# **Assignment#3: Kaleidoscope and MLIR**

**Sai Harsha Kottapalli**

**CS17BTECH11036**

1. Intermediate representation is where the actual source code is represented as “code” used by compiler(internally) with an objective to be optimized or to apply transformation.

It transforms the source code to different target languages.

It acts as an interface for the front-end and back-end parts of compiler.

That is, first the source is successfully transformed to IR with the help of front-end, which is first optimized before one of the corresponding back-end come into play, translating to target machine code.

It therefore, helps in reducing the number of optimizers and code generators for translating multiple source languages to multiple target machines.

Hence, this allows for the support of a lots of target environments.

2. LLVM(Low Level Virtual Machine) IR is a low level language whose objective to provide Intermediate Code which is machine-independent while being available in human readable format.

This ensures that source code from one compiler front end giving LLVM IR which can be compiled to machine code of different machines.

Hence,the optimizer would be common for different languages and different machines.

If most front-ends can give rise to LLVM IR then analysis for LLVM IR can be re-used widely.

MLIR(Multi Level Intermediate Representation) is platform for modern optimizing compilers to identify an IR and apply necessary transformations on it.

It has a flexible type system.

It can obtain different abstract representation from source code of same compilation unit, by representing, analyzing and transforming them.

It’s design philosophy is to provide implementations of compilers from multi-level abstraction of a source code.

3. Lexer: Two variables are used. One to store the identifier name while the other depends if the token is string or number(double).

Tokens defined are follow the same hierarchy.

Parser: The prototypes used for the both the language is almost same(for callee, arguments and body of function).

Based on precedence of left and right side of operators, recursive the functions are called, for the given exoressions.