Examen de Arquitectura de Computadoras

22 de febrero de 2016

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 se entiende por distancia entre dos representaciones binarias?.
¿Qué es el Código de Hamming?. Explique con un ejemplo como se construyen las representaciones válidas en dicho código. Indicar si es capaz de detectar y corregir errores. En caso que sea posible, especifique la cantidad de bits que puede corregir.

Pregunta 2

Indique como se forman las direcciones en Intel 8086.
Indique qué valores deben tener los registros de segmento para que los segmentos sean todos consecutivos (no se solapen) y arranquen luego del vector de interrupciones.

Pregunta 3

¿Cuantas ROMs de 2Kx8 se necesitan para construir una ROM de 4Kx4?. Dibujar el circuito correspondiente.

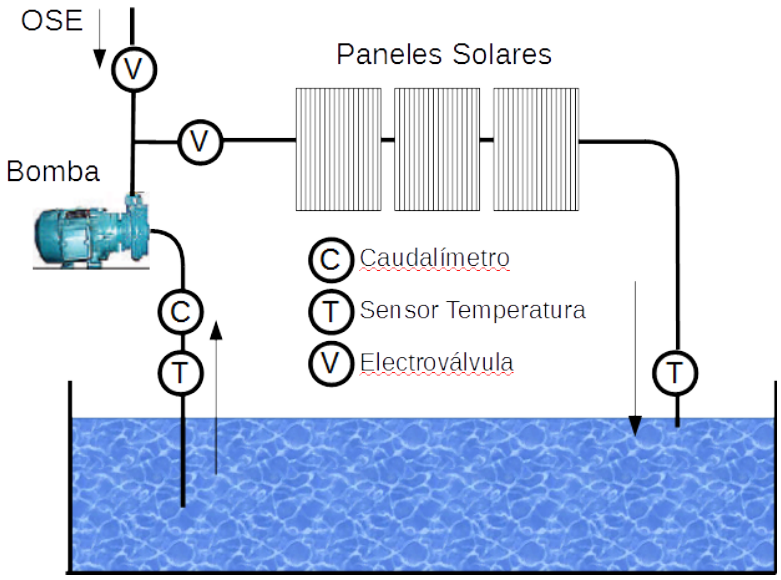
Pregunta 4

Dada la siguiente secuencia de bits indique a qué instrucción microassembler (de MIC-1) corresponde:

```
11001000000100011001000000001000
```

Problema 1

La empresa ***Poolsunheat*** nos ha encargado la programación del controlador para su sistema de calentamiento del agua de una piscina mediante energía solar. El sistema de calentamiento está compuesto por una bomba que hace circular el agua de la piscina por unos paneles solares a los efectos de aumentar la temperatura de la misma, según el siguiente diagrama:



Se dispone de un caudalímetro que permite saber si circula agua, dos sensores de temperatura (uno en la toma de agua y otro en la salida del circuito de calentamiento) y dos electroválvulas (una que controla la entrada de agua desde OSE y otra que controla la salida del agua hacia los paneles solares y luego la piscina).

El controlador es el responsable de llevar adelante las siguientes reglas.

- Al comenzar (el controlador es encendido) debe ponerse en funcionamiento la bomba.
- Mientras la temperatura de salida del agua se mantenga 0,3 °C por encima de la entrada y haya circulación de agua, la bomba debe permanecer encendida.
- Si se apaga la bomba por la condición de temperatura, debe esperarse media hora y ponerse en funcionamiento nuevamente.
- Si encendida la bomba no circula agua durante 1 minuto, deben reiterarse el proceso de puesta en funcionamiento.

El cualquier caso el proceso de puesta en funcionamiento de la bomba implica los siguientes pasos:

- Se debe cebar la bomba, para lo cual se hace circular agua de OSE en sentido inverso por la bomba (apagada) durante 30 segundos. Para esto es necesario cerrar la válvula del caño que va a los paneles solares y la piscina, y abrir la que conecta con OSE.
- Pasados los 30 segundos de cebado hay que encender la bomba, para lo cual la válvula de OSE debe estar cerrada y abierta la de salida hacia los paneles y la piscina.

Nota: en todos los casos se entiende que circula agua si el caudal es de 1 litro por segundo como mínimo. Las medidas de temperatura solamente son válidas mientras circula agua.

Se pide:

Escribir en un lenguaje de alto nivel (preferentemente C) todas las rutinas necesarias para implementar el controlador descrito sobre un procesador dedicado.

