THE BASIC ARITHMETIC AND MATHEMATICS OF TWO'S COMPLEMENTARY COMPUTATIONS

George Epstein

Computer Science Department
Indiana University
Bloomington, Indiana 47401

TECHNICAL REPORT No. 26

THE BASIC ARITHMETIC AND MATHEMATICS OF TWO'S COMPLEMENTARY COMPUTATIONS

GEORGE EPSTEIN

APRIL, 1975

The Basic Arithmetic and Mathematics Of Two's Complementary Computation

George Epstein
Computer Science Department
Indiana University
Bloomington, Indiana 47401

This note provides basic information on two's complementary computations, describing addition, subtraction, left and right shifts, multiplication, and division. An appendix provides mathematical detail on the multiplication and division processes.

INTRODUCTION

It is essential that there be a distinction between the computer representation of a number and its actual arithmetic value. A computer represents a number by means of a sign digit x_0 and non-sign digits x_i ($i = 1, 2 \text{ --- } 18$, say); that is, the computer contains the number in the form $x_0\ x_1\ x_2\ x_3\ x_4 \text{ --- } x_{16}\ x_{17}\ x_{18}$. The relationship between the actual arithmetic value x, and the binary digits $x_0, x_1, x_2, \text{ --- } x_{18}$ depends on the arithmetic system being used.

In two's complementary computations $-1 \le x < 1$ and the arithmetic value x of the number is given by the equation

$$x = -x_0 + \sum_{i=1}^{18} 2^{-i} x_i$$

where the sign digit x_0 is 0 if x is positive and x_0 is 1 if x is negative. Thus for positive numbers ($x_0 = 0$),

$$x = \sum_{i=1}^{18} 2^{-i} x_i \quad ,$$

and for negative numbers $(x_0 = 1)$,

$$x = -1 + \sum_{i=1}^{18} 2^{-i} x_i \quad .$$

Example

$+7/8 = 1\cdot 1/2 + 1\cdot 1/4 + 1\cdot 1/8 + 0\cdot 1/16 + 0\cdot 1/32 + \text{---} + 0\cdot 2^{-18}$

so that the computer contains $+7/8$ as $0.111000\cdots00$.

Example

$-7/8 = -1 + 1/8$ and

$1/8 = 0\cdot 1/2 + 0\cdot 1/4 + 1\cdot 1/8 + 0\cdot 1/16 + 0\cdot 1/32 + \text{---} + 0\cdot 2^{-18}$

so that the computer representation of $-7/8$ is $1.0010000\cdots00$.

In other words, if x is negative, write the computer representation of the positive number $x + 1$ and set $x_0 = 1$ to obtain the computer representation for x .

An alternate procedure for obtaining the computer representation of a negative number is to write the computer representation of the absolute value of the number, replace zeros by ones and ones by zeros and add 2^{-18} .

Example

$|-7/8| = +7/8 = 0.111000\cdots00$

$1.000111\text{---}11$ ones replaced by zeros and zeros by ones

$+ \underline{0.000000\text{---}01}$ add 2^{-18}

$1.001000\text{---}00$ representation of $-7/8$.

Example

What does $1.111\text{---}1100$ represent?

This is a negative number. To find its absolute value, subtract it from 0 .

$$\begin{array}{l} 0.00\text{---}00000 \\ \underline{1.11\text{---}11100} \\ 0.00\text{---}00100 \quad \text{representation of } 2^{-16} \end{array} .$$

Hence the given number is -2^{-16} .

Example

Positive full scale = $1 - 2^{-18} = 0.111\text{---}111$.

Example

Negative full scale = $-1 = 1.000\text{---}000$.

Example

$-1 + 2^{-18} = 1.000\text{---}001$.

Example

$-2^{-18} = 1.1111\text{---}111$.

LEFT SHIFT

Let a number in the computer have the arithmetic value x . Then the number which represents $2^n \cdot x$ is obtained by shifting the original number left n times and inserting zeros in the n vacant positions at the right of the number (least significant end).

Example

$-2^{-18} = 1.111\text{---}1111$.

$2^2 \cdot (-2^{-18}) = -2^{-16} = 1.111\text{---}1100$ which is $1.111\text{---}1111$ shifted left twice.

A number can be shifted left once (i.e., multiplied by 2) only if $x_0 = x_1$; otherwise the range of the computer is exceeded. If the number x is positive ($x_0 = 0$) and the most significant

digit $x_1 = 1$, then $x \geq 1/2$ and $2x \geq 1$, which is outside the range of the computer. Similarly, if x is negative ($x_0 = 1$) , and $x_1 = 0$ then $x < -1/2$ so that $2x < -1$, which exceeds the range of the computer.

RIGHT SHIFT

The number which represents $2^{-n} \cdot x$ is obtained by shifting the original number right n times. The sign digit must be preserved, and also propagated into the n -most significant positions just after the sign digit (i.e., $x_0 = x_1 = x_2 = \text{---} = x_n$ in the shifted result).

Example

0.11100---00 shifted right one time is 0.011100---00 ; that is, $1/2 \cdot 7/8 = 7/16$.

