

# Compiler



**Zwischencode**

**für abstrakte Maschine**

## Teilaufgaben der Übersetzung in **Zwischencode** **Maschinencode**

1. Übersetzung von Ausdrücken und Kontrollstrukturen
2. Übersetzung von Blöcken und Prozeduren
3. Übersetzung von Datenstrukturen
4. Inkrementelle Übersetzung von Modulstrukturen

1. Registerallokation
2. Code-Auswahl
3. Instruktionsanordnung
4. ...

# Syntax von PSA

Ganze Zahlen  $Int$  :  $Z$

Bezeichner  $Ide$  :  $I$

Deklarationen  $Decl$  :  $\Delta ::= \Delta_C \Delta_V$   
 $\Delta_C ::= \varepsilon \mid \mathbf{const} \ I_1 = Z_1; \dots; I_n = Z_n$   
 $(n \geq 1)$   
 $\Delta_V ::= \varepsilon \mid \mathbf{var} \ I_1, \dots, I_n; \quad (n \geq 1)$   
(\* nur Integervariablen \*)

Arithmetische  $AExp$  :  $E ::= Z \mid I \mid (E_1 \ aop \ E_2)$   
Ausdrücke  
 $(aop \in \{+, -, *, \dots\})$

Boolesche  $BExp$  :  $B ::= E \ relop \ E \mid$   
Ausdrücke  
 $\mathbf{not} \ B \mid (B \ \mathbf{and} \ B) \mid (B \ \mathbf{or} \ B)$   
 $(relop \in \{=, \neq, <, \dots\})$

Anweisungen  $Cmd$  :  $\Gamma ::= I := E \mid \Gamma_1; \Gamma_2$   
 $\mid \mathbf{if} \ B \ \mathbf{then} \ \Gamma \ \mathbf{else} \ \Gamma$   
 $\mid \mathbf{while} \ B \ \mathbf{do} \ \Gamma$

Programme  $Prog$  :  $P ::= \Delta \Gamma$

## kontextsensitive Bedingungen

- Bezeichner in  $\Delta$  müssen paarweise verschieden sein.
- Bezeichner in  $\Gamma$  müssen deklariert sein.

# Semantik von PSA

Speicherplätze (locations)  $Loc := \{\alpha_1, \alpha_2, \alpha_3, \dots\}$

Zustandsraum (states)  $S := \{\sigma \mid \sigma : Loc \dashrightarrow \mathbb{Z}\}$

Umgebung (environment)  $Env := \{\rho \mid \rho : Ide \dashrightarrow \mathbb{Z} \cup Loc\}$

---

## Deklarationssemantik

$\mathcal{D} :: Decl \times Env \dashrightarrow Env$

$\mathcal{D}[\Delta_C \Delta_V]_\rho := \mathcal{D}[\Delta_V](\mathcal{D}[\Delta_C]\rho)$

$\mathcal{D}[\varepsilon]\rho := \rho$

$\mathcal{D}[\mathbf{const} \ I_1 = Z_1; \dots; I_n = Z_n]_\rho := \rho[I_1/Z_1, \dots, I_n/Z_n]$

$\mathcal{D}[\mathbf{var} \ I_1, I_2, \dots, I_n]_\rho := \rho[I_1/\alpha_1, \dots, I_n/\alpha_n]$

einfache Speicherverwaltung!

---

## Semantik arithmetischer Ausdrücke

$\mathcal{E} :: AExp \times Env \times S \dashrightarrow \mathbb{Z}$

$\mathcal{E}[Z]\rho\sigma := Z$

$\mathcal{E}[I]\rho\sigma := \begin{cases} \rho(I) & \text{falls } \rho(I) \in \mathbb{Z} \\ \underbrace{\sigma(\underbrace{\rho(I)}_{\text{l-Wert}})}_{\text{r-Wert}} & \text{falls } \rho(I) \in Loc \end{cases}$

$\mathcal{E}[(E_1 \text{ op } E_2)]\rho\sigma := \mathcal{E}[E_1]\rho\sigma [\text{op}] \mathcal{E}[E_2]\rho\sigma$

