TP n°8

Manipuler du code-octet Java

Le but de ce TP est de manipuler le Java bytecode (JBC). Il y a plusieurs assembleur/disassembleur pour le JBC. Nous utilisons Krakatau v1 :

github.com/Storyyeller/Krakatau/tree/master

Pour l'installer, il suffit de télécharger l'archive `zip` et la décompresser (Attention, cet outil est en développement et peut être erroné). Cela crée un répertoire contenant des programmes `python2` pour produire du JBC (mis dans un fichier `.j`) à partir d'un fichier `.class` (`disassemble`) et l'inverse (`assemble`). Par exemple, prenons le fichier `HelloWorld.java` suivant :

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

et compilons le vers `HelloWorld.class`. En exécutant `disassemble.py HelloWorld.class` le fichier `HelloWorld.j` en JBC est produit :

```
.version 49 0
.class public super HelloWorld
.super java/lang/Object

.method public <init> : ()V
    .code stack 1 locals 1
L0:     aload_0
L1:     invokespecial Method java/lang/Object <init> ()V
L4:     return
L5:
        .linenumbertable
            L0 1
        .end linenumbertable
    .end code
.end method

.method public static main : ([Ljava/lang/String;)V
    .code stack 2 locals 1
L0:     getstatic Field java/lang/System out Ljava/io/PrintStream;
L3:     ldc 'Hello, World'
L5:     invokevirtual Method java/io/PrintStream println (Ljava/lang/String;)V
L8:     return
L9:
        .linenumbertable
            L0 4
            L8 5
        .end linenumbertable
    .end code
.end method
.sourcefile 'HelloWorld.java'
.end class
```

Vous pouvez comparer avec la sortie de `javap -c -verbose HelloWorld.class`. On observe entre autres que le "`Constant Pool`" est directement résolu dans `HelloWorld.j`.
Les exercices suivants consistent à écrire directement du JBC en modifiant le fichier `Exercice.j` (voir page Web, Attention : si vous utilisez un compilateur `java` récent pour produire du bytecode, la version sera supérieure à `49` et le bytecode est enrichi par certains informations de typage, notamment pour les sauts. Dans la suite, il faut veiller à ce que votre bytecode est bien en version `49` pour éviter des problèmes de typage). Pour tester que les solutions sont correctes, il faut exécuter `assemble.py Exercice.j` et ensuite `java Exercice`. Le fichier `Exercice.j` contient pour l'instant une méthode `f` qui renvoie l'entier `1` et une méthode `main` qui imprime la valeur renvoyée par `f`.
Attention : Dans `Exercice.j` la méthode `f` est declarée `static`. Du coup, les variables locales n'incluent pas `this`. Si on veut ajouter des arguments à `f` on peut commencer donc avec la variable locale `0` au lieu de la variable locale `1` dans les méthodes normales (non `static`).

Exercice 1. Modifiez successivement la méthode `f` pour quelle renvoie `2`, `42`, `1000`, `100000`.

Exercice 2. Écrivez successivement des méthodes qui renvoient

- `1.0`
- `2.0`

Indications : Il faut changer entre autres le type de retour de `f` et le type de l'appel à `println` dans `main`

Exercice 3. On considère le bytecode suivant (fichier entier `Optim.j` avec `main` sur la page Web) :

```
.method public static f : (I)I
    .code stack 2 locals 2
L0:     iconst_1
L1:     istore_1
L2:     iconst_2
L3:     iload_1
L4:     imul
L5:     iload_0
L6:     iadd
L7:     ireturn
    .end code
.end method
```

- Que fait le code de la méthode `f` ?
- Changez `stack 2` en `stack 1` et éxécutez (après `assemble`). Que se passe-t-il ?
- Pareil pour `locals 1` à la place de `locals 2`.
- Pareil pour `fconst_1` à la place de `iconst_1`.
- Optimisez le code en l'écrivant en quatre instructions.

Exercice 4. Écrivez une méthode qui renvoie la valeur ax^2+bx+c (où `a`, `b`, `c` et `x` sont des paramètres entiers `int`) ? Quelle est la hauteur maximale de la pile de calcul ? Écrivez ensuite la même méthode avec des paramètres `long`. Quelle est la hauteur maximale de la pile dans ce cas ?

Exercice 5. L'instruction JVM `newarray int` crée une référence vers un tableau d'entier de taille indiquée en haut de la pile. Écrivez une méthode `f` avec un argument entier `n` qui renvoie un tableau entier de taille `n`. Pour cela, complétez le fichier `Array.j`. Il a une méthode `main` qui appelle `f` avec `10` et qui imprime le deuxième élément du tableau.

Exercice 6. Écrivez une méthode `fact` qui prend un argument entier `n` et qui renvoie la factorielle de `n`.

Exercice 7. Écrivez une méthode `fib` prend un argument entier et qui renvoie la valeur de la fonction de Fibonacci.