Stage de recherche en informatique : Allocation de registres pour langage dédié à la cryptographie

2023-2024

Responsable du projet : M. Sid Touati, professeur des universités

Contacts : `Sid.Touati@inria.fr`

Co-encadrant du projet : M. Benjamin Grégoire, chargé de recherche INRIA

Laboratoires : I3S/INRIA-Sophia, Université Côte d'Azur

1 Motivations

Les programmes de cryptographie ont des exigences et des spécificités propres, qui les rendent différents des programmes généralistes : le code doit être certifié et ses performances stables et satisfaisantes afin d'éviter les attaques. Dans ce contexte, un langage de programmation spécialisé, appelé `Jasmin`, a été conçu pour l'écriture de programmes cryptographiques bas niveau, efficaces et sûrs. Une bibliothèque appelée `libjade` a été écrite en `Jasmin`. Bien que ciblant avant tout les primitives cryptographiques post-quantiques, c'est-à-dire résistant à un attaquant quantique, elle contient aussi de la cryptographie classique.

Par sa nature délibérément bas niveau, un code écrit en `Jasmin` dépend de l'architecture du processeur cible. Le pari de ce langage est de renoncer à la portabilité en écrivant du code dédié pour tirer un maximum de performances du matériel. Ce langage est actuellement proche de l'assembleur intel `x86`, il est prévu de l'étendre pour considérer d'autres architectures comme `ARM-V7` et `RISC-V`

L'avantage de programmer avec `Jasmin` est d'avoir accès à des outils de certification, étape nécessaire pour tout code de cryptographie fiable. Des mots clés permettent à l'utilisateur d'exiger au compilateur de mettre une donnée en registre ou en mémoire. Le programmeur a ainsi le plein contrôle sur l'allocation de registres. Malheureusement, il arrive que des codes soient rejetés par le compilateur à cause de contraintes fortes sur les registres, créant ainsi de la frustration. Il n'est pas aisé au programmeur de comprendre pourquoi l'allocateur de registres échoue à satisfaire les contraintes.

Afin de remédier à ce problème, une extension du langage permettra au programmeur de créer une troisième famille de données : outre les données en registre et les données en mémoire, cette troisième famille de données sera constituée de variables laissées libres au compilateur de les placer en registres s'il peut, et les spiler en mémoire lorsqu'il veut. Cette nouvelle liberté du compilateur permettra à plus de codes de compiler sans se faire rejetés.

Cela nous amène donc à intégrer un nouvel allocateur de registres dans le compilateur du langage `Jasmin`, qui devra cohabiter avec les exigences de ce langage : permettre au programmeur d'interférer dans l'allocation de registres en exigeant soit de mettre des données en registres soit de les placer en mémoire.

2 État de l'art

L'allocation de registres est l'un des thèmes les plus importants et les plus étudiés en compilation optimisantes. Compte tenu de la complexité et des variantes possibles de ce problème fondamental pour les performances d'un compilateur, beaucoup d'articles sont publiés régulièrement pour apporter de nouveaux algorithmes. Dans notre cadre, le problème ici est d'adapter un algorithme existant dans le contexte d'un langage bas niveau comme `Jasmin` : l'ordre des instructions ne doit pas être modifié, et les exigences du programmeur vis à vis des registres doivent être respectées.

3 Contenu du stage de recherche

L'étudiant devra aborder le sujet sur plusieurs axes

1. Faire une étude bibliographique détaillée sur le sujet, en ce focalisant sur les résultats formels obtenus à l'ENS-Lyon et à l'INRIA, voir la

section de bibliographie ;

2. Adapter l'algorithme d'allocation de registres et d'insertion de spill dans le contexte du compilateur du langage `Jasmin` ;
3. Implémenter l'algorithme et faire une évaluation des performances.

4 Prérequis

Compilation

Références

[1] Florent Bouchez, Alain Darte, Christophe Guillon, and Fabrice Rastello. Register allocation and spill complexity under SSA. Research report, Laboratoire de l'informatique du parallélisme, August 2005.

[2] Florent Bouchez, Alain Darte, and Fabrice Rastello. Register allocation : what does Chaitin's NP-completeness proof really prove ? Research Report LIP RR-2006-13, Laboratoire de l'informatique du parallélisme, March 2006.

[3] Boubacar Diouf, Albert Cohen, and Fabrice Rastello. A Polynomial Spilling Heuristic : Layered Allocation. In *CGO 2013 - International Symposium on Code Generation and Optimization*, Shenzhen, China, February 2013. IEEE.

[4] Boubacar Diouf, Albert Cohen, Fabrice Rastello, and John Cavazos. Split Register Allocation : Linear Complexity Without the Performance Penalty. In *International Conference on High Performance and Embedded Architectures and Compilers*, page 15 p, Pisa, Italy, October 2010.