Arbitrary-Precision Division*

Rod Howell
Kansas State University

September 18, 2000

This paper presents an algorithm for arbitrary-precision division and shows its worst-case time complexity to be related by a constant factor to that of arbitrary-precision multiplication. The material is adapted from [1], pp. 264, 295-297, where S. A. Cook is credited for suggesting the basic idea.

We assume a smooth bound $g(n) \in \Omega(n)$ for the worst-case time complexity of n-bit fixed-point multiplication. Furthermore, we assume that for some $n_0 \in \mathbb{N}$ and some real $c > 1$, $g(2n) \geq cg(n)$ for all $n \geq n_0$. Intuitively, this condition ensures that $g(n)$ eventually maintains a growth rate of at least n^ϵ for some $\epsilon \in \mathbb{R}^+$ (i.e., it does not grow more slowly than this for arbitrarily long periods of time).

In order to simplify the problem, we will restrict the input to positive integers $u \geq v$. In particular, we wish to find $\lfloor u/v \rfloor$. Suppose v is an m-bit integer; i.e., $2^{m-1} \leq v < 2^m$. Then

$$\left\lfloor \frac{u}{v} \right\rfloor = \left\lfloor u2^{-m}\left(\frac{1}{v2^{-m}}\right)\right\rfloor,$$

and $1/2 \leq 1/(v2^{-m}) < 1$. We therefore begin by presenting an algorithm to find a high-presision approximation for $1/x$, where x is a fixed-point rational number, $1/2 \leq x < 1$.

The idea is based on Newton's method, which generates successive approximations according to the following rule:

$$z_{k+1} = 2z_k - xz_k^2$$

This method converges very quickly: if $z_k = (1-\epsilon)/x$, then

$$\begin{aligned} z_{k+1} &= \frac{2(1-\epsilon)}{x} - x\left(\frac{1-\epsilon}{x}\right)^2 \\ &= \frac{2-2\epsilon-1+2\epsilon-\epsilon^2}{x} \\ &= \frac{1-\epsilon^2}{x} \end{aligned}$$

However, the time for convergence depends upon the accuracy required. Thus, the total time is not within a constant factor of the time to multiply. In order accomplish this goal, we use roughly the high-order half of x to obtain recursively an approximation of roughly half the needed accuracy, then apply Newton's method a single iteration to obtain the desired result.

We assume the following functions:

- $\mathit{trunc}(x, p)$: returns the fixed-point x truncated to p bits to the right of the radix point. Thus, $0 \leq x - \mathit{trunc}(x, p) < 2^{-p}$.
- $\mathit{roundup}(x, p)$: returns the fixed-point x rounded up to p bits to the right of the radix point. Thus, $0 \leq \mathit{roundup}(x, p) - x < 2^{-p}$.

We now define $\mathit{reciprocal}(x, p)$ as follows:

```
function reciprocal(x, p)
begin
  if p ≤ 2
    then
        return trunc(3/2, p)
     else
        z ← reciprocal(x, ⌊p/2⌋ + 1)
        return roundup(2z − trunc(x, p + 2)z², p)
  fi
end
```

We will first show the correctness of the algorithm. The following theorem follows from the definitions of *trunc* and *roundup*:

Theorem 1 *$\mathit{reciprocal}(x, p)$ returns a value with at most p bits to the right of the radix point.*

We need the following lemma in order to bound the error incurred by *reciprocal*.

Lemma 1 *The value returned by* $\mathrm{reciprocal}(x, p)$ *is at most 2.*

Proof: The lemma clearly holds when $p \leq 2$. Suppose $p > 2$. Consider the expression

$$2z - \mathit{trunc}(x, p + 2)z^2.$$

Because $z^2 \geq 0$, the value of this expression is maximized when $\mathit{trunc}(x, p + 2)$ is minimized. Thus, it suffices to show

$$2z - \frac{z^2}{2} \leq 2. \tag{1}$$

Rearranging terms, we find that (1) holds iff

$$0 \leq z^2 - 4z + 4 = (z + 2)^2,$$

which holds for all $z \in \mathbb{R}$. □

The following theorem shows the accuracy of the value returned by *reciprocal*:

Theorem 2 *reciprocal*(x,p) *returns a value* y *such that*

$$\left|\frac{1}{x} - y\right| \leq 2^{1-p}$$

Proof: By generalized induction on p.

