Examen de Arquitectura de Computadoras

27 de julio de 2024

Instrucciones:

- Indique su nombre, apellido y número de cédula en todas las hojas que entregue.
- 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) Implementar la función de cuatro variables $\Sigma(1, 2, 4, 7, 8, 9, 11, 14)$ utilizando, obligatoriamente, multiplexores de 8 a 1. Se dispone, además, de compuertas básicas.
b) ¿Cambiaría algo si la función de cuatro variables fuera $\Sigma(1, 2, 4, 7, 9, 10, 12, 15)$?

Pregunta 2

a) Describa las etapas del ciclo de instrucción clásico visto en el curso, detallando las actividades que se realizan en cada una de ellas.
b) ¿Qué condiciones deberían cumplirse para implementar dicho ciclo en un pipeline al máximo nivel de eficiencia del mismo?

Pregunta 3

a) Explique un algoritmo para convertir un número entero de una base b a una base B utilizando la aritmética de la base B.
b) Convierta 152_6 a base 10 utilizando la técnica de la parte anterior.

Pregunta 4

Explique en qué se diferencia un registro visible de uno parcialmente visible. Indique un ejemplo de cada uno para la arquitectura x86. Justifique.

Ejercicio 1

Se desea controlar el portón de un garaje que puede subir y bajar. El mismo cuenta con un motor para mover el portón que se puede encender y apagar, y además indicar en que sentido trabaja (para arriba o para abajo), un control remoto con un solo botón, un sensor que detecta la presión de dicho botón y un par de sensores que indican si el portón está completamente arriba o completamente abajo.

Cuando el portón está cerrado (abajo) y se presiona el botón se debe encender el motor en el sentido de subida y mantenerlo así hasta que el sensor indique que está completamente arriba. En ese momento se debe apagar el motor. Cuando está completamente abierto (arriba) y se presiona el botón el comportamiento es el inverso con el motor bajando.
Si se presiona el botón **mientras el portón está subiendo o bajando** se considera que puede haber una emergencia por lo que el motor se deberá detenerse por N segundos y luego comenzar a funcionar nuevamente en el sentido contrario. Si el botón se presiona nuevamente durante la espera de N segundos, no tiene efecto. Asimismo, por razones de seguridad se dispone de un sensor de presencia, y en caso de detectarse una persona se deberá detener el portón. El portón se volverá activar M segundos después de dejar de detectar a la persona, manteniendo el sentido.

Cada vez que se presiona el botón se invoca a la rutina de atención **boton**() y se dispone de un reloj que interrumpe a una frecuencia de 10Hz que genera una interrupción atendida por la rutina **tiempo**(). Además, se dispone de dos puertos para resolver el problema, el puerto CONTROL de tipo byte de solo escritura y el puerto ESTADO de tipo palabra de solo lectura. A continuación se presenta la estructura de estos puertos.

CONTROL	**ESTADO**
Bit 1: controla el sentido del motor (0 arriba)	Bit 1: indica el estado del sensor abajo
Bit 2: controla el encendido del motor (1 encendido)	Bit 2: idem sensor arriba
	Bit 3: indica si el sensor de presencia detecta (1 detecta)

Se pide:
Implementar todas las rutinas necesarias para implementar el control del portón sabiendo que el procesador está dedicado a la tarea.

