RAPPORT

Projet Traduction L3

PETIT-HENGEN Maxime & PICHOU Maxime

25/05/2014

Rapport

Projet Traduction L3

Table des matières

Introduction

Le but du projet est d'écrire un compilateur du langage TPC en utilisant les outils Flex et Bison. Le compilateur traduira le langage TPC en langage de la machine virtuelle. Pour cela on utilisera la grammaire donnée dans le sujet afin d'écrire des actions sémantiques avec l'analyse grammaticale et former des productions avec l'analyse lexicale.

Choix d'implémentations

Structure

Pour faciliter la traduction, nous avons choisis de stocker les variables et les constantes dans une structure contenant :

- Son nom - *char nom[NAMESIZE]*,
- Son adresse - *int adr*,
- Un booléen permettant de savoir si la variable est constante - *Bool is_cst*.

Nous avons également choisi de stocker les fonctions dans une structure, elle contient :

- Son nom - *char nom[NAMESIZE],*
- Son label - *int label*,
- Le nombre de variables et paramètres - *int nb_alloc*,
- Le nombre de paramètres - *int nb_param*,
- Un tableau de variable - *Var var[MAX].*

Variables globales

Pour gérer les labels et les adresses, on utilise des variables globales nous permettant de gérer les labels des sauts (*if/else/while*) mais aussi les labels de fonctions. Aussi, ces variables nous servent d'indices de tableau pour les fonctions et les variables. Ce sont les suivantes :

- *int jump_label* permet de savoir à quel label le programme doit sauter,
- int global_var_adr permet de savoir quelle est la première adresse libre,
- *int label_fct_inUse* nous indique l'indice de la fonction en cours (différent de l'adresse),
- *int nb_global* est le compteur du nombre de variables globales,
- *int nb_ifElse* nous indique dans quelle branche du *if else* on se trouve, cela nous permet de gérer des *if else* imbriqués,
- *int nb_while* de même *nb_while* permet de gérer les while imbriqués,
- *int label_if[MAX]*, avec l'aide du compteur *nb_ifElse* on peut récupérer le label dans le cas où le if est faux (*JUMPF*),
- *int label_while[MAX],* agit de la même façon que *label_if*, si le test du *while* n'est plus valide on pourra sauter au bon label.

Fonctionnement

Notre compilateur utilise deux structures qui permettent pour la première de stocker l'adresse et le nom de la variables (resp. constante) globale et pour la seconde d'enregistrer le label et le nom de la fonction ainsi que l'ensemble des variables présente dans celle-ci (paramètre inclus).

Déclaration

Lors de la déclaration d'une variable, on commence par vérifier si cette dernière n'existe pas déjà grâce au parcours du tableau des variables. Si la variable existe, le programme sort une erreur et s'arrête. Sinon, on crée cette variable en enregistrant son identifiant ainsi que son adresse dans la pile. Plus tard, lors de son utilisation, on lui affectera sa valeur avec les instruction *SAVE* si elle est globale (ou dans le *main*) ou *SAVER* si elle est contenue dans une fonction. On peut modifier une variable seulement si ce n'est pas une constante. On peut vérifier cela à l'aide du booléen *is_cst*.

Fonction

Chaque fonction est créée lors de sa déclaration. La structure prévue à cet effet contient donc son identifiant ainsi que son label pour pouvoir l'appeler plus tard avec l'instruction *CALL*. Toutes les variables présentes dans les fonctions sont totalement indépendantes du *main* et des autres fonctions. On peut donc avoir des variables avec le même nom dans deux fonctions différentes sans problème. Les paramètres de chaque fonction sont enregistrés dans la structure et ont une adresse calculée par rapport au nombre de paramètre dans la fonction et de leur position dans les parenthèses. Enfin pour retourner une valeur, on la place dans le registre *reg1* puis on emploie l'instruction *RETURN*. Par convention, on n'a pas besoin de marquer le mot-clé *return* à chaque fin de fonction. Le compilateur fera un *RETURN* automatique.

Sauts et boucles conditionnels

Afin d'exécuter un if et/ou un else proprement, la machine virtuelle a besoin de faire des sauts à des labels particuliers. Pour cela, on a dû enrichir la grammaire TPC par des règles intermédiaires afin d'exécuter des actions sémantiques. La variable globale *jump_label* sert de numéro de label pour les instructions *JUMPF* et *JUMP*. Cependant, ce sont les tableaux *label_if* et *label_while* qui sont utilisés comme relation entre les sauts et les labels (ils doivent avoir le même numéro).

Difficultés rencontrées

La première difficulté a été de trouver des structures répondant à nos besoins, ce qui nous a conduits à réécrire plusieurs fois le projet, ces structures nous ont également permis de résoudre nos problèmes liés à la gestion des variables plus efficacement. Enfin nous avons eu des problèmes pour la gestion des *if/else* et *while* imbriqués, pour cela nous avons créé deux tableaux pour mémoriser les labels.

La plus grande difficulté a été la gestion des *FREE*, *MALLOC* et tableaux. Nous n'avons pas réussi à les implanter correctement sans obtenir d'erreur.

Conclusion

Pour conclure, ce projet nous a permis d'avoir un aperçu des éléments qu'un compilateur doit gérer. Il nous a également permis de nous familiariser avec les outils Flex et Bison.