TLC-S7: Rapport de projet

Pour le 26/01/2025

Responsable: LAMARCHE Fabrice

Sommaire

Le rapport de projet aborde à la fois la partie technique de la réalisation du compilateur ainsi que la partie organisationnelle du projet.

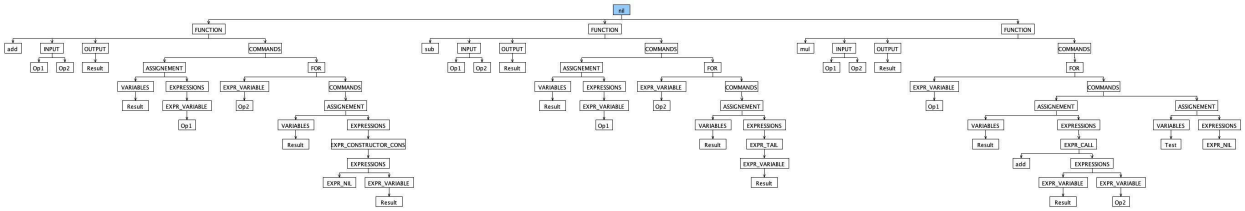
1. Description technique

1.1. Architecture du compilateur et de la chaine de compilation

(depuis le code source en WHILE à la récupération d'un programme exécutable)

1.2. AST

Voici notre AST. Nous avons essayé de le rendre le plus propre possible. Nous l'avons contruit à partir du fichier `full.while` (répertoire `test/lang`) :



Sur cet AST, nous remarquons que notre programme contient 3 fonctions :

- Une fonction `add`
  - 2 paramètres d'entrée : `Op1` et `Op2`
  - 1 paramètre de sortie : `Result`
  - Une suite de commandes :
    - Une assignation stockée dans la variable `Result` prenant la valeur de `Op1`
    - Une boucle `for` itérant sur `Op2`. Elle stocke dans `Result` la construction d'un arbre ayant pour fils gauche `nil` et pour fis droit `Result`
- Une fonction `sub`
  - 2 paramètres d'entrée : `Op1` et `Op2`
  - 1 paramètre de sortie : `Result`
  - Une suite de commandes :
    - Une assignation stockée dans la variable `Result` prenant la valeur de `Op1`
    - Une boucle `for` itérant sur `Op2`. Elle stocke dans `Result` la `tail` de `Result`
- Une fonction `mul`
  - 2 paramètres d'entrée : `Op1` et `Op2`
  - 1 paramètre de sortie : `Result`
  - Une bouce `for` itérant sur `Op1`. Elle stocke dans `Result` le résultat de la fonction `add` qui est appelée sur les paramètres `Result` et `Op2`

1.3. Table des symboles

- Nous avons implémenté une classe `SymbolInfo` qui a pour attributs `line` (numéro de ligne), `column` (numéro de colonne) et `content` (contenu du symbol). Elle permetd'énumérer les informatation concernant le symbole.
- Ensuite, nous avons implémenté `SymbolTable`, la table des symboles. Nous l'avons représenté en `Stack<Map<String, SymbolInfo>>`. Nous y avons implémenté plusieurs méthodes pour ajouter des symboles à un contexte, ajouter un contexte à la table des symboles, vérifier si le symbole est dans un contexte etc.

attention symbols enum c'est de la d ça sert null part, je fais un git blame pour voir c'est qui qui a fait

1.4. Design Pattern Visiteur

Nous avons mis en place une classe abstraite `Visitor.java` se basant sur le Design Pattern visitor. Elle permet de visiter n'importe quel label présent dans l'AST (fonctions, inputs, outputs, expressions, variables etc.)

Grâce à cette classe abstraite, nous avons pu faire un visiteur pour la table des symboles (`SymbolsVisitor.java`).

1.5. Génération de code 3 adresses à partir de l'AST

- visiteur pour le code 3 adresses (`IntermediateCodeVisitor.java`)

parler de FunctionSignature - les types (`TypesVisitor.java`)

```
+-----------+----------------------+----------------------+
| Operation |         arg1         |         arg2         |
+-----------+----------------------+----------------------+
| input     | input name           | stack position       |
| return    |                      | register's address   |
| define    | new register's label | tree                 |
| retrieve  | new register's label | register's address   |
| mov       | register's label     | register's address   |
| setHead   | register's label     | register's address   |
| setTail   | register's label     | register's address   |
| call      | function's label     | number of parameters |
| param     |                      | register's address   |
| return    |                      | register's address   |
+-----------+----------------------+----------------------+
```

Traduction complète d'un programme !!

1.6. Optimisation de code si elle a été réalisée

1.7. Génération de code à partir du code 3 adresses

1.8. Bibliothèque runtime de WHILE écrite dans le langage cible

2. Description de la validation du compilateur

2.1. Méthodologie utilisée

2.2. Code coverage

2.3. Bilan

2.3.1. Ce qui fonctionne

2.3.2. Ce qui ne fonctionne pas

2.3.3. Fonctionnalités restant à implémenter ?

3. Description de la méthodologie de gestion de projet

3.1. Outils utilisés pour la gestion du projet

3.2. Etapes de développement et découpage des tâches

3.3. Rapport de travail individuel

4. Post mortem : Organisation du projet

4.1. Ce qui a bien fonctionné

4.2. Ce qui a moins bien fonctionné

4.3. Avec plus de recul, que ferions-vous ?