An Efficient Algorithm to Compute the Reciprocal of a Binary Number

P.C. Perera

Department of Engineering Mathematics, Faculty of Engineering
University of Peradeniya

Introduction

In recent times, much research has been focused on developing both time and memory efficient algorithms for solving complicated mathematical problems. Contrarily, very efficient algorithms for calculation of simple operations that are performed very frequently, tremendously improve almost every complicated computation performed by a computer. The algorithm developed in this work may be used to compute the reciprocal of a binary integer while eliminating the underflow errors occurred due to the limitations of length of registers employed in a CPU.

Preliminaries

Let N be the binary integer of which the reciprocal should be computed. Denote $\mathcal{A} = \{0,1\}$. To avoid trivialities, let the least significant digit of N be 1. Suppose $k \in \mathbb{N}$ is the smallest number such that $N < 2^k$. Then $N = \sum_{i=0}^{k-1} a_i 2^i$, where $a_{k-1} = 1$ and $a_i \in \mathcal{A}$ for $i \in \{1, 2, ..., k-2\}$. The assumption on the least significant bit makes $a_0 = 1$. It turns out that

$$\frac{1}{N} = \frac{1}{\sum_{i=0}^{k-1} a_i 2^i} = \frac{1}{2^k - \sum_{i=0}^{k-1} b_i 2^i},$$

where $a_0 = b_0 = 1$ and $b_i = \bar{a}_i$ for $1 \le i \le k-1$.

The binomial expansion of $\dfrac{1}{1-x}$ yields that

$$\frac{1}{N} = \frac{1}{2^k} \sum_{j=0}^{\infty} \left(\sum_{i=0}^{k-1} b_i 2^{i-k} \right)^j .$$

Considering the binary expansion of $\frac{1}{N}$, we have

$$\frac{1}{N} = \frac{1}{2^k} \sum_{j=0}^{\infty} \left(\sum_{i=0}^{k-1} b_i 2^{i-k} \right)^j = 2^{-k} + \sum_{i=k+1}^{\infty} d_i 2^{-i}, \tag{1}$$

where $d_l \in \{0,1\}$ is the binary digit corresponding to the l^{th} binary place of the binary sequence of the reciprocal of N.

The objectives of this work are

(i) to determine the number of binary places to be determined in order to obtain the exact answer, and

(ii) to devise an efficient algorithm to determine d_i for $i \ge k+1$.

Theorems and lemmas

For the sake of brevity, the proofs of the lemma and the theorems presented in this section are omitted in this paper.

It is a known fact that the binary sequence of every rational number $\mathbb{Q}$ has a recursive subsequence in it.

Example 1. *Let* $x = \dfrac{1}{7}$. *Clearly* $x \in \mathbb{Q}$, *and* $\frac{1}{7} = 0.142857142857... = 0.\overline{142857}$. *If* $x = \frac{1}{101_2} \in \mathbb{Q}$, *then* $\frac{1}{101_2} = 0.00110011..._2 = 0.\overline{0011}_2$.

Lemma 1. *If N is a prime, $N|2^{N-1}-1$ and the upper bound for the length of the string repeated in the binary sequence of $\frac{1}{N}$ is $N-1$. For a given prime N, there exists $a \in \mathbb{N}$ such that $N|2^{(N-1)/a} - 1$ and the exact length of the string repeated in the binary sequence of $\frac{1}{N}$ is $\frac{N-1}{a}$.*

Example 2. *Recall that $N = 17$ is a prime. As the second assertion of Lemma 1, the smallest number in the sequence $\{2^i - 1\}_{i=1}^{N-1}$ which is divisible by 17 is 255 which is*

$$2^{(17-1)/2} - 1 \text{ or } 2^{(N-1)/a} - 1 \text{ with } a = 2.$$

For composite N, the following theorem is applicable.

