Script generated by TTT

Title: Petter: Compiler Construction (02.07.2020) - 53: Conditional Statements

Date: Thu Jul 02 09:48:06 CEST 2020

Duration: 20:44 min

Pages: 11

Chapter 3:
Statements and Control Structures

About Statements and Expressions

General idea for translation: $\text{code}^i\ s\ \rho$: generate code for statement s
$\text{code}^i_{\text{R}}\ e\ \rho$: generate code for expression e into R_i

Throughout: $i, i+1, \ldots$ are free (unused) registers

For an *expression* $x = e$ with $\rho\ x = a$ we defined:

$$\text{code}^i_{\text{R}}\ x = e\ \rho \quad = \quad \text{code}^i_{\text{R}}\ e\ \rho$$
$$\text{move}\ R_a\ R_i$$

However, $x = e;$ is also an *expression statement*:

About Statements and Expressions

General idea for translation: $\text{code}^i\ s\ \rho$: generate code for statement s
$\text{code}^i_{\text{R}}\ e\ \rho$: generate code for expression e into R_i

Throughout: $i, i+1, \ldots$ are free (unused) registers

For an *expression* $x = e$ with $\rho\ x = a$ we defined:

$$\text{code}^i_{\text{R}}\ x = e\ \rho \quad = \quad \text{code}^i_{\text{R}}\ e\ \rho$$
$$\text{move}\ R_a\ R_i$$

However, $x = e;$ is also an *expression statement*:

- Define:

$$\text{code}^i\ e_1 = e_2;\ \rho \quad = \quad \text{code}^i_{\text{R}}\ e_1 = e_2\ \rho$$

The temporary register R_i is ignored here. More general:

$$\text{code}^i\ e;\ \rho = \text{code}^i_{\text{R}}\ e\ \rho$$

About Statements and Expressions

General idea for translation: $\text{code}^i\ s\ \rho$: generate code for statement s

$\text{code}^i_{\text{R}}\ e\ \rho$: generate code for expression e into R_i

Throughout: $i, i+1, \ldots$ are free (unused) registers

For an *expression* $x = e$ with $\rho\ x = a$ we defined:

$$\begin{aligned} \text{code}^i_{\text{R}}\ x = e\ \rho &= \text{code}^i_{\text{R}}\ e\ \rho \\ &\quad \text{move}\ R_a\ R_i \end{aligned}$$

However, $x = e;$ is also an *expression statement*:

- Define:

$$\text{code}^i\ e_1 = e_2;\ \rho = \text{code}^i_{\text{R}}\ e_1 = e_2\ \rho$$

The temporary register R_i is ignored here. More general:

$$\text{code}^i\ e;\ \rho = \text{code}^i_{\text{R}}\ e\ \rho$$

- Observation: the assignment to e_1 is a side effect of the evaluating the expression $e_1 = e_2$.

Translation of Statement Sequences

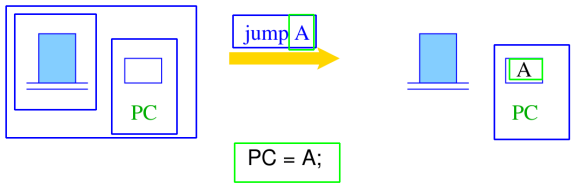
The code for a sequence of statements is the concatenation of the instructions for each statement in that sequence:

$$\begin{aligned} \text{code}^i\ (s\ ss)\ \rho &= \text{code}^i\ s\ \rho \\ &\quad \text{code}^i\ ss\ \rho \\ \text{code}^i\ \varepsilon\ \rho &= \quad // \quad \textit{empty sequence of instructions} \end{aligned}$$

Note here: s is a statement, ss is a sequence of statements

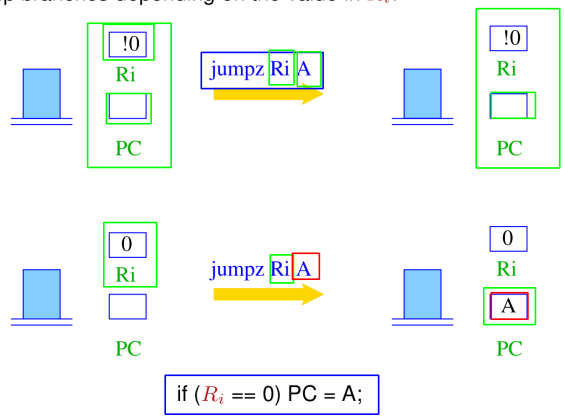
Jumps

In order to diverge from the linear sequence of execution, we need *jumps*:



Conditional Jumps

A conditional jump branches depending on the value in R_i:



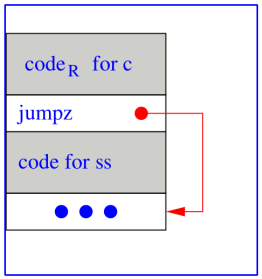
Simple Conditional

We first consider $s \equiv$ **if** (c) ss.
...and present a translation without basic blocks.

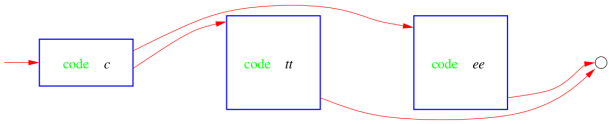
Idea:

- emit the code of c and ss in sequence
- insert a jump instruction in-between, so that correct control flow is ensured

$\text{code}^i\ s\ \rho$ = $\text{code}^i_R\ c\ \rho$; jumpz R_i A ; $\text{code}^i\ ss\ \rho$; A : ...



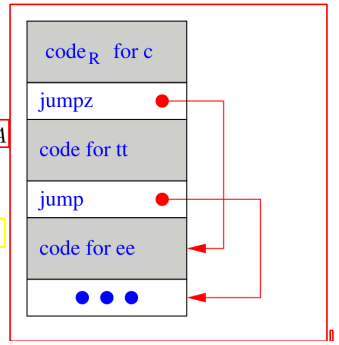
General Conditional



Translation of **if** (c) tt **else** ee.

code^i **if**(c) tt **else** $ee\ \rho$ =

$\text{code}^i_R\ c\ \rho$; jumpz R_i A ; $\text{code}^i\ tt\ \rho$; jump B ; A : $\text{code}^i\ ee\ \rho$; B :



Example for if-statement

Let $\rho = \{x \mapsto 4, y \mapsto 7\}$ and let s be the statement

```
if (x>y) {          /* (i) */
  x = x - y;        /* (ii) */
} else {
  y = y - x;        /* (iii) */
}
```

Then $\text{code}^i\ s\ \rho$ yields:

Example for if-statement

Let $\rho = \{x \mapsto 4, y \mapsto 7\}$ and let s be the statement

```
if (x>y) {          /* (i) */
  x = x - y;        /* (ii) */
} else {
  y = y - x;        /* (iii) */
}
```

Then $\text{code}^i\ s\ \rho$ yields: