Notes on Computer Science

Jacopo Tissino

October 11, 2016

Chapter 1

Numbers

1.1 Natural Numbers

1.1.1 Change of basis

We are looking for a practical, recursive formula to change between bases. The problem is posed as such: given a number q in a base R_1 find the coefficients b_n such that in base R_2

$$q = \sum_{i=0}^{n-1} a_i R_1^i = \sum_{i=0}^{m-1} b_i R_2^i \tag{1.1.1}$$

We can write q as:

$$q_0 = b_0 + R_2 \left(b_1 + R_2 \left(b_2 + R_2 \left(b_3 + R_2 \left(b_4 + R_2 \cdots \right)\right)\right)\right) = b_0 + R_2 q_1 \tag{1.1.2}$$

Or in general

$$q_j = b_j + R_2 q_{j-1} \tag{1.1.3}$$

So, given q, we can divide through by R_2.

1.2 Relative numbers

There are four different ways to represent relative numbers, three of which are still in use.

1.2.1 Complements

Given a number x with n digits in base R, the *complement to the root* R is:

$$C_R(x) = R^n - x \tag{1.2.1}$$

The complement to the root R diminished by one is:

$$C_{R-1}(x) = (R^n - 1) - x = (R^n - x) - 1 = C_R(x) - 1 \tag{1.2.2}$$

In binary, the complement to a number to 1 is just the number with every bit switched. This gives us an easy way to also compute the complement to 2.

1.2.2 Representing sign

A computer will need to use a bit to represent sign, in the position of the Most Significant Bit. The bit will be 0 for positive numbers and 1 for negative numbers.

So the final value of the number $d = d_n d_{n-1} \cdots d_2 d_1 d_0$ will be:

$$d = (1 - 2d_n) \sum_{i=0}^{n-1} d_i R^i \tag{1.2.3}$$

Thus, we need to decide beforehand what the maximum number of bits our number can consist of will be. There are the ways:

1. Modulus and Sign
2. Complement to One
3. Complement to Two
4. Excess-q

Modulus and Sign

This is the method we saw earlier. First, we calculate the modulus, and then we add a leading bit for the sign. We need to define the total number of bits to represent the number (we do count the leading bit).

With n bits, we can represent any integer in $[2^{n-1} + 1; 2^{n-1} - 1]$. This representation is, however, redundant: $+0 \neq -0$.

This was used in the first computers, like the IBM 7090. It is still used for the significant part of numbers.

Complement to One

$C_1(C_1(x)) = x$. We still add a leading bit to represent sign. However, we represent negative numbers as the complement (to one) of their positive counterparts.

Our interval is still $[2^{n-1} + 1; 2^{n-1} - 1]$. $-0 \neq +0$

Complement to Two

We add 1 to the complement to one, or we switch the sign of all the bits starting from the MSB, up to the one left of the rightmost 1.

Here we can use the same circuit for adding and subtracting. The interval is now $[2^{n-1}; 2^{n-1} - 1]$. Over- or underflow can only happen if the two numbers have the same sign.

When adding, if the last two amount carried forward are equal the result is correct, otherwise either over or underflow happened.

Excess-q

We offset every number by adding $q = 2^{n-1} - 1$. So $-2^{n-1} + 1$ is represented as n zeroes, and the allowed interval is $[-2^{n-1} + 1, 2^{n-1}]$. Unlike all the other methods, here if the first bit is a 0 the number is negative and vice versa.

1.3 Real Numbers

1.3.1 Fixed point (q-point)

The number of bits we use to represent the number is written as: "$Qm.b$", where 1 bit is for the sign, m are for the floor of the number, and b are for the fractionary part.

The *resolution* is $1/2^b$. The number can be represented as:

$$x = \frac{1}{2^b} \sum_{i=0}^{N} x_i 2^i \tag{1.3.1}$$

where $N = m + b + 1$. We ALWAYS chop the number without rounding. When dealing with negative q-point numbers, we add one to the LSB (not the units digit) to compute the two-complement.

To represent the product of $Qm_1.b_1$ and $Qm_2.n_2$ we need $m_1 + m_2 + 1$ bits for the floor and $b_1 + b_2$ for the fractionary part.

1.3.2 Floating point

The number is represented by a mantissa M_R, an exponent E_R, and a sign bit ($\pm$). Then $x = \pm M_R R^{E_R} = \pm M 2^E$. The mantissa will always have a leading 1, which is omitted.

The absolute error is $|x - \mathrm{fl}(x) \leq b^{E+1-p}$.

The relative error is $|x - \mathrm{fl}(x)|/|x| \leq b^{-p+1}$, also called "precisione macchina".

In the IEEE-754 standard, 1 bit is for the sign, 8 are for the exponent, 23 are for the mantissa.

The exponent is expressed in offset-127, but the values 0 and 255 are reserved.

Contents