Documentation de Développement Durable

OUAZZANI CHAHDI Mohammed
BENTOUIMOU Yassine
ABDELWAHID Amine
AIT DRISS Salma
LEMDAGHRI ALAOUI Ibrahim

27 janvier 2023

Table des matières

1 Introduction

Tout au long du projet, l'optimisation et la minimisation de l'utilisation des ressources étaient une priorité. Cela se traduit dans les instructions qu'on utilise et les choix de conception pour élaborer des solutions vis-à- vis des problèmes qu'on rencontre.

2 Evaluer la consommation énergétique

Pour évaluer la consommation énergétique du projet nous nous basons sur une analyse de complexité des algorithmes et les structures de données qu'on utilise, ainsi que les cycles nécessaires pour effectuer les instructions.
La machine ima nous présente avec l'option -s pour afficher le nombre de cycles internes effectués lors de l'exécution. Nous avons utiliser alors cette options pour analyser l'état d'efficacité de notre projet.
Donc pour les tests de performances dans la génération du codes nous avons eu les résultats suivants :

- syracuse42 : 1448 cycles
- ln2 : 18358 cycles
- ln2_fct : 21665 cycles

De plus nous avons pu comparer l'efficacité de notre compilateur à GCC pour un programme qui utilise des listes chaînées, et nous avons pu atteindre environ 70% du temps de compilation.

3 discussion de l'impact de vos choix de compilation de Deca vers ima

Les différentes fonctionnalités du projet nous ont présenté plusieurs défis qui nous nécessitaient de trouver des solutions tout en gardant un coût minimal.
En effet, nous avons eu besoin de garder des informations sur les états des registres et de la mémoire. La première idée était d'utiliser une classe pour garder toutes ces informations et la mettre à jour avec chaque changement de l'état du programme.
Au fur et à mesure qu'on avançait sur le projet, on s'est rendu compte qu'on avait stocké un surplus d'informations. Alors pour minimiser l'utilisation des ressources, nous avons décidé de suivre l'évolution des registres et de la mémoire juste en utilisant des entiers, et les mettre à jour en les incrémentant et décrémentant.
Nous avons aussi eu besoin de stocker des données. Donc en fonction de la manipulation des données, nous avons choisi la structure de données correspondante. Par exemple, pour les messages d'erreur nous avons eu besoin de l'unicité des messages pour ne pas les reproduire dans le code assembleur. Les instructions d'ajout et les tests d'appartenance ont été beaucoup utilisé pour cette structure, alors nous avons choisit de l'implémenter comme une HashMap.
Toujours dans la perspective de minimiser l'utilisation des ressources, nous avons pu réutiliser plusieurs données pour résoudre des problèmes. Notamment, pour résoudre le problème d'unicité des étiquettes, nous utilisons la Location des lexèmes ou les adresses

Nous disposons des cycles internes nécessaires pour l'exécution d'une instruction. Alors pour avoir une exécution optimale, nous avons choisi les instructions qui minimisent ce nombre de cycles internes. Donc pour interagir avec la mémoire nous utilisons des LOAD/STORE le maximum possible au lieu de PUSH/POP. C'est-à-dire en fonction des disponibilités des registres nous donnons la priorité à utiliser des LOAD/STORE
De plus, la machine IMA nous présente la possibilité d'avoir une opérande flexible lors des opérations. Nous utilisons cela alors pour minimiser le nombre d'instructions utilisées.

4 discussion de votre processus de validation, et stratégie mise en œuvre pour en diminuer l'impact énergétique sans nuire à l'effort de validation et à la qualité du compilateur

Notre processus de validation consistait à exécuter plusieurs séries de tests qui parcourent nos différents classes java du package deca, et de les exécuter tous à travers des scripts conçues pour chaque étape du compilateur.
Afin de diminuer l'impact énergétique de nos tests, on a essayé de concevoir des tests qui étaient précis et ciblait les classes à tester, et qu'on a pas créés un nombre de tests excessif.

5 prise en compte de l'impact énergétique dans votre extension

Notre choix pour l'architecture ARM a été pris en se basant sur plusieurs raisons. Parmi les principales raisons influant ce choix est son impact énergétique. En effet, l'architecture ARM est connue pour sa faible consommation d'énergie faible : les processeurs ARM se basent sur l'architecture RISC "Reduced Instruction Set Computing". Cette dernière, par sa simplicité, permet d'assurer une efficacité énergétique supérieure. Ce choix respecte ainsi les valeurs de notre équipe en ce qui concerne le développement durable.

6 Propositions

Comme proposition, nous avons pensé à quelques solutions pour optimiser le temps et le cout d'execution de certaines instructions. Malheureusement on a pas eu le temps de les implémentés correctement. Voici quelques exemples :

6.1 Division et multiplication par un multiple de 2 en utilisant un shift :

En effet, au lieu de faire des calculs simples en utilisant l'instruction assembleur : **mult**, on pourrait très bien utilisé le **shift**. Cette instruction est beaucoup plus optimum en termes de coût par rapport à l'instruction **mult**. En effet, son principe est beaucoup plus simple : à chaque fois qu'on souhaite multiplier par 2, il suffit de décaler les bits vers la gauche et à chaque fois qu'on souhaite diviser par 2, on décale les bits vers la droite.

6.2 modulo

Pour le modulo : $a \equiv ?[2]$.
Il suffit de prendre le dernier bit de poids faible (le bit le plus à droite), s'il est égale à 0 alors le reste de la division est 0 (a $\equiv 0[2]$)$ets'ilestégaleà1lerestedeladivisionest1(a \equiv 1[2])$.