COMP332 – Assignment 3

Dannyil Kucherenko, 45235902

**INTERESTING POINTS**

for loop translateToFrame list concatenation

for loop condition\_left condition\_right variables

array creation type absence (semantic analyser should check type)

for step value should not be none, check it (should be checked in semantic analyser)

***Report structure***

***Introduction*** *– the structure and the aim of the Assignment*

***SECD Machine*** *– basic structure of the machine language*

***Translator –*** *translator methods*

***Logical operators***

***Array***

***for loop***

***Testing*** *– tests for the added translator methods*

***Macros*** *– use of macros for test naming*

***Logical operators***

***Array***

***for loop***

**Introduction**

Given skeleton of the assignment contains

Assignment’s aim is to complete translator cases for logical operators, array handling and for loop handling.

**SECD Machine**

Variable binding can only be done while IClosure()

So code

let a = 1;  
let b = 2;  
let c = 3

would translate into nested IClosure()’s

List(

IInt(1),

IClosure(

None,

List("a"),

List(

IInt(2),

IClosure(

None,

List("b"),

List(

IInt(3),

IClosure(

None,

List("c"),

Nil),

ICall())),

ICall())),

ICall())

**Translator**

**for loop:**

In my opinion, the appropriate way to store machine instruction is to store final list of instructions without any condition cases. For loop contains step value addition which changes depends on the value of step (less or greater than zero). Assignment paper propose to switch between blocks depends on the value (CHECK). However, to make final generated list of instruction cleaned, I implemented two variables (condit

There is a condition, where step value may be incorrect. For example, “from” is larger than “to”, but step is still positive. There are a few similar cases. Readme does not provide information on handling such situation (at least, I could not find any). Thus, I assumed that the handling of these cases is immediate break (no iteration at all). Similar concept is used in other programming languages.

**Test**

**Failure tests**

As it was mentioned in Lintilla documentation, the only fatal error that must be printed is “IndexOutOfBoundException”. This is the only failure test I can think of for the part of this assignment. However, testing resulted that when this error occurs, machine stops without any notices.

Fro example:

let a = array int;  
a += 1;  
print(a!0);  
print(a!1)

result 1

let a = array int;  
a += 1;  
print(a!1);  
print(a!0)

none result

**For loop**

For loop