COSE312: Compilers

Lecture 12 — Translation (2)

Hakjoo Oh
2015 Fall

S: The Source Language

$$
\begin{array}{rcl}
\textit{program} & \rightarrow & \textit{block} \\
\textit{block} & \rightarrow & \textit{decls}\ \textit{stmts} \\
\textit{decls} & \rightarrow & \textit{decls}\ \textit{decl} \mid \epsilon \\
\textit{decl} & \rightarrow & \textit{type}\ x \\
\textit{type} & \rightarrow & \texttt{int} \mid \texttt{int}\texttt{[}n\texttt{]} \\
\textit{stmts} & \rightarrow & \textit{stmts}\ \textit{stmt} \mid \epsilon \\
 & & \\
\textit{stmt} & \rightarrow & \textit{lv}\ \texttt{=}\ e \\
 & \mid & \texttt{if}\ e\ \textit{stmt}\ \textit{stmt} \\
 & \mid & \texttt{while}\ e\ \textit{stmt} \\
 & \mid & \texttt{do}\ \textit{stmt}\ \texttt{while}\ e \\
 & \mid & \texttt{read}\ x \\
 & \mid & \texttt{print}\ e \\
 & \mid & \textit{block} \\
 & & \\
\textit{lv} & \rightarrow & x \mid x\texttt{[}e\texttt{]} \\
 & & \\
e & \rightarrow & n \qquad \text{integer} \\
 & \mid & \textit{lv} \qquad \text{l-value} \\
 & \mid & e\texttt{+}e \mid e\texttt{-}e \mid e\texttt{*}e \mid e\texttt{/}e \mid \texttt{-}e \qquad \text{airthmetic operation} \\
 & \mid & e\texttt{==}e \mid e\texttt{<}e \mid e\texttt{<=}e \mid e\texttt{>}e \mid e\texttt{>=}e \qquad \text{conditional operation} \\
 & \mid & \texttt{!}e \mid e\texttt{||}e \mid e\texttt{\&\&}e \qquad \text{boolean operation}
\end{array}
$$

T: The Target Language

$$
\begin{array}{rcl}
program & \rightarrow & LabeledInstruction^* \\
LabeledInstruction & \rightarrow & Label \times Instruction \\
Instruction & \rightarrow & \texttt{skip} \\
& | & x = \texttt{alloc}(n) \\
& | & x = y \ bop \ z \\
& | & x = y \ bop \ n \\
& | & x = uop \ y \\
& | & x = y \\
& | & x = n \\
& | & \texttt{goto} \ L \\
& | & \texttt{if} \ x \ \texttt{goto} \ L \\
& | & \texttt{ifFalse} \ x \ \texttt{goto} \ L \\
& | & x = y[i] \\
& | & x[i] = y \\
& | & \texttt{read} \ x \\
& | & \texttt{write} \ x \\
bop & \rightarrow & \texttt{+} \mid \texttt{-} \mid \texttt{*} \mid \texttt{/} \mid \texttt{>} \mid \texttt{>=} \mid \texttt{<} \mid \texttt{<=} \mid \texttt{==} \mid \texttt{\&\&} \mid \texttt{||} \\
uop & \rightarrow & \texttt{-} \mid \texttt{!}
\end{array}
$$

Translation of Expressions

Examples:

- 2:
- `x`:
- `x[1]`:
- 2+3:
- `-5`:
- `(x+1)+y[2]`:

Translation of Expressions

$$\textbf{trans}_e : e \rightarrow \textit{Var} \times \textit{LabeledInstruction}^*$$

$$
\begin{array}{rcll}
\textbf{trans}_e(n) & = & (t, [t = n]) & \cdots \text{new } t \\
\textbf{trans}_e(x) & = & (t, [t = x]) & \cdots \text{new } t \\
\textbf{trans}_e(x[e]) & = & \text{let } (t_1, code) = \textbf{trans}_e(e) & \\
 & & \text{in } (t_2, code@[t_2 = x[t_1]]) & \cdots \text{new } t_2 \\
\textbf{trans}_e(e_1 + e_2) & = & \text{let } (t_1, code_1) = \textbf{trans}_e(e_1) & \\
 & & \text{let } (t_2, code_2) = \textbf{trans}_e(e_2) & \\
 & & \text{in } (t_3, code_1@code_2@[t_3 = t_1 + t_2]) & \cdots \text{new } t_3 \\
\textbf{trans}_e(-e) & = & \text{let } (t_1, code_1) = \textbf{trans}_e(e) & \\
 & & \text{in } (t_2, code_1@[t_2 = -t_1]) & \cdots \text{new } t_2
\end{array}
$$

Translation of Statements

Examples:

