Sujet de stage M2 – Support du jeu d'instructions vectoriel RISC-V dans la bibliothèque d'abstraction MIPP

Encadrants : Adrien Cassagne et Roselyne Chotin

Laboratoire LIP6, équipes ALSOC et CIAN, Paris

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.

D'un autre côté, l'Europe a pris un peu de retard sur la déferlante IA et plus précisément sur les architectures dédiées au traitement efficace des réseaux de neurones profond. Nvidia est aujourd'hui le fournisseur numéro 1 de matériel pour l'apprentissage et l'inférence de ces réseaux. Une possibilité pour combler le retard serait de concevoir des processeurs RISC-V avec des accélérateurs sous la forme d'extensions dédiées pour l'IA. RISC-V IME [2] est un exemple d'instructions spécialisées, en cours de standardisation, conçues au dessus de l'extension vectorielle "V". En ce sens, des travaux récents de la littérature scientifique démontrent que RVV est un point de départ crédible pour ajouter des instructions dédiées pour l'IA [3, 5]. Enfin, l'extension vectorielle RVV a été standardisée dans sa version 1.0 et est implantée dans des SoC du marché (SpacemiT K1, SiFive P550, ...).

Cependant, aujourd'hui, le support logiciel de l'extension vectorielle RVV est assez limité. Les compilateurs récents comme GCC 14+ ou Clang 17+ supportent cette nouvelle extension et propose une interface bas niveau (un jeu de fonctions intrinsèques) pour adresser ces nouvelles instructions. 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.

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, SSE, AVX, AVX-512) ont été menés autour de la bibliothèque MIPP [1]. Le code source de cette dernière est ouvert [3]

1. RISC-V ISA : `https://riscv.org/wp-content/uploads/2019/12/riscv-spec-20191213.pdf`
2. RISC-V IME : `https://github.com/space-mit/riscv-ime-extension-spec`
3. Dépôt Git MIPP : `https://github.com/aff3ct/MIPP`

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.

Dans un premier temps le 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 les intégrer. Le stagiaire aura au moins une architecture moderne RVV à disposition pour tester ses contributions (la Banana Pi BPI-F3). Des tests unitaires seront à passer enfin de valider que chaque opération donne bien les résultats attendus. Le support des nombres flottants sur 16 bits, essentiels dans les applications d'IA, pourra faire l'objet d'une contribution supplémentaire.

Ensuite, grâce au nouveau support de RVV dans MIPP, le stagiaire pourra exécuter des codes existants et étudier le gain sur des architectures RISC-V supportant l'extension V. Cette étape permettra de quantifier le gain apporté par RVV ainsi que d'éventuellement identifier des problèmes de conception et de revoir l'optimisation de l'interfaçage avec MIPP. En ce sens, la bibliothèque AFF3CT [2], dédiée aux communications numériques et reposant sur MIPP, sera testée. En effet, d'autres travaux dans ce domaine ont montré qu'il était possible d'atteindre des gains de performance significatifs en utilisant RVV [4].

Enfin, le candidat pourra étudier l'intégration de MIPP dans des frameworks d'IA comme PyTorch (*backend* ATen), TensorFlow et/ou OneDNN. Là aussi, une analyse des performances apportées par MIPP+RVV est attendue.

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
- Gestionnaire de version (Git)

Candidater : Merci d'envoyer un e-mail aux deux encadrants, à savoir adrien.cassagne@lip6.fr et roselyne.chotin@lip6.fr, contenant un CV avec au minimum les notes de M1 et de M2.

Références

[1] 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`.

[2] A. Cassagne, O. Hartmann, M. Léonardon, K. He, C. Leroux, R. Tajan, O. Aumage, D. Barthou, T. Tonnellier, V. Pignoly, B. Le Gal et C. Jégo : AFF3CT : A fast forward error correction toolbox ! *Elsevier SoftwareX*, 10:100345, octobre 2019. ISSN 2352-7110. URL `https://doi.org/10.1016/j.softx.2019.100345`.

[3] V. N. Chander et K. Varghese : A soft RISC-V vector processor for edge-AI. *In International Conference on Embedded Systems (VLSID)*, pages 263–268. IEEE, février 2022. URL `https://doi.org/10.1109/VLSID2022.2022.00058`.

[4] V. Razilov, E. Matúš et G. Fettweis : Communications signal processing using RISC-V vector extension. *In International Wireless Communications and Mobile Computing (IWCMC)*, pages 690–695. IEEE, mai 2022. URL `https://doi.org/10.1109/IWCMC55113.2022.9824961`.

[5] C. Wang, C. Fang, X. Wu, Z. Wang et J. Lin : SPEED : A scalable RISC-V vector processor enabling efficient multiprecision DNN inference. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pages 1–14, 2024. ISSN 1557-9999. URL `https://doi.org/10.1109/TVLSI.2024.3466224`.