Theorem 1. *Let $N \in \mathbb{N}$ and the prime factorization of N be given by $N = p_1^{r_1}p_2^{r_2}...p_m^{r_m}$. Suppose a_i is the largest positive integer such that $p_i|\left(2^{(p_i-1)/a_i} - 1\right)$ for $i = 1, 2, ..., m$. Then there exists a least positive integer $\hat{N}$ such that $N|2^{\hat{N}} - 1$ and is given by*

$$\hat{N} = \left(\frac{(p_1-1)}{a_1}p_1^{r_1-1}, \frac{(p_2-1)}{a_2}p_2^{r_2-1}, ..., \frac{(p_m-1)}{a_m}p_m^{r_m-1}\right).$$

Moreover, the exact length of the recursive portion of the binary sequence of $\frac{1}{N}$ is $\hat{N}$.

Example 3. *Let $N = 119 = 7 \times 17$. Since $7|2^3 - 1$ and $17|2^8 - 1$, as in Theorem 1, the length of the recursive binary sequence is $3 \times 8 = 24$. In particular*

$$\frac{1}{119} = \frac{1_2}{1110111_2} = 0.\overline{000000100010011010111001}_2$$

The required binary digits in the sequence can be determined by constructing a denominator dependent difference equation as indicated in the following theorem. The sum of the outputs produced by the difference equation converges to the reciprocal of our interest.

Theorem 2. *Let $N = \sum_{i=0}^{k-1} b_i 2^i$ where $b_i \in \mathcal{A}$ $\forall\ i$. Then $\frac{1}{N} = \sum_{i=1}^{\infty} 2^{-i}x_i$ and y_i for $i = \mathbb{N}$ are the sequence of solutions to the k^{th} order difference equation $y_{n+k} = \left(\sum_{i=1}^{k-1} \bar{b}_i y_{n+i}\right) + b_0 y_n$ with initial conditions $y_i = 0$ for $i = 1, ..., k-1$ and $y_k = 1$ such that the sequence $\{z_j\}_{j=1}^{\infty}$ has the property $z_{r-k+1} = \left|\sum_{i=1}^{r} 2^{-i}x_i - \sum_{i=1}^{r} 2^{-i}y_i\right| \to 0$ as $r \to \infty$.*

Example 4. *Let $N = 15$. Thus, $N = 1111_2$. The corresponding difference equation is $y_{n+4} = y_n$ with initial conditions $y_i = 0$ for $i = 0, 1, 2$ and $y_3 = 1$. Thus it turns out that*

$$y_n = \begin{cases} 1 & \text{if } n = 4k - 1 \\ 0 & \text{otherwise} \end{cases}$$

$y_4 = y_5 = y_6 = 0$, $y_7 = 1$, $y_8 = y_9 = y_{10} = 0...$ and by continuing this process $\frac{1_2}{1111_2} = 0.000100010001.. = 0.\overline{0001}$ can be computed accurately.

Example 5. *Let $N = 119$. Thus, $N = 1110111_2$. The corresponding difference equation is $y_{n+7} = y_{n+3} + y_n$ with initial conditions $y_i = 0$ for $i = 0, 1, ..., 6$ and $y_7 = 1$. Thus it turns out that $y_8 = y_9 = y_{10} = y_{12} = y_{13} = y_{16} = y_{17} = y_{20} = y_{24} = 0_2$, $y_{11} = y_{14} = y_{15} = y_{19} = y_{21} = y_{23} = y_{27} = y_{28} = 1_2$, $y_{18} = 10_2$, $y_{22} = y_{25} = 11_2$, $y_{26} = 100_2...$ and after 26^{th} iteration, truncating the tail after the 24^{th} binary place (in the light of Example 3), it yields that*

$$\frac{1_2}{1110111_2} = 0.\overline{000000100010011010111001}_2.$$

Conclusions

The algorithm outlined in this paper outperforms all of its counterparts in the context of division of integers and is applicable while utilizing registers with any length as opposed to the other available algorithms. The aforemention method which is based on difference equations, guarantees faster convergence and accurate results when all or most of the binary digits of N are equal to 'one'. Otherwise, the convergence may be slow requiring large number of iterations for the exact solution. Even though this approach is promising, avenues are open for improvements.

References

Bennett, W.S. (1973) Quotient generation with conventional binary multipliers, *Proceedings of IEEE*, 61(5), 664-665