Simulacro Examen de Arquitectura de Computadoras
Febrero de 2021

Ejercicio 1

Considere la siguiente variación al sistema de punto flotante de media precisión (1 bit de signo, 6 bits de exponente, 9 bits de mantisa).

a) Indique el desplazamiento utilizado en el exponente normalizado.
b) Indique el exponente implícito en números desnormalizados. Justifique la respuesta.
c) Indique el menor número representable normalizado.
d) Indique el menor número representable desnormalizado.
e) Codificar en esta representación el número 12.25.

Para las partes c) y d) considere números mayores que cero.

Ejercicio 2

La codificación *run_length_encoding* consiste en transmitir una serie de bits en forma de una serie de *paquetes,* cada uno de 8 bits, que contienen el valor del bit que se desea transmitir junto a su cantidad:

Ejemplo: la tira 0000100011111000 se transmite como 5 *paquetes*:

paquete informando 4 unidades 0
paquete informando 1 unidad 1
paquete informando 3 unidades 0
paquete informando 5 unidades 1
paquete informando 3 unidades 0

En un codificador RLE, los bits a transmitir se reciben a través del byte de E/S, de solo lectura, en la dirección ENTRADA_BITS. El bit 1 se coloca en 1 cuando hay un nuevo bit disponible en el bit 0 (menos significativo) del mismo puerto. El contenido del puerto es eliminado cuando éste es leído.

La transmisión se debe enviar a través del byte de E/S, de solo escritura, en la dirección SALIDA_RLE*,* indicando en el bit más significativo si se está enviando un paquete de 0s o de 1s y en los 7 bits menos significativos la cantidad transmitida.

Se pide:

Implementar en un lenguaje de alto nivel (preferentemente C) todas las funciones necesarias para implementar el codificador RLE descripto, sabiendo que el sistema está dedicado a la tarea.

Ejercicio 3

Considere una CPU de 16 bits (con direccciones de 16 bits), operando con una memoria cache de 1 KByte y 16 líneas, con una correspondencia totalmente asociativa y un algoritmo de sustitución random.
.
a) Indique cómo se interpreta la dirección de memoria para acceder a la cache.

b) Considere el siguiente código:

```
short arreglo[1024];

for (int i = 0; i < 1024; i++){
     arreglo[i] = arreglo[i] * 2;
}
```

Sabiendo que el arreglo se ubica a partir de la dirección 0 de memoria y que la cache se encuentra inicialmente vacía, calcule el hit rate y el miss rate en la ejecución del código.

Ejercicio 4

Se desea construir una ROM que implemente una ALU simple de 8 bits, con las siguientes cuatro operaciones para los operandos A y B, cada uno de 8 bits:

i) Suma en complemento a 2
ii) Resta en complemento a 2
iii) Shift de A, B lugares hacia la izquierda
iv) Shift de A, B lugares hacia la derecha

Para las operaciones iii) y iv) B se interpreta como entero sin signo. La ALU además del resultado en 8 bits brinda como salida las banderas de Z (cero) y N (Negativo) en función del resultado.

Se pide:

a) Indique entradas, salidas, capacidad y organización de la ROM necesaria para implementar la función indicada.

b) Escriba en alto nivel (preferentemente C) un programa que genere el contenido de dicha ROM.

Ejercicio 5

Se desea implementar un sistema de control basado en un procesador 8086. El sistema recibe datos de sensores y debe accionar válvulas en función de los datos informados por los sensores comparados con rangos de funcionamiento que dependen de qué sensor se trata.
Cada vez que un sensor tiene un nuevo dato se genera una interrupción que invoca a la rutina ***new_data_from_sensor()***, quedando disponible en la palabra de E/S, de solo lectura, en la dirección DATA_SENSOR el identificador del sensor (8 bits mas significativos) y el dato (8 bits menos significativos).
Se dispone de una estructura de datos definida de la siguiente forma:

```
typedef struct {
  unsigned char min;
  unsigned char max;
} Data_Range;

Data_Range rango_sensor[256];
```

que contiene la información de rangos de funcionamiento para cada uno de los sensores posibles.
El sistema debe mantener la válvula correspondiente al sensor en su estado actual mientras el dato sea mayor o igual al mínimo o menor o igual que el máximo. Si el dato está por debajo del mínimo la válvula debe cerrarse y si está por encima del máximo debe abrirse.

La válvula correspondiente a un sensor dado se acciona escribiendo en el bit cuyo número coincide con el dígito

verificador de su cédula en la palabra de E/S, de solo escritura, en la dirección VALVULAS+i, siendo i el identificador del sensor. Escribir un 1 cierra la válvula.

