



Compilerbau & Virtuelle Maschinen

1. Übungsblatt

Abgabe: 23. April 2007, bis 16 Uhr in der Vorlesung oder Raum 02.07.59

Aufgabe 1: Code-Erzeugung

3+3 Punkte

Gegeben seien folgende Anweisungsfolgen:

```
i)
x = 1;
y = 0;
while (y < m) {
    z = n - y;
    x = x * z;
    y = y + 1;
}
```

```
ii)
while (x != y) {
    if (x > y)
        x = x - y;
    else
        y = y - x;
}
```

- Was berechnen diese Anweisungsfolgen?
- Erzeugen Sie CMA-Code für diese Anweisungsfolgen.
Verwenden Sie dabei die Adressumgebung $\rho = \{n \mapsto 0, m \mapsto 1, x \mapsto 2, y \mapsto 3, z \mapsto 4\}$.

Aufgabe 2: Short circuit evaluation

4 Punkte

Seien b , e_1 und e_2 drei beliebige Ausdrücke.

- Ein $?$ -Ausdruck in C hat die Form $b ? e_1 : e_2$ und sein Wert ist e_1 falls $b \neq 0$ und e_2 falls $b = 0$. Geben Sie das CMA-Übersetzungsschema für $code_R(b ? e_1 : e_2) \rho$ an.
- Unter *short circuit evaluation* für boolesche Ausdrücke versteht man, dass das zweite Argument eines $||$ -Operators (*oder* in C) nicht mehr ausgewertet wird, wenn die Auswertung des ersten Arguments nicht 0 ergibt. Das erspart die überflüssige Auswertung des zweiten Arguments.
Geben Sie das CMA-Übersetzungsschema für $code_R(e_1 || e_2) \rho$ an, um *short circuit evaluation* zu realisieren.

Aufgabe 3: Continue-Anweisung

4 Punkte

Modifizieren Sie das Schema zur Übersetzung von Schleifen, so dass die *continue*-Anweisung zum sofortigen Sprung an das Ende des zugehörigen Schleifenrumpfes führt.

Tipp: Erweitern Sie dazu die Code-Erzeugungsfunktion um ein weiteres Argument l , welches das Sprungziel (label) beschreibt, zu dem bei einem *continue* gesprungen werden soll.