Compiler Design in C - Chapter 6.9 (Statements and Control Flow)

Presented by Jiří Šálek and Petr Mrázek

1. Unconditional jump

- **Prerequisite for everything else**
- **Simple ☻**

```
begin {This example uses unconditional jump}
  goto skok;
    showmessage('Zprava1'); // This is dead code
  skok:
    showmessage('Zprava2');
end;
```

2. Return

- **A jump to the end of the current function**
- **Optionally sets a return variable**

```
Result := 5;
Exit;
{* This is equivalent to C++ 'return 5;' *}
Result := 0; // Dead code
```

```
mov [Result], 5                ; Result := 5;
jmp END                        ; Exit;
xor eax,eax                    ; Result := 0
mov [Result],eax
END: mov eax,[Result]

… End; - stack cleanup
```

3. Conditionals - If

- **Evaluate a list of instructions**
- **Only if the condition is true**

```
a := 1; b := 2;
if (a < b) then showmessage('zprava');
```

```
mov eax, [a]
cmp eax, [b]
jnl SKIP
mov eax,'zprava'
call ShowMessage
SKIP:
```

4. Conditionals – If/Else

- **Evaluate a different list of instuctions when the condition isn't true**
- **Can be nested and chained**

```
a:=1; b:=2;
if (a < b) then showmessage('zprava')
else showmessage('zprava2');
```

```
mov eax, [a]
cmp eax, [b]
jnl DoElse
mov eax,'zprava1'
call ShowMessage
jmp SKIP
```

```
DoElse: mov
eax,'zprava2'
call ShowMessage
SKIP:
```

5. Cycles – While

- **Simplest cycle**
- **While a condition is true, repeat**

```
i:=0;
while (i < 5) do begin
  showmessage(IntToStr(i));
  inc(i); // C++ equivalent: i++;
end;
```

```
i:=0;
again:
if (i < 5) then begin
  showmessage(IntToStr(i));
  inc(i);
  goto again;
end;
```

6. Cycles – Continue

- **Skip to the next iteration – basically a jump**

```
i:=0;
while (i < 5) do Begin
  inc(i);
  if (i < 5) then continue;
  showmessage('Zprava');
End;
```

```
i:=0;
pokracuj:
if (i < 5) then Begin
  inc(i);
  if (i < 5) then goto pokracuj;  // continue
  showmessage('Zprava');
  goto pokracuj;
End;
```

7. Cycles – Break

- **Jump out of the cycle, to its end**

```
while (i < 1000) do begin
  inc(i);
  if i=5 then begin
    showmessage(IntToStr(i)); break;
  end;
end;
```

```
znovu:
if (i < 1000) then begin
  inc(i);
  if (i = 5) then begin
    showmessage(IntToStr(i)); goto konec; // break
  end;
  goto znovu;
end;
konec:
```

8. Cycles – For

- **Iterates over a range, syntactic sugar**
- **Convenient**
- **C variant of for is more powerful**

```
for i:=1 to 5 do showmessage(IntToStr(i));
```

```
i := 1; // initialization
opakuj:
if (i <= 5) then // condition
begin
  showmessage(IntToStr(i)); // body
  inc(i); // appended code
  goto opakuj;
end;
```

9. Switch/case – simple

- **Evaluate lists of instructions based on an integer**
- **Compiler uses jumps and decrements for less than 5 cases**

```
case i of
  0: showmessage('Zprava 1');
  1: showmessage('Zprava 2');
  2: showmessage('Zprava 3');
  3: showmessage('Zprava 4');
end;
```

9. Switch/case – simple

- **Example: Compiler uses jumps and decrements for less than 5 cases**
- **Could also check for each possible value directly**

```
int i;
if(i<0)
  goto finish;
i--;
if(i<0)
  showmessage('Zprava 1');
  goto finish;
else if(i==0)
  showmessage('Zprava 2');
  goto finish;
i--;
if(i==0)
  showmessage('Zprava 3');
  goto finish;
i--;
if(i==0)
  showmessage('Zprava 4');
finish:
```

10. Switch/case – jump table

- **Evaluate blocks of instructions based on an integer**
- **Compiler uses a jump table for >= 5 cases**

```
case i of
  0: showmessage('Zprava 1');
  1: showmessage('Zprava 2');
  2: showmessage('Zprava 3');
  3: showmessage('Zprava 4');
  4: showmessage('Zprava 5');
end;
```

10. Switch/case – jump table

- **A table of addresses where instruction blocks start**
- **One is picked by looking up the table by index**

```
int i;
if(i<0 || i > max) // check boundaries
  goto finish;
goto [jumptable + i*sizeof(ptr)]
ptr jumptable = [addr1, addr2, addr3, …];
addr1:
  some code
  goto finish;
…
finish:
```