5,341,321

Paul Zimmermann
INRIA Lorraine/LORIA

8 June 2005

5,341,321: this is the number of a U.S. patent for a "floating-point arithmetic unit using modified Newton-Raphson technique for division and square root", published in 1994. Although apparently written with only hardware applications in mind[1], it is not clear at all that the claims of that patent do not apply to software. Indeed, 8 the 9 claims mention a "floating point arithmetic unit", which does not completely rule out a software implementation.

The algorithm behind this patent is due to Karp and Markstein [2]. Let us explain it briefly. The classical way to perform a division of two n-bit floating-point numbers B and A is first to compute a n-bit approximation, say C, of $1/A$ using Newton-Raphson iteration:

$$x_{k+1} = x_k + x_k(1 - Ax_k),$$

and then to compute $B \times C$. If we assume that a multiplication of two n-bit number costs $M(n) = n^\alpha$, the last iteration — to go from x_k with $n/2$ bits of accuracy to x_{k+1} with n bits of accuracy — costs $2(n/2)^\alpha$ to compute Ax_k, plus $(n/2)^\alpha$ to compute $x_k(1 - Ax_k)$ (we need only consider the n most significant bits of Ax_k, and we know that $n/2$ bits cancel with 1), thus a total of $3(n/2)^\alpha$. The total cost to compute $1/A$ to precision n is thus $3(n/2)^\alpha + 3(n/4)^\alpha + 3(n/8)^\alpha + \cdots = 3M(n)\frac{1}{2^\alpha - 1}$. Together with the cost of the multiplication $B \times C$, the total cost to compute B/A is $M(n)[\frac{3}{2^\alpha - 1} + 1]$.

Karp-Markstein's idea consists in incorporating the dividend B in the last step of Newton's iteration. Concretely, one computes an approximation x_k of $1/A$ accurate to $n/2$ bits, and one replaces the last iteration by $y_k = Bx_k$ (with $n/2$ bits of working precision), and

$$y_{k+1} = y_k + x_k(B - Ay_k).$$

The total cost is exactly that of Newton's iteration for the inverse, plus the cost of the multiplication $y_k = Bx_k$ to $n/2$ bits, which costs $M(n/2)$. Thus the cost to compute B/A is $M(n)[\frac{3}{2^\alpha - 1} + \frac{1}{2^\alpha}]$. The following table gives the cost of the classical approach — as a factor of the multiplication cost $M(n)$ — and that using Karp-Markstein's trick, for different values of the multiplication exponent α.

product	α	classical	K/M	gain
naive	2	2	1.25	38%
Karatsuba	1.585	2.5	1.833	27%
Toom-Cook	1.465	2.704	2.066	24%
FFT	1	4	3.5	13%

It is clear from this table that the complexity gain using Karp-Markstein's idea is quite important, especially for large α, i.e. for

[1] The abstract says "The invention computes approximated results faster and simplifies *hardware* requirements ...

small precision n. In addition, there is also a gain in terms of memory, since the largest multiplications performed when using Karp-Markstein's idea are of size $n/2$, assuming one splits the product Ay_k in two $n/2 \times n/2$ products.

A similar trick exists for compute $A^{1/2}$, multiplying by A the last iteration of Newton's iteration for $A^{-1/2}$.

If one wants to implement floating-point division in a free software like the GNU MP library [1][2], one is faced to the following dilemma:

- either implement classical division, which is suboptimal (see the above table);
- or use Karp-Markstein's trick, taking the risk to publicly distribute code whose use violates the patent.

Clearly the second option is not available for someone working in a public institute like me, since I would thereby involve my employeer in a potential trial with the patent assignee, namely the Hewlett-Packard company.

This example shows the nasty effects of patents for the development of free software, and the dissemination of scientific ideas, even for patents that were not written with software applications in mind!

References

[1] Granlund, T. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 4.1.4 ed., 2004. `http://www.swox.se/gmp/`.

[2] Karp, A. H., and Markstein, P. High-precision division and square root. *ACM Trans. Math. Softw. 23*, 4 (Dec. 1997), 561–589.

[2] The floating-point division in GMP is currently based on the integer division, i.e. the division with remainder, which is more expensive.