Abstract

The integer division of a numerator $n$ by a divisor $d$ gives a quotient $q$ and a remainder $r$. Optimizing compilers accelerate software by replacing the division of $n$ by $d$ with the division of $c \cdot n$ (or $c \cdot n + c$) by $m$ for convenient integers $c$ and $m$ chosen so that they approximate the reciprocal: $c/m \approx 1/d$. Such techniques are especially advantageous when $m$ is chosen to be a power of two and when $d$ is a constant so that $c$ and $m$ can be precomputed. The literature contains many bounds on the distance between $c/m$ and the divisor $d$. Some of these bounds are optimally tight, while others are not. We present optimally tight bounds for quotient and remainder computations.

Keywords: Integer Division, Compiler Optimization, Tight Bounds

1. Introduction

The problem of computing the integer division given constant divisors has a long history in computer science [1–4]. Granlund and Montgomery [5] present the first general-purpose algorithms to divide integers by constants using a multiplication and a division by a power of two: their work was adopted by the GNU Compiler Collection (GCC). Given any non-zero 32-bit divisor known at compile time, the optimizing compiler can replace the division by a multiplication followed by a shift. Warren [6] improved on the Granlund and Montgomery technique by deriving a better bound that gives a wider range of choices. Warren’s better approach is found in LLVM’s Clang compiler. Many optimizing compilers rely on equivalent techniques, either based on the original Granlund-Montgomery article or on Warren’s technique.

Robison [7] describes a slightly superior alternative for some divisors in that we multiply and add the multiplier before dividing by a power of two (henceforth the multiply-add technique). Though it comes at the cost of an addition, it allows one to choose a smaller multiplier, which can be advantageous. Robison’s approach is implemented in the popular libdivide library [8].

Most of the literature is focused on the computation of the quotient $q$ of the division of $n$ by $d$. From the quotient $q$, we can compute the remainder as
\[ n - q \ast d. \] We can also compute the remainder directly \[9\] without first computing the quotient: it is given by taking remainder of \( c \ast n \) divided by \( m \), and then multiplying it by \( m \). However, for the remainder and the quotient to be exact, it is necessary that \( c/m \) approximates \( 1/d \) more closely than if we merely need the quotient.

From the computation of remainders, we can derive a divisibility check, that is check whether \( d \) divides \( n \), or, equivalently, check that \( n \) is a multiple of \( d \). Though it may seem that computing the remainder and checking whether it is zero is efficient, we can simplify and accelerate the algorithm by avoiding the computation of the remainder.

The literature commonly assumes that \( m \) is a power of two. We approach the problem more generally, letting \( m \) and \( c \) be any integer, and restricting the numerator to an interval \([0, N]\) where \( N \) can be any integer. It makes our exposition more general, while simplifying the notation.

Some our novel contributions are as follows:

- We improve Robison’s bound \[7\], in a manner similar to how Warren improved Granlund and Montgomery’s bound. That is, we provide an optimal bound for the multiply-add technique \[1\]

- We derive a new tighter bounds for computing the quotient directly and checking the divisibility, thus improving on the work of Lemire at al. \[9\]

- We show that we can adapt Robison’s technique to compute remainders directly and derive a novel bound. We adapt the multiply-add technique for the purpose of a divisibility check. To our knowledge, these results are novel.

All our bounds on how close \( c/m \) must be to \( 1/d \) are optimal and form necessary and sufficient conditions. Table \[1\] presents our core results in concise manner.

2. Other Related Work

The problem of quickly computing the division by a constant in computers dates back to at least the 1970s. Jacobsohn \[2\] shows that we can divide by an odd integer by multiplying by a fractional inverse, followed by some rounding. Artzy et al. \[1\] describe a related algorithm to divide multiples of a known divisor (exact division). Li \[3\] presents algorithms for integer division by all odd integers up to 55 \[11, § 10-18\]. Divisions are executed as series of “shift and add” instructions.

Magenheimer et al. \[12\] describe how to compute the division of integers by odd divisors as a multiplication and an addition followed by a division by a power of two. Their approach was later refined by Robison \[7\]. Similarly,\footnote{Drane et al. \[10\] have a related bound, but they also have additional constraints on the divisor \( d \).}
Table 1: Summary of main results. Throughout, all values are non-negative integers, the divisor is non-zero \(d > 0\), and the numerator is bounded by \(N \geq d\), so that \(n \in [0, N]\). We add the constraint that \(c \in [0, m]\) so that \(c\) is as small as possible.

