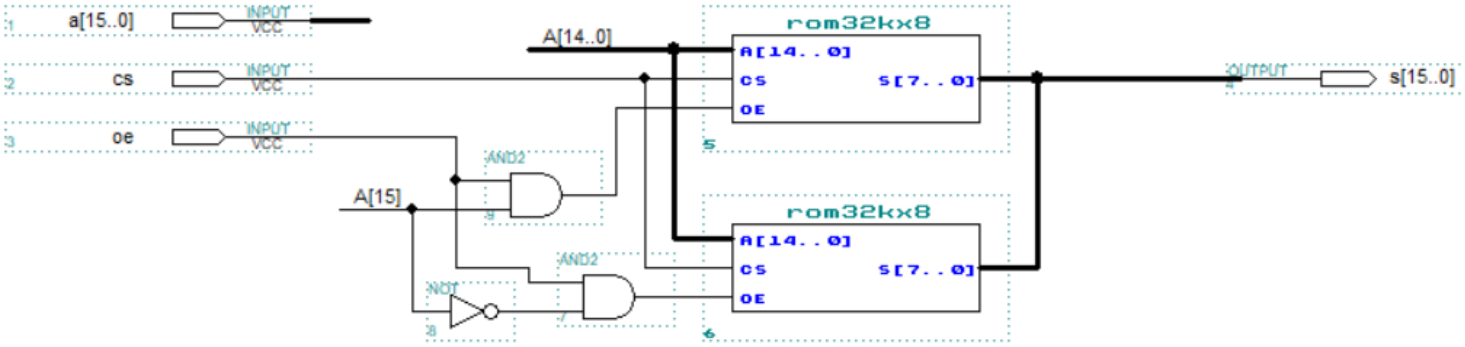
Solución Problema 1

a)
Cada dígito BCD ocupa 4 bits. Por lo tanto se tienen 16 bits de entrada (8 para numero 1 y 8 para número 2). La suma no puede exceder 198 por lo que se precisan 8 bits de salida. Por lo tanto se requiere una ROM de 64K x 8

Para armar la rom: con una ROM de 16Kx16 se arma RomA de 32Kx8



Luego con dos RomA se arma la rom de 64Kx8:




```
unsigned char ROM[65536];
void carga()
{
  unsigned char a = 0, bcdA = 0;
  do{
    unsigned char b = 0, bcdB = 0;
    do{
      ROM[bcdA<<8 | bcdB] = a + b;
      b++;
      bcdB++;
      if(bcdB & 0xF > 9)
        bcdB += 6;
    }while(b<100);
    a++;
    bcdA++;
```

```
    if(bcdA & 0xF > 9)
      bcdA += 6;
  }while(a<100);
}
```

b)

```
carga proc
  xor AH, AH ; AH=a
  xor BH, BH ; BH=bcdA
loop1:
  xor AL, AL ; AL=b
  xor BL, BL ; BL=bcdB
loop2:
  mov CL, AH
  add CL, AL
  mov ES:[BX], CL
  inc AL
  inc BL
  mov CL, BL
  and CL, 0xF
  cmp CL, 10
  jb fin_if1
  add BL,6
fin_if1:
  cmp AL, 100
  jb loop2
  inc AH
  inc BH
  mov CL, BH
  and CL, 0xF
  cmp CL, 10
  jb fin_if2
  add BL,6
  cmp AH, 100
  jb loop1
  ret
carga endp
```

Solución Problema 2

```
#define MAXPERSONAS 1000
#define CANT_CAM 128
#define 5_SEG 50
#define 10_SEG 100
#define 30_SEG 300
#define 60_SEG 600
#define ALARMA_CANTIDAD_DESACT 0
#define ALARMA_CANTIDAD_ACT 1
#define ALARMA_CANTIDAD_PAUSA 2
#define ALARMA_TEMP_DESACT 0
#define ALARMA_TEMP_ACT 1
$define TRUE 1

void interrupt isr_timer(){
  tics++;
  if (tics % 5_SEG == 0) OUT(PANTALLAS, cantidad);
  if (alarma_cantidad == ALARMA_CANTIDAD_ACT){
    if (tics_alarma_act == 0)
      //activar alarma
      OUT(ALARMA_MAX, 0x01);
    tics_alarma_act++;
    if (tics_alarma_act == 10_SEG){
      alarma_cantidad = ALARMA_CANTIDAD_PAUSA;
      tics_alarma_inact = 0;
      tics_alarma_act = 0;
      //desactivar alarma
      OUT(ALARMA_MAX, 0x00);
    }
  }
  if (alarma_cantidad == ALARMA_CANTIDAD_PAUSA){
    tics_alarma_inact++;
    if (tics_alarma_inact == 30_SEG){
      if (cantidad > MAXPERSONAS){
        alarma_cantidad = ALARMA_CANTIDAD_ACT;
        tics_alarma_act++;
        //activar alarma
        OUT(ALARMA_MAX, 0x01);
      }
      else{
        alarma_cantidad = ALARMA_CANTIDAD_DESACT;
      }
    }
  }
  if (alarma_temp_estado == ALARMA_TEMP_ACT){
    int cont_alarmas_activas = 0;
    for (int i=0; i < CANT_CAM; i++){
      if (estado_cam[i]){
       tics_cam[i]++;
```

```
      if (tics_cam[i] == 60_SEG){
        tics_cam[i] = 0;
        estado_cam[i] = 0;
      }
      else{
        cont_alarmas_activas++;
      }
    }
  }
  if (!cont_alarmas_activas)
    alarma_temp_estado = ALARMA_TEMP_DESACT;
    OUT(ALARMA_TEMP, 0x00);
  }
}

void interrupt isr_contadores(){
  int lectura_cont = IN(CONTADORES);
  id_puerta = lectura_cont >> 6;
  int cant = (lectura_cont & 0x3F) >> 1;
  if (lectura_cont & 0x01)
    cant = cant * (-1);
  cantidad += cant;
  if (cantidad > MAXPERSONAS) && alarma_cantidad != ALARMA_CANTIDAD_PAUSA)
    alarma_cantidad = ALARMA_CANTIDAD_ACT;
}

void interrupt isr_camaras(){
  char camaras_lect = IN(CAMARAS) >> 1;
  if !(estado_cam[camaras_lect]){
    estado_cam[camaras_lect] = 1;
    tics_cam[camaras_lect] = 0;
    //emito señal de alarma
    OUT(ALARMA_TEMP, 0x01);
    OUT(AVISAR_PUESTO, ( camaras_lect << 1) | 0x01);
    alarma_temp_estado = ALARMA_TEMP_ACT;
  }
}

int tics, tics_alarma_act, tics_alarma_inact, cantidad, alarma_temp_estado;
int tics_cam[CANT_CAM], estado_cam[CANT_CAM];

void main(){
  // instalar ISRs
  tics = 0;
  tics_alarma_act = 0;
  tics_alarma_inact = 0;
  id_puerta = 0;
  cantidad = 0;
  for (int i = 0; i < CANT_CAM; i++) {
    estado_cam[i] = 0;
```

```
    tics_cam[i] = 0;
  }
  enable();
  while (TRUE);
}
```