

Übungen zu „Übersetzerbau“

Blatt 11

Aufgabe 41. C-Construction hat Tiger um Konstanten und konstante Parameter erweitert:

```
let const N := 8
    type table = array of int
    const a := table [N] of 0
    function fac (const n : int) : int =
        if n = 0 then 1 else n * fac (n - 1)
in for i := 0 to N - 1 do
    a [i] := fac (i)
end
```

Präzisiere die Details der Spracherweiterung (Syntax und Semantik). Wie muss der Typechecker erweitert werden? Welche Optimierungsmöglichkeiten ergeben sich durch die neuen Konstrukte? (Eine naive Übersetzung behandelt Konstanten wie Variablen und ignoriert die **const** Annotationen von Parametern.)

Aufgabe 42. Als Alternative zu expliziten **const** Annotationen kann man den Übersetzer um eine entsprechende Analysephase erweitern. Entwerfe eine solche Programmanalyse. *Hinweis:* berücksichtige, dass Ausdrücke in Tiger auch Effekte haben können.

```
var debug := 0
var N := (if debug then print ("initializing N"); 256)
```

Aufgabe 43. Implementiere eine *optimale* Instruktionsauswahl für ‘:=’ Anweisungen der Zwischensprache. Verwende als Zielsprache die folgenden Instruktionen einer virtuellen RISC-Maschine.

add %r _i , %r _j , %r _k	$r_i \leftarrow r_j + r_k$
add %r _i , %r _j , c	$r_i \leftarrow r_j + c$
sub %r _i , %r _j , %r _k	$r_i \leftarrow r_j - r_k$
sub %r _i , %r _j , c	$r_i \leftarrow r_j - c$
mul %r _i , %r _j , %r _k	$r_i \leftarrow r_j * r_k$
div %r _i , %r _j , %r _k	$r_i \leftarrow r_j / r_k$
mov %r _i , c(%r _j)	$r_i \leftarrow [r_j + c]$
mov c(%r _i), %r _j	$[r_i + c] \leftarrow r_j$
mov (%r _i), (%r _j)	$[r_i] \leftarrow [r_j]$

Aufgabe 44. Implementiere einen Registerallokator für reine Ausdrücke. Gehe der Einfachheit halber davon aus, dass jeder Knoten eines Ausdrucks zu genau einer Maschineninstruktion korrespondiert. *Hinweis:* mache Gebrauch davon, dass Ausdrücke in beliebiger Reihenfolge ausgewertet werden können.