**Script** **generated by TTT**

Title: Petter: Compiler Construction (02.07.2020)
- 55: Switch Statements

Date: Thu Jul 02 11:03:52 CEST 2020

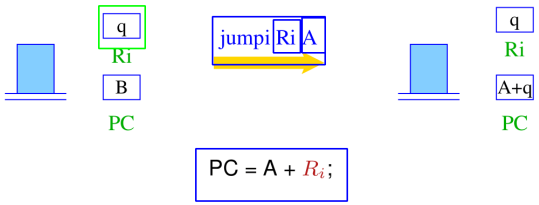
Duration: 18:08 min

Pages: 10

## The switch-Statement

Idea:

- Suppose choosing from multiple options in *constant time* if possible
- use a *jump table* that, at the $i$th position, holds a jump to the $i$th alternative
- in order to realize this idea, we need an *indirect jump* instruction






## The switch-Statement

Idea:

- Suppose choosing from multiple options in *constant time* if possible
- use a *jump table* that, at the $i$th position, holds a jump to the $i$th alternative
- in order to realize this idea, we need an *indirect jump* instruction



## Consecutive Alternatives

Let **switch** $s$ be given with $k$ consecutive **case** alternatives:

```
switch (e) {
  case 0: s_0; break;
  ⋮
  case k − 1: s_{k−1}; break;
  default: s_k; break;
}
```

## Consecutive Alternatives

Let **switch** $s$ be given with $k$ consecutive **case** alternatives:

```
switch (e) {
  case 0: s_0; break;
  ⋮
  case k − 1: s_{k−1}; break;
  default: s_k; break;
}
```

Define $\text{code}^i\ s\ \rho$ as follows:

$$
\begin{array}{lll}
\text{code}^i\ s\ \rho = & \text{code}^i_{\text{R}}\ e\ \rho & \\
& \mathit{check}^i\ 0\ k\ B & \\
A_0: & \text{code}^i\ s_0\ \rho & \\
& \text{jump}\ C & \\
\vdots & \vdots & \\
A_k: & \text{code}^i\ s_k\ \rho & \\
& \text{jump}\ C & \\
B: & \text{jump}\ A_0 & \\
\vdots & \vdots & \\
& \text{jump}\ A_k & \\
C: & &
\end{array}
$$



## Consecutive Alternatives

Let **switch** $s$ be given with $k$ consecutive **case** alternatives:

```
switch (e) {
  case 0: s_0; break;
  ⋮
  case k − 1: s_{k−1}; break;
  default: s_k; break;
}
```

Define $\text{code}^i\ s\ \rho$ as follows:

$$
\begin{array}{lll}
\text{code}^i\ s\ \rho = & \text{code}^i_{\text{R}}\ e\ \rho & \\
& \mathit{check}^i\ 0\ k\ B & \\
A_0: & \text{code}^i\ s_0\ \rho & \\
& \text{jump}\ C & \\
\vdots & \vdots & \\
A_k: & \text{code}^i\ s_k\ \rho & \\
& \text{jump}\ C & \\
B: & \text{jump}\ A_0 & \\
\vdots & \vdots & \\
& \text{jump}\ A_k & \\
C: & &
\end{array}
$$

$\mathit{check}^i\ l\ u\ B$ checks if $l \leq R_i < u$ holds and jumps accordingly.



## Translation of the $\mathit{check}^i$ Macro

The macro $\mathit{check}^i\ l\ u\ B$ checks if $l \leq R_i < u$. Let $k = u - l$.

- if $l \leq R_i < u$ it jumps to $B + R_i - l$
- if $R_i < l$ or $R_i \geq u$ it jumps to $A_k$

$$
\begin{array}{ll}
B: & \text{jump}\ A_0 \\
\vdots & \vdots \\
& \text{jump}\ A_k \\
C: &
\end{array}
$$



## Translation of the $\mathit{check}^i$ Macro

The macro $\mathit{check}^i\ l\ u\ B$ checks if $l \leq R_i < u$. Let $k = u - l$.

- if $l \leq R_i < u$ it jumps to $B + R_i - l$
- if $R_i < l$ or $R_i \geq u$ it jumps to $A_k$

we define:

$$
\begin{array}{lll}
\mathit{check}^i\ l\ u\ B = & \text{loadc}\ R_{i+1}\ l & \\
& \text{geq}\ R_{i+2}\ R_i\ R_{i+1} & \\
& \text{jumpz}\ R_{i+2}\ E & \\
& \text{sub}\ R_i\ R_i\ R_{i+1} & \\
& \text{loadc}\ R_{i+1}\ k & \\
& \text{geq}\ R_{i+2}\ R_i\ R_{i+1} & \\
& \text{jumpz}\ R_{i+2}\ D & \\
E: & \text{loadc}\ R_i\ k & \\
D: & \text{jumpi}\ R_i\ B & \\
B: & \text{jump}\ A_0 & \\
\vdots & \vdots & \\
& \text{jump}\ A_k & \\
C: & &
\end{array}
$$

## Translation of the $check^i$ Macro

The macro $check^i\ l\ u\ B$ checks if $l \leq R_i < u$. Let $k = u - l$.

- if $l \leq R_i < u$ it jumps to $B + R_i - l$
- if $R_i < l$ or $R_i \geq u$ it jumps to $A_k$

we define:

$$
\begin{array}{lll}
check^i\ l\ u\ B & = & \text{loadc } R_{i+1}\ l \\
 & & \text{geq } R_{i+2}\ R_i\ R_{i+1} \\
 & & \text{jumpz } R_{i+2}\ E \\
 & & \text{sub } R_i\ R_i\ R_{i+1} \\
 & & \text{loadc } R_{i+1}\ k \\
 & & \text{geq } R_{i+2}\ R_i\ R_{i+1} \\
 & & \text{jumpz } R_{i+2}\ D \\
E: & & \text{loadc } R_i\ k \\
D: & & \text{jumpi } R_i\ B
\end{array}
\qquad
\begin{array}{ll}
B: & \text{jump } A_0 \\
 \vdots & \vdots \\
 & \text{jump } A_k \\
C: &
\end{array}
$$

Note: a jump $\text{jumpi } R_i\ B$ with $R_i = u$ winds up at $B + u$, the default case



## Improvements for Jump Tables

This translation is only suitable for *certain* **`switch`**-statement.

- In case the table starts with $0$ instead of $u$ we don't need to subtract it from $e$ before we use it as index
- if the value of $e$ is guaranteed to be in the interval $[l, u]$, we can omit $check$



## General translation of switch-Statements

In general, the values of the various cases may be far apart:

- generate an **`if`**-ladder, that is, a sequence of **`if`**-statements
- for $n$ cases, an **`if`**-cascade (tree of conditionals) can be generated $\rightsquigarrow O(\log n)$ tests
- if the sequence of numbers has small gaps ($\leq 3$), a jump table may be smaller and faster
- one could generate several jump tables, one for each sets of consecutive cases
- an **`if`** cascade can be re-arranged by using information from *profiling*, so that paths executed more frequently require fewer tests