Examen de Arquitectura de Computadoras

22 de julio de 2022

Instrucciones:

- **Indique su nombre, apellido y número de cédula en todas las hojas que entregue.**
- **Escriba las hojas de un solo lado. Empiece cada ejercicio en una hoja nueva.**
- **Las hojas deben estar numeradas y en la primer hoja debe escribirse el total de hojas entregadas.**
- **Apague su celular. No puede utilizar material ni calculadora.**
- **La duración del examen es de tres horas, incluyendo el tiempo para completar sus datos.**
- **Para aprobar debe contestar correctamente un ejercicio entero y dos preguntas.**

Pregunta 1

a) Dibuje el circuito de un decodificador de 3 bits.
b) Utilizando el decodificador de la parte a) y compuertas OR, implemente la siguiente función lógica combinatoria:

f(a,b,c) = ab + !ac

Pregunta 2

Describa las entradas de un flip-flop D, indicando si son sincrónicas o asincrónicas. Explique las diferencias de funcionamiento entre un flip-flop D con señal de control por nivel y un flip-flop D con señal de control por flanco.

Pregunta 3

Describa el problema de coherencia de caché y explique cómo es que se puede producir cuando se utiliza un sistema de acceso directo a memoria con política “write-back”.

Pregunta 4

Para representar números en base 64, se utiliza la siguiente tabla:

Decimal	Base 64	Decimal	Base 64	Decimal	Base 64	Decimal	Base 64
0	A	26	a	52	0	62	+
1	B	27	b	53	1	63	/
2	C	28	c	54	2		
..	..	..	..	..	..		
24	Y	50	y	60	8		
25	Z	51	z	61	9		

a) Convierta el número z8B+Z de base 64 a base 2.
b) Convierta el número 0x5ACFD de base 16 a base 64.

Problema 1

a) Construya una memoria ROM de $2^{18} \times 10$ a partir de la cantidad mínima de chips de memorias de $2^{15} \times 20$ que sea posible.

b) Programe el contenido de la ROM de $2^{18} \times 10$ construida en la parte a) para simular el comportamiento de una ALU con una entrada compuesta por dos bits de operación y dos operandos de 8 bits, y una salida compuesta por un bit de Z, un bit de Overflow y 8 bits para el resultado, con el siguiente formato:

Entradas

- Bit 0 y bit 1 :: seleccionan la operación con el siguiente formato:
  - 00 → suma en complemento a 2
  - 01 → AND bit a bit
  - 10 → OR bit a bit
  - 11 → NOT A (solo para el operando A)
- Bit 2 al bit 9 :: operando A
- Bit 10 al bit 17:: operando B

Salidas

- Bit 0 :: bandera Z
- Bit 1 :: bandera overflow (solo aplica para la operación 00)
- Bit 2 al bit 9 :: resultado de la operación

Problema 2

La empresa Fit-Flop le ha encomendado la programación de un microcontrolador para su nueva línea de caminadores domésticos (*tipo D*). El caminador se compone de una cinta potenciada por un motor, varios botones para controlar el funcionamiento y un display donde se despliega información, por ejemplo el tiempo de caminata o la velocidad alcanzada.

El caminador dispone de 7 botones para controlar la velocidad. Al presionar cada botón, la velocidad cambia a 11, 9, 7, 5, 3, 1 o 0 km/h, respectivamente. Los 7 botones son accesibles en los bits 6 al 0 del puerto de entrada/salida de solo lectura de un byte, BOTONES. El contenido del puerto es eliminado una vez leído. Al presionar un botón se activa la rutina de interrupción hay_boton(). Se asume que como máximo un único bit de BOTONES puede estar encendido en cada lectura.

El motor se enciende al escribir un 1 en el bit menos significativo del byte de entrada/salida de solo escritura MOTOR y se apaga colocando un 0.

Toda vez que la velocidad de la cinta se mantiene fuera del rango de seguridad durante 30 segundos, se debe encender un led de alarma en el display. La alarma debe apagarse en el momento en que se vuelve al rango de seguridad. La tolerancia permitida está determinada por la constante TOL_VEL (expresada en km/h). Adicionalmente, el display debe mostrar el tiempo transcurrido desde que el motor se puso en funcionamiento y la velocidad de movimiento de la cinta. El display se debe actualizar una vez por segundo, escribiendo en dos puertos de entrada/salida de solo escritura de 16 bits, DISPLAY_HI y DISPLAY_LO, con el siguiente formato:

- El bit 15 de DISPLAY_HI controla el led de alarma
- El bit 14 de DISPLAY_HI indica si el motor está en funcionamiento
- En los bits 13...7 de DISPLAY_HI se indica la velocidad de la cinta durante el último segundo, en décimas de km/h (por ejemplo, una velocidad de 9,5km/h se indica como 95)
- Los bits 6...0 de DISPLAY_HI no se utilizan
- En DISPLAY_LO se indica el tiempo transcurrido en segundos

Para medir la velocidad de funcionamiento, la cinta dispone de conectores cada 10 cm, los cuales al pasar frente a un sensor (incorporado en la cara interna de la cinta), disparan la interrupción cinta().