Se sabe que:

- El caudalímetro interrumpe cada vez que detecta el pasaje de un litro de agua, invocando a la rutina **caudal()**.
- Los sensores de temperatura son accesibles en los puertos de E/S de 16 bits en las direcciones TEMP_IN y TEMP_OUT y la lectura es en centésimas de grado centigrado.
- Las electroválvulas se accionan en los bits menos significativos del puerto de E/S de 8 bits en la dirección CONTROL (bit 0 = válvula de OSE, bit 1 = válvula de salida). El bit en 0 significa cerrada.
- La bomba se controla con el bit más significativo del mismo puerto de E/S en la dirección CONTROL. El bit en 0 significa apagada.
- Se dispone de un timer que interrumpe con una cadencia de 10 Hz, invocando a la rutina **tiempo()**.

Problema 2

Considere una CPU de 16 bits con memoria principal de 64 KB y una caché de 16KB con correspondencia asociativa de 4 vías, tamaño de bloque 16 bytes y una política de remplazo *FIFO*.

a) Indique como se interpretan las direcciones de memoria, desde el punto de vista del sistema de memoria caché, especificando cuantos bits corresponden a cada campo.

b) Considere que la memoria caché está inicialmente vacía y se realizan de forma consecutiva los accesos a las siguientes direcciones de memoria en el orden indicado: 0x5C20, 0xAB13, 0x5C27, 0x4B1F, 0xCB10, 0xEB14, 0xDB1F.

Para cada una de las direcciones, indique cuál es el valor de cada uno de los campos, indicados en la parte anterior.

Detalle para luego de cada acceso a memoria, cuántas direcciones hay almacenadas en cada conjunto y cuáles son. Especifique en cada caso si el acceso fue un HIT o un MISS.

c) Luego de todos los accesos de la parte anterior, se cambia la política de acceso a *"el accedido hace más tiempo (LRU)"* y se realizan los accesos a las siguientes direcciones de memoria en el orden indicado: 0x4B1E, 0x0B1F.

Añada los nuevos accesos al esquema de la parte anterior.

Solución Pregunta 1

Distancia entre dos representaciones binarias es la cantidad de bits distintos entre ambos.
El código de Hamming es una forma de generar códigos de distancia 3.
Si tenemos 16 objetos binarios a4 a3 a2 a1 se agregan bits de redundancia, en este caso 3.
Se calculan entonces los tres bits de redundancia p1 p2 p3 de la siguiente manera:

$p1 = a4 \oplus a2 \oplus a1$
$p2 = a4 \oplus a3 \oplus a1$
$p3 = a4 \oplus a3 \oplus a2$

Las representaciones válidas son aquellas que cumplen las ecuaciones válidas para p1, p2 y p3.
El código de Hamming es capaz de detectar errores de dos bits y corregir errores de un bit.

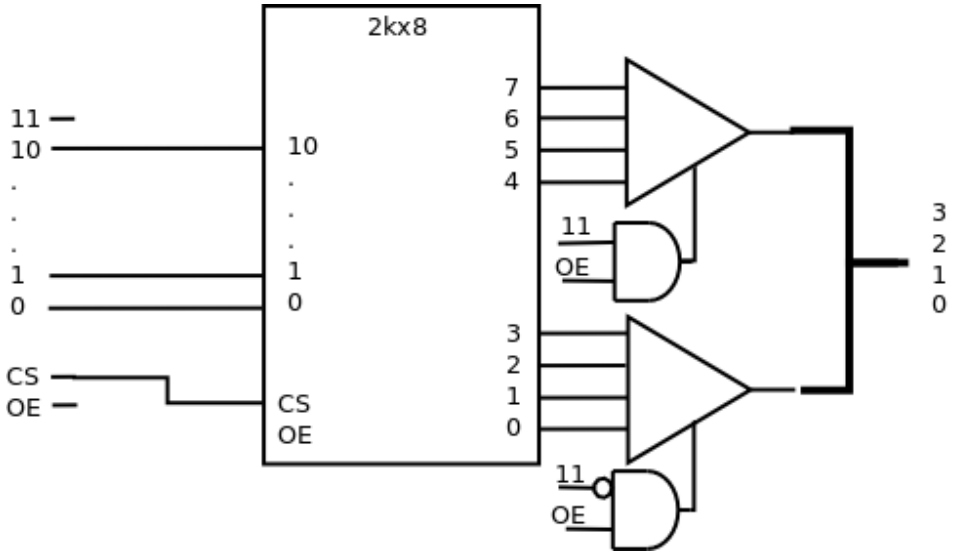
Solución Pregunta 2

Las direcciones en Intel 8086 son segmentadas, esto es, se utiliza un registro de segmento para generar direcciones de 20 bits a partir de registros de 16 de la siguiente forma
Dirección Física = Registro de Segmento * 16 + Desplazamiento

El vector de interrupciones tiene 256 posiciones de 4 bytes ubicado a partir de la dirección 0. Esto hace que el primer segmento deba comenzar en la dirección física 0x00400.
Cada registro ocupa 64kbytes (por los 16 bits de desplazamiento) por lo que el siguiente segmento comienza en la dirección 0x10400.
Para obtener los registros de desplazamiento, dividimos por 16. Si los alineamos como CS, DS, ES y SS los valores son:
CS = 0x0040
DS = 0x1040
ES = 0x2040
SS = 0x3040

Solución Pregunta 3

Se precisa 1.

