Technische Universität Berlin
Fachgebiet Übersetzerbau und Programmiersprachen
Prof. Dr. Peter Pepper · Sekr. TEL12-2 · www.uebb.tu-berlin.de

Compilerbau I — WS 2011/2012

Pepper · Lorenzen

Ausgabe: 23.01.2011
Abgabe: 05.02.2011

8. Übung

4. Meilenstein: Codegenerator für μOpal

Aufgabe 1. Coder

Im Rahmen des 4. und letzten Meilensteins soll nun ein Codegenerator für μOpal realisiert werden. Die Zielplattform der Übersetzung ist die virtuelle Stackmaschine μOM mit folgendem Instruktionssatz:

`LDPAR` n	Lade den n-ten Parameter auf den Stack.
`LDNAT` c	Lade die Konstante c auf den Stack.
`ADD`	Ersetze die beiden obersten Werte auf dem Stack durch das Resultat ihrer Addition. Setzt das `error`-Flag bei Überlauf (*overflow*).
`SUB`	Ersetze die beiden obersten Werte auf dem Stack durch das Resultat Subtraktion des obersten von dem darunterliegenden Wert. Setzt das `error`-Flag bei Unterlauf (*underflow*).
`MUL`	Ersetze die beiden obersten Werte auf dem Stack durch das Resultat ihrer Multiplikation. Setzt das `error`-Flag bei Überlauf (*overflow*).
`DIV`	Ersetze die beiden obersten Werte auf dem Stack durch das Resultat der Division des unteren durch den darüberliegenden Wert. Setzt das `error`-Flag bei Division durch 0.
`CALL` n	Rufe die n-te Funktion des Programms auf, wobei der erste Parameter zuunterst und der letzte Parameter zuoberst auf dem Stack liegt.
`RET`	Kehre von einem Funktionsaufruf zurück. Das Resultat der Funktion ist das oberste Element auf dem Stack.
`BRANCH mode offs`	Falls `mode` zutrifft, addiere `offs` zum Stand des Programmzählers *nach* dieser Instruktion. `mode` ist eine der folgenden Möglichkeiten: always — springe unbedingt, z — springe, falls das oberste Element auf dem Stack gleich 0 ist, nz — springe, falls das oberste Element auf dem Stack ungleich 0 ist, eq — springe, falls die beiden obersten Elemente auf dem Stack gleich sind, lt — springe, falls das oberste Element auf dem Stack größer als das darunterliegende ist, error — springe, falls das `error`-Flag gesetzt ist. Durch *mode* geforderte Elemente auf dem Stack werden entfernt.
`SWAP`	Vertausche die beiden obersten Elemente auf dem Stack.
`ABORT`	Breche die Programmausführung ab.

Ein μOM-Programm besteht aus:

- einer Funktionstabelle, in welcher Name, Anzahl der Parameter, und Codeoffset jeder Funktion enthalten sind. (Die `CALL` Instruktion greift über Indizierung auf diese Tabelle zu.)
- dem eigentlichen Code.

```
TYPE funEntry == funEntry(name:denotation, argc:nat, offset:nat)

TYPE opcode   == LDPAR(n:nat)
                 LDNAT(c:nat)
                 ...

TYPE program  == program(main:nat, funTable:array[funEntry], code:array[opcode])
```

Weitere Details finden sich in der Struktur `MOM` bzw. werden im Tutorium besprochen.