

Aufgabe 1 (AGS 15.17 b, 15.1 a)

(a) Gegeben sei folgender AM_1 -Code:

```

1:  INIT 1;           10:  MUL;           19:  READ(global,1);
2:  CALL 18;         11:  STOREI(-3);    20:  LOADA(global,1);
3:  INIT 0;          12:  LOAD(lokal,-2); 21:  PUSH;
4:  LOAD(lokal,-2);  13:  LIT 1;         22:  LOAD(global,1);
5:  LIT 0;           14:  SUB;           23:  PUSH;
6:  GT;              15:  STORE(lokal,-2); 24:  CALL 3;
7:  JMC 17;          16:  JMP 4;         25:  WRITE(global,1);
8:  LIT 2;           17:  RET 2;         26:  JMP 0;
9:  LOADI(-3);       18:  INIT 0;
    
```

Dokumentieren Sie 12 Schritte der AM_1 mit der Startkonfiguration $\sigma = (22, \epsilon, 1 : 3 : 0 : 1, 3, \epsilon, \epsilon)$.

(BZ ,	DK ,	Laufzeitkeller	REF	Inp	Out)
(22 ,	ϵ ,	1 : 3 : 0 : 1	3	ϵ ,	ϵ)
(23 ,	1 ,	1 : 3 : 0 : 1	3	ϵ ,	ϵ)
(24 ,	ϵ ,	1 : 3 : 0 : 1 : 1	3	ϵ ,	ϵ)
(3 ,	ϵ ,	1 : 3 : 0 : 1 : 1 : 25 : 3	7	ϵ ,	ϵ)
(4 ,	ϵ ,	1 : 3 : 0 : 1 : 1 : 25 : 3	7	ϵ ,	ϵ)
(5 ,	1 ,	1 : 3 : 0 : 1 : 1 : 25 : 3	7	ϵ ,	ϵ)
(6 ,	0 : 1 ,	1 : 3 : 0 : 1 : 1 : 25 : 3	7	ϵ ,	ϵ)
(7 ,	1 ,	1 : 3 : 0 : 1 : 1 : 25 : 3	7	ϵ ,	ϵ)
(8 ,	ϵ ,	1 : 3 : 0 : 1 : 1 : 25 : 3	7	ϵ ,	ϵ)
(9 ,	2 ,	1 : 3 : 0 : 1 : 1 : 25 : 3	7	ϵ ,	ϵ)
(10 ,	1 : 2 ,	1 : 3 : 0 : 1 : 1 : 25 : 3	7	ϵ ,	ϵ)
(11 ,	2 ,	1 : 3 : 0 : 1 : 1 : 25 : 3	7	ϵ ,	ϵ)
(12 ,	ϵ ,	2 : 3 : 0 : 1 : 1 : 25 : 3	7	ϵ ,	ϵ)
(13 ,	1 ,	2 : 3 : 0 : 1 : 1 : 25 : 3	7	ϵ ,	ϵ)

1 > 0 ?

1 : 2



(b) Gegeben sei folgendes C₁-Programm:

```

1 #include <stdio.h>           10 }
2 int b;                       11
3 void f(int a, int *b) {      12 void main() {
4     int c;                   13     int a;
5     c = a;                   14     scanf("%i", &a);
6     while (c > 0) {          15     b = 1;
7         *b = *b * 2;         16     f(a, &b);
8         c = c - 1;           17     printf("%d", b);
9     }                         18 }

```

activation record for f:

a	b	ra	par	c
-3	-2	-1	0	1

↑
REF

Übersetzen Sie das Programm mittels *trans* in ein AM₁-Programm mit Baumstrukturierten Adressen. Geben Sie zunächst die Symboltabellen tab_{main} und tab_f zur Übersetzung der Statements in den Funktionen *main* bzw. *f* mittels *stseqtrans* an. Geben Sie keine weiteren Zwischenschritte an.

$tab_{main} = [f / (proc, 1), b / (var, global, 1), a / (var, lokal, 1)]$

$tab_f = [f / (proc, 1), \quad (globale Variable wird durch lokale "überschrieben")$
 $a / (var, lokal, -3), b / (var-ref, -2), c / (var, lokal, 1)]$

INIT 1; (Speicher für globale Variable b)
 CALL 2; (Aufruf der main-Funktion)
 JMP 0; (Programmende, wenn main fertig)

// Übersetzung von f

```

1:   INIT 1;           (lok. Variable)
      LOAD (lokal, -3);
      STORE (lokal, 1);   } c = a
1.2.1: LOAD (lokal, 1);
      LIT 0;
      GT;                } c > 0
      JMC 1.2.2;
      LOADI (-2);
      LIT 2;
      MUL;
      STOREI (-2);      } *b = *b * 2
      LOAD (lokal, 1);
      LIT 1;
      SUB;
      STORE (lokal, 1); } c = c - 1
      JMP 1.2.1;

