

Informatik II für Verkehrsingenieure

Übersetzung $C_0 \rightarrow AM_0$ (Kapitel 14.3)

Janis Voigtländer

Technische Universität Dresden

Sommersemester 2007

Beispiel

```
#include<stdio.h>
int main()
{ int i,n,s;
  scanf("%i",&n);
  i=1;
  s=0;
  while (i<=n)
    { s=s+i*i;
      i=i+1;
    }
  printf("%d",s);
  return 0;
}
```

Syntaxgesteuerte Übersetzung (I)

$\langle \text{Program} \rangle ::= \#include \langle \text{stdio.h} \rangle \text{ int main}() \langle \text{Block} \rangle.$

$\rightsquigarrow \underline{trans}(\#include \langle \text{stdio.h} \rangle \text{ int main}() \text{ block})$
 $= \underline{blocktrans}(\text{block})$
für alle $\text{block} \in W(\langle \text{Block} \rangle)$

$\langle \text{Block} \rangle ::= \{ \hat{\langle \text{VarDeclaration} \rangle} \hat{\langle \text{StatementSequence} \rangle} \text{ return 0;} \}.$

$\rightsquigarrow \underline{blocktrans}(\{ \text{vardecl statseq return 0;} \})$
 $= \underline{stseqtrans}(\text{statseq}, \dots)$
für alle $\text{vardecl} \in \{ \varepsilon \} \cup W(\langle \text{VarDeclaration} \rangle)$
und $\text{statseq} \in \{ \varepsilon \} \cup W(\langle \text{StatementSequence} \rangle)$

$\langle \text{StatementSequence} \rangle ::= \langle \text{Statement} \rangle \{ \langle \text{Statement} \rangle \}.$

$\rightsquigarrow \underline{stseqtrans}(\text{stat}_1 \dots \text{stat}_n, \dots)$
 $= \underline{sttrans}(\text{stat}_1, \dots)$
...
 $\underline{sttrans}(\text{stat}_n, \dots)$
für alle $\text{stat}_1, \dots, \text{stat}_n \in W(\langle \text{Statement} \rangle)$

3

Syntaxgesteuerte Übersetzung (II)

$\langle \text{Statement} \rangle ::= \langle \text{Assignment} \rangle \hat{\langle \text{IfStatement} \rangle} \hat{\langle \text{WhileStatement} \rangle} \hat{\langle \text{scanf}(\%i, \&\langle \text{Ident} \rangle); \langle \text{printf}(\%d, \langle \text{Ident} \rangle); \langle \text{CompStatement} \rangle}.$

\rightsquigarrow Fallunterscheidung,
zum Beispiel: $\underline{sttrans}(\text{scanf}(\%i, \&id);, \dots)$
 $= \text{READ ?}$
für alle $id \in W(\langle \text{Ident} \rangle)$

Wir brauchen Informationen über die Zuordnung von Bezeichnern
(im Programm) zu Speicherplätzen (im HS der AM₀) !

4

Erzeugung einer Symboltabelle

$\langle \text{Block} \rangle ::= \{ \hat{\langle \text{VarDeclaration} \rangle} \hat{\langle \text{StatementSequence} \rangle} \text{return } 0; \}$.

$\rightsquigarrow \text{blocktrans}(\{ \text{vardecl statseq return } 0; \})$
 $= \text{stseqtrans}(\text{statseq}, \text{update}(\text{vardecl}), \dots)$
für alle $\text{vardecl} \in \{ \varepsilon \} \cup W(\langle \text{VarDeclaration} \rangle)$
und $\text{statseq} \in \{ \varepsilon \} \cup W(\langle \text{StatementSequence} \rangle)$

Menge der Symboltabellen:

$\underline{\text{Tab}} = \{ \text{tab} \mid \text{tab} : W(\langle \text{Ident} \rangle) \rightarrow (\{ \text{var} \} \times \mathbb{N}) \}$

$\langle \text{VarDeclaration} \rangle ::= \text{int } \langle \text{Ident} \rangle \{ \langle \text{Ident} \rangle \};$.

$\rightsquigarrow \text{update}(\varepsilon) = []$ (leere Abbildung)
 $\text{update}(\text{int } id_1, \dots, id_m;)$
 $= [id_1 / (\text{var}, 1), \dots, id_m / (\text{var}, m)]$
für alle $id_1, \dots, id_m \in W(\langle \text{Ident} \rangle)$

Die Symboltabelle wird von stseqtrans aus in weitere Übersetzungsfunktionen propagiert !

