Introducción a los sistemas de computo

Por Kronoman
En memoria de mi querido padre
Mayo/2003

Sistemas numéricos posicionales

Formula general

$$N = a_n . r^n + ... + a_1 . r^1 + a_0 . r^0$$

Siendo r la base (ejemplo, base 10, base 2) y n la posición del dígito.

Sistema binario

Cada dígito del sistema binario se llama bit.
Para pasar de decimal a binario, se divide el numero por 2, sucesivamente hasta que no se pueda mas, y se arma el numero binario tomando el ultimo resultado (como bit mas significativo, es decir, mas a la derecha) y luego todos los restos de las divisiones.

Paso de binario a octal:

Se separa el binario en grupos de 3 bits, y se convierten por separado,
ejemplo: 110|100|111|001|001| = 64711
Para pasarlo de octal a binario, lo mismo, al reves: 467 = 100|110|111

Hexadecimal

Hexadecimal a binario: se toman grupos de 4 bits:
ejemplo: 50F = 0101|0000|1111
Ídem al reves.

Registros

Modulo: cantidad de combinaciones del registro (2^n)

Complemento a la base: lo que le falta al numero para llegar al modulo del registro
Complemento a la base menos 1: Ídem, pero modulo-1

Números enteros con signo, representación decimal

Hay 3 tipos de convenciones:

Signo y magnitud: el bit mas significativo indica el signo solamente. Los otros bits son el numero en si.
Ejemplos: +1 = 00001, -1 = 10001

Complemento a la base: No tiene bit mas significativo, todos los bits toman parte del numero.
Si el numero es positivo, se usa un binario puro.
Si es negativo, se lo representa por el complemento a la base.
Para complementar, se lee de izquierda->derecha el numero binario, se van dejando los ceros, hasta que aparece un uno, el cual se deja también, y a partir de allí, se invierten unos y ceros.
Ejemplo:
+2 = 00010
-2 = 11110 <- notar la complementacion realizada

El bit mas significativo se considera negativo, y suma en la conversión a decimal:
Ej: 11101 (-3) = $-1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = -16 + 8 + 4 + 0 + 1 = -3$

Complemento a la base -1: Los negativos se representan **invirtiendo** los bits.
Ejemplo: -0 = 11111, -1 = 11110, -2 = 11101, etc...

Operaciones aritméticas

Método de Suma del FPU

Método usando complemento a la base
Ejemplo:

	0	0	1	1	1
+	0	0	0	1	1
0	0	1	0	1	0

(esto es 7 + 3 = 10)
El 0 ultimo de la izquierda es el carrier.
Notar como se hizo el acarreo de bits al sumar los dos 1.
Resta del FPU
Usando complemento a la base
Para restar, se suma el primer numero mas el complemento a la base, del segundo.
Ejemplo:

	0	0	1	1	1
+	1	1	1	0	1
0	0	0	1	0	0

(esto es 7 - 3 = 4)
Notar como se sumo el complemento de 7.

Nota importante:
En caso de que se usara **complemento a la base – 1,** si A > B y el carrier contiene un 1, se le suma 1 al resultado, expresado así: $A-B \rightarrow A+C_B^{B-1}=A+2^n-1-B=2^n+(A-B)-1$
O sea, si el carrier es 1, sumarlo al primer bit de la derecha.

Overflow

El resultado de una operación aritmética fuera de rango.
Ejemplo:

	0	1	0	0	1
+	0	1	0	1	1
0	1	0	1	0	0

(esto es 9 + 11= 20)
Overflow, porque el resultado, como estamos usando complemento a la base, seria un numero negativo.
Para que esto no ocurra, el carrier del bit mas significativo debe ser 0, si sumo números positivos.
Además, el carrier de ultima y penúltima etapa deben ser 0.

Ejemplo:

	1	1	0	0	1
+	1	1	1	0	1
1	1	1	1	0	1

(esto es -7 + -3 = - 10)
Esto esta bien. Aquí no hubo overflow, porque son números negativos.

Ejemplo:

	1	0	1	1	1
+	1	0	1	0	1
1	0	1	1	0	0

(esto es -9 + -11 = -20)
Aquí si hubo overflow, nos dio un numero positivo!

Conclusión: para que no haya overflow, los transportes ultimo y ante ultimo deben ser iguales.
En sumas de números de diferente signo, **no** hay overflow.

