Benoit Kevin
Corblin Aurélien

Projet de Compilation

1.Introduction

Dans le cadre du projet de compilation en L3 Informatique, nous avons eu a faire un compilateur ecrit en c qui traduit un sous langage du c , le TPC, en nasm suivant les convention AMD64. Pour faire cela , nous avons utilisé l'arbre abstrait construit par le projet d'analyse syntaxique (qui est la base de depart pour ce projet).

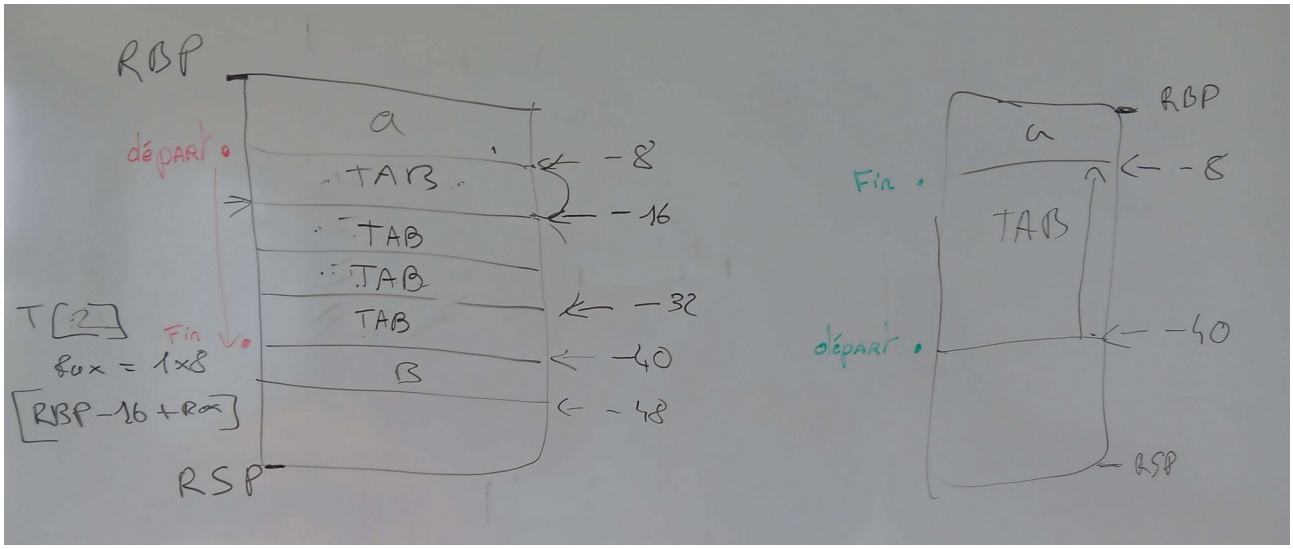
2.Implémentation du Compilateur : Difficultés Rencontrées

Pour implémenter ce compilateur , on a du respecter les contraintes imposées pour le langage tpc (que nous ne répèterons pas ici car tpc ,étant un sous langage du c, est assez evident et sa syntaxe est entierement decrite dans le sujet d'analyse syntaxique et son comportement ,pour la semantique , est decrit dans le sujet de ce projet)

Le premier problème rencontré a été la structure pour notre table des symboles, en effet nous avons mis du temp avant de trouver la bonne, nous avons d'abord tester de faire q'une table et d'avoir toutes les informations (sa porté, de quelle fonction vient t-il si il est ldeocal, son emplacement dans les parametre si s'en est un, etc...) sur un symbole dans la structure d'un symbole directement , ce qui etait une mauvaise idée puisque l'on devait parcourir toute la table a chaque fois et faisait des verifications sur chaque symbole pour trouver celui que l'on voulait.
Notre deuxieme tentative nous amener a avoir une table de variable globale et une table par fonction , mélangeant les parametres et les variables locale.
Finalement on a opté pour une table pour les variables globale et deux tables par fonction , une pour les parametres et une autre pour les variables locale.

Deuxième problème, à cause de l'offset on lisait les tableaux dans le mauvais sens (à l'envers, et du coup on ne les lisais même pas, segfault selon les cas), exemple :

si on avait une variable locale son offset était a -8 ;
si un tableau de 3 cases la suivait, son offset lui était de -16 ;
et si on mettait une variable ensuite l'offset était de -40 ;



à gauche : l'erreur qu'on faisait au début
à droite : la solution

En résumé : On donnait l'offset au symbole du tableau et ensuite on décalait l'offset de la taille du tableau pour la prochaine variable.

Solution : On a fais l'inverse, on décale d'abord l'offset de la taille du tableau et ensuite on affecte l'offset à son symbole.

Troisième problème, on a oublié une ligne de code :

```
section .bss
        tab resq 2

push tab
call <function>

<function> :
        mov rax, qword [rax + 8]
        (notre erreur était de s'être arrêté la)
        mov rax, qword[rax]
```

par exemple au lieu d'accéder à la première case du tableau (la valeur), on accédait à l'adresse de la premiere case du tableau.

Quatrième problème, pour check un symbole dans la table des symboles ce qu'on faisait c'était de :

- chercher dans la table des variables locales de la fonction :
    si on ne trouvait pas le symbole ou qu'il était de mauvais type
    (ex : on cherche un tableau et on trouve a la place une variable)

- dans ce cas on va chercher dans la table des paramètres de la fonction :
    si on ne trouvait pas le symbole ou qu'il était de mauvais type

- dans ce cas on va chercher dans la table des variables globales :
    si on ne trouvait pas le symbole ou qu'il était de mauvais type :
        on renvoie une erreur

l'erreur c'était de continuer s'il était de mauvais type.

Résolution : Dès qu'on trouve le symbole mais qu'il est de mauvais type → erreur

3.Annexe

(si vous avez besoin d’aide « make help » dans le terminal)

- pour construire le compilateur :
    make

- après faire ./bin/tpcc -h pour savoir comment s’en servir (un peu de lecture)

- après l’obtention d’un fichier <nasm_name>.asm vous pouvez construire son exécutable en effectuant la commande : make build TARGET=<nasm_name>.asm (l’exécutable se trouve dans le dossier bin)

- tests :
    sh test.sh