Example

1.001000---0 shifted right twice is 1.1100100---0 ; that is, $1/4 \cdot (-7/8) = -7/32$.

ADDITION

The sum of x and y is obtained by adding the binary number $x_0\ x_1\ x_2\ \text{---}\ x_{18}$ to $y_0\ y_1\ y_2\ \text{---}\ y_{18}$ in usual binary fashion. If $-1 \leq x + y < 1$, the result is the correct sum in two's complementary form.

Example

```
  1.01100---00   representation of  -5/8
+ 1.11010---00   representation of  -3/16
  1.00110---00   representation of  (-5/8) + (-3/16) = -13/16 .
```

Example

1.010100---00 representation of -11/16

\+ 0.100100---00 representation of +9/16

1.111000---00 representation of (-11/16) + (9/16) = -1/8 .

SUBTRACTION

Subtraction is performed by subtracting the binary number $x_0\ x_1\ x_2\ ---\ x_{18}$ from $y_0\ y_1\ y_2\ ---\ y_{18}$ in usually binary fashion. If $-1 \leq x - y < 1$, the result is the correct difference in two's complementary form.

Example

1.11010---00 representation of (-3/16)

\- 1.01100---00 representation of (-5/8)

0.01110---00 representation of (-3/16) - (-5/8) = 7/16 .

Example

1.111100---00 representation of -1/16

\- 0.011000---00 representation of +3/8

1.100100---00 representation of (-1/16) - (3/8) = -7/16 .

MULTIPLICATION

The multiplication process for numbers in two's complementary form is described by the following rules. The successive partial products and final product are contained in a fixed register, or accumulator, which accomodates a sign bit and eighteen non-sign bits.

1. Set the accumulator to its initial value, p_0 (i.e., set the initial partial product).

2. Examine the least significant digit of the multiplier, y .
3. If it is a 1 and the sign bits of the multiplicand and accumulator are opposite, add multiplicand, x , to contents of accumulator and shift right once, preserving the sign digit of the sum; otherwise, insert the sign digit of the multiplicand into the vacant sign digit position.
4. If it is a 0 , shift contents of accumulator right once, preserving the sign digit.
5. Shift multiplier right once, preserving the sign digit.
6. Repeat steps 2-5 until all non-sign digits of the multiplier have been sensed.
7. For the last step in the process, examine the least significant digit of the multiplier. (This is now the sign digit of the multiplier since all non-sign digits have already been sensed and shifted through.) If the sign digit of the multiplier is 1 (negative multiplier), subtract the multiplicand from the accumulator. If the sign digit of the multiplier is 0 (positive multiplier), do nothing.

There is no shift at this last step. The accumulator contains the final product in two's complementary form. The example below illustrates the process. For the sake of brevity there are only 4 non-sign bits.

8. The final answer in the accumulator is $y \cdot x + 2^{-18} p_0$.

Example

Multiplier	Accumulator	Multiplicand = 1.1101
1.0110	0.0000	
sense 0	0.0000	shift right once
sense 1	1.1101	add multiplicand to accumulator

Multiplier	Accumulator	Multiplicand = 1.1101
	1.1110	shift right once, inserting sign digit of multiplicand.
sense 1	1.1011	add multiplicand to accumulator.
	1.1101	shift right once, inserting sign digit of multiplicand.
sense 0	1.1110	shift right once, preserving sign digit.
sense 1 multiplier is negative	1.0001	subtract multiplicand; do not shift.
Final product	= 0.0001	

DIVISION

The following are the rules for dividing:

1. Examine sign digits of partial remainder (the initial partial remainder is the dividend) in accumulator and the divisor.
2. Shift accumulator and quotient register left once.
3. If the sign digits in 1. were alike, write a one to the right of the digits of the quotient thus far obtained and subtract the divisor from the accumulator.
4. If the sign digits in 1. were not alike, write a zero to the right of the digits of the quotient thus far obtained and add the divisor to the accumulator.
5. On the first word time, however, take the complement of the inserted least significant digit of the quotient and set

all digits of the quotient to this value (i.e., set all digits of the quotient to the correct sign digit of the quotient).

6. Repeat above steps until all quotient bits are obtained.

Example

Divide 0.001111 by 1.011000 .

	Accumulator		Quotient
(1)	0.001111		1.111111
	0.011110	shift left	
(2)	1.110110	add divisor	1.111111
	1.101100	shift left	
(3)	0.010100	subtract divisor	1.111110
	0.101000	shift left	
(4)	0.000000	add divisor	1.111100
	0.000000	shift left	
(5)	1.011000	add divisor	1.111001
	0.110000	shift left	
(6)	1.011000	subtract divisor	1.110011
	0.110000	shift left	
(7)	1.011000	subtract divisor	1.100111
	Final quotient		= 1.100111

Note that it requires 7 steps to obtain the 6 non-sign bits, the first step being devoted to obtaining the sign bit of the quotient. If the process were stopped at step 5, the quotient register would contain the correct quotient shifted right twice. In general, it is possible to stop the division process at any step, obtaining a shifted quotient in the quotient register.

