The Euclidean Division Implemented with a Floating-Point Multiplication and a Floor[*]

Vincent Lefèvre

11th July 2005

Abstract

This paper is a complement of the research report *The Euclidean division implemented with a floating-point division and a floor*, RR-5604, INRIA, June 2005. Here we study conditions under which the floor of an exact product (or the Euclidean division, when the divisor can be known in advance) can be implemented using a floating-point multiplication followed by a floor function. The example of the Euclidean division by 3 is given.

1 Introduction

A similar problem to the one dealt with in [3] is whether $\lfloor \diamond(x.z) \rfloor = \lfloor x.z \rfloor$. It is not clear whether this is interesting in practice; but in the division problem, if y is a constant, then performing a multiplication by a precomputed approximate value z of $1/y$ would be faster than performing a division by y. This is not exactly the same problem, though. Indeed, even if the above equality is satisfied, this does not imply that $\lfloor \diamond(x.z) \rfloor = \lfloor x/y \rfloor$ would be satisfied too, as z is only an approximation to $1/y$. However y may also be only an approximation to some real value, such as 2π; such Euclidean divisions occur in the range reduction to implement the trigonometric functions.

2 The Floor of a Floating-Point Product

First, let us analyze the problem of whether the equality $\lfloor \diamond(x.z) \rfloor = \lfloor x.z \rfloor$ is always satisfied or not, where x and z are both representable floating-point numbers. We assume that the arithmetic has no intermediate extended precision, and to make the discussion simpler, we just consider the rounding problems,

[*]This preprint comes from a section that was removed from [3]. Please refer to it for the context and the notations.

i.e. we ignore the possible limitations due to the exponent range. For symmetry reasons, we also assume that x and z are nonnegative numbers.

The exact result of the multiplication $x.z$ is denoted u, and we assume that $u < 2^n$ (i.e. a constraint similar to $x/y \leq 2^n$ in [3, Section 3]).

In the rounding downward mode, we always have $\lfloor \diamond(x.z) \rfloor = \lfloor x.z \rfloor$ for the same reasons as with the division (see [3, Section 3.1]).

Now, let us consider the rounding to nearest mode. The rounding rule in case of tie can be important here, so let us talk about it. In general (e.g., with the IEEE-754 standard), one rounds to the number that has an even mantissa; this choice is always unique for precisions larger or equal to 2. But one may also choose to round away from 0; however, in general, this rule is chosen in a different context: for instance, the `round` function to round to the nearest integer. This rule is still interesting to consider since this is the worst choice in our context and also because it is more regular. Note that $\lfloor \diamond(u) \rfloor$ will give the same result with both rounding rules if $u \leq 2^{n-1}$. Indeed, if u is exactly the middle of two consecutive representable numbers, the inequality $\lfloor \diamond(u) \rfloor \neq \lfloor u \rfloor$ can occur only when u has the following form:

$$u_1 u_2 \ldots u_r . \underbrace{111 \ldots 111}_{n+1-r \text{ bits}} \ldots$$

with $0 \leq r \leq n$. If $r < n$, then the n-th bit in u is always 1, therefore the even mantissa is obtained when rounding upward. Thus a difference between these two rules may occur only when $r = n$, i.e. $u > 2^{n-1}$; these cases are not common. So, for the sake of simplicity, we just consider the rule "round away from 0" in halfway cases.

For any $u < 1/2$, the equality $\lfloor \diamond(u) \rfloor = \lfloor u \rfloor$ is satisfied. So, in the following, let us assume that $1/2 \leq u < 2^n$ and let f_u be the number of fractional bits when the mantissa of u is truncated to the precision of the system, i.e. $f_u = n - 1 - \lfloor \log_2(u) \rfloor$. We also define the position of the bits in the exact mantissa of u as follows: The most significant bit (always equal to 1) has the position 0, the following bit has the position 1, and so on, up to the position $2n - 1$, since the exact mantissa of u can be written on at most $2n$ bits.

Let m be the number of 1's immediately following the fractional point of u. If $m \leq f_u$, then the equality $\lfloor \diamond(u) \rfloor = \lfloor u \rfloor$ is satisfied. If $m > f_u$, then this equality is not satisfied.

Moreover when x or z is multiplied by a power of two, the result $u = x.z$ has the same exact mantissa. So, to build a failing case, x and z can be adjusted to put the fractional point where we want (but not farther than the first n bits, so that $u < 2^n$). For instance, if the bit at position n is a 1, then we can scale x or z in such a way that $\lfloor \diamond(u) \rfloor \neq \lfloor u \rfloor$. But on such an example, u is very large: $u \geq 2^{n-1}$. However, with a stronger condition on u (i.e., on x and z), we can get smaller cases: for instance, if the bits r, $r + 1$, …, $n - 1$, n are all equal to 1, then we have a failing case such that $2^{r-1} \leq u < 2^r$.

We have an obvious worst case: $x = 1 + 2^{1-n}$ and $z = 1 - 2^{1-n}$, leading to $u = 1-2^{2-2n}$. And we can even restrict the precision on x and z: for $x = 1+2^{-q}$ and $z = 1 - 2^{-q}$, we have $u = 1 - 2^{-2q}$; and if $q > n/2$, then $1 - 2^{-n-1} \leq u < 1$, and $\diamond(u) = 1$. Hence $\lfloor \diamond(u) \rfloor \neq \lfloor u \rfloor$.

If z is fixed, then one can try to build failing cases by choosing the largest representable number $x_k < k/z$, for $k = 1$, 2, 3, etc. The analysis in the following sections is based on a similar idea.

3 Implementing a Division with a Floating-Point Multiplication

We now assume that y is a positive real constant (not necessarily representable, like 2π) and x is any floating-point number such that $0 \leq x/y < 2^n$, and we wish to obtain $\lfloor x/y \rfloor$ by approximating $1/y$ by some precomputed floating-point number $z = 1/y + \varepsilon$ and computing $\lfloor \diamond(x.z) \rfloor$ in some rounding mode $\diamond$. This is possible if and only if for all x, we have: $\lfloor \diamond(x.z) \rfloor = \lfloor x/y \rfloor$.

When y is a floating-point number, a related problem is whether we have $\diamond(x.z) = \diamond(x/y)$ for all x. In the rounding to nearest mode, there would be very few floating-point numbers y satisfying this condition, according to [1]. So, this will not help us in the following.

Since the multiplication by the constant z, the chosen rounding function $\diamond$ and the floor function are all non-decreasing functions, we can deduce that the function f defined by $f(x) = \lfloor \diamond(x.z) \rfloor$ is also non-decreasing. Thus it is sufficient to consider it on the boundary values $x_k = \triangle(k.y)$ and x_k^- for any integer k from 1 to 2^n, where $\triangle$ denotes the rounding upward mode and x^- the largest floating-point value less than x.

Indeed, x being a floating-point number, we have:

$$x/y \geq k \quad \Leftrightarrow \quad x \geq k.y \quad \Leftrightarrow \quad x \geq x_k$$

from the definition of $\triangle$, and

$$x/y < k \quad \Leftrightarrow \quad x \leq x_k^-.$$

Therefore

$$\lfloor x/y \rfloor = k \quad \Leftrightarrow \quad k \leq x/y < k+1 \quad \Leftrightarrow \quad x_k \leq x \leq x_{k+1}^-.$$

Since f is non-decreasing, we have:

$$\forall\, x \in [x_k, x_{k+1}^-],\ f(x) = k \quad \Leftrightarrow \quad f(x_k) \geq k \text{ and } f(x_{k+1}^-) \leq k.$$

Now let us consider good candidates for z. If $1/y$ is exactly representable, we can choose $z = 1/y$; this case has already been dealt with in Section 2. So, let us assume that $1/y$ is not exactly representable and let us denote the two floating-point numbers enclosing $1/y$ by $z^- = \nabla(1/y)$ and $z^+ = \Delta(1/y)$.

If y is exactly representable and we choose z^-, then we can rule out the rounding downward mode, as $x_1.z^- = y.z^- < y/y = 1$. Tests of some integers y using the MPFR library [2] show that the results depend very much on y (in fact, the binary expansion of $1/y$) and the precision. The next section just gives an example: $y = 3$. Future work could consist in finding general properties.

4 An Example: $y = 3$

In this section, we consider the simple case $y = 3$. For each rounding mode $\diamond$, we are interested in knowing under what conditions we have $\lfloor \diamond(x.z) \rfloor = \lfloor x/3 \rfloor$, where the floating-point number z is either $z^- = \nabla(1/3)$ or $z^+ = \Delta(1/3)$.

Thanks to the tests with MPFR, we could guess some properties. These properties depend on the parity of the precision of the floating-point system; this is not surprising as the binary expansion of 1/3, 0.010101..., has a periodic sequence of length 2. So, we need to consider odd precisions (4.1) and even precisions (4.2) separately. The precision 2 has some different properties, and as it is neither common nor useful, we will not consider it here. To do a comparison with the division (4.4), we will also need to find the smallest positive value of x such that $\lfloor \diamond(x/3) \rfloor \neq \lfloor x/3 \rfloor$ (4.3).

In the proofs below, we will need the following lemma.

Lemma 1 *If u is a positive floating-point number in a system with a mantissa size of n bits and u^- denotes its predecessor (with an unbounded exponent range), then $u^- = u(1 - \varepsilon)$, where ε satisfies $2^{-n} \leq \varepsilon < 2^{1-n}$.*

4.1 Odd Precisions for $n \geq 3$

We have: $z^- = (1 - 2^{-n-1})/3$ and $z^+ = (1 + 2^{-n})/3$. We seek to prove the following guessed properties:

1. In the rounding upward mode, $\diamond(x_5^-.z^-) \geq 5$. Therefore we also have $\diamond(x_5^-.z^+) \geq 5$, and this rounding mode is not interesting.
2. In the rounding to nearest mode, $\diamond(x_1^-.z^+) \geq 1$.
3. In the rounding to nearest mode, $\diamond(x_k^-.z^-) < k$ and $\diamond(x_k.z^-) \geq k$ for all integer $k \leq 2^n$.
4. In the rounding downward mode, $\diamond(x_k^-.z^+) < k$ and $\diamond(x_k.z^+) \geq k$ for all integer $k \leq (2^n - 2)/3$, but $\diamond(x_k^-.z^+) = k$ for $k = (2^n + 1)/3$.

Proof of Property 1 for odd precisions. We want to prove that $\diamond(x_5^-.z^-) \geq 5$ in the rounding upward mode.

For $n \geq 5$, 15 is exactly representable, therefore $x_5 = 15$ and $x_5^- = 15 - 2^{4-n}$. If $n = 3$, then 15 is not representable, and $x_5^- = \triangledown(15) = 14 > 15 - 2^{4-n}$. Therefore, for any odd integer n, we have:

$$x_5^-.z^- \geq (15 - 2^{4-n})(1 - 2^{-n-1})/3 = 5 - \varepsilon$$

with:

$$\begin{aligned} \varepsilon &= \frac{1}{3}(2^{4-n} + 15 \times 2^{-n-1} - 2^{3-2n}) \\ &= \frac{1}{3}(47 \times 2^{-n-1} - 2^{3-2n}) \\ &< 16 \times 2^{-n-1} = 2^{3-n}. \end{aligned}$$

It follows that $x_5^-.z^- > 5 - 2^{3-n} = 5^-$. Hence $\diamond(x_5^-.z^-) \geq 5$. □

Proof of Property 2 for odd precisions. We want to prove that $\diamond(x_1^-.z^+) \geq 1$ in the rounding to nearest mode.

As 3 is exactly representable, we have:

$$x_1^-.z^+ = (3 - 2^{2-n})(1 + 2^{-n})/3 = 1 - \varepsilon, \quad \text{with} \quad \varepsilon = \frac{2^{-n}(1 + 2^{2-n})}{3}.$$

Since $n \geq 3$, we have: $2^{2-n} \leq 1/2$, and $\varepsilon \leq 2^{-n-1}$. As a consequence, $x_1^-.z^+ \geq (1 + 1^-)/2$, and $\diamond(x_1^-.z^+) \geq 1$ since in Section 2, we chose to round away from zero in case of tie (but we recall that the rounding-to-even-mantissa rule would have given here the same result). □

Proof of Property 3 for odd precisions. We want to prove that in the rounding to nearest mode, $\diamond(x_k^-.z^-) < k$ and $\diamond(x_k.z^-) \geq k$ for all integer $k \leq 2^n$.

