Programmierung

Aufgabe 1 (AGS 14.14)

(a) Gegeben sei folgendes C_0-Programm.

```
1  #include <stdio.h>
2  int main()
3  {
4    int x1, x2;
5    scanf("%i", &x1);
6    scanf("%i", &x2);
7    while (x1 > 0){
8      x1 = x2 - x1;
9      if (x2 > x1)
10       x2 = x2 / 2;
11   }
12   printf("%d", x1);
13   return 0;
14 }
```

Übersetzen Sie das Programm mittels *trans* in AM_0-Code mit linearen Adressen. Geben Sie nur das Endergebnis der Übersetzung, keine Zwischenschritte an!

(b) Gegeben sei der folgende Ausschnitt aus einem AM_0-Programm.

```
3:  LOAD 2;     6:  JMC 14;     9:  LIT 2;     12:  STORE 2;
4:  LIT 5;      7:  LOAD 1;     10: MUL;       13:  JMP 3;
5:  LT;         8:  LOAD 2;     11: ADD;       14:  WRITE 1;
```

Erstellen Sie ein Ablaufsprotokoll für dieses Programmfragment, bis die AM_0 terminiert. Die Startkonfiguration ist $(7, \varepsilon, [1/3, 2/1], \varepsilon, \varepsilon)$.

Aufgabe 2 (AGS 15.17)

(a) Gegeben sei folgendes Fragment eines C_1-Programms mit den Funktionen `f` und `g`:

```
while (*p > i) { f(p); i = i + 1; }
p = &i;
```

Übersetzen Sie die Sequenz dieser Statements in entsprechenden AM_1-Code mit baumstrukturierten Adressen (mittels *stseqtrans*). Sie müssen keine Zwischenschritte angeben. Nehmen Sie an, die `while`-Anweisung sei das zweite Statement in `g`, und es sei

$$tab_{g+lDecl} = \{\texttt{f}/(\text{proc}, 1), \texttt{g}/(\text{proc}, 2), \texttt{i}/(\text{var}, \text{lokal}, 1), \texttt{p}/(\text{var-ref}, -2)\} .$$

(b) Gegeben sei folgender AM_1-Code:

```
1:  INIT 1;            5:  LIT 0;            9:  LOADI(-3);
2:  CALL 18;           6:  GT;               10: MUL;
3:  INIT 0;            7:  JMC 17;           11: STOREI(-3);
4:  LOAD(lokal,-2);    8:  LIT 2;            12: LOAD(lokal,-2);
```

```
13: LIT 1;              18: INIT 0;             23: PUSH;
14: SUB;                19: READ(global,1);     24: CALL 3;
15: STORE(lokal,-2);    20: LOADA(global,1);    25: WRITE(global,1);
16: JMP 4;              21: PUSH;               26: JMP 0;
17: RET 2;              22: LOAD(global,1);
```

Führen Sie 13 Schritte der AM_1 auf der Konfiguration $\sigma = (22, \varepsilon, 1:3:0:1, 3, \varepsilon, \varepsilon)$ aus.

Aufgabe 3 (AGS 15.16)

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

```
1  #include <stdio.h>                 8    int c;  c = 3;
2                                      9    if (c == *b)
3  int x, y;                          10     while (a > 0)
4                                     11       f(&a, b);
5  void f(...) {...}                  12 }
6                                     13
7  void g(int a, int *b) {            14 void main() {...}
```

Übersetzen Sie die Sequenz der Statements im Rumpf von g in entsprechenden AM_1-Code mit baumstrukturierten Adressen (mittels *stseqtrans*). Sie brauchen keine Zwischenschritte anzugeben. Geben Sie zunächst die benötigte Symboltabelle $tab_{g+lDecl}$ an.

(b) Gegeben sei folgender AM_1-Code:

```
1: INIT 1;              9: RET 2;              17: PUSH;
2: CALL 10;            10: INIT 0;             18: LOADA(global,1);
3: JMP 0;              11: READ(global,1);     19: PUSH;
4: INIT 1;             12: LOAD(global,1);     20: CALL 4;
5: LOAD(lokal,-3);     13: LIT 0;              21: JMP 12;
6: LIT 1;              14: GE;                 22: WRITE(global,1);
7: SUB;                15: JMC 22;             23: RET 0;
8: STOREI(-2);         16: LOAD(global,1);
```

Führen Sie 16 Schritte der AM_1 auf folgender Konfiguration aus:

$$\sigma = (7,\ 1:1,\ 1:3:0:1:1:21:3:0,\ 7,\ \varepsilon,\ \varepsilon)$$

Zusatzaufgabe 1 (AGS 12.4.26)

(a) Berechnen Sie die Normalform des untenstehenden λ-Terms, indem Sie ihn *schrittweise* reduzieren. Geben Sie dabei vor jedem Schritt für die relevanten Teilausdrücke die Mengen der gebunden bzw. frei vorkommenden Variablen an.

$$(\lambda xy.y(\lambda x.x))(y(\lambda x.x))z$$

(b) Gegeben seien der λ-Term

$$\begin{aligned}\langle G\rangle = (\lambda gxy.\langle ite\rangle(\langle iszero\rangle x\ )\ y \\ (\langle ite\rangle(\langle iszero\rangle(\langle pred\rangle x)) \\ (\langle mult\rangle\langle 2\rangle y) \\ (\langle mult\rangle(g\ (\langle pred\rangle x\ )\ y)(g\ (\langle pred\rangle(\langle pred\rangle x))\ (\langle succ\rangle y)))))\end{aligned}$$

und der Fixpunktkombinator $\langle Y\rangle = (\lambda z.((\lambda u.z(uu))(\lambda u.z(uu))))$. Geben Sie die durch $\langle Y\rangle\langle G\rangle$ beschriebene rekursive Funktion als Haskell-Funktion `g` an.

(c) Gegeben sei der λ-Term

$$\langle F\rangle = (\lambda fxyz.\langle ite\rangle(\langle iszero\rangle\ y)\ (\langle add\rangle\ x\ x)\ (\langle mult\rangle\ z\ (\ f\ (\langle succ\rangle x)\ (\langle pred\rangle y\ )\ z)))$$

Berechnen Sie die Normalform des Terms $\langle Y\rangle\langle F\rangle\langle 4\rangle\langle 1\rangle\langle 2\rangle$ unter Angabe geeigneter Zwischenschritte. Führen Sie im Rechenprozess zweckmäßige Abkürzungen der λ-Terme ein.