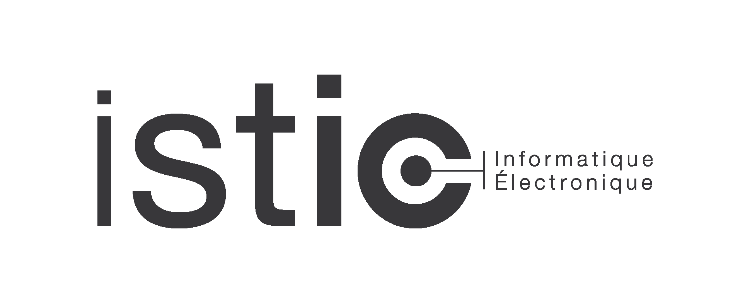
**

Rapport de TP de compilation

*Compilateur VSL+*

Mathieu Grandmontagne & Maël Nogues

08/12/2016

Rapport de TP de compilation

Compilateur VSL+

L’objectif de ce TP est de faire la partie avant d’un compilateur de VSL+ vers du code MIPS. On génèrera donc du code 3 adresses à partir du code VSL+ donné à notre compilateur. Ce code 3 adresses est ensuite traduit en code MIPS par la partie arrière du compilateur, fournie par les professeurs dans le cadre de ce TP.

# Méthodologie :

Pour réaliser le TP, nous avons tout d’abord lu le sujet, puis nous avons codé le compilateur. Étant déjà familier avec l’outil ANTLR et ne pouvant pas faire d’autre tests que la génération du code 3 adresses depuis chez nous, nous n’avons pas pu réellement découper le codage en plusieurs étapes. Nous avons développé le compilateur en testant la génération du code 3 adresses à chaque étape, puis, après le retour des vacances de la Toussaint, nous avons testé avec MIPS et nachos afin de découvrir et tenter de résoudre les quelques bugs qui n’apparaissaient pas à la compilation du code 3 adresses. Nous avons ensuite développé un fichier Bash pour faciliter les tests de notre compilateur par les professeurs, le fichier execute permet la compilation d’un fichier de test du vsl au MIPS, puis au binaire nachos, et l’exécute.

# Bilan de réalisation :

Le compilateur que nous présentons est capable de générer du code 3 adresses fonctionnel en MIPS sous nachos pour les expressions, les affectations, les tableaux, les print, les read, la gestion des blocs, la déclaration des variables, les expressions avec des variables, les instructions de contrôle if, while et la séquence. Il gère aussi la compilation en code 3 adresses des déclarations et appels de fonctions (avec les prototypes). Cependant, l’exécution après compilation en code MIPS (partie fournie par les professeurs responsables du TP) ne fonctionne pas correctement.

# Programme de test :

Le programme de test suivant teste toutes les fonctionnalités de la grammaire qui fonctionnent aussi bien en code 3 adresses qu’en code MIPS.

|  |
| --- |
| FUNC VOID main()  {  // Test de déclaration  INT x, y, z, a[2]  // Test d'affectation  y := 0  // Test d'affectation d'expression avec variable  x := 1 + y  // Test de condition  IF x THEN  x := 5  ELSE  x := 6  FI  // Test d'affectation d'expression dans un tableau  a[0] := 1  PRINT a[0], "\n"  // Test d'affectation de variable dans un tableau  a[1] := x  // Test d'affectation d'un tableau dans une variable  z := a[1]  // Test d'affichage de variables, de tableau et de texte  PRINT "Test sample\n", x, "\n", a[1], "\n", z, "\nSecond test sample\n"  // Test d'affectation d'une variable depuis la fonction système READ  READ z  PRINT z  } |

Le programme de test suivant génère un code 3 adresses correct mais ne donne pas les résultats attendus lors de l’exécution avec nachos.

|  |
| --- |
| PROTO INT test(k,l)  FUNC VOID main()  {  INT a, b  a := 1  b := test(a,a)  PRINT b  }  FUNC INT test(k,l)  {  RETURN k + l  } |

Nous avons réalisé plusieurs petits programmes de tests afin de tester des fonctionnalités précises de notre compilateur, tel qu’un programme simple pour tester le fonctionnement des tableaux, un autre programme pour tester le fonctionnement des affectations dans les variables et la fonction de print. Nous avons aussi réalisé un test pour vérifier le bon isolement des blocs internes à d’autres blocs.

Tous ces fichiers de tests peuvent être trouvé dans le dossier test fourni dans l’archive contenant ce rapport.

# Conclusion :

Nous avons rencontré quelques difficultés lors de ce TP. La première et la plus importante étant l’impossibilité de faire des tests à partir de chez nous, ce qui nous a empêché d’utiliser une méthode agile pour le développement du compilateur.

Après les vacances de la toussaint, nous avons donc pu commencer à tester avec la compilation en MIPS pour exécuter sous nachos. Nous avons remarqué que les opérations que nous faisions sur les tableaux (affectations, accès à des emplacements du tableau, …) ne fonctionnaient pas, malgré un code 3 adresses généré correct. Nous avons donc retracé le problème à travers le code assembleur MIPS pour finalement trouver le problème, qui venait d’une référence erronée dans notre grammaire.

Nous avons ensuite rencontré un problème sur l’utilisation des paramètres dans les fonctions. Nous avons identifié que le problème vient des références aux paramètres dans le code assembleur MIPS. Cependant, nous n’avons pas pu retracer le problème jusqu’à notre grammaire pour pouvoir le corriger avant la date de rendu. Pour des raisons de complétion, nous avons laissé les fonctions dans la grammaire, cela vous permet de bien voir que le code 3 adresses est correctement généré et que les fonctions, en elles-mêmes, fonctionnent correctement. Il s’agit seulement d’un problème de passage de paramètres dans ces fonctions qui ne fait pas correctement le lien entre la variable utilisée dans la fonction et le paramètre passé à cette fonction.