Ejercicio 2

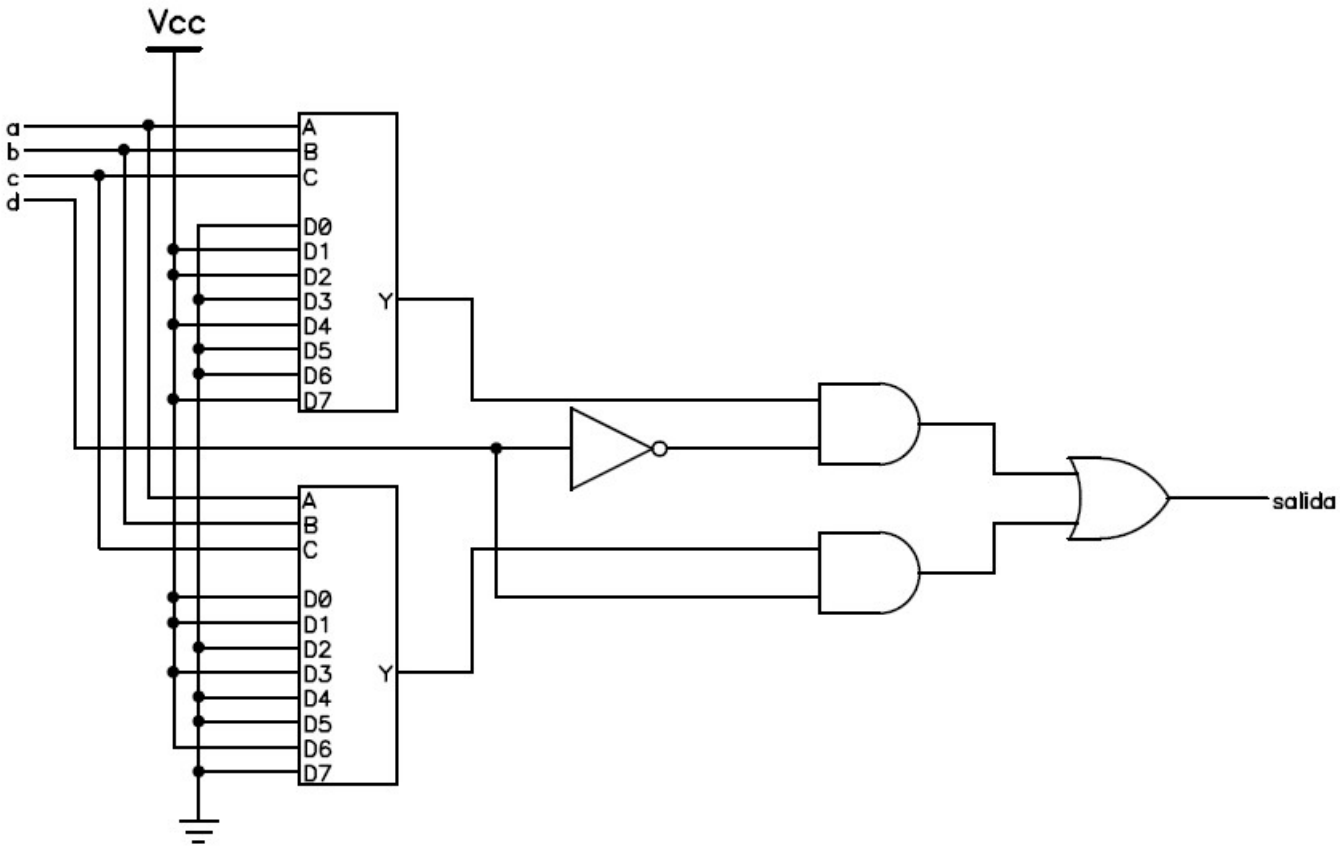
Se desea diseñar, construir y programar una ROM que convierta una hora en formato HH:MM:SS en su valor equivalente en segundos (entero sin signo). Para la entrada de la ROM asuma que cada valor es un dígito BCD y no se deben considerar los separadores (“:”).

Parte a) Indique entradas, salidas y organización de la ROM solicitada.
Parte b) A partir de ROMs de 2^{21}x32 construya la ROM de la parte a).
Parte c) Implemente un programa en C que cargue el contenido de la ROM.

Nota: El valor máximo para HH es 11, para MM es 59 y para SS también 59. Una entrada fuera de rango debe dar como salida todos los bits en 1.

