CS4121 Pascal Junior Compiler Project 4: Function and Function Call

Due Date: Thursday, April 3, 2025 @ 11:59pm

Purpose

The purpose of this project is to gain experience in giving meaning to a programming language by generating x86-64 assembly for a subset of PJ. Specifically, you will be generating assembly for functions and function calls.

Project Summary

In this project, you will add to the compiler developed in the previous project. Specifically, you will need to

1. Differentiate between global and local variables for multiple functions through symbol tables
2. Handle local variables
3. Generate code for multiple functions without parameters
4. Generate code for function calls without parameters, including saving/restoring registers and handling return values of functions.

Requirements

Write all of your code in C or C++ . It will be tested on the Rekhi CS lab machines and MUST work there. You will receive no special consideration for programs which "work" elsewhere.

Input. I have provided my solution to Project 3 in the file `PJProject4.tgz`. Sample input is provided in the directory `PJProject4/input`. I will go over my code for those who wish to use it.
To run your compiler, use the command

```
pjc <file>.pas
```

which will output to `<file>.s`.
To create an executable, run the following command on the assembly file

```
gcc -o <file> <file>.s
```

Submission. Your code should be well-documented. You will submit all of your files, by tarring up the working directory using the command

```
tar -czf PJProject4.tgz PJProject4
```

Submit the file `PJProject4.tgz` via Canvas. Make sure you do 'make clean' of your working directory before executing the tar command. This will remove all of the '.o' files and make your tar file much smaller.

Additional Notes

Test cases 16 through 19 involve function calls to the external functions in `foo.c`. PJ does not support function prototype declaration, so here we assume that all those external functions have no parameters and return an integer. To create an executable for the these three test cases, run the following template command

```
gcc -o <file> <file>.s foo.c
```