Solución Examen de Arquitectura de Computadoras

26 de julio del 2014

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

¿Qué es un hazard estructural? Ejemplifique y proponga dos posibles soluciones para este tipo de hazard.

Pregunta 2

¿Un sistema de codificación binario de distancia 3 permite corregir errores de dos bits? Justifique su respuesta.

Pregunta 3

¿Qué son los registros visibles, parcialmente visibles y los registros internos de un CPU?
Clasifique los registros IP, AX, IR y SP de la arquitectura Intel 8086 según las categorías anteriores.

Pregunta 4

Describa las organizaciones de memoria cache de correspondencia asociativa y correspondencia directa.

¿En qué principio se basa el buen funcionamiento de la jerarquía de memoria?

Problema 1

Se desea construir un circuito **Cronómetro** que cuenta minutos y segundos. El mismo debe mostrar en todo momento el tiempo en un display de cuatro dígitos de 7 segmentos (dos para los minutos, dos para los segundos), es decir de (**00:00** a **59:59**).

El cronómetro tiene dos botones para ajustar el tiempo:

- Un botón de **reset** pone el sistema en **00:00**
- Un botón **incrementar** que incrementa el tiempo a razón de 1 minuto por cada segundo que el botón esté presionado. Mientras que se está pulsando este botón los segundos no deben incrementarse.

Mientras no se presiona ninguno de los dos botones el cronómetro actualiza el tiempo cada 1 segundo.

Una imagen del **Cronómetro** se puede ver en la figura 1 y un detalle de las entradas de cada módulo de display de 7 segmentos en la figura 2:

Figura 1

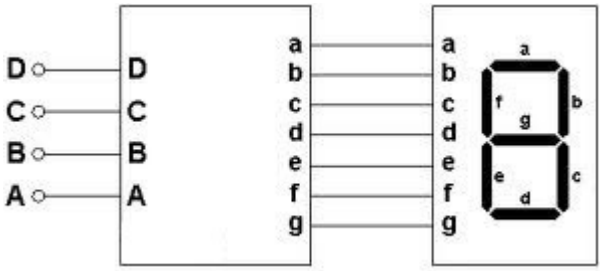


Figura 2

El sistema cuenta con una señal de reloj de 1Hz.

Se pide:

Construir de circuito **Cronómetro** descripto. Para ello se dispone de flip flops tipo D, módulos display 7 segmentos (1 dígito cada módulo) y compuertas básicas.

Problema 2

La empresa ***Bitbeat*** decidió desarrollar una batería electrónica compacta, basándose en un procesador dedicado.

La batería consta de 4 pads (o gomas) que simulan ser los tambores, los que al ser golpeados, provocan el llamado a la rutina de interrupción ***golpe()***. El o los pads que provocaron la interrupción se pueden identificar a través del registro de E/S de 8 bits en la dirección **PAD_ACTIVADO**, donde el bit menos significativo corresponde al pad 1, el segundo bit al pad 2, etc., estando en 1 en caso de haber sido activados.

A su vez los pads detectan la intensidad del golpe que se le aplica, para la que se usa una escala que va de 0 a 15. El valor de intensidad puede ser obtenido a través del registro de E/S de sólo lectura de 16 bits en la dirección **INTENSIDAD.** La intensidad del pad 1 estará en los 4 bits menos significativos, la intensidad del pad 2 en los siguientes 4 bits y así sucesivamente.

Para generar el sonido de cada pad es necesario escribir el nivel de intensidad en el registro de E/S de sólo escritura de 16 bits en la dirección **INTENSIDAD_OUT**, donde deberá escribirse la intensidad de cada pad en los mismos bits que en en el registro **INTENSIDAD**.

Como es sabido, cuando uno golpea un tambor real, éste resuena por una cierta cantidad de tiempo, en el cual la intensidad de su reproducción irá disminuyendo hasta ser nula. Este comportamiento deberá ser imitado por la batería electrónica. Para ello tendrá que disminuir a razón de 4 unidades por segundo hasta ser cero. Notar que, entonces, si ningún pad fue disparado en 4 segundos el registro **INTENSIDAD_OUT** debería tener 0x0000.

