9. Übung - Programmierung

Wiederholung C_1, AM_1

SS 18

Übersetzung mit baumstrukturierten Adressen: $C_0 \leftrightarrow AM_0$

die Funktion *trans(...)*

trans("programmname")

= *trans*(#include ...)

= *blocktrans*({"deklarationen" "statements" return 0;})

= *stseqtrans*("statements"); *update*("declarationen"; $tab_\emptyset$;1)

= *sttrans*("statement";,tab_1;1.1)
sttrans("complexes statement";, tab_1;1.2)
sttrans("statement";, tab_1;1.3)

$\Rightarrow$ X.Y heißt: Falls im übersetzten Block eine Sprungadresse gebraucht wird ist die erste freie Adresse X.Y.1

$\Rightarrow$ übersetzen von statements in *simpletrans()*, *booltrans()* oder AM_0-Befehl

Aufgabe 1 a)

$=$ *sttrans*(scanf("%i", &x1),tab_1,1.1)
sttrans(scanf("%i", &x2),tab_1,1.2)
sttrans(while...;, tab_1,1.3)
sttrans(printf("%d", x1);,tab_1,1.4)

$=$ READ 1; READ 2;

1.3.1 *booltrans*($(x_1 > 0)$,tab_1)
JMC 1.3.2;
simpletrans($(x_1 = x_2 - x_1)$,tab_1)
stseqtrans(if ...,tab_1,1.3.3)
JMP 1.3.1;

1.3.2 WRITE 1;

Aufgabe 1 a)

$=$ READ 1; READ 2;
1.3.1 LOAD 1; LIT 0; GT;
JMC 1.3.2;
LOAD 2; LOAD 1; SUB; STORE 1;
stseqtrans(if ...,tab_1,1.3.3)
JMP 1.3.1;
1.3.2 WRITE 1;

Aufgabe 1 a)

=	READ 1; READ 2;	
1.3.1	LOAD 1; LIT 0; GT;	
	JMC 1.3.2;	
	LOAD 2; LOAD 1; SUB; STORE 1;	
	$booltrans(x_2 > x_1; tab_1)$;	Basisadresse war 1.3.3
	JMC 1.3.3.1;	
	$simpletrans(x_2 = x_2/2,\ tab_1)$;	
1.3.3.1	JMP 1.3.1;	
1.3.2	WRITE 1;	

Aufgabe 1 a)

```
=       READ 1; READ 2;
1.3.1   LOAD 1; LIT 0; GT;
        JMC 1.3.2;
        LOAD 2; LOAD 1; SUB; STORE 1;
        LOAD 2; LOAD 1; GT;
        JMC 1.3.3.1;
        LOAD 2; LIT 2; DIV; STORE 2;
1.3.3.1 JMP 1.3.1;
1.3.2   WRITE 1;
```

Aufgabe 1 a)

$\Rightarrow$ Durchnummerieren == Linearisieren

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;

Ablaufprotokoll: Aufgabe 1 b)

BZ	DK	HS	Inp	Out
7	ϵ	[1/3,2/1]	ϵ	ϵ
8	3			
9	1:3			
10	2:1:3			
11	2:3			
12	5			
13	ϵ	[1/3,2/5]		
3				
4	5			
5	5:5			
6	0			
14	ϵ			
15	ϵ	[1/3,2/5]		3

AM_1

$$AM_1 = BZ \times DK \times \mathbf{LK} \times \mathbf{REF} \times Inp \times Out$$

Befehle:

- Arithmetisch, Logisch, Sprungbefehle : wie C_0
- Adressierung: b $\in$ {global, local}, r: aktueller REF

$$adr(r, b, o) = \begin{cases} o + r, & \text{if b = lokal.} \\ o, & \text{otherwise.} \end{cases}$$

- Transport ($DK \leftrightarrow LK$):
  LOAD(b,o), STORE(b,o)
  ...lade/speichere x = Wert an der adr(r,b,o)
  LOADI(o), STOREI(o)
  ...lade/speichere den Wert an der Adresse x = Wert an adr(r,b,o)
  LOADA(b,o) ...schreibe adr(r,b,o) auf den LK

AM_1

Befehle:

- Prozedurbefehle:

  PUSH: oberstes Element vom DK auf den LK

  INIT n: n x "0" auf den LK

  CALL n: "Funktionsaufruf"

  1. BZ+1 auf den LK
  2. BZ auf n setzen
  3. REF auf den LK
  4. Neues REF = Länge(LK)

  RET n : "Funktionsende"

  1. Lösche alles nach REF vom LK
  2. oberstes Element vom LK in REF
  3. nächstes Element vom LK in BZ
  4. n Elemente vom LK löschen

AM_1

Befehle:

- Schreiben, Lesen ($LK \leftrightarrow Inp, Out$):
  READ(b, o) , WRITE(b,o) ...direkte Adresse
  READI o, WRITEI o... indirekte Adresse