LV „ Programmierung “
Prof. Dr. H. Vogler / Dipl.-Math. R. Vater

SS 11

Aufgabenblatt zur 8. Übung

Zeitraum: 30.05. bis 03.06.2011

Bitte beachten: Ab Mittwoch, dem 08.06.11, wird der Übungsbetrieb **mittwochs in der 1. DS** wegen extrem geringer Teilnehmerzahlen eingestellt.

1. Aufgabe: (AGS 13.4)

Übersetzen Sie nachfolgende C_1-Statements, die sich im Rumpf der Funktion g befinden, in entsprechenden baumstrukturierten AM_1-Kode. Falls notwendig, treffen Sie für die Übersetzung zweckmäßige Annahmen. Zwischenschritte brauchen Sie keine anzugeben.
Die aktuelle Symboltabelle ist:
$tab_{g+lDecl} = [f/(\text{proc}, 1), g/(\text{proc}, 2), d/(\text{var}, \text{global}, 1), x/(\text{var}, \text{global}, 2), y/(\text{var}, \text{lokal}, 1), z/(\text{var-ref}, -2)]$.

```
...
if (d == 0) f(x); else {x=x+y; g(&y);}
printf("%d", x);
...
```

2. Aufgabe: (AGS 13.10)

(a) Übersetzen Sie das nachfolgende C_1-Statement, das sich im Rumpf der Funktion f befindet, in entsprechenden baumstrukturierten AM_1-Code. Nehmen Sie an, `if...else` sei das vierte Statement in f. Zwischenschritte brauchen Sie keine anzugeben.
Die aktuelle Symboltabelle ist:
$tab_{f+lDecl} = [g/(\text{proc}, 1), f/(\text{proc}, 2), a/(\text{var}, \text{global}, 1), b/(\text{var}, \text{lokal}, 1), c/(\text{var}, \text{lokal}, 2),$
$x/(\text{var}, \text{lokal}, -3), y/(\text{var-ref}, -2)]$.

```
...
if (a > 1) f(b,y); else {c=x; g(&b);}
...
```

(b) Gegeben sei folgender AM_1-Code:

```
 1: INIT 1;
 2: CALL 12;
 3: JMP 0;
 4: INIT 2;
 5: LOADI(-2);
 6: STORE(global,1);
 7: LIT 3;
 8: STORE(lokal,1);
 9: LOAD(global,1);
10: STORE(lokal,2);
11: RET 1;
12: INIT 1;
13: READ(lokal,1);
14: LOADA(lokal,1);
15: PUSH;
16: CALL 4;
17: WRITE(global,1);
18: RET 0;
```

Betrachten Sie nun die AM_1, die sich bereits im Zustand (Konfiguration)

$$\sigma = (13,\ \varepsilon,\ 0 : 3 : 0 : 0,\ 3,\ 8,\ \varepsilon)$$

befindet. Lassen Sie die AM_1, beginnend mit σ, solange ablaufen, bis die Maschine stoppt. Dokumentieren Sie den Zustand der AM_1 nach Ausführung jedes Befehls.

3. Aufgabe: (AGS 13.12)

(a) Folgendes Fragment eines C_1-Programms sei bekannt:

```
#include <stdio.h>

int x;

void h(...)
{...}

void g(...)
{...}

void f(int a; int *b)
{ int c;
    if (x>1) g(b); else h(a,&x); c=*b + 1;
}

void main()
{...}
```

Übersetzen Sie die Sequenz der Statements im Rumpf von `f` in entsprechenden AM_1-Code mit baumstrukturierten Adressen (mittels *stseqtrans*). Sie brauchen keine Zwischenschritte anzugeben.

Geben Sie zunächst die dazu benötigte Symboltabelle $tab_{f+lDecl}$ an.

(b) Gegeben sei folgender AM_1-Code:

1:	`INIT 1;`	6:	`LOADI(-2);`	11:	`READ(lokal,1);`	16:	`PUSH;`
2:	`CALL 10;`	7:	`ADD;`	12:	`READ(global,1);`	17:	`CALL 4;`
3:	`JMP 0;`	8:	`STORE(global,1);`	13:	`LOAD(lokal,1)`	18:	`WRITE(global,1);`
4:	`INIT 1;`	9:	`RET 2;`	14:	`PUSH;`	19:	`RET 0;`
5:	`LOAD(lokal,-3);`	10:	`INIT 1;`	15:	`LOADA(global,1);`		

Betrachten Sie nun die AM_1, die sich bereits im Zustand (Konfiguration)

$$\sigma = (12,\ \varepsilon,\ 0:3:0:7,\ 3,\ 5,\ \varepsilon)$$

befindet. Lassen Sie die AM_1, beginnend mit σ, auf dem oben gegebenen AM_1-Code solange ablaufen, bis die Maschine stoppt. Dokumentieren Sie den Zustand der AM_1 nach Ausführung jedes Befehls.

Zusatzaufgabe: (AGS 13.1*)

Gegeben sei folgendes C_1-Programm:

```
/* Exp */
#include <stdio.h>
int b;

void f(int a, int *b) {
  int c;
  c = a;
```

```
  while (c > 0) {
    *b = *b * 2;
    c = c - 1;
  }
}

void main() {
  int a;
  scanf("%i", &a);
  b = 1;
  f(a, &b);
  printf("%d", b);
}
```

(a) Berechnen Sie schrittweise das baumstrukturierte Programm $bExp_1 = trans(Exp)$ mit Hilfe der in der Vorlesung und im Skript angegebenen Übersetzungsfunktionen.

(b) Wandeln Sie $bExp_1$ in ein Programm Exp_1 mit linearisierten Adressen um, und berechnen Sie die operationelle Semantik $\mathcal{P}[\![Exp_1]\!](1)$. Dokumentieren Sie den Zustand der AM_1 nach Ausführung jedes Befehls.

(c) Geben Sie den Aufbau des Laufzeitkellers beim zweiten Erreichen der Befehlszähleradresse 7 ähnlich den Darstellungen im Vorlesungsskript an.
Markieren Sie dabei insbesondere die Aktivierungsblöcke sowie den Referenzzeiger REF.