---

## Semantik Boolescher Ausdrücke

$$\mathcal{B} :: BExp \times Env \times S \dashrightarrow \{\mathbf{true}, \mathbf{false}\}$$

analog mit strikten bzw. nicht-strikten Varianten

---

## Semantik von Anweisungen

$$\mathcal{C} :: Cmd \times Env \times S \dashrightarrow S$$

$$\mathcal{C}[I := E]\rho\sigma \quad := \quad \sigma[\rho(I) \mapsto \mathcal{E}[E]\rho\sigma]$$

$$\mathcal{C}[\Gamma_1; \Gamma_2]\rho\sigma \quad := \quad \mathcal{C}[\Gamma_2]\rho(\mathcal{C}[\Gamma_1]\rho\sigma)$$

$$\mathcal{C}[\mathbf{if} \ B \ \mathbf{then} \ \Gamma_1 \ \mathbf{else} \ \Gamma_2]\rho\sigma \quad := \quad \begin{cases} \mathcal{C}[\Gamma_1]\rho\sigma & \text{falls } \mathcal{B}[B]\rho\sigma = \mathbf{true} \\ \mathcal{C}[\Gamma_2]\rho\sigma & \text{falls } \mathcal{B}[B]\rho\sigma = \mathbf{false} \end{cases}$$

$$\mathcal{C}[\mathbf{while} \ B \ \mathbf{do} \ \Gamma]\rho\sigma \quad := \quad \begin{cases} \mathcal{C}[\mathbf{while} \ B \ \mathbf{do} \ \Gamma]\rho(\mathcal{C}[\Gamma]\rho\sigma) & \\ \text{falls } \mathcal{B}[B]\rho\sigma = \mathbf{true} & \\ \sigma & \text{falls } \mathcal{B}[B]\rho\sigma = \mathbf{false} \end{cases}$$

---

## Programmsemantik $\mathcal{M} :: Prog \times S \dashrightarrow S$

$$\mathcal{M}[\Delta\Gamma]\sigma := \mathcal{C}[\Gamma](\mathcal{D}[\Delta]\rho_\emptyset)\sigma$$

## Beispiel zur Übersetzung von PSA in MA-Code

$$\text{Sei } P = \begin{array}{l} \Delta \left\{ \begin{array}{l} \text{const } I = 3, J = 10; \\ \text{var } K, L; \end{array} \right. \\ \Gamma \left\{ \begin{array}{l} \Gamma_1 \{ K := I + L; \\ \Gamma_2 \{ \text{if } K > L \text{ then } K := K-L \text{ else } K := 1 \end{array} \right. \end{array}$$

Dann gilt:

$$\text{trans}(P) = \text{ct}(\Gamma, \text{up}(\Delta, st_\emptyset), 1),$$

$$\begin{aligned} \text{wobei } & \text{up}(\text{const } I = 3, J = 10; \text{ var } K, L; , st_\emptyset) \\ &= \text{up}(\text{var } K, L; , \text{up}(\text{const } I = 3, J = 10; , st_\emptyset)) \\ &= \text{up}(\text{var } K, L; , st_\emptyset[I/(const, 3), J/(const, 10)]) \\ &= st_\emptyset[I/(const, 3), J/(const, 10), K/(var, 1), L/(var, 2)] \\ &=: st_1 \end{aligned}$$

$$\text{und } \text{ct}(\Gamma, st_1, 1) = \text{ct}(\Gamma_1, st_1, 1); \text{ct}(\Gamma_2, st_1, 5)$$

$\begin{aligned} & \text{ct}(\Gamma_1, st_1, 1) \\ &= \text{et}(I + L, st_1) \\ & \quad 4: \text{STORE } 1; \\ &= 1: \text{LIT } 3; \\ & \quad 2: \text{LOAD } 2; \\ & \quad 3: \text{ADD}; \\ & \quad 4: \text{STORE } 1 \end{aligned}$	$\begin{aligned} & \text{ct}(\Gamma_2, st_1, 5) \\ &= 5: \text{LOAD } 1; \\ & \quad 6: \text{LOAD } 2; \\ & \quad 7: \text{GREATER}; \\ & \quad 8: \text{JPFALSE } 14; \\ & \quad 9: \text{LOAD } 1; \\ & \quad 10: \text{LOAD } 2; \\ & \quad 11: \text{SUB}; \\ & \quad 12: \text{STORE } 1; \\ & \quad 13: \text{JMP } 16; \\ & \quad 14: \text{LIT } 1; \\ & \quad 15: \text{STORE } 1. \end{aligned}$
--	--