Universität Karlsruhe (TH)

Lehrstuhl für Programmierparadigmen

Compilerpraktikum SS 2010 — http://pp.info.uni-karlsruhe.de/
Dozent: Prof. Dr.-Ing. G. Snelting — snelting@ipd.info.uni-karlsruhe.de
Betreuer: Matthias Braun — matthias.braun@kit.edu
Betreuer: Sebastian Buchwald — sebastian.buchwald@kit.edu

Übungsblatt 9 | Ausgabe: 23.06.2010 | Besprechung: 30.06.2010

Aufgabe 1: *Jasmin Assembler*

Die an Javabytecode angelehnte Assemblersprache Jasmin finden Sie unter `http://jasmin.sf.net`.

- Installieren Sie den Jasmin Assembler und lesen Sie sich in die Dokumentation ein. Die meisten Befehle sind relativ wenig dokumentiert, da sie sich exakt wie die gleichnamigen Befehle aus der *Java Virtual Machine Specification* verhalten. Sie müssen zum Einarbeiten folglich beide Dokumente verwenden.
- Schreiben Sie die Programme Opt1.java und Opt2.java als Assemblerprogram.
- Die Unterschiede zu Bytecode wurden schon teilweise angesprochen. Gibt es noch weitere Unterschiede?
- Finden Sie heraus wie man Jasmin am Besten in ihren bestehenden Compiler integrieren kann.

Aufgabe 2: *Firm nach Jasmin Assembler*

Ihr endgültiges Ziel wird das Erzeugen von Jasmin Assemblercode sein. Dazu sind einige vorbereitende Schritte notwendig.

- Erzeugen Sie passende Deskriptoren für alle Methoden und Felder in ihren Firmgraphen. Der Aufbau dieser Deskriptoren ist in der *Java Virtual Machine Specification* (Abschnitt 4.3) beschrieben.
- Erzeugen Sie Jasmin Assembler Code für alle Klassen, Methoden und Felder. Die Methodenrümpfe können Sie diese Woche noch leer lassen!
- Überlegen Sie sich wie Sie für einen einzelnen Block im Firmgraph Assembler Code erzeugen würden. Worin sehen Sie problematische Stellen?

```
class opt1 {
    public static void main(String[] args) {
        System.out.println(new opt1().foo(5) + new opt1().foo2(4));
    }

    public int foo(int l) {
        return 5 + (2−30);
    }

    public int foo2(int a) {
        return 2∗5∗(5−5)∗23;
    }

    public int foo3() {
        int x;
        x = 2 + 3;
        return x + x + 4;
    }
}
```

Abbildung 1: Opt1.java

```
class opt2 {
    public static void main(String[] args) {
        System.out.println(new opt2().foo(5));
        System.out.println(new opt2().foo2(42));
        System.out.println(new opt2().foo3(23, 5));
    }

    public int foo(int l) {
        return l−l;
    }

    public int foo2(int a) {
        return a+0;
    }

    public int foo3(int a, int b) {
        return a ∗ 1 − a + b;
    }
}
```

Abbildung 2: Opt2.java