Examen de Arquitectura de Computadoras

23 de julio del 2012

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.
- 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

Describa los componentes esenciales de la arquitectura Von Neumann así como los elementos internos a la CPU. ¿Cuál es la diferencia principal de las arquitecturas RISC respecto a las CISC?. Enumere además otras dos características que permitan diferenciar dos computadoras Von Neumann.

Pregunta 2

Los procesadores Intel 8086 poseen registros de segmento. ¿Qué son?. ¿Qué ventaja otorgan?. ¿Cuál es la dirección de memoria física expresada por ES:BX si BX = 0x0026 y ES = 0x0010?

Pregunta 3

Un *pipeline* de ejecución permite mejorar el desempeño de un procesador. ¿Cuál es el fundamento detrás de esto?. ¿Qué problemas perjudican el rendimiento en un procesador con *pipeline* y por qué?. Presente un caso que ilustre esto último.

Pregunta 4

Describa el funcionamiento de una memoria caché con Correspondencia Asociativa por Conjuntos. ¿Cómo se determina un caché hit?. ¿Cómo se determina que lugar en caché ocupa un bloque de memoria?. Enumere sus ventajas respecto a Correspondencia Directa. *Nota: tome a modo de referencia una arquitectura de 16 bits para direccionamiento, con una memoria caché de 4 KB dividida en líneas de de 8 bytes.*

Problema 1

La serie de Fibonacci es la sucesión infinita de números naturales definida por

$$F_0 = 1$$
$$F_1 = 1$$
$$F_{n+2} = F_n + F_{n+1}$$

Los primeros 7 valores de dicha serie son {1, 1, 2, 3, 5, 8, 13}.

Se desea construir un circuito secuencial que genere los primeros 7 valores de la serie de Fibonacci. El circuito deberá comenzar la serie cuando es reseteado y pasado el tiempo i-ésimo deberá tener como salida el valor i-ésimo de la serie de Fibonacci. Una vez que llegue al valor 13 deberá mantener este valor hasta que se reinicie el circuito.

Se pide:

Diseñar dicho circuito en base a *flip-flops* tipo D y compuertas básicas, utilizando la metodología vista en el curso.

Problema 2

Se desea implementar el sistema de control de una máquina *tragamonedas*. Se considera una *tragamonedas* "clásica" con tres ruedas iguales, con figuras que comienzan a girar al accionar una palanca y se van deteniendo de a una, en secuencia, de derecha a izquierda. Se acierta el juego cuando las tres ruedas se detienen en la misma figura, en cuyo caso la maquina paga el premio.

La máquina dispone de los siguientes elementos:

- Un detector de monedas que genera un pedido de interrupción cada vez que ingresa una moneda a la maquina. La rutina que se invoca para atender la interrupción se llama `moneda()`.
- Una puerta de expulsión de monedas, que devuelve una moneda al escribir en el byte de E/S en la dirección `EXPULSAR`.
- Un sensor de la palanca de disparo de la máquina. El sensor es accesible a través del bit menos significativo del byte de E/S de solo lectura en la dirección `PALANCA` (el bit en 1 significa que se accionó la palanca, y al leerlo se vuelve a 0).
- Tres ruedas giratorias con figuras, que comienzan a girar todas a la vez al escribir en el byte de E/S de solo escritura en la dirección `RUEDAS`. Las ruedas se detienen, en forma independiente, escribiendo un "1" en el bit correspondiente de los tres bits menos significativos del byte de E/S, de solo escritura, en la dirección `FRENOS` (el menos significativo corresponde a la rueda de más a la derecha).
- Un sensor que genera una interrupción cada vez que las ruedas cambian de figura (se supone las ruedas sincronizadas, es decir cambian de figura al mismo tiempo). La rutina invocada al atender la interrupción es `figuras()`.

Se pretende que la maquina acepte una sola moneda por juego (si se ingresan más deberán ser devueltas). El juego comienza al accionar la palanca (siempre que antes haya sido ingresada una moneda). Al iniciarse el juego las ruedas deben comenzar a girar y luego deben detenerse en secuencia (de derecha a izquierda) luego que haya pasado un cierto número aleatorio de figuras. El número correspondiente a la segunda rueda debe contarse a partir del momento en que se detiene la primera. Igual para la tercera respecto a la segunda.

