CmpE 230 Spring 2021

Project I Report

Nurlan Dadashov 2019400300

Aziza Mankenova 2018400387

1. Problem description

The given problem requires us to develop a translator called mylang2IR for a language called MyLang. The main purpose is to read input file.my and produce LLVM (low-level virtual machine) IR(intermediate representation) code in file.ll. Therefore, the program should not evaluate the given statements in the input file but should just create an LLVM IR, so that LLVM could correctly output the results. We had to consider the assignment, if, while, print, choose statements, and in these statements, there might be expressions with arithmetic operations, and the correct precedence should be followed.

2. Problem Solution

The first thing we started to consider is how expressions will be evaluated. Arithmetic operations must be executed in accordance with their precedence. The parentheses have the highest precedence, followed by multiplication and division. And addition and subtraction have the lowest precedence. According to BNF, the operations with lower precedence should be stated first, and we followed the same logic. Whenever there is an arithmetic operation, the program starts looking for addition outside of parentheses, as they have the highest precedence and need to be handled at last. Then it recursively calls the addition function on the left part of the “+” sign and on the right part. After that whenever there is no “+” sign left, it goes into a subtraction function and performs similar operations. Then, multiplication and division functions will be called in the given order. Finally, we will be left with either a variable, a number, an expression in parentheses or choose function. If it is an expression in the parentheses, the program calls addition function and repeats the previous steps. If it is a choose function, the respective function is called, which will be described below. If it is a variable, we will just look it up and return its value.

The choose(expr1,expr2,expr3,expr4) function is given four parameters and it returns expr2 if expr1 is equal to 0, returns expr3 if expr1 is positive and returns expr4 if expr1 is negative. For the choose function, we defined a function named “choose” with four parameters inside the file.ll. This way whenever there is a call for the “choose” function, it goes into that function and performs necessary operations. This way is the most convenient since you do not need to write the operations of all the choose functions but write it only once and reuse it for the same calls. Firstly, we evaluate all four expressions of the choose function, and pass calculated values as parameters to the LLVM choose function.

```
define i32 @choose(i32 %$expr1, i32 %$expr2, i32 %$expr3, i32
%$expr4) {
$entry:
      %$retval = alloca i32
      %$0 = icmp ne i32 %$expr1, 0
      br i1 %$0, label %$ne0, label %$e0
$ne0:
      %$1 = icmp slt i32 %$expr1, 0
      br i1 %$1, label %$slt0, label %$sgt0
$slt0:
      store i32 %$expr4, i32* %$retval
      br label %$end
$sgt0:
      store i32 %$expr3, i32* %$retval
      br label %$end
$e0:
      store i32 %$expr2, i32* %$retval
      br label %$end
$end:
      %$2 = load i32* %$retval
      ret i32 %$2
}
```

The execution starts by reading the input and storing all the lines in the Array List, so that later we could easily parse them one by one. While parsing each line, firstly we check for comments, if there are some, we continue parsing only the part before the comment(before “#”). Then, we check for basic errors using the handle method. It checks for the correct number of opening and closing parentheses and the number of “=” signs. Then, we call the parse method. It checks which type of statement is given in a particular line, consequently it calls the appropriate functions for that type of statement. Whenever an error is encountered, the logError method is called, program execution is stopped, and LLVM code which outputs the error is produced.

3. Conclusion

The program executes and gives the expected output. We successfully implemented if, while loop, choose function, print statement, and assignment statements. Overall, the given problem was solved. In the future nested if and while loops could be implemented, and we could also support unary minus.