5

Syntaxgesteuerte Übersetzung (III)

$\langle \text{Statement} \rangle ::= \langle \text{Assignment} \rangle \hat{\mid} \langle \text{IfStatement} \rangle \hat{\mid} \langle \text{WhileStatement} \rangle \hat{\mid}$
 $\text{scanf}(\text{"\%i"}, \&\langle \text{Ident} \rangle); \hat{\mid} \text{printf}(\text{"\%d"}, \langle \text{Ident} \rangle); \hat{\mid}$
 $\langle \text{CompStatement} \rangle.$

\rightsquigarrow Fallunterscheidung,
zum Beispiel: $\text{sttrans}(\text{scanf}(\text{"\%i"}, \&id);, \text{tab}, \dots)$
 $=$ wenn $\text{tab}(id) = (\text{var}, n)$, dann READ n ;
für alle $id \in W(\langle \text{Ident} \rangle)$ und $\text{tab} \in \underline{\text{Tab}}$

$\langle \text{Assignment} \rangle ::= \langle \text{Ident} \rangle = \langle \text{SimpleExpression} \rangle;.$

$\rightsquigarrow \text{sttrans}(id = \text{exp};, \text{tab}, \dots)$
 $=$ wenn $\text{tab}(id) = (\text{var}, n)$, dann:
simpleexptrans(exp, tab)
STORE n ;
für alle $id \in W(\langle \text{Ident} \rangle)$, $\text{exp} \in W(\langle \text{SimpleExpression} \rangle)$
und $\text{tab} \in \underline{\text{Tab}}$

6

Syntaxgesteuerte Übersetzung (IV)

$\langle \text{SimpleExpression} \rangle ::= \hat{[+ \hat{] - \hat{]}} \langle \text{Term} \rangle \{ \hat{(+ \hat{] - \hat{]}} \langle \text{Term} \rangle \hat{)} \}$.

$\rightsquigarrow \underline{\text{simpleexprtrans}}(\text{sign}_0 t_1 \text{sign}_1 t_2 \dots \text{sign}_{n-1} t_n, \text{tab})$

$= \underline{\text{termtrans}}(t_1, \text{tab})$

SIGN₀

$\underline{\text{termtrans}}(t_2, \text{tab})$

SIGN₁

...

$\underline{\text{termtrans}}(t_n, \text{tab})$

SIGN_{n-1}

für alle $t_1, \dots, t_n \in W(\langle \text{Term} \rangle)$, $\text{sign}_0 \in \{+, -, \varepsilon\}$,

$\text{sign}_1, \dots, \text{sign}_{n-1} \in \{+, -\}$ und $\text{tab} \in \underline{\text{Tab}}$,

wobei SIGN₀ = ε , falls $\text{sign}_0 \in \{+, \varepsilon\}$

SIGN₀ = LIT -1; MUL; falls $\text{sign}_0 = -$

SIGN_i = ADD; falls $\text{sign}_i = +$ und $i \geq 1$

SIGN_i = SUB; falls $\text{sign}_i = -$ und $i \geq 1$

7

Syntaxgesteuerte Übersetzung (V)

$\langle \text{Term} \rangle ::= \langle \text{Factor} \rangle \{ \hat{(* \hat{] / \hat{] \% \hat{]}} \langle \text{Factor} \rangle \hat{)} \}$.

$\rightsquigarrow \underline{\text{termtrans}}(f_1 \text{op}_1 f_2 \text{op}_2 f_3 \dots \text{op}_{n-1} f_n, \text{tab})$

$= \underline{\text{factortrans}}(f_1, \text{tab})$

$\underline{\text{factortrans}}(f_2, \text{tab})$

OP₁;

$\underline{\text{factortrans}}(f_3, \text{tab})$

OP₂;

...

$\underline{\text{factortrans}}(f_n, \text{tab})$

OP_{n-1};

für alle $f_1, \dots, f_n \in W(\langle \text{Factor} \rangle)$, $\text{op}_1, \dots, \text{op}_{n-1} \in \{*, /, \%\}$

und $\text{tab} \in \underline{\text{Tab}}$,

wobei OP_i = MUL, falls $\text{op}_i = *$

OP_i = DIV, falls $\text{op}_i = /$

OP_i = MOD, falls $\text{op}_i = \%$

8

Syntaxgesteuerte Übersetzung (VI)

$\langle \text{Factor} \rangle ::= \langle \text{Ident} \rangle \hat{\mid} \langle \text{Number} \rangle \hat{\mid} (\langle \text{SimpleExpression} \rangle)$.

