Pablo Rauzy
pr@up8.edu
https://pablo.rauzy.name/teaching/liec

UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Langages : interprétation et compilation

Université Paris 8 – Vincennes à Saint-Denis
UFR MITSIC / L3 informatique

Séance c (TP) : Démarrage projet

N'oubliez pas :

- Les TPs doivent être rendus par courriel au plus tard la veille de la séance suivante avec "[liec]" suivi du numéro de la séance et de votre nom dans le sujet du mail, par exemple "[liec] TPc Rauzy".
- Quand un exercice demande des réponses qui ne sont pas du code, vous les mettrez dans un fichier texte `reponses.txt` à rendre avec le code.
- Le TP doit être rendu dans une archive, par exemple un tar gzippé obtenu avec la commande `tar czvf NOM.tgz NOM`, où `NOM` est le nom du répertoire dans lequel il y a votre code (idéalement, votre nom de famille et le numéro de la séance, par exemple "rauzy-tpc").
- Si l'archive est lourde (> 1 Mo), merci d'utiliser `https://bigfiles.univ-paris8.fr/`.
- Les fichiers temporaires (si il y en a) doivent être supprimés avant de créer l'archive.
- Le code doit être proprement indenté et les variables, fonctions, constantes, etc. correctement nommées, en respectant des conventions cohérentes.
- Le code est de préférence en anglais, les commentaires (si besoin) en français ou anglais, en restant cohérent.
- **N'hésitez jamais à chercher de la documentation par vous-même sur le net!**

Dans ce TP :

- Écriture d'un compilateur complet très simple.

Exercice 0.

Nous allons commencer par écrire ensemble en Racket un petit compilateur qui prend en entrée un programme consistant en un unique entier et qui produit le code MIPS qui va afficher cet entier.

1. Le lexer (`lexer.rkt`).
2. Le parseur (`parser.rkt`) et les structures pour la syntaxe parsée (`ast.rkt`).
3. L'analyse sémantique (`analyzer.rkt`) et les structures pour l'AST (`ast.rkt`).
4. La compilation (`compiler.rkt`) et les structures pour la représentation intermédiaire de l'assembleur MIPS (`ast.rkt`).
5. L'écriture des instructions MIPS (`mips-printer.rkt`).

Exercice 1.

Nous allons ensuite (toujours ensemble) rajouter la possibilité d'écrire des additions d'entiers plutôt qu'un simple entier.

1. Mise à jour du lexer (`lexer.rkt`).
2. Mise à jour du parser (`parser.rkt`) et des structures pour la syntaxe parsée (`ast.rkt`).
3. Mise à jour de l'analyse sémantique (`analyzer.rkt`) et des structures pour la représentation intermédiaire de l'assembleur MIPS (`ast.rkt`).
4. Mise à jour de l'écriture des instructions MIPS (`mips-printer.rkt`).

Exercice 2.

À vous de continuer, par couches successives allant d'un bout à l'autre, à améliorer le compilateur.

1. D'autres opérations arithmétiques.
2. Des variables.
3. Un type booléen.
4. Des opérations de comparaisons.
5. Des opérations logiques.
6. Etc.

Ce TP n'est pas à rendre.