Multiplicación binaria de números positivos

La manera humana de realizar la operación, es hacerlo como se hace una multiplicación con números decimales.
La unidad aritmética lógica (ALU) opera de manera diferente a los humanos.
Supongamos una maquina de registros de 5 bits.
A la unidad, ingresan 2 registros de 5 bits, y dentro, la operación se efectúa en un registro doble, de 10 bits, el cual lleva el multiplicando y el resultado, para ahorrar un registro.
Este registro doble contiene en los 5 dígitos inferiores el multiplicador, y en los 5 superiores el resultado.
Así:

R	E	S	U	L	M	U	L	T	P

Si el ultimo bit (P) es un 1, se envía el multiplicando de nuevo.

Procedimiento de ejemplo:
7 x 3
Multiplicando (7):

0	0	1	1	1

Registro doble, contiene el 3 en el registro de la derecha (celeste):

0	0	0	0	0	0	0	0	1	1

Primera vuelta, se le suma el 7 binario al registro doble:

0	0	1	1	1	0	0	0	1	1

Cuando el ultimo dígito es 1, entra de nuevo en la ALU, si no, envía todos 0
Se hace un corrimiento a la derecha, se llena con 0 si es positivo, y con 1 si es negativo, a partir de la primera vez que sumo el multiplicando.

0	0	0	1	1	1	0	0	0	1

Sumo el 7 que se ingreso antes.
+00111

0	1	0	1	0	1	0	0	0	1

Corrimiento

0	0	1	0	1	0	1	0	0	0

Ahora, como el ultimo bit es un 0, ingresan todos 0 a la suma
+00000

0	0	1	0	1	0	1	0	0	0

Corrimiento

0	0	0	1	0	1	0	1	0	0

+00000
queda igual, hago el corrimiento

0	0	0	0	1	0	1	0	1	0

+00000
queda igual, hago el corrimiento

0	0	0	0	0	1	0	1	0	1

Resultado final, registro de la derecha (celeste)
El algoritmo termina luego de 5 corrimientos (si fuera de 8 bits, de 8 corrimientos, etc)
La operación se resume de la siguiente manera:
Ingresan los dos números:
Se coloca uno en el registro doble, en la parte de la derecha, el otro se agrega en la de la izquierda.
Cuando el ultimo dígito del registro doble es 1, entra de nuevo en la ALU la parte izquierda del registro doble, si no, envía ceros.
Se suma esta entrada en la parte izquierda.
Se hace un corrimiento a la derecha, se llena con 0 si es positivo, y con 1 si es negativo, a partir de la primera vez que sumo el multiplicando.
Se repite el proceso hasta haber realizado 5 corrimientos (o la cantidad de corrimientos según el ancho del registro, si fuera de 8 bits, el registro doble tendrá 16 bits, y se harán 8 corrimientos, etc).

Nota: si los operadores son negativos, antes de hacer la Multiplicación, se complementa (para pasarlos a positivos) y luego acomodo el signo al resultado.
O sea, si uno de los dos es negativo, resultado negativo, si ambos son negativos o positivos, resultado positivo.

Nota: para pasar de un registro a otro de mas bits, hay que copiar (rellenar con) el bit mas significativo los primeros bits que sobran (por lo del complemento a la base)
Ejemplo:
5 a 10 bits: (en celeste, los bits 'rellenados')
+7 seria:

0	0	1	1	1

0	0	0	0	0	0	0	1	1	1

-7 seria:

1	1	0	0	1

1	1	1	1	1	1	1	0	0	1

(en celeste, los bits 'rellenados')

Esta es la justificación de porque, cuando multiplicamos, se rellena con 1 o 0, dependiendo del primer bit.

Algoritmo de división con restauración

Ejemplo con registros de 4 bits.
Este algoritmo se realiza por prueba y error, o sea, se van probando resultados hasta encontrar el correcto.
O sea, se efectúa la división, y si el resto es < 0, me pase, si es > divisor, es muy chico.

La ALU recibe un registro de 4 bits, y otro doble, donde los primeros 4 lleva el resto, y los otros 4 lleva el resultado.

Ejemplo:
[[tarea pendiente]]

Números reales, números de punto flotante

Se utiliza un exponente, el cual indica hacia donde se "corre" la coma.
Ejemplo, con un E de centro 50, un numero > 50, indica la coma hacia la derecha, y < hacia la izquierda, = 50 queda igual.

