Practical Integer Division with Karatsuba Complexity *

Tudor Jebelean

Research Institute for Symbolic Computation
A4232 Schloß Hagenberg, Austria

`tudor@risc.uni-linz.ac.at`
`http://www.risc.uni-linz.ac.at`

Abstract

Combining Karatsuba multiplication with a technique developed by Krandick for computing the high-order part of the quotient, we obtain an integer division algorithm which is only two times slower, on average, than Karatsuba multiplication. The main idea is to delay part of the dividend update until this can be done by multiplication between large balanced operands. An implementation under `saclib` is faster than classical multiplication at 40 words, and becomes two times faster at 250 words.

Introduction

The Karatsuba method for long integer multiplication [4] is probably the only asymptotically fast algorithm of practical use for integer arithmetic. Depending on the implementation, the break-even point against the classical algorithm is typically between 5 and 50 words.

However, integer division with remainder does not benefit from this algorithm. Indeed, although theoretically division has the same time complexity as multiplication (see e.g. [5], p. 275), a division algorithm designed along the lines as explained by Knuth will be about 30 times slower than multiplication (see analysis in [6]). By making use of the Krandick-Johnson multiplication [8, 7] which computes only the high-order digits, and of a squaring routine which is twice as fast as general multiplication, one may hope to reduce the gap to 15 times. If Karatsuba multiplication is used, then the break-even point against classical division will be above $n = 1000$ words. (If the Karatsuba threshold is t, then n can be obtained from the equation $n^2 = 15cn^{\log 3/\log 2}$, where c is found from $t^2 = ct^{\log 3/\log 2}$. For instance, if $t = 10$, then $n \approx 7000$.) We present a technique which combines Krandick's division algorithm for computing the high-order part of the quotient [6] with Karatsuba multiplication, leading to a division algorithm which is about two times slower than Karatsuba multiplication. The main idea of the method is to delay the update of part of the dividend until this can be done by multiplication of large balanced operands. The update can be delayed because Krandick's algorithm needs only 3 word-updates below the current position in the dividend.

The new algorithm does not have Karatsuba complexity in the traditional sense, because in the worse case Krandick's algorithm fails to produce the right quotient digits, which means one will have to resort to classical division – having again quadratic time complexity.

However, the probability of failure is very small (quotient-length divided by $2^{\text{word}-\text{length}}$), hence the average running time is practically not affected. Experiments using the `saclib` computer algebra system [1] reveal that the new algorithm becomes faster than the classical method at about 40 words, and twice as fast at 250 words (these values may vary under different implementations).

*Supported by Austrian Forschungsförderungsfonds (FWF), project P10002-PHY.

1 The Algorithm

We consider the division with remainder of positive integers: given a dividend A and a divisor B, find the positive quotient Q and remainder R such that $A = BQ + R$ and $R < Q$.

The classical division algorithm (see [5] p. 237) can be seen as a series of successive updates of the dividend A by subtracting the divisor B multiplied by the current digit of the quotient Q. (This quotient digit is computed before each update using the 3 most significant digits of the current A and the 2 most significant digits of B.) Each update is right–shifted one position w.r.t. the previous one. This process is represented pictorially by the parallelogram in Fig. 1. The final value of A will be the remainder R.

Let us split the quotient Q into its high-order part Q_H of length q_H and its low-order part Q_L of length q_L such that $q_L \leq q_H \leq q_L + 1$, and let us also split the divisor B into its high order part B_H of length b_H and its low-order part B_L such that $q_H + 3 \leq b_H$ (if B is too short to allow this then one only splits the quotient until the above becomes possible).

In the classical division algorithm, the computation of the high-order part Q_H of the quotient requires the updates corresponding to the upper half of the parallelogram in Fig. 1 (areas 1 and 2). However, Krandick [6] proves that in most cases it is enough to update only 3 words below the lowest digit of A needed for the lowest quotient digit to be computed (i.e. down to the vertical line crossing area 1). Doing so, the probability that a quotient digit will be wrong is less than $q_H/2^w$, where w is the bit-length of the word (in our implementation $w = 29$). Moreover, such a

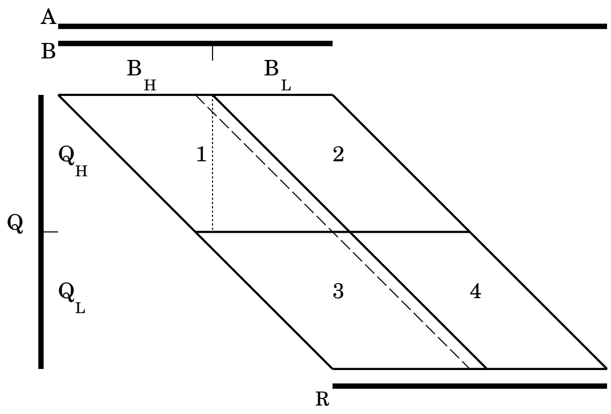

Figure 1: Organization of the dividend updates.

failure is easy to detect by inspecting the most significant digit of the updated value of A at each step, and by testing a supplementary condition after the computation of Q_H (see [6]).