Se dispone de un timer que interrumpe con frecuencia de 1hz, invocando a la rutina tiempo().

Se pide:

Implementar en un lenguaje de alto nivel (preferentemente C), todas las rutinas necesarias para el correcto funcionamiento del sistema, considerando que la máquina está dedicada a la tarea.

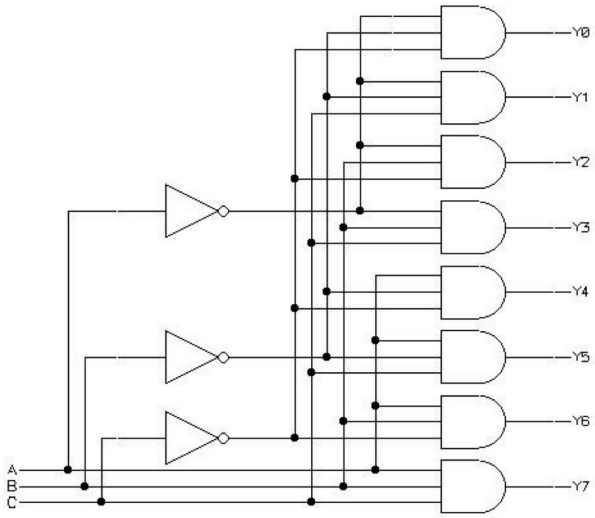
Nota: Se puede utilizar la siguiente aproximación: 1km/h ≈ 30cm/s

Aclaración durante la lectura de la letra: Al encender el motor su velocidad aumenta gradualmente hasta llegar al máximo posible (depende de las características físicas y de su potencia), y cuando se apaga decrementa su velocidad gradualmente hasta detenerse.

Solución

Respuesta Pregunta 1

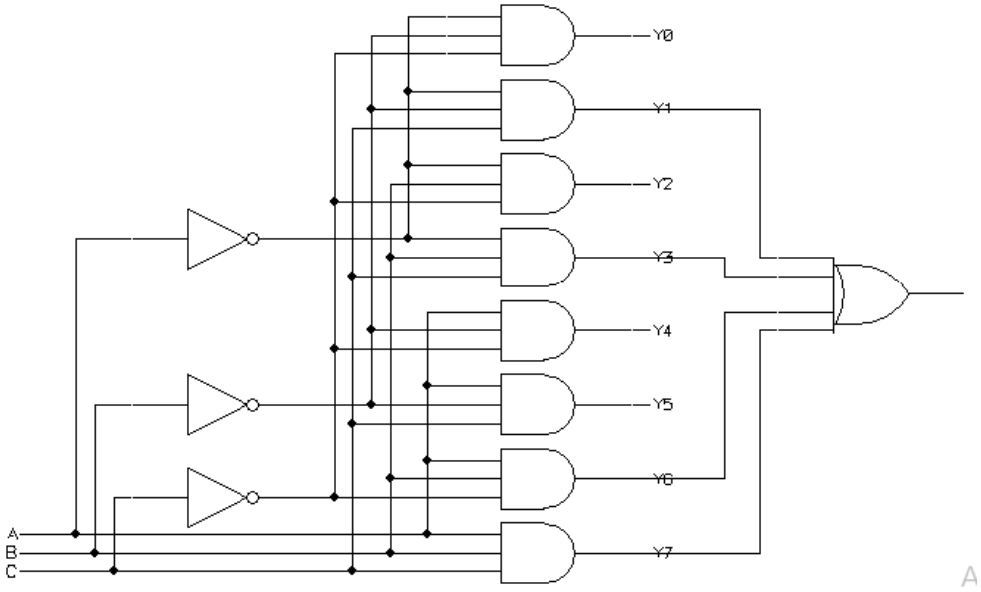
a) El circuito de un decodificar de 3 bits es el siguiente:



b) La función f tiene la siguiente tabla de verdad

a	b	c	ab + !ac
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Por lo que el circuito se puede construir en base al decodificador y una compuerta OR de 4 entradas:



Nota: no es necesario repetir el circuito interno del decodificador, es válido reemplazarlo con una caja con sus correspondientes entradas y salidas.