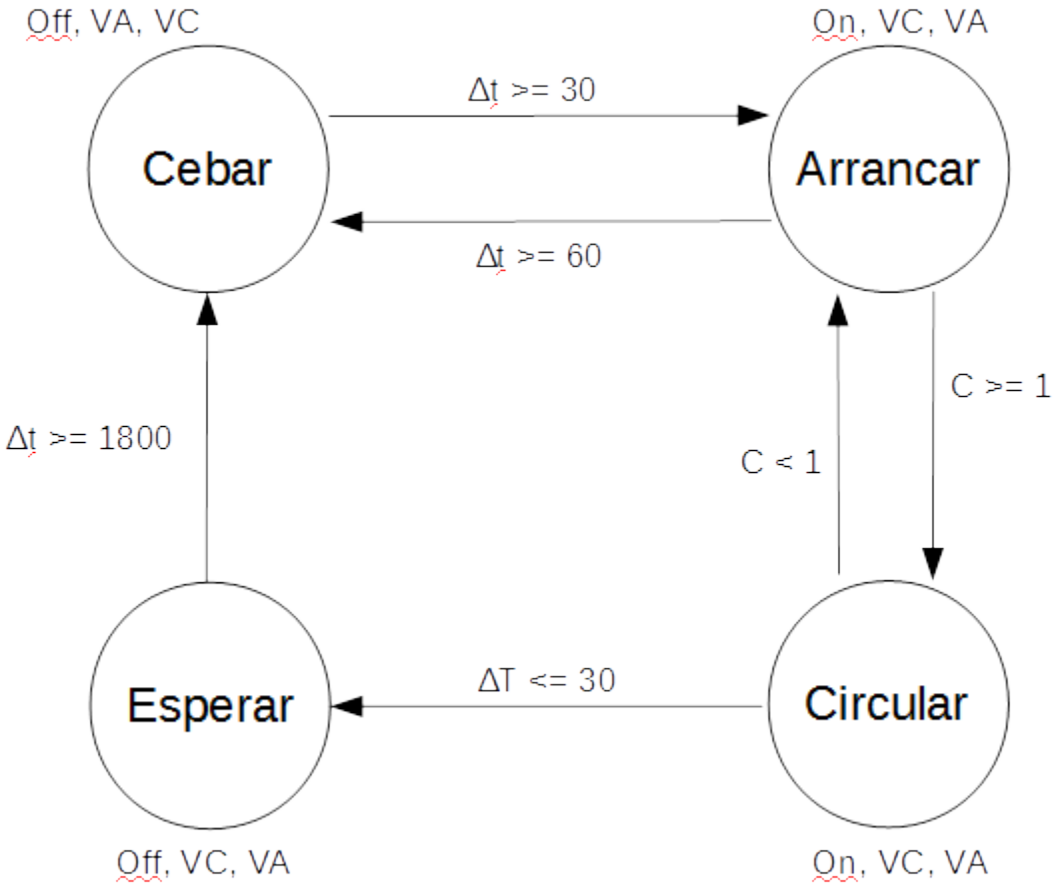


Solución Pregunta 4

```
1 10 01 00 00001 0001 1001 0000 00001000
AC=band(MBR,SMASK);if z goto 0x08
```

Solución Problema 1

Utilizaremos un diseño del sistema basado en una máquina de estados de tipo Máquina de Moore. Consideramos las entradas: ΔT (diferencia de temperatura entre la salida y la entrada de agua, en centésimas de grado centígrado), Δt (tiempo transcurrido en segundos en el estado actual), C (caudal en litros por segundo), y las salidas: Bomba, Válvula de Salida, Válvula OSE .