We make use of Krandick's argument in asserting that the digits of Q_H will be computed correctly (with at least the same high probability) if we perform only the updates in area 1. In order to detect possible failures, we keep the check on the most significant updated digit of A at each step. However we do not need to use Krandick's condition after Q_H is computed. Instead of this, we check the condition $R < B$ after the whole division is finished. As it will be seen in the sequel, the overall algorithm insures that the resulting Q, R satisfy $A = BQ + R$, hence the former condition proves the correctness of the result.

Now let us split the updates of the dividend A into 4 parts as shown in Fig. 1 by the plain lines, (the dotted line represents 3 words into areas 1 and 3) and let us perform them in the order indicated by the numbers:

- Part 1 is split again into 4 parts by the same technique, and its effect will be to compute a Q'_H and to update A to:
$$A_1 = A - B_H Q'_H \beta^{bL+qL},$$
where β is the radix (in our implementation $\beta = 2^{29}$).
- Part 2 is performed by Karatsuba multiplication and updates A_1 to:
$$A_2 = A_1 - B_L Q'_H \beta^{qL}.$$
- Part 3 is split again into 4 parts recursively and its effect is to compute a Q'_L and to update:
$$A_3 = A_2 - B_H Q'_L \beta^{bL}.$$
- Part 4 is performed by Karatsuba multiplication and performs the update:
$$A_4 = A_3 - B_L Q'_L.$$

When b_H is below a constant threshold h , then the parts 1 and 3 are not split anymore, but the computation of the corresponding quotient digits and the updates are performed using the modified Krandick algorithm as explained above, which requires that 3 word-updates are done below the digits which are used for the computation of the next quotient digit. (This is insured by the condition $q_L \leq q_H \leq b_H - 3$.) The constant threshold h has to be such that, after splitting, the new b_L is above the Karatsuba threshold t, hence $h = 2t + 3$. (In our experiments, the Karatsuba threshold is 15, hence is $h = 33$.)

Obviously: $A_4 = A - Q'B$, where $Q' = Q'_H \beta^{qL} + Q'_H$. Furthermore, if $0 \leq A_4 < B$, then $Q' = Q$ and $A_4 = R$ due to the uniqueness of the integer quotient and remainder.

2 Complexity Analysis and Experiments

Let us assume the Karatsuba threshold is t and let us consider $n = 2^k t$ for some positive integer k. Then the number of digit products when multiplying two n-word numbers by Karatsuba algorithm is:

$$M(n) = M(2^k t) = 3M(2^{k-1} t) = \ldots = 3^k M(t) = 3^k t^2.$$

Let us compute the number $D(n)$ of digit products required by the new division algorithm in order to perform the updates when the length of the quotient Q is n and the length of the divisor B is $n + 3$:

$$D(n) = D(2^k t) = 2D(2^{k-1} t) + 2M(2^{k-1} t)$$

and by induction:

$$D(n) = 2^k D(t) + \sum_{i=1}^{k} 2^i M(2^{k-i} t),$$

hence, using $M(n) = 3^k t^2$ and $D(t) = t(t+3)$:

$$D(n) = 2M(t)\frac{3^k - 2^k}{3 - 2} + 2^k D(t) = 2M(n) - n(t - 3).$$

This allows us to conclude that division by the new algorithm is roughly two times as expensive as multiplication of the (balanced length) quotient and divisor by the Karatsuba algorithm.

This assertion is supported by the empirical facts: as shown in Table 1, at divisor lengths between 40 and 100 words, the ratio division-time per multiplication-time varies between 1.68 and 1.90. For longer operands, the ratio keeps similar values.

We implemented the new algorithm under the computer algebra system `saclib` [1], using also a modified version of the implementation of Krandick high-order division from [6]. The timings presented in Table 1 are obtained with **gnu** optimizing C compiler on a Sequent Symmetry architecture.

The speed-up of the new algorithm over the classical one starts to be visible at 40 words (the threshold is at 33 words), and at 250 words the new algorithm is 2 times faster (see Fig.2).

Conclusions and Further Work

Combining Karatsuba multiplication with high-order division brings the asymptotically fast algorithm into the reach of practical application. Although the theoretical worse-case complexity remains quadratic, this has practically no influence over the average time complexity, which is (almost) Karatsuba-like. For instance, if the length of the computer-word is 2^{29}, then incidence of divisions needing quadratic time is under 10^{-5} for operands up to 1000 words.

