On the modular inverse algorithm.*

Cassio Neri

31 August 2019†

Abstract

The modular inverse algorithm can be used to evaluate modular expressions of the forms `n % d == r` and `n % d == m % d`, provided that `d` is a compile time constant. In many situations it is faster than what is currently implemented by major compilers.

The algorithm is presented in [1] by means of an example and this document complements the presentation by providing the mathematical proofs of the algorithm's correctness.

1 Introduction

Generally, the expression `n % d == r` is evaluated by major compilers as `n - (n / d) * d == r`. (Recall that `/` is integer division). When `d` is a compile time constant (but not a power of two) the costly division is replaced by a multiplication and other cheaper operations. Therefore, two multiplications are required to evaluate the expression.

The modular inverse algorithm provides an alternative that performs only one multiplication. Although, this is not a new algorithm and has been covered by [3] for a couple of years, surprisingly, it has not yet been incorporated into compilers. (The development version of GCC, poised to be relased soon, implements an incomplete flavour of the algorithm.)

The lack of support for this algorithm was the motivation behind [1] which contains a detailed analysis showing the superior performance of the modular inverse algorithm with respect to the original method. Other important contributions of [1], compared to [3] are:

- It makes clear the method can be used for any compile time non-negative constant remainder `r < d` and not only for zero;
- It shows the algorithm also applies to variable remainders with improved performance under certain conditions;
- It also considers the expression `n % d == m % d`.

Although [1] presents the algorithm, it does so by means of an example because the article targeted an audience that is not necessarilly interested in the nitty-grittys. This document fills the gap and provides a complete mathematical proof of the algorithm's correctness.

†The latest version of this document is available in [2].

2 Definitions, conventions and basic results

This section presents definitions and conventions that will be used throughout. For easy of reference, they will be presented in **red bold** typeface at the point of definition. It also recalls some basic results.

Let $\boldsymbol{w} \in \mathbb{N}$ be the word size (the cases of practical interest are $w = 32$ and $w = 64$), $\boldsymbol{\Omega} := \{0, \ldots, 2^w - 1\}$ be the set of representable numbers and $\boldsymbol{d} \in \Omega$, $d \geq 1$, be the divisor. A ***d*-remainder** is any element of $\{0, \ldots, d-1\}$. Given a d-remainder r, let $\boldsymbol{N_r}$ be the number of elements in Ω that are equivalent to r modulo d.

For the sake of clarity we shall use $\oplus$ for addition modulo 2^w in Ω and $+$ for the usual addition in $\mathbb{Z}$. More precisely, given integers $a, b \in \Omega$, $a \oplus b$ is the unique $c \in \Omega$ such that $c \equiv a + b \pmod{2^w}$. Analogous conventions apply to subtraction ($\ominus$ and $-$) and multiplication ($\odot$ and $\cdot$). Arithmetic operations in Ω do overflow if, and only if, their results do not match those in $\mathbb{Z}$. More precisely, for $a \in \Omega$ and $b \in \Omega$ we have

$$a + b \in \Omega \Leftrightarrow a \oplus b = a + b; \qquad a - b \in \Omega \Leftrightarrow a \ominus b = a - b; \qquad \text{and} \qquad a \cdot b \in \Omega \Leftrightarrow a \odot b = a \cdot b. \quad (1)$$

The Fundamental Theorem of Arithmetic implies the existence of unique integers $\boldsymbol{h} \geq 1$ and $\boldsymbol{k} \geq 0$ such that $d = h \cdot 2^k$ and h is odd. Since $d \in \Omega$, we have $h, k \in \Omega$ and thus, $d = h \odot 2^k$.

Given $n \in \Omega$, the quotient and the remainder of the division of n by 2^k are, respectively, denoted by $\boldsymbol{n_H}$ and $\boldsymbol{n_L}$. These numbers are uniquely defined by the following relations $n = 2^k \cdot n_H + n_L$, $0 \leq n_H < 2^{w-k}$ and $0 \leq n_L < 2^k$.

For $n \in \Omega$ the rotation of n (by k bits) to the right is defined by $\boldsymbol{ror(n)} := 2^{w-k} \cdot n_L + n_H$. It is easy to see that ror is bijective and for multiples of 2^k it matches division by 2^k, that is, $ror(2^k \odot i) = ror(2^k \cdot i) = i$ provided that $0 \leq i < 2^{w-k}$.

Since h is odd, yet another classic arithmetic result, yields the existence of a unique $\boldsymbol{g} \in \Omega$ such that $g \odot h = 1$. (In general, $g \cdot h \neq 1$.) These numbers are said to be modular inverse of one another. Since $h \odot (g \odot n) = g \odot (h \odot n)) = n$ for all $n \in \Omega$, the function $n \mapsto g \odot n$ is bijective.