Como solo nos interesa saber si el caudal es mayor o menor a 1 litro por segundo, lo implementaremos mediante una bandera de condición apropiadamente asignada mediante las rutinas tiempo() y caudal().

```
#define 1SEG        10
#define 30SEG       300
#define 60SEG       600
#define 30MIN       18000
#define 3DECGRADO   30
#define TRUE        1
#define FALSE       0
#define CEBAR       0
#define ARRANCAR    1
#define CIRCULAR    2
#define ESPERAR     3
#define OFFVCVA     0x01
#define ONVAVC      0x82
#define OFFVAVC     0x02

boolean circula_agua;
unsigned int estado, tics_tiempo, tics_caudal;

void interrupt caudal() {
     if (tics_caudal <= 1SEG) hay_caudal = TRUE;
     tics_caudal = 0;
}

void interrupt tiempo() {
     tics_tiempo++;
     if (tics_caudal <= 1SEG) tics_caudal++;          //evito overflow
     if (tics_caudal > 1SEG) hay_caudal = FALSE;
}

void main() {
     unsigned int deltaT;

     estado = CEBAR;
     out(CONTROL, OFFVCVA);
     tics_caudal;
     tics_tiempo = 0;
     pasa_litro = FALSE;
     hay_caudal = FALSE;
     // instalo rutinas de interrupción
     enable();
     while(TRUE) {
          switch (estado){
               case CEBAR:
                    if (tics_tiempo >= 30SEG) {
                         estado = ARRANCAR;
                         out(CONTROL, ONVAVC);
                         tics_caudal = 0;
                         hay_caudal = FALSE;
                         tics_tiempo = 0;
                    }
```

```
                    break;
               case ARRANCAR:
                    if (hay_caudal) {
                         estado = CIRCULAR;
                         tics_tiempo = 0;
                    }
                    if (tics_tiempo >= 60SEG) {
                         estado = CEBAR;
                         out(CONTROL, OFFVCVA);
                         tics_tiempo = 0;
                    }
                    break;
               case CIRCULAR:
                    deltaT = in(TEMP_OUT) - in(TEMP_IN);
                    if (deltaT < 30DECGRADO) {
                         estado = ESPERAR;
                         out(CONTROL, OFFVAVC);
                         tics_tiempo = 0;
                    }
                    if (!hay_caudal) {
                         estado = ARRANCAR;
                         out(CONTROL, OFFVCVA);
                         tics_tiempo = 0;
                    }
                    break;
               case ESPERAR:
                    if (tics_tiempo >= 30MIN) {
                         estado = CEBAR;
                         out(CONTROL, OFFVCVA);
                         tics_tiempo = 0;
                    }
                    break;
          }
     }
}
```

Solución Problema 2

a) 16KB/2^4 = 1024 líneas en el cache. Como cada conjunto tiene 4 líneas, hay 1024/4 = 256 conjuntos. En definitiva, se necesitan 4 bits para el campo “BYTE”, 8 bits para el campo “CONJUNTO” y 4 bits para el campo “TAG”.

|15 - - TAG - - 12 | 11 - - CONJUNTO - - 4 | 3 - - BYTE - - 0|

b)
T_1) 0x5C20: TAG: 5, CONJUNTO: C2, BYTE: 0
T_2) 0xAB13: TAG: A, CONJUNTO: B1, BYTE: 3
T_3) 0x5C27: TAG: 5, CONJUNTO: C2, BYTE: 7
T_4) 0x4B1F: TAG: 4, CONJUNTO: B1, BYTE: F
T_5) 0xCB10: TAG: C, CONJUNTO: B1, BYTE: 0
T_6) 0xEB14: TAG: E, CONJUNTO: B1, BYTE: 4
T_7) 0xDB1F: TAG: D, CONJUNTO: B1, BYTE: F

T_0) Cache vacía.
T_1) MISS. Conjunto C2: 0x5C20.
T_2) MISS. Conjunto C2: 0x5C20. Conjunto B1: 0xAB10.
T_3) HIT. Conjunto C2: 0x5C20. Conjunto B1: 0xAB10. (Se está accediendo al mismo bloque que se almaceno en T_1.
T_4) MISS. Conjunto C2: 0x5C20. Conjunto B1: 0xAB10,0x4B10.
T_5) MISS. Conjunto C2: 0x5C20. Conjunto B1: 0xAB10,0x4B10,0xCB10.
T_6) MISS. Conjunto C2: 0x5C20. Conjunto B1: 0xAB10,0x4B10,0xCB10,0xEB10.
T_7) MISS. Conjunto C2: 0x5C20. Conjunto B1: 0x4B10,0xCB10,0xEB14,0xDB10. (hay un remplazo porque las 4 vías del conjunto están en uso).

c)
T_8) 0x4B1E: TAG: 4, CONJUNTO: B1 (B1), BYTE: E
T_9) 0x0B1F: TAG: 0, CONJUNTO: B1 (B1), BYTE: F

T_8) HIT. Conjunto C2: 0x5C20. Conjunto B1: 0x4B10,0xCB10,0xEB10, 0xDB10. Cambia el LRU
T_9) MISS. Conjunto C2: 0x5C20. Conjunto B1: 0x4B10, 0xEB10 , 0xDB10, 0x0B10 (se reemplaza el bloque 0xCB10 por ser el que fue accedido hace más tiempo).