Respuesta Pregunta 2

Las entradas de un flip-flop D son:

- Entrada D: sincrónica
- Clock: sincrónica
- Clear: asincrónica
- Set: asincrónica
- Clock Enable: sincrónica

En el caso de los flip-flops con entrada de control por nivel, el nuevo valor de la salida corresponde al valor de la entrada D mientras la señal de control está en 1. Mientras la señal de control está en 0 la salida mantiene el valor de la entrada D inmediatamente antes de la transición de 1 a 0 de la entrada de control.

En el caso de los flip-flops con entrada de control por flanco, el nuevo valor de la salida corresponde al valor de la entrada D al momento de la transición de la entrada de reloj (CLK) de 0 a 1 (es decir el "flanco ascendente"). Este nuevo valor de la salida es adoptado inmediatamente después de ocurrido dicho flanco.

Respuesta Pregunta 3

El problema de la coherencia de la cache se genera cuando la eventual desincronización entre los datos en la memoria principal y la copia existente en la cache puede dar lugar a una inconsistencia en el manejo de dichos datos.

Para optimizar los intercambios de datos entre los dispositivos de Entrada/Salida y la memoria principal se utilizan sistemas de acceso directo a memoria (DMA = Direct Access Memory). Estos sistemas permiten realizar transferencias en ambos sentidos, entre la E/S y la memoria, sin la intervención de la CPU. Esto puede desencadenar el problema, ya que la CPU estaría accediendo a la copia de los datos en la cache mientras que el DMA a la copia en la memoria principal.

En particular en la política de escritura “write-back”, cuando la CPU realiza una escritura, si el dato a modificar está en la cache solo escribe en ésta. Y recién actualiza la memoria principal cuando corresponde reemplazar el bloque que contiene el dato modificado aplicando la política de reemplazo. En este escenario si el DMA quiere leer un dato de memoria prinicpal de un bloque “cacheado” que ha sido modificado por la CPU y aún no actualizado, se dará el problema de la choerencia .

