PRÁCTICA 1

CONVERSIÓN ENTRE BASES. ARITMÉTICA BINARIA
REPRESENTACIÓN DE LOS NÚMEROS CON SIGNO

OBJETIVOS:

Una vez finalizada la práctica deberemos ser capaces de:

- Realizar tareas de representación en los diferentes sistemas de numeración.
- Realizar tareas de conversión entre las diferentes bases numéricas.
- Realizar operaciones de conversión de números binarios a sistema octal o hexadecimal (y viceversa) de forma inmediata.
- Sumar, restar, multiplicar y dividir números binarios.
- Expresar los números positivos y negativos en formato signo-magnitud
- Expresar números positivos y negativos en complemento a 1 y complemento a 2.
- Evaluar y representar números con signo con notación sesgada

REFERENCIAS:

- T.L. Floyd: *Fundamentos de Sistemas Digitales*, 9ª Edición, Capítulo 2. ”Sistemas de numeración, operaciones y códigos”; secciones 2-1 a 2-3 y 2-5 a 2-11
- M. Morris & C. R. Kime: *Fundamentos de Diseño Lógico y Computadoras.* Capítulo 1. “Computadoras Digitales e Información”; secciones 1-2 y 1-3.
- P. de Miguel Anasagasti, *Fundamentos de los computadores*, 9ª Edición, Capítulo 2. “Representación de la Información”; Sección 2.5: Representaciones Numéricas.
- Transparencias Tema 2 “Representación de la información”. Fundamentos de los Computadores.

ELEMENTOS NECESARIOS:

Se deberán realizar las operaciones sobre papel, pudiéndose comprobar los resultados obtenidos haciendo uso de asistentes informáticos:

- Calculadora.
- Hojas de Cálculo.
- Programas específicos de conversión:
  - http://wims.unice.fr/wims/es_tool~number~baseconv.es.html
  - http://calc.50x.eu

PRESENTACIÓN DE LOS RESULTADOS.

Se deberá realizar una memoria de la práctica en el que aparezcan las operaciones realizadas y los resultados obtenidos como solución a cada ejercicio.

INTRODUCCIÓN TEÓRICA. EJERCICIOS A REALIZAR

1. Un método sistemático para convertir un número entero decimal a una base *b* es el de las *divisiones sucesivas*. Por ejemplo, para convertir a binario (base 2) el número decimal 23, comenzaremos dividiendo 23 entre 2 (base destino). Luego cada cociente resultante se divide entre 2 hasta que se obtenga uno cuya parte entera es 0. Los restos generados en cada división formarán el número binario buscado. Para ello, los ordenaremos de forma inversa a que los hemos obtenido. Es decir, el primer resto es el bit menos significativo (LSB) del número binario, y el último resto es el más significativo (MSB).

Utilizando este método, realiza las conversiones de los siguientes números decimales a la base que se indica:

a) 145 a base 2 (binario)

b) 168 a base 6

c) 488 a base 12

2. Los números decimales fraccionarios se pueden convertir a una base *b* mediante el método de la *multiplicación sucesiva* por dicha base *b.* Por ejemplo, si queremos pasar un número a binario, empezaremos multiplicando el número por 2, y, después se multiplica cada parte fraccionaria del número resultante del producto por 2 hasta que se obtenga una parte fraccionaria que sea cero o hasta que se alcance el número deseado de dígitos. Las partes enteras generadas en cada una de las multiplicaciones dan lugar al número buscado: la primera será el MSB, y la última el LSB.

Realiza las conversiones de los siguientes números decimales a la base que se indica. Emplea 7 cifras significativas para la representación binaria y 5 en el resto:

a) 0,716 a binario

b) 0,422 a octal

c) 0,72 a base 16

3. Si queremos convertir un número decimal compuesto por una parte entera y una fraccionaria a una base *b*, debemos emplear los dos métodos anteriores de forma independiente. Primero convertiremos la parte entera mediante el método de las divisiones sucesivas y a continuación el de las multiplicaciones sucesivas para la parte fraccionaria.

