TP4 – Génération de code machine

hivers 2019

Chargé de laboratoire : Félix Brunet (felix.brunet@polymtl.ca)

1. Objectifs

— Générer du code machine à partir du code intermédiaire
— Gérer l'allocation des registres

2. Travail à faire

Pour ce dernier laboratoire, vous devrez convertir un bloc de base de code intermédiaire en code machine. Ensuite, vous utiliserez le code machine généré pour compléter l'implémentation du calcul d'un élément de la suite de Fibonacci en utilisant un interpréteur.

Le langage machine que vous avez à supporter est celui décrit à la section 8.2.1 du livre. Vous n'avez besoin que d'une fraction des commandes, puisque vous ne gérer pas les tableaux, les pointeurs et les flux de contrôle. Vous aurez surtout besoin des commandes « LD », « ST » et « OP ».

Langage en entrée

Le langage que comprend la grammaire fournis avec le TP est différente de celle des TP précédent. Elle comprend un nœud pour enregistrer le nombre de registre disponible, la définitions des variables vives pour chaque instruction et les instructions, qui forme un bloc de base (et ne comporte donc aucun flux de contrôle). De plus, le langage ne comprend que des entiers.

Étapes

1. Modifier le visiteur qui convertit le code intermédiaire en code machine dans le fichier PrintMachineCodeVisitor.java pour tenir compte des contraintes indiquées plus bas.
2. Générer le code machine pour les deux blocs de code intermédiaire qui seront utilisés dans le calcul d'un nombre de la suite de Fibonacci pour :
    - 3 registres
    - 5 registres
3. Insérer manuellement les deux blocs de code machine dans le fichier examples/fibb.asm du simulateur.
4. Lancer le simulateur pour vérifier si votre code est correct. Vous devriez être capables de calculer n'importe quel nombre de la suite de Fibonacci.

Contraintes

- Vous avez un maximum de 256 adresses mémoires disponibles. Toutefois, les blocs de code fourni ne dépasseront jamais cette limite.
- Vous devez supporter un nombre variable de registres.
- Comme décrit dans la description du langage, vous ne pouvez pas utiliser les variables ailleurs que dans les LD et dans les ST. Vous devrez donc transférer vos valeurs dans les registres et vice-versa pour appliquer les opérations.
- Le code intermédiaire contient aussi les informations sur les variables vives.
- L'allocation de registres respecte l'algorithme d'allocation simple (slides 2-6)

3. Procédure

3.1. Données

Les données des tests se trouvent dans le dossier test-suite, séparées par visiteur. Le contenu de chaque sous-répertoire est le suivant :

- data/ : Le code intermédiaire
- result/ : Le résultat de l'exécution du visiteur, soit le code machine

Dans les fichiers de test (data/), les informations sur les variables vives à chaque instruction sont présentes avant le code intermédiaire.

Vous pouvez aussi exécuter le visiteur sur un seul fichier et observer la sortie dans un terminal. Pour ce faire, dans un terminal, exécutez la commande suivante dans le dossier du projet :

java -cp out/production/Langage analyzer.Main nom-fichier

3.2. Utilisation du simulateur

Le simulateur a été écrit en Python 3 et nécessite l'utilisation des bibliothèques Arpeggio et NumPy. Si elles ne sont pas installées, vous pouvez les installer avec la commande pip3 install --user arpeggio==1.5. et pip install --user numpy.

Au début de la méthode simulate du fichier simulator.py, l'environnement est créé avec les contraintes du nombre de registres : c'est à cet endroit que vous pouvez le changer.

Pour exécuter le code machine, lancez la commande python3 run_tests.py. Tous les fichiers présents dans le dossier examples seront lancés.

Remarque : vous pouvez constater que ce simulateur a été écrit avec un analyseur lexical et syntaxique. Le fichier .peg décrit la grammaire et un ensemble de visiteurs qui prépare et effectue la simulation.

4. Barème

Le TP est évalué sur 20 points.

Les critères sont l'implémentation, le respect de l'algorithme d'allocation simple et des contraintes, et le bon fonctionnement du simulateur de Fibonacci avec le code généré.

5. Remise

Remettez sur Moodle une archive nommée log3210-tp4-*matricule1*-*matricule2*.zip avec uniquement le fichier PrintMachineCodeVisitor.java.

L'échéance pour la remise est le dimanche 14 avril 2019 à 23 h 55.

Le TP **doit** être fait en équipe de deux. Il y aura exactement une équipe de trois par groupe si le nombre d'étudiant est impair.

Une pénalité de 10 points (50 %) s'appliquera par jour de retard.

Une pénalité de 4 points (20%) s'appliquera si la remise n'est pas conforme aux exigences (nom du fichier de remise, fichier demandé seulement).