6CCS3CFL

Coursework

Rui Han Ji Chen
K20027110

Problem 1

```
def compile_stmt(s: Stmt, env: Env) : (String, Env) = s match {
  case Skip => ("", env)
  case Assign(x, a) => {
    val index = env.getOrElse(x, env.keys.size)
    (compile_aexp(a, env) ++ i"istore $index \t\t; $x", env + (x -> index))
  }
  case If(b, bl1, bl2) => {
    val if_else = Fresh("If_else")
    val if_end = Fresh("If_end")
    val (instrs1, env1) = compile_block(bl1, env)
    val (instrs2, env2) = compile_block(bl2, env1)
    (compile_bexp(b, env, if_else) ++
     instrs1 ++
     i"goto $if_end" ++
     l"$if_else" ++
     instrs2 ++
     l"$if_end", env2)
  }
  case While(b, bl) => {
    val loop_begin = Fresh("Loop_begin")
    val loop_end = Fresh("Loop_end")
    val (instrs1, env1) = compile_block(bl, env)
    (l"$loop_begin" ++
     compile_bexp(b, env, loop_end) ++
     instrs1 ++
     i"goto $loop_begin" ++
     l"$loop_end", env1)
  }
  case WriteVar(x) =>
    (i"iload ${env(x)} \t\t; $x" ++
     i"invokestatic XXX/XXX/write(I)V", env)
  case WriteStr(x) =>
    (i"""ldc "$x"""" ++
     i"invokestatic XXX/XXX/writes(Ljava/lang/String;)V", env)
  case Read(x) => {
   val index = env.getOrElse(x, env.keys.size)
   (i"invokestatic XXX/XXX/read()I" ++
    i"istore $index \t\t; $x", env + (x -> index))
  }
}
```

Listing 1: Problem 1

Problem 2

```

case class ForLoop (s: AExp, n: AExp, n1: AExp, bl: Block) extends Stmt

(p"for" ~ AExp ~ p":=" ~ AExp ~ p"upto" ~ AExp ~ p"do" ~ Block)
   .map[Stmt]{ case _ ~ x ~ _ ~ y ~ _ ~ u ~ _ ~ w =>
        ForLoop(x , y, u, w)
   }

 case ForLoop(x, y, u, bl) => {
      val adding = Assign(stringfy(x) , Aop("+", x , Num(1)))
      val (instrs1, env1) = compile_stmt(Assign(stringfy(x) ,y), env)
      val (instrs2, env2) = compile_stmt(While((Bop("<=", x, u)) , bl ++ List(adding)), env1)
      (instrs1 ++ instrs2 ,env2)
  }
```

Listing 2: Problem 2

Problem 3

The way I designed for loop is simulating a while loop with an initial value outside, therefore, the nested for loop will work accordingly since they are using the same variable, and since the inner for loop was ended with a value n of 10, it actually matches with the upto requirement of the outer for loop, therefore, the following output is correct.

```
1
2
3
4
5
6
7
8
9
10
CommandResult(exitCode = 0, chunks = ArraySeq())
```

Listing 3: Output