| Theorem | Statement | Condition |
|---------|-----------|-----------|
| 1 (and Warren [11]) | \(\text{division}(n, d) = \text{division}(c \times n, m)\) for all \(n \in [0, N]\) | \(1/d \leq c/m < \left(1 + \frac{1}{N - \text{remainder}(N + 1, d)}\right)1/d\) |
| 2 (novel, improves Lemire et al. [9, Theorem 1]) | \(\text{division}(n, d) = \text{division}(c \times n, m)\) and \(\text{remainder}(n, d) = \text{division}(\text{remainder}(c \times n, m) \times d, m)\) for all \(n \in [0, N]\) | \(1/d \leq c/m < \left(1 + \frac{1}{N}\right)1/d\) |
| 3 (novel, generalizes Lemire et al. [9]) | \(d\) divides \(n \in [0, N]\) if and only if \(\text{remainder}(c \times n, m) < c\) | \(1/d \leq c/m < \left(1 + \frac{1}{N}\right)1/d\) |
| 4 (improves Robison [14]) | \(\text{division}(n, d) = \text{division}(c \times n + c, m)\) for all \(n \in [0, N]\) | \(1 - \frac{1}{N - \text{remainder}(N, d) + 1} \leq c/m < 1/d\) |
| 5 (novel) | \(\text{division}(n, d) = \text{division}(c \times n + c, m)\) and \(\text{remainder}(n, d) = \text{division}(\text{remainder}(c \times n + c, m) \times d, m)\) for all \(n \in [0, N]\) | \(1 - \frac{1}{N + 1} \leq c/m < 1/d\) |
| 6 (novel) | \(d\) divides \(n \in [0, N]\) if and only if \(\text{remainder}(c \times n + c, m) < c\) | \(1 - \frac{1}{N + 1} \leq c/m < 1/d\) |

Granlund and Montgomery’s approach [5] (without an intermediate addition) was refined by Cavagnino and Werbrouck [13], and later by Warren [11]. As remarked by Robison [7], the two approaches (with and without an intermediate addition) are complementary: we can choose one or the other depending on the divisor. We review and elaborate on this complementarity in §5.

To our knowledge, the latest work on the software acceleration of the division by constants was Lemire et al. [9]. They revisited two specific problems: the direct computation of the remainder—without first computing the quotient—and the related divisibility tests. Compared to optimizing compilers that compute the remainder by first computing the quotient, they found that their direct approach could be up to 30% faster. Their divisibility test could be twice as fast as the code produced by popular optimizing compilers and libraries. It can also be up to twice as fast as the state-of-the-art divisibility check proposed by Granlund and Montgomery [5]. They did not consider the multiply-add approach, a gap that we fill with §5. We also make their main result [9, Theorem 1] tighter (see Theorem 2). We similarly improve mathematically on their divisibility check [9, Proposition 1] (see Proposition 1). Our improvements may not immediately result in improved software performance, but they fill a conceptual gap. The systematic computation of the remainder directly as proposed by Lemire et al., without first computing the quotient, has received attention in the hardware
and circuit literature [14,17] but had never been generally exploited in software as far as we know. One practical exception was the work by Vowels [4] who described the direct computation of both the quotient and remainder, in the special case where we divide by 10.

3. Technical Preliminaries

For non-negative real numbers \( z \), \( \text{floor}(z) \) is the greatest integer no larger than \( z \). It is a monotonic function: if \( z_1 \geq z_2 \) then \( \text{floor}(z_1) \geq \text{floor}(z_2) \).

We define \( \text{division}(x, y) \equiv \text{floor}(x/y) \) and

\[
\text{remainder}(x, y) \equiv x - \text{division}(x, y) \times y \tag{1}
\]

\[
= x - \text{floor}(x/y) \times y \tag{2}
\]

for positive real numbers \( x, y \) with the constraint that \( y \neq 0 \). We have that \( \text{remainder}(x, y) \in [0, y) \). If \( y \) is an integer and \( x \) is not an integer, then \( \text{remainder}(x, y) \neq 0 \). By definition, we always have that \( x = \text{division}(x, y) \times y + \text{remainder}(x, y) \).

**Lemma 1.** Consider a positive integer \( d > 0 \), a non-negative integer \( n \) and a non-negative real number \( x \). We have that \( \text{remainder}(n, d) = \text{floor}(\text{remainder}(x, d)) \) and \( \text{division}(n, d) = \text{division}(x, d) \) if and only if \( n \leq x < n + 1 \).

**Proof.** \((\Leftarrow)\) We can verify that if \( n \leq x < n + 1 \), the previous two conditions are satisfied.

\((\Rightarrow)\) Assume that \( \text{remainder}(n, d) = \text{floor}(\text{remainder}(x, d)) \) and \( \text{division}(n, d) = \text{division}(x, d) \). We have

\[
\text{remainder}(n, d) \leq \text{remainder}(x, d) \tag{3}
\]

\[
< \text{remainder}(n, d) + 1. \tag{4}
\]

By expanding out \( \text{remainder}(n, d) = \text{floor}(\text{remainder}(x, d)) \), we have that

\[
n - \text{floor}(n/d) \times d \leq x - \text{floor}(x/d) \times d \tag{5}
\]

\[
< n - \text{floor}(n/d) \times d + 1. \tag{6}
\]

Expanding out \( \text{division}(n, d) = \text{division}(x, d) \) and multiplying by \( d \), we have \( \text{floor}(n/d) \times d = \text{floor}(x/d) \times d \). We establish the lemma by adding this last equation to the previous inequality. \(\square\)

4. Multiply-Divide Results

Given a non-negative numerator \( n \) and non-zero divisor \( d \), we want to show that by choosing integer constants \( c \) and \( m \) carefully, we can compute \( \text{division}(n, d) \) and \( \text{remainder}(n, d) \) by starting from \( c \times n \) and dividing by \( m \).
4.1. Quotient

We want to find \( c \) and \( m \) such that \( \text{division}(n, d) = \text{division}(c * n, m) \). Intuitively, this equation implies that \( n/d \approx c * n/m \) and \( n \approx c * n * d/m \). Let us formalize this intuition.