Base Case 1: $p = 0$. The value returned is 1. Because $1/2 \leq x < 1$,

$$\begin{aligned}\left|\frac{1}{x} - 1\right| &= \frac{1}{x} - 1 \\ &\leq 1 \\ &\leq 2^1.\end{aligned}$$

Base Case 2: $1 \leq p \leq 2$. Because $3/2$ requires only 1 bit to the right of the radix point, the value returned is $3/2$. Then

$$\begin{aligned}\left|\frac{1}{x} - \frac{3}{2}\right| &\leq \frac{1}{2} \\ &= 2^{-1} \\ &\leq 2^{1-p}.\end{aligned}$$

Induction Step: Let $p > 2$. Let $1/x + \alpha$ be the value returned by

$$\textit{reciprocal}(x, \lfloor p/2 \rfloor + 1).$$

By Lemma 1,

$$\frac{1}{x} + \alpha \leq 2.$$

By the Induction Hypothesis,

$$|\alpha| \leq 2^{\lfloor p/2 \rfloor}$$

Let β be the value truncated by the call to *trunc*; i.e.,

$$\beta = x - \textit{trunc}(x, p+2).$$

Then

$$0 \leq \beta < 2^{-p-2}.$$

Let γ be the value added by the call to *roundup*;. i.e.,

$$\gamma = \textit{roundup}(2z - \textit{trunc}(x, p+2), p) - (2z - \textit{trunc}(x, p+2)).$$

Then

$$0 \leq \gamma < 2^{-p}.$$

Then the value y returned is given by

$$\begin{aligned} y &= 2\left(\frac{1}{x}+\alpha\right) - (x-\beta)\left(\frac{1}{x}+\alpha\right)^2 + \gamma \\ &= \frac{2}{x} + 2\alpha - \frac{1}{x} - 2\alpha - x\alpha^2 + \beta\left(\frac{1}{x}+\alpha\right)^2 + \gamma \\ &= \frac{1}{x} - x\alpha^2 + \beta\left(\frac{1}{x}+\alpha\right)^2 + \gamma. \end{aligned}$$

We need to derive a bound on $|\beta(1/x+\alpha)^2 + \gamma - x\alpha^2|$. First, we have

$$\begin{aligned} 0 \leq \beta\left(\frac{1}{x}+\alpha\right)^2 + \gamma &\leq 2^{-p-2} \cdot 2^2 + 2^{-p} \\ &= 2^{-p} + 2^{-p} \\ &= 2^{1-p} \end{aligned}$$

Furthermore,

$$\begin{aligned} 0 \leq x\alpha^2 &\leq 2^{-2\lfloor p/2 \rfloor} \\ &\leq 2^{-2(p-1)/2} \\ &= 2^{1-p} \end{aligned}$$

Therefore,

$$\left|\beta\left(\frac{1}{x}+\alpha\right)^2 + \gamma - x\alpha^2\right| \leq 2^{1-p}$$

□

We are now ready to show the worst-case time complexity of *reciprocal*. Recall that $g(n)$ is a bound on the worst-case time complexity for multiplying two n-bit fixed-point numbers. We will show that the time complexity for *reciprocal* satisfies a recurrence of the form

$$t(n) = t(n/2) + cg(n)$$

where n is a sufficiently large power of 2. We therefore need the following lemma.

Lemma 2 *Let $f : \mathbb{N} \to \mathbb{R}^{\geq 0}$ be a smooth function such that $f(2n) \geq cf(n)$ for some $c > 1$ whenever $n \geq n_0 \in \mathbb{N}$. Let $t : \mathbb{N} \to \mathbb{R}^{\geq 0}$ be an eventually nondecreasing function satisfying*

$$t(n) = t(n/2) + f(n)$$

when $n = n_0 2^k$ for some $k \geq 1$. Then $t(n) \in \Theta(f(n))$.

Proof: Because f is smooth, it is eventually positive. Without loss of generality, we may assume that $f(n) > 0$ for $n \geq n_0$. Because f is smooth, it suffices to show that

$$t(n) \in \Theta(f(n) \mid n = n_0 2^k \text{ for some } k \geq 1).$$

Because $t(n) \geq 0$ for all n, clearly,

$$t(n) \in \Omega(f(n) \mid n = n_0 2^k \text{ for some } k \geq 1).$$

We will show by induction on $k \geq 1$ that for $n = n_0 2^k$, $t(n) \leq df(n)$, where

$$d = \max\left\{\frac{1 + t(n_0)}{f(2n_0)}, \frac{c}{c-1}\right\}.$$

Base: $k = 1$. Then $n = 2n_0$, and

$$t(n) = t(n_0) + f(n)$$

Because $d \geq 1 + t(n_0)/f(n)$, we have

$$\begin{aligned} df(n) &\geq \left(1 + \frac{t(n_0)}{f(n)}\right) f(n) \\ &= f(n) + t(n_0) \\ &= t(n) \end{aligned}$$

Induction Hypothesis: Assume that for some $k \geq 1$, $t(k) \leq df(n)$.

