Aufgabenblatt zur 9. Übung

Zeitraum: 15.06. bis 19.06.2009

1. Aufgabe: (AGS 13.5 a)

(a) In einem C_1-Programm gibt es neben `main` noch die Funktionen `g` und `f`. Die Variable `x` sei die einzige globale Variable; in `main`, `g` und `f` gibt es jeweils genau eine lokale Variable.
Übersetzen Sie nachfolgende C_1-Statements, die sich im Rumpf der Funktion `main` befinden, in entsprechenden baumstrukturierten AM_1-Code. Falls notwendig, treffen Sie für die Übersetzung zweckmäßige Annahmen. Zwischenschritte brauchen Sie keine anzugeben.
Geben Sie zunächst die von Ihnen genutzte Symboltabelle $tab_{main+lDecl}$ an.

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

2. Aufgabe:

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

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 13.6*)

(a) Übersetzen Sie die beiden nachfolgenden C_1-Statements, die sich im Rumpf der Funktion f befinden, in entsprechenden baumstrukturierten AM_1-Code. Falls notwendig, treffen Sie für die Übersetzung eine zweckmäßige Annahme. Zwischenschritte brauchen Sie keine anzugeben.
Die aktuelle Symboltabelle ist:
$tab_{f+lDecl} = [\texttt{f}/(proc, 1), \texttt{g}/(proc, 2), \texttt{a}/(var, global, 1), \texttt{b}/(var, global, 2), \texttt{c}/(var\text{-}ref, -2)]$.

```
...
while (a >= 0){ b=b+*c; g(a, &b);}
f(c);
...
```

(b) Gegeben sei folgender AM_1-Code:

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

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

$$\sigma = (10,\ \varepsilon,\ 0:3:0,\ 3,\ 5,\ \varepsilon)$$

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.

Zusatzaufgabe: (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);
...
```