From the lemma in a system with $n + 1$ bits, we have:

$$k(1 - 2^{-n}) \leq \frac{k + k^-}{2} \leq k(1 - 2^{-n-1}).$$

First let us consider $x_k.z^-$. We have $x_k \geq 3k$, thus $x_k.z^- \geq k(1 - 2^{-n-1})$. Due to the above inequality, it follows that $x_k.z^- \geq (k + k^-)/2$. Hence $\diamond(x_k.z^-) \geq k$.

Now let us consider $x_k^-.z^-$. If $3k$ is exactly representable, then, from the lemma, $x_k^- \leq 3k(1 - 2^{-n})$; therefore

$$x_k^-.z^- < x_k^-/3 \leq k(1 - 2^{-n}) \leq (k + k^-)/2,$$

i.e. $x_k^-.z^- < (k+k^-)/2$. Now assume that $3k$ is not exactly representable. Then $3k \geq 2^n+1$, $x_k^- \leq 3k-1$ and

$$x_k^-.z^- \leq \frac{(3k-1)(1-2^{-n-1})}{3} = k - \frac{1}{3} - k.2^{-n-1} + \frac{2^{-n-1}}{3}.$$

Since $3k \geq 2^n+1$, we have: $k.2^{-n-1} \geq 1/6 + 2^{-n-1}/3$, where the equality occurs if and only if $3k = 2^n+1$. Therefore $x_k^-.z^- \leq k-1/2$, where the equality occurs if and only if $3k = 2^n+1$. If $(2^n+1)/3 \leq k < 2^{n-1}$, then $x_k^-.z^- < k-1/4 = (k+k^-)/2$. Otherwise $2^{n-1} < k < 2^n$ and $x_k^-.z^- < k-1/2 = (k+k^-)/2$. □

Proof of Property 4 for odd precisions. We want to prove that in the rounding downward mode, $\diamond(x_k^-.z^+) < k$ and $\diamond(x_k.z^+) \geq k$ for all integer $k \leq (2^n-2)/3$, but $\diamond(x_k^-.z^+) = k$ for $k = (2^n+1)/3$.

First let us assume that $k \leq (2^n-2)/3$; this means that $3k$ is exactly representable. Concerning the second inequality, since $x_k = 3k$ and $z^+ > 1/3$, we have $x_k.z^+ \geq k$, which is exactly representable, therefore $\diamond(x_k.z^+) \geq k$. Concerning the first inequality, we have $x_k^- \leq 3k(1-2^{-n})$ from the lemma, and $x_k^-.z^+ \leq k(1-2^{-2n})$. Since $x_k^-.z^+ < k$, which is exactly representable, we have $\diamond(x_k^-.z^+) < k$.

Now assume that $k = (2^n+1)/3$. Then $x_k^- = 2^n$ and $x_k^-.z^+ = (2^n+1)/3 = k$, which is exactly representable. Therefore $\diamond(x_k^-.z^+) = k$. □

We proved that for $0 \leq x \leq 2^n-2$, we have $\lfloor \diamond(x.z^+) \rfloor = \lfloor x/3 \rfloor$, and for $x = 2^n$, this equality is no longer satisfied. We can prove a little more. Indeed, there is a representable floating-point number between 2^n-2 and 2^n: 2^n-1. If $x = 2^n-1$, then $x.z^+ = (2^n-2^{-n})/3$. Therefore

$$\frac{2^n-2}{3} < \diamond(x.z^+) < \frac{2^n+1}{3} \quad \text{and} \quad \lfloor \diamond(x.z^+) \rfloor = \frac{2^n-2}{3} = \lfloor x/3 \rfloor.$$

So, for $0 \leq x \leq 2^n-1$, we have $\lfloor \diamond(x.z^+) \rfloor = \lfloor x/3 \rfloor$.

4.2 Even Precisions for $n \geq 4$

We have: $z^- = (1-2^{-n})/3$ and $z^+ = (1+2^{-n-1})/3$. We seek to prove the following guessed properties:

1. In the rounding upward mode, $\diamond(x_1.z^-) < 1$ and $\diamond(x_1^-.z^+) = 1$. Therefore this rounding mode is not interesting.

