Aufgabe 1 (AGS 14.1 ⋆)

Gegeben sei folgendes C_0-Programm *Max*:

```
1  #include <stdio.h>
2
3  int main() {
4    int a, b, max;
5  1 scanf("%i", &a);
6  2 scanf("%i", &b);
7    if (a > b)
8  3   max = a;
9    else max = b;
10   printf("%d", max);
11   return 0;
12 }
```

(a) Berechnen Sie *schrittweise* das baumstrukturierte Programm $bMax_0 = \text{trans}(Max)$ mit Hilfe der in der Vorlesung angegebenen Übersetzungsfunktionen. Dokumentieren Sie dabei jeden rekursiven Funktionsaufruf.

trans(Max) = trans (#include <stdio.h> int main(){... ; return 0; })

= blocktrans ({ int a , b , max ; ... ; return 0 ; })

= stseqtrans (scanf("%i", &a) ; ... ; printf("%d", max) ,
update(int a,b, max ; tab$_\emptyset$),
1 ← Basisadresse

= stseqtrans (scanf("%i", &a) ; ... ; printf("%d", max) ,
tab_1 := $tab_\emptyset$ [a / (var, 1) , b / (var, 2) , max / (var, 3)],
1 ← Basisadresse

= sttrans (scanf(..., &a) , tab_1 , 1.1)
sttrans (scanf(..., &b), tab_1, 1.2)
sttrans (if (a>b) max = a else max = b , tab_1 , 1.3)
sttrans (printf(... , max) , tab_1, 1.4)

= READ 1 ;
READ 2 ;
boolexptrans(a>b , tab_1)
JMC 1.3.1 ;
sttrans(max = a , tab_1, 1.3.2)
JMP 1.3.3 ;

1.3.1 : sttrans(max = b , tab_1, 1.3.4)

1.3.3: WRITE 3 ;

=

READ 1 ;	JMC 1.3.1 ;	1.3.1:	LOAD 2 ;
READ 2 ;	LOAD 1 ;		STORE 3 ;
LOAD 1 ;	STORE 3 ;	1.3.3:	WRITE 3 ;
LOAD 2 ;	JMP 1.3.3 ;		
GT ;			

(b) Wandeln Sie $bMax_0$ in ein Programm Max_0 mit linearisierten Adressen um und berechnen Sie $\mathcal{P}[\![Max_0]\!](5:7)$. Dokumentieren Sie den Zustand der AM_0 nach jedem Ausgeführten Befehl.

```
1  READ 1;        6  JMC 10 ;       10 :  LOAD 2;
2  READ 2;        7  LOAD 1;        11    STORE 3;
3  LOAD 1;        8  STORE 3;       12 :  WRITE 3;
4  LOAD 2;        9  JMP 12 ;
5  GT;
```

BZ	DK	HS	Inp	Out
(1 ,	ε ,	[ ] ,	5:7 ,	ε)
(2 ,	ε ,	[1/5] ,	7 ,	ε)
(3 ,	ε ,	[1/5, 2/7] ,	ε ,	ε)
(4 ,	5 ,	[1/5, 2/7] ,	ε ,	ε)
(5 ,	7:5 ,	[1/5, 2/7] ,	ε ,	ε)
(6 ,	5>7? 0 ,	[1/5, 2/7] ,	ε ,	ε)
(10 ,	ε ,	[1/5, 2/7] ,	ε ,	ε)
(11 ,	7 ,	[1/5, 2/7] ,	ε ,	ε)
(12 ,	ε	[1/5, 2/7, 3/7] ,	ε ,	ε)
(13 ,	ε ,	[1/5, 2/7, 3/7] ,	ε ,	7)

Aufgabe 2 (AGS 14.14)

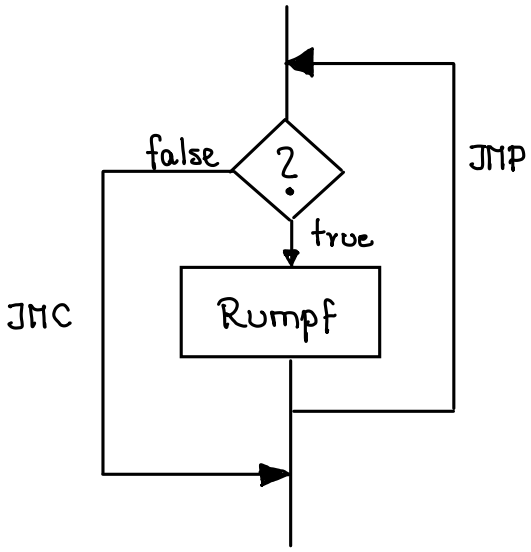
(a) Gegeben sei folgendes C_0-Programm.

```
1  #include <stdio.h>
2
3  int main() {
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!

```
1   READ 1;
2   READ 2;

3   LOAD 1;
4   LIT 0;
5   GT;
6   JMC 20;

7       LOAD 2;
8       LOAD 1;
9       SUB;
10      STORE 1;

11      LOAD 2;
12      LOAD 1;
13      GT;
14      JMC 19;

15          LOAD 2;
16          LIT 2;
17          DIV;
18          STORE 2;

19  JMP 3;
20  WRITE 1;
```

WHILE-Schleifen:



(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 $(7, \varepsilon, [1/3, 2/1], \varepsilon, \varepsilon)$.

Bz		DK	HS	Inp	Out
(7 ,		ε ,	[1/3, 2/1] ,	ε ,	ε)
(8 ,		3 ,	[1/3, 2/1] ,	ε ,	ε)
(9 ,		1:3 ,	[1/3, 2/1] ,	ε ,	ε)
(10 ,		2:1:3 ,	[1/3, 2/1] ,	ε ,	ε)
(11 ,	1*2	2:3 ,	[1/3, 2/1] ,	ε ,	ε)
(12 ,	3+2	5 ,	[1/3, 2/1] ,	ε ,	ε)
(13 ,		ε ,	[1/3, 2/5] ,	ε ,	ε)
(3 ,		ε ,	[1/3, 2/5] ,	ε ,	ε)
(4		5 ,	[1/3, 2/5] ,	ε ,	ε)
(5		5:5 ,	[1/3, 2/5] ,	ε ,	ε)
(6 ,	5<5 ?	0 ,	[1/3, 2/5] ,	ε ,	ε)
(14 ,		ε ,	[1/3, 2/5] ,	ε ,	ε)
(15 ,		ε ,	[1/3, 2/5] ,	ε ,	ε)