Projet de Compilation

Alexis Theas-laban

June 6, 2021

1 Introduction

Un compilateur du langage tpc, un sous-ensemble du langage C.

1.1 Installation

Le makefile fourni permet de générer l'exécutable avec la commande `make`.

1.2 Utilisation

./tpcc [OPTION...] [FILE]

- File Le fichier d'entrée. Si aucun fichier n'est spécifié, le compilateur lit sur l'entrée standard.
- Options
  - **tree, -t** affiche l'arbre abstrait sur la sortie standard.
  - **symtab, -s** affiche toutes les tables des symboles sur la sortie standard.
  - **help, -h** affiche une description de l'interface utilisateur et termine l'exécution.

1.3 Modules

- **symbol-table** fournit toutes les fonctions nécessaires à la manipulation d'une table des symboles (make, add, get, clear, etc.).
- **abstract-tree** contient les structures et fonctions nécessaires à la construction de l'AST.
- **sem-check** construit différentes tables des symboles pour vérifier les erreurs sémantiques du programme.
- **translator** parcourt l'AST et le traduit en instructions assembleur.

2 Implémentations

2.1 Arbre abstrait

Chaque noeud de l'arbre contient un pointeur vers une table des symboles. Cette table sera utilisée lors de la vérification des erreurs sémantiques.

2.2 Gestion des types

Les types sont implémentés par un type structuré à deux champs: le premier indique le type primitif (int, void, char, struct, etc.), le second contient un pointeur vers la chaine qui représente le type structuré.

2.3 Table des symboles

On définit une structure `SymbolTable` par une table de hachage contenue dans le fichier htable.h.

Les tables des symboles contiennent des `ST_Entry` pouvant représenter une fonction, une structure ou une variable. La structure `ST_Entry` de cette table contient un champ union représentant les paramètres d'une fonction ou les champs d'une structure. Cette structure contient également un champ offset pour indiquer le décalage au registre rbp. Pour une fonction il s'agit de la taille de ses paramètres, pour une structure la taille de ses champs. Ce champ permet de faciliter la traduction en code assembleur.

3 Difficultés rencontrées

3.1 Vérification sémantique

Partie la plus longue du projet de par la quantité d'erreurs à testé.

3.1.1 Control reaches end of non-void function

On doit vérifier qu'une fonction contient un return accessible dans le programme. Il n'est pas nécessaire d'examiner toutes les branches conditionnelles mais seulement celle qui sont obligatoires. Par exemple, une fonction avec un return dans chaque instruction d'un if-else n'appartient pas à cette catégorie d'erreur car on est certain d'y accéder. En revanche, les return dans une instruction if et while ne permettent pas d'éviter cette erreurs. Il n'est donc pas nécessaire de les vérifier.

Solution: On utilise une fonction récursive pour examiner chaque branche du programme, évitant les instructions qui ne nous intéressent pas.

3.2 Traduction en assembleur

Cette partie du projet a posé le plus de difficulté, l'ajout des structures en particulier a nécessité la révision de notion telle que la gestion de la pile et les accès mémoires.

3.2.1 Paramètres structuré

Les fonctions peuvent recevoir un paramètre structuré.

Solution: Les arguments des fonctions sont placés dans la pile pour faciliter l'implémentation des types structurés. Une fonction à un paramètre structuré est traitée comme une fonction contenant autant de paramètre que de champs dans cette structure. De cette façon, il suffit de recopier chaque paramètre contenue dans la pile au-dessus de l'appel de la fonction vers le bloc d'activation locale (rbp - offset, où offset est une valeur définit dans la table des symboles).

3.2.2 Renvoi de structure

Les fonctions peuvent également renvoyer une structure. On ne peut pas employer le même stratagème que pour le retour de primitive car il faut autant de registres que de champs.

Solution: On alloue un espace dédiée dans la pile avant l'appel de la fonction et l'empilement des arguments. Cet espace contiendra la structure renvoyé par la fonction.

3.2.3 Assignation de structures

Une structure renvoyée par une fonction ou contenue dans une variable doit être assignable à une autre.

Solution: De la même manière que pour les paramètres structurée on traite les structures comme un ensemble de variables primitives. Il suffit donc de dépiler les valeurs de la pile dans un registre puis de les assigner aux variables locales ou globales correspondantes.

4 Conclusion

Ce projet a été particulièrement enrichissant en ce qui concerne le fonctionnement d'un compilateur et du language nasm en général. Cependant de par la complexité et la longueur des travaux à effectuer je n'ai pas pu implémenter la totalité des fonctionnalités demandées. Les erreurs sémantiques et lexicales n'affichent pas la ligne d'erreurs et les instructions reade et readc ne fonctionnent pas si elles précédent une expression conditionnelle.