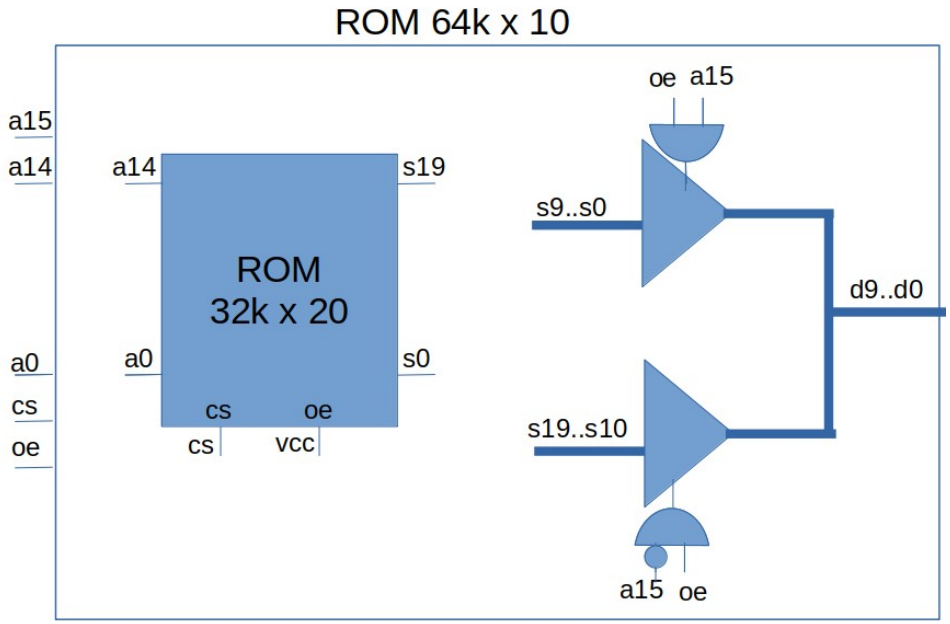
Respuesta Pregunta 4

a) 110011 111100 000001 111110 011001

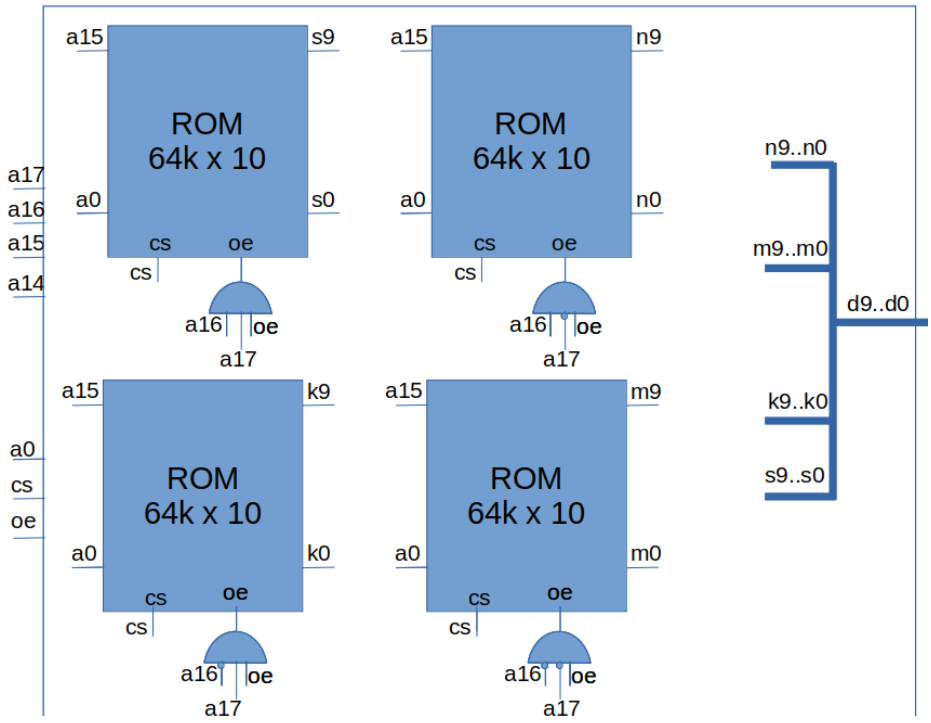
b) 0x5ACFD = 0101 1010 1100 1111 1101b = 01 011010 110011 111101b = Baz9 (base 64)

Solución Problema 1

a) Para construir la ROM solicitada partir de ROMs de 2^{15} x 20, lo primero que haremos es transformar las las ROMs dadas en ROMs de 2^{16} x 10 (64k x 10). Luego utilizaremos 4 de estas memorias para construir la memoria final de 256k x 10.



ROM 256k x 10



b) Carga de la ROM

```
void cargarRom (){
  short ROM [262144]; //voy a trabajar solo con los 10 primeros bits
  int z = 0; int v = 0;
  char res, a, b, op;
  for (int i = 0; i < 262144; i++){
    op = i % 4;
    a = (i >> 2) % 256;
    b = (i >> 10);
    switch (op) {
      case 0:
        res = a + b;
        v = (a * b > 0 &&          // sumandos del mismo signo
             a * res < 0) ? 1 : 0; // y resultado del signo opuesto
        break;
          case 1:
        res = a & b;
        break;
     case 2:
        res = a | b;
        break;
     case 3:
        res = ~a;
     }
     z = res == 0 ? 1 : 0;
     ROM [i] = res << 2 | v << 1 | z;
  }
}
```

Una forma alternativa de determinar el overflow es:

```
if ((res < -128) || (res > 127))  v = 1;
else v = 0;
```

Para lo cual hay que definir res como int y agregar

```
res = res % 256;
```

antes de asignar a ROM[i].

Solución Problema 2

```
#define OFF 1
#define ON 2
#define  TOL_DEC TOL_VEL * 10
int  v_obj, n_cinta, t_total, t_alarma;
char estado;

void main() {
  out(MOTOR, 0);
  out(DISPLAY_H,0);
  out(DISPLAY_L,0);
  estado = OFF;
  // instalar interrupciones
  enable();
  while true {}
}

void hay_boton() {
  char botones = in(BOTONES);
  if (botones & 1) {
    out(DISPLAY_H,0);
    out(DISPLAY_L,0);
    out(MOTOR, 0);
    estado = OFF;
  }
  else {
    int i = 1;
    while (botones & (1 << i) == 0){
      i++;    // válido porque por letra se asegura que existirá
    }         // un botón presionado
    v_obj = (i * 2 - 1) * 10;
    t_alarma = 0;
    if (estado == OFF)
      t_total = 0;
    estado = ON;
  }
}

void cinta() {
  n_cinta ++;
}
```

```
void timer() {
  int vi;        // velocidad instantánea en decenas de km/h
  if (estado == ON) {
    t_total++;
    vi = (10*n_cinta) / 3;
    if (v_obj - TOL_DEC < vi && vi < v_obj + TOL_DEC)
      t_alarma = 0;
    else         // velocidad fuera de rango
      t_alarma ++;
    char m_power =  v_obj < vi? 1 : 0;
    out(MOTOR, m_power);
    if (vi > 127) vi = 127; // tope: se muestra hasta 12.7 km/h
                            // porque es lo que permite la representación
    out(DISPLAY_H, (t_alarma > 30? 1: 0) << 15 + m_power << 14 + vi << 7);
    out(DISPLAY_L, t_total);
  }
  n_cinta = 0;
}
```