Single Precision Barrett Reduction

Jacob Wells

June 28, 2014

Abstract

NOTE: I am not a professional mathematician nor am I a professional programmer, so please do not hold this paper to the same standards you would them. If you find any errors, please feel free to contact me at jacob.f.wells@gmail.com

1 Introduction

Modular Multiplication is a heavily researched area of Mathematics and Computer Science. It's use in Public Key Encryption has been a driving force for faster multiplication and modular reduction algorithms. Here I will show a way to reduce a 2N-bit by N-bit integer using a precomputed N-bit reciprocal, and two N-bit multiplications. This can be used in both word size arithmetic, as well as arbitrary-precision arithmetic.

2 Background

I arrived at this idea while trying to speed up Modular Multiplication in Assembly Language. There are already several well known ways for speeding up Division, the first of which being multiplication by a precomputed Reciprocal, followed by a Bit Shift. Unfortunately this only works if both the Dividend and Divisor fit in a computer word. In my case, I was multiplying two N-Bit integers and getting a 2N-Bit Product. There is also Barrett Reduction[1], which is designed to work in with 2N-Bit answers. Here, you calculate a 2N-Bit Reciprocal, and Multiply it by your 2N-Bit product. Unfortunately this technique is better suited for BigInteger algorithms: while it can be done in word size arithmetic, it is almost always slower than the naive multiply and divide.

The last commonly technique is to use Montgomery Reduction[3]. Unfortunately this requires the two multipliers to first be converted into Montgomery Form, apply this reduction technique, then convert them back, which is expensive if you don't use the same numbers multiple times. Also, this technique is slower if the Modulus is an even number[2], in which case you must perform additional multiplications.

I arrived at this idea after I observed that a trial quotient could be generated using only the Significant Bits of the product we wish to reduce. In this case, Significant Bits refers to all the bits higher than the most significant bit of the modulus, similar to Barrett Multiplication. The downsize to this technique is that there is a higher error in the trial Quotient. With Barrett Multiplication, the Quotient is usually exact, or 1 less than the true quotient. With this technique, the Quotient can be off by at most 4. In practice however, the Quotient is usually off by 1 or 2, sometimes 3, and rarely 4. Compared to Barrett Reduction, this allows us to trade off the extra multiplications for at most 3 additional subtractions.

3 Technique

Let M, the Modulus, be a positive integer. Let A, the Dividend to be reduced, be a positive integer such that $(M-1)^2 \geq A$. Let X be a positive integer such that $2^X \geq M > 2^{X-1}$.

First we must calulate the N-Bit Reciprocal.

$$R = \lfloor \frac{2^{2X-1}}{M} \rfloor$$

Next we need to get the Significant Bits from the Dividend.

$$H = \lfloor \frac{A}{2^x} \rfloor$$

And we get our Trial Quotient Q_T by Dividing RH by 2^{X-1}.

$$Q_T = \frac{RH}{2^{X-1}}$$

Of course, as we are using integer arithmetic, this can done using a Bit Shift.

$$Q_T = F >> (X-1)$$

4 Examples

For our examples we will use $M = 121$ and $A = 100 * 111 = 11100$. Our Exponent X will be 7, as $2^7 = 128$ and $128 \geq 121 > 64$.

Our Reciprocal R will be

$$R = \lfloor \frac{2^{2*7-1}}{121} \rfloor = \lfloor \frac{8192}{121} \rfloor = 67$$

And our most Significant Bits are

$$H = \lfloor \frac{11100}{128} \rfloor = 86$$

Now we can calculating our Trial Quotient Q_T

$$Q_T = \lfloor \frac{86 * 67}{2^{X-1}} \rfloor = \lfloor \frac{5762}{64} \rfloor = 90$$

