how while loop are translated into code

Menghao Han

May 2, 2015

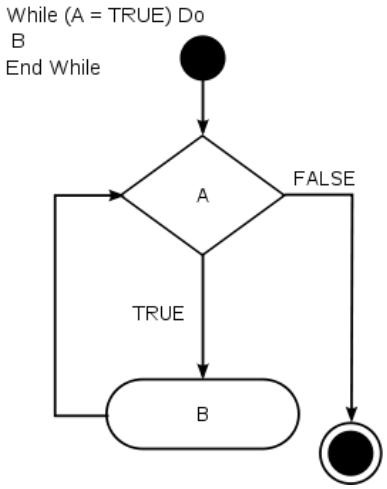
Overview

While loop is a very important part in most computer programming languages. A while loop is a control flow statement that allows code to be executed repeatedly based on a given boolean condition. The while loop can be thought of as a repeating if statement.

This paper will discuss how to translated while loop into UMachine code.

How to build while loop

The main part of while is build as figure 1. The while construct consists of a block of code and a condition/expression. The condition/expression is evaluated, and if the condition/expression is true, the code within the block is executed. This repeats until the condition/expression becomes false. Because the while loop checks the condition/expression before the block is executed, the control structure is often also known as a pre-test loop. Compare this with the do while loop, which tests the condition/expression after the loop has executed.



As we discuss before, the main idea of while loop is jump back and condition check.

For example if we want to translate program3 in lab 10

```
program Program3;
    var Sum: Integer;
    I: Integer;
    N: Integer;

begin Read(N);
    I := 1;
    Sum := 0;
    while (I ≤ N) or (I > 100) do begin
    Sum := Sum + I;
    I := I + 1;
    Write(I, Sum)
    end
    end.
```

The Umachine code will become

```
;Build the symbol table
    MOV SP D0
    PUSH SP
    PUSH ♯3
    ADDS
    POP SP

RD 2(D0)

MOV ♯1 1(D0)
    MOV ♯0 0(D0)

;while
    L1:

;loop condition
    PUSH 1(D0)
    PUSH 2(D0)
    CMPLES
    BRTS L3
    ;If I ≤ N jump to loop body

;or
    PUSH 1(D0)
    PUSH ♯100
```

```
    CMPGTS
    BRFS L0
    ;If I > 100 end the loop

L3:
    ;loop body

PUSH 0(D0)
    PUSH 1(D0)
    ADDS
    POP 0(D0)

PUSH 1(D0)
    PUSH #1
    ADDS
    POP 1(D0)

WRT 1(D0)
    WRT #" "
    WRTLN 0(D0)

BR L1
    ;jump back check condition again

;end the loop
    L0:

MOV D0 SP
    HLT
```

As we see above, first we push a label L1 for jump back, then check the loop condition. After condition check we push one jump point BRFS L0 which means if condition check fail we go to L0 end the loop. If not, we keep going do the loop body, at the end of loop body, we use BR L1 to back to title, do condition check again

Code

When we find while use this statement, first write L for jump label. RepeatLabel is a variable count the number of label we used.

```
void SemanticAnalyser::whileStatementPrecondition(int& repeatLabel)
{
        if (ENABLE_DEBUGGING)
                _outFile << "\n; Start 'while' statement \n";
        repeatLabel = getNextLabelVal();
        _outFile << "L" << repeatLabel << ":\n";
}
```

After write the label we keep write condition part and the loop body.

the end of condition check called whileStatementPostcondition, we push BRFS L, so that if condition check return false, jump to exit label end the loop.

```
void SemanticAnalyser::whileStatementPostcondition(int& exitLabel)
{
        //if false exit
        exitLabel = getNextLabelVal();
        _outFile << "BRFS L" << exitLabel << " \n";
        if (ENABLE_DEBUGGING)
                _outFile << "; end 'while' condition \n";

        //exitLabel = getNextLabelVal();
        //_outFile << "L" << exitLabel << ":\n";
}
```

Then we use this statement to build the loop. Jump back to while condition check and set the exit label. Then push L exit for end the loop, if we want to end the loop, we will just jump to this label.

```
void SemanticAnalyser::whileStatementPostbody(int repeatLabel, int exitLabel)
{
        _outFile << "BR L" << repeatLabel << " \n";
        _outFile << "L" << exitLabel << ":\n";

        if (ENABLE_DEBUGGING)
                _outFile << "; end 'while' loop \n\n";
}
```