Universität des Saarlandes
FR 6.2 - Informatik
Prof. Dr. W.J. Paul
Dipl.-Ing. Christoph Baumann

11. Übungsblatt Informatik II

(Abgabe: 14.04.2008)

Aufgabe 1: (r-consis, return) (7 Punkte)

In der Definition der Stack-Konsistenz $r - consis$ wurde über die i-te *return*-Anweisung im C_0-Programmrest argumentiert. Erstelle eine formale Definition der Funktion $ithreturn(c, i)$, die für $i \in [1 : c.rd - 1]$ die i-te *return*-Anweisung in $c.pr$ zurückgibt!
Hinweis: Du darfst dir wenn nötig Hilfsfunktionen definieren.

Aufgabe 2: (Aho-Ullmann-Algorithmus) (10+10 Punkte)

Für den Aho-Ullmann Algorithmus stehen die folgenden Funktionen zur Verfügung:

`set(i)`	: Setze eine Marke auf Knoten i
`move(i,j)`	: Verschiebe eine Marke von Knoten i nach Knoten j
`remove(i)`	: Lösche Marke auf Knoten i

a) Wende den Aho-Ullmann Algorithmus mit Hilfe der gegebenen Funktionen auf folgende Graphen an!

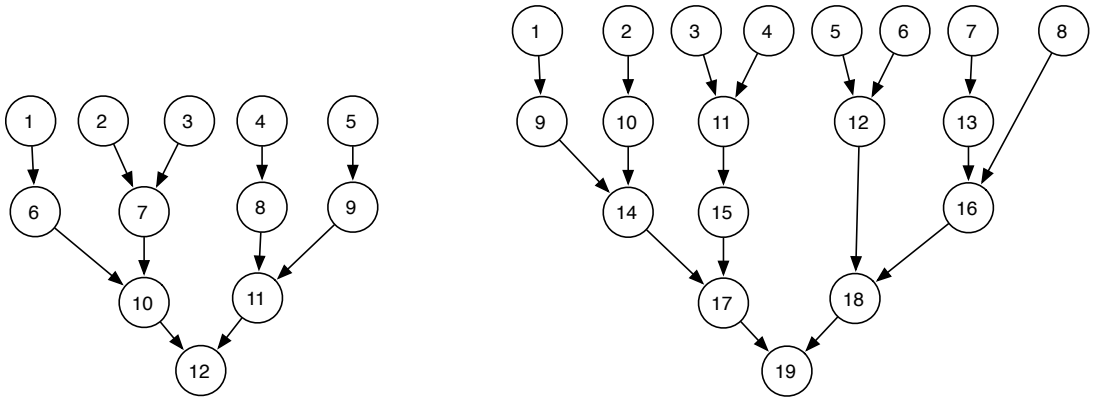

Abbildung 1: Tree1 / Tree2

b) Die Variablen der folgenden Ausdrücke seien alle im Speicher vorhanden. Wieviele Register werden vom C0-Compiler benötigt, um die Ausdrücke auszuwerten? Gib eine mögliche Reihenfolge der Auswertung durch Anwendung des Aho-Ullmann-Algorithmus an.

- $-(((-x_1 + (-x_2)) * (-(-x_3 - x_4) * (-x_5)))/(x_6 * (-x_7 + x_8)))$
- $-((a + b) * (c + d))/((e - f) * (g - h) - x)$

Aufgabe 3: (Ausdrucksübersetzung, Konstanten) **(8 Punkte)**

In der Vorlesung wurde die C_0-Ausdrucksübersetzung behandelt. Für die Zuweisung $u = c$ einer 32-Bit Konstante $c = \langle b \rangle, b \in \{0,1\}^{32}$ führt der Compiler folgende Assembler-Befehle aus, um c in Register j zu laden.

ecode:

- `lhgi` j $b[31:16] \oplus b[15]$
- `xori` j j $b[15:0]$

Wir betrachten die DLX-Konfigurationen d_0 und d_2, für die $d_0 \longrightarrow_{ecode} d_2$ gilt. Beweise:

$$d_2.gpr(j) = b$$

Aufgabe 4: (while-Schleife) **(5 Punkte)**

In der Vorlesung wurde dargestellt, wie der Compiler C_0-Anweisungen nach DLX-Assembler übersetzt. Gib den entsprechenden Code für folgende Anweisung mit Ausdruck e und Anweisungsfolge a an!

$$while\ e\ do\ \{a\}$$

Aufgabe 5 BONUS: (Schnelle Software-Multiplikation) **(3*+12* Punkte)**

In der Vorlesung wurde definiert, wie Ausdrücke vom Compiler rekursiv übersetzt werden. Ist der Ausdruck ein Arrayzugriff $a[l]$ so erzeugt der Compiler Code, mit folgender Semantik:

$$d^t.gpr(j) = d^{t-1}.gpr(k) + 4 * d^{t-1}.gpr(l) * size(t')$$

Da $d^{t-1}.gpr(l)$ erst zur Laufzeit bekannt ist und unsere Hardware keine Multikplikation zur Verfügung stellt, muss der obige Wert zunächst durch Software-Multiplikation errechnet werden.

a) Wie kann eine Multiplikation mit Zweierpotenzen 2^n, $n \in \mathbb{N}$ effizient implementiert werden, angenommen der andere Faktor stehe zu Beginn in Register $R1$? Gib den entsprechenden Assembler-Code an! Das Ergebnis der Multiplikation soll am Ende in Register $R1$ stehen. Die Multiplikation mit 2^{16} kann sogar nur mit einer einzigen Instruktion ausgeführt werden. Wie lautet der dazu benötigte Assemblerbefehl?

b) Gib ein Assembler-Programm zur Multiplikation zweier Binärzahlen a, b an! Die Laufzeit soll in $\mathcal{O}(\log(\langle b \rangle))$ liegen. Weiterhin stehen a und b anfangs in $R1$ bzw. $R2$. Das Produkt soll in $R3$ gespeichert werden. Man darf $\langle a \rangle, \langle b \rangle < 2^{16}$ annehmen.

Die Lösung zu Teilaufgabe b) ist elektronisch zu erstellen und per Email fristgerecht bei deinem Übungsleiter einzureichen. Unkommentierte Programme werden nicht bewertet.