Using this we can calculate our Remainder Estimate

$$R_T = 11100 - (121 * 90) = 210$$

$210 > 121$, so we subtract 121 and get 89. And indeed, $11100-(121*91) = 89$

5 Proof

This technique is based on the identity

$$\frac{A}{M} = \frac{\frac{A}{2^X}\frac{2^{2X-1}}{M}}{2^{X-1}}$$

What I will prove is

$$\frac{A}{M} = \lfloor\frac{\lfloor\frac{A}{2^X}\rfloor\lfloor\frac{2^{2X-1}}{M}\rfloor}{2^{X-1}}\rfloor + E_T$$

Which I will do by showing that the Error Term E_T is

$$0 \leq E_T < 5$$

So first off, we need to know some conditions for R and H.

$$R = \lfloor\frac{2^{2X-1}}{M}\rfloor, R < 2M$$

$$H = \lfloor\frac{A}{2^x}\rfloor, H < M$$

So using the floor identity, we can show that

$$\frac{A}{2^X} = \lfloor\frac{A}{2^X}\rfloor + E_1$$

$$0 \leq E_1 < 1$$

$$\frac{2^{2X-1}}{M} = \lfloor\frac{2^{2X-1}}{M}\rfloor + E_2$$

$$0 \leq E_2 < 1$$

Now we can show that

$$\frac{A}{2^X}\frac{2^{2X-1}}{M} = \lfloor\frac{A}{2^X}\rfloor\lfloor\frac{2^{2X-1}}{M}\rfloor + P_1 + P_2 + P_3$$

$$P_1 = E_2\lfloor\frac{A}{2^X}\rfloor$$

$$P_2 = E_1\lfloor\frac{2^{2X-1}}{M}\rfloor$$

$$P_3 = E_1E_2$$

Using our already established restrictions, we know that

$$0 \leq P_1 < M$$

$$0 \leq P_2 < 2M$$

$$0 \leq P_3 < 1$$

P_1, P_2, and P_3 constitute our current error term, so we can simplify our equation by stating that

$$\frac{A}{2^X}\frac{2^{2X-1}}{M} = \lfloor\frac{A}{2^X}\rfloor\lfloor\frac{2^{2X-1}}{M}\rfloor + E_3$$

$$E_3 = P1 + P2 + P3$$

$$0 \leq E_3 < 3M + 1$$

And for the final part, we can show that

$$\frac{A}{M} = \frac{\lfloor\frac{A}{2^X}\rfloor\lfloor\frac{2^{2X-1}}{M}\rfloor}{2^{X-1}} + \frac{E_3}{2^{X-1}}$$

Again, using the floor identity we can show that

$$\frac{\lfloor\frac{A}{2^X}\rfloor\lfloor\frac{2^{2X-1}}{M}\rfloor}{2^{X-1}} = \lfloor\frac{\lfloor\frac{A}{2^X}\rfloor * \lfloor\frac{2^{2X-1}}{M}\rfloor}{2^{X-1}}\rfloor + E_4$$

$$0 \leq E_4 < 1$$

And for the second part of the Error Term

$$\frac{E_3}{2^{X-1}} < \frac{3M+1}{2^{X-1}} < 4$$

Now can define the Total Error Term to be

$$E_T = E_4 + E_5$$

$$0 \leq E_T < 5$$

And so we can show that

$$\frac{A}{M} = \lfloor\frac{\lfloor\frac{A}{2^X}\rfloor\lfloor\frac{2^{2X-1}}{M}\rfloor}{2^{X-1}}\rfloor + E_T$$

$$0 \leq E_T < 5$$

6 Conclusion

This technique provides an excellent way of speeding up Modular Multiplications. While not quite as fast as Montgomery Reduction for odd moduli, it is faster for even moduli, as well as situations where additional operations would need to be performed that required the numbers to be converted back and forth from the Montgomery Domain.

References

[1] Paul Barrett. Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In *Advances in cryptology—CRYPTO'86*, pages 311–323. Springer, 1987.

[2] ÇK Koç. Montgomery reduction with even modulus. *IEE Proceedings-Computers and Digital Techniques*, 141(5):314–316, 1994.

[3] Peter L Montgomery. Modular multiplication without trial division. *Mathematics of computation*, 44(170):519–521, 1985.