Projet Compilation
Compilateur pour le langage SoS

BONNAIL Julie - FAID Mohamed - MAREGHNI Sofian - PARISI Charles

09/01/2023

Université de Strasbourg

Contents

1 Présentation

1.1 Le sujet

Le but du projet est de réaliser en groupe de quatre étudiants un compilateur pour le langage SoS. Nous avons du tenir compte des spécificités du langage pour réussir à réaliser un programme qui gènere un code assembleur MIPS.

1.2 Notre organisation

Pour s'organiser à plusieurs nous avons travaillé en groupe dans des bibliothèques ou à distance quand cela n'était pas possible. Le reste du temps nous travaillions de notre côté mais toujours en collaboration grâce à GitHub.

Notre travail a été effectué dans l'ordre suivant. Nous avons tout d'abord écrit la grammaire. Nous avons repris les structures quad et liste chainée vues en TP. Nous avons ensuite implémenté notre table des symboles. Puis nous avons commencé à paralléliser le travail. Une partie du groupe à traduit les quadruplets générés par l'analyse semantique ascendate. L'autre partie a implémenté l'analyse sémantique.

Nous avons ensuite pu passer à l'élaboration des tests, à la rédaction de rapport, et à la préparation de l'oral.

Étapes suivies
Opérations arithmétiques
La toute première étape du projet a été de faire fonctionner les opérations arithmétiques basiques (+, -, *, /, %) qui ne fonctionnent que sur les entiers

L'étape suivante a été de faire fonctionner le stockage de variables de type tableau d'entiers.

Opérations logiques :

Avant de pouvoir implémenter les structures de contrôle, il faut implémenter les opérations (portes) logiques. Les opérations implémentées sont les suivantes : eq (pour ==), ne (pour !=), lt (pour inf), le (pour $\leq$ gt (pour sup), ge (pour $\geq$), a (pour & &), o (pour $\|\|$), et!.

Structures de contrôle :

Nous avons implémenté les structures de contrôle logiques suivantes : if et if-else. La structure de contrôle de flot for est aussi while

Et finalement, nous avons implémenté les fonctions (la récursivité marche aussi).

2 Usage

Pour utiliser notre compilateur il faut:
Utiliser 'make' pour générer l'executable 'bin/sos'. Utiliser 'make clean' pour supprimer les fichiers temporaires.

Indication des erreurs:
La plupart des erreurs détectées pendant l'analyse sont indiquées et expliquées en indiquant la ligne où à été provoqué l'erreur.

3 Dossiers

Le projet est séparé en plusieurs dossiers :
bin contient l'éxécutable généré
include contient tous les en-têtes c
src contient les fichiers sources c
obj contient les fichiers o
test contient tous les tests

4 Capacités

Voici les fonctions que nous avons réussi à implémenter :
Opération sur les entiers : +,-,/,*
Opération logique sur les entiers et les strings
Structure de contrôle : while, if, ifelse
Les tableaux
Les fonctions

5 Tests

Nous avons crée des tests pour notre programme mais ceci ne sont pas automatiques. Il faut les réaliser manuellement.
Ils sont situés dans tests/in et permettent de tester les différentes fonctionnalités de notre compilateur.

6 Difficultés et améliorations

Nous n'avons pas eu le temps d'implémenter toutes les fonctions voulues. Cela vient principalement du fait que nous avions mal debuté ce projet.Nous avons eu du mal à commencer le projet, et avons eu besoin de beaucoup de temps pour comprendre le sujet. Nous avons commencé par travailler sur un répertoire à une seule branche, ce qui nous a compliqué la tâche de travailler en parallèle. De plus, en ayant moins de recul sur le projet nous avions fait des choix au départ qui n'étaient pas judicieux. Nous avons implémenté des structures qui se sont révelées fausses ou inutiles. Nous nous sommes retrouvés face à de nombreuses segmentation fault au vu du nombre de pointeur que notre projet contient et nécessite. Nous nous en sommes aperçu et nous avons préféré recommencer sur des bases plus sûres, quitte à aller moins loin dans ce projet. Nous n'avons pas eu le temps dâimplémenter des tests, mais nous générons du mips que nous avons testé avec le vm. Le programme généré n'est pas optimisè, encore une fois par manque de temps.

Pour améliorer notre projet, nous pourrions ajouter de nouvelles fonctions, et des tester automatisés.