AEC3: Arquitectura de Ordenadores

ISMAEL HERNÁNDEZ CLEMENTE

PROBLEMA 1

Para convertir un número decimal en binario, multiplicamos la parte decimal por 2 y anotamos el número entero resultante. Repetimos este proceso hasta que la parte decimal sea 0 o hasta que hayamos obtenido suficientes dígitos significativos en binario.

0,296875 * 2 = 0,59375 (anotamos 0)

0,59375 * 2 = 1,1875 (anotamos 1)

0,1875 * 2 = 0,375 (anotamos 0)

0,375 * 2 = 0,75 (anotamos 0)

0,75 * 2 = 1,5 (anotamos 1)

0,5 * 2 = 1,0 (anotamos 1)

Por lo tanto, 0,296875 en binario es 0.01001.

Representar los números decimales en Signo-Magnitud (S-M):

En la representación de Signo-Magnitud, el bit más significativo representa el signo (0 para positivo, 1 para negativo) y el resto de los bits representan la magnitud del número. Por ejemplo, para representar -42 en Signo-Magnitud: Primero convertimos 42 en binario, que es 101010 y luego, como el número es negativo, agregamos 1 al principio. Por lo tanto, -42 en Signo-Magnitud es 1101010.

Para representar los números decimales en Complemento a dos los números positivos se representan de la misma manera que en Signo-Magnitud. Para los números negativos, invertimos los bits del número positivo correspondiente y luego sumamos 1. Con el ejemplo anterior primero convertimos 42 en binario, que es 101010. Luego, invertimos los bits para obtener 0101011. Finalmente, sumamos 1 para obtener 0101100. Por lo tanto, -42 en Complemento a dos es 11010110.

Para extender los números a un tamaño de 10 bits, simplemente agregamos ceros al principio hasta que el número tenga 10 bits.

S-M

123	01111011
-42	10101010
365	101101101
274	100010010
-274	1100010010
28	011100

Complemento a dos

123	01111011
-42	11010110
365	101101101
274	100010010
-274	011110110
28	00011100

10 bits

123	00001111011	00001111011
-42	11010101000	11101011000
365	01011011010	01011011010
274	01000100100	01000100100
-274	11000100100	10111011100
28	00000111000	00000111000

PROBLEMA 2

A	B	Op	Suma	Carry
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	1
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

Donde aquí:

- A y B son los bits de entrada.
- Op es el bit de operación (0 para suma, 1 para resta).
- Suma es el resultado de la suma/resta.
- Carry es el acarreo de la operación.

Ejemplos:

- Para A=1, B=0, y Op=0 (suma), obtenemos Suma=1 y Carry=0.
- Para A=1, B=1, y Op=1 (resta), obtenemos Suma=1 y Carry=1.

Diferencia entre Carry y Overflow:

El carry ocurre cuando la suma de dos bits excede el valor máximo que puede ser representado con esos bits. Por ejemplo, en una suma de un solo bit, si sumamos 1 + 1 obtenemos un carry ya que el resultado (2) no puede ser representado con un solo bit.

El overflow es un concepto similar, pero se aplica a la suma de números enteros con signo. Ocurre cuando el resultado de una operación excede el rango de valores que puede ser representado con los bits disponibles. Por ejemplo, si sumamos dos números positivos y obtenemos un número negativo, eso es un overflow. En la aritmética de complemento a 2, esto puede suceder si el bit de carry de la suma de los bits más significativos no coincide con el bit de carry de la suma total.

PROBLEMA 3

El algoritmo de suma-desplazamiento es un método para realizar multiplicaciones binarias. Aquí te muestro cómo se realiza la multiplicación de 4 x 7 y 7 x 4 paso a paso:

Multiplicación 4 x 7:

Primero, representamos los números en binario:

- 4 en binario es 0100
- 7 en binario es 0111

Luego, aplicamos el algoritmo de suma-desplazamiento:

Multiplicando: 0100 (4)

Multiplicador: 0111 (7)

- Paso 1: 0100 << 1 = 1000 (Desplazamiento)
- Paso 2: 1000 + 0100 = 1100 (Suma porque el bit menos significativo del multiplicador es 1)
- Paso 3: 1100 << 1 = 10000 (Desplazamiento)
- Paso 4: 10000 (No se suma porque el bit menos significativo del multiplicador es 0)
- Paso 5: 10000 << 1 = 100000 (Desplazamiento)
- Paso 6: 100000 + 0100 = 100100 (Suma porque el bit menos significativo del multiplicador es 1)

Resultado: 100100 (28 en decimal)

Multiplicación 7 x 4:

Ahora, intercambiamos el multiplicando y el multiplicador:

Multiplicando: 0111 (7)

Multiplicador: 0100 (4)

- Paso 1: 0111 << 1 = 1110 (Desplazamiento)
- Paso 2: 1110 (No se suma porque el bit menos significativo del multiplicador es 0)
- Paso 3: 1110 << 1 = 11100 (Desplazamiento)
- Paso 4: 11100 + 0111 = 100011 (Suma porque el bit menos significativo del multiplicador es 1)
- Paso 5: 100011 << 1 = 1000110 (Desplazamiento)
- Paso 6: 1000110 (No se suma porque el bit menos significativo del multiplicador es 0)

Resultado: 1000110 (28 en decimal)