Indica qué representación tendrían los siguientes números decimales en la base pedida:

a) 11,716 en binario

b) 19,72 en base 16

4. El valor de cada dígito de un número representado en un determinado sistema de numeración depende del dígito en sí y de la posición que ocupa en la secuencia que lo forma. La posición que ocupa cada dígito nos dará idea de la magnitud representada y se le puede asignar un peso. Estos pesos serán las potencias de la base en que estemos trabajando con un exponente que será el de la posición que ocupa dentro del número. Con ello, cualquier número podrá escribirse como un polinomio de potencias de la base en que esté representado. Por ejemplo, el número x ($x = x_{n-1}, x_{n-2}, \ldots, x_1, x_0$) compuesto por n dígitos en la base b lo podemos representar de la forma:

$$x = \sum_{i=0}^{n-1} x_i b^i$$

Obtén el equivalente decimal de los siguentes números teniendo en cuenta que la base en que se encuentran representados viene indicada por sus subíndices.

a) 246_8

b) $AC1,04_{16}$

c) $10011,1001_2$

5. La conversión entre la base binaria y otra base que sea potencia de 2 es una tarea sencilla y que puede realizarse directamente sin tener que recurrir a los mecanismos empleados en el punto anterior. Por ejemplo, si tenemos que realizar la conversión de un número expresado en la base *b1* (siendo $b1 = 2^n$) a base 2, bastará con sustituir cada dígito del número original por su cadena equivalente escrita con n dígitos binarios. Si la conversión que queremos hacer es la contraria y queremos convertir un número binario a esa base *b1*, el método sigue funcionando igualmente. Bastará con que, partiendo del bit menos significativo, formemos grupos de n dígitos binarios y escribamos su dígito equivalente en la base *b1*.

En el caso de que el número de bits no sea múltiplo de n, añadiremos a la izquierda los bits que sean necesarios. Si el número tiene parte fraccionaria, agruparemos hacia la izquierda para la parte entera y hacia la derecha para la parte fraccionaria. Si los dígitos que componen la parte fraccionaria tampoco fueran múltiplo de n, ahora necesitaremos añadir los bits necesarios por la derecha.

Las bases que son potencia de 2 y se utilizan habitualmente son la base 8 (octal, 2^3) y sobre todo la 16 (hexadecimal, 2^4), pues dada la facilidad de conversión nos servirán para representar de forma compacta números binarios compuestos por muchos bits.

Realiza las siguientes conversiones entre bases:

a) $1001110110,1001_2$ a base 8

b) 67714_8 a base 2

c) $10001111101010,01_2$ a base 16

d) $9A5,C_{16}$ a base 2

e) $5476,6_8$ a base 16

6. La aritmética binaria no es diferente de la decimal. Sólo tendremos que tener en cuenta la base en que estamos trabajando y que, por tanto, las opciones que tenemos son las que comprendan los dígitos binarios. En concreto, la suma se reducirá a:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 y llevo 1 (acarreo)

Observa que la última opción produce un acarreo (nos llevamos 1) que debemos sumar con la columna situada inmediatamente a la izquierda. Esto nos conduce a una situación en la que se deben sumar tres bits (un bit de cada uno de los números y un bit de acarreo). Puesto que podremos aplicar la propiedad asociativa, los resultados que se tendremos en cada caso serán:

1 + 0 + 0 = 1	(no hay acarreo)
1 + 0 + 1 = 0	y 1 de acarreo
1 + 1 + 0 = 0	y 1 de acarreo
1 + 1 + 1 = 1	y 1 de acarreo

a) Si tenemos los números binarios *A* = 10100111 y *B* = 10110110, siguiendo estas reglas, realiza la suma de *A* + *B*.

7. La multiplicación binaria la realizaremos de la misma forma que la decimal. Es necesario realizar productos los parciales de cada bit del multiplicador por el multiplicando, desplazando cada producto parcial a la izquierda, y luego sumar dichos productos.

0 × 0 = 0
0 × 1 = 0
1 × 0 = 0
1 × 1 = 1

a) Sean *A* = 10011011 y *B* = 110. Realiza el producto *A* × *B*.

8. Las reglas básicas para la resta de números binarios son:

0 – 0 = 0
0 – 1 = 1 y llevo 1 (acarreo)
1 – 0 = 1
1 – 1 = 0

Existen varias formas de interpretación del acarreo que se produce en el segundo caso. Si seguimos el mismo criterio que en las operaciones anteriores y operamos como en decimal, este acarreo lo pasaremos a la columna situada a la izquierda y lo sumaremos con el bit del sustraendo.

a) Sean *A* = 11010011 y *B* = 01110101. Realiza la resta *A* – *B*.

Hemos de tener en cuenta que hasta ahora solamente hemos tratado con números sin signo. Esto quiere decir que cuando realicemos la resta de dos números hemos de asegurarnos que el minuendo sea mayor que el sustraendo. Si no fuera así, el resultado que tendríamos que obtener sería negativo y la representación carecería de sentido.

9. Para realizar la división de dos números binarios utilizaremos el mismo criterio que empleamos con los números decimales, si bien se suelen incluir de forma explícita las restas del divisor que se realizan en cada paso.

0 ÷ 0 = operación no definida
0 ÷ 1 = 0
1 ÷ 0 = operación no definida
1 ÷ 1 = 1

a) Sean *A* = 110110011 y *B* = 110. Realiza la división *A* ÷ *B*.

b) Toma ahora *C* = 10000110 y haz *C* ÷ *B*.

10. Los números, además de su magnitud, pueden tener signo. Para ello utilizamos símbolos (+ y –) que anteponemos a la magnitud. Sin embargo, en el sistema binario, solo tenemos dos símbolos (0 y 1) y no tenemos capacidad para colocar un signo como los anteriores, que son los que utilizamos habitualmente. Encontramos varias formas básicas para la representación de números con signo:

- Signo y magnitud.
- Complemento a 1.
- Complemento a 2.
- Mediante Sesgo (o exceso).

El criterio de signo y magnitud responde a un razonamiento intuitivo: se añade a la magnitud un bit de signo. Un 0 como signo positivo y un 1 como signo negativo. El número binario compuesto por *n* bits queda formado por:

- 1 bit de signo.
- *n* - 1 bits de magnitud

a) Representa los siguientes números decimales en signo y magnitud utilizando 8 bits:

  i. + 82

  ii. – 68

  iii. – 24

  iv. + 110

b) ¿Qué rango de representación tenemos utilizando este número de bits?

c) Identifica los siguientes números binarios expresados en signo y magnitud

  i. 10100001

  ii. 100011

  iii. 010011

11. Para realizar la división de dos números binarios utilizaremos el mismo criterio que. En el criterio de los complementos, los números positivos tienen la misma representación que en signo y magnitud, mientras que un número negativo puede ser escrito como el complemento del correspondiente positivo. De forma genérica, la representación de –A en una determinada base B sería:

$$(-A) = B^n - A$$

Donde *n* es el número de dígitos con que queremos representar el número.

Si particularizamos para la base 2, la expresión se convierte en:

$$(-A) = 2^n - A$$

Y si se tratara de complemento a 1 (complemento restringido a la base 2) sería:

$$(-A) = 2^n - A - 1$$

De forma sencilla podemos decir que el complemento a 1 se obtiene invirtiendo los ceros por unos y viceversa del número a complementar. Es decir, obteniendo su complemento lógico.

a) Representa los siguientes números decimales en complemento a 1. Emplea para ello 8 bits.

  i. +87

  ii. -99

  iii. -74

b) Identifica los siguientes números binarios con signo expresados en complemento a 1.

   i. 11010110

   ii. 01101100

   iii. 10000

c) Representa los siguientes números decimales en complemento a 2. Emplea para ello 8 bits.

   i. -64

   ii. +58

   iii. -108

d) Identifica los siguientes números binarios con signo expresados en complemento a 2.

   i. 00100110

   ii. 1000000

   iii. 111100

12. En el caso de la representación sesgada, o en exceso, el código se forma sumando un cierto valor al número original. Para codificar un número con signo en binario con sesgo se procede sumándole el sesgo, que suele ser 2^{n-1}, al número decimal y posteriormente codificándolo en binario puro.

a) Si queremos trabajar con 8 dígitos binarios, obtén la representación de los siguientes números decimales en notación sesgada:

   i. +72

   ii. -68

   iii. -112

b) Identifica los siguientes números binarios con signos expresados en notación sesgada

   i. 1100110

   ii. 00011100