Examen de Arquitectura de Computadoras

14 de febrero 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

Defina el problema de coherencia de caché e indique tres situaciones en el que se puede manifestarse. Explique cómo se resuelve el problema en el caso de utilizar un sistema de acceso directo a memoria con política write-back.

Pregunta 2

a) Explique el concepto de distancia aplicado a representaciones binarias.
¿Cuál es la distancia entre las siguientes representaciones binarias? Justifique.
000011001001
000111001101

b) ¿Cuál es la cantidad de bits de redundancia mínima necesaria para corregir errores de hasta 1 bit en los bancos de memoria de una máquina de 32 bits de ancho de palabra? Justifique.

Pregunta3

Implemente una ROM de 4Kx4 a partir de una de 1Kx16.

Pregunta 4

a) ¿En qué consiste el direccionamiento de memoria segmentado en 8086?

b) Presente tres formas de lograr la dirección 0xFEDE en el modo segmentado .

Problema 1

La empresa ***IELOU*** está diseñando su modelo de patineta eléctrica y para su control utilizará un microprocesador dedicado.

El sistema de control recibe de un sistema externo la indicación de activación / desactivación. Si la patineta no está activada no debe funcionar y debe quedar frenada.

Estando activada la patineta, al mantener apretado el botón de avanzar el sistema de control debe conectar el moto-generador en modo motor, de forma que la patineta vaya hacia adelante. Si está apretado el botón de freno se debe conectar el moto-generador en modo generador, de forma de aprovechar la energía cinética acumulada para cargar las baterías, a la vez que se produce el frenado. El botón de frenado debe prevalecer sobre el de avanzar.

La Municipalidad ha reglamentado el uso de estos vehículos y ha impuesto límites de velocidad. Por este motivo el diseño incluye un velocímetro que se implementa a partir de una interrupción que ocurre por cada giro completo de una de las ruedas, que tienen una circunferencia de **60 cm**. Cuando la velocidad supera **21.6 km/hora** debe desconectarse el moto-generador y en caso de superar los **43.2 km/hora** se debe conectar el moto-generador en modo generador cada 1 segundo (un segundo conectado, un segundo desconectado), de modo de reducir la velocidad por frenado no continuo.

Se cuenta con las siguientes interrupciones:

- ***command()****:* Se ejecuta cada vez que el sistema externo desea activar/desactivar la patineta.
- ***turn()***: Se ejecuta cada vez que una rueda da un giro completo.
- ***timer()***: Rutina que se ejecuta a una frecuencia de 1 kHz.

También se cuenta con los siguientes puertos:

- Puerto de E/S de 8 bits de solo lectura en la dirección **COMANDO**, donde el bit menos significativo indica el comando enviado por el sistema de activación (1 = activar, 0 =.desactivar). Su contenido solo es válido cuando ocurre la interrupciíon atendida por ***command()***.
- Puerto de E/S de 8 bits de solo lectura en la dirección **BOTONES**, donde el bit b7 (más significativo) indica el estado del botón de avanzar (1 = apretado) y el b6 indica el estado del botón de freno (1 = apretado).
- Puerto de 8 bits de solo escritura en la dirección **MOTOGEN**, en cuyo bit menos significativo se activa el moto-generador (1 = activado) y en su bit más significativo se indica el modo (1 = motor, 0 = generador).

Se pide: implementar en un lenguaje de alto nivel, preferentemente C, todas las rutinas necesarias del sistema de control de la patineta eléctrica.