Induction Step: $n = n_0 2^{k+1}$. Then

$$\begin{aligned} t(n) &= t(n_0 2^k) + f(n) \\ &\leq df(n_0 2^k) + f(n) \quad \text{from the IH} \\ &= df\left(\frac{n}{2}\right) + f(n) \\ &\leq \frac{df(n)}{c} + f(n) \\ &= \left(1 + \frac{d}{c}\right) f(n) \end{aligned}$$

Because $d \geq c/(c-1)$, we have

$$\begin{aligned} d &\geq \frac{c}{c-1} \\ dc - d &\geq c \\ dc &\geq c + d \\ d &\geq 1 + \frac{d}{c}. \end{aligned}$$

Therefore, $t(n) \leq df(n)$.

$\square$

Theorem 3 *reciprocal(x, p) operates in a time in $O(g(p))$.*

Proof: Suppose $p > 2$. By Theorem 1, the value z contains at most $\lfloor p/2 \rfloor + 1$ bits to the right of the radix point. From Lemma 1, the value of z is at most 2. z can therefore be stored in $\lfloor p/2 \rfloor + 2$ bits. z^2 therefore contains at most $p + 4$ bits. Because $trunc(x, p + 2)$ contains at most $p + 2$ bits, the multiplication

$$trunc(x, p + 2)z^2$$

takes a time in $O(g(p+4))$. Because $g(n) \in \Omega(n)$, this operation dominates the remainder of the work done outside the recursive call. We can therefore bound the total time with the following recurrence:

$$t(p) = t(\lfloor p/2 \rfloor + 1) + cg(p + 4)$$

for some $c \in \mathbb{R}$ and $p > n_0 \in \mathbb{N}$. Let $p = 2^k$, and define

$$\begin{aligned} T(p) &= t(p + 1) \\ &= t\left(\left\lfloor \frac{p+1}{2} \right\rfloor + 1\right) + cg(p + 5) \\ &= t\left(\frac{p}{2} + 1\right) + cg(p + 5) \\ &= T\left(\frac{p}{2}\right) + cg(p + 5) \end{aligned}$$

for $p > n_0$. From Lemma 2, $T(p) \in \Theta(cg(p+5)) = \Theta(g(p)$, because g is smooth. Then

$$\begin{aligned} t(p) &= T(p - 1) \\ &\in \Theta(g(p - 1)) \\ &\in \Theta(g(p)) \end{aligned}$$

Therefore, the time compexity of *reciprocal*(x, p) is in $O(g(p))$. $\square$

We can now use *reciprocal* to construct an integer division algorithm. We assume the existence of a function *numbits*, which takes a natural number and returns the number of bits in its representation. Thus, for $2^{n-1} \leq u < 2^n$, *numbits*(u) returns n. The algorithm is as follows:

```
procedure divide(u, v, var q, r)
begin
  n ← numbits(u); m ← numbits(v)
  x ← reciprocal(v2^(−m), n − m + 1)
  q ← ⌊ux2^(−m)⌋
  r ← u − qv
  if r < 0 then r ← r + v; q ← q − 1
  elsif r ≥ v then r ← r − v; q ← q + 1
  fi
end
```

The following theorem shows the correctness of *divide*.

Theorem 4 *Let $u \geq v$ be positive integers. Upon execution of* $\mathit{divide}(u, v, q, r)$*, q contains the value $\lfloor u/v \rfloor$, and r contains the value $u \bmod v$.*

Proof: From the definition of *numbits*, we have

$$2^{n-1} \leq u < 2^n$$

and

$$2^{m-1} \leq v < 2^m.$$

From Theorem 2,

$$x = \frac{1}{v2^{-m}} + \alpha, \text{ where } |\alpha| \leq 2^{m-n}.$$

It then follows that after the first assignment to q,

$$\begin{aligned} q &= \left\lfloor u\left(\frac{1}{v2^{-m}} + \alpha\right) 2^{-m} \right\rfloor \\ &= \left\lfloor \frac{u}{v} + u\alpha 2^{-m} \right\rfloor \\ |u\alpha 2^{-m}| &\leq |\alpha 2^{n-m}| \\ &\leq 1. \end{aligned}$$

Hence,

$$\left| \left\lfloor \frac{u}{v} \right\rfloor - q \right| \leq 1$$

Clearly, the remaining statements set q and r to $\lfloor u/v \rfloor$ and $u \bmod v$, respectively. □

We conclude by showing the time complexity of divide.

Theorem 5 *Let $u \geq v$ be positive integers containing at most n bits. The worst-case time complexity of* $\mathit{divide}(u, v, q, r)$ *is in $O(g(n))$.*

Proof: From Theorem 3, the call to *reciprocal* takes $O(g(n))$ time. From Theorem 1, x contains at most $n - m + 2$ bits, so the product ux can be computed in $O(g(n))$ time. Finally, q has at most $n - m + 1$ bits, so qv can be computed in $O(g(n))$ time. Because everything else can be done in at most linear time and $g(n) \in \Omega(n)$, the total time is in $O(g(n))$. □

References

[1] Donald Knuth. *The Art of Computer Programming*, volume 2, Seminumerical Algorithms. Addison-Wesley, 2nd edition, 1981.