$\rightsquigarrow \underline{\text{factortrans}}(id, tab)$

= wenn $tab(id) = (var, n)$, dann LOAD n ;
für alle $id \in W(\langle \text{Ident} \rangle)$ und $tab \in \underline{\text{Tab}}$

$\underline{\text{factortrans}}(z, tab)$

= LIT z ;
für alle $z \in W(\langle \text{Number} \rangle)$ und $tab \in \underline{\text{Tab}}$

$\underline{\text{factortrans}}(se, tab)$

= $\underline{\text{simpleexptrans}}(se, tab)$
für alle $se \in W(\langle \text{SimpleExpression} \rangle)$ und $tab \in \underline{\text{Tab}}$

9

Syntaxgesteuerte Übersetzung (VII)

$\langle \text{BoolExpression} \rangle ::= \langle \text{SimpleExpression} \rangle \langle \text{Relation} \rangle \langle \text{SimpleExpression} \rangle$.

$\langle \text{Relation} \rangle ::= == \hat{\mid} != \hat{\mid} < \hat{\mid} > \hat{\mid} <= \hat{\mid} >=$.

$\rightsquigarrow \underline{\text{boolexptrans}}(se_1 \text{ rel } se_2, tab)$

= $\underline{\text{simpleexptrans}}(se_1, tab)$

$\underline{\text{simpleexptrans}}(se_2, tab)$

REL;

für alle $se_1, se_2 \in W(\langle \text{SimpleExpression} \rangle)$,

$rel \in \{==, !=, <, >, <=, >=\}$ und $tab \in \underline{\text{Tab}}$,

wobei REL = EQ, falls $rel ==$

REL = NE, falls $rel !=$

REL = LT, falls $rel <$

...

Syntaxgesteuerte Übersetzung (VIII)

$\langle \text{WhileStatement} \rangle ::= \text{while } (\langle \text{BoolExpression} \rangle) \langle \text{Statement} \rangle.$

$\rightsquigarrow \underline{\text{sttrans}}(\text{while } (exp) \text{ stat}, tab, \dots)$

$= \underline{\text{boolexptrans}}(exp, tab)$

JMC ?;

$\underline{\text{sttrans}}(stat, tab, \dots)$

JMP ?;

für alle $exp \in W(\langle \text{BoolExpression} \rangle)$,

$stat \in W(\langle \text{Statement} \rangle)$ und $tab \in \underline{\text{Tab}}$

- Problem:**
- ▶ keine konkreten Adressen bekannt
 - ▶ hängen unter anderem von Länge des übersetzten Codes für exp und $stat$ ab
- Lösung:**
- ▶ zunächst nur abstrakte Adressen, später Nachbearbeitung
 - ▶ „baumstrukturierte Adressen“: Listen über natürlichen Zahlen (Notation 3.2.4.1)

11

Syntaxgesteuerte Übersetzung (IX)

$\underline{\text{sttrans}}(\text{while } (exp) \text{ stat}, tab, a)$

$= \text{a.2: } \underline{\text{boolexptrans}}(exp, tab)$

JMC a;

$\underline{\text{sttrans}}(stat, tab, \text{a.1})$

JMP a.2;

a:

für alle $exp \in W(\langle \text{BoolExpression} \rangle)$,

$stat \in W(\langle \text{Statement} \rangle)$, $tab \in \underline{\text{Tab}}$ und $a \in \mathbb{N}^*$

12

Syntaxgesteuerte Übersetzung (X)

$\underline{blocktrans}(\{vardecl\ statseq\ return\ 0;\})$
= $\underline{stseqtrans}(statseq, \underline{update}(vardecl), 1)$
für alle $vardecl \in \{\varepsilon\} \cup W(\langle VarDeclaration \rangle)$
und $statseq \in \{\varepsilon\} \cup W(\langle StatementSequence \rangle)$

$\underline{stseqtrans}(stat_1 \dots stat_n, tab, a)$
= $\underline{sttrans}(stat_1, tab, a.1)$
...
 $\underline{sttrans}(stat_n, tab, a.n)$
für alle $stat_1, \dots, stat_n \in W(\langle Statement \rangle)$, $tab \in \underline{Tab}$ und $a \in \mathbb{N}^*$

Noch einige Fälle offen:

$\langle Statement \rangle ::= \langle Assignment \rangle \hat{\vee} \langle IfStatement \rangle \hat{\vee} \langle WhileStatement \rangle \hat{\vee}$
 $\quad \text{scanf}(\"%i", \&\langle Ident \rangle); \hat{\vee} \text{printf}(\"%d", \langle Ident \rangle); \hat{\vee}$
 $\quad \langle CompStatement \rangle.$

13

Syntaxgesteuerte Übersetzung (XI)

$\underline{sttrans}(\text{if } (exp) \text{ stat}, tab, a)$
= $\underline{boolexptrans}(exp, tab)$
JMC a;
 $\underline{sttrans}(stat, tab, a.1)$
a:
für alle $exp \in W(\langle BoolExpression \rangle)$, $stat \in W(\langle Statement \rangle)$,
 $tab \in \underline{Tab}$ und $a \in \mathbb{N}^*$

$\underline{sttrans}(\text{if } (exp) \text{ stat}_1 \text{ else } \text{stat}_2, tab, a)$
= $\underline{boolexptrans}(exp, tab)$
JMC a;
 $\underline{sttrans}(stat_1, tab, a.1)$
JMP a.3;
a: $\underline{sttrans}(stat_2, tab, a.2)$
a.3:
für alle $exp \in W(\langle BoolExpression \rangle)$, $stat_1, stat_2 \in W(\langle Statement \rangle)$,
 $tab \in \underline{Tab}$ und $a \in \mathbb{N}^*$

14

Syntaxgesteuerte Übersetzung (XII)

$\underline{sttrans}(\text{printf}(\text{"\%d"}, id);, tab, a)$
= wenn $tab(id) = (var, n)$, dann WRITE n ;
für alle $id \in W(\langle \text{Ident} \rangle)$, $tab \in \underline{\text{Tab}}$ und $a \in \mathbb{N}^*$

$\underline{sttrans}(\{stat_1 \dots stat_n\}, tab, a)$
= $\underline{stseqtrans}(stat_1 \dots stat_n, tab, a)$
für alle $stat_1, \dots, stat_n \in W(\langle \text{Statement} \rangle)$,
 $tab \in \underline{\text{Tab}}$ und $a \in \mathbb{N}^*$

15

Zusammenfassung (I)

$\underline{trans}(\#include \langle \text{stdio.h} \rangle \text{ int main() } block)$
= $\underline{blocktrans}(block)$

$\underline{blocktrans}(\{vardecl \text{ statseq return 0;} \})$
= $\underline{stseqtrans}(statseq, \underline{update}(vardecl), 1)$

$\underline{update}(\varepsilon) = []$
 $\underline{update}(\text{int } id_1, \dots, id_m;) = [id_1/(var, 1), \dots, id_m/(var, m)]$

$\underline{stseqtrans}(stat_1 \dots stat_n, tab, a)$
= $\underline{sttrans}(stat_1, tab, a.1)$
...
 $\underline{sttrans}(stat_n, tab, a.n)$

16

Zusammenfassung (II)

sttrans(*id = exp*; , *tab*, *a*)
= wenn *tab*(*id*) = (*var*, *n*), dann:
 simplexptrans(*exp*, *tab*)
 STORE *n*;

sttrans(if (*exp*) *stat*, *tab*, *a*)
= boolexptrans(*exp*, *tab*)
 JMC *a*;
 sttrans(*stat*, *tab*, *a.1*)
 a:

sttrans(if (*exp*) *stat*₁ else *stat*₂, *tab*, *a*)
= boolexptrans(*exp*, *tab*)
 JMC *a*;
 sttrans(*stat*₁, *tab*, *a.1*)
 JMP *a.3*;
 a: sttrans(*stat*₂, *tab*, *a.2*)
 a.3:

17

Zusammenfassung (III)

sttrans(while (*exp*) *stat*, *tab*, *a*)
= *a.2*: boolexptrans(*exp*, *tab*)
 JMC *a*;
 sttrans(*stat*, *tab*, *a.1*)
 JMP *a.2*;
 a:

sttrans(scanf ("%i", &*id*) ; , *tab*, *a*)
= wenn *tab*(*id*) = (*var*, *n*), dann READ *n*;

sttrans(printf ("%d", *id*) ; , *tab*, *a*)
= wenn *tab*(*id*) = (*var*, *n*), dann WRITE *n*;

sttrans({*stat*₁ ... *stat*_{*n*}}, *tab*, *a*)
= stseqtrans(*stat*₁ ... *stat*_{*n*}, *tab*, *a*)

