PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen J.Berthold

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

27.Oktober 2003

Übungen zur „Theorie des Compilerbaus“, WS 2003/04

Nr. 1 , Abgabe und Besprechung: 3. November in der Übung

Hinweise: Mündliche Aufgaben sollen zum Abgabetermin vorbereitet werden und werden im Tutorium besprochen. Die Lösungen zu schriftlichen Aufgaben sollen grundsätzlich schriftlich abgegeben, Programme ausgedruckt und *zusätzlich* per EMail an den jeweiligen Tutor geschickt werden. Die Abgabe ist in Gruppen bis zu zwei Personen erlaubt.

Mündliche Aufgaben

1.1 **T-Diagramme**
Nehmen Sie an, daß Sie über folgendes verfügen:

- eine Rechnerplattform M,
- einen C-Compiler der auf M läuft und Maschinensprache für M generiert,
- ein Java-nach-C-Übersetzungsprogramm, das in C geschrieben ist.

Benutzen Sie die in der Vorlesung vorgestellten Diagramme, um die folgenden Vorgänge darzustellen:

a) Kompilation des Java-nach-C-Übersetzers in Maschinensprache
b) Entwicklung eines Java-nach-M-Übersetzers in Maschinensprache

1.2 **Entfernung von Whitespace**
In dieser Aufgabe soll ein Haskell-Programm geschrieben werden, das aus einer gegebenen Zeichenkette Reihen von Leerzeichen, Tabulatoren (`\t`) und Zeilenumbrüchen (`\n`) entfernt und durch jeweils ein Leerzeichen ersetzt.

a) Schreiben Sie zunächst eine Funktion

```
isWhiteSpace :: Char -> Bool
```

die erkennt, ob der Argumentbuchstabe zu den oben genannten Zeichen gehört.

b) Schreiben Sie nun eine Funktion

```
compress :: [Char] -> [Char]
```

die unter Benutzung von `isWhiteSpace` Ketten von mehr als einem Leerzeichen zu einem einzigen Leerzeichen reduziert.

Zum Beispiel:

```
"Emil und   die Detektive\n\n \t\t von\n Erich Kaestner    "
```

soll zu

```
"Emil und die Detektive von Erich Kaestner "
```

konvertiert werden.

Schriftliche Aufgaben

1.3 Konvertierungsfunktionen für römische Zahlen

8 Punkte

Im römischen System gibt es bekanntlich verschiedene grundlegende Zahlzeichen entsprechend der nebenstehenden Tabelle. Andere Zahlen werden als Verkettung dieser Grundzeichen ausgedrückt. Dabei sollen (hier) die folgenden *Regeln* gelten:

1	I
5	V
10	X
50	L
100	C
500	D
1000	M

- Die Zeichen `I,X,C,M` kommen höchstens dreimal *hintereinander* vor, die Zeichen `V,L,D` insgesamt nur einmal.
- Steht *rechts* von einem Zeichen ein Zeichen mit gleichem oder kleinerem Wert, so wird sein Wert addiert.
- Es darf *höchstens ein* Zeichen eines Werts *links* neben einem mit höherem Wert stehen. In diesem Fall wird sein Wert subtrahiert und das kleinere Zeichen darf nicht nochmal direkt rechts vom größeren stehen.

Beispiele: `LXXXIX`= 89 `CDIL`= 449 `MCMLXXVIII`= 1978 `VIV,XXXX,XLX` ungültig.

a) Definieren Sie eine Haskell-Funktion / 3

`fromRoman :: String -> Int`,

die korrekte römische Zahlen in Dezimalzahlen umwandelt.

b) Schreiben Sie eine Umkehrfunktion

`toRoman :: Int -> String`, / 3

die Zahlen aus dem Dezimalsystem in korrekte römische Zahlen umwandelt. Es soll gelten: $n =$ `fromRoman ( toRoman` n `)` $\forall n \in [1, 3999]$.

c) Gilt auch `s = toRoman ( fromRoman s)` für gültige römische Zahlen s? / 2
Sorgen Sie dafür, dass ungültige Strings eine verständliche Fehlermeldung erzeugen.

1.4 GNU-Compiler

4 Punkte

In GNU-Compilern wird eine plattformunabhängige Maschinensprache namens RTL als Zwischensprache verwendet. Es gibt Übersetzer, die Programme, die in Hochsprachen wie z.B. C, C++, Pascal etc. geschrieben sind, nach RTL übersetzen; analog gibt es Übersetzer, die RTL-Programme auf Plattformen wie z.B. Alpha, PPC, SPARC spezialisieren. Außerdem gibt es einen RTL-Optimierer, der ein RTL-Programm in ein effizienteres RTL-Programm transformiert. Alle diese Übersetzer sind in C implementiert.

a) Zeigen Sie, wie Sie diese Übersetzer auf einem SPARC-Rechner benutzen können, wenn Sie auf dem Rechner über einen C-Compiler verfügen.

Zeigen Sie nun, wie Sie auf einem SPARC-Rechner

(b) ein Pascal-Programm P in SPARC-Maschinensprache übersetzen können

(c) die Übersetzung von P mittels des Optimierers verbessern können

(d) ein C++-Programm Q in PPC-Maschinensprache übersetzen können.