Nota: *se asume que las ruedas no patinan en ningún momento.*

Problema 2

La empresa ***ArquiCola*** está diseñando una máquina expendedora de mini-botellas de refrescos de 200ml.
Cada refresco cuesta $16 y la máquina acepta monedas de $5 y $10. El funcionamiento del sistema es el siguiente:

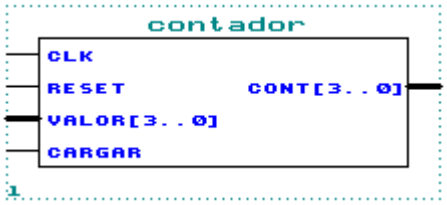
- Cuando la máquina esta libre la entrada de monedas queda automáticamente disponible para comenzar a insertarlas.
- El usuario va colocando una a una monedas hasta acumular más de $15. En ese momento la máquina dispensa una botella y devuelve el cambio en monedas de $1 a razón de una moneda por segundo. Mientras que la máquina está dispensando la botella y devolviendo el cambio la entrada de monedas queda cerrada impidiendo insertar nuevas monedas
- Una vez que se devolvió todo el cambio la máquina queda pronta para comenzar un nuevo servicio.

El sistema cuenta con un sensor en la entrada de monedas que mantiene una señal de 2 bits MN con el siguiente significado:

M	**N**	**Significado**
0	0	No se detecta moneda
0	1	Se detecta moneda de $5
1	0	Se detecta moneda de $10
1	1	No usado (no puede ocurrir)

Cuando se inserta una moneda la señal MN mantiene el valor adecuado por 1 período de reloj completo.
Además se cuenta con las siguientes salidas:
- **Dispensar** dispensa una botella
- **Devolver** devuelve monedas de $1 a razón de una moneda por segundo
- **Habilitar** habilita la entrada de monedas

Se pide: diseñar un circuito que modele este sistema. Se dispone de flip flops tipo D y compuertas básicas. Además se cuenta con un contador descendente de 4 bits (no cíclico), con inicialización sincrónica paralela (se carga el valor de las entradas VALOR[3..0] en el momento de un flanco en la entrada CARGAR).



El reloj general del sistema trabaja a una frecuencia de 1Hz.

Respuesta Pregunta 1:

El problema de coherencia de caché sucede cuando se pierde el sincronismo entre el contenido de la caché y el de la memoria principal, afectando la consistencia en los datos. Lo importante no es la falta de sincronismo, si no el efecto no deseado sobre la consistencia de los datos vistos por distintas partes del sistema que los utilizan.

El problema se puede dar en distintas situaciones:

1. Cuando se utilizan sistemas de acceso directo a memoria (DMA) para optimizar los intercambios de datos entre los dispositivos de entrada/salida y la memoria principal, el problema se da, por ejemplo, cuando el DMA escribe en la memoria principal en un bloque que está almacenado en la caché.

2. Cuando se utiliza cache con write-back, la cache pierde el sincronismo con la memoria principal cada vez que se realizan escrituras, hasta que se reemplaza el bloque y se actualiza en memoria. El problema de consistencia se puede dar cuando el DMA lee en un bloque almacenado en la caché que fue modificado por una escritura en el cache.

3. En sistemas multi-procesador con memoria compartida se dan los mismos problemas que suceden al usar DMA, cuando un procesador lee o escribe en bloques de memoria almacenados en la cache de otro procesador y además, en el caso de write-back, se puede dar la situación que el mismo bloque de memoria esté simultáneamente en más de una cache y se desincronizará su valor cuando se escribe en las caches.

Para el caso del cache con write back, si el problema ocurre en una lectura, el problema se resuelve deteniendo el DMA para actualizar la memoria con el contenido de la caché. Si se trata de una escritura del DMA, el problema se resuelve invalidando la entrada del caché y el próximo acceso al bloque producirá un fallo (“miss”) provocando la actualización de la cache.

