Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores (II/2012)

Solución Interrogación 1

Pregunta 1

a) Describa un caso en que es mejor utilizar números en big-endian y otro en que sea mejor utilizarlos en little-endian. **(1 pto.)**

Solución: En una situación en que lleguen datos de manera secuencial, dependiendo del caso es más conveniente little-endian o big-endian. Si la operación a realizar con los datos es una suma, conviene que sea little-endian, ya que el carry se traspasa desde los bytes menos significativos a los más significativos. Por otro lado, si la operación a realizar es una división, es preferible usar big-endian, ya que la división se realiza desde los bytes más significativos a los menos significativos.

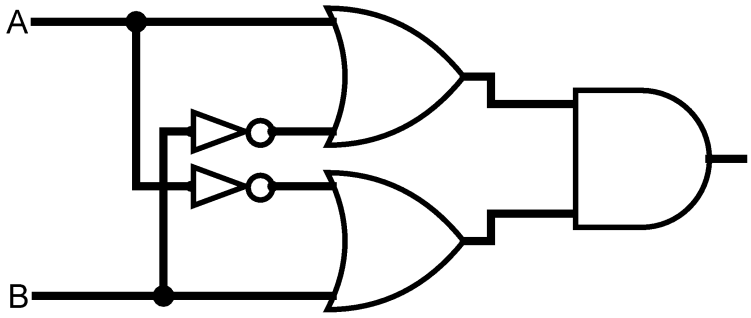
b) Escriba en formato float el número 16,375. Indique como se divide y que significa cada una de las partes del string de bits. **(1 pto.)**

Solución: En el formato float se asigna el primer bit al signo del significante, los siguientes 8 al exponente (desfasado en 128) y los restantes 23 al significante (normalizado). Luego, dado que $16,375 = 10000,011b = 1,0000011b \times 10b^{100b}$, el número en float se escribe de la siguiente manera: 0 10000011 00000110000000000000000

c) Implemente, utilizando sólo las compuertas lógicas **AND**, **OR** y **NOT**, el conectivo binario bicondicional ($\leftrightarrow$), que está definido por la siguiente tabla de verdad: **(1 pto.)**

A	B	A$\leftrightarrow$B
F	F	V
F	V	F
V	F	F
V	V	V

Solución: La tabla de verdad del conectivo bicondicional es analoga a la de la negación de un XOR, luego, se puede implementar de la siguiente manera con compuertas lógicas:



d) Un número entero de **16** bits en **little-endian** está almacenado en la posición 0x12 de una memoria RAM con palabras de 1 byte. Un programador extrae este dato de la memoria RAM y al no saber como interpretarlo prueba con un entero de **16** bits en **big-endian**. Si el número almacenado inicialmente era **255**, que valor obtuvo el programador? **(1 pto.)**

Solución: El número entero 255, se representa con 16 bits y little-endian cómo $0x00FF$. Luego, en big-endian, el número queda $0xFF00$, lo que corresponde al número -256 en decimal.

e) ¿Cuántas direcciones tiene una memoria RAM de 4.5 KB que utiliza palabras de 3 bytes? (1 KB = 1024 bytes) **(1 pto.)**

Solución: El tamaño total de la memoria es 4.5KB, que corresponde al producto entre el tamaño de las palabras y la cantidad de direcciones. Luego, la cantidad de direcciones de la memoria es $\frac{4{,}5\times1024}{3} = 1{,}5 \times 1024 = 1536$

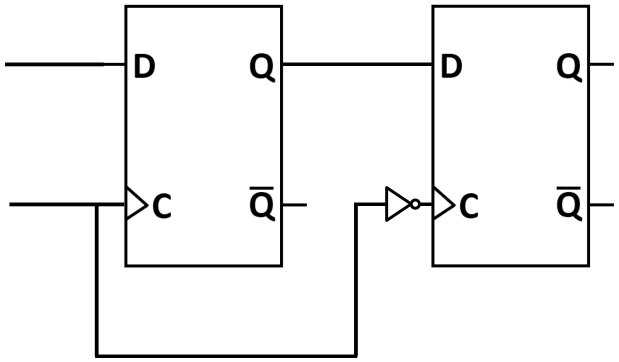
f) Un registro de 8 bits tiene almacenado el **número entero** 102. Que número se obtiene al realizar 2 operaciones **shift right** seguido de 3 operaciones **shift left**? **(1 pto.)**

Solución: El número 102 se escribe en binario como 01100110. Después de los 2 shift right, se obtiene 00011001, y después de los 3 shift left se obtiene finalmente 11001000, que es -56 en decimal.

