Salim Kemal Tirit
Bengisu Kübra Takkin

PROJECT DOCUMENTATION

General Process

The project takes the mylang code as an input, converts it into a LLVM code. This means this project is a basic but a strong compiler for mylang codes. During this process, it first takes all the lines from the input and puts them into a vector. After that, it tries to figure out the type of the line and what to do with it. If there is something other than valid statements which are given below there is an error. Our code will generate a LLVM code which prints the line of the error in the input file with an error warning.

1.Assign Statement:

When a line contains "=" code decides that the given line is an assignment statement. After checking the left part is a valid variable, It sends the expression in the right part to the evaluation function, gets a temporary variable and then writes codes to make LLVM store the value in the variable.

2.While/If Loop:

When a line contains "if(" and "while(", the code decides whether It is a if or while loop. Then tries to take the inner statement given. Then the code sends it to the "evaluate" function and gets a temporary variable from the function. Then stores this value as the while/if loop's statement. It also checks whether there is a line that only contains "}" to finish the loop.

3.Print:

When a line contains the word "print", It decides it is a print statement. Then tries to take the inner statement and sends it to evaluate function.

4.Choose:

When an assignment statement's right part contains "choose(" our code concludes that there is a choose function. The Choose function takes four parameters. First parameter is the condition. If it is equal to zero choose returns the second parameter, if it is above zero choose returns third parameter and otherwise returns the fourth parameter. It also allows nested implementation. This was clearly one of the hardest things in this project.

Salim Kemal Tirit
Bengisu Kübra Takkin

Some Important Functions:

a. Postfix:

This function takes the infix expression as an input before its evaluation and turns it to a postfix by using a stack.

b. Evaluate:

Takes the expression as an input, after turning it into a postfix expression by using postfix function, writes one-address LLVM instructions for the evaluation of the expression.

Ups and Downs:

In the project, the algorithm for making a postfix expression is different from the one that is used in the lecture. It does not need extra inner functions and It handles the problem neatly.

In the project, while and if loops are handled in the same part of the code, however, choose function also uses if loops and Its similar structure is rewritten somewhere else and this hinders simplicity of the code.

What Could be the Improvements:

In choose function, the project's code writes three consecutive if loops by using a function to write a single if loop. It has a structure special for the choose function. However, It could be modified a little and could be used for while and if loops too.

There could be other types of variables like string, character or boolean. This would make this project much more complicated.

Also, we did not like to have all variables defined as global. It could be a better choice to have some non-global variables. When we define a variable inside a loop in most of the languages it cannot be used outside the loop. This seems more logical to us.

How We Worked?

It was a painful process. We worked so hard and spent many hours debugging and trying to find some small mistakes. We spoke through discord around 24 hours in total. Could not count the times we worked individually or spoke through other platforms. We pushed our codes to GitHub to keep track of the changes and we made a total of 75 commits. Most importantly we had fun and learned a lot of things throughout this project. We almost forgot that we are not seeing each other for a while because we were working together. Thank you for giving this project to us.