CPSC 323 Documentation

About 2 Pages

1. Problem Statement

The task is to develop two key components for a compiler targeting a simplified version of the Rat23F language. The first component involves handling a symbol table, requiring the creation of procedures for insertion, checking, and printing of identifiers, each associated with a unique memory address. The second component focuses on generating assembly code compliant with the simplified Rat23F's restrictions, such as no <Function Definitions> and no “real” type, using a provided set of virtual machine instructions. This assembly code should be stored in an array with a minimum capacity for 1000 instructions. The assignment emphasizes error handling for identifier declaration and type checking, and requires adherence to specific language semantics, particularly around boolean values and arithmetic operations.

2. How to use your program

In the project folder launch “assembly.exe”. When in the program the user will be prompted to select a file, for example “test1.txt”. Then the user enters an output file, in this example “output1.txt”. The output file is then stored in the main project folder.

3. Design of your program

The program is an integrated compiler design combining a lexer, symbol table, and parser for source code compilation. The lexer tokenizes the input, categorizing elements into keywords, identifiers, operators, among others, while the symbol table manages these identifiers by assigning unique memory addresses and ensuring error-free declarations. The parser, leveraging the lexer and symbol table, adheres to specific grammar rules to interpret the code, ultimately generating assembly code corresponding to the

source code's high-level instructions. This cohesive system reads input from a file, processes it through these components, and outputs the resulting assembly code and symbol table content, showcasing an efficient blend of lexical analysis, symbol management, and syntactic parsing in the compilation process.

4. **Any limitations**

   The current implementation has some limitations when it comes to loops. It is easy for code to go past the range of the compiler, causing an error.

5. **Any shortcomings**

   The program may still have some issues with certain functions. Although each of our tests worked there may still be certain situations with loops inside functions that break the compiler.