Compiler

Frontend **Backend**

$$PS \longrightarrow Z \longrightarrow MC$$

Zwischencode

für abstrakte Maschine

Teilaufgaben der Übersetzung in

Zwischencode

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

Maschinencode

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

Syntax von PSA

Ganze Zahlen	*Int*	:	Z		
Bezeichner	*Ide*	:	I		
Deklarationen	*Decl*	:	Δ	$::=$	$\Delta_C \Delta_V$
			Δ_C	$::=$	$\varepsilon \mid$ **const** $I_1 = Z_1; \ldots ; I_n = Z_n$ $(n \geq 1)$
			Δ_V	$::=$	$\varepsilon \mid$ **var** $I_1, \ldots, I_n;$ $(n \geq 1)$ (* nur Integervariablen *)
Arithmetische Ausdrücke	*AExp*	:	E	$::=$	$Z \mid I \mid (E_1 \; aop \; E_2)$ $(aop \in \{+, -, *, \ldots\})$
Boolesche Ausdrücke	*BExp*	:	B	$::=$	$E \; relop \; E \mid$ **not** $B \mid (B$ **and** $B) \mid (B$ **or** $B)$ $(relop \in \{=, \neq, <, \ldots\})$
Anweisungen	*Cmd*	:	Γ	$::=$	$I := E \mid \Gamma_1; \Gamma_2$ $\mid$ **if** B **then** Γ **else** Γ $\mid$ **while** B **do** Γ
Programme	*Prog*	:	P	$::=$	$\Delta\Gamma$

kontextsensitive Bedingungen

- Bezeichner in Δ müssen paarweise verschieden sein.
- Bezeichner in Γ müssen deklariert sein.

Semantik von PSA

Speicherplätze (locations)	Loc	$:=$	$\{\alpha_1, \alpha_2, \alpha_3, \ldots\}$
Zustandsraum (states)	S	$:=$	$\{\sigma \mid \sigma : Loc \dashrightarrow \mathbb{Z}\}$
Umgebung (environment)	Env	$:=$	$\{\rho \mid \rho : Ide \dashrightarrow \mathbb{Z} \cup \mathsf{Loc}\}$

Deklarationssemantik

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

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

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

$$\mathcal{D}[\![\textbf{const}\ I_1 = Z_1; \ldots; I_n = Z_n]\!]_\rho \quad := \quad \rho\,[I_1/Z_1, \ldots, I_n/Z_n]$$

$$\mathcal{D}[\![\texttt{var}\ \ I_1, I_2, \ldots, I_n]\!]_\rho \quad := \quad \rho[I_1/\alpha_1, \ldots, I_n/\alpha_n]$$

einfache Speicherverwaltung!

Semantik arithmetischer Ausdrücke

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

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

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

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

Semantik Boolescher Ausdrücke

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

analog mit strikten bzw. nicht-strikten Varianten

Semantik von Anweisungen

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

$$\begin{array}{lll}
\mathcal{C}[\![I := E]\!]\rho\sigma & := & \sigma\,[\rho(I) \mapsto \mathcal{E}[\![E]\!]\rho\sigma] \\
\mathcal{C}[\![\Gamma_1; \Gamma_2]\!]\rho\sigma & := & \mathcal{C}[\![\Gamma_2]\!]\rho(\mathcal{C}[\![\Gamma_1]\!]\rho\sigma) \\
\mathcal{C}[\![\textbf{if } B \textbf{ then } \Gamma_1 \textbf{ else } \Gamma_2]\!]\rho\sigma & := & \begin{cases} \mathcal{C}[\![\Gamma_1]\!]\rho\sigma & \text{falls } \mathcal{B}[\![B]\!]\rho\sigma = \text{true} \\ \mathcal{C}[\![\Gamma_2]\!]\rho\sigma & \text{falls } \mathcal{B}[\![B]\!]\rho\sigma = \text{false} \end{cases} \\
\mathcal{C}[\![\textbf{while } B \textbf{ do } \Gamma]\!]\rho\sigma & := & \begin{cases} \mathcal{C}[\![\textbf{while } B \textbf{ do } \Gamma]\!]\rho(\mathcal{C}[\![\Gamma]\!]\rho\sigma) & \\ \qquad\qquad \text{falls } \mathcal{B}[\![B]\!]\rho\sigma = \text{true} & \\ \sigma \qquad\quad \text{falls } \mathcal{B}[\![B]\!]\rho\sigma = \text{false} & \end{cases}
\end{array}$$

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*

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

Dann gilt:

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

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

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

$ct(\Gamma_1, st_1, 1)$

$=$ $et(\texttt{I} + \texttt{L}, st_1)$
4: STORE 1;

$=$ 1: LIT 3;
2: LOAD 2;
3: ADD;
4: STORE 1

$ct(\Gamma_2, st_1, 5)$

$=$ 5: LOAD 1;
6: LOAD 2;
7: GREATER;
8: JPFALSE 14;
9: LOAD 1;
10: LOAD 2;
11: SUB;
12: STORE 1;
13: JMP 16;
14: LIT 1;
15: STORE 1.

Übersetzung Boolescher Ausdrücke

$$\boxed{\textbf{not}\ (I = 0)\ \ \textbf{and}\ \ (J > I)}$$

Standardübersetzung
(strikte Semantik))

```
LOAD I;
LIT 0;
EQ;
NOT;
LOAD J;
LOAD I;
GREATER;
AND
```

$\underbrace{\qquad}_{\text{8 Befehle}}$

Falls $\sigma(I) = 0$ Ausführung von **8 Befehlen**

Jumping Code
(nicht-strikte Semantik))

```
   LOAD I;
   LIT 0;
   EQ;
   JPFALSE a;
   JMP a_false;
a: LOAD J;
   LOAD I;
   GREATER;
   JPFALSE a_false;
   JMP a_true
```

$\underbrace{\qquad\qquad}_{\text{10 Befehle}}$

Falls $\sigma(I) = 0$ Ausführung von **5 Befehlen**

Jumping Code länger, aber kürzer in Laufzeit

Compilerkorrektheit

Für ein PSA-Programm P ist

$$\begin{array}{lcl} M[\![P]\!] & : & S \dashrightarrow S \text{ und} \\ I[\![trans\ (P)]\!] & : & ZR \dashrightarrow ZR. \end{array}$$

Es gilt $S \ni \sigma : Loc \dashrightarrow \mathbb{Z}$ und

$$ZR = BZ \times DK \times HS \text{ mit } HS \ni h : \mathbb{N} \dashrightarrow \mathbb{Z}.$$

Mit der folgenden Beziehung zwischen S und HS

$$\sigma \mapsto h_\sigma \text{ mit } h_\sigma(i) := \begin{cases} \sigma(\alpha_i) & \text{falls } \sigma(\alpha_i) \text{ definiert.} \\ 0 & \text{sonst} \end{cases}$$

gilt dann:

$$\boxed{\mathcal{M}[\![P]\!]\sigma = \sigma' \iff I[\![trans(P)]\!](1, \varepsilon, h_\sigma) = (m, \varepsilon, h_{\sigma'}).}$$