Université Paris 7 2011-2012
Master 1 Ingénierie Informatique

Roberto.Amadio@pps.jussieu.fr
Nicolas.Ayache@gmail.com

# Compilation - Projet n°1
## Sélection d'instruction par tuiles

*Vous pouvez trouver tous les sujets au format électronique sur internet à l'adresse :*
`http://www.pps.jussieu.fr/~ayache`
*dans la rubrique* Enseignements.

Ce sujet propose d'optimiser l'étape de sélection d'instruction lors de la traduction des programmes Cminor en programmes RTLabs. L'optimisation se fera selon deux axes : ajouter des instructions au langage RTLabs, et implanter la sélection d'instruction par tuiles.

*Pensez à tester régulièrement votre code !*

# 1 Présentation

Dans le compilateur CerCo, c'est lors du passage de Cminor à RTLabs que les expressions sont déstructurées. Or, on ne peut sélectionner efficacement une instruction machine pour traduire une expression qu'en connaissant son contexte. Comme la structure de graphe rend difficile la détection de *schéma* d'expressions, c'est pendant la traduction d'un programme Cminor en un programme RTLabs qu'il est le plus simple de sélectionner les instructions les plus efficaces pour représenter une expression.

Cependant, le langage RTLabs ne dispose pas de toutes les instructions MIPS. Dans un premier temps, votre travail consistera à ajouter des instructions au langage RTLabs afin de produire du code plus efficace.

Dans un second temps, vous devrez modifier la traduction des programmes Cminor en programmes RTLabs pour utiliser autant que possible les instructions les moins coûteuses.

# 2 Ajout d'instructions

## 2.1 RTLabs

Les instructions MIPS dont il faut ajouter une représentation en RTLabs (fichier `src/RTLabs/RTLabs.mli`) sont :

1. `addi dstr, srcr,` $i$ : ajoute l'entier $i$ au contenu du registre `srcr` et stocke le résultat dans le registre `dstr` ;
2. `lw dstr,` $i$`(addr)` : ajoute l'entier $i$ au contenu du registre `addr`. Ceci forme une adresse en mémoire dont la valeur qui s'y trouve est stockée dans le registre `dstr` ;
3. `sw srcr,` $i$`(addr)` : ajoute l'entier $i$ au contenu du registre `addr`. Ceci forme une adresse en mémoire où la valeur contenue dans le registre `srcr` est stockée ;
4. *cmp_zero* `srcr,` *label* où *cmp_zero* $\in \{$`bltz, blez, bgtz, bgez, beqz, bnez`$\}$ : saute à l'étiquette *label* selon le résultat de la comparaison de la valeur contenue dans le registre `srcr` avec 0 ;
5. *cmp* `srcr1, srcr2,` *label* où *cmp* $\in \{$`blt, ble, bgt, bge, beq, bne`$\}$ : saute à l'étiquette *label* selon le résultat de la comparaison des valeurs contenues dans les registres `srcr1` et `srcr2`.

Vous pouvez également ajouter des instructions supplémentaires si celles-ci vous semblent pertinentes.

### 2.2 Dépendances

En plus des modifications effectuées sur le langage RTLabs (et/ou sur d'autres modules selon votre solution), vous devez adapter les modules qui en dépendent :

1. `src/RTLabs/RTLabsPrinter.ml` : vous devez déterminer un affichage adéquate pour les nouvelles instructions ;
2. `src/RTLabs/RTLabsInterpret.ml` : vous devez donner une sémantique aux nouvelles instructions en vous inspirant de celles déjà existantes ;
3. `src/RTLabs/RTLabsToRTL.ml` : pour le cas des nouvelles instructions, vous pouvez choisir de retourner une erreur ou de les traduire (très faible priorité) ;
4. pour tous les autres modules, vous pouvez retourner une erreur pour les nouveaux cas, le cas échéant.

*Note :* pour tester simplement les modifications des modules `RTLabsPrinter` et `RTLabsInterpret`, vous devrez probablement attendre d'avancer dans la section suivante.

