Montgomery Multiplication

Duncan A. Buell

October 11, 2005

Abstract

We describe Montgomery multiplication.

1 Montgomery Multiplication

Peter Montgomery has devised a way to speed up arithmetic in a context in which a single modulus is used for a long-running computation [Mon85]. This method has also been explored as a hardware operation [BD97, EW93].

The basic idea goes back to a standard trick that has been used for arithmetic modulo Mersenne numbers.

Let $M_n = 2^n - 1$ be the n-th Mersenne number. Assume that we are doing arithmetic modulo M_n. The crucial operation is multiplication: if A and B are integers modulo M_n, that is to say, n-bit numbers, then the product $C = A \cdot B$ can be written as $C = C_1 \cdot 2^n + C_0$; C_1 and C_0 are the digits of the product C written with radix 2^n.

The trick is to observe the following.

$$\begin{aligned} C &= C_1 \cdot 2^n + C_0 \\ &= C_1 \cdot 2^n - C_1 + C_1 + C_0 \\ &= C_1 \cdot (2^n - 1) + C_1 + C_0 \\ &= C_1 \cdot M_n + C_1 + C_0 \\ &\equiv C_1 + C_0 \pmod{M_n} \end{aligned}$$

So instead of having to divide by M_n in order to produce the remainder, we only need to add the left half of a product to the right half of the product.

For example, let's do the arithmetic modulo $base^n - 1$ for $base = 10$. Specifically, let's do arithmetic this way modulo $99 = 10^2 - 1$. Take $53 \cdot 77 = 4081$, say. This is

$$\begin{aligned} 4081 &= 40 \cdot 100 + 81 \\ &= 40 \cdot 100 - 40 + 40 + 81 \\ &= 40 \cdot 99 + 40 + 81 \\ &\equiv 121 \pmod{99}. \end{aligned}$$

In this case, we happen to get a sum larger than the modulus, and we have to subtract 99 from 121 to get the final result of 22. But one addition and possibly one subtraction is a major advantage over a full multiprecise divide.

So now let's see how to do this for an arbitrary integer and not just for $base^n - 1$.

Assume we're going to do a lot of arithmetic modulo some fixed N. Choose $R = 2^k > N$ for a suitable k. Assuming that R and N are relatively prime (and if not, bump k by one and we should be able to get an R that is relatively prime), then we can solve for R' and N' such that $RR' - NN' = 1$.

What we will do is multiply everything by R. All the constants, all the numbers, etc. So instead of doing arithmetic with integers a and b, say, we will be doing arithmetic with integers aR and bR. At the very end of the computation, we multiply any result by R'. Since $RR' \equiv 1 \pmod{N}$, we recover the result we would have Addition and subtraction are fine, since

$$a + b = c \Leftrightarrow aR + bR = cR.$$

The problem is with multiplication:

$$aR \cdot bR = abR^2$$

which means that we have an extra factor of R. What we want to do is have a function to which we can pass the product abR^2 and that will return abR. We could do this by multiplying modulo N by R', but that would be a multiplication modulo N, and it's exactly that that we are trying to avoid.

Here's how we do it. Start with $T = abR^2$.

$$\begin{aligned} m &\leftarrow (T \pmod{R}) \cdot N' \pmod{R} \\ t &\leftarrow (T + mN)/R \end{aligned}$$

and we return either t or $t - N$, whichever lies in the range 0 to $N - 1$.

Example: Let $N = 79$, and instead of using a power of 2 for R, we'll use $R = 100$ for readability. We find that $64 \cdot 100 - 81 \cdot 79 = 1$, so we have $R = 100$, $R' = 64$, $N = 79$, $N' = 81$.

Now let's say that we multiply $a = 17$ times $b = 26$ to get 442. The number 17 is really $a' \cdot 100$ modulo 79 for some a'. Multiplying $17 \cdot 64 \equiv 61 \pmod{79}$, we find that $a' = 61$. Similarly, $26 \cdot 64 \equiv 5 \pmod{79}$. So when we multiply 17 and 26 in this representation, we're really trying to multiply $61 \cdot 5 = 305 \equiv 68 \pmod{79}$.

Knowing that we can in fact work modulo 79, we know that what we have is

$$\begin{aligned} 17 \cdot 26 = 442 &\equiv (61 \cdot 100) \cdot (5 \cdot 100) \\ &\equiv 305 \cdot 100 \cdot 100 \\ &\equiv 68 \cdot 100 \cdot 100 \pmod{79} \end{aligned}$$

and if we multiply by 64 and reduce modulo 79 we should get the right answer:

$$442 \cdot 64 \equiv 28288 \equiv 6 \equiv 68 \cdot 100 \pmod{79}.$$

The function we want is the function that will take as input the 442 and return 6. And the function described above does exactly that:

$$\begin{aligned} m &= (442 \pmod{100}) \cdot 81 \pmod{100} \\ &= 42 \cdot 81 \pmod{100} \\ &= 3402 \pmod{100} \\ &\equiv 2 \pmod{100} \\ t &= (442 + 2 \cdot 79)/100 \\ &= (442 + 158)/100 \\ &= 600/100 \\ &= 6 \end{aligned}$$

and we return $t = 6$ as the result.

Proof that the algorithm works: We assume that value T is a product, and hence is double length. Since we choose $R > N$ but not too much bigger, the products can be taken to be double length in R.

The first modular reduction simply converts T to a single length number modulo R. Again modulo R, we have that $m = TN'$. Thus

$$mN \equiv TN'N \equiv -T \pmod{R}.$$

So when we take $T + mN$ we get an integer that is zero modulo R and we can legitimately divide out the R and get an integer quotient for t.

Now the fact that we get the right quotient comes from the fact that

$$tR = T + mN \equiv T \pmod{N}$$

so that modulo N we have $t \equiv TR'$.

References

[BD97] Jean-Claude Bajard and Laurent-Stéphane Dider. An RNS Montgomery modular multiplication algorithm. *Proceedings, IEEE Symposium on Computer Arithmetic*, pages 234–239, 1997.

[EW93] Stephen E. Eldridge and Colin D. Walter. Hardware implementation of Montgomery's modular multiplication algorithm. *IEEE Transactions on Computers*, 42:693–699, 1993.

[Mon85] Peter L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.