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

IIC2343 – Arquitectura de Computadores

Guía preparación I1

1. Describa un proceso 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. Puede escribirlo en pseudo-código o en el lenguaje que más le acomode.

2. Para cada uno de los items siguientes, describa el proceso requerido. Puede escribirlo en pseudo-código o en el lenguaje que más le acomode.

   - `Sumar números binarios`: Asuma que los números no tienen signo, que no necesariamente tienen el mismo largo y que la suma no sobrepasará la cantidad de bits del mayor de los sumandos.
   - `Complemento a 2 de un número (inverso aditivo)`: Asuma primero que recibe un número binario sin signo. Luego, repita el ejercicio asumiendo que recibe un número que ya se encuentra en complemento a 2.
   - `Restar números binarios`: Asuma que los números no tienen signo y que no necesariamente tienen el mismo largo. El resultado si debe estar representado con signo.

3. Describa un proceso para transformar un número binario en complemento a 2, en un número de punto flotante de precisión simple en formato IEEE754. Puede escribirlo en pseudo-código o en el lenguaje que más le acomode.

4. Describa un proceso para transformar un número binario de 32 bits, codificado en IEEE754 de precisión simple, a su valor cómo número racional. En su conversión debe considerar los casos especiales mencionados en los apuntes. Puede escribirlo en pseudo-código o en el lenguaje que más le acomode.

5. Implemente un circuito ocupando compuertas binarias, que recibe como entrada los 32 bits de un número binario que representa a un número del tipo «single precision floating point» según el estándar IEEE754, y entrega cuatro salidas:

   - la primera salida debe entregar un 1 cuando el número recibido sea el **cero** especificado por dicha representación, y un 0 en caso contrario;
   - la segunda salida debe entregar un 1 cuando el número recibido sea **+infinito**, y un 0 en caso contrario;
   - la tercera salida debe entregar un 1 cuando el número recibido sea **-infinito**, y un 0 en caso contrario;
   - la cuarta salida debe entregar un 1 cuando el número recibido sea **NaN**, y un 0 en caso contrario.

   Para armar su circuito puede utilizar compuertas **and** de múltiples entradas y compuertas **or** de múltiples entradas.

6. Suponga que se tiene un programa en Java encargado de controlar el funcionamiento de un conjunto de lavadoras. La información de cada lavadora se almacena en la clase `Lavadora` descrita a continuación. Cada lavadora tiene un **estado** que puede ser lavado, estrujado o centrifugado; un **nive del detergente** que puede tomar un valor entero desde 0 hasta 10 (ambos extremos inclusive); un indicador de si **esta encendida** o no; y un indicador de si **está llena** o no.

```
public class Lavadora {

  public int estado;        //0=lavado, 1=estrujado, 2=centrifugado
  public boolean estaEncendida;
  public boolean estaLlena;
  public int nivelDetergente;   //0-10

  public Lavadora(int estado, boolean estaEncendida, boolean estaLlena, int nivelDetergente)
  {
    this.estado = estado;
    this.estaEncendida = estaEncendida;
    this.estaLlena = estaLlena;
    this.nivelDetergente = nivelDetergente;
  }

}
```

Debido a restricciones de almacenamiento de información, se desea codificar la información de cada lavadora en un sólo `byte`. Describa procedimientos para las siguientes tareas:

- `Codificar lavadora`: Codifica la información de una lavadora (i.e. atributos) en 1 byte (8 bits).
- `Decodificar lavadora`: Basado en la codificación del item anterior, extrae de un byte toda la información relacionada con una lavadora.

Puede codificar la información como estime conveniente. Lo importante es que quepa en 1 byte y que al convertir una lavadora a una valor 1 byte y luego el valor de 1 byte de vuelta a una lavadora, se tengan almacenados los mismos valores originales.

