

Homework 8

1. Write a program in the WHILE-language that calculates the factorial function.
2. What optimisations could a compiler perform when compiling a WHILE-program?
3. What is the main difference between the Java assembler (as processed by Jasmin) and Java Byte Code?
4. Remember symbolic labels in the Jasmin-assembler are meant to be used for jumps (like in loops or if-conditions). Assume you generated a Jasmin-file with some redundant labels, that is some labels are not used in your code for any jumps. For example L_begin and L_end are not used in the following code-snippet:

```
L_begin:
ldc 1
ldc 2
ldc 3
imul
ldc 4
ldc 3
isub
iadd
iadd
L_end:
```

Do these redundant labels affect the size of the generated JVM-code? (Hint: What are the labels translated to by the Jasmin-assembler?).

5. Consider the following Scala snippet. Are the two functions is_even and is_odd tail-recursive?

```
def is_even(n: Int) : Boolean = {
  if (n == 0) true else is_odd(n - 1)
}

def is_odd(n: Int) : Boolean = {
  if (n == 0) false
  else if (n == 1) true else is_even(n - 1)
}
```

Do they cause stack-overflows when compiled to the JVM (for example by Scala)?

6. The JVM method given below is code for the following recursive function

```
def add(x, y) = if x == 0 then y else add(x - 1, y + 1)
```

Rewrite the JVM code such that the recursive call at the end is removed.

```
.method public static add(II)I
.limit locals 2
.limit stack 7
  iload 0
  ldc 0
  if_icmpne If_else_4
  iload 1
  goto If_end_5
If_else_4:
  iload 0
  ldc 1
  isub
  iload 1
  ldc 1
  iadd
  invokestatic defs/defs/add(II)I ;; recursive call
If_end_5:
  ireturn
.end method
```

7. Explain what is meant by the terms lazy evaluation and eager evaluation.
8. **(Optional)** This question is for you to provide regular feedback to me: for example what were the most interesting, least interesting, or confusing parts in this lecture? Any problems with my Scala code? Please feel free to share any other questions or concerns. Also, all my material is ~~erap~~ imperfect. If you have any suggestions for improvement, I am very grateful to hear.

If **you** want to share anything (code, videos, links), you are encouraged to do so. Just drop me an email or send a message to the Forum.