Para implementar el sistema se dispone de un reloj que interrumpe a **4 Hz** llamando al procedimiento ***reloj().***

Se pi

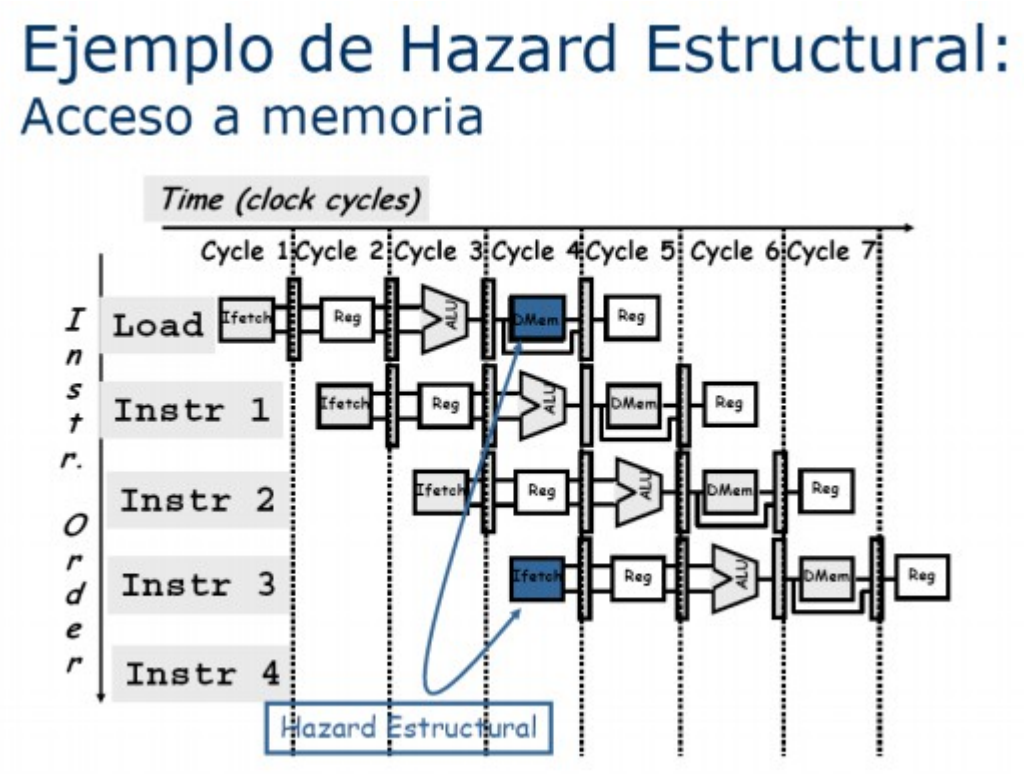
Parte a) Escribir en un lenguaje de alto nivel (preferentemente C) todas las rutinas necesarias para implementar el sistema.

Parte b) Compilarlas a lenguaje ensamblador 8086.

Solución

Pregunta 1

Los Hazards Estructurales se producen cuando dos o más etapas del pipeline intentan acceder al mismo recurso (por ejemplo etapas de fetch de instrucción y operandos tratando de acceder a la memoria) Se pueden resolver esperando (se deben tener mecanismos para detectar el hazard y “no hacer nada”), agregando más hardware (por ejemplo otro puerto de acceso a memoria) o diseñando el set de instrucciones de forma que no se den.



Pregunta 2

Un sistema de codificación binario de distancia 3 no permite la corrección de errores de dos bits, como se puede ver en el siguiente contraejemplo:

Sea el sistema de codificación binario de 4 bits, que codifica en las siguientes dos palabras:

1010
0101

Si se envía la palabra 1010, ocurre un error doble y el receptor recibe la palabra 1111, el receptor no podrá deducir si se envió la palabra 0101 y falllaron los bits 3 y 1, o si se envió la palabra 1010 y fallaron los bits 2 y 0. Por lo tanto no puede corregir el error.

Pregunta 3

Los registros visibles (o programables), son aquellos que el programador puede manipular directamente en los programas.

Los registros parcialmente visibles son aquellos que participan de modo indirecto en las instrucciones (es decir que participan en ellas, pero no son referenciados explícitamente).