Respuesta Pregunta 2:

a) La distancia entre dos representaciones binarias se define como el número de bits distintos entre los dos códigos.
La distancia entre las dos representaciones binarias es 2. Los bits número cuatro y diez son los únicos diferentes.

b) La cantidad de bits 'p' de redundancia requeridos, de modo de tener la capacidad de corrección de errores, para un código de tamaño 'k' está dado por la siguiente ecuación:

$$2^p >= p + k + 1$$

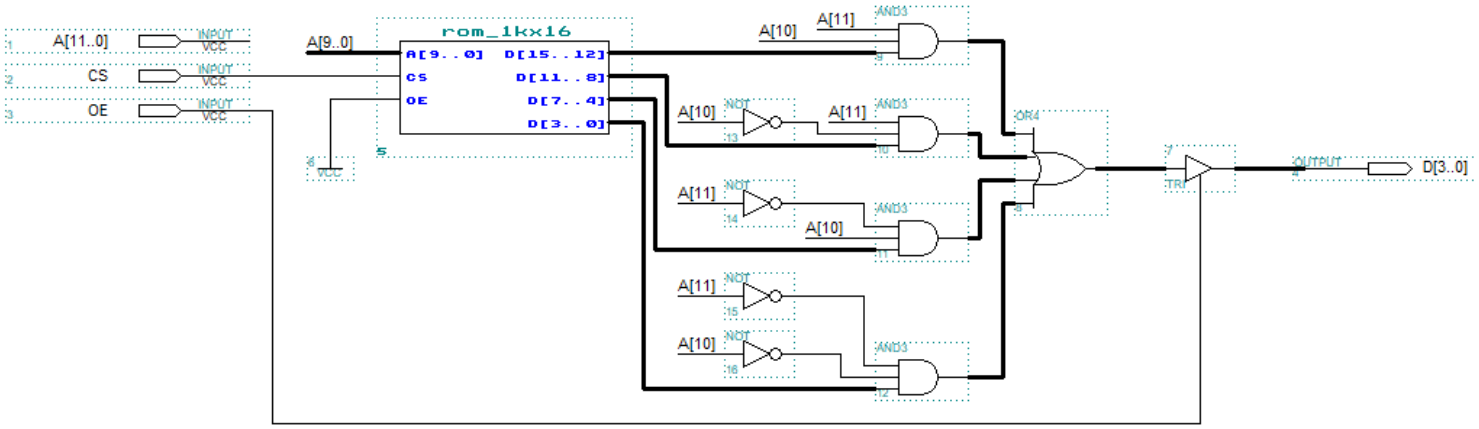
El razonamiento detrás de esta afirmación es que para poder identificar el bit que tiene error dispondremos de 2^p combinaciones posibles, generadas por los p bits. Con esas combinaciones debemos poder señalar cual es el bit errado de los p + k bits que tendrá el código completo (el original más los bits de redundancia agregados). Además precisaremos tener una combinación para indicar que no hay error y de ahí surge que el lado derecho de la expresión es p + k + 1.

En el caso de un código de 32 bits, la cantidad mínima de bits de redundancia para que se cumpla la condición sería 6, ya que:

2^6 >= 6 + 32 + 1

Por lo tanto 6 es la cantidad de bits de redundancia mínima para este caso.

Respuesta Pregunta 3:



Respuesta Pregunta 4:

a)
Consiste en construir la dirección de memoria de 20 bits (el 8086 direcciona hasta 1 Mbyte de memoria principal) a partir de dos elementos de 16 bits cada uno (que es el ancho de palabra de la arquitectura): SEGMENTO y DESPLAZAMIENTO.

La dirección física se calcula de la siguiente manera

DIRECCION = SEGMENTO * 16 + DESPLAZAMIENTO

b)
0x0FED:0x000E
0x0FE0:0x00DE
0x0F00:0x0EDE

Solución Problema 1:

Debemos convertir los limites de velocidad (km / hora) a límites de giros / segundo que es lo que podemos medir.

Como la rueda es de 60 cm de circunferencia, la patineta recorre 60 cm en cada giro, por lo que su velocidad es de 60 cm x giro / seg.

Limite de 21,6 km / hora:

> 60 cm x giro / seg < 21,6 km / hora = 21,6 x 1000 x 100 cm / hora = 2160000 cm / 60 x 60 seg
> giro / sec < 2160000 / 216000 = 10, es decir un giro cada 100 ms

Limite de 43,2 km / hora, es el doble de giros / segundo que en el limite anterior, es decir un giro cada 50 ms.

