Program 3: Server Based Language Documentation
Riley Hirn and Josh Johnson

File Structure

This project contains three core files: server.rkt, compiler.rkt, and cminus.rkt.

- server.rkt contains the code to launch the server as well as accept and handle requests, before shutting down. Currently, the server only accepts one connection and shuts down. If the server loops to accept more than one, some values change in the compiled code that causes some instruction to change.
- compiler.rkt contains the code to create the compiler for testing purposes.
- cminus.rkt contains the parser for the C- language. A string of C- code is passed to the function (make) and the function returns a string of compiled code.

How it Works

The server starts and accepts a connection. Once the connection is received, it is passed to the handler, which reads the string passed through the input port. The handle function then calls the compiler and returns the compiled string.
Inside the compiler, the (make) method is called. Using the C- grammar, the C- code is broken down into “tokens”. A tree is made containing the tokens, and is then parsed into machine code.

Commands to Use

- `racket server.rkt`
    - Start the server
- `cat input | nc 127.0.0.1 2112 > output`
    - Send the contents of the file ‘input’ to the server, and put the compiled machine code in the file ‘output’
- `java -cp CPU6502.jar Main output | grep “=>”`
    - Run the compiled code through the 6502 tester and display the code output.

What Doesn’t Work

Some aspects of this program do not work completely.

- The server starts and parses one file correctly, but then causes stack overflows on the consecutive files, which is why it closes after one connection.
- `if` statements work, but no `else` was defined.
- No procedures besides `main` are implemented.
- There is no “<=” or “>=” operators. “>” evaluates as “>=”.