7. Para este código escrito en lenguaje Java, resuelva lo siguiente:

   ¿Que valor se almacena en `var3` luego de ejecutar el `main`? ¿Que operación está realizando el método `foo`?

```
public static void main(String[] args)
{
  byte var1 = 12;
  byte var2 = 3;
  byte var3 = 0;
  var3 = foo(var1, var2);
}

private static byte foo (byte a, byte b)
{
   byte c = 0;
   while(true)
   {
     byte d = (byte)(b & 1);
     if(d != 0)
       c += a;
     b >>= 1;
     if(b == 0)
       return c;
     a <<= 1;
```

```
    }
}
```

8. Dado los siguientes métodos en Java, donde `var` representa un tipo básico que no se conoce:

```
var foo1(var a, var b)
{
  var c = b & a;
  while (a != 0)
  {
    c = b & a;
    b = b ^ a;
    c = c << 1;
    a = c;
  }
  return b;
}

var foo2(var a, var b)
{
  a = a ^ b;
  b = a ^ b;
  a = a ^ b;

  return a;
}
```

a) Suponga que se tiene una memoria de 16 entradas, con palabras de 1 byte con los valores que se muestra a continuación:

Dirección	Palabra
0	$0x0A$
1	$0x71$
2	$0x00$
3	$0x00$
4	$0x12$
5	$0x44$
6	$0x00$
7	$0x0F$
8	$0x00$
9	$0x00$
10	$0x01$
11	$0x05$
12	$0x00$
13	$0x02$
14	$0x00$
15	$0x0F$

Calcule el valor de retorno de ambos métodos para los siguientes casos:

- Las variables `a` y `b` son de tipo `byte`; `a` está almacenada a partir de la dirección 0; `b` está almacenada a partir de la dirección 6.
- Las variables `a` y `b` son de tipo `int`; `a` está almacenada a partir de la dirección 3; `b` está almacenada a partir de la dirección 10; ambas variables ocupan orden big endian.
- Las variables `a` y `b` son de tipo `short`, el cual corresponde a un número en complemento a 2, de 2 bytes; `a` está almacenada a partir de la dirección 0; `b` está almacenada a partir de la dirección 11; ambas variables ocupan orden little endian.

b) ¿Que operación está realizando el método `foo1` sobre los valores a y b?

c) ¿Que operación está realizando el método `foo2` sobre los valores a y b?

9. Se tiene el siguiente método escrito en Java:

```
byte foo(byte num)
{
  byte op = num;
  byte res = 0;
  byte one = (1 << 6);

  while (one > op)
    one >>= 2;

  while (one != 0)
  {
    if (op >= res + one)
    {
      op -= res + one;
      res = (res >> 1) + one;
    }
    else
    {
      res >>= 1;
    }
    one >>= 2;
  }
  return res;
}
```

Suponga que se tiene una memoria de 4 entradas, con palabras de 4 bits con los valores que se muestra a continuación:

Dirección	Palabra
0	0
1	7
2	0
3	1

Responda lo siguiente:

a) Suponga que el parámetro `num` es de tipo `byte` y está almacenado a partir de la dirección 0 en orden big endian. Determine el resultado del método `foo`.

b) Suponga que el parámetro `num` es de tipo `byte` y está almacenado a partir de la dirección 2 en orden little endian. Determine el resultado del método `foo`.

c) ¿Que operación está realizando el método `foo` sobre el valor `num`?

10. Responda lo siguiente:

   a) ¿Qué ventajas y desventajas tiene una representación de punto flotante en comparación a una de punto fijo?

   b) ¿Qué ventajas tiene la representación de punto flotante del estándar IEEE754 por sobre la representación de complemento a 2 al ejecutar una división por cero?

   c) ¿Es posible diseñar una representación de punto flotante de 8 bits que permita almacenar sin pérdida de información el número fraccional $\frac{1}{3}$? En caso de estimar que se puede, explique en detalle como lo haría. En caso contrario, justifique por qué no es posible.