Debemos pasar los limites en giros / seg a “tics” del timer. Dado que cada tic del timer ocurre cada 1 ms (la frecuencia es 1 kHz), nos quedan los limites en 100 y 50 tics respectivamente.

```
#define TICS_LIMITE_21.6 100
#define TICS_LIMITE_43.2 50
#define BIT_COMANDO      0x01
#define ACTIVAR          0x01
#define BOTON_AVANZO     0x80
#define BOTON_FRENO      0x40
#define AVANZO           0x80
#define FRENO            0x40
#define AVANZAR          0x80
#define FRENAR           0x01
#define DESCONECTAR      0x00

boolean activada;

int tics, tics_giro, velo_tics;


void interrupt command() {

     if ((in(COMANDO) & BIT_COMANDO) == ACTIVAR) activada = TRUE;
     else activada = FALSE;
}


void interrupt turn() {

     velo_tics = tics_giro;
     tics_giro = 0;
}
```

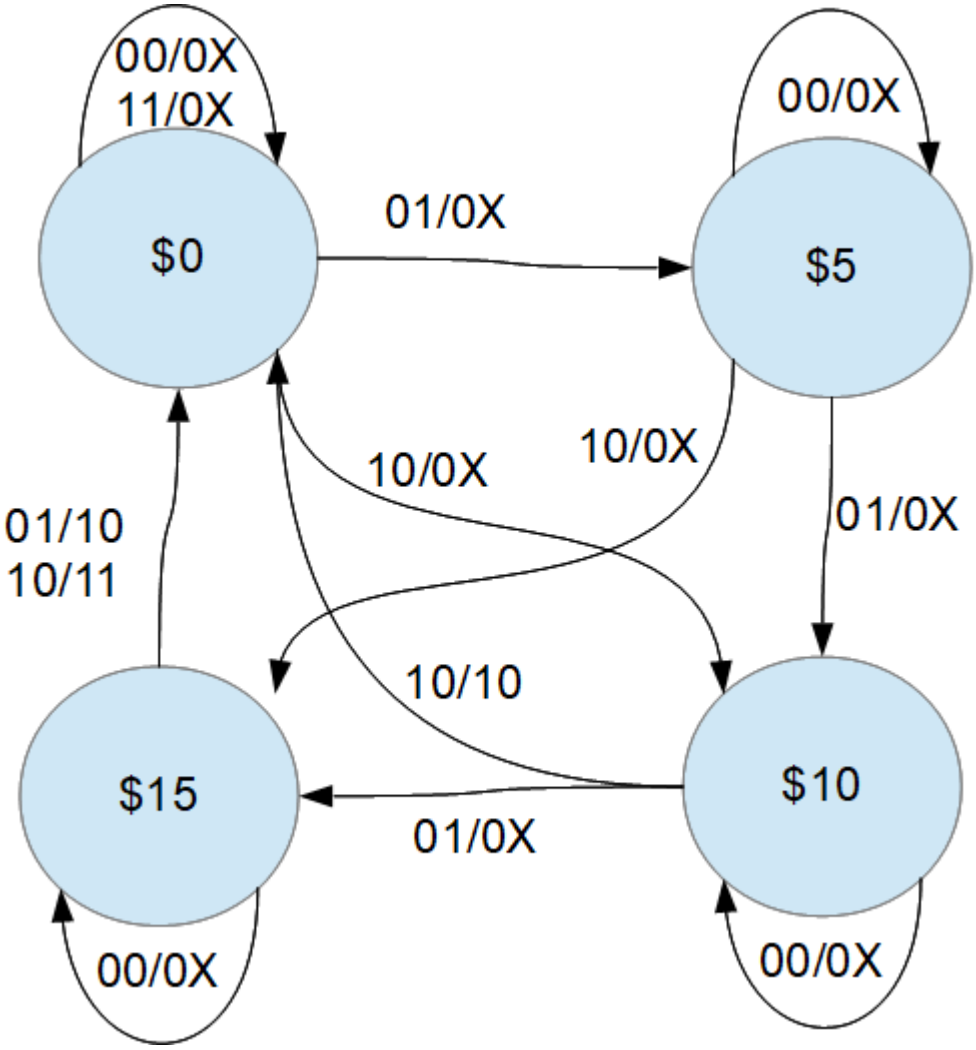
```
void interrupt timer() {

     tics++;
     tics_giro++;
     if (activada) {
          if ((in(BOTONES) & BOTON_FRENO) == FRENO)
               out(MOTOGEN, FRENAR);                         // freno
          else if ((velo_tics <= TICS_LIMITE_21.6) &&
                              (velo_tics > TICS_LIMITE_43.2))
                    out(MOTOGEN, DESCONECTAR);               // desconecto
               else if (velo_tics <= TICS_LIMITE_43.2)
                         if (tics % 2000 < 1000)
                              out(MOTOGEN, FRENAR);          // freno
                         else
                              out(MOTOGEN, DESCONECTAR);     // desconecto

                    else if ((in(BOTONES) & BOTON_AVANZO) == AVANZO)
                              out(MOTOGEN, AVANZAR);         // avanzo
                         else
                              out(MOTOGEN, DESCONECTAR);     // desconecto
          }
     }
     else out(MOTOGEN, FRENAR);              // dejo frenada la patineta
}


void main() {

     // instalo rutinas de interrupción
     activada = FALSE;
     tics = 0;
     tics_giro = 0;
     velo_tics = 0;
     enable();
     while (TRUE);
}
```