Los registros internos (o invisibles), son aquellos que utiliza la unidad de control para poder ejecutar las instrucciones.

IP: parcialmente visible
AX: visible
IR: interno
SP: parcialmente visible

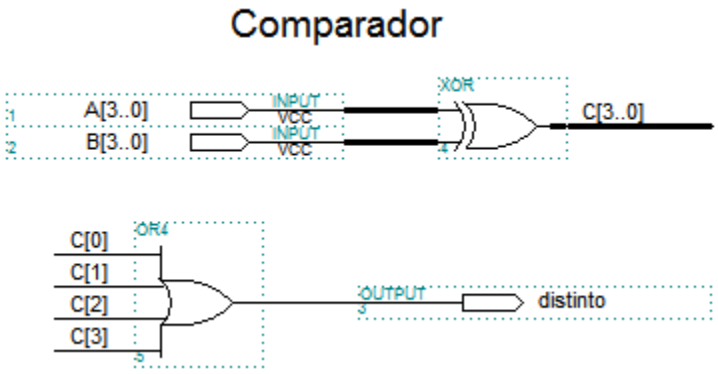
Pregunta 4

Las líneas de un cache se organizan en conjuntos. Cada bloque de memoria tiene un conjunto asignado según la función de correspondencia utilizada y dentro de ese conjunto se puede ubicar en cualquier línea. La cantidad de líneas que contiene un conjunto se denomina 'asociatividad' del conjunto. Si la asociatividad es 1, se dice que la función de corresondencia es **directa**, porque para cada bloque tiene asignada una única línea dentro del cache. Si la asociatividad es mayor que uno se dice que la función de correspondencia es **asociativa por conjuntos de n vías**, donde n es la cantidad de líneas de los conjuntos. El caso extremo es cuando existe un único conjunto formado por todas las líneas del cache, en ese caso se dice que la función de correspondencia es **totalmente asociativa.**

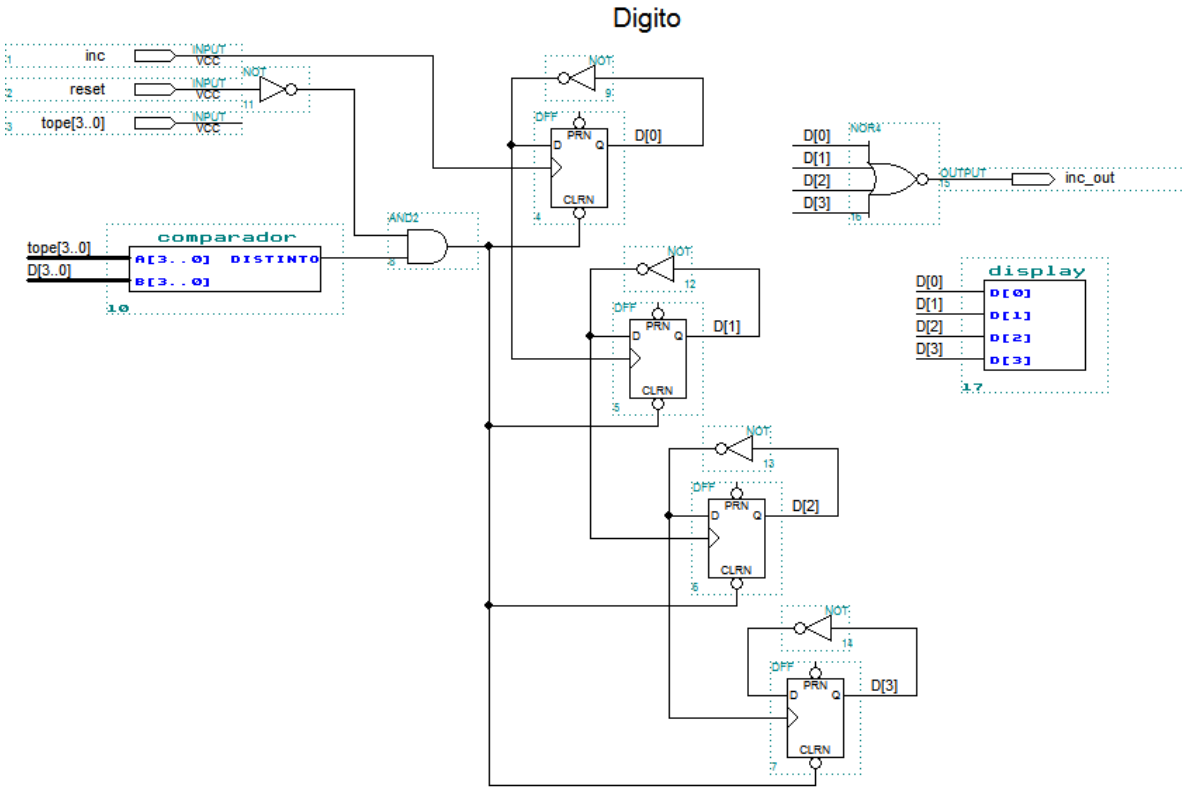
La jerarquía de memoria se basa en los principios de localidad temporal y espacial, los cuales indican algunas generalidades de los programas en el uso de la memoria. El principio de localidad temporal dice que típicamente cuando una posición de memoria es referenciada, vuelve a serlo un tiempo corto después. El principio de localidad espacial dice que cuando una posición de memoria es referenciada, en un tiempo cercano también lo son las posiciones cercanas de memoria.

