8. Übung - Programmierung

C_1, AM_1

SS 18

Implementieren einer (imperativen) Programmiersprache

- Zielmaschine (abstrakte Maschine, Befehle, Befehlssemantik, Programmsemantik)
- Compiler zur Übersetzung der Programmier- in die Maschinensprache

$\Rightarrow$ Abstrakte Maschine AM_0 $\leftrightarrow$ Sprache C_0

C_0

Beschränkungen

- genau eine Funktion $\rightarrow$ `int main()`
- obligatorisches Einbinden von `stdio.h`
- Nur Typ `int` Variablen und Konstanten
- Kontrollstrukturen: Ein-/Ausgabe, Zuweisung, Sequenz, Verzweigung und bedingte Schleife

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 2 a)

$$
\begin{aligned}
& \mathit{trans}(\text{Max}) \\
&= \mathit{trans}(\#\text{include } <\text{stdio.h}> \text{ int main() } \{\ldots\text{return 0;}\}) \\
&= \mathit{blocktrans}(\{ \text{ int a,b,max; scanf}(\ldots \text{ return 0;}\}) \\
&= \mathit{stseqtrans}(\text{scanf}(\text{"\%i"}, \&\text{a})\ldots\text{printf}(\text{"\%d"}, \text{max});, \\
&\quad \mathit{update}(\text{ int a,b,max;}, \mathit{tab}_{\emptyset}), 1) \\
&= \mathit{stseqtrans}(\text{scanf}(\text{"\%i"}, \&\text{a})\ldots\text{printf}(\text{"\%d"}, \text{max});, \\
&\quad \underbrace{\mathit{tab}_{\emptyset} = [a/(\mathit{var}, 1), b/(\mathit{var}, 2), \mathit{max}/(\mathit{var}, 3)]}_{\text{tab1}}, 1) \\
&= \mathit{sttrans}(\text{scanf}(\text{"\%i"}, \&\text{a}), \mathit{tab}_1, 1.1) \\
&\quad \mathit{sttrans}(\text{scanf}(\text{"\%i"}, \&\text{b}), \mathit{tab}_1, 1.2) \\
&\quad \mathit{sttrans}(\text{if (a>b) max = a; else max = b;}, \mathit{tab}_1, 1.3) \\
&\quad \mathit{sttrans}(\text{printf}(\text{"\%d"}, \text{max});, \mathit{tab}_1, 1.4)
\end{aligned}
$$

Aufgabe 2 a)

$=$ *sttrans*(scanf("%i", &a),tab_1,1.1)
sttrans(scanf("%i", &b),tab_1,1.2)
sttrans(if (a>b) max = a; else max = b;, tab_1,1.3)
sttrans(printf("%d", max);,tab_1,1.4)

$=$ READ 1; *...1 ist Adresse von a laut* tab_1
READ 2;
booltrans((a>b),tab_1) ...if
JMC 1.3.1;
sttrans(max = a,tab_1,1.3.2) ...then
JMP 1.3.3;

1.3.1 *sttrans*(max = b;, tab_1,1.3.4) *...else*

1.3.3 WRITE 3;

Aufgabe 2 a)

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

$\Rightarrow$ 2 b) Befehle durchnummerieren und baumstrukturierte Sprungadressen durch lineare ersetzen

C_1

$C_1 = C_0+$ Funktionen ohne Rückgabewert

- `void` Funktionen $\Rightarrow$ wir brauchen:
  1. Referenzparameter `int`*
  2. Referenzzeiger

- Bedingungen:
  1. Wird `void f()` von `void g()` aufgerufen, muss g vor f deklariert sein
  2. Referenzparametern dürfen nur Adressen, keine Werte zugewiesen werden
  3. nur Referenzparameter dürfen dereferrenziert werden

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

Ablaufprotokoll: Aufgabe 4

BZ	DK	LK	REF	Inp	Out
14	ϵ	0:0:1	3	4	ϵ
15		4:0:1		ϵ	
16	1				
17	ϵ	4:0:1:1			
3		4:0:1:1:18:3	6		
4					
5	4				
6	2:4				
7	1				
8	ϵ				
9	4				
10	2:4				
11	2				
12	ϵ	2:0:1:1:18:3	6		
18		2:0:1	3		
19					2