18

Zusammenfassung (IV)

boolexptrans($se_1 \text{ rel } se_2, tab$)

= simplexptrans(se_1, tab)

simplexptrans(se_2, tab)

REL;

wobei REL = EQ, falls $rel ==$

...

simplexptrans($sign_0 t_1 sign_1 t_2 \dots sign_{n-1} t_n, tab$)

= termtrans(t_1, tab)

SIGN₀

...

termtrans(t_n, tab)

SIGN _{$n-1$}

wobei SIGN₀ = ε , falls $sign_0 \in \{+, \varepsilon\}$

SIGN₀ = LIT -1; MUL; falls $sign_0 = -$

SIGN _{i} = ADD; falls $sign_i = +$ und $i \geq 1$

SIGN _{i} = SUB; falls $sign_i = -$ und $i \geq 1$

19

Zusammenfassung (V)

termtrans($f_1 op_1 f_2 op_2 f_3 \dots op_{n-1} f_n, tab$)

= factortrans(f_1, tab)

factortrans(f_2, tab)

OP₁;

factortrans(f_3, tab)

OP₂;

...

factortrans(f_n, tab)

OP _{$n-1$} ;

wobei OP _{i} = MUL, falls $op_i = *$

...

factortrans(id, tab)

= wenn $tab(id) = (var, n)$, dann LOAD n ;

factortrans(z, tab) = LIT z ;

factortrans($(se), tab$) = simplexptrans(se, tab)

20

Beispiel

```
#include<stdio.h>
int main()
{ int i,n,s;
  scanf("%i",&n);
  i=1;
  s=0;
  while (i<=n)
    { s=s+i*i;
      i=i+1;
    }
  printf("%d",s);
  return 0;
}
```

21

Beispiel — übersetzt

| | | |
|----------------|----------|---------------|
| READ 2; | LE; | STORE 3; |
| LIT 1; | JMC 1.4; | LOAD 1; |
| STORE 1; | LOAD 3; | LIT 1; |
| LIT 0; | LOAD 1; | ADD; |
| STORE 3; | LOAD 1; | STORE 1; |
| 1.4.2: LOAD 1; | MUL; | JMP 1.4.2; |
| LOAD 2; | ADD; | 1.4: WRITE 3; |

Linearisierung:

1. Durchnummerierung der Befehle, beginnend mit 1
2. Merken von Paaren aus baumstrukturierter Adresse und numerierter Adresse
3. Anpassen von Sprungbefehlen entsprechend der gemerkten Paare

22

Beispiel — linearisiert

| | | |
|-------------|-------------|--------------|
| 1: READ 2; | 8: LE; | 15: STORE 3; |
| 2: LIT 1; | 9: JMC 21; | 16: LOAD 1; |
| 3: STORE 1; | 10: LOAD 3; | 17: LIT 1; |
| 4: LIT 0; | 11: LOAD 1; | 18: ADD; |
| 5: STORE 3; | 12: LOAD 1; | 19: STORE 1; |
| 6: LOAD 1; | 13: MUL; | 20: JMP 6; |
| 7: LOAD 2; | 14: ADD; | 21: WRITE 3; |

(1 , ε , [] , 1 , ε)

23

Beispiel — linearisiert

| | | |
|-------------|-------------|--------------|
| 1: READ 2; | 8: LE; | 15: STORE 3; |
| 2: LIT 1; | 9: JMC 21; | 16: LOAD 1; |
| 3: STORE 1; | 10: LOAD 3; | 17: LIT 1; |
| 4: LIT 0; | 11: LOAD 1; | 18: ADD; |
| 5: STORE 3; | 12: LOAD 1; | 19: STORE 1; |
| 6: LOAD 1; | 13: MUL; | 20: JMP 6; |
| 7: LOAD 2; | 14: ADD; | 21: WRITE 3; |

(18 , 1:1 , [1/1,2/1,3/1] , ε , ε)

(19 , 2 , [1/1,2/1,3/1] , ε , ε)

(20 , ε , [1/2,2/1,3/1] , ε , ε)

(6 , ε , [1/2,2/1,3/1] , ε , ε)

(7 , 2 , [1/2,2/1,3/1] , ε , ε)

(8 , 1:2 , [1/2,2/1,3/1] , ε , ε)

(9 , 0 , [1/2,2/1,3/1] , ε , ε)

(21 , ε , [1/2,2/1,3/1] , ε , ε)

(22 , ε , [1/2,2/1,3/1] , ε , 1)

23