Certainly this technique can be applied to the right-to-left exact division algorithm of [3], where it is even easier

Dividend / divisor length	Absolute timings and *ratios*				
	New division	Karatsuba multiplication		Classical division	
80/40	59	35	*1.68*	64	*.92*
100/50	87	53	*1.64*	97	*.89*
120/60	118	62	*1.90*	140	*.84*
140/70	147	84	*1.75*	189	*.77*
160/80	188	108	*1.74*	245	*.76*
180/90	233	133	*1.75*	308	*.75*
200/100	285	161	*1.77*	381	*.74*
300/150	521	286	*1.82*	847	*.61*
400/200	887	491	*1.80*	1495	*.59*
500/250	1191	629	*1.89*	2331	*.51*
600/300	1618	868	*1.86*	3349	*.48*
700/350	2143	1152	*1.86*	4553	*.47*
800/400	2745	1469	*1.86*	5942	*.46*

Table 1: Timings in milliseconds.

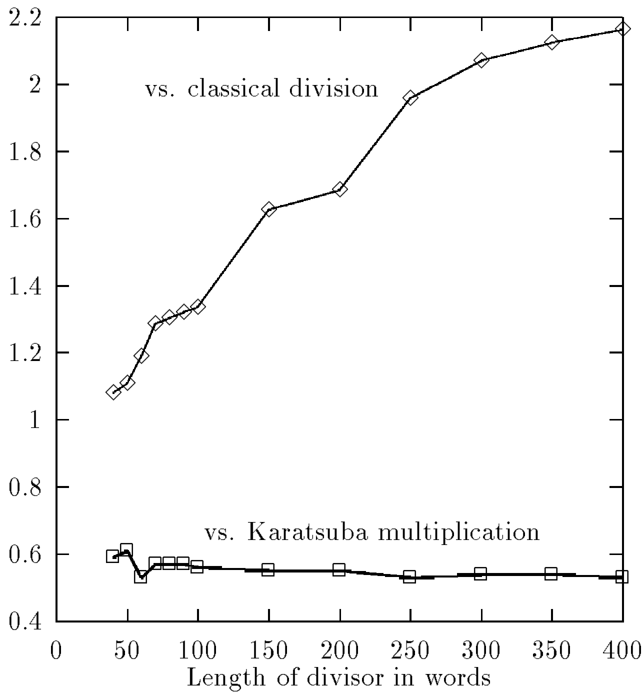

Figure 2: Speedup of the new algorithm over the classical division and the Karatsuba multiplication.

since the carries propagate in the opposite direction. Moreover, the technique is applicable to the bidirectional division of [6], which will probably make exact division about 4 times faster than the hereby presented algorithm.

A similar strategy of delaying operations until Karatsuba multiplication can be used should be investigated for GCD computation, especially for the recent Jebelean-Weber algorithm [2, 9] which is performed least-significant digits first.

And, finally, by using the parallel version of Karatsuba multiplication, the algorithms above may get an additional speed-up on a parallel architecture.

Acknowledgements. I express special thanks to Werner Krandick for the excellent work in designing, implementing, and describing the high-order division, which constitutes the starting point for this work.

References

[1] Buchberger, B., Collins, G. E., Encarnacion, M. J., Hong, H., Johnson, J. R., Krandick, W., Loos, R., Mandache, A. M., Neubacher, A., and Vielhaber, H. SACLIB 1.1 User's Guide. Tech. Rep. 93–19, RISC–Linz, 1993.

[2] Jebelean, T. A generalization of the binary GCD algorithm. In *ISSAC'93: International Symposium on Symbolic and Algebraic Computation* (Kiev, Ukraine, July 1993), M. Bronstein, Ed., ACM Press, pp. 111–116.

[3] Jebelean, T. An algorithm for exact division. *Journal of Symbolic Computation 15*, 2 (February 1993), 169–180.

[4] Karatsuba, A., and Ofman, Y. Multiplication of multidigit numbers on automata. *Sov. Phys. Dokl. 7* (1962), 595–596.

[5] Knuth, D. E. *The art of computer programming*, 2 ed., vol. 2. Addison-Wesley, 1981.

[6] Krandick, W., and Jebelean, T. Bidirectional exact integer division. *Journal of Symbolic Computation 21* (1996), 441–455.

[7] Krandick, W., and Johnson, J. R. Efficient multiprecision floating point multiplication with exact rounding. Tech. Rep. 93-76, RISC-Linz, RISC-Linz, Johannes Kepler University, A-4040 Linz, Austria, 1993. presented at the Rhine Workshop on Computer Algebra, Karlsruhe, Germany, 1994.

[8] Krandick, W., and Johnson, J. R. Efficient multiprecision floating point multiplication with optimal directional rounding. In *Proceedings of the 11th IEEE Symposium on Computer Arithmetic* (P.O.Box 3041, Los Alamitos, CA 90720-126, Phone: 714 821-838, 1993), E. Swartzlander, Jr., M. J. Irwin, and G. Jullien, Eds., IEEE, IEEE Computer Society Press, pp. 228–233.

[9] Weber, K. The accelerated integer GCD algorithm. *ACM Trans. on Math. Software 21*, 1 (March 1995), 111–122.