Solución Problema 2:

Para resolver el problema vamos a diseñar una máquina de estados que mantenga la cantidad de dinero depositada en el sistema y utilizaremos el contador descendente para devolver el vuelto y determinar si la máquina está habilitada para ingresar nuevas monedas.
Para ello multiplexaremos la información recibida en las entradas MN con la información de estado del contador para obtener una nueva señal de 2 bits OP de la siguiente manera:

Contador	**MN**	**OP**	**Significado**
0	MN	MN	Máquina habilitada
<>0	XX	11	Devolviendo monedas

Construimos ahora la máquina de estados:
Entradas: O P
Salidas: Dispensar Cargar9



Dado que tenemos 4 estados precisamos 2 flip flops. Codificamos los estados de acuerdo a la siguiente tabla:

Estado	Q1 Q0
$0	0 0
$5	0 1
$10	1 0
$15	1 1

Y obtenemos la siguiente tabla de transiciones y salidas:

Estado Q1 Q0	OP	Prox. Estado D1 D0	Dispensar	Cargar9
00	00	00	0	X
00	01	01	0	X
00	10	10	0	X
00	11	00	0	X
01	00	01	0	X
01	01	10	0	X
01	10	11	0	X
01	11	XX	X	X
10	00	10	0	X
10	01	11	0	X
10	10	00	1	0
10	11	XX	X	X
11	00	11	0	X
11	01	00	1	0
11	10	00	1	1
11	11	XX	X	X

Ahora minimizamos mediante diagramas de Karnaugh:

Q1Q0/OP	00	01	11	10
00	0	0	0	1
01	0	1	X	1
11	1	0	X	0
10	1	1	X	0

D1=Q1.!O.!P + !Q1.Q0.P+ Q1.!Q0.P + !Q1.O.!P

Q1Q0/OP	00	01	11	10
00	0	1	0	0
01	1	0	X	1
11	1	0	X	0
10	0	1	X	0

D0= Q0.!O.!P + !Q0.!O.P+!Q1.Q0.O

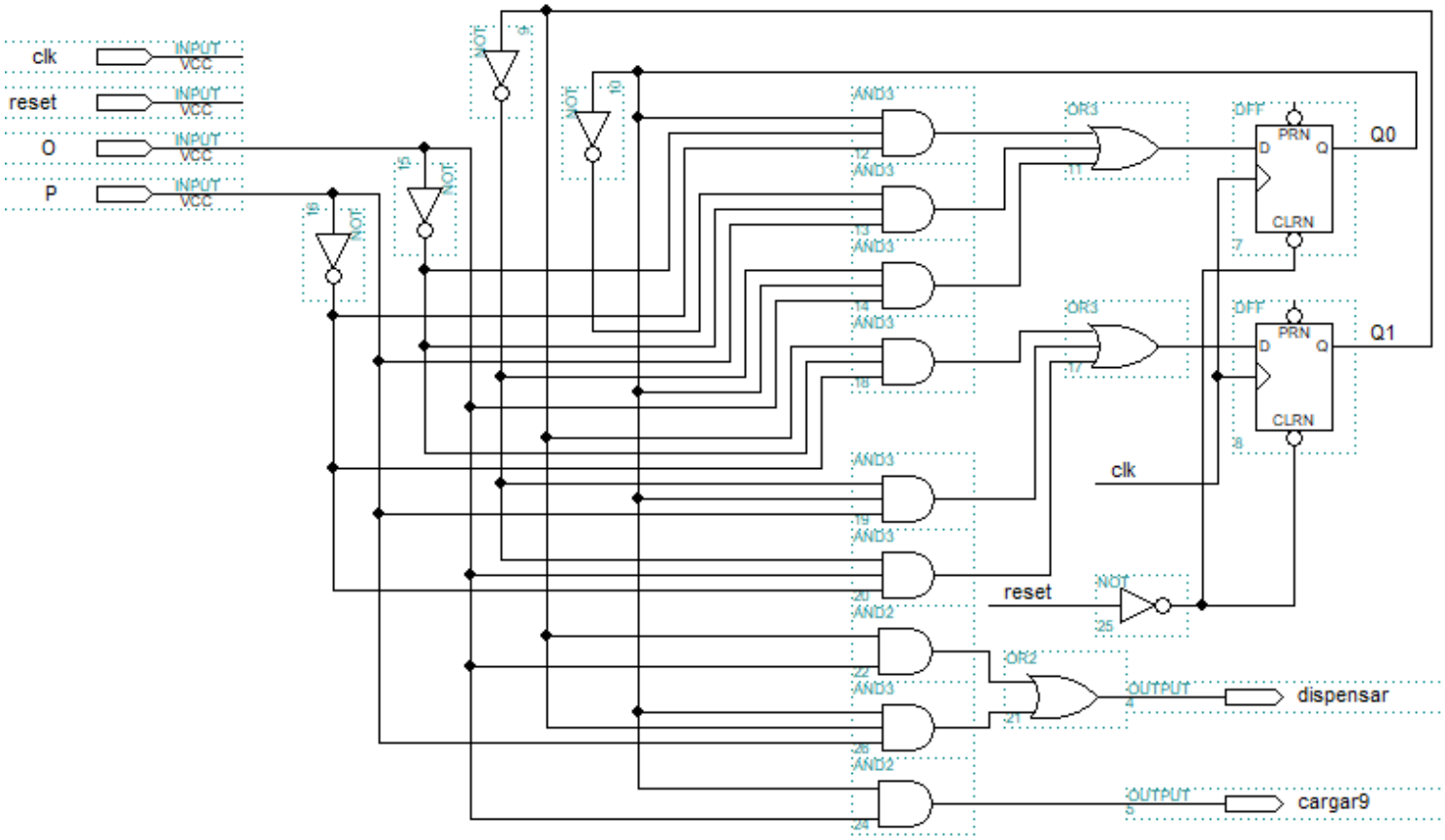
Q1Q0/OP	00	01	11	10
00	0	0	0	0
01	0	0	X	0
11	0	1	X	1
10	0	0	X	1

Dispensar= Q1.O + Q1.Q0.P

Q1Q0/OP	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	0	X	1
10	X	X	X	0

Cargar9= Q0.O

Ahora armamos el circuito de la máquina de estados:



Dibujamos el circuito final: