Sujet de stage M2 – Abstraction des jeux d'instructions vectoriels
Application sur l'architecture RISC-V et sur des solveurs HPC

Encadrants : Adrien Cassagne[*], Raphaël Gayno[†], Mihail Popov[‡] et Ani Anciaux Sedrakian[†]

[*]LIP6, équipe ALSOC, Paris
[†]IFP Energies nouvelles, Rueil-Malmaison
[‡]Inria, équipe TADaaM, Bordeaux

Description du sujet

Contexte

De nos jours, le jeu d'instructions ouvert RISC-V [1] de l'Université de Berkeley et ses implantations matérielles gagnent fortement en popularité. En effet, dans un contexte socio-économique complexe, l'Europe, dépendante des processeurs américains, veut se donner les moyens de devenir souveraine sur les semi-conducteurs. L'architecture RISC-V est donc une opportunité pour développer des processeurs européens capables de venir concurrencer les *leaders* du marché, tels que : Intel, AMD, ARM et Nvidia.

L'architecture RISC-V, abrégée RV, est organisée en extensions. L'une d'entre elles, l'extension "V" pour vectorielle (abrégée RVV), propose un sous jeu d'instructions dédié pour le calcul intensif de type *Single Instruction Multiple Data* (SIMD). Cette extension est une des clefs pour concevoir des processeurs compétitifs. RVV a été standardisé dans sa version 1.0 et elle commence a être implantée dans des SoC (Systems on a Chip) du marché (SpacemiT K1, SiFive P550, ...). Contrairement à AVX/NEON, qui ont une largeur fixe (128/256/512 bits), RVV est "indépendant de la longueur du vecteur" communément appelé *vector-length agnostic* (VLA). La longueur réelle des registres vectoriels (VLEN) dépend du matériel. Par conséquent, le même code fonctionne sur toutes les tailles sans recompilation. Cependant, aujourd'hui, son support logiciel est assez limité. Les compilateurs récents comme GCC 14+ ou Clang 17+ la supporte et propose une interface bas niveau (un jeu de fonctions intrinsèques). Dans le cas où le code est suffisamment régulier, ces compilateurs sont capables de produire un code optimisé contenant des instructions RVV. Cependant, sur des codes plus complexes, il devient essentiel de pouvoir adresser ces instructions et de ne pas uniquement se reposer sur le compilateur.

Cela est le cas pour les simulateurs d'écoulement dans les milieux poreux, tels que le simulateur de stockage de CO2, où la prise en compte des lois physiques et la résolution des systèmes linéaires constituent l'étape la plus gourmande en temps de calcul. La prise en compte de la vectorisation peut permettre de réaliser des gains significatifs pour ce type d'application.

Objectifs

L'objectif de ce stage est de combler le gap entre les applications de "haut niveau" et le code bas niveau. Pour ce faire, adresser l'extension V de RISC-V est un bon point de départ. Il sera nécessaire de concevoir une abstraction de plus haut niveau que du code assembleur (ou des fonctions intrinsèques). Des précédents travaux sur les jeux d'instructions SIMD (NEON, SVE, SSE, AVX, AVX-2, AVX-512) ont été menés autour de la bibliothèque MIPP [2] (écrite en C++). Le code source de cette dernière est ouvert [2] et utilisé à la fois dans le monde industriel et dans le mode académique. Une des tâches principale de ce stage est d'enrichir la bibliothèque MIPP pour qu'elle puisse prendre en charge RVV. Pour y parvenir, les paragraphes suivants détaillent les sous-étapes envisagées.

Compréhension et prise en main de RVV. Dans un premier temps le ou la stagiaire devra comprendre le jeu d'instructions RISC-V et son extension vectorielle. Ensuite, il faudra étudier les modifications logicielles à apporter dans MIPP puis en intégrer un sous ensemble représentatif. Le stagiaire aura au moins une architecture moderne RVV à disposition pour evaluer ses contributions (la Banana Pi BPI-F3 [3]). Des tests unitaires seront à établir et à passer enfin de valider que chaque opération donne bien les résultats attendus.