For any non-negative real number \( x \) and non-negative integer \( Q \), we have that \( \text{floor}(x/d) = Q \) is equivalent to \( x \in [Qd, Qd + d) \). Letting \( Q = \text{floor}(n/d) \) and \( x = c * n * d/m \), we get \( d * \text{floor}(n/d) \leq c * n * d/m < d * \text{floor}(n/d) + d \). Since \( d * \text{floor}(n/d) = n - \text{remainder}(n,d) \), \( \text{floor}(x/d) = \text{division}(c * n, m) \) and \( Q = \text{division}(n, d) \), we have that \( \text{division}(n, d) = \text{division}(c * n, m) \) is equivalent to \( n - \text{remainder}(n, d) \leq c * n * d/m < n - \text{remainder}(n, d) + d \).

Consider a range of integer numerators \( n \in [0, N] \) for some maximal integer numerator \( N \geq d \). We want this equation to hold for all \( n \). The equation is satisfied trivially when \( n = 0 \) and \( d > 0 \). Suppose that \( n > 0 \) and rewrite the inequalities as \( (n - \text{remainder}(n, d))/n \leq c * d/m < (n - \text{remainder}(n, d) + d)/n \).

Given any \( n \in [1, N] \) for some integer \( N \geq d \), we have that the leftmost expression \( (n - \text{remainder}(n, d))/n = 1 - \text{remainder}(n, d)/n \) is largest and equal to 1 when \( \text{remainder}(n, d) = 0 \). Meanwhile the rightmost expression \( 1 + (d - \text{remainder}(n, d))/n \) is smallest when \( n \) is as large as possible with \( \text{remainder}(n, d) = d - 1 \). To prove this bound, partition the possible numerators into sets \( N_k = \{n \in [0, N] \mid \text{remainder}(n, d) = k\} \). Fixing \( N \) and \( d \), we seek the value \( n \in [1, N] \) minimizing \( f(n) = 1 + (d - \text{remainder}(n, d))/n \). For \( n \in N_k \) we have \( f(n) = 1 + (d - k)/n \) which is minimized for the largest member of \( N_k \).

Let \( v \) be the largest member of \( N_{d-1} \); we have \( f(v) = 1 + 1/v \). We see that the values of \( n \) in \([N - d + 1, N] \) are the minimizing values in each \( N_k \). Among these, we can show \( v \) minimizes \( f \).

- Consider any \( n \in [N - d + 1, N] \) with \( n > v \). Write it as \( n = v + k \) with \( k > 0 \) and \( k \leq d - 1 \) and so \( \text{remainder}(n, d) = k - 1 \) and \( f(n) = 1 + (d - k + 1)/(v + k) \). As \( k \) increases, the numerator decreases and the denominator increases, we have that the minimum is reached when \( k \) is largest \((d - 1)\), in which case \( f(n) = 1 + 2/(v + d - 1) \geq 1 + 2/(v + v) = 1 + 1/v = f(v) \), since \( v \geq d - 1 \).

- Consider any \( n \in [N - d + 1, N] \) with \( n < v \). Write \( n \) as \( v - d + k \), so \( \text{remainder}(n, d) \) is again \( k - 1 \). We have \( f(n) = 1 + (d - k + 1)/(v - d + k) \). Again, as \( k \) increases, the numerator decreases and the denominator increases, we have that the minimum is reached when \( k \) is largest \((d - 1)\) in which case \( f(n) = 1 + 2/(v - 1) < 1 + 1/v = f(v) \).

Thus \( n = v \) minimizes \( f(n) \). We have shown Lemma 2 because \( v = N - \text{remainder}(N + 1, d) \).

**Lemma 2.** Given an integer \( d > 0 \), the value of \( 1 + (d - \text{remainder}(n, d))/n \) over \( n = 0, 1, \ldots, N \) is minimized when \( n \) is \( n = N - \text{remainder}(N + 1, d) \).

Hence we have that \( 1 \leq c * d/m < 1 + \frac{1}{N - \text{remainder}(N + 1, d)} \) is equivalent to \( \text{division}(n, d) = \text{division}(c * n, m) \) for all \( n \in [0, N] \).
Theorem 1. Consider an integer divisor $d > 0$ and a range of integer numerators $n \in [0, N]$ where $N \geq d$ is an integer. We have that $\text{division}(n, d) = \text{division}(c \ast n, m)$ for all integer numerators $n$ in the range if and only if

$$\frac{1}{d} \leq c/m < \left(1 + \frac{1}{N - \text{remainder}(N + 1, d)}\right) \frac{1}{d}. \quad (7)$$

Remark 1. Granlund and Montgomery \cite{GM} have an upper bound of $c/m \leq \left(1 + \frac{1}{N + 1}\right) \frac{1}{d}$ as a sufficient (but not necessary) condition. A bound equivalent to Theorem 1 is derived by Warren \cite{W5}.

Once we have a pair of inequalities as in Theorem 1 we can solve for $c$ and $m$. It is always possible to do so: we can verify that $c = 1$, $m = d$ is always a solution. However, we may have further constraints on $c$ and $m$: maybe we require $m$ to be a power of two. We can show that as long as we can choose $m$ arbitrarily large, there is always a solution. Letting $K = N - \text{remainder}(N + 1, d)$, we can rewrite the inequalities as $m/d \leq c < \left(1 + \frac{1}{K}\right) \frac{1}{m/d}$. Thus if $c$ is to be as small as possible, we must have that $c = \text{ceiling}(m/d)$. It remains to solve for $m$ such that $\text{ceiling}(m/d) - m/d < 1$, we have that the inequality is always satisfied when $m \geq K \ast d = (N - \text{remainder}(N + 1, d)) \ast d$. This bound indicates that it is always possible to find a solution, by picking $m$ large enough.