For any d-remainder r, the function $n \mapsto ror(g \odot (n \ominus r))$ is bijective as a composition of three bijective functions: $n \mapsto n \ominus r$, $n \mapsto g \odot n$ and ror.

3 The modular inverse algorithm

The modular inverse algorithm uses g and k (it uses ror which implicitly depends on k) which depend only on w and d. Efficient methods to find these quantities are well known and not covered here. (See [3].) The final quantity required by the algorithm is N_r which depends on w, d and r. Obtaining N_r is trivial as shown by the following proposition.

Proposition 1. *Let $\tilde{q}$ and $\tilde{r}$ be the quotient and remainder of the division of 2^w by d. For any d-remainder r, we have*

$$N_r = \tilde{q} + \begin{cases} 1, & \text{if } r < \tilde{r}, \\ 0, & \text{otherwise.} \end{cases}$$

Moreover, $N_r \leq 2^{w-k}$.

Proof. Set $n := 2^w - \tilde{r}$. Then, $\tilde{q} \cdot d = n$ and there exactly $\tilde{q}$ elements in $X := \{0, \ldots, n-1\}$ that are equivalent to r modulo d. Now, the elements of $\Omega \setminus X$ are $n, \ldots, n + (\tilde{r} - 1)$ which are, respectively, equivalent to $0, \ldots, \tilde{r} - 1$ modulo d. Therefore, either $N_r = \tilde{q} + 1$, if r is amongst these remainders, or $N_r = \tilde{q}$, otherwise.

We have shown that $N_r \leq \tilde{q} + 1$. Now, it is easy to see that $\tilde{q} \leq 2^{w-k}$ and equality holds if, and only if, $\tilde{r} = 0$, in which case, $N_r = \tilde{q}$ (since $r \geq 0 = \tilde{r}$). Therefore, $N_r \leq \tilde{q} + 1 \leq 2^{w-k} + 1$ but equality holds for at most one these two relations. We conclude that $N_r < 2^{w-k} + 1$, i.e., $N \leq 2^{w-k}$. ■

The following result is the proof of correctness of the modular inverse algorithm.

Theorem 2. *Let r be a d-remainder. Then, for any $n \in \Omega$, we have*

$$n \equiv r \ (\textit{mod}\ d) \quad \Longleftrightarrow \quad ror(g \odot (n \ominus r)) < N_r. \tag{2}$$

Proof. For any $n \in \Omega$, $n \equiv r \pmod{d}$ if, and only if, there exist $i \in \{0, \ldots, N_r - 1\}$ such that $n = d \odot i \oplus r$. In this case, we get

$$ror(g \odot (n \ominus r)) = ror(g \odot d \odot i) = ror(g \odot h \odot 2^k \odot i) = ror(2^k \odot i).$$

From $i < N_r$ and Proposition 1 we obtain $i < 2^{w-k}$ and then $ror(2^k \odot i) = i$. Therefore, the elements of Ω that are equivalent to r modulo d are mapped onto $\{0, \ldots, N_r - 1\}$. Reciprocally, if $n \not\equiv r \pmod{d}$ then, by the injectiveness of $n \mapsto ror(g \odot (n \ominus r))$, n must be mapped into $\Omega \setminus \{0, \ldots, N_r - 1\}$, that is, $ror(g \odot (n \ominus r)) \geq N_r$. ■

4 Extra optimisation

Obviously, when $r = 0$ the subtraction in equation (2) can be elided. Similarly and as explained in [1], each divisor admits a non-zero remainder for which the subtraction is also avoidable and the test takes the form $ror(g \odot n) \geq 2^w - N_r$. This simple optimisation is the subject of this section and require a few lemmas to be stablished.

Lemma 3. *Let $n \in \Omega$, $m \in \Omega$ and $i \in \mathbb{Z}$ be such that $n - m = 2^k \cdot i$. Then $n_L = m_L$ and $n_H = m_H + i$.*

Proof. From $n - m = (2^k \cdot n_H + n_L) - (2^k \cdot m_H + m_L) = 2^k \cdot (n_H - m_H) + (n_L - m_L) = 2^k \cdot i$ we conclude, first, that $n_L - m_L$ divides 2^k and since $0 \leq n_L < 2^k$ and $0 \leq m_L < 2^k$ we get $n_L = m_L$. Then, substituting $n_L - m_L = 0$ in the first identity yields $2^k \cdot (n_H - m_H) = 2^k \cdot i$, i.e., $n_H - m_H = i$. ■

