Prof. Dr. Arnd Poetzsch-Heffter
Dr.-Ing. Jan Schäfer

Technische Universität Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Übungsblatt 9: Übersetzer und sprachverarbeitende Werkzeuge (SS 2011)

Hand Out: 22. Juni 2011
Hand In: 29. Juni 2011

Aufgabe 1 Prüfungsfragen zu den Vorlesungen

Überlegen Sie sich mindestens zwei Prüfungsfragen und tragen Sie diese unter `http://bit.ly/kJORSi` ein.

Aufgabe 2 Einfache Codegenerierung

Erweitern Sie Ihren Compiler um ein Backend zur Erzeugung von MIPS-Assembler. Gehen Sie dabei davon aus, dass Sie nur Eingaben behandeln, die aus einer Klasse mit einer statischen Methode (`Main.main`) bestehen. Diese Methode ruft sich **nicht** rekursiv selbst auf. Konzentrieren Sie sich auf die Übersetzung von Statements und Expressions.

Kopieren Sie sich dazu den Code aus dem `common/src/minijava/backend` Ordner des Subversion-Repositories in Ihren Gruppen-Ordner. Der Ordner enthält eine Katja-Spezifikation für abstrakte Syntax des MIPS-Assembler und Code, um diesen als Assemblercode auszugeben. Die Klasse `Backend` stellt Methoden bereit, die Sie aus Ihrem Compiler aufrufen, um das Backend auf einen MiniJava-AST anzuwenden.

In der Klasse `MIPSTranslation` fügen Sie ihren Code zur Übersetzung von Statements und Expressions an den dokumentierten Stellen ein.

Sie können den vorgegebenen Code auch ändern.

Hinweise:

- Sehen Sie den zur Verfügung gestellten Code erst durch, und lesen Sie die Kommentare, bevor Sie anfangen Ihren Code zu entwickeln.
- Der Assembler-Code soll in Dateien geschrieben werden, die bis auf die Endung, genauso heißen wie die zu übersetzenden Quelldateien. Verwenden Sie die Endung `.asm`. Code hierfür ist in der Klasse `Backend` zur Verfügung gestellt.
- Verwenden Sie MARS, um Ihre Compilerausgaben zu testen.

Aufgabe 3 MIPS-Assembler für Nicht-Strikte Ausdrücke

Übersetzen Sie nach dem Verfahren der Vorlesung die folgenden Statements mit nicht-strikten Boolschen Ausdrücken nach MIPS-Assembler. Übersetzen Sie das Statement Skip in die NOOP-Instruktion. Führen Sie **keine** Optimierung durch.

a) ItThenElse(Not(And(True, False)), Skip, Skip)

b) While (Or(False, True), Skip)

Aufgabe 4 MIPS-Assembler für Prozeduraufrufe

Übersetzen Sie, basierend auf dem Vorlesungswissen über Stackframes und Prozeduren, folgende MiniJava-Methode nach MIPS-Assembler von Hand. Schreiben Sie auch einen exemplarischen Aufruf dieser Methode in MIPS-Assembler.

```
static int test(int i, boolean j) {
  int result = 0;
  if (j) {
    result = i + 1;
  } else {
    result = result - i;
  }
  return result;
}
```

Formulieren Sie Ihren Pro- und Epilog so generisch, dass klar wird, wie er für andere MiniJava-Methoden aussehen müsste.

Aufgabe 5 Verschachtelte Prozeduren

Betrachten Sie folgende verschachtelte Prozeduren:

```
proc P
  var vp
  proc Q
    var vq
  begin
  end
  proc R
    var vr1
    var vr2
    proc S
    begin
      call Q
    end
  begin
   call S
  end
begin
  call R
end
```

a) Geben Sie zu jeder Prozedur die Verschachtelungstiefe, die sichtbaren Variablen, die aufrufbaren Prozeduren und den direkten statischen Vorgänger an.

b) Betrachten Sie folgende Aufrufkette: `P -> R -> S -> Q`. Geben Sie einen passenden Stack mit SPR-Referenzen an (wie das Beispiel auf Folie 115).