SWE 514 Computer Systems
Spring 2022
Project (due June 1st)
(This project can be done in groups of at most 2 students)

LLVM (low level virtual machine - http://llvm.org/) is a compiler infrastructure that provides IR (intermediate representation) that can be used to generate code for various target architectures.

$$\begin{tikzcd}\boxed{\text{Source code}} \arrow[r] & \boxed{\text{LLVM IR code}} \arrow[r] & \boxed{\text{Target assembly code}}\end{tikzcd}$$

In this project you will develop a translator called STM2IR that will input assignment statements and expressions (one on each line). STM2IR will generate low level LLVM IR code that will compute and output these statements.

For example, given the following code in file.stm:

```
x1=3
y= 5
zvalue=23+x1*(1+y)
zvalue
k=x1-y-zvalue
k=x1+3*y*(1*(2+5))
k+1
```

STM2IR will output the following IR code:

LLVM IR (file.ll)

```
; ModuleID = 'stm2ir'
declare i32 @printf(i8*, ...)
@print.str = constant [4 x i8] c"%d\0A\00"

define i32 @main() {
  %k = alloca i32
  %x1 = alloca i32
  %y = alloca i32
  %zvalue = alloca i32
  store i32 3, i32* %x1
  %1 = udiv i32 11,2
  store i32 %1, i32* %y
  %2 = load i32* %x1
  %3 = load i32* %y
  %4 = add i32 1,%3
  %5 = mul i32 %2,%4
  %6 = add i32 23,%5
  store i32 %6, i32* %zvalue
  %7 = load i32* %zvalue
  call i32 (i8*, ...)* @printf(i8* getelementptr ([4 x i8]* @print.str, i32 0, i32 0), i32 %7 )
  %9 = load i32* %x1
  %10 = load i32* %y
  %11 = sub i32 %9,%10
  %12 = load i32* %zvalue
  %13 = sub i32 %11,%12
  store i32 %13, i32* %k
  %14 = load i32* %x1
  %15 = load i32* %y
  %16 = mul i32 3,%15
  %17 = add i32 2,5
  %18 = mul i32 1,%17
  %19 = mul i32 %16,%18
  %20 = add i32 %14,%19
  store i32 %20, i32* %k
  %21 = load i32* %k
  %22 = add i32 %21,1
  call i32 (i8*, ...)* @printf(i8* getelementptr ([4 x i8]* @print.str, i32 0, i32 0), i32 %22 )
  ret i32 0
}
```

Please note the following about the IR code:

- LLVM IR uses static single assignment (SSA) based representation. In assignment statements, variables are assigned a value once.

- alloca is used to allocate space for variables and return address of allocation.
- In IR, variables start with % sign.
- The keyword i8, i16 and i32 means 8 bit, 16 bit and 32 bit type respectively.
- The keyword i32 * means 32 bit pointer.
- Variables %i (where i is an integer) are temporary variables.
- The yellow colored code defines the module name and the prototype for the printf output statement. Generate this part as it is shown in the above example.
- The green colored code is for printing the value of a variable using the printf function.
- You can assume only binary operations (+,-,*, /) will be used in the expressions. All variables and operations are integer operations. Division operation gives the quotient.

Commands	**Explanation**
`stm2ir file.stm`	Runs stm2ir on file.stm and produces IR code in file.ll
`lli file.ll`	Runs the llvm interpreter & dynamic compiler. For the above example, this command produces the output: 41 109
`llc file.ll –o file.s`	Invoke llc compiler to produce assembler code
`clang  file.s  -o file.exe`	Compiles assembler code to produce the executable
`./file.exe`	Runs the executable. For the above example, this command produces the output: 41 109

Note that you are implementing only the `stm2ir` program. The others are LLVM commands.
You should prepare a makefile that compiles your source code. Your project will be graded according to the following criteria:

Documentation (written document describing how you implemented your project)	12%
Comments in your code	8%
Implementation and tests	80%

You should submit your project to Moodle and also give a demo of your project.