Compilation – TP 10 : Allocation des registres en RETROLIX

Université Paris Diderot – Master 1

(2014-2015)

Cette feuille de TP vous donne les étapes à suivre pour rendre réaliste l'allocation de registres de RETROLIX. Il a deux grandes parties dans ce sujet : la première partie est obligatoire tandis que la seconde ne l'est pas strictement pour réussir à obtenir un compilateur ciblant l'architecture MIPS32 (Linux).

Le code source correspondant à ces travaux pratiques se trouve sur le GIT, dont on rappelle l'URL :

`http://moule.informatique.univ-paris-diderot.fr:8080/Yann/compilation-m1`

On rappelle que vous devez faire des *commits* réguliers (à chaque modification de votre code) pour que nous puissions suivre votre avancement.

1 Une allocation de registres réaliste

Pour que l'allocation de registres puisse être utilisée sur l'architecture MIPS32 sous Linux, il faut absolument implémenter les conventions d'appel. On rappelle que les conventions d'appels sont décrites dans la spécification de MIPS et servent à rendre possible l'appel de code de bibliothèques (compilées avec d'autres compilateurs) depuis le code compilé par notre compilateur.

Exercice 1 (Implémentation des conventions d'appels) *1. Modifier la compilation des appels de fonction pour prendre en compte les conventions de passage des arguments et la récupération du résultat de l'appel.*

2. *Modifier la compilation des corps de fonction pour respecter les conventions de passage des arguments et la récupération du résultat de l'appel de cette fonction.*
3. *Modifier la compilation des corps de fonction pour qu'une fonction commence par sauvegarder tous les registres "callee-save" avant de commencer son calcul, et finisse par restaurer tous ces registres avant de retourner à son appelant.*

□

Exercice 2 (Rajout des registres physiques dans le graphe d'interférence) *1. Pourquoi doit-on prendre en compte les registres physiques dans le graphe d'interférence ?*

2. *Étendre l'analyse de vivacité pour qu'elle prenne en compte les registres physiques. Attention à bien prendre en compte les registres utilisés et définis par un appel de fonction. Comment s'assurer que les registres "caller-save" sont bien sauvegardés par l'appelant d'une fonction ?*

□

Exercice 3 (Précoloriage) *1. Modifier l'algorithme de coloriage du graphe en s'assurant que les registres physiques sont précolorés. (Pour rappel, les nœuds précolorés ne sont pas simplifiables.)*

□

Exercice 4 (Variables globales) *1. Modifier votre coloriage pour vous assurer que les variables globales ne sont jamais représentées par des registres.*

□

2 Une allocation de registres réaliste et efficace

Les conventions d'appel introduisent un grand nombre d'instruction `MOVE V, R` où `R` est un registre physique et `V` une variable. En favorisant, tant que possible, l'affectation du registre `R` à la variable `V` transforme ces instructions en `MOVE R,R`, qui peut être supprimée sans modifier la sémantique du programme source.

Pour introduire ce biais dans le coloriage, on introduit des relations de préférence entre les nœuds du graphe d'interférence et on modifie l'algorithme de coloriage pour favoriser la fusion des nœuds en relation de préférences.

Exercice 5 (Étendre le type du graphe d'interférence) *1. Modifier le type des arêtes du graphe pour pouvoir représenter les deux types de relation "conflit"et "préférence".*

2. Modifier le type des nœuds pour qu'un nœud puisse représenter plusieurs variables et/ou un registre (plutôt qu'au plus une variable ou un registre).

3. Modifier la production du graphe d'interférence pour introduire une relation de préférence entre deux variables (ou/et registre) apparaissant dans la même instruction `MOVE`*.*

□

Exercice 6 (Prise en compte des préférences dans le coloriage) *Soit K le nombre de couleurs disponibles. On rappelle les deux critères de fusion qui assurent l'un et l'autre que le problème du coloriage n'est pas plus compliqué dans le graphe où les nœuds sont fusionnés que dans le graphe initial :*

- ***Briggs*** *: on peut fusionner deux nœuds A et B dans le graphe si, dans le graphe obtenu après fusion, il y a strictement moins de K nœuds du voisinage du nouveau nœud qui ont un degré supérieur ou égal à K.*
- ***Georges*** *: on peut fusionner deux nœuds A et B dans le graphe si pour tout nœud du voisinage de A, soit ce nœud est déjà dans le voisinage de B ou son degré est strictement inférieur à K.*

1. Implémenter une opération de fusion de deux nœuds dans un graphe.

2. Implémenter une opération qui transforme une préférence en conflit.

3. Modifier l'algorithme de coloriage pour prendre en compte les relations de préférence.

□

Exercice 7 (Optimisation du code final) *1. Implémenter le module* `RetrolixKillMove` *qui s'occupe de remplacer toutes les instructions de la forme* `MOVE X,X` *par des commentaires.*

2. Implémenter le module `RetrolixCompression` *qui transforme le programme en supprimant tous les commentaires (et en mettant à jour les sauts en conséquence).*

□