- `x=1+2`:
- `x[1]=2`:
- `if (1) x=1; else x=2;`
- `while (x<10) x++;`

Translation of Statements

$$\textbf{trans}_s : \mathit{stmt} \to \mathit{LabeledInstruction}^*$$

$$\begin{array}{rcl}
\textbf{trans}_s(x = e) & = & \text{let } (t_1, code_1) = \textbf{trans}_e(e) \\
 & & code_1 @ [x = t_1] \\
\textbf{trans}_s(x[e_1] = e_2) & = & \text{let } (t_1, code_1) = \textbf{trans}_e(e_1) \\
 & & \text{let } (t_2, code_2) = \textbf{trans}_e(e_2) \\
 & & \text{in } code_1 @ code_2 @ [x[t_1] = t_2] \\
\textbf{trans}_s(\texttt{read } x) & = & [\texttt{read } x] \\
\textbf{trans}_s(\texttt{print } e) & = & \text{let } (t_1, code_1) = \textbf{trans}_e(e) \\
 & & \text{in } code_1 @ [\texttt{write } t_1]
\end{array}$$

Translation of Statements

$$
\begin{array}{l}
\textbf{trans}_s(\texttt{if}\ e\ stmt_1\ stmt_2) = \\
\quad \text{let } (t_1, code_1) = \textbf{trans}_e(e) \\
\quad \text{let } code_t = \textbf{trans}_s(stmt_1) \\
\quad \text{let } code_f = \textbf{trans}_s(stmt_2) \\
\quad \text{in } code_1 @ \qquad\qquad\qquad\qquad \cdots \text{new } l_t, l_f, l_x \\
\qquad [\texttt{if}\ t_1\ \texttt{goto}\ l_t] @ \\
\qquad [\texttt{goto}\ l_f] @ \\
\qquad [(l_t, \texttt{skip})] @ \\
\qquad\quad code_t @ \\
\qquad\quad [\texttt{goto}\ l_x] @ \\
\qquad [(l_f, \texttt{skip})] @ \\
\qquad\quad code_f @ \\
\qquad\quad [\texttt{goto}\ l_x] @ \\
\qquad [(l_x, \texttt{skip})]
\end{array}
$$

Translation of Statements

$\textbf{trans}_s(\texttt{while}\ e\ stmt) =$

$$
\begin{array}{l}
\text{let } (t_1, code_1) = \textbf{trans}_e(e) \\
\text{let } code_b = \textbf{trans}_s(stmt) \\
\text{in } [(l_e, \texttt{skip})]@ \qquad\qquad\qquad \cdots \text{new } l_e, l_x \\
\qquad code_1 @ \\
\qquad [\texttt{ifFalse}\ t_1\ l_x]@ \\
\qquad code_b @ \\
\qquad [\texttt{goto}\ l_e]@ \\
\quad [(l_x, \texttt{skip})]
\end{array}
$$

$\textbf{trans}_s(\texttt{do}\ stmt\ \texttt{while}\ e) =$

Others

Declarations:

$$\begin{array}{rcl} \textbf{trans}_d(\texttt{int}\ x) & = & [x = 0] \\ \textbf{trans}_d(\texttt{int}[n]\ x) & = & [x = \texttt{alloc}(n)] \end{array}$$

Blocks:

$$\textbf{trans}_b(d_1, \ldots, d_n\ s_1, \ldots, s_m) = \\ \textbf{trans}_d(d_1) @ \cdots @ \textbf{trans}_d(d_n) @ \textbf{trans}_s(s_1) @ \cdots @ \textbf{trans}_s(s_m)$$

Summary

Every automatic translation from language S to T is done *recursively* on the structure of the source language S, while preserving some *invariant* during the translation.

Exercise

- The source language: $E \to n \mid -E \mid E + E$
- The target language:

$$\begin{array}{rcll} C & \to & \epsilon & \\ & \mid & \texttt{push}\ n.C & (n \in \mathbb{Z}) \\ & \mid & \texttt{add}.C & \\ & \mid & \texttt{rev}.C & \end{array}$$

Exercise

A C program is executed by a "stack machine":

Stack	Command
	push **1**.push **2**.add.rev
1	push **2**.add.rev
2.1	add.rev
3	rev
−3	

Execution rules:

$$\begin{array}{rcll} \langle S, \text{push } n.C\rangle & \rightarrow & \langle n.S, C\rangle & \\ \langle n.S, \text{pop}.C\rangle & \rightarrow & \langle S, C\rangle & \\ \langle n_1.n_2.S, \text{add}.C\rangle & \rightarrow & \langle n.S, C\rangle & (n = n_1 + n_2) \\ \langle n, S, \texttt{rev}.C\rangle & \rightarrow & \langle -n.S, C\rangle & \end{array}$$

Exercise

Define the translation rule:

$$\textbf{trans} : E \rightarrow C$$

while preserving the invariant:

$$\forall e \in E.\ (S, \textbf{trans}(e)) \rightarrow^{*} (n.S, \epsilon) \qquad (n \text{ is the value of } e)$$