Pregunta 2

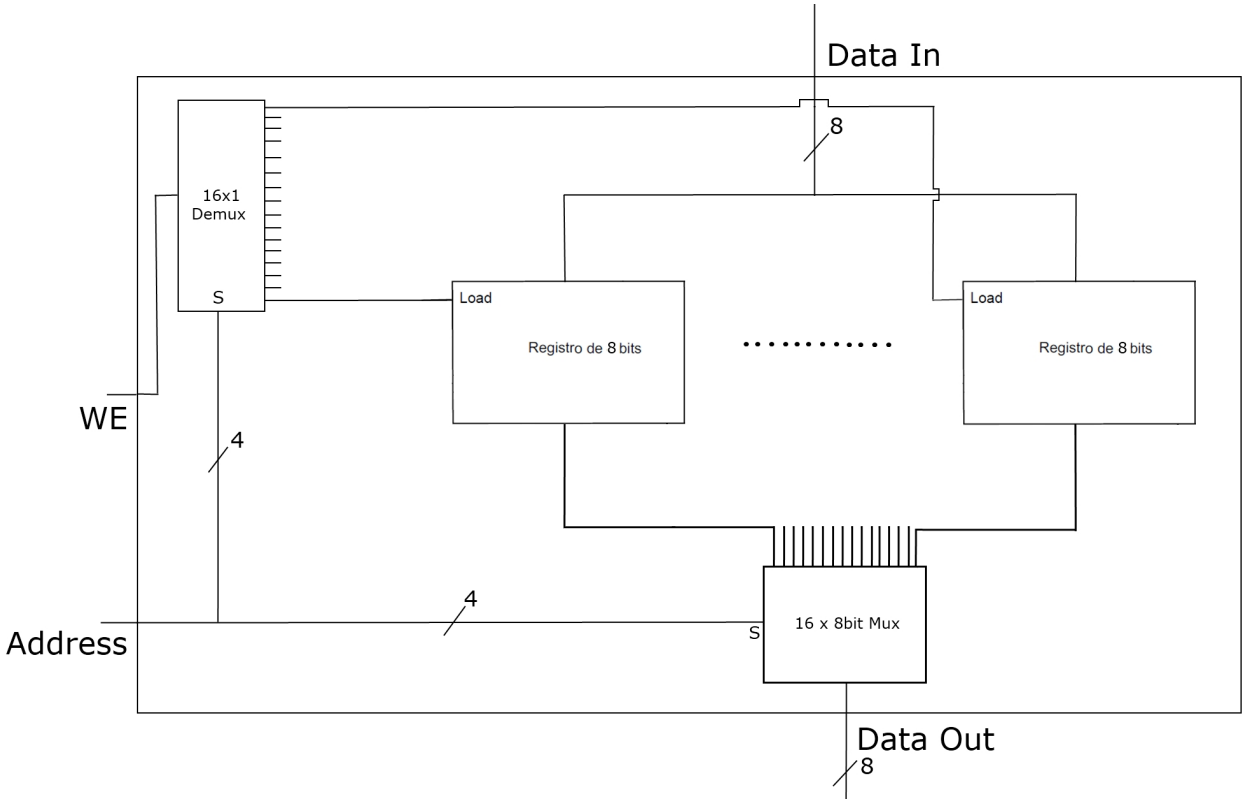
a) Implemente mediante compuertas lógicas, elementos de control y latches, un flip-flop tipo **D** que funcione con flanco de **bajada**. **(3 ptos.)**

Solución:



b) Implemente mediante compuertas lógicas, elementos de control y flip-flops, una memoria RAM de 16 palabras de 1 byte. **(3 ptos.)**

Solución:



Pregunta 3

Para los siguientes ejercicios, escriba su respuesta en el lenguaje de programación que más le acomode. Deje por escrito claramente cual es su elección.

a) Escriba un programa para convertir un número **natural** en base 12 a su valor en base decimal. Los digitos de un número en base 12 son: 0,1,2,3,4,5,6,7,8,9,A,B. Asuma que los números están almacenados en strings. **(3 ptos.)**

Solución:

```
public static int base12aBase10(char[] num)
{
    int res = 0;
    for (int i = num.length-1, factorBase = 1; i >= 0; i--, factorBase *=12)
    {
        int valor;
        char base12 = num[i];
        if (base12 == 'A')
        {
            valor = 10;
        }
        else if (base12 == 'B')
        {
            valor = 11;
        }
        else
        {
            valor = Character.getNumericValue(base12);
        }
        res += valor*factorBase;
    }
    return res;
```

b) Escriba un programa para obtener el inverso aditivo de un número en base 12, sin utilizar otra base como paso intermedio. Asuma que los números están almacenados en strings. **(2 ptos.)**

Solución:

```
public static char[] inversoAditivoBase12(char[] num)
{
    char[] res = new char[num.length];
    for (int i = 0; i < res.length; i+)
    {
        char c = num[i];
        if (c == '0')
        {
            res[i] = 'B';
        }
        else if (c == '1')
        {
            res[i] = 'A';
        }
        .
        .
        .
        .
        else if (c == 'A')
        {
            res[i] = '1';
        }
        else
        {
            res[i] = '0';
        }
    }
    //asumimos la existencia de un metodo que calcula
    //la suma de dos numeros en base 12
    res = sumaBase12(res, 1);
    return res;
```

c) Escriba un programa para obtener la dirección de memoria de un elemento de una matriz almacenada en orden de columnas. Asuma que el programa recibe los siguientes parámetros de entrada: la matriz, la fila del elemento y la columna del elemento. Puede utilizar los operadores para indexación vistos en clases. **(1 ptos.)**

Solución:

```
public static int obtenerDireccion(Matriz A, int i, int j)
{
    return dir(A) + j*A.filas*sizeof(A[i][j]) + i*sizeof(A[i][j]);
}
```