2. In the rounding to nearest mode, $\diamond(x_1.z^-) < 1$ and $\diamond(x_5^-.z^+) = 5$. Therefore this rounding mode is not interesting.

3. In the rounding downward mode, $\diamond(x_k^-.z^+) < k$ and $\diamond(x_k.z^+) \geq k$ for all integer $k \leq (2^{n+1}-2)/3$, but $\diamond(x_k^-.z^+) \geq k$ for $k = (2^{n+1}+1)/3$.

Proof of Property 1 for even precisions. We want to prove that $\diamond(x_1.z^-) < 1$ and $\diamond(x_1^-.z^+) = 1$ in the rounding upward mode.

We have $x_1.z^- = 1 - 2^{-n}$, which is exactly representable. As a consequence, $\diamond(x_1.z^-) = 1 - 2^{-n} < 1$.

$x_1^-.z^+ = (3 - 2^{2-n})(1 + 2^{-n-1})/3 = 1 + \frac{1}{3}(3 \times 2^{-n-1} - 8 \times 2^{-n-1} - 2^{1-2n})$, and since $n > 2$, we have $2^{1-2n} < 2^{-n-1}$. It follows that $x_1^-.z^+ > 1 - 2^{-n} = 1^-$, and $\diamond(x_1^-.z^+) = 1$. □

Proof of Property 2 for even precisions. From Property 1, we already know that $\diamond(x_1.z^-) < 1$ in the rounding to nearest mode. Therefore we just need to prove that $\diamond(x_5^-.z^+) = 5$.

We have $n \geq 4$, therefore 15 is exactly representable, $x_5^- = 15 - 2^{4-n}$, and $x_5^-.z^+ = (15 - 2^{4-n})(1 + 2^{-n-1})/3 = 5 - \varepsilon$, with:

$$\begin{aligned} \varepsilon &= \frac{1}{3}(2^{4-n} - 15 \times 2^{-n-1} + 2^{3-2n}) \\ &\leq \frac{1}{3}(17 \times 2^{-n-1} + 2^{-n-1}) \\ &< 8 \times 2^{-n-1} = 2^{2-n}. \end{aligned}$$

It follows that $x_5^-.z^+ > 5 - 2^{2-n} = (5 + 5^-)/2$. Hence $\diamond(x_5^-.z^+) = 5$. □

Proof of Property 3 for even precisions. We want to prove that in the rounding downward mode, $\diamond(x_k^-.z^+) < k$ and $\diamond(x_k.z^+) \geq k$ for all integer $k \leq (2^{n+1} - 2)/3$, but $\diamond(x_k^-.z^+) \geq k$ for $k = (2^{n+1} + 1)/3$.

First let us assume that $k \leq (2^{n+1} - 2)/3$. Concerning the second inequality, since $x_k \geq 3k$ and $z^+ \geq 1/3$, we have $x_k.z^+ \geq k$, which is exactly representable; hence $\diamond(x_k.z^+) \geq k$. Concerning the first inequality, if $3k$ is exactly representable, then $x_k^- \leq 3k(1 - 2^{-n})$ from the lemma, otherwise $3k$ is an $n+1$-bit odd integer and $x_k^- = 3k - 1 = 3k(1 - 1/(3k)) < 3k(1 - 2^{-n-1})$ since $3k < 2^{n+1}$; so, in both cases, $x_k^- < 3k(1 - 2^{-n-1})$. It follows that:

$$x_k^-.z^+ < k(1 - 2^{-n-1})(1 + 2^{-n-1}) = k(1 - 2^{-2n-2}) < k.$$

Hence $\diamond(x_k^-.z^+) < k$.

