Technische Universität München
Fakultät für Informatik
Prof. Dr. H. Seidl

Thomas Gawlitza
Alex Berlea
Peter Ziewer
Andrea Flexeder

Compilerbau & Abstrakte Maschinen

Sommersemester 2006

2. Übungsblatt *Abgabetermin: Mo, 8. Mai 2006 in der Vorlesung*

Für diese Übung soll die CMa des Visualisierung-Systems VAM verwendet werden, das unter `http://wwwseidl.in.tum.de/~ziewer/vam/` heruntergeladen werden kann. Die korrekte Dateiendung für CMa-Programme lautet `.cma`.

Aufgabe 1: *Code-Erzeugung* *7 Punkte*

Übersetze das folgende Programm:

```
int result;

int fibo (int n){
  int result;

  if (n<0) return -1;
  switch (n){
    case 0 : return 0;
    case 1 : return 1;
    default: result = fibo(n-1) + fibo(n-2);
             return result;
  }
}

int main(){
  int n;

  n = 5;
  result = fibo(n);
  return 0;
}
```

Abgabe des lauffähigen, kommentierten CMa-Programms per E-mail.

Aufgabe 2: *Extreme Pointer* *7 Punkte*

Zum Setzen des Extreme-Pointers EP während der Ausführung benötigt der Übersetzer den maximalen Zuwachs des Stacks. Dieser Wert kann während der Übersetzung (statisch) berechnet werden.

Definiere eine Funktion t, die einem Ausdruck e eine möglichst präzise obere Schranke $t(e)$ für die Anzahl der Stackzellen zuordnet, die für die Auswertug des Ausdrucks e benötigt werden.

a) Berechne $t(e)$ zunächst für die beiden Extremfälle
$e = a_1 + (a_2 + (\cdots + (a_{n-1} + a_n) \ldots))$ und $e = (\ldots((a_1 + a_2) + a_3) + \ldots) + a_n$
(a_i sind Konstanten oder einfache Variablen).

b) Berechne $t(e)$ für die im Skript auf Seite 26 ff angegebenen Ausdrücke.

c) Erweitere den Definitionsbereich der Funktion t auf C-Anweisungen, insbesondere **if**, **for** und **while**.

Aufgabe 3: *Blöcke* *6 Punkte*

Erweitere die Codeerzeugungsfunktion für Anweisungsfolgen um Blöcke. Berücksichtige dabei, daß Variablendeklarationen an beliebigen Stellen innerhalb von Blöcken vorkommen dürfen. In Blöcken deklarierte Variablen sollen von dem Punkt, an dem sie eingeführt werden, bis zum zugehörigen Blockende sichtbar sein.

Bei der Ausführung des folgenden Beispielprogramms wird erst `2` und anschließend `1` als Wert der Variablen `x` ausgegeben.

```
int x;
x = 1;
{
    int x;
    x = 2;
    write(x);
}
write(x);
```

Hinweis: Verwalte zusätzlich zur Adressumgebung die erste vergebbare Relativadresse.