COVER PAGE

PROJECT 3

CPSC 323:

Group 9

- **Joshua Ungheanu**
- **Derek Dorr**
- **Adam Weesner**

Assignment Number **[3]**

Due Dates:

- Hardcopy 12/11 in class by 5:15 pm
- Softcopy 12/11 titanium by 11:55 pm

Executable FileName [CPSC323_ICG_Project3.*exe*]
(A file that can be executed without compilation by the instructor)

Operating System [*Windows 10*]

GRADE:

COMMENTS:

1. Problem Statement

The third assignment is to write a symbol table handler and to generate assembly code for our simplified version of Rat18S. Built from the foundation we used from our previous projects, this project is the culmination of what we learned from the course. For our symbol table handler, we placed every identifier inside the table. Furthermore, each lexeme was held in an entry of the table. Our program also checks if an identifier is already in the table. As for our assembly code generation, we used a simplified Rat18S. Lastly, our program can detect faulty syntax errors and output those errors to the console in detail.

2. How to use your program

Can be done using a terminal from either; a Mac OSX, Linux, or titan server through Putty. Note that for method 2, In order to use the program, you should have your terminal setup to run an executable file. Look for the directory that contains the files to be tested (NOTE: using the terminal requires more steps). Once you have accessed the "Debug" directory, type the following command: cd [filepath] and hit enter, then type:
CPSC323_ICG_Project3*.exe "Test Cases"\<file.txt>*
which contains our source code. Our program will take an input of a .txt path, which will be used to analyze. Note that the .txt must be from the directory which contains our 3 test cases. In order to test more test cases, it is recommended to add those extra ".txt" files into your directory. Once the .txt path have been selected, hit enter and our program will then read the file and write the results to the terminal. The executable file should be working fine on windows OS only and was provided to satisfy the requirements of the assignment.

3. Design of your program

Our program, utilizing the Lexer and Parser functions built from previous projects, seeks to translate simple source code input to assembly code output. Instructions for writing assembly code was kept using an array (capable of holding at least 1000 assembly instructions). As the program scans the input file, it begins separating tokens and lexemes. It will then calculate and generate equivalent assembly code to the console. Error handling was accomplished using a method. If an error was found, it will print the message to the console, then move on to the next code segment. Once all segments are completed, the program will wait for the user to press a key to end.

4. Any Limitation

The source code must be simple or else the assembly does not function properly. Additionally, the program is limited to the ax memory space right now.

5. Any shortcomings

None.

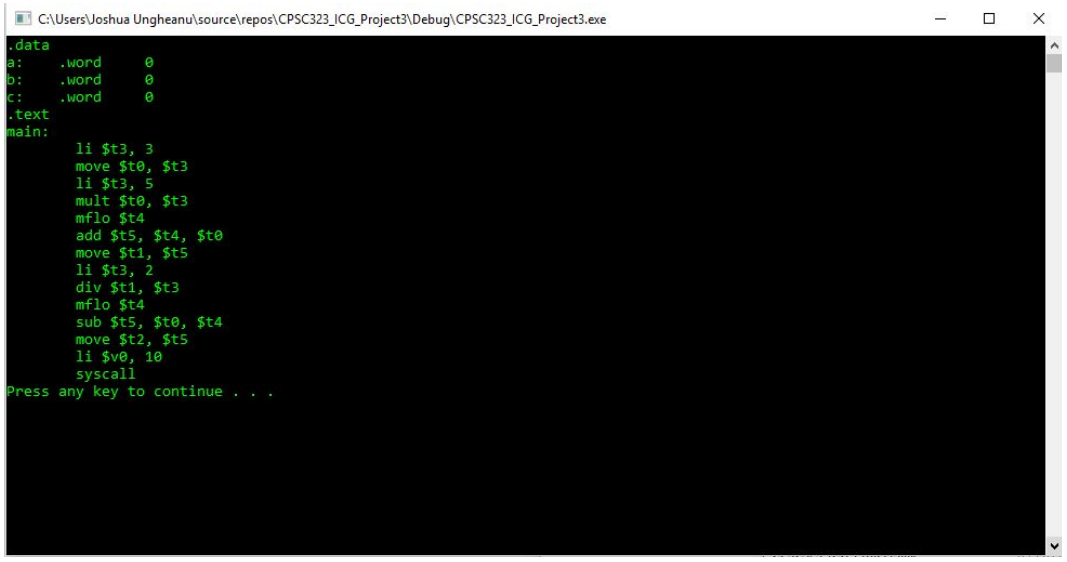
TEST CASES

Case - 1

Input:

```
program
        int a, b;
        int c;
 begin
        a := 3;
        b := a * 5 + a;
        c := a - b / 2;
 end.
```

Output:



C:\Users\Joshua Ungheanu\source\repos\CPSC323_ICG_Project3\Debug\CPSC323_ICG_Project3.exe

