Compilers Project

Decaf compiler

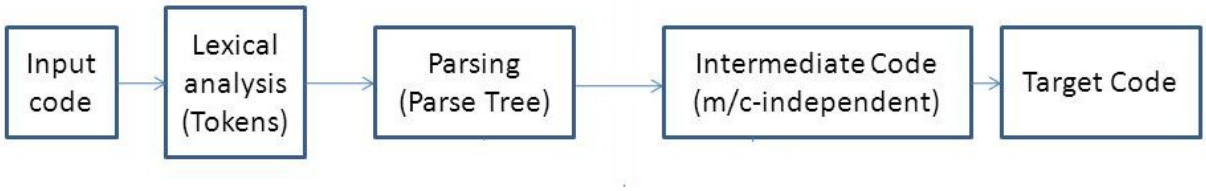
Mohit Sharma
201505508

Phase 4:

Our objective for phase 4 is to generate LLVM IR for the Abstract Syntax Tree (AST) generated in phase 3.

Tools used:

1. **LLVM:** The LLVM compiler infrastructure project (formerly **Low Level Virtual Machine**) is a "*collection of modular and reusable compiler and toolchain technologies*" used to develop compiler front ends and back ends.



Issues faced:

1. It took the most amount of time to understand LLVM API and how to use them as not much examples are available for the same except LLVM documentation on llvm.org. The documentation is very difficult to understand as no examples are mentioned.
2. There are multiple versions available for LLVM. All versions vary a lot in terms of header file's locations. It was also difficult to find that on which version the any sample program will run. Finally I started implementation on 3.4. It took lot of time to make a sample program run.
3. How to store the variable/parameter information i.e. how to implement symbol table was another challenge. Finally used a map to store the information.
4. As we can have nested basic blocks so generating code for such a program required stack implementation.
5. Implementation for if-else-then, for loop was also time consuming as not many sample examples were available for them.

Limitations:

1. Not much error handling is done. It'll report errors like "variable/function not declared". But errors like "type mismatch, incorrect parameters passed to function" aren't checked. If any syntax error is there then it'll print "Syntax error".

References:

1. http://llvm.org/docs/LangRef.html
2. http://llvm.org/docs/tutorial/index.html

Code repository

https://github.com/thegame61916/DecafCompilerProject