P. Thiemann, S. Wehr Wintersemester 2006/2007

Compilerbau

`http://proglang.informatik.uni-freiburg.de/teaching/compilerbau/2006ws/`

Übungsblatt 11

24.1.2006

Achtung: Vorlesung und Übung fallen in der Woche vom 29.1.2007 bis zum 2.2.2007 aus. Als Ersatz geht die Vorlesung am Mittwoch ab sofort bis 18 Uhr; die Übung findet dann im Anschluss daran statt.

Aufgabe 1 ("Hello World" in MIPS–Assembler; 2 Punkte)
SPIM (`http://www.cs.wisc.edu/~larus/spim.html`) ist ein Simulator für MIPS–Assembler. Neben dem Simulator findet sich auf der Homepage auch eine gut lesbare Dokumentation zu MIPS–Assembler (`http://www.cs.wisc.edu/~larus/HP_AppA.pdf`). Programmieren sie in MIPS–Assembler ein Programm, welches einen String s von der Standardeingabe einliest und dann `Hello` s auf der Standardausgabe ausgibt. Speichern sie ihr Programm in einer Datei mit Namen `hello_world.asm`.

Aufgabe 2 (Ackermann Funktion in MIPS–Assember; 6 Punkte)
Implementieren sie die Ackermann Funktion a in MIPS–Assembler. Dabei ist a wie folgt definiert:

$$a(n,m) = \begin{cases} m+1 & \text{falls } n \leq 0, \\ a(n-1,1) & \text{falls } m \leq 0, \\ a(n-1, a(n, m-1)) & \text{sonst.} \end{cases}$$

Das Programm soll zunächst die beiden Eingabezahlen n und m von der Standardeingabe einlesen und dann das Ergebnis $a(n,m)$ auf der Standardausgabe ausgeben. Speichern sie ihr Programm in einer Datei namens `ack.asm`.

Aufgabe 3 (Optimale Codegenerierung; 4 Punkte)
IBurg (`http://www.cs.princeton.edu/software/iburg/`) ist ein Codegeneratorgenerator für Trivial Term Templates, der Codegeneratoren in C erzeugt. Die folgende IBurg-Spezifikation gibt Baum-Templates mit Kosten für Kombinationen aus vier Instruktionen und Konstanten an:

```
%term MOVE=1 MEM=2 PLUS=3 NAME=4 CONST=6
%%
stm: MOVE(MEM(loc),reg) = 1 (4);

reg: PLUS(con,reg) = 2 (3);
reg: PLUS(reg,reg) = 3 (2);
reg: PLUS(MEM(loc),reg) = 4 (4);
reg: MEM(loc) = 5 (4);
reg: con = 6 (2);
```

```
loc: reg = 7;
loc: NAME = 8;
loc: PLUS(NAME,reg) = 9;

con: CONST = 10;
%%
```

Die erste Zeile spezifiert die Namen aller verwendeten Baumknoten-Labels. Die dann folgenden Regeln beginnen jeweils mit einem Nonterminal und enthalten weiter ein Baumpattern, ein Gleichheitszeichen (=), die Nummer einer assoziierten Aktion (die hier ignoriert werden kann) und optional die assoziierten Kosten (in Klammern). Falls keine Kosten angegeben sind, sind diese als 0 anzusehen.

Benutzen sie den in der Vorlesung vorgestellten Bottom-Up Ansatz mit dynamischer Programmierung, um eine global–optimale Überdeckung für den folgenden Ausdrucksbaum zu finden:

```
MOVE(MEM(NAME),
     PLUS(MEM(PLUS(NAME,MEM(NAME))),
          CONST))
```

Abgabe: 7.2.2007
Die Abgabe erfolgt bis zu Beginn der Übungsstunde. Einreichungen, die nicht den Abgabemodalitäten entsprechen, werden abgelehnt. Für Plagiate werden keine Punkte vergeben.

Abgabemodalitäten:

- Code muss per Email an die Adresse `wehr@informatik.uni-freiburg.de` geschickt werden. Zu diesem Blatt ist kein Papierabgabe nötig.
- Der Code muss in einem Archiv mit dem Namen `vorname_nachname.tar.gz` oder `vorname_nachname.zip` an die Email angehängt werden.
- Entpacken des Archivs muss ein einzelnes Verzeichnis mit dem Namen `vorname_nachname` liefern.
- Innerhalb dieses Verzeichnisses müssen die Dateien `hello_world.asm` (für Aufgabe 1) und `ack.asm` (für Aufgabe 2) enthalten sein.
- Alle Dateien müssen von SPIM akzeptiert werden sein; Teile einer Datei, welche nicht von SPIM akzeptiert werden, müssen auskommentiert sein. Abgaben, die SPIM nicht akzeptiert, werden nicht bewertet.
- Wenn eine Aufgabe von der Lösung ein gewisses Format verlangt (wie etwa einen festen Namen oder Typ für eine Funktion, eine vorgegebene Signatur für ein Modul etc.), so ist dieses Format zwingend einzuhalten.