4.2. Remainder

From the quotient division($c \ast n, m$), we get the quotient of the division of $n$ by $d$; it is maybe intuitive that we can derive the remainder of the division of $n$ by $d$ from remainder($c \ast n, m$).

Formally, we want to find integer constants $c > 0$ and $m > 0$ such that for any integer numerator $n \in [0, N]$ and integer divisor $d > 0$, we have that $\text{remainder}(n, d) = \text{division}(\text{remainder}(c \ast n, m) \ast d, m)$.

If we find $c$ and $m$ such that $\text{remainder}(n, d) = \text{division}(\text{remainder}(c \ast n, m) \ast d, m)$ is satisfied, then replacing $c$ with $c + m$ or $c + 2m$ would still work: in fact $\text{remainder}(c \ast n, m) = \text{remainder}(c \ast n + k \ast m \ast n, m) = \text{remainder}((c + k \ast m) \ast n, m)$ for any integer $k$. Thus we require $c$ to be in $[0, m)$.

With this constraint ($c \in [0, m)$), we are able to show (see Lemma 3) that the ability to compute remainders via remainder($n, d) = \text{division}(\text{remainder}(c \ast n, m) \ast d, m)$ implies that the quotient of $n$ divided by $d$ is given by division($n, d) = \text{division}(c \ast n, m)$. Intuitively, it is strictly more difficult to compute the remainder than to compute the quotient. Hence, if we just need the remainder, and not the quotient, we cannot relax our conditions when $c \in [0, m)$.

Lemma 3. Consider an integer divisor $d > 1$. Suppose that we have integer constants $c$ and $m$ such that $c \in [0, m)$ and remainder($n, d) = \text{division}(\text{remainder}(c \ast n, m) \ast d, m)$ for all numerators $n \in [0, N]$ then we must have that division($n, d) = \text{division}(c \ast n, m)$.
Proof. When \( n = 0 \), we have that \( \text{division}(n, d) = \text{division}(c \cdot n, m) \) holds trivially. Since \( c \in [0, m) \) then \( c \cdot (n + 1) - c \cdot n < m \) so when \( \text{division}(c \cdot n, m) \) increases following an increment of \( n \) by one, it must increase by at most one. We just have to show that it happens exactly when \( \text{remainder}(n, d) = 0 \).

We have \( c \cdot n = \text{remainder}(c \cdot n, m) + m \cdot \text{division}(c \cdot n, m) \). The left side of this equation increases by \( c \) exactly when \( n \) is incremented by one. When \( \text{division}(c \cdot n, m) \) increases by one, then it contributes \( m \) to the right side. Since \( m > c \), we have that an increase of \( \text{division}(c \cdot n, m) \) corresponds to a decrease of \( \text{remainder}(c \cdot n, m) \).

However, we have that \( \text{remainder}(n, d) = \text{division}(\text{remainder}(c \cdot n, m) \cdot d, m) \). From this equation, we have that whenever \( \text{remainder}(n, d) \) increases when we increment \( n \) by one, then \( \text{remainder}(c \cdot n, m) \) must also increase. We know that when \( n \) is incremented, then either \( \text{remainder}(n, d) \) decreases: it must therefore be that a decrease in \( \text{remainder}(c \cdot n, m) \) corresponds to \( \text{remainder}(n, d) = 0 \). Thus we have that an increase of \( \text{division}(c \cdot n, m) \) following an increment of \( n \) corresponds remainder \( (n, d) = 0 \). It follows that \( \text{division}(n, d) = \text{division}(c \cdot n, m) \).

We still must derive the conditions on \( c \) and \( m \). We can expand the condition that \( \text{remainder}(n, d) = \text{division}(\text{remainder}(c \cdot n, m) \cdot d, m) \) as follows:

\[
\text{remainder}(n, d) = \text{floor}\left(\frac{(c \cdot n - \text{floor}\left(\frac{c \cdot n}{m}\right) \cdot m) \cdot d}{m}\right)
\]

\[
= \text{floor}\left(c \cdot n \cdot d/m - \text{floor}\left(\frac{c \cdot n \cdot d/m}{d}\right) \cdot d\right)
\]

\[
= \text{floor}(\text{remainder}(c \cdot n \cdot d/m, d)).
\]

Then by Lemma 1, we have that the two constraints \( \text{remainder}(n, d) = \text{floor}(\text{remainder}(c \cdot n \cdot d/m, d)) \) and \( \text{division}(n, d) = \text{division}(c \cdot n \cdot d/m, d) \) are equivalent to \( n \leq c \cdot n \cdot d/m < n + 1 \), or \( m/d \leq c < (1 + 1/n)m/d \). This condition should hold for all applicable values of \( n \), and thus we choose to use the maximal value of \( n \) (i.e., \( N \)) as it provides the tightest bound — so an equivalent expression is \( m/d \leq c < (1 + 1/N)m/d \).

We have derived the following theorem.