Se pide:
a) Implemente en un lenguaje de alto nivel (preferentemente C) todas las rutinas necesarias para resolver la realidad planteada.

b) Compile a 8086 sabiendo que el identificador de la interrupción generada por los sensores es el dígito verificador de su cédula y la rutina ***new_data_from_sensor()*** se carga en la dirección absoluta 0xFFFF.

Ejercicio 6

Se desea construir un circuito para realizar el control del peso de los productos comprados en un supermercado para ser usado en los nuevos sistemas de auto-atención.
El circuito recibirá como entrada la diferencia de peso entre lo indicado por la balanza y el peso esperado del producto en función de lo registrado en la base de datos del sistema informático del supermercado.
La diferencia es un número representado por dos bits (que representa un entero sin signo).
El circuito a construir tendrá una entrada de dos bits (la diferencia de peso), y una salida que se deberá colocar en uno cuando se detecten errores. El circuito recibirá la diferencia en cada tic del reloj y deberá señalar un error cuando se detecte que la diferencia es mayor a dos gramos por cuatro tics seguidos.

Se pide:
a) Diseñar una máquina de estados que solucione el problema. Dibujar su diagrama de estados y escribir su tabla de transición.
b) Construir el circuito mínimo según la metodología vista en el curso que implemente dicha máquina utilizando filp-flops tipo D y compuertas básicas. Dibujarlo

Solución Ejercicio 1

a)

$$D = 2^{N-1} - 1 = 2^5 - 1 = 31$$

b)

El exponente implícito coincide con el menor exponente normalizado. El menor exponente normalizado es E = 000001 => exp = E – D = -30 . El exponente implícito es -30

c)

El menor número normalizado (positivo) posible es

s = 0
e = 000001 (E = -30)
f = 000 000 000

$min\_normalizado = 2^{-30} * 1.0 = 2^{-30}$

d)

El menor número desnormalizado posible es

s = 0
e = 000000
f = 000 000 001 ($0.000000001 = 2^{-9}$)

$min\_desnormalizado = 2^{-30} * 2^{-9} = 2^{-39}$

e)

$12_d = 1100_b$
$0{,}25_d = 0{,}01_b$
$12{,}25_d = 1100{,}01_b = 1{,}10001 * 2^3$

s = 0
$e = 3 + 31 = 34_d = 100010_b$
f = 100010000

Solución Ejercicio 2

```
#define ESTADO_PAQUETE_0    1
#define ESTADO_PAQUETE_1    2

void main(){
      char bit_transmitido = 0;
      int cantidad = 0;
      while(true){
            char entrada = in(entrada_bits);
            if (entrada & 2){
                  char bit = entrada & 1;
                  if (bit == bit_transmitido){
                        cantidad++;
                  }else{
                        if (cantidad > 0){
                              // transmitir paquete
                              char transmi = bit_transmitido << 7 | cantidad % 128;
                              out(salida_lre, transmi);
                        }
                        cantidad = 0;
                        bit_transmitido = bit;
                  }
            }
      }
}
```

Solución Ejercicio 3

a)

Como la cache tiene 1 KB de capacidad y 16 líneas, entonces las líneas ocupan 64 bytes, y por tanto el campo byte ocupa 6 bits.

La línea se interpreta como tag | byte, dado que el campo byte ocupa 6 bits, el campo tag ocupa 10 bits.

b)

Cada short ocupa 16 bits, por lo tanto en cada línea de la cache entran 32 posiciones del arreglo.
En las primeras 33 iteraciones del for se accede a las siguientes posiciones de memoria:

arreglo[0] (lectura)
arreglo[0] (escritura)
arreglo[1] (lectura)
arreglo[1] (escritura)
arreglo[2] (lectura)
arreglo[2] (escritura)
…
...
arreglo[31] (lectura)
arreglo[31] (escritura)
arreglo[32] (lectura)
arreglo[32] (escritura)
...
...

El primer acceso es a la dirección 0 de memoria y es un MISS por estar la cache vacía. Esto provoca que se cargue el bloque de tag 0 en una línea cualquiera de la cache, conteniendo las posiciones 0, 1... 31 del arreglo. Los siguientes 63 accesos son a posiciones 0, 1... 31, por lo tanto los siguientes 63 accesos son HIT. Al iniciar la iteración con i = 32, se accede a arreglo[32], el cual no se encuentra en la línea cargada anteriormente, por lo tanto se produce un MISS y se carga la siguiente línea de cache.

