COMPILERS
REPORT

PAAWAN GUPTA
201501046

Flat-B Programming Language Specifications :-

All the variables have to be declared in the declblock{....} before being used in the codeblock{...}. Multiple variables can be declared in the statement and each declaration statement ends with a semi-colon.

```
declblock{
      int x , y , ar[101];
      int m;
}
codeblock {
      x = y+8;
      for i=1,2,10 {
       //Statements
      }
      for i=1,10 {
      ....
      }

      while expression {
      ....
      }

      if expression {
      ....
      }
      ....

label:     if expression {
      ...
      }
      else {
      ....
      }
//    conditional and unconditional goto
      goto label;
      goto label if expression;
      print "blah...blah", val;  //print with a space after
      println "new line at the end";
 // print a line after every string or variable
      read sum;
      read data[i];
}
```

Expressions are evaluated with arithematic expressions first then evaluates AND and then OR .

Syntax and Semantics

As you can see the parser, program consists of declblock and codeblock.
Declblock consists of vector of variables to be declared.
Codeblock consists of vector of statements.
Statements can be Assignment/For/While/If-Else/If/Goto/Read/Print
Expressions are of type T1 OR T1
Term are of S1 AND S2
S1 are arithmentic expressions with operator precedence /,*,% > +,-

Design of AST

Ast is designed by creating a programhead through which two child are connected. One is declblock which is a vector of Declarations of variables and the other is codeblock which is a vector of statments and so it goes on as explained in Syntax and Semantics by calling the further children of the child node using the accept() function.

Visitor Design Pattern and how it is used

First a Interpreter class is designed which consists of function visits.
When we call a child of a node we call it by using the accept function of the child class which is a public method of the child Ast class.
I have used the return functions in some cases for compiling the goto label and expressions.

Design of Interpreter

Using Visitors pattern I have made a Interpreter where I evaluate the Ast functions in the Interpreter function of that AST class. For that i do traversal on the AST tree. After that I go to node and evaluate that. By this I could create an Interpreter.

Design of LLVM Code Generator

It is similar to the Interpreter design. But in this case I have used LLVM IR builder to create an IR which then can be used with lli or llc to run the Flat-B program.

PerFormance Comparsions

CODE	Interpreter	lli	llc
	Wall clock Time	**Wall clock Time**	**Wall clock Time**
Bubble Sort(N=1000)	0.401s	0.017s	0.019s
Bubble Sort(N=10000)	37.035s	0.289s	0.310s
gcd(N=1000)	0.008s	0.017s	0.018s
gcd(N=10000)	0.037s	0.021s	0.020s
pairsum(N=1000)	0.298s	0.019s	0.020s
pairsum(N=10000)	28.376s	0.145s	0.150s