RAPPORT COMPILLING TPCC

Documentation :

Le but de ce projet est de pouvoir créer du code nasm à partie d'un arbre abstrait précédemment créer grâce au projet d'analyse syntaxique.

Le projet est sous la forme d'un dossier contenant des sous dossiers :

- *Src* : fichiers sources de développements
- *Bin* : fichiers binaires
- *Obj* : fichiers intermédiaires pour passer des fichiers « src » au fichier « bin »
- *Test* : jeux d'essaies dans des sous répertoires (good, syn-err, sem-err, warn)
- *Nasm* : fichier contenant le programme demandé en nasm

Lancement du projet :

- ' *make* ' : génération du fichier « /bin/tpcc »
- '*./tpcc [OPTION] FILE.tpc* ' :
    - [OPTION] :
        - -t, --tree affiche l'arbre abstrait sur la sortie standard
        - -h, --help affiche une description de l'interface utilisateur et termine l'exécution
        - -s, --symtabs affiche toutes les tables de symbole sur la sortie standard
- ' cd nasm' : fichier contenant le fichier « FILE.asm » créer
- ' make ' : génération de l'exécutable du fichier « FILE.asm »
- ' ./FILE' : exécution du programme

Valeur de retour :

- « 0 » si le programme source ne contient aucune erreur (même s'il y a des avertissements)
- « 1 » s'il contient une erreur lexicale ou syntaxique
- « 2 » s'il contient une erreur sémantique
- « 3 » ou plus pour les autres sortes d'erreurs : ligne de commande, fonctionnalité non implémentée, mémoire insuffisante.

Organisation :

Nous avons programmé ensemble ce projet lors des séances de TP avec l'aide de notre charger de TD si nous avions des problèmes. Et pendant nos temps de travail apprentis à l'université.

Difficulté rencontrée :

- Vérifier les erreurs sémantiques en parcourant les fonctions : Nous avons opté pour un switch pour vérifier dans l'ordre de l'arbre les différents nœuds en appelant récursivement les fils avec la table des symboles pour vérifier qu'un paramètre donné était du bon type.

- Traduire les conditions avec les opérateurs logiques en nasm : Pour traduire les opérateurs logiques nous avons utilisés deux paramètres en plus qui sont des chaines de caractères et qui représente les futurs sauts des opérateurs logiques. Par exemples dans une condition comme « a && b » on vérifie « a » si « a » est vrai on va à la vérification de « b » mais si a est faut on va directement au bloc de condition.

- Trouvé la bonne adresses des paramètres et de la valeur de retour : Nous avons utilisé une petite formule de calcul qui est « 16 + 8*i » avec i le numéro du paramètre. Le 16 correspond à l'ancien « rbp » + l'adresse de « ret ». Bien sûr avant d'allouer la place pour les paramètres on utilise le bloc d'activation.