# Y a-t-il des bugs dans GNU MPFR ?

# (proposition de stage M2)

**Lieu.** INRIA/LORIA, Nancy, `www.loria.fr`.

**Encadrant.** Paul Zimmermann, directeur de recherche, équipe Caramba, `Paul.Zimmermann@inria.fr`.

**Contexte.** Le calcul flottant est régi depuis 1985 par le standard IEEE 754. Révisé en 2008, ce standard définit : (i) des formats de représentation des nombres flottants, comme le format *double précision* sur 64 bits, comprenant 53 bits de mantisse — dont un bit implicite —, 1 bit de signe et 11 bits d'exposant, (ii) des modes d'arrondi (au plus près, vers zéro, vers plus l'infini, vers moins l'infini), et (iii) des opérations qui doivent renvoyer l'*arrondi correct*, à savoir l'unique valeur la plus proche de la valeur mathématique exacte $f(x)$ dans le format cible et avec l'arrondi donné.

Depuis 1985, les fabricants de micro-processeurs et certains concepteurs de logiciels s'acharnent à calculer l'arrondi correct le plus efficacement possible. Parfois, des « bugs » se glissent pernicieusement, comme celui de la division du Pentium découvert en 1994 [5]. C'est pour cette raison que Intel et AMD ont embauché des chercheurs pour prouver formellement les algorithmes avant de les implanter en silicium. C'est ainsi que John Harrison a rejoint Intel, et David Russinoff AMD.

En 2000 a commencé le développement de la bibliothèque GNU MPFR [1], bibliothèque de calcul flottant en précision arbitraire avec arrondi correct, utilisée entre autres pour compiler GCC, et faisant partie des distributions Linux. Si cette bibliothèque contient un certain nombre de tests de non-régression et de tests aléatoires, ces tests ne donneront jamais la même garantie qu'une preuve formelle. La raison est toute simple. Supposons qu'on veuille diviser deux nombres flottants $a$ et $b$ de 64 bits, et arrondir leur quotient vers zéro. On peut supposer qu'on a calculé une approximation sur 128 bits du quotient, à savoir $h \cdot 2^{64} + \ell$, avec $h$ et $\ell$ deux entiers d'au plus 64 bits, et que l'erreur sur cette approximation est d'au plus 1 sur $\ell$. Les cas difficiles pour l'arrondi correct sont ceux où $\ell = 000...000$ ou $\ell = 111...111$. Or, si on suppose que toutes les $2^{64}$ valeurs de $\ell$ sont équiprobables, ces cas difficiles se produisent avec une probabilité de $2^{-64}$. Il est donc très difficile de « tomber » sur ces cas difficiles via des tests aléatoires.

**Pistes de travail.** Le but du stage est de trouver des bugs dans GNU MPFR, via deux voies complémentaires : via une approche directe on cherchera des bugs qui ne dépendent pas de la taille des mots-machine, et donc qu'on pourra trouver via une approche exhaustive pour une petite taille ; et via une approche indirecte on cherchera à prouver formellement qu'il n'existe pas de bug, un blocage dans la preuve formelle nous guidant vers un bug potentiel.

D'une part, on implantera donc dans un nouveau logiciel MINI-MPFR les algorithmes utilisés pour les opérations arithmétiques de base de MFPR (addition, soustraction, multiplication, division, racine carrée). Ces algorithmes sont basés sur des opérations élémentaires sur des tableaux de mots (*limbs*) qui sont implantées via la bibliothèque GMP. Sur les machines modernes, un *limb* a 64 bits. Ce logiciel MINI-MPFR permettra de paramétrer le nombre $w$ de bits par limb, et avec $w = 2$ ou $w = 3$ on espère pouvoir détecter plus facilement des bugs, soit par des tests exhaustifs sur un petit nombre de mots, soit par des tests aléatoires.

D'autre part, on essaiera de prouver formellement les algorithmes utilisés par MPFR, que ce soit en petite précision [4] ou en précision arbitraire [2, 3]. On espère ainsi pouvoir trouver des bugs passés inaperçus jusque là, ou bien détecter des branches mortes, ce qui permettrait d'optimiser la bibliothèque MPFR.

Ce stage est proposé dans le cadre d'une collaboration avec Karthik Bhargavan (équipe Prosecco, Inria Paris).

## Références


[1] Fousse, L., Hanrot, G., Lefèvre, V., Pélissier, P., and Zimmermann, P. MPFR : A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw. 33*, 2 (2007), article 13.

[2] Lefèvre, V. The Generic Multiple-Precision Floating-Point Addition With Exact Rounding (as in the MPFR Library). In *6th Conference on Real Numbers and Computers 2004 - RNC 6* (Dagstuhl, Germany, Nov. 2004), pp. 135–145. Conference URL : http ://cca-net.de/rnc6/.

[3] Lefèvre, V. Correctly Rounded Arbitrary-Precision Floating-Point Summation. In *23rd IEEE Symposium on Computer Arithmetic (ARITH)* (Santa Clara, CA, United States, July 2016), IEEE.

[4] Lefèvre, V., and Zimmermann, P. Optimized Binary64 and Binary128 Arithmetic with GNU MPFR. In *24th IEEE Symposium on Computer Arithmetic (ARITH 24)* (London, United Kingdom, July 2017).

[5] Muller, J.-M. Algorithmes de division pour microprocesseurs : illustration à l'aide du "bug" du Pentium. *Technique et Science Informatiques 14*, 8 (1995), 1031–1049.