PHILIPPS-UNIVERSITÄT MARBURG D-35032 Marburg
Fachbereich Mathematik und Informatik Hans Meerwein Straße
Prof. Dr. R. Loogen
J.Berthold

19.Januar 2006

10. Übung zu „Grundlagen des Compilerbaus“, WS 2005/06

Abgabe schriftlicher Aufgaben: Do, 26.Januar 2006 (vor der Vorlesung)
Besprechung mündlicher Aufg.: ab 23.Januar 2006 in der Übung

Mündliche Aufgaben

10.1 **Codelängenberechnung**

Definieren Sie passend zu den Übersetzungsfunktionen der Vorlesung eine Funktion zur Bestimmung der Codelänge:

$$cl : AExp \cup BExp \cup Cmd \to \mathbb{N}$$

Erläutern Sie, wo und wie diese Funktion für die Übersetzung von PSA-Programmen in MA-Code zu verwenden ist.

10.2 **Jumping Code**

Für zusammengesetzte Boolesche Ausdrücke kann anstelle der Maschinenoperationen (AND, OR) sog. *Jumping Code* generiert werden, so dass unnötige Teile gar nicht erst ausgewertet werden (siehe Script S. 92).

Gegeben sei die Symboltabelle $st = st_\emptyset[I/(\text{var}, 1),\ J/(\text{var}, 2)]$
sowie die PSA-Anweisung

```
while (I > 0 and (J > I or J < 0)) do
        I := J - I;
        J := 1 - J
```

Übersetzen Sie diese PSA-Anweisung mit und ohne Verwendung von Jumping Code. Vergleichen Sie die Codelänge und die Zahl der ausgeführten Befehle für die Schleife bei verschiedenen Variablenbelegungen.

Schriftliche Aufgaben

10.3 **PSA-Erweiterung** 4 Punkte

Der Programmiersprache PSA soll eine Laufanweisung (`for`-Schleife) hinzugefügt werden, welche die folgenden syntaktischen und semantischen Eigenschaften hat:

$$\Gamma ::= \ldots \mid \textbf{for } I := E_1 \textbf{ to } E_2 \textbf{ step } E_3 \textbf{ do } \Gamma$$

Die Zählvariable soll innerhalb der Schleife implizit neu deklariert sein, überdeckt also vorhergehende Deklarationen des verwendeten Bezeichners. Textueller Gültigkeitsbereich der Deklaration sei der Schleifenrumpf.[1]

[1] Die hier beschriebene Konvention gilt für die **for**-Schleife in der Programmiersprache Ada.

Bitte wenden!

(a) Erweitern Sie die Übersetzungsfunktion *ct*, so dass für das neue Sprachkonstrukt korrekter Zwischencode im Sinne der oben beschriebenen Konvention erzeugt wird.

(b) Gegeben sei ein in obigem Sinne erweitertes PSA-Programm P:

```
var X, Y;
X := 42; Y := 1;
for X := 1 to 10 step 2 do Y := Y + X
```

Ermitteln Sie *trans(P)*.

10.4 PSA-Übersetzung mit Attributgrammatik

8 Punkte

Die auf Blatt 6 beschriebene While-Sprache ist der Sprache PSA sehr ähnlich. Die Ausgabe des in Aufgabe 6.3 erstellten Parsers kann als abstrakter Syntaxbaum eines PSA-Programms betrachtet werden, wenn die **print**-Anweisung weggelassen wird.

(a) Skizzieren Sie eine L-Attributierung der Grammatik aus Aufgabe 6.3, welche in einem synthetischen Attribut der Wurzel die Übersetzung des beschriebenen Programms in MA-Code erzeugt. Sie benötigen als weitere Attribute mindestens eine Symboltabelle sowie die Codelängenfunktion aus Aufgabe 10.1. / 3

(b) Implementieren Sie in Haskell einen Compiler, welcher ein gegebenes While-Programm in MA-Code übersetzt. Schreiben Sie dazu Haskell-Funktionen, welche aus einem gegebenen Syntaxbaum (gemäß Aufgabe 6.3) sowie ggf. inheriten Attributen die jeweils benötigten synthetischen Attribute berechnen. / 5

Testen Sie, ob Ihr Compiler korrekten Code erzeugt.

Auf der Webseite zur Vorlesung finden Sie eine Implementierung der MA-Maschine, mit der Sie den generierten MA-Code ausführen können, sowie bei Bedarf einen RD-Parser (Lösung zu Aufgabe 6.3).