Respuesta Pregunta 1

a) Para la función Σ(1, 2, 4, 7, 8, 9, 11, 14) utilizamos dos multiplexores de 8 a 1 (tres entradas de control “a”, “b” y “c”). Uno implementa la función Σ(1, 2, 4, 7) y el otro la Σ(0, 1, 3, 6) (que resulta de restar 8 a los puntos originales mayores a 7). Elegimos la salida de uno u otro en función de la cuarta entrada “d”



b) En el caso de la función Σ(1, 2, 4, 7, 9, 10, 12, 15), los dos multiplexores estarían implementando la misma función Σ(1, 2, 4, 7), con lo cual solo se precisa uno de los multiplexores y la entrada “d” no se usa.

Respuesta Pregunta 2

a) Describa las etapas del ciclo de instrucción clásico visto en el curso, detallando las actividades que se realizan en cada una de ellas.
b) ¿Qué condiciones deberían cumplirse para implementar dicho ciclo en un pipeline al máximo nivel de eficiencia del mismo?

a) El ciclo de instrucción clásico que vimos en el curso consta de 5 etapas: fetch, decode, read, execute y write.
Las actividades que ocuren son las siguientes:
Fetch: realiza la lectura desde la memoria de la próxima instrucción a ser ejecutada, utilizando para eso la dirección almacenada en el registro IP (Instruction Pointer) y la coloca en un registro interno.

Típicamente también actualiza el valor del registro IP para que quede apuntando a la próxima instrucción.
Decode: analiza el código binario de la instrucción leída y en función del formato de instrucción determina la operación a realizar, los operandos a utilizar y los modos de direccionamiento de los mismos.
Read: en caso que haya operandos en la memoria procede a leerlos. Esta etapa puede no realizarse si los operandos residen todos en registros.
Execute: realiza la operación indicada, para lo que típicamente utiliza la ALU.
Write: en caso que el destino de la operación esté localizado en memoria procede a escribir el resultado en la misma.

b) Para que pudiera implementarse un pipeline y el mismo funcionara a su máxima eficiencia debería ocurrir:
- las cinco etapas deberían durar la misma cantidad de ciclos de reloj.
- las cinco etapas deberían ser independientes entre sí, es decir no deberían existir interrelaciones de ningún tipo entre ellas (por ejemplo: no requerir dos etapas el mismo hardware, no requerir un resultado anterior aún no escrito, no depender de la decodificación para saber el largo de la instrucción a los efectos de realizar el fetch).

Respuesta Pregunta 3

a) El algoritmo visto en el curso para convertir un número entero de una base b a una base B utilizando la aritmética de la base B es la evaluación del polinomio característico. Partiendo del número a representar en base b:

$a_n...a_1a_0\ (b = a_nb^n + \ldots. + a_1 * b^1 + a_0 * b^0$

Luego, se evalúa el polinomio utilizando la aritmética de la base B.

b) $152_6 = 1 * 6^2 + 5 * 6^1 + 2 * 6^0 = 36 + 30 + 2 = 68_{10}$

Respuesta Pregunta 4

La diferencia principal entre los registros visibles y lo parcialmente visibles es que los registros visibles son directamente utilizados por el programador en las instrucciones, mientras que los parcialmente visibles son registros que pueden ser alterados de forma indirecta en las instrucciones, pero no son referenciados explícitamente.

En x86, un ejemplo de registro visible es el AX, debido a que es un registro de propósito general, y un ejemplo de registro parcialmente visible es el registro de FLAGS, cuyo valor es modificado por ejemplo luego de una instrucción aritmética o lógica.

