

## TP de Compilation n° 0 : Compilation du langage jouet vu en cours

30 Janvier 2014

*Vous pouvez trouver tous les sujets au format électronique sur internet à l'adresse :*

<http://www.pps.univ-paris-diderot.fr/~pboutill/CompilM1/>.

Ce sujet propose d'implémentation un compilateur du langage jouet vu en cours. Implémentez la machine virtuelle jouet `Vm` égalément vu en cours.

Toutes les informations sur le langage source et la machine virtuelle cible sont dans le support de cours : <http://www.pps.univ-paris-diderot.fr/~mdio/Ens/Compilation/>

Vous pouvez prendre comme base la mise à jour de la sémantique d'Imp en ML disponible sur la même page. Pour implémenter ce compilateur nous vous conseillons de suivre les étapes suivantes :

### Exercice 1 :

1. Téléchargez `imp.ml`.
2. Lisez et comparez les types algébriques `exp`, `bexp`, `com`, `prog` avec le langage `Imp`.
3. Comparez le code de `eval_*` et `step` avec la formalisation de la sémantique à grand pas présentée pas vu en cours.
4. Exécutez le code dans un interprète OCaml.
5. Écrivez une fonction `print_mem` pour imprimer un valeur de type `mem`.
6. Modifiez les deux dernières lignes pour imprimer l'état mémoire.
7. Créez un exécutable en utilisant `ocamlbuild imp.native`, exécutez et comparez avec la précédente exécution.

### Exercice 2 :

1. Créez un fichier `vm.ml` où vous définirez les instructions assembleurs de la `Vm`, un imprimeur de code `Vm` ainsi qu'une fonction d'exécution de la machine virtuelle (vous devez compiler souvent avec `ocamlbuild vm.byte`). Vous pouvez faire appel aux fonctions définies dans `imp.ml` (`Imp.*`).
2. Définissez les fonctions de compilation de `Imp` vers `Vm`. Testez sur l'exemple de `imp.ml` et comparez avec le résultat des fonctions d'interprétation.