Problema 1

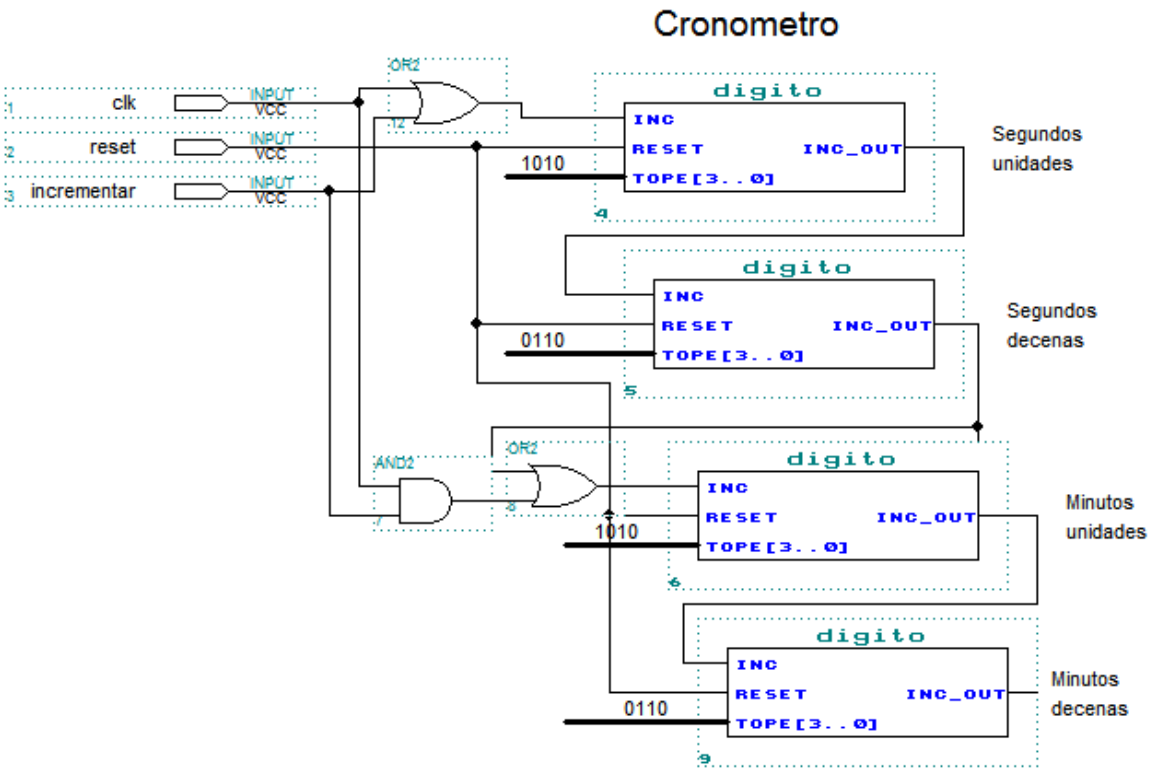
Primero implementamos un comparador que toma como entrada dos buses de 4 bits y genera como salida una señal que indica si los números son distintos:



Luego implementamos un circuito que maneja un digito del reloj. Este circuito toma como entrada una señal de reset, la señal de reloj (flanco ascendente) y hasta que valor contar. Utiliza un módulo 7 segmentos para mostrar el dígito y produce como salida una señal que indica cuando el dígito llegó al tope y volvió a 0.