```

1.2.2: RET 2; (zwei Parameter-Speicherplätze von f befreien)

// Übersetzung von main

```

2:   INIT 1;           (lok. Variable)
      READ (lokal, 1)
      LIT 1;
      STORE (global, 1); } b = 1
      LOAD (lokal, 1);
      PUSH;
      LOADA (global, 1); } f(a, &b)
      PUSH;
      CALL 1;
      WRITE (global, 1);
      RET 0;

```

Aufgabe 2 (AGS 15.16 a, AGS 15.18 b)

(a) Gegeben sei folgendes Fragment eines C₁-Programms:

```

1 #include <stdio.h>
2
3 int x, y;
4
5 void f(...) {...}
6 void main() {...}

7 void g(int a, int *b) {
8     int c;
9     c = 3;
10    if (c == *b)
11        while (a > 0) f(&a, b);
12 }

```

kein INIT und RET benötigt

Übersetzen Sie die Sequenz der Statements im Rumpf von *g* in entsprechenden AM₁-Code mit baumstrukturierten Adressen (mittels *stseqtrans*). Sie brauchen keine Zwischenschritte anzugeben. Geben Sie zunächst die benötigte Symboltabelle *tab_g* an.

$$tab_g = [\text{\$} / (\text{proc}, 1), \text{g} / (\text{proc}, 2), \\ \text{x} / (\text{var}, \text{global}, 1), \text{y} / (\text{var}, \text{global}, 2), \\ \text{a} / (\text{var}, \text{lokal}, -3), \text{b} / (\text{var-ref}, -2), \text{c} / (\text{var}, \text{lokal}, 1)]$$

LIT 3;

STORE (lokal, 1);

LOAD (lokal, 1);

LOADI (-2);

EQ;

JMC 2.2.1;

} if-Bed.

Fkt. g → 2. Statement in g

2.2.2.1: LOAD (lokal, -3);

LIT 0;

GT;

JMC 2.2.2.2

} Schleifenbed.

LOADA (lokal, -3);

PUSH;

LOAD (lokal, -2);

PUSH;

CALL 1;

JMP 2.2.2.1;

} Schleifenrumpf

2.2.2.2:

2.2.1:

(b) Gegeben sei folgender AM_1 -Code:

```

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

```

Erstellen Sie ein Ablaufprotokoll der AM_1 , indem Sie sie schrittweise ablaufen lassen, bis die Maschine terminiert. Die Anfangskonfiguration sei $(14, \epsilon, 0 : 0 : 1, 3, 4, \epsilon)$. Sie müssen nur Zellen ausfüllen, deren Wert sich im Vergleich zur letzten Zeile geändert hat.

(PC,	DK,	Laufzeitkeller,	REF,	Inp,	Out)
(14,	ϵ ,	0 : 0 : 1 ,	3,	4,	ϵ)
(15,	ϵ ,	4 : 0 : 1 ,	3,	ϵ ,	ϵ)
(16,	1,	4 : 0 : 1 ,	3,	ϵ ,	ϵ)
(17,	ϵ ,	4 : 0 : 1 : 1 ,	3,	ϵ ,	ϵ)
(3,	ϵ ,	4 : 0 : 1 : 1 : 18 : 3 ,	6,	ϵ ,	ϵ)
(4,	ϵ ,	4 : 0 : 1 : <u>1</u> : 18 : 3 ,	6,	ϵ ,	ϵ)
(5,	4,	4 : 0 : 1 : 1 : 18 : 3 ,	6,	ϵ ,	ϵ)
(6,	2 : 4,	4 : 0 : 1 : 1 : 18 : 3 ,	6,	ϵ ,	ϵ)
(7,	4 > 2?	1,	6,	ϵ ,	ϵ)
(8,	ϵ ,	4 : 0 : 1 : <u>1</u> : 18 : 3 ,	6,	ϵ ,	ϵ)
(9,	4,	4 : 0 : 1 : 1 : 18 : 3 ,	6,	ϵ ,	ϵ)
(10,	2 : 4,	4 : 0 : 1 : 1 : 18 : 3 ,	6,	ϵ ,	ϵ)
(11,	4 / 2 =	4 : 0 : 1 : <u>1</u> : 18 : 3 ,	6,	ϵ ,	ϵ)
(12,	ϵ ,	2 : 0 : 1 : 1 : 18 : 3 ,	6,	ϵ ,	ϵ)
(18,	ϵ ,	2 : 0 : 1 ,	3,	ϵ ,	ϵ)
(19,	ϵ ,	2 : 0 : 1 ,	3,	ϵ ,	2)
(0,	ϵ ,	2 : 0 : 1 ,	3,	ϵ ,	2)