Lemma 4. *Let $n \in \Omega$ and r be a d-remainder. Then, for all $i \in \{0, \ldots, N_r - 1\}$ we have $g \odot r + 2^k \cdot i < 2^w$ or, in other words, $g \odot r \oplus 2^k \cdot i = g \odot r + 2^k \cdot i$.*

Proof. Assume by contradiction the existence of $i \in \{0, \ldots, N_r - 1\}$ such that $g \odot r + 2^k \cdot i \geq 2^w$ and take the smallest i with this property. Surely $i > 0$ since, by definition, $g \odot r < 2^w$. Hence, $i - 1$ do not possess this property which means $g \odot r + 2^k \cdot (i - 1) < 2^w$. Therefore, if we set $s := g \odot r + 2^k \cdot i - 2^w$, then we get $0 \leq s < 2^k$ and $g \odot r + 2^k \cdot i - s = 2^w$. The last equality gives $g \odot r \oplus i \cdot 2^k \ominus s = 0$.

Let $n \in \Omega$ be defined by $n := r + d \cdot i$. Then, we have

$$g \odot n = g \odot r \oplus g \odot d \odot i = g \odot r \oplus g \odot h \odot (2^k \cdot i) = g \odot r \oplus 2^k \cdot i = s.$$

Multiplication by h gives $n = h \odot s$. Now, $0 \leq h \cdot s < h \cdot 2^k = d < 2^w$ implies $h \odot s = h \cdot s$, i.e., $n = h \cdot s < d$. We have proven that $n = r + d \cdot i < d$ but this is absurd since $r \geq 0$, $d \geq 0$ and $i \geq 1$. ■

Theorem 5. *Let r be a d-remainder and $i \in \{0, \ldots, N_r - 1\}$. Then, $ror(g \odot (r + i \cdot d)) = ror(g \odot r) + i$.*

Proof. Let $n \in \Omega$ be defined by $n := r + d \cdot i$. Then, $g \odot n = g \odot r \oplus g \odot d \odot i = g \odot r \oplus g \odot h \odot (2^k \cdot i) = g \odot r \oplus 2^k \cdot i$. Lemma 4 gives $g \odot n = g \odot r + 2^k \cdot i$, that is, $g \odot n - g \odot r = 2^k \cdot i$.

Now Lemma 3, applied to $g \odot n$ and $g \odot r$, gives $(g \odot n)_L = (g \odot r)_L$ and $(g \odot n)_H = (g \odot r)_H + i$. Therefore,

$$ror(g \odot n) = 2^{w-k} \cdot (g \odot n)_L + (g \odot n)_H = 2^{w-k} \cdot (g \odot r)_L + (g \odot r)_H + 1 = ror(g \odot r) + i.$$

which is the desired result. ■

We finally reach the result that proves the correctness of the simplified test:

Proposition 6. *Let s be the d-remainder of $2^w - h$. Then, for any $n \in \Omega$ we have $n \equiv s \pmod d$ if, and only if,*

$$ror(g \odot s) \leq ror(g \odot n).$$

Proof. The previous theorem states, in other words, that for any given d-remainder r, the elements of Ω which are equivalent to $r \pmod d$ are mapped by $n \mapsto ror(g \odot n)$ into the interval $[ror(g \odot r), ror(g \odot r) + N_r)$. Hence, $n \in \Omega$ is equivalent to $r \pmod d$ if, and only if,

$$ror(g \odot r) \leq ror(g \odot n) < ror(g \odot r) + N_r. \tag{3}$$

Set $m := 2^w - h$. By definition, s is m's d-remainder and, moreover, m is the largest element of Ω which is equivalent to $s \pmod d$ since $m + d \geq m + h = 2^w \notin \Omega$. Hence, $m = s + d \cdot (N_s - 1)$ and the previous theorem again gives $ror(g \odot m) = ror(g \odot s) + N_s - 1$.

On the other hand, $g \odot m = g \odot (2^w - h) = g \odot (\ominus h) = \ominus 1 = 2^w - 1$. By noticing that

$$ror(2^w - 1) = ror(2^k \cdot (2^{w-k} - 1) + (2^k - 1)) = 2^{w-k} \cdot (2^k - 1) + (2^{w-k} - 1) = 2^w - 1,$$

we conclude $ror(g \odot m) = 2^w - 1$ which, together with previous paragraph, yields $ror(g \odot s) + N_s = 2^w$. Therefore, when $r = s$, any $n \in \Omega$ verifies the second inequality in (3). We conclude that testing whether n is equivalent to $s \pmod d$ is the same as verifying the first inequality in (3) with $r = s$. ■

References

[1] Neri, C., *Quick Modular Calculations – Part I*, to appear in Overload.

[2] Neri, C., http://github.com/cassioneri/qmodular

[3] Warren, H.S., *Hacker's delight*. Addison-Wesley, 2013.