**Theorem 2.** Consider an integer divisor \( d > 0 \) and a range of integer numerators \( n \in [0, N] \) where \( N \geq d \) is an integer. We have that \( \text{division}(n, d) = \text{division}(c \cdot n, m) \) and \( \text{remainder}(n, d) = \text{division}(\text{remainder}(c \cdot n, m) \cdot d, m) \) for all integer numerators \( n \) in the range if and only if

\[
1/d \leq c/m < \left(1 + \frac{1}{N}\right)1/d.
\]

We can check that the conditions of Theorem 2 are always met with \( c = \text{ceiling}(m/d) \) and \( m \geq N \cdot d \).
Remark 2. In previous work [12, Theorem 1], Lemire et al. reported an upper bound of $c/m \leq (1+1/(N+1))1/d$ as a sufficient (but not necessary) condition. For the difference to matter, we need that there is an integer in the interval $((1+1/(N+1))m/d,(1+1/N)m/d)$. It happens in some instances, for example if $N = 10$, $d = 5$, and $m = 2^5$, we have that $7 \in ((1+1/(N+1))m/d,(1+1/N)m/d)$. However, in the previous work [9], Lemire et al. considered only the case where $m = N+1$. We can show that if $m = N+1$ and $m \geq 3$ then the earlier bound is tight. Indeed, if there is an integer $z$ in $((1+1/(N+1))m/d,(1+1/N)m/d)$ then there must be an integer $z \in [0,m]$ such that $z$ is a divisibility test. However, we show the more elegant result that remainder$(c*n, m) < c$ is a divisibility test (see Proposition 1).

By the assumption of Theorem 1, we have that $m/d \leq c$. Thus if $n$ is a multiple of $d$, then we have that remainder$(c*n, m) < c$. We need to prove the counterpart, that remainder$(c*n, m) < c$ implies that $n$ is a multiple of $d$. By Theorem 2, we have that division$(n, d) = division(c*n, m)$. Hence we have that division$(c*n, m)$ = division$(c*(n - remainder(n, d)), m)$ since $n$ and $n - remainder(n, d)$ have the same quotient with respect to $d$. When two values $z_1, z_2$ have the same quotient (division$(z_1, m) = division(z_2, m)$) then their difference must be captured by their remainders: $z_2 - z_1 = remainder(z_2, m) - remainder(z_1, m)$. In this case, taking $z_1 = c*n$ and $z_2 = c*(n - remainder(n, d))$, we have that their difference is $c*remainder(n, d)$. It follows that remainder$(c*n, m) - remainder(c*(n - remainder(n, d)), m) = c*remainder(n, d)$ and therefore $c*remainder(n, d) \leq remainder(c*n, m)$. Thus if remainder$(c*n, m) < c$, we have $c*remainder(n, d) < c$ which implies remainder$(n, d) = 0$.

**Proposition 1.** Consider an integer divisor $d > 0$. We have that $d$ divides $n \in [0, N]$ if and only if remainder$(c*n, m) < c$ subject to the condition that

$$1/d \leq c/m < \left(1 + \frac{1}{N}\right)1/d.$$  \hspace{1cm} (12)

Proposition 1 selects a value of $c$ in $[0, m)$ when $d > 1$. 

8
5. Multiply-Add-Divide Results

Some authors \cite{7, 12} have considered the case where we replace the division by a formula of the multiply-add form \( \text{floor}\left(\frac{c \ast n + b}{m}\right) \) for some \( b \). The benefit of the multiply-add approach is that it may allow one to pick a smaller value of \( c \), compared to the simpler form \( \text{floor}(c \ast n / m) \). The derivations are nearly identical as in \S 4, so we just give our results.

**Theorem 3.** Consider an integer divisor \( d > 0 \) and a range of integer numerators \( n \in [0, N] \) where \( N \geq d \) is an integer. We have that \( \text{division}(n, d) = \text{division}(c \ast n + c, m) \) for all integer numerators \( n \) in the range if and only if

\[
\left(1 - \frac{1}{N - \text{remainder}(N, d) + 1}\right) \frac{1}{d} \leq c/m < 1/d. \tag{13}
\]

**Remark 3.** Robison \cite{7} derived the sufficient condition \( (1 - 1/(N + 1))1/d \leq c/m < 1/d \). When \( \text{remainder}(N, d) \neq 0 \), Robison’s bound is suboptimal unlike Theorem 3. Drane et al. \cite{17} derive a similar result to ours for the case where \( d \) is odd.

**Theorem 4.** Consider an integer divisor \( d > 0 \) and a range of integer numerators \( n \in [0, N] \) where \( N \geq d \) is an integer. We have that \( \text{division}(n, d) = \text{division}(c \ast n + c, m) \) and \( \text{remainder}(n, d) = \text{division}(\text{remainder}(c \ast n + c, m) \ast d, m) \) for all integer numerators \( n \) in the range if and only if

\[
\left(1 - \frac{1}{N + 1}\right) \frac{1}{d} \leq c/m < 1/d. \tag{14}
\]

**Proposition 2.** Consider an integer divisor \( d > 0 \). We have that \( d \) divides \( n \in [0, N] \) if and only if \( \text{remainder}(c \ast n + c, m) < c \) subject to the condition that

\[
\left(1 - \frac{1}{N + 1}\right) \frac{1}{d} \leq c/m < 1/d. \tag{15}
\]

We can check that the conditions of Theorem 3 are met when \( c = \text{floor}(m/d) \) and \( m \geq d \ast (N - \text{remainder}(N, d) + 1) \) as long as \( m \) is not divisible by \( d \). Proposition 2 is satisfied with the more stringent inequality \( m \geq d \ast (N + 1) \).

