Karlsruher Institut für Technologie
Lehrstuhl für Programmierparadigmen
Sprachtechnologie und Compiler WS 2010/2011
Dozent: Prof. Dr.-Ing. G. Snelting
Übungsleiter: Matthias Braun `matthias.braun@kit.edu`

Übungsblatt 10 | Ausgabe: 25.6.2012 | Besprechung: 2.7.2012

Aufgabe 1: *Codeerzeugung*

Im Folgenden soll x86-Assembler erzeugt werden.

1.1 *Codegenerierung für Ausdrucksbäume*

Gegeben sei ein Ausschnitt aus der (abstrakten) Grammatik einer Programmiersprache:

Expression	→	*Expression* + *Expression*
Expression	→	*Expression* * *Expression*
Expression	→	*Expression* **xor** *Expression*
Expression	→	*Expression* **and** *Expression*
Expression	→	*Expression* **or** *Expression*
Expression	→	- *Expression*
Expression	→	**number**
Expression	→	**identifier**

Geben Sie Regeln zur Codegenerierung an, bei denen das Ergebnis eines Ausdrucks berechnet wird und sich danach in Register EAX befindet.

1.2 *Codegenerierung für Kontrollstrukturen*

Ein weiterer Ausschnitt aus der (abstrakten) Grammatik der Programmiersprache:

Statement	→	**if** (*Expression*) *Statement* **else** *Statement*
Statement	→	**while** (*Expression*) *Statement*
Statement	→	*Expression*

Geben Sie Regeln zur Codegenerierung für das if und das while Konstrukt an.

1.3 *Codegenerierung für Multiplikation mit Konstanten (Zusatzaufgabe)*

Auf vielen Maschinen benötigen Multiplikations- und Divisonsbefehle deutlich mehr Zeit als Addier- und Shift-Befehle.

- Wie kann man sich bei Arrays bei denen die Größe der Elemente ja bekannt ist die Multiplikationsbefehle sparen?
- Nehmen Sie an eine Multiplikation benötigt 5 Takte, alle anderen Befehle nur einen Takt. Wann lohnt sich dann ein Ersetzen der Multiplikationen?

1.4 *Codegenerierung für Switch*

Gegeben folgendes Programmstück (mit C oder Java Syntax+Semantik):

```
switch (var+2) {
    case 4: S1; break;
    case 5: S2; break;
    case 7: S3;
    case 8: S4; break;
    default: S5; break;
}
```

Geben Sie effizienten Code für das Programmstück an. Der Wert von var befinde sich in Register EAX. Für S1-S5 können im Maschinencode die Platzhalter A1-A5 benutzt werden.

Aufgabe 2: *Ershov-Zahlen*

Berechnen Sie Ershov-Zahlen für die folgenden Ausdrücke:

1. $a * (b + c) - d * (e + f)$
2. $a + b * (c * (d + e))$
3. $(-a + *p) * ((b - *q) * (-c + *r))$

2.1 *Codeerzeugung*

Erzeugen Sie für obige x86 Code der möglichst ohne Address-Mode [1] so wenig Register wie möglich verwendet.

Aufgabe 3: LR-*Parser zur Codegenerierung*

Anstatt Baumersetzungsverfahren kann man auch einen LR-Parser zur Erzeugung von Maschinencode nutzen. Dazu werden die Ausdrücke der Zwischensprache in einer normalisierten Präfixdarstellung als String dargestellt.

Die Codeerzeugung geschieht dann mit Hilfe eines LR-Parsers, dessen Aktionen bei der Komplettierung einer Regel das Ausgeben von Quelltext ist. Passende Regeln zu den Ersetzungen der Vorlesungsfolien sehen dann z.B. so aus:

1	R_i	$\rightarrow$	$\mathbf{c}_a$	{ LD Ri, #a }
2	R_i	$\rightarrow$	M_x	{ LD Ri, x }
3	M	$\rightarrow$	$=\ M_x\ R_i$	{ ST x, Ri }
4	M	$\rightarrow$	$=\ \mathbf{ind}\ R_i\ R_j$	{ ST *Ri, Rj }
5	R_i	$\rightarrow$	$\mathbf{ind} + \mathbf{c}_a\ R_j$	{ LD Ri, a(Rj) }
6	R_i	$\rightarrow$	$+\ R_i\ \mathbf{ind} + \mathbf{c}_a\ R_j$	{ ADD Ri, Ri, a(Rj) }
7	R_i	$\rightarrow$	$+\ R_i\ R_j$	{ ADD Ri, Ri, Rj }
Z1	R	$\rightarrow$	**sp**	
Z2	M	$\rightarrow$	**m**	

Das Terminal **m** steht dabei für einen bestimmten Ort im Arbeitsspeicher (z.B. den Platz einer globalen Variablen). Das Stackregister wird durch das Terminal **sp** gekennzeichnet. Das Terminal **c** steht für Konstanten.

Der Ausdruck $a[i] = b + 1$ sieht damit zum Beispiel so aus:

$$\texttt{= ind + +}\ \mathbf{c}_a\ \texttt{sp ind +}\ \mathbf{c}_i\ \texttt{sp +}\ M_b\ \mathbf{c}_1$$

3.1 *Anwendung*

[1] Zum laden der a,b,c,d,e,f,p,q Variablen und für die Dereferenzieroperation ist ohne Address-Mode natürlich nicht möglich und darf dort benutzt werden

- Wie sehen die Anweisungen aus der letzten Aufgabe in Präfixform aus?
- Schreiben Sie die Zusatzregeln aus der letzten Aufgabe sowie eine Regel zum indirekten Laden (LD Ri, Rj) als Grammatikregeln auf.
- Wenden Sie das Verfahren auf die Ausdrücke an.

3.2 *Regeln schreiben*

Erweitern Sie das Verfahren auf while-Anweisungen.