Compilers report

1) FlatB Programming Language Description

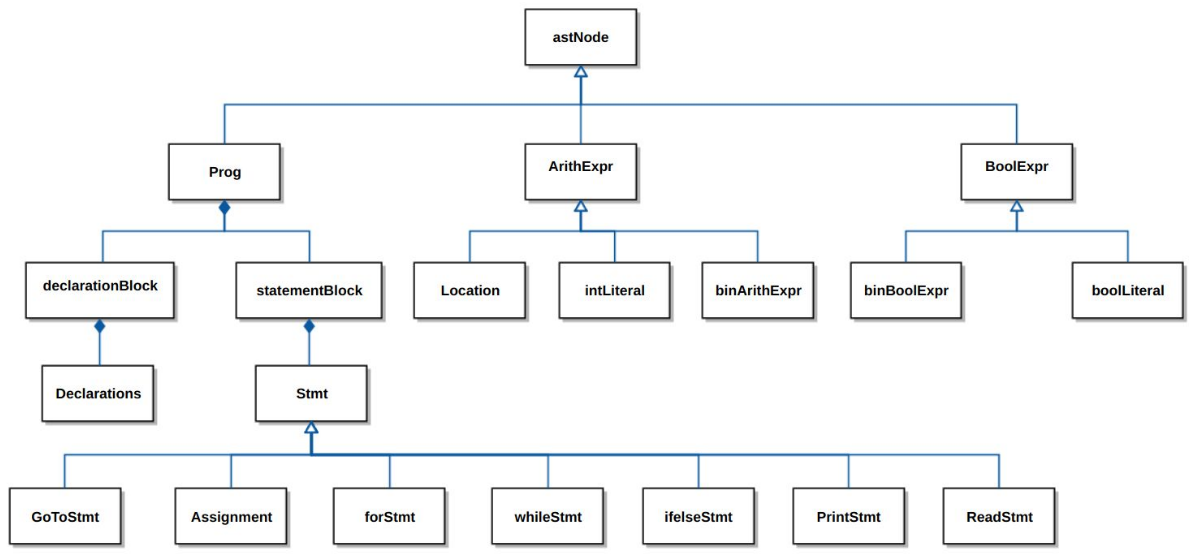
FlatB is a programming language that can execute a subset of statements in C.
There are mainly two kinds of data structures, integer and integer array. A typical code in FlatB is divided into two parts, Declaration and statement blocks.

2) Syntax and Semantics (explain using CFG and examples)

The language is strong typed.
‘For’ loop uses an iterator variable to iterate over the statements till the variable reaches some upper_bound value.
‘While’ loop executes the statements inside till the condition is satisfied.
‘Goto‘ statement can be used to create an unconditional branch to any label in the program.
‘If-Else’ statement is the traditional conditional block.
‘Print‘ statement is used to print the variable or string.
Another variation of this is ‘Println’ thats print a new line after every variable.
‘Read’ statement can be used to read in variables. Since only integers can be scanned. The only input the program can take is integer.
The assignment statement is used to assign values to a variable, on the right hand side of the equal to, there can be a expression or simply another variable.
In the condition, only boolean expressions are accepted.

3) Design of AST

The compiler is divided into three parts. The front end parser, that scans the program file and creates tokens. There tokens are used to generate AST IR middle end as the parsing is going on. Then the AST is used to generate the back end IR in llvm. There is a top most abstract class is called AstNode.
The declarations blocks and statements blocks are the main two partitions in the program. Ever, for loop has its own statement block, similarly in while and if-else statements.



4) Visitor Design Pattern and how it is used.

Visitor Design pattern is used in the interpreter. There are many kinds of ast node classes, and It is generally used when we dont know what is the type of the astnode we are traversing. visitore design pattern is used to traverse across all the nodes in a dfs manner.

5) Design of Interpreter

The interpreter uses visitor design pattern to traverse the nodes of the ast tree and evaluate all the nodes . A Visitor class is defined, that keeps track of all variable allocations in the form of class variables. In a DFS manner, all the nodes are traversed. The major challenge faced was the challenge was in the implementation of goto statement. Because in the interpreter, statements are read and executed at runtime. But, goto statements require that their locations are computed beforehand and when the goto statement is executed, the execution transfers to the label, and the old execution sequence is abandoned. This strategy does not fare well with the stack based traversal methods. So, a pre computation was done to store the labels and it is searched when goto is reached. This causes time delay due to search.
The variables are updated on the go. A string to int map is used to store integers and
String to int vector is used to store the values of array.

6) Design of LLVM Code Generator

The codegen executes in a DFS manner, generating code for the node it visits. Here, llvm IR is generated, by using the api interface of the llvm. This was challenging part because the documentation of of api is very complex to read and understand.

7) Performance Comparison

Program | Type ->	Interpreter	lli	llc
Bubblesort N=1000 O(n^2)	2.5002sec	0.023s	0.006s
Bubblesort N=10000 O(n^2)	4m18.363s	0.237s	0.225s
Bubblesort N=100000 O(n^2)	15+min	28.580s	26.536s
Find the Product of factorials N = 100000 O(n^2)	15+ min	40.704s	47.482s
Fibonacci N = 1000000 O(n)	3.765s	0.020s	0.013s