Se dispone de dos rutinas pre-programadas:

- `random()` que devuelve un número aleatorio.
- `pagar()` que verifica la posición final de las ruedas y abona el premio en caso de acierto.

Se pide:

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

Solución

Pregunta 1

Los componentes esenciales de una arquitectura Von Neumann son: la memoria (utilizada tanto para instrucciones como para datos), la CPU, la E/S y el bus que las interconecta. Los elementos internos de la CPU en una arquitectura Von Neumann son la unidad aritmética lógica (ALU), la unidad de control, el banco de registros y el bus que los interconecta.

La arquitectura RISC se define en contraposición a la arquitectura CISC. Siendo su principal objetivo definir un set reducido de instrucciones, las cuales sean de fácil decodificación y ejecución por el CPU. El objetivo de diseño de un arquitectura RISC es ejecutar una instrucción por ciclo de máquina.

Otras características que permiten diferenciar dos computadoras Von Neumann son el set de intrucciones y su formato, modos de direccionamiento, cantidad de registros y su organización.

Pregunta 2

Son registros utilizados para generar direcciones de memoria al momento de realizar accesos a la misma. Permiten acceder a un espacio de memoria mayor a los 64Kb otorgados por los registros de 16 bits del procesador. La dirección se construye mediante un segmento y un desplazamiento dentro del segmento, permitiendo direcciones de 20 bits.

La dirección de memoria física se construye de esta manera:

ES*16 + BX = 0x10 * 0x10 + 0x26 = 0x126

Pregunta 3

El fundamento detrás del pipeline es la línea de montaje de Henry Ford, que busca minimizar el tiempo ocioso de las diferentes unidades funcionales de un procesador al trabajar estas en paralelo en diferentes instrucciones.

Permite mejorar el desempeño de un procesador al poder ejecutar un conjunto de instrucciones en un tiempo menor al que le llevaría si se tomase el tiempo individual de cada instrucción.

Los problemas que ocurren en arquitecturas basadas en pipeline son los llamados riesgos (hazards), que impiden que se aproveche la capacidad del procesador de ir adelantando la ejecución de las siguientes instrucciones.

Un caso de esto son los hazzard de dependencia de datos que surge cuando una instrucción depende de los resultados de una instrucción anterior y en consecuencia no puede avanzar en su procesamiento hasta que la anterior no esté pronta.

Pregunta 4

En una memoria cache con correspondencia asociativa por conjuntos la función de correspondencia establece que cada bloque de memoria tiene asociado un conjunto de líneas de la memoria cache y puede estar almacenado en cualquiera de las líneas del conjunto.

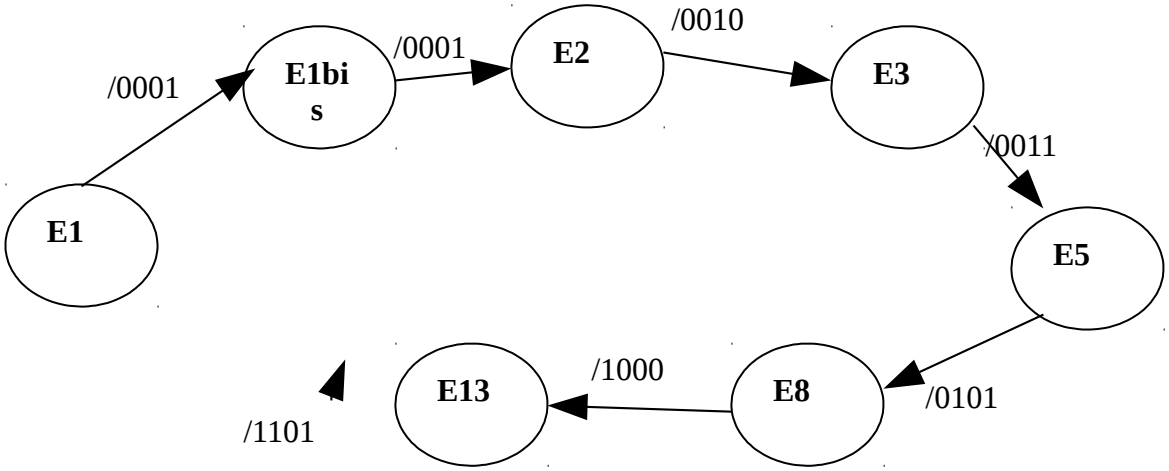
Un cache hit se determina comparando los TAGs de los elementos almacenados en el conjunto.