## 3 Sélection d'instruction par tuiles

À présent, vous devez modifier la traduction des programmes Cminor en programmes RTLabs (fichier `src/cminor/cminorToRTLabs.ml`). La traduction doit retourner un programme le moins coûteux possible, où chaque instruction RTLabs a un coût de 1.

Pour cela, vous pouvez reprendre la méthode actuellement utilisée qui consiste à spécialiser le traitement de certaines expressions ou instructions Cminor grâce au filtrage `ocaml` afin d'utiliser des instructions RTLabs adaptées. Cependant, la meilleure solution consiste en une traduction qui calcule le coût des différentes possibilités et choisit la moins coûteuse.

## 4 Modalités

**Récupération des sources.** Vous pouvez récupérer les sources de CerCo complètes jusqu'à RTL à l'aide de la commande suivante :

`git clone ~ayachen/CerCo` répertoire/de/votre/choix/

Il est conseillé d'utiliser un répertoire différent de celui que vous avez utilisé pour les séances précédentes afin de ne pas perdre votre travail.

À partir de ces sources, vous pouvez organiser votre projet comme vous le souhaitez : ajouter des fichiers, en retirer, modifier ceux déjà existants, etc.

**Délivrable.**
- Le projet est à réaliser par groupe d'au plus 4 personnes.
- Chaque groupe doit envoyer une tarball compressée à l'adresse `nicolas.ayache@gmail.com` au plus tard le mardi 6 décembre 2011.
- Le nom de cette archive doit être les noms des membres du groupe séparés par des tirets. *Exemple :* si un groupe est constitué de Vincent Dupont et de Pierre Vacheret, leur archive devra être nommée `dupont-vacheret.tgz`.

– L'archive doit contenir un répertoire du même nom. *Exemple :* dans le cas du groupe de Dupont et de Vacheret, leur archive `dupont-vacheret.tgz` doit contenir le répertoire `dupont-vacheret`.
– Ce répertoire doit à son tour contenir au moins : un répertoire nommé `CerCo` qui contient les sources du projet, et un rapport nommé `rapport.pdf` au format `pdf`. Au final, l'arborescence de la tarball doit ressembler à ce qui suit :

```
⊢ dupont-vacheret/
  ⊢ CerCo/
    ⊢ src/
    ⊢ tests/
    ⊢ Makefile
    ⊢ ...
  ⊢ rapport.pdf
```

– Les sources doivent compiler, et ce par l'unique invocation de la commande `make` à leur racine.

**Rapport.** Vous êtes libre de rédiger le rapport comme vous le souhaitez. Cependant, il doit faire apparaître au moins les éléments suivants :
– une courte description de la méthode de traduction des programmes Cminor en programmes RTLabs ;
– ce qui a été réalisé, c'est-à-dire les instructions qui ont été ajoutées. Pour chaque nouvelle instruction, décrivez brièvement l'ensemble des modifications apportées, et listez les différents fichiers et fonctions qui ont nécessité une modification ;
– ce qui n'a pas été réalisé ou ce qui ne fonctionne pas correctement, avec une description de l'idée du travail à fournir ou du problème ;
– une comparaison de l'efficacité (en nombre d'instructions RTLabs générées) de votre traduction par rapport à la version d'origine ;
– les problèmes rencontrés s'il y en a eus, quelles que soient leurs natures : bugs importants (qui proviennent éventuellement des sources originales), difficultés d'implantation, d'organisation, de compréhension, etc.

**Évaluation.** La note attribuée à chaque groupe dépendra de plusieurs critères, par exemple : l'avancée du travail, la qualité du code (lisibilité), sa robustesse (peu d'erreurs à l'exécution), sa correction (peu d'erreurs de traduction), la qualité du rapport, le respect des consignes, le nombre d'étudiants dans le groupe (à qualité de projet égale, un groupe moins nombreux aura une meilleure note), l'autonomie.