Coursework 4

Sunday, November 24, 2019 3:03 PM

Q1:
You need to lex and parse WHILE programs, and then generate Java Byte Code instructions for the Jasmin assembler (or Krakatau assembler). As solution you need to submit the assembler instructions for the Fibonacci and Factorial programs. Both should be so modified that a user can input on the console which Fibonacci number and which Factorial should be calculated. The Fibonacci program is given in Figure 1. You can write your own program for calculating factorials. Submit your assembler code as a file that can be run, not as PDF-text.

A1: See attached files fib.j and fact.j

Q2:
Extend the syntax of your language so that it contains also for-loops, like

```
for Id := AExp upto AExp do Block
```

The intended meaning is to first assign the variable Id the value of the first arithmetic expression, test whether this value is less or equal than the value of the second arithmetic expression. If yes, go through the loop, and at the end increase the value of the loop variable by 1 and start again with the test. If no, leave the loop. For example the following instance of a for-loop is supposed to print out the numbers 2, 3, 4.

```
for i := 2 upto 4 do {
        write i
}
```

There are two ways how this can be implemented: one is to adapt the code generation part of the compiler and generate specific code for for-loops; the other is to translate the abstract syntax tree of for-loops into an abstract syntax tree using existing language constructs. For example the loop above could be translated to the following while-loop:

```
i := 2;
while (i <= 4) do {
        write i;
        i := i + 1;
}
```

A2: I chose to generate fresh assembly code, but it ended up being more or less the same as if I would have generated a different abstract syntax tree structure and then compiled that. Oh well.

Q3:
In this question you are supposed to give the assembler instructions for the program

```
for i := 1 upto 10 do {
        for i := 1 upto 10 do {
        write i
        }
}
```

Note that in this program the variable i is used twice. You need to make a decision how it should be compiled? Explain your decision and indicate what this program would print out.

A3: Well, there are three ways I can see of approaching this problem. First one would be to implement
a variable scoping system for the while language. This would be way out of scope for this coursework.
Another option would be to throw an error and just not compile that kind of code, as it is ambiguous.
This is still way to complicated in my humble opinion so I chose the third option: put all the blame of
the ambiguity on the programmer, ignore the issue completely and treat the two i's as if they are referencing
the same variable. As such, in my implementation, the program will just print out 1 2 3 … 10 and then stop.

In depth explanation:

Control flows into first loop, i is initialized to 1.
Control flows into second loop, i is initialized (again) to 1. Loops finishes executing normally. i is now 10.
All numbers from 1 to 10 have been printed out.
Control returns to first loop. i gets incremented to 11. 11 is not less then or equal to then, so control
exits the first loop and the program terminates.