El lugar que ocupa en la cache un bloque de memoria se determina según el número de conjunto y el bloque asignado como disponible por el algoritmo de remplazo.

Con respecto a correspondencia directa presenta una mejora en el hit rate ya que se dispone de más de un lugar donde se puede almacenar cada bloque en la cache además permite implementar políticas.

Problema 1

Primero diseñamos la máquina de estados que computa los primeros 7 valores de la serie. La máquina no tendrá entradas y producirá como salida el número actual en la serie ($S_3S_2S_1S_0$).



Ahora realizamos la tabla de estados y transiciones

Tenemos 7 estados por lo que precisaremos 3 flip flops:

Estado	Q_2	Q_1	Q_0	Estado	Q_2	Q_1	Q_0
E1	0	0	0	E5	1	0	1
E1bis	0	0	1	E8	1	1	1
E2	0	1	0	E13	1	0	0
E3	0	1	1				

Ahora realizamos la tabla de salidas y transiciones:

Estado			Próximo Estado			Salida			
Q_2	**Q_1**	**Q_0**	**D_2**	**D_1**	**D_0**	**S_3**	**S_2**	**S_1**	**S_0**
0	0	0	0	0	1	0	0	0	1
0	0	1	0	1	0	0	0	0	1
0	1	0	0	1	1	0	0	1	0
0	1	1	1	0	1	0	0	1	1
1	0	0	1	0	0	1	1	0	1
1	0	1	1	1	1	0	1	0	1
1	1	0	X	X	X	X	X	X	X
1	1	1	1	0	0	1	0	0	0