Este comportamiento se repite a lo largo de toda la ejecución del código, y por tanto el hit rate es 63/64 y el miss rate 1/64

Solución Ejercicio 4

a)
La entrada de la ROM debe incluir:

- Ambos operandos (8 bits cada uno)
- Codificación de la operación a realizar (2 bits)

Se codificará como [operación][operando A][operando B]
2 bits 8 bits 8 bits

La operación se codificará como:

Suma (00)
Resta (01)
Shift left (10)
Shift right (11)

Mientras que la salida es el resultado de la operación, de 8 bits y los dos bits de banderas. Se codificarán como [banderas ZN][resultado]
2 bits 8 bits

La ROM necesaria tendrá 2^18 palabras de 10 bits. La capacidad es de 2560 Kbits.

b)

```
short ROM[...];

void cargaRom(){
	int tope = 1 << 18;
	for (int i = 0; i < tope; i++){
		char op = i >> 16;
		char a = i >> 8;
		char b = i;
		unsigned char b_int = i;
		char res = 0;
		if (op == 0){
			res = a + b;
		}else if (op == 1){
			res = a - b;
		}else if (op == 2){
			res = a << b_int;
		}else if (op == 3){
			res = a >> b_int;
		}
		char z = (res == 0? 1 : 0) ;
		char n = (res < 0? 1 : 0);
		ROM[i] = z << 9 | n << 8 | res;
	}
}
```

Solución Ejercicio 5

a)

```
#define DATA_SENSOR …
#define VALVULAS …
#define D_CEDULA ...
#define MASK_CTRL_VALVULA 1 << D_CEDULA

void main () {
     //instalar rutina de atención a la interrupcion
     enable();
     while true {}
}

interrupt new_data_from_sensor() {
     unsigned char info_sensor ,  id_sensor;

     info_sensor = in(DATA_SENSOR);
     id_sensor = info_sensor / 256:
     data_sensor = info_sensor & 0xFF;
     if (data_sensor < rango_sensor[id_sensor].min)
          out(VALVULAS+id_sensor, 0);
     else if (data_sensor > rango_sensor[id_sensor].max)
          out(VALVULAS+id_sensor, MASK_CTRL_VALVULA);
}
```

b)

```
DATA_SENSOR EQU …
VALVULAS EQU …
D_CEDULA EQU …
ISR_OFFSET  EQU D_CEDULA*4
ISR_SEGMENT EQU ISR_OFFSET+2
MASK_CTRL_VALVULA EQU 1 SHL D_CEDULA

rango_sensor DB 512 dup(0)

main proc
       xor ax,ax
       mov es,ax
       mov word ptr es:[ISR_OFFSET], 0xF
       mov word ptr es:[ISR_SEGMENT], 0xFFF
       sti; {habilitoInterrupciones}
  loop:
       jmp loop
main endp
```

```
new_data_from_sensor proc far
       mov DX, DATA_SENSOR
       in AX, DX    ; lee puerto de E/S para obtener la informacion del
sensor
       xor BH, BH
       mov BL, AH
       SHL BX, 1    ; multiplico por el tamaño del elemento de la estructura
       cmp AL, [rango_sensor+BX] ; comparo contra el min
       jae else
       XOR DH, DH        ; cierra la valvula
       mov DL, AH
       add DX, VALVULAS
       mov AX, 0
       out DX, AX
       jmp fin
  else:
       cmp AL, [rango_sensor+BX+1] ; comparo contra el max
       jbe fin
       xor DH, DH        ; abre la valvula
       mov DL, AH
       add DX, VALVULAS
       mov AX, MASK_CTRL_VALVULA
       out DX, AX
  fin;
       iret
new_data_from_sensor endp
```

Solución Ejercicio 6

a)

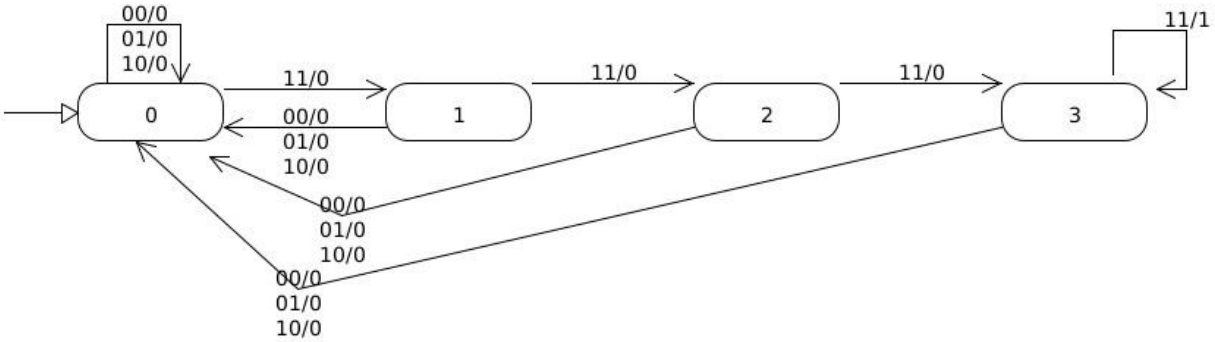


Tabla de Estado

Estado	Próximo estado				Salida			
	E=00	E=01	E=10	E=11	E=00	E=01	E=10	E=11
0	0	0	0	1	0	0	0	0
1	0	0	0	2	0	0	0	0
2	0	0	0	3	0	0	0	0
3	0	0	0	3	0	0	0	1

