Compiler

$$\begin{array}{ccccc} & & \textbf{Compiler} & & \\ & / & & \backslash & \\ \textbf{Frontend} & & & & \textbf{Backend} \\ PS & \rightarrow & Z & \rightarrow & MC \end{array}$$

Zwischencode

für abstrakte Maschine

Strukturen von Programmiersprachen (PS)

- Basistypen und Basisoperationen
- statische und dynamische Datenstrukturen
- Kontrollstrukturen
- Ausdrücke, Funktionen, Prozeduren, Methoden
- Modulstrukturen: Blöcke, Module, Klassen

Strukturen von Maschinencode (MC)

- Speicherhierarchie:
  Register, Cache, Hauptspeicher, Hintergrundspeicher
- Befehle: ALU, Tests/Sprünge, Transfer, Ein-/Ausgabe
- Adressierung: relativ mit Index-/Basisregister, indirekt
- RISC / CISC

Strukturen von Zwischencode (Z)

- Typen und Operationen wie in PS
- Datenkeller mit Basisoperationen
- Sprungbefehle für Kontrollstrukturen
- Prozedurkeller für Blöcke/Prozeduren „deletion strategy"
- Heap für dynamische Datenstrukturen, Objekte
  „retention strategy"

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

Teilaufgaben der Übersetzung in Maschinencode

- Registerallokation
- Instruktionsanordnung
- Code-Auswahl
- ...

Syntax von PSA

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

kontextsensitive Bedingungen

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

Semantik von PSA (Skizze)

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

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

$$\begin{array}{ll}
\mathcal{D}[\![\Delta_C \Delta_V]\!]_\rho & := \mathcal{D}[\![\Delta_V]\!](\mathcal{D}[\![\Delta_C]\!]\rho) \\
 & \\
\mathcal{D}[\![\varepsilon]\!]\rho & := \rho \\
\mathcal{D}[\![\textbf{const } I_1 = Z_1; \ldots; I_n = Z_n]\!]_\rho & := \rho\,[I_1/Z_1, \ldots, I_n/Z_n] \\
\mathcal{D}[\![\texttt{var } I_1, I_2, \ldots, I_n]\!]_\rho & := \rho[I_1/\alpha_1, \ldots, I_n/\alpha_n]
\end{array}$$

Semantik arithmetischer Ausdrücke

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

$$\begin{array}{ll}
\mathcal{E}[\![Z]\!]\rho\sigma & := Z \\
\mathcal{E}[\![I]\!]\rho\sigma & := \begin{cases} \rho(I) & \text{falls } \rho(I) \in \mathbb{Z} \\ \sigma(\rho(I)) & \text{falls } \rho(I) \in \mathit{Loc} \end{cases} \\
\mathcal{E}[\![(E_1 op E_2)]\!]\rho\sigma & := \mathcal{E}[\![E_1]\!]\rho\sigma\ [\![op]\!]\ \mathcal{E}[\![E_2]\!]\rho\sigma
\end{array}$$

Semantik Boolescher Ausdrücke

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

analog mit strikten bzw. nicht-strikten Varianten

Semantik von Anweisungen

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

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

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