**Peanut Compiler Object Code Generation - CS323 Documentation**

1. **Problem Statement**

At this stage, the compiler focuses on two main aspects: Symbol table handling and generating assembly code. When it comes to the simplified Rat23F, it has no function definitions, and the "real" type is not allowed as well. Symbol table handling includes things such as managing identifiers, lexemes, memory addresses, and types. Assembly code generation is required to translate this simplified code into a directly executable set of instructions for the target VM.

1. **Usage**

***This is a command-line utility.***

1. Using the Command Prompt, navigate to the directory that *peanut-compiler.exe* is located in.

2. Run *peanut-compiler.exe* while passing the path to the Rat23F source file to process as the first argument.

3. See the output in the console to view the results. A copy of this output is also produced in a new file, bearing the name of the old file but with “-out.txt” appended.

Example usage:

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

* To view more information about the accepted arguments, run *peanut-compiler.exe* without any arguments.

A screenshot of a computer program

Description automatically generated

1. **Program design**

The source code can be found at <https://github.com/jiink/peanut-compiler/tree/main/src/>

**3.1. Symbol Table Handling:**

The symbol table stores identifiers, their lexemes, memory addresses, and types. A global variable, currentSymbolTableAddress, is used to assign memory addresses to the identifiers. Procedures are present that check, insert, and print identifiers within the symbol table. Additionally, type matching is enforced to make sure that there is correct usage.

**3.2. Generating Assembly Code:**

The syntax analyzer was modified according to the simplified rules, including the semantic actions required. Assembly instructions are stored in an array as they are generated. When parsing is complete, the content of this array is printed to generate the assembly code listing. The target machine is a virtual machine (VM) with its own set of instructions (e.g., PUSHI, POPM, ADD).

1. **Limitations**

* Identifiers, reals, and integers are limited to 500 characters.
* Syntax errors in the source code may affect the assembly code generation process.
* Several features of Rat23F not supported by the target machine’s instruction set cause no instructions to be generated for them.

1. **Shortcomings**

* “else” statements do not generate properly working instructions.