Y procedemos a realizar los diagramas de Karnaugh

$Q_2\backslash Q_1Q_0$	00	01	11	10
0	0	0	1	0
1	1	1	1	X

D2=Q2 + Q1.Q0

$Q_2\backslash Q_1Q_0$	00	01	11	10
0	0	1	0	1
1	0	1	0	X

D1=!Q1.Q0 + Q1.Q0

$Q_2\backslash Q_1Q_0$	00	01	11	10
0	1	0	1	1
1	0	1	0	X

D0=!Q2.Q1 + !Q2.!Q0 + Q2.!Q1.Q0

$Q_2\backslash Q_1Q_0$	00	01	11	10
0	0	0	0	0
1	1	0	1	X

S3=Q2.!Q0 + Q2.Q1

$Q_2\backslash Q_1Q_0$	00	01	11	10
0	0	0	0	0
1	1	1	0	X

S2=Q2.!Q1

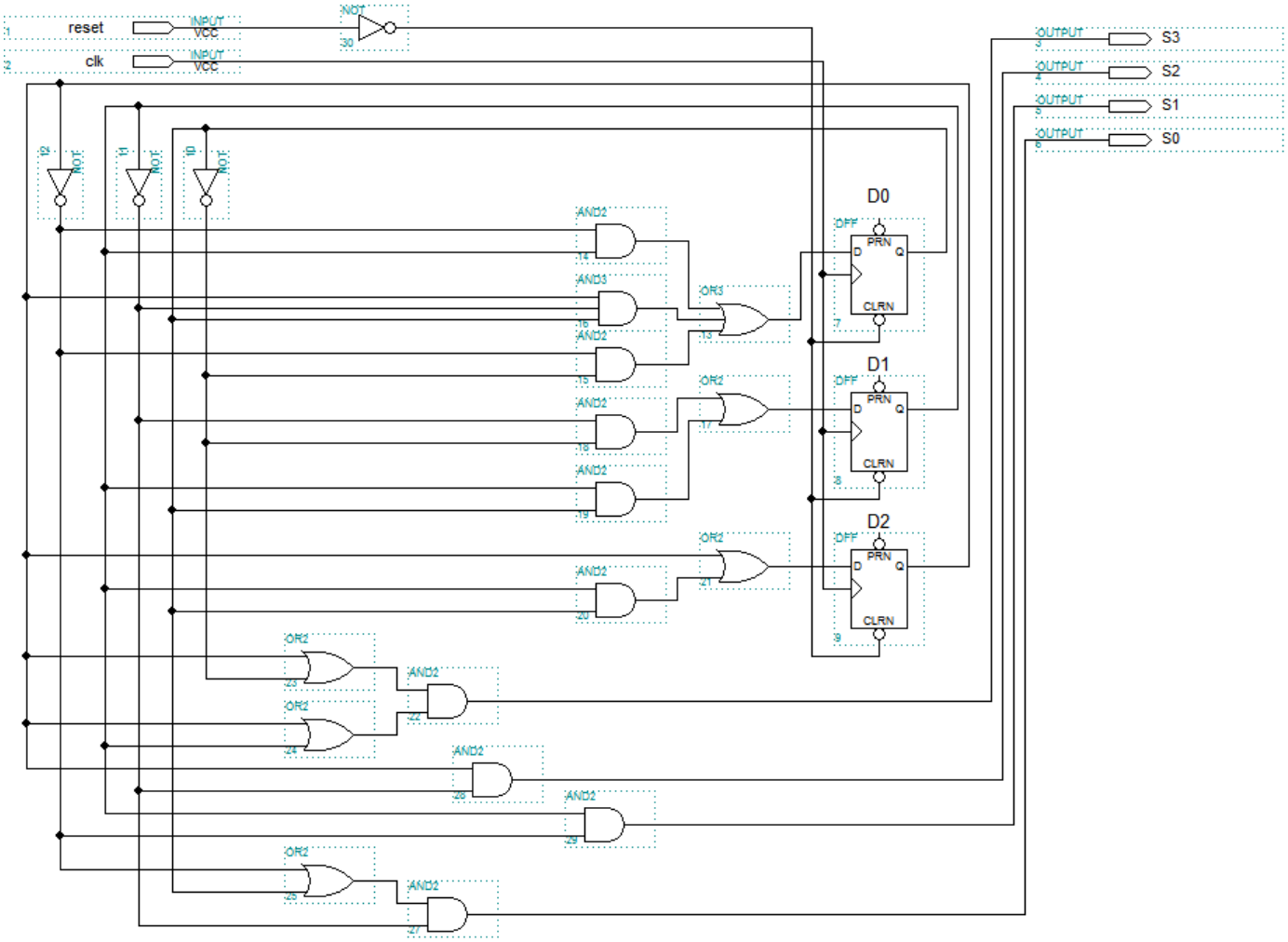
$Q_2\backslash Q_1Q_0$	00	01	11	10
0	0	0	1	1
1	0	0	0	X

S1=!Q2.Q1

$Q_2\backslash Q_1Q_0$	00	01	11	10
0	1	1	1	0
1	1	1	0	X

S0=!Q1 + !Q2.Q0

Ahora dibujamos el circuito:



Problema 2

```
#define    TRUE       1
#define    FALSE      0

int hayMoneda;

void interrupt moneda() {
     if (hayMoneda) out(EXPULSAR, 0x00);        // solo una moneda
     else hayMoneda = TRUE;
}


void interrupt figuras() {
     cuenta--;
}

void main() {
     int cuenta;
```

```
     //instalo interrupciones
     enable();
     while(TRUE) {
          hayMoneda = FALSE;
          while !(hayMoneda);                    // espero moneda
          while ((in(PALANCA) && 0x01) == 0);    // espero accionen la palanca
          out(RUEDAS, 0x00);                     // ruedas girando
          cuenta = random();
          while (cuenta > 0);
          out (FRENOS, 0x01);                    // freno rueda derecha
          cuenta = random();
          while (cuenta > 0);
          out (FRENOS, 0x03);                    // freno rueda del medio
          cuenta = random();
          while (cuenta > 0);
          out (FRENOS, 0x07);                    // freno rueda izquierda
          pagar();                               // verifico y pago si gana
     }
}
```