Solución Ejercicio 1

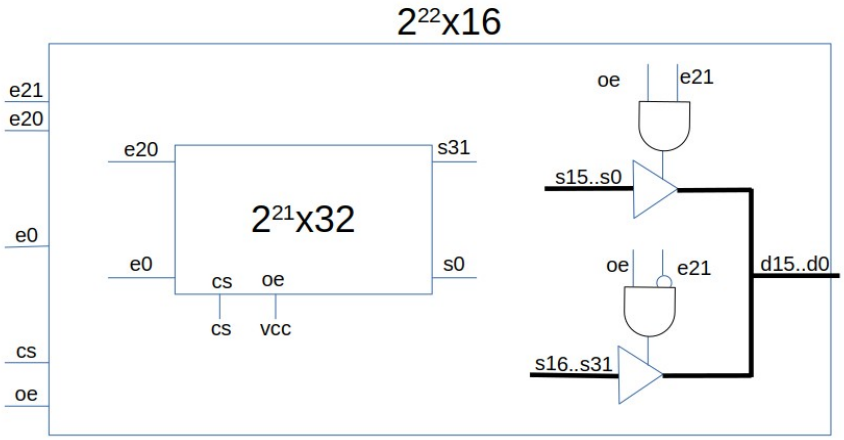
```
#define N …
#define M …

#define T_NSEG = N * 10 // paso de segundos a tics
#define T_MSEG = M * 10

#define E_ CERRADO 0    // portón cerrado
#define E_ SUBIENDO 1   // portón subiendo
#define E_ SUBIENDO_P 2 // subiendo pero se detectó persona
#define E_ DETENER_S 3  // subiendo pero se apretó el botón
#define E_ ABIERTO 4    // portón abierto
#define E_ BAJANDO 5    // portón bajando
#define E_ BAJANDO_P 6  // bajando pero se detectó persona
#define E_ DETENER_B 7  // bajando pero se apretó el botón

int tics, boton_a = 0, estado = EST_CERRANDO;

void interrupt tiempo() {tics++;}

void interrupt boton() {boton_a = 1;}

void main() {
  int p_estado, boton_p;
  out(CONTROL, 0);
  // instalar interrupciones
  enable();

  while (1) {
    p_estado = in(ESTADO);
    boton_p = boton_a;
    boton_a = 0; // registro si se apretó el botón y borro su valor

    switch (estado) {
      case E_SUBIENDO:
        if (p_estado & 4) { // sensor arriba activado
          estado = E_ABIERTO;
          out(CONTROL, 0);
        } else if (p_estado & 8) { // sensor de persona
          estado = E_SUBIENDO_P;
          tics = 0;
          out(CONTROL, 0);
        } else if (boton_p) {
          estado = E_DETENER_S;
          tics = 0;
          out(CONTROL, 0);
        }
        break;
      case E_BAJANDO:
        if (p_estado & 2) { // sensor abajo activado
          estado = E_CERRADO;
          out(CONTROL, 0);
        } else if (p_estado & 8) { // sensor de persona
          estado = E_BAJANDO_P;
          tics = 0;
          out(CONTROL, 0);
        } else if (boton_p) {
          estado = E_DETENER_B;
          tics = 0;
          out(CONTROL, 0);
        }
        break;
      case E_CERRADO:
        if (boton_p) {
          estado = E_SUBIENDO;
```

