Übersetzungsfunktionen:

Anweisungen:

$$
\begin{array}{lll}
\text{code } e;\ \rho & = & \text{code}_{\text{R}}\ e\ \rho \\
 & & \text{pop}
\end{array}
$$

$$
\begin{array}{lll}
\text{code } (s\, ss)\ \rho & = & \text{code } s\ \rho \\
 & & \text{code } ss\ \rho \\
\text{code } \varepsilon\ \rho & = & \quad /\!/ \quad \textit{leere Folge von Befehlen}
\end{array}
$$

$$
\begin{array}{llll}
\text{code } (if\ (e)\ s)\ \rho & = & & \text{code}_{\text{R}}\ e\ \rho \\
 & & & \text{jumpz A} \\
 & & & \text{code } s\ \rho \\
 & & \text{A :} & \dots \\
\text{code } (if\ (e)\ s_1\ else\ s_2)\ \rho & = & & \text{code}_{\text{R}}\ e\ \rho \\
 & & & \text{jumpz A} \\
 & & & \text{code } s_1\ \rho \\
 & & & \text{jump B} \\
 & & \text{A :} & \text{code } s_2\ \rho \\
 & & \text{B :} & \dots
\end{array}
$$

$$
\begin{array}{lll}
\text{code } (while\ (e)\ s)\ \rho\ = & & \\
 & \text{A :} & \text{code}_{\text{R}}\ e\ \rho \\
 & & \text{jumpz B} \\
 & & \text{code } s\ \rho \\
 & & \text{jump A} \\
 & \text{B :} & \dots
\end{array}
$$

$$
\begin{array}{llll}
\text{code } (for\ (e_1; e_2; e_3)\ s)\ \rho & = & & \text{code}_{\text{R}}\ e_1 \\
 & & & \text{pop} \\
 & & \text{A :} & \text{code}_{\text{R}}\ e_2\ \rho \\
 & & & \text{jumpz B} \\
 & & & \text{code } s\ \rho \\
 & & & \text{code}_{\text{R}}\ e_3\ \rho \\
 & & & \text{pop} \\
 & & & \text{jump A} \\
 & & \text{B :} & \dots
\end{array}
$$

$$
\begin{array}{lllllll}
\text{code } s\ \rho & = & \text{code}_{\text{R}}\ e\ \rho & \text{C}_0\text{:} & \text{code } ss_0\ \rho & \text{B:} & \text{jump C}_0 \\
 & & \text{check } 0\ k\ \text{B} & & \text{jump D} & & \dots \\
 & & & & \dots & & \text{jump C}_k \\
 & & & \text{C}_k\text{:} & \text{code } ss_k\ \rho & \text{D:} & \dots \\
 & & & & \text{jump D} & &
\end{array}
$$

wenn $s =$ **switch** (e) {
  case 0: ss_0 **break**;
  case 1: ss_1 **break**;
  ⋮
  case $k-1$: ss_{k-1} **break**;
  default: ss_k
}

wobei	check 0 k B	=	dup	dup		jumpi B
			loadc 0	loadc k	A:	pop
			geq	leq		loadc k
			jumpz A	jumpz A		jumpi B

Ausdrücke:

$\text{code}_\text{L}\ (e_1[e_2])\ \rho$	$=$	$\text{code}_\text{R}\ e_1\ \rho$ $\text{code}_\text{R}\ e_2\ \rho$ loadc $\lvert t\rvert$ mul add	sofern e_1 Typ $t\,[\,]$ hat
$\text{code}_\text{L}\ (e.a)\ \rho$	$=$	$\text{code}_\text{L}\ e\ \rho$ loadc $(\rho\, a)$ add	
$\text{code}_\text{L}\ (*e)\ \rho$	$=$	$\text{code}_\text{R}\ e\ \rho$	
$\text{code}_\text{L}\ x\ \rho$	$=$	loadc $(\rho\, x)$	
$\text{code}_\text{R}\ (\&e)\ \rho$	$=$	$\text{code}_\text{L}\ e\ \rho$	
$\text{code}_\text{R}\ (\text{malloc}\,(e))\ \rho$	$=$	$\text{code}_\text{R}\ e\ \rho$ new	
$\text{code}_\text{R}\ e\ \rho$	$=$	$\text{code}_\text{L}\ e\ \rho$	falls e ein Feld ist
$\text{code}_\text{R}\ (e_1 \,\square\, e_2)\ \rho$	$=$	$\text{code}_\text{R}\ e_1\ \rho$ $\text{code}_\text{R}\ e_2\ \rho$ op	op Befehl zu Operator '$\square$'
$\text{code}_\text{R}\ q\ \rho$	$=$	loadc q	q Konstante
$\text{code}_\text{R}\ (e_1 = e_2)\ \rho$	$=$	$\text{code}_\text{R}\ e_2\ \rho$ $\text{code}_\text{L}\ e_1\ \rho$ store	
$\text{code}_\text{R}\ e\ \rho$	$=$	$\text{code}_\text{L}\ e\ \rho$ load	sonst

Funktionen:

$$\begin{aligned}\text{code}_\text{R}\ g(e_1,\ldots,e_n)\ \rho \;=\;& \text{mark}\\ & \text{code}_\text{R}\ e_1\ \rho\\ & \ldots\\ & \text{code}_\text{R}\ e_n\ \rho\\ & \text{code}_\text{R}\ g\ \rho\\ & \text{call m}\end{aligned}$$

wobei m der Platz für die aktuellen Parameter ist.

$\text{code}_\text{R}\ f\ \rho \quad = \quad \text{loadc}\ (\rho\ f)$ — f ein Funktions-Name

$\text{code}_\text{R}\ (*e)\ \rho \quad = \quad \text{code}_\text{R}\ e\ \rho$ — e ein Funktions-Zeiger

$\text{code}_\text{R}\ e\ \rho \quad = \quad \text{code}_\text{L}\ e\ \rho$
move k — e eine Struktur der Größe k

code t f (*specs*){*V_defs* ss} ρ =

_f : enter q // *setzen des EP*
alloc k // *Anlegen der lokalen Variablen*
code ss ρ_f
return // *Verlassen der Funktion*

$$\text{code}_\text{L}\ x\ \rho = \begin{cases} \text{loadc j} & tag = G \\ \text{loadrc j} & tag = L \end{cases}$$

code return e; ρ = $\text{code}_\text{R}\ e\ \rho$
storer -3
return

Ganze Programme:

code $p\ \emptyset$ =
enter (k + 6)
alloc (k + 1)
mark
loadc _main
call 0
pop
halt
$\_\text{f}_1$: code *F_def*$_1$ ρ
$\vdots$
$\_\text{f}_n$: code *F_def*$_n$ ρ

wobei $p \equiv$ *V_defs* *F_def*$_1$ ... *F_def*$_n$
$\emptyset \mathrel{\widehat{=}}$ leere Adress-Umgebung;
$\rho \mathrel{\widehat{=}}$ globale Adress-Umgebung;
k $\mathrel{\widehat{=}}$ Platz für globale Variablen