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

SS 10

Aufgabenblatt zur 9. Übung

Zeitraum: 21.06. bis 25.06.2010

1. Aufgabe: (AGS 13.8)

(a) Übersetzen Sie das nachfolgende C_1-Statement, das sich im Rumpf der Funktion h befindet, in entsprechenden baumstrukturierten AM_1-Code. Falls notwendig, treffen Sie für die Übersetzung eine zweckmäßige Annahme. Zwischenschritte brauchen Sie keine anzugegeben.
Die aktuelle Symboltabelle ist:
$tab_h = [f/(\text{proc}, 1), g/(\text{proc}, 2), h/(\text{proc}, 3), a/(\text{var}, \text{global}, 1), b/(\text{var}, \text{global}, 2), d/(\text{var}, \text{lokal}, 1)]$.

```
...
if (a<b) f(d, &a);
else if (b<d) {a=d; g(b, &d);}
...
```

(b) Gegeben sei folgender AM_1-Code:

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

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

$$\sigma = (14,\ 5,\ 0:5:3:0:0:0,\ 4,\ 3,\ \varepsilon)$$

befindet. Lassen Sie die AM_1, beginnend mit σ, ablaufen, bis der Befehlszähler einen Wert ≥ 18 erreicht hat. Dokumentieren Sie den Zustand der AM_1 nach Ausführung jedes Befehls.

(c) Stellen Sie den Aufbau des Laufzeitkellers für den Zustand σ graphisch dar. Markieren Sie dabei die Aktivierungsblöcke und die maßgeblichen Abhängigkeiten innerhalb des Laufzeitkellers.

2. Aufgabe: (Klausuraufgabe 02.2010)

(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 an (in den Übungen wurde diese mit $tab_{f+lDecl}$ bezeichnet).

(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.

3. 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:
$tab = [f/(\text{proc}, 1), g/(\text{proc}, 2), d/(\text{var}, \text{global}, 1), x/(\text{var}, \text{global}, 2), y/(\text{var}, \text{lokal}, 1), z/(\text{var-ref}, -2)]$.

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

Zusatzaufgabe: (AGS 13.11*)

(a) Gegeben sei folgendes Fragment eines C_1-Programms.

```
#include <stdio.h>

int a, b;

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

```
void f(int *y)
{ int x;
  while (a == 0) { a=a-x; g(b, &x);}
  f(y);
}

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

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

Geben Sie zunächst die dazu benötigte Symboltabelle an.

(b) Gegeben sei folgender AM_1-Code:

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

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

$$\sigma = (12,\ \varepsilon,\ 0:0:3:0:0:8,\ 4,\ 6,\ \varepsilon)$$

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.