If $k = (2^{n+1} + 1)/3$, then $x_k^- = 2^{n+1}$ and $x_k^-.z^+ = (2^{n+1} + 1)/3 = k$, which is exactly representable. Therefore $\diamond(x_k^-.z^+) = k$. □

4.3 A Weaker Condition for the Division by 3

We already know that $\lfloor \diamond(x/3) \rfloor = \lfloor x/3 \rfloor$ in the rounding downward mode [3, Section 3.1]. But in the rounding to nearest mode, we just know that this equality is valid under the condition that $x - y$ is exactly representable [3, Theorem 1], i.e., in our particular case $y = 3$, $x \leq 2^n + 2$. But we can find a weaker condition for this case. Let us prove that, in the rounding to nearest mode, $\diamond(x_k^-/3) < k$ and $\diamond(x_k/3) \geq k$ for all integer $k \leq 2^{n-1}$, but $\diamond(x_k^-/3) = k$ for $k = 2^{n-1} + 1$.

First, $x_k \geq 3k$. Therefore $x_k/3 \geq k$, which is exactly representable. Hence $\diamond(x_k/3) \geq k$.

If $3k < 2^n$, then we know that $\lfloor \diamond(x_k^-/3) \rfloor = \lfloor x_k^-/3 \rfloor$, since $x_k^- - 3$ is exactly representable. Now, let us assume that $3k > 2^n$ and $k \leq 2^{n-1}$. Then $x_k^- \leq 3k-1$, and $x_k^-/3 \leq k - 1/3 < k - 1/4 = (k+k^-)/2$. Hence $\diamond(x_k^-/3) < k$.

If $k = 2^{n-1}+1$, then $x_k^- = 3 \times 2^{n-1}+2$, and $x_k^-/3 = 2^{n-1}+2/3 > 2^{n-1}+1/2$. It follows that $\diamond(x_k^-/3) = k$.

4.4 Conclusion Concerning $\lfloor x/3 \rfloor$

We have analyzed three possible implementations for $\lfloor x/3 \rfloor$ in the various rounding modes: $\lfloor \diamond(x/3) \rfloor$, $\lfloor \diamond(x.z^-) \rfloor$ and $\lfloor \diamond(x.z^+) \rfloor$, where $z^- = \nabla(1/3)$ and $z^+ = \Delta(1/3)$. Some choices were not valid even for small values of x. Table 1 summarizes the choices valid in a large interval.

Operation	Rounding mode	Parity of n	Bound on x
$\lfloor \diamond(x/3) \rfloor$	downward	-	3×2^n
$\lfloor \diamond(x/3) \rfloor$	to nearest	-	$3 \times 2^{n-1}$
$\lfloor \diamond(x.z^-) \rfloor$	to nearest	odd	3×2^n
$\lfloor \diamond(x.z^+) \rfloor$	downward	odd	$2^n - 1$
$\lfloor \diamond(x.z^+) \rfloor$	downward	even	$2^{n+1} - 2$

Table 1: *Various implementations of $\lfloor x/3 \rfloor$. Each line contains the operation, the rounding mode $\diamond$ under which it must be performed, the parity of the precision n under which it applies, and the largest value X such that the implementation is valid for all $x \in [0, X]$.*

We can notice that, like $\lfloor \diamond(x/y) \rfloor$, the implementations using a multiplication are valid under some constraints on the rounding mode and the domain. Static rounding modes better suit these kinds of operations.

Moreover, in odd precisions (which include the IEEE-754 double precision) and the rounding to nearest mode, the implementation $\lfloor \diamond(x.z^-) \rfloor$ is valid on a larger interval than the interval for $\lfloor \diamond(x/3) \rfloor$, and it should also be faster. In particular, this is useful in languages that work in double precision and the rounding to nearest mode.

It would be interesting to study other cases and see if similar properties hold for other values of y.

References

[1] N. Brisebarre, J.-M. Muller, and S. K. Raina. Accelerating correctly rounded floating-point division when the divisor is known in advance. *IEEE Transactions on Computers*, 53(8):1069–1072, August 2004.
http://perso.ens-lyon.fr/jean-michel.muller/DivIEEETC-aug04.pdf

[2] D. Daney, G. Hanrot, V. Lefèvre, P. Pélissier, F. Rouillier, and P. Zimmermann. The MPFR library, 2005.
http://www.mpfr.org/

[3] V. Lefèvre. The Euclidean division implemented with a floating-point division and a floor. Research report RR-5604, INRIA, June 2005.
http://www.vinc17.org/research/papers/rr_intdiv