Tabla de Transiciones y Salidas

Q1	**Q0**	**E1**	**E0**	**D1**	**D0**	**S**
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	1	1	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	0	0	0
1	1	1	1	1	1	1

b)

Mapa de Karnaugh para D0

Q1Q0 \ E1E0	00	01	11	10
00	0	0	1	0
01	0	0	0	0
11	0	0	1	0
10	0	0	1	0

D0 = Q1 E0 E1 + Q0’ E0 E1

Mapa de Karnaugh para D1

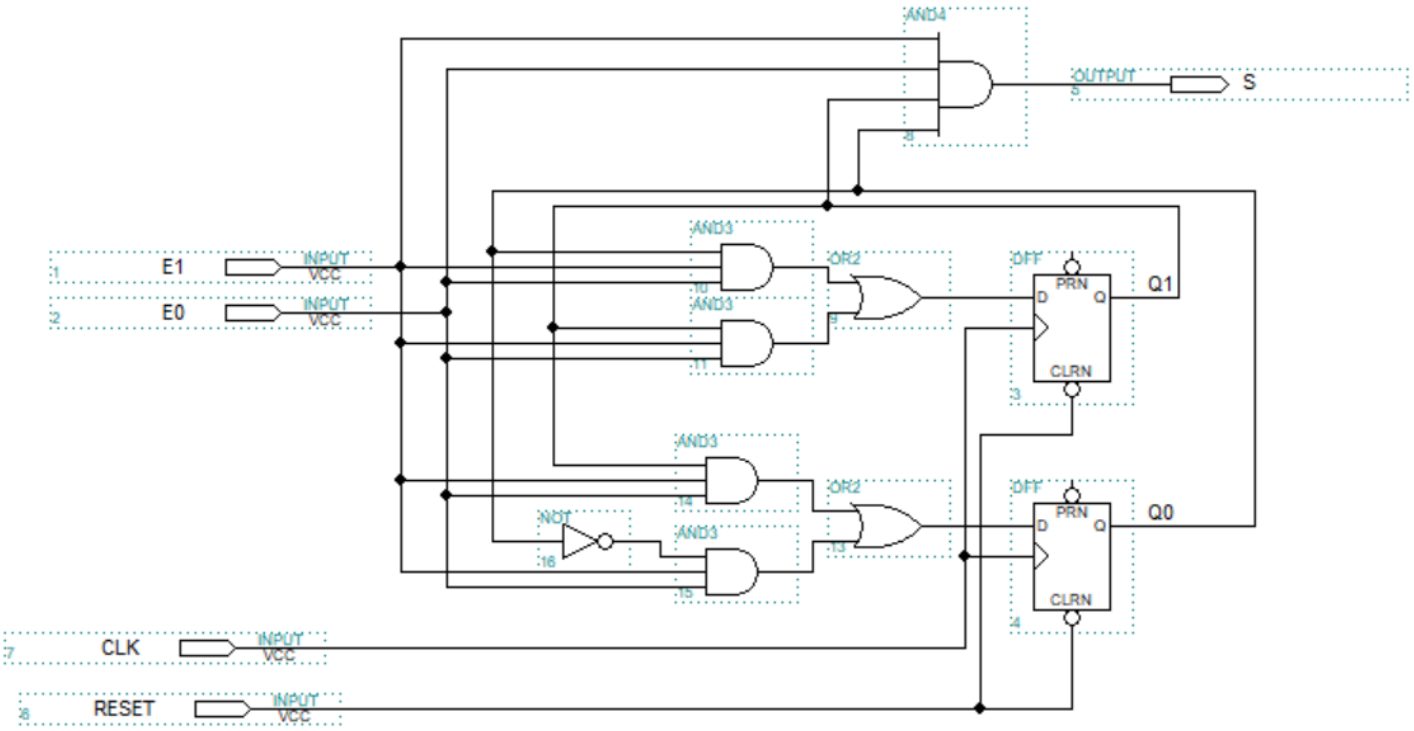
Q1Q0 \ E1E0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	0	1	0
10	0	0	1	0

D1 = Q1 E0 E1 + Q0 E0 E1

Mapa de Karnaugh para S

Q1Q0 \ E1E0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	0
10	0	0	0	0

S = Q0 Q1 E0 E1


AND4
OUTPUT
S
E1
INPUT
VCC
E0
INPUT
VCC
AND3
AND3
OR2
DFF
PRN
D
Q
Q1
CLRN
AND3
AND3
NOT
OR2
DFF
PRN
D
Q
Q0
CLRN
CLK
INPUT
VCC
RESET
INPUT
VCC