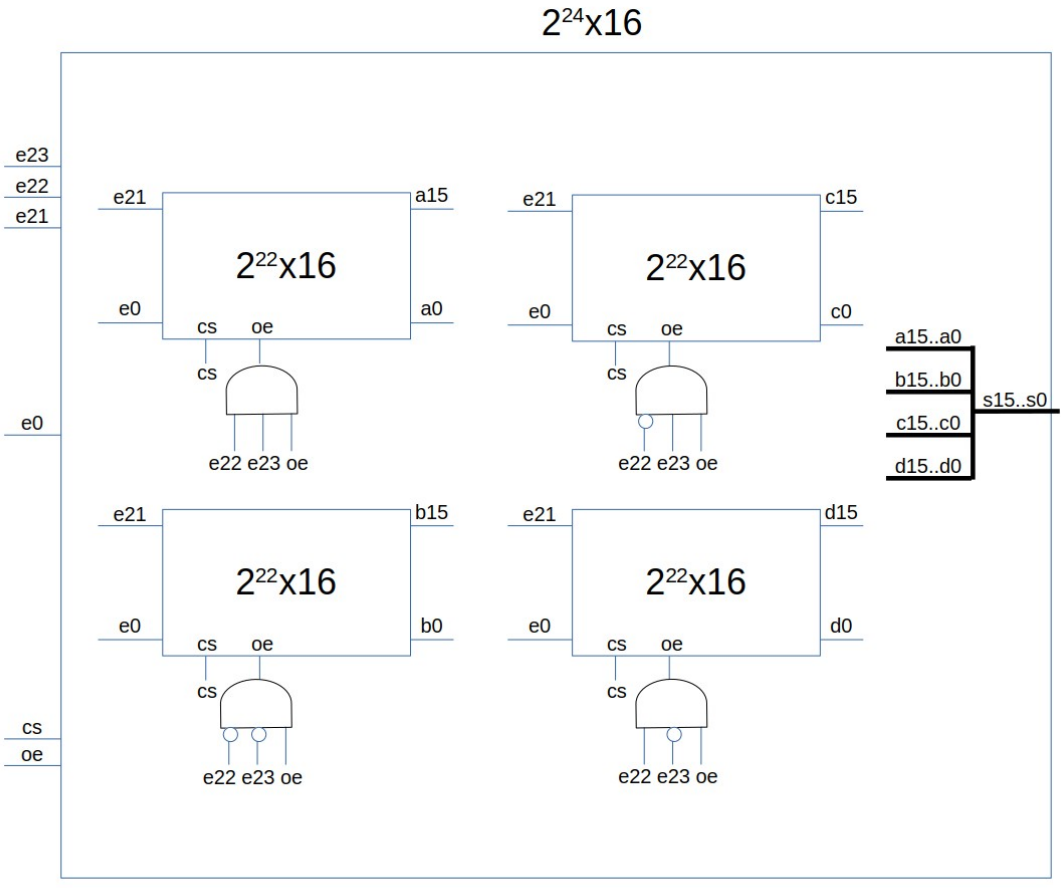
```
        out(CONTROL, 4);
      }
      break;
    case E_ABIERTO:
      if (boton_p) {
        estado = E_BAJANDO;
        out(CONTROL, 6);
      }
      break;
    case E_DETENER_S:
      if (tics >= T_NSEG) {
        estado = E_BAJANDO;
        out(CONTROL, 6);
      }
      break;
    case E_DETENER_B:
      if (tics >= T_NSEG) {
        estado = E_SUBIENDO;
        out(CONTROL, 4);
      }
      break;
    case E_SUBIENDO_P:
      if (tics >= T_MSEG) {
        estado = E_SUBIENDO;
        out(CONTROL, 4);
      } else if (p_estado & 8) { // sensor de persona
        tics = 0;
      }
      break;
    case E_BAJANDO_P:
      if (tics >= T_MSEG) {
        estado = E_BAJANDO;
        out(CONTROL, 6);
      } else if (p_estado & 8) { // sensor de persona
        tics = 0;
      }
      break;
    } // case
  } // while
}
```

Solución Ejercicio 2

a)
La entrada son 6 dígitos BCD, por lo que ocupa 6*4 = 24 bits.
Para saber cuántos bits se precisan para la salida precisamos saber la cantidad de segundos en 11:59:59. Como es un segundo menos que 12:00:00 podemos calcular (12*60*60) - 1 = 43199. Por lo tanto la salida es un número entre 0 y 43199 que se puede representar con 16 bits.
Precisamos entonces una ROM de 2^{24}x16

b)





c)

```
short ROM[1 << 24];

void cargar_rom() {
    unsigned char digitos[6];
    for (int i = 0; i < (1<<24); i++) {
        int hora_completa = i;
        for (int j=0; j<6; j++) {
            digitos[j] = hora_completa & 15;
            hora_completa = hora_completa >> 4;  // digitos queda [S, S, M, M, H, H]
        }
        if (digitos[0] <= 9 && digitos[1] <= 5 && digitos[2] <= 9 && digitos[3] <= 5
            && ((digitos[4] <= 9 && digitos[5] == 0) || (digitos[4] <= 1 && digitos[5] == 1)){

            mem[i] = digitos[0] + digitos[1]*10 + digitos[2]*60 + digitos[3]*10*60 +
                     digitos[4]*60*60 + digitos[5]*10*60*60;
        } else {
            mem[i] = -1; // 0xFFFF
        }
    }
}
```