6. Complementarity

When processing numerators and divisors in \([0, n]\), it may be most convenient if the constant \( c \) is also in \([0, n]\). In this respect, the multiply-shift and multiply-add-shift results are complementary as first shown by Robison \cite{7}. Suppose that we want to divide all integers \( n \in [0, N] \) by \( d \) in the case where \( N + 1 \) is a power of two. For example, we may have \( N = 2^{64} - 1 \). We want the constant \( m \) to be a power of two.

When \( d \) is a power of two, efficient division and remainder routines are available. The quotient requires a single binary shift while the remainder
requires selecting the low-weight bits with a mask. Thus suppose that the divisor \( d \) is not a power of two.

To satisfy the constraints of Theorems 1 and 2 we can pick \( c = \left\lceil \frac{m}{d} \right\rceil \) and \( m = 2^\left\lceil \log_2(d) \right\rceil (N + 1) \). Unfortunately, \( c = \left\lceil \frac{m}{d} \right\rceil \) is not in \([0, N]\) which may cause implementation issues. Indeed, if we want to do 64-bit arithmetic on hardware with 64-bit machine words, it is most convenient if all constants fit in 64-bit words. Thus we may try a smaller constant. The choice \( m = 2^\left\lfloor \log_2(d) \right\rfloor (N + 1) \) is convenient since \( c = \left\lceil \frac{m}{d} \right\rceil \) is then an integer in \([0, N]\). Unfortunately, it is not a valid choice for all divisors \( d \), as per the requirements of Theorems 1 and 2. For example, if \( N + 1 = 2^{32} \) and \( d = 19 \), picking \( m = 2^\left\lfloor \log_2(d) \right\rfloor (N + 1) = 2^{4+32} \), we get \( c = \left\lceil \frac{m}{d} \right\rceil = 3616814566. \) We have that \( N - \text{remainder}(N + 1, d) = 2^{32} - 6 \). We can verify that \( 3616814566 > \frac{2^{4+32}}{19} \approx 3616814565.89 \). Thus the conditions of Theorem 1 are not satisfied.

Thankfully, we can fall back on the multiply-add-shift results. Suppose that setting \( c = \left\lceil \frac{m}{d} \right\rceil \) and \( m = 2^\left\lfloor \log_2(d) \right\rfloor (N + 1) \) fails to satisfy the conditions of Theorem 1, then we have that \( \left\lceil \frac{m}{d} \right\rceil \geq \left(1 + \frac{1}{N - \text{remainder}(N + 1, d)}\right) \frac{m}{d}. \) (16)

It may be convenient to simplify this equation further. We can multiply both sides by \( d \). We have that \( d \cdot \left\lceil \frac{m}{d} \right\rceil = m + d - \text{remainder}(m, d) \) on the left-hand-side. On the right-hand-side, we have \( m \) plus some quantity that may not be integer, but we can safely apply the ceiling function since the left-hand-side is an integer. After subtracting \( m \) from both sides, we get

\[
d - \text{remainder}(m, d) \geq \left\lceil \frac{m}{N - \text{remainder}(N + 1, d)} \right\rceil. \quad (17)
\]

(E.g., with \( d = 19 \) and \( N + 1 = 2^{32} \), we get \( 18 \geq 17 \).) We want to show that the conditions of Theorem 3 are satisfied when keeping \( m = 2^\left\lfloor \log_2(d) \right\rfloor (N + 1) \) and setting \( c = \left\lfloor \frac{m}{d} \right\rfloor \). That is, if Theorem 1 fails us, we can use Theorem 3 so that it is always possible to pick \( c \in [0, N] \).

We have that \( 2^\left\lfloor \log_2(d) \right\rfloor > d/2 \) and hence \( m > d \cdot (N + 1)/2 \) and therefore \( d/m < 2/(N + 1) \). We assume that \( d \) is not a power of two. Since we assume \( N + 1 \) and hence \( m \) are powers of two, we have that \( \text{ceiling}(m/d) = \text{floor}(m/d) + 1 \) and thus from (16)

\[
\text{floor}(m/d) + 1 \geq \left(1 + \frac{1}{N - \text{remainder}(N + 1, d)}\right) \frac{m}{d}. \quad (18)
\]
We can divide by $m$ and subtract $1/m$ to get
\[
\text{floor}(m/d)/m \geq \left(1 + \frac{1}{N - \text{remainder}(N + 1, d)}\right) \frac{1}{d} - \frac{1}{m} \quad (19)
\]
\[
= \left(1 + \frac{1}{N - \text{remainder}(N + 1, d)} - \frac{d}{m}\right) \frac{1}{d} \quad (20)
\]
\[
\geq \left(1 + \frac{1}{N+1} - \frac{d}{m}\right) \frac{1}{d} \quad (21)
\]
\[
> \left(1 + \frac{1}{N+1} - \frac{2}{N+1}\right) \frac{1}{d} \quad (22)
\]
\[
= \left(1 - \frac{1}{N+1}\right) \frac{1}{d} \quad (23)
\]
\[
\geq \left(1 - \frac{1}{N - \text{remainder}(N + 1, d) + 1}\right) \frac{1}{d}. \quad (24)
\]

We have shown Proposition 3, which tells us that it is always possible to compute the quotient using $c \in [0, N]$ selecting the approach using Equation 17.