APPENDIX

It is easy to verify that the above rules yield the correct results. The proofs for the multiplication and division processes are below.

A. Multiplication

p_k is the partial product at the k^{th} step, p_o is the number initially in the accumulator, y is the multiplier, and x is the multiplicand.

The non-sign digits of the computer representation of y are given by

$$\sum_{i=1}^{18} 2^{-i} y_i = y + y_o \quad .$$

According to the rules stated for multiplication the recursion formulas for partial products is

$$p_{k+1} = 1/2(p_k + y_{18-k} x) \quad .$$

From this it follows that

$$p_{18} = 2^{-18} p_o + \left(\sum_{i=1}^{18} 2^{-i} y_i \right) x = 2^{-18} p_o + (y + y_o)\, x \quad .$$

After the correction of step 7 (i.e., after subtracting $y_o \cdot x$)

$$p_{18} = 2^{-18} p_o + y \cdot x \quad .$$

The range of the computer must never be exceeded. At any step in the process, if N is the number in the accumulator and $-1 \leq N < 1$, it is either shifted right once or added with the multiplicand and then shifted right once. It is clear that

$$-1 \leq 1/2(N + y_o x) < 1$$

and the number in the accumulator is always in range.

If $p_o = 0$, $p_{18} = yx$, the desired result, called the truncation product. Thus the accumulator is cleared to zero at the first step. With this condition the absolute value of the multiplicand is always greater than the absolute value of the number in the accumulator at any step, and so, if the multiplicand is added to the accumulator, as in step 3, the sign digit of the result is the same as the sign digit of the multiplicand. If $p_o = 1/2$, $p_{18} = 2^{-18} \cdot 1/2 + y \cdot x = 2^{-19} + y \cdot x$ and p_{18} is the desired result for multiplication with unbiased round-off, called the round product.

B. Division

The method presented here is a non-restoring process. For an arbitrary base b the digits $-(b-1)$, --- -1 , $+1$, --- $+(b-1)$ are required for the quotient, a total of $2(b-1)$ digits. This may be thought of as a duplex number system without zero. The example below illustrates the process for a division with $b = 10$.

```
  (2)(-8)6.(-9)(-9)
8|1000
 -16
  -60
  +64
    40
   -48
    -80
    +72
     -80
     +72
      -8---
```

For the binary system there is a set of digits b_i which are $+1$ or -1 . The problem is to find the equivalent set of digits x_i which are 0 or $+1$ for the representation of the binary fraction in two's complementary form.

That is, given $x = \sum_{i=1}^{19} x_i 2^{-i}$ $\qquad\qquad$ $b_i = +1, -1$

find x_i such that

$$x = x_o + \sum_{i=1}^{18} x_i 2^{-i} \qquad x_i = 0, 1$$

First let $b_i = 2a_{i-1} - 1$

Then

$$x = \sum_{i=1}^{19} b_i 2^{-1} = \sum_{i=1}^{19} a_{i-1} 2^{-(i-1)} - \sum_{i=1}^{19} 2^{-i}$$

$$= a_o + \sum_{i=1}^{18} a_i 2^{-i} - (1-2^{-19})$$

$$= (a_o - 1) + \sum_{i=1}^{18} a_i 2^{-1} + 2^{-19}$$

Thus, the equivalent complementary representation is found by letting

$$x_o = 1 - a_o \quad x_i = a_i \quad x_{19} = 1$$

That is, replace (-1)'s by zeros, shift left, inserting 1 in the 19^{th} digit position, and complement the sign digit.

The recursion relationship for partial remainders is

$$r_{n+1} = 2r_n - (-1)^{p_n + y_o}$$

where p_n is the sign digit of the n^{th} partial remainder r_n. y is the divisor. Then

$$r_{19} 2^{-19} = r_o - y \sum_{i=1}^{19} 2^{-i} (-1)^{y_o + p_{i-1}} .$$

The expression for the n^{th} quotient digit inserted, z_n, which is $(+1)$ or (-1), is $z_n = (-1)^{p_{n-1} + y_o}$.

The arithmetic value of the quotient q is then

$$q = \sum_{i=1}^{19} z_i 2^{-i} = \sum_{i=1}^{19} 2^{-i} (-1)^{p_{i-1} + y_o} .$$

To check the accuracy of the process, multiply quotient by divisor, add the remainder, and compare with the dividend.

$$qy + 2^{-19}r_{19} - r_o = y \sum_{i=1}^{19} 2^{-i}(-1)^{p_{i-1}+y_o}$$

$$+ r_o - y \sum_{i=1}^{19} 2^{-i}(-1)^{y_o+p_{i-1}}$$

$$- r_o = 0 \quad .$$

Looking at the rules for converting a binary fraction with digits (+1) and (-1) to a fraction with digits 1 and 0 , we see that all that is wanted is a complementation of the sign digit. This is accomplished by step 5, at the same time making provision for a variable length order. The insertion of a 1 in the 19^{th} digit position falls outside computer accuracy.

Typed by Christopher Charles