Université Gustave Eiffel

Travaux Dirigés de Compilation n°5

Licence d'informatique

———

Traduction des expressions en assembleur. Bloc d'activation. Entrées-sorties

Les objectifs de ce TD sont de développer une première version de la traduction des expressions en assembleur pour le projet de compilation, de réaliser un bloc d'activation, et de commencer à implémenter les entrées-sorties.

▸ Exercice 1. *Traduire des expressions arithmétiques en assembleur*

1. *Assurez-vous que votre projet de compilation construit des arbres abstraits pour au moins les soustractions simples en* `tpc`.
2. *Dans le TP de compilation 1, vous avez implémenté en C un parcours de l'arbre abstrait, qui crée des tables de symboles. Implémentez du code en C qui ouvre un fichier* `_anonymous.asm` *en écriture avant le parcours de l'arbre, et qui referme ce fichier après. Ce fichier contiendra la traduction en* `nasm` *du fichier d'entrée en* `tpc`.
3. *Implémentez du code en C qui, quand on rencontre une fonction pendant le parcours de l'arbre abstrait, vérifie si le nom de la fonction est* `main`. *Si c'est le cas, votre code écrira dans le fichier de sortie le minimum en* `nasm` *pour commencer et terminer le programme cible :*

   ```
   global _start
   section .text
   _start:
   ```

   (ici vous lancerez dans le 1.4 le parcours de l'arbre abstrait du main du TPC)

   ```
   mov rax, 60
   mov rdi, 0
   syscall
   ```

4. *Implémentez du code qui traduit les soustractions entre constantes en utilisant la pile. Vous pouvez vous limiter au cas où la soustraction fait partie de la fonction* `main`. *Testez en assemblant* `_anonymous.asm` *et en exécutant le résultat.*

▸ Exercice 2. *Réaliser un bloc d'activation*

Reprenez votre code du TP 3, exercice 2.1, et transformez-le en fonction. Allouez la mémoire pour les 4 variables (le plus petit entier, le plus grand, le nombre d'entiers non nuls et la somme) dans le bloc d'activation.

▸ **Exercice 3.** ***Lire et afficher en assembleur***

Écrivez en assembleur les fonctions suivantes :

1. `my_putchar` *qui affiche à l'écran un caractère ASCII donné en paramètre,*
2. `my_getint` *qui renvoie comme valeur un entier positif ou nul lu au clavier en décimal. On ne demande pas de sauter les caractères blancs : si le premier caractère lu n'est pas un chiffre, la fonction doit terminer l'exécution du programme et le programme doit renvoyer la valeur de retour 5, qui code une erreur d'entrées-sorties.*

Ces fonctions serviront dans votre projet à traduire les appels de fonction `putchar(c)` *et* `getint()`*. Dans le projet, les entrées-sorties en* `tpc` *se feront en appelant 4 fonctions dont voici les prototypes :*

```
void putchar(char c);
void putint(int i);
char getchar(void);
int getint(void);
```

Le compilateur écrira ces fonctions directement en `nasm`*, et le code* `tpc` *ne pourra pas les redéfinir.*