**Proposition 3.** Consider an integer divisor $d > 0$ that is not a power of two. Let $N$ be an integer such that $N + 1$ is a power of two, then we can compute the quotient of any integer $n \in [0, N]$ by $d$ using a constant $c \in [0, N]$ as follows. Let $m = 2^{\text{floor}(\log_2(d))} \cdot (N + 1)$.

- if $d - \text{remainder}(m, d) < \text{ceiling}(m/(N - \text{remainder}(N + 1, d)))$, we let $c = \text{ceiling}(m/d)$ and we have $\text{division}(n, d) = \text{division}(c \cdot n, m)$.

- Otherwise we let $c = \text{floor}(m/d)$ and we have $\text{division}(n, d) = \text{division}(c \cdot n + c, m)$.

The same complementarity exists between the novel theorems for the computation of the quotient and remainder: Theorems 2 and 4. Indeed, if we choose $m = 2^{\text{floor}(\log_2(d))} \cdot (N + 1)$ and $c = \text{ceiling}(m/d)$, but the conditions of Theorem 2 are not met, then we have that
\[
\text{ceiling}(m/d) \geq \left(1 + \frac{1}{N}\right) m/d. \quad (25)
\]

We have that $\text{ceiling}(m/d) = \text{floor}(m/d) + 1$ since $m$ is not divisible by $d$ and

\[\footnote{This result is not novel \cite{7}.} \]
thus

$$\text{floor}(m/d) \geq \left(1 + \frac{1}{N}\right) m/d - 1$$  \hfill (26)
$$= \left(1 + \frac{1}{N} - \frac{d}{m}\right) 1/d$$ \hfill (27)
$$> \left(1 + \frac{1}{N+1} - \frac{d}{m}\right) 1/d$$ \hfill (28)
$$> \left(1 + \frac{1}{N+1} - \frac{2}{N+1}\right) 1/d$$ \hfill (29)
$$= \left(1 - \frac{1}{N+1}\right) 1/d.$$ \hfill (30)

Thus, again, it is always possible to pick $c \in [0, N]$; if not with Theorem 2, then with Theorem 4, using the analog of Equation 17 to select the correct approach. We formalize the result with the novel Proposition 4.

**Proposition 4.** Consider an integer divisor $d > 0$ that is not a power of two. Let $N$ be an integer such that $N + 1$ is a power of two, then we can compute the quotient and the remainder of any integer $n \in [0, N]$ by $d$ using a constant $c \in [0, N]$ as follows. Let $m = 2^\text{floor}(\log_2(d)) \ast (N + 1)$.

- if $d - \text{remainder}(m, d) < \text{ceiling}(m/N)$, we let $c = \text{ceiling}(m/d)$ and we have $\text{division}(n, d) = \text{division}(c \ast n, m)$ and $\text{remainder}(n, d) = \text{division}(\text{remainder}(c \ast n, m) \ast d, m)$.
- Otherwise we let $c = \text{floor}(m/d)$ and we have $\text{division}(n, d) = \text{division}(c \ast n + c, m) \ast d, m)$ and $\text{remainder}(n, d) = \text{division}(\text{remainder}(c \ast n + c, m) \ast d, m)$.

**Ideal divisors.** When $N + 1$ is a power of two, we have shown that it is always possible to pick $m = 2^\text{floor}(\log_2(d)) \ast (N + 1)$. However, for some divisors, we can pick an even smaller $m$, even with the constraint that $m$ be a power of two. Suppose that you would like to compute both the remainder and the quotient, as in Theorem 2. Picking $m = N + 1$ would be especially convenient. (In architectures where the product is stored in a pair of registers, $\text{division}(x, m)$ and $\text{remainder}(x, m)$ are essentially free if $m = 2^W$, where $W$ is the architecture’s word size.) We choose $m = (N + 1)$ and $c = \text{ceiling}((N + 1)/d)$. To satisfy the conditions of Theorem 2, we need that $c \ast d < (1 + 1/N) \ast (N + 1) = N + 2 + 1/N$. We have that $\text{ceiling}((N + 1)/d) \ast d = (N + 1) + d - \text{remainder}(N + 1, d)$. Assuming that $d$ does not divide $N + 1$, the inequality holds if and only if $d - \text{remainder}(N + 1, d) = 1$ which is true if and only if $d$ divides $N + 2$. We refer to any such divisors as being *ideal*, as they enable us to pick $m = N + 1$.

See Table 2.\footnote{They are related to Fermat numbers [18].}
Table 2: Some ideal divisors

| Range $[0, N+1)$ | ideal divisor     |
|------------------|-------------------|
| $[0, 2^{32})$    | 641               |
| $[0, 2^{32})$    | 6700417           |
| $[0, 2^{64})$    | 274177            |
| $[0, 2^{64})$    | 67280421310721    |

We can show that we cannot pick $m$ to be a smaller power of two—unless $d$ divides $N+1$. Indeed, if we pick $m = (N + 1)/2$, then the conditions of Theorem 2 require that $c \cdot d < (1 + 1/N) * (N + 1)/2$ but $1/N * (N + 1)/2 < 1$ for $N > 1$ and since $c \cdot d$ is an integer, we then must have that $c \cdot d \leq (N + 1)/2$. Yet we have $\text{ceiling}((N + 1)/(2 \cdot d)) \cdot d > (N + 1)/d$ when $d$ does not divide $N + 1$.

