Universität Karlsruhe (TH)

Lehrstuhl für Programmierparadigmen

Sprachtechnologie und Compiler WS 2009/2010 http://pp.info.uni-karlsruhe.de/
Dozent: Prof. Dr.-Ing. G. Snelting snelting@ipd.info.uni-karlsruhe.de
Übungsleiter: Sebastian Buchwald sebastian.buchwald@kit.edu

Übungsblatt 10 | Ausgabe: 7.1.2010 | Besprechung: 20.1.2010

Aufgabe 1: *Codeerzeugung*

Gegeben Sei eine einfache RISC-Architektur. Der Einfachheit halber seien beliebig viele Register GP1, GP2, GP3, ... vorhanden. Bei Befehlen kann statt eines Registers das gelesen wird auch eine Konstante angegeben werden. Die folgenden Befehle werden unterstützt:

mov R1, R2	Kopiert den Inhalt von Register R2 in Register R1
jmp label	Springt an die Stelle "label"
jmp R1	Springt an die Stelle im Programm die im Register R1 steht
bz R1, label	springe an die Stelle "label" falls die Register R1 den Wert 0 hat
bnz R1, label	Springe an die Stelle "label" falls das Register R1 einen Wert ungleich 0 hat
slt R1, R2, R3	Setzt Register R1 auf 1, falls der Inhalt von Register R2 kleiner als der von R3 ist, sonst auf 0
seq R1, R2, R3	Setzt Register R1 auf 1, falls der Inhalt der Register R2 und R3 gleich ist, sonst auf 0
add R1, R2, R3	Addiert den Inhalt von R2 und R3 und schreibt das Ergebnis in Register R1
mul R1, R2, R3	Multipliziert den Inhalt von R2 und R3 und schreibt das Ergebnis in Register R1
shl R1, R2, R3	Die Bits in Register R2 werden um den Wert im Register R3 nach links verschoben, das Ergebnis wird in R1 geschrieben
ld R1, R2	Lädt 4 Byte aus dem Hauptspeicher von Adresse R2 und speichert das Ergebnis in Register R1
st R1, R2	Schreibt 4 Byte des Wertes von R2 in den Hauptspeicher an Adresse R1
ldb R1, R2	Lädt 1 Byte aus dem Hauptspeicher von Adresse R2 und speichert das Ergebnis in Register R1
stb R1, R2	Schreibt 1 Byte des Wertes von R2 in den Hauptspeicher an Adresse R1
label:	Assembler Pseudobefehl um die Programmstelle mit dem Namen "label" zu versehen.
.long C	Assembler Pseudobefehl der den Wert der Konstante C ins Programm einsetzt (für C kann auch das Label einer Programmstelle angegeben werden)

1.1 *Codegenerierung für Ausdrucksbäume*

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

$$
\begin{aligned}
\textit{Expression} &\rightarrow \textit{Expression} + \textit{Expression} \\
\textit{Expression} &\rightarrow \textit{Expression} * \textit{Expression} \\
\textit{Expression} &\rightarrow \textit{Expression} < \textit{Expression} \\
\textit{Expression} &\rightarrow \textit{Expression} == \textit{Expression} \\
\textit{Expression} &\rightarrow \textbf{not } \textit{Expression} \\
\textit{Expression} &\rightarrow \textit{Expression } \textbf{and } \textit{Expression} \\
\textit{Expression} &\rightarrow \textit{Expression } \textbf{or } \textit{Expression} \\
\textit{Expression} &\rightarrow \textbf{number} \\
\textit{Expression} &\rightarrow \textbf{identifier}
\end{aligned}
$$

Geben Sie Regeln zur Codegenerierung an, bei denen das Ergebnis eines Ausdrucks berechnet wird und sich

danach in Register GP1 befindet.

1.2 *Codegenerierung für Kontrollstrukturen*

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

$$
\begin{array}{lcl}
\textit{Statement} & \rightarrow & \textbf{if} \ (\ \textit{Expression}\ )\ \textit{Statement}\ \textbf{else}\ \textit{Statement} \\
\textit{Statement} & \rightarrow & \textbf{while}\ (\ \textit{Expression}\ )\ \textit{Statement} \\
\textit{Statement} & \rightarrow & \textit{Expression}
\end{array}
$$

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

1.3 *Codegenerierung für Arrays 1*

Gegeben ein eindimensionales Array a von 0 bis 100 mit 1 Byte großen Werten.

- Geben Sie Code an, der die Elemente 0, 4 und 5 aus einem Array an Speicheradresse 1243 in die Register GP1, GP2 und GP3 liest.
- Geben Sie Code an, der auf Element 3 eines Arrays schreibt dessen Speicheradresse in Register GP1 steht. Der zu schreibende Wert befindet sich im Register GP2.

1.4 *Codegenerierung für Arrays 2*

Gegeben ein Array a dessen erste Dimension im Bereich 5 bis 20, dessen zweite Dimension im Bereich -10 bis 5 und dessen dritte Dimension im Bereich 0 bis 100 liegt. Das Array enthalte 4 Byte breite Integer Werte.

Geben Sie Code an, der das Element $a[i_1, i_2, i_3]$ liest und nach GP1 schreibt. Die Basisadresse von a befinde sich im Register GP2, GP3-GP5 enthalten die Werte i_1-i_3.

1.5 *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?

1.6 *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 GP4. Für S1-S5 können im Maschinencode die Platzhalter A1-A5 benutzt werden.

Aufgabe 2: *Speicherabbildung*

- Welchen Zweck haben die Funktionen f und g im Algorithmus zur Speicherabbildung (Folien Seite 41-42)? Wie könnten diese aussehen?
- Wende den Algorithmus zur Bestimmung der Offsets aller Elemente in den folgenden Datenstrukturen:

```
struct s1 {
    int     a;
    char    c;
    struct s2 {
        int c;
        union k {
            int l;
            struct s3 {
                char   c;
                double dd;
                char   f;
            } sk;
        } u;
    } s;
};
```

- Wende den Algorithmus zur Bestimmung von Frame-Offsets aller lokalen Variablen in folgendem Programm:

```
int f(void)
{
    int a;
    int b = 14;

    if (rand() % 2 != 0) {
        int buffer[200];
        snprintf(buffer, sizeof(buffer), "%d%d%d", rand(), rand(), rand());
        fputs(buffer, stderr);
        a = b = 0;
    } else {
        int f = 42;
        int buffer[200];
        snprintf(buffer, sizeof(buffer), "%d%d%d", rand(), rand()-f, rand()+f);
        fputs(buffer, stderr);
        a = f - b;
    }

    return a + b;
}
```