Complier Project - Documentation

Adar Agai

Project structure –

cpq.py – The main file, which reads the input and initiates the compiler.

lexer.py – Implements the lexer.

parser.py - Implements the lexer and generates the QUAD.

symbol\_table.py – Implements the symbol table.

quad\_generator.py – Implement the generation of the QUAD commands, functions are called from parser.py

quad\_result.py – Defines the object that the parser returns on each rule, store the generated code for each rule, and when needed, also the resulting value.

consts.py – Defines the constants of the program – QUAD commands etc.

Compiler Flow –

Read the input file and feed it to the lexer.

The lexer tokenize the CPL file, and search for syntax errors, such as illegal characters like ‘@’ for example.

Then, the tokens get fed to the parser, which parses it per the defined rules –

First, the symbol table get populated with the variables, then it continues to generate the code per the rules – In every rule there is a check for errors and if not, it will call the appropriate function from quad\_generator.py to generate the QUAD code ( separate file to better the elegance ) and if needed, also return the calculated resulting value of the rule.

There is an error check during the parsing of course, type mismatch etc.

There is also a syntax error check during the parsing which comes handy in situations where there are no ‘{}’ or excessive ‘{}’ (more ‘{}’ after the first ones with for example).