7. Rounding

Instead of computing the integer division ($\text{floor}(n/d)$), we sometimes wish to round the result to the nearest integer. Unsurprisingly and maybe obviously, it is possible to do with an expression of the form $\text{floor}(z/d)$ for some integer $z$ that depends on $n$ and $d$. Hence, our efficient integer quotient computations extend to the computation of the rounded division. Indeed, we have that $\text{floor}((n + \text{floor}(d/2))/d)$ is the round-to-nearest function; when $d$ is odd and $n$ is between two multiple of $d$, it rounds up. To round down, we can use $\text{floor}((n + \text{ceiling}(d/2) - 1)/d)$ instead.

It is also possible to handle more complicated scenarios. For example, what if we wish to round to the nearest integer, rounding to the nearest even integer when we are in-between two integers? It is only relevant when the division $d$ is even. Let $z = n + \text{floor}(d/2)$. Whenever $z$ is a multiple of $d$ and $\text{floor}(z/d)$ is odd, we return floor($z/d) - 1$, else we return floor($z/d$). We can check that an integer is a multiple of $d$ efficiently with Proposition 4 for example. The division and the divisibility test can reuse the same intermediate computations.

8. Conclusion

Our work shows that a unified approach, with the same precomputed constants, allows the calculation of the quotient and remainder, while further providing fast divisibility checks. Thus, for example, an algorithm could check efficiently whether an integer is divisible by another and, in the negative case, compute the remainder while reusing the prior work.

Future work might address the problem of computing generalized expressions such as $\text{floor}(n \cdot x)$ for integer values $n$ and real numbers $x$. For example, we

---

We define $\text{ceiling}(x)$ as the smallest integer that is no smaller than $x$. 
have that \( \text{floor}(\log_2(5^x)) = \text{floor}(x \cdot \log_2(5)) \) can be computed as multiplication and a division by a power of two: \((152170 \cdot x) \div 2^{16}\) for \(x \in (-400, 350)\). We find such optimizations hand-coded in highly optimized algorithms \([19]\): it might prove useful to formalize the derivation of such routines.

References

[1] E. Artzy, J. A. Hinds, H. J. Saal, A fast division technique for constant divisors, Commun. ACM 19 (1976) 98–101.

[2] D. H. Jacobsohn, A combinatoric division algorithm for fixed-integer divisors, IEEE Trans. on Comput. 100 (1973) 608–610.

[3] S.-Y. R. Li, Fast constant division routines, IEEE Trans. Comput. 34 (1985) 866–869.

[4] R. A. Vowels, Division by 10, Aust. Comput. J. 24 (1992) 81–85.

[5] T. Granlund, P. L. Montgomery, Division by invariant integers using multiplication, SIGPLAN Not. 29 (1994) 61–72.

[6] H. S. Warren, Jr., Hacker’s Delight, Addison-Wesley, Boston, 1st edition, 2002.

[7] A. D. Robison, N-bit unsigned division via n-bit multiply-add, in: Proceedings of the 17th IEEE Symposium on Computer Arithmetic, ARITH ’05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 131–139.

[8] Anonymous author, Labor of division (episode iii): Faster unsigned division by constants, \(\text{http://ridiculousfish.com/blog/posts/labor-of-division-episode-iii.html}\) (last checked Nov. 2020), 2011.

[9] D. Lemire, O. Kaser, N. Kurz, Faster remainder by direct computation: Applications to compilers and software libraries, Software: Practice and Experience 49 (2019) 953–970.

[10] T. Drane, W.-c. Cheung, G. Constantinides, Correctly rounded constant integer division via multiply-add, in: 2012 IEEE International Symposium on Circuits and Systems, IEEE, pp. 1243–1246.

[11] H. S. Warren, Jr., Hacker’s Delight, Addison-Wesley, Boston, 2nd edition, 2013.

[12] D. J. Magenheimer, L. Peters, K. Pettis, D. Zuras, Integer Multiplication and Division on the HP Precision Architecture, SIGARCH Comput. Archit. News 15 (1987) 90–99.

[13] D. Cavagnino, A. E. Werbrouck, Efficient algorithms for integer division by constants using multiplication, Comput. J. 51 (2008) 470–480.
[14] P. S. Raghuram, F. E. Petry, Constant-division algorithms, IEE Proceedings - Computers and Digital Techniques 141 (1994) 334–340.

[15] R. W. Doran, Special cases of division, Journal of Universal Computer Science 1 (1995) 176–194.

[16] F. Ugurdag, F. D. Dinechin, Y. S. Gener, S. Gören, L.-S. Didier, Hardware division by small integer constants, IEEE Transactions on Computers 66 (2017) 2097–2110.

[17] F. de Dinechin, L.-S. Didier, Table-based division by small integer constants, in: Reconfigurable Computing: Architectures, Tools and Applications: 8th International Symposium, Springer, Berlin, Heidelberg, 2012, pp. 53–63.

[18] J. H. Jaroma, K. N. Reddy, Classical and alternative approaches to the Mersenne and Fermat numbers, The American Mathematical Monthly 114 (2007) 677–687.

[19] D. Lemire, Number parsing at a gigabyte per second, Software: Practice & Experience (2021). To appear.