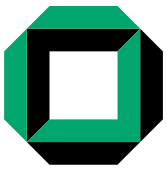

Universität Karlsruhe (TH)

Lehrstuhl für Programmierparadigmen

Compilerpraktikum SS 2009 — http://pp.info.uni-karlsruhe.de/
Dozent: Prof. Dr.-Ing. G. Snelting — snelting@ipd.info.uni-karlsruhe.de
Betreuer: Matthias Braun — braun@ipd.info.uni-karlsruhe.de
Betreuer: Jürgen Graf — graf@ipd.info.uni-karlsruhe.de

Übungsblatt 8 | Ausgabe: 2.07.2009 | Besprechung: 15.07.2009

Aufgabe 1: *Backend Rahmen*

- Für einige Typen werden Sie Java Deskriptoren erzeugen müssen. Am einfachsten erzeugt man diese beim Firmgraph Aufbau und benutzt Sie als Name für Firm-Typen.
- Schreiben Sie eine Funktion die für alle Klassen in ihrem Programm `emitClass(...)` aufruft.
- Implementieren Sie emitClass: Jasmin Code für den Header und Standardkonstruktor der Klasse soll ausgegeben werden. Danach `emitField(...)` und `emitMethod(...)` für alle Felder und Methoden aufgerufen werden.

Aufgabe 2: *Übersetzen von Methoden*

2.1 *Analysephase*

- Die Analyse durchläuft den Firm-Graph in Postorder (`Graph.walkPostorder`) und erstellen dabei Befehlslisten für jeden Grundblock.
- Bedenken Sie, dass Sprungbefehle stets als letzte in einem Grundblock ausgeführt werden müssen (dies ist nicht explizit über Datenabhängigkeiten gegeben).
- Markieren Sie Baumwurzeln und Teilen Sie Variablennummern zu wie auf den Praktikumsfolien vermerkt.
- Erstellen Sie für jeden Block eine Liste mit Φ-Instruktion im Nachfolgerblock.

2.2 *Code für einen Grundblock erzeugen*

- Implementieren Sie die Funktionen `pushValue` und `createValue` wie auf den Folien angedeutet.
- Implementieren Sie die Codeausgabe für Wurzelknoten (nutzen Sie die Reihenfolge die in den Befehlslisten vorgegeben wird).
- Erzeugen Sie die Store Befehle für die Φ-Instruktionen im Nachfolgerblock.
- Initial genügt es `.limit stack 100` zu benutzen.

2.3 *Verbesserungen (Optional)*

- Bei einer geschickten Anordnung der Grundblöcke kann man sich `goto` Befehle sparen falls von einem Block direkt in den nächsten gesprungen wird. Eine Postorder Reihenfolge genügt um eine statisch minimale Anzahl von `goto` Befehlen zu erzeugen.
- Geht die Blockanordnung sogar noch „besser"?
- Die maximale Höhe des Stacks wird im Moment mit 100 angegeben. Mit Hilfe einer Stacksimulation lässt Sie sich für einen Grundblock genau bestimmen. Zusätzliche könnte man die Methoden aus der Compiler 1 Vorlesung (Stichwort starkes Normalformtheorem) anwenden um die maximale Höhe möglichst niedrig zu halten.