Por último se implementa el circuito Cronómetro:



Problema 2

Parte a)

```
//  PAD_ACTIVADO   | pad que fue disparado
//  INTENSIDAD     | intensidad del golpe
//  INTENSIDAD_OUT | intensidad de ejecución

unsigned char out_pad_intensidad[4];

void golpe(){
  // obtengo la intensidad de cada pad.
  unsigned short p_i = IN(INTENSIDAD);
  unsigned char p_a = IN(PAD_ACTIVADO);

  unsigned char i_masc = 1;
  unsigned short status_index;
  for (status_index = 0; status_index < 4; status_index++){
    if (p_a & i_masc){
      out_pad_intensidad[status_index] = (p_i & (0x000f << status_index*4)) >> status_index*4;
    };
    i_masc = i_masc * 2;
  };
};
```

```
reloj(){

  unsigned short tmp = 0;
  tmp = out_pad_intensidad[0] | (out_pad_intensidad[1] << 4) | (out_pad_intensidad[2] << 8) |
(out_pad_intensidad[3] << 12);

  OUT( INTENSIDAD_OUT, tmp);
  int i;
  for ( i = 0; i < 4; i++){
    if (out_pad_intensidad[i] > 0)
      out_pad_intensidad[i]--;
  };
};

void main(){
//  desabilito int
// instalo ISR
  int i;
  for ( i = 0; i < 4; i++){
    out_pad_intensidad[i] = 0x00;
// habilito int
  while(1);
};
```

Parte b)

```
out_pad_intensidad db 4 dup 0h

; deja en la variable global out_pad_intensidad las intensidades de los pads
proc far golpe
  push AX
  push BX
  push CX
  push DX
  push SI

  ; leo INTENSIDAD de todos los pads
  mov DX, INTENSIDAD
  in AX, DX
  mov BX, AX  ; BX=p_i
  ; leo PAD_ACTIVADO
  mov DX, PAD_ACTIVADO
  in AL, DX
  mov AH, AL  ; AH=p_i

  mov AL, 1; i_masc
  xor SI, SI; status_index = 0
for:
  cmp SI, 4
  jz fin_for
  ; chequeto si pad fue activado
  mov CL, AH
  and CL, AL
  jz next; si no está activo paso al siguiente
  ; si no da cero, entonces pad activado
  ; preparo máscara para obtener la intensidad del pad1
  mov CX, SI
  shl CL,1
  shl CL,1 ; CL=status_index*4
  mov DX, 0xF
  shl DX, CL ;DX=0xf << (status_index*4)
  and DX, BX
  shr DX, CL
  mov CS:[SI + out_pad_intesidad], DL
```

```
next:
  shl AL,1
  inc SI
  jmp for
fin_for:

  pop SI
  pop DX
  pop CX
  pop BX
  pop AX
  iret
endp

proc far reloj
  push AX
  push CX
  push DX
  push SI

  xor AX,AX  ; AX = tmp = 0
  mov AL, CS:[out_pad_intensidad]
  mov CH, CS:[out_pad_intensidad+1]
  mov CL, 4
  shl CH, CL
  OR AL,  CH
  mov AH, CS:[out_pad_intensidad+2]
  mov CH, CS:[out_pad_intensidad+3]
  shl CH, CL
  OR AH,  CH

  mov DX, INTENSIDAD_OUT
  out DX, AX

  xor SI, SI
for:
  cmp SI, 4
  je fin_for
  cmp byte ptr CS:[SI+out_pad_intensidad], 0
  je fin_if
  dec byte ptr CS:[SI+out_pad_intensidad]
fin_if:
  inc SI
  jmp for
fin_for:
  pop SI
  pop DX
  pop CX
  pop AX

  iret
endp
```