1. RISC-V ISA : `https://riscv.org/wp-content/uploads/2019/12/riscv-spec-20191213.pdf`
2. Dépôt Git MIPP : `https://github.com/aff3ct/MIPP`
3. Banana Pi BPI-F3 : `https://docs.banana-pi.org/en/BPI-F3/BananaPi_BPI-F3`

Génération de la bibliothèque SIMD. Le LIP6 et l'IFPEN ont conjointement mis au point un générateur de code pour automatiser certaines tâches réplétives lors de la construction de la bibliothèque MIPP. Dans un second temps, le ou la stagiaire devra prendre en main ce générateur et l'enrichir pour supporter RVV. Cette étape est très importante et marque une rupture avec la philosophie initiale de MIPP. En effet, la génération de code permettra de grandement accélérer les processus de portage. Elle ouvre aussi la porte à des sujets ayant une forte valeur ajoutée comme par exemple la génération des micro-noyaux vectoriel/SIMD via des LLMs (comme Claude, GPT) et l'optimisation automatique de ces micro-noyaux.

Étude de performance sur des codes HPC réels. Ensuite, le ou la stagiaire devra évaluer cette nouvelle version de MIPP dans la bibliothèque de résolution de systèmes linéaires MCGSolver, où MIPP est déjà intégrée [1]. Cette étape permettra d'abord de vérifier l'absence de régression liée aux nouvelles implémentations générées via les instructions maitrisées comme AVX-512 et SVE, et de mettre en évidence la qualité des nouvelles interfaces. Une étude de performance avant/après est attendue. À l'issue de cette étude de validation et de non-régression, le travail consistera à évaluer le gain de performance sur les architectures RISC-V avec RVV.

Au-delà de l'optimisation vectorielle. La finalité du stage est d'étudier l'impact de la vectorisation sur l'entièreté de la pile logicielle. C'est un objectif ambitieux car beaucoup de paramètres interagissent ensemble. Par exemple, l'usage de vecteurs large sur une partie des systèmes x86 peut engendrer plus de consommation énergétique, causant une réduction forcée de la fréquence de calcul. Il est ainsi difficile de prévoir si une application bénéficiera plus d'une fréquence élevée ou d'une vectorisation plus agressive sur une architecture cible. D'autre exemples de paramètres interagissant avec la vectorisation incluent la stratégie des caches mémoire, les *prefetch hardware*, le parallélisme au niveau des threads et processus ou même les passes de compilations [3].

Exploration de paramètres affectant la vectorisation avec de l'IA. L'IFPEN et l'Inria ont co-développé CORHPEX, un outil utilisant de l'IA pour guider et quantifier l'interaction entre différents types d'optimisations sur la performance et la consommation énergétique. Le ou la stagiaire pourra ainsi utiliser l'outil et différentes techniques basées sur les LLMs pour explorer différents compromis de design software et hardware.

Hébergement et organisation

Le stage sera hébergé au laboratoire LIP6 (Jussieu, Paris) et à l'IFP Énergies nouvelles (Rueil-Malmaison). Les détails sur l'hébergement sont à discuter en entretien mais le ou la candidat(e) retenu(e) sera amené(e) à partager sont temps entre ces deux entités. Des réunions régulières (en distanciel et parfois en présentiel) seront organisées avec l'Inria (Bordeaux).

Compétences attendues

- Programmation assembleur (RISC-V, MIPS, ARM et/ou x86, au moins un des 4)
- Architecture SIMD (SSE, AVX, NEON et/ou SVE, au moins un des 4)
- Bases en programmation orientée objet (langages C/C++ et Python)
- Environnement Unix et gestionnaire de version (Git)
- Excellente communication

Candidater : Merci d'envoyer un e-mail aux quatre encadrants, à savoir adrien.cassagne@lip6.fr, raphael.gayno@ifpen.fr, mihail.popov@inria.fr et ani.anciaux-sedrakian@ifpen.fr contenant un CV avec au minimum les notes de M1 et de M2.

Références

[1] Ani Anciaux-Sedrakian, Raphael Gayno, Thomas Guignon et Aboul Karim Maarouf : Efficient high-fidelity simulations for the energy transition using ARM and x86 64 architectures. *In ECCOMAS*, 2024. URL `https://www.scipedia.com/public/Anciaux-Sedrakian_et_al_2024a`.

[2] A. Cassagne, O. Aumage, D. Barthou, C. Leroux et C. Jégo : MIPP : A portable C++ SIMD wrapper and its use for error correction coding in 5G standard. *In Workshop on Programming Models for SIMD/Vector Processing (WPMVP)*, Vösendorf/Wien, Austria, février 2018. ACM. URL `https://doi.org/10.1145/3178433.3178435`.

[3] Lana Scravaglieri, Mihail Popov, Laércio Lima Pilla, Amina Guermouche, Olivier Aumage et Emmanuelle Saillard : Optimizing performance and energy across problem sizes through a search space exploration and machine learning. *Journal of Parallel and Distributed Computing*, 180:104720, 2023.