Aclaración: mantisa entera: *significa que la coma esta al extremo derecho****; mantisa fraccionaria*** *significa que la mantisa esta al extremo izquierdo (siempre si tenemos el exponente en valor de polarización [centro])*

Ejemplo utilizando números decimales, y un exponente de "centro" 50:
Tomamos la operación: 485,487 + 2,37
Aquí coloco lo que me entre del primer numero, en este caso 4854, y como tengo que correr la coma un lugar a la izquierda, por eso el valor 49 en el exponente.

0	4	9	4	8	5	4
S	E		M			

S = signo, E = exponente, M = mantisa, y la coma *comienza* ubicada en el extremo derecho.

Numero 2,37, notar como completo con ceros, y el valor del exponente para ubicar la coma decimal.

0	4	7	2	3	7	0

Para poder restarlos, igualo los exponentes (49 y 47) así:
49 - 47 = 2 => corro la mantisa 2 lugares a la derecha en el numero 2,37; quedando así:

0	4	9	0	0	2	3

Ahora efectúo la suma de las mantisas: 4854+23 = 4877 y formo el resultado:

0	4	9	4	8	7	7

Como vemos que el 49 del exponente indica que la coma debe ser corrida un lugar, este numero es 487,7

Otro ejemplo:

Suma: en este caso, vemos que los exponentes son iguales, entonces no es necesaria la igualación, pero al sumar las mantisas, obtenemos un overflow porque : 9725+8997 = 18722, entonces lo que se hace es despreciar el ultimo dígito, y sumarle 1 al exponente.

0	4	9	9	7	2	5

0	4	9	8	9	9	7

Resultado:

0	5	0	1	8	7	2

Conclusión: si al sumar la mantisa, se produce overflow, se descarta el bit menos significativo, y se suma 1 a E, se pierde precisión.
Si E > 99 al sumar +1, entonces es OVERFLOW.
Lo mismo,a la inversa, es valido para UNDERFLOW.

Multiplicación de números flotantes

$$N_1 = m_1 . r_1^e$$
$$N_2 = m_2 . r_2^e$$
$$N_1 . N_2 = (m_1 . m_2)^{r.(e_1+e_2)}$$

O sea, se suman los exponentes (restándole luego la polarización) y se multiplican las mantisas.

Ejemplo: **20 x 320**

0	4	8	2	0	0	0
0	4	9	3	2	0	0

Sumo los exponentes y les resto la polarización (en este caso, la polarización es 50; esta polarización es el "centro" que antes mencionamos).
48+49 = 97 - 50 = **47**
Efectuó la multiplicación de mantisas: 2000 x 3200 = 6400*000*; descarto los *bits que sobran* para que entre en la mantisa (3 bits), los cuales también tienen que ser sumados al exponente (como se explico antes en la suma)
Y queda:

0	4	7	6	4	0	0

Que al sumarle los 3 bits que se descartaron al E, queda el resultado final:

0	5	0	6	4	0	0

División de números flotantes

$$N_1 / N_2 = (m_1 / m_2)^{r.(e_1-e_2)}$$

O sea, se restan los exponentes (sumándole luego la polarización) y se dividen las mantisas.

Ejemplo: **4820 / 2**

0	5	0	4	8	2	0
0	4	7	2	0	0	0

Resto los exponentes, y le sumo la polarización: 50 - 47 = 3 + 50 = 53
Luego divido las mantisas: 4820 / 2000 = 2410, y el corrimiento de 3 que me indica el E lo aplico, y el resultado final es 2410:

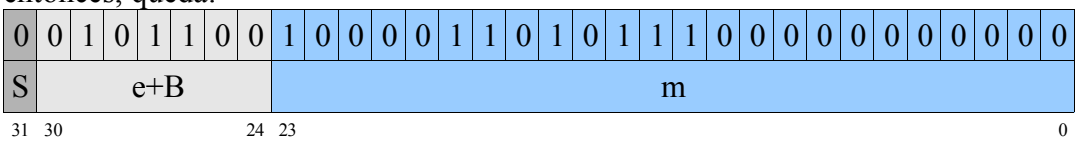
0	5	0	2	4	1	0

Como convertir de decimal a un numero flotante binario

Tomemos de ejemplo el numero **8,42**
Se convierte la parte entera (8) a binario = 1000
Luego para convertir la parte decimal viene la parte tediosa; se toma solo la parte decimal (0,42) y se va multiplicando por 2, hasta encontrar una secuencia que se repita (o alcanzar la precisión deseada) y se van tomando los 1 y 0 de la parte entera, y descartando los decimales así (el primer 0, de 0,42, se ignora):

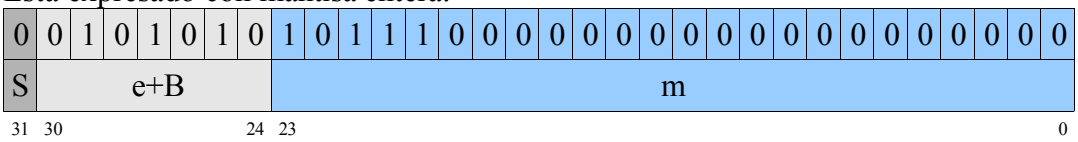
0,42 x 2
0,84 (tomo el 0), y multiplico 0,84 por 2
1,68 (tomo el 1), y multiplico 0,68 por 2
1,36 (tomo el 1), y multiplico 0,36 por 2
0,72
1,44
0,88
1,76
1,52
1,04

Y así, forme la secuencia: 011010111
Entonces, al unir la parte entera y la decimal:
1000,011010111
Supongo que esto lo voy a almacenar en un registro de 32 bits (mi polarización sera 64, porque usare un ancho de 7 bits para el exponente (0..127, ½ = 64)
Entonces, el exponente sera: 64-20=44 (20 es la posición (bit) de la coma, contando desde donde comienza la posición de la coma, en este caso, es mantisa entera, comienza desde la derecha, y recordar que el conteo comienza en 0..31, desde la derecha)
entonces, queda:



Como convertir de numero flotante binario nuevamente a decimal

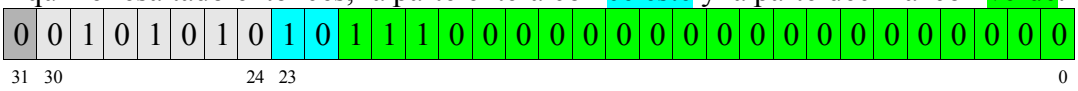
Tomemos de ejemplo el siguiente numero:
Esta expresado con mantisa entera.



Como S = 0, sabemos que el numero es positivo.

E+B = 42-64 = -22; esto nos indica correr la coma hacia la izquierda 22 lugares, donde se posicionara en el siguiente lugar de la mantisa asi:
10,11100...0

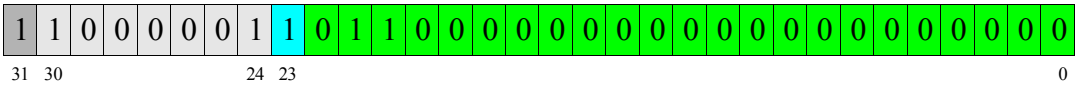
Aquí he resaltado entonces, la parte entera con celeste y la parte decimal con verde:



De aquí surge que:
10 = parte entera = 2 ($1.2^1+0.2^0$)
111 = parte decimal = 0,875 (esto es: $2^{-1}+2^{-2}+2^{-3}$, es decir ½ + ¼ + 1/8)
Entonces, el numero es: 2,875

O sea, aquí deducimos que, la parte entera se calcula como cualquier entero binario, mientras que la parte decimal, son potencias de 2 negativas, que es lo mismo que ½, ¼, 1/8, 1/16, 1/32, 1/64, 1/128, etc es decir 1 dividido la potencia de 2 creciente correspondiente.

Ejemplo con mantisa fraccionaria



Como S = 1, el numero es negativo
El exponente es 65; 65-64 = 1 => corro la coma (desde la izquierda, esto es mantisa fraccionaria) una posición hacia la derecha, quedando situada después del 1 resaltado con celeste.
La parte entera quedo resaltada con celeste, la fraccionaria con verde.
Entonces, este numero es:
parte entera = 1
parte fraccionaria = 0 + ¼ + 1/8 = 0,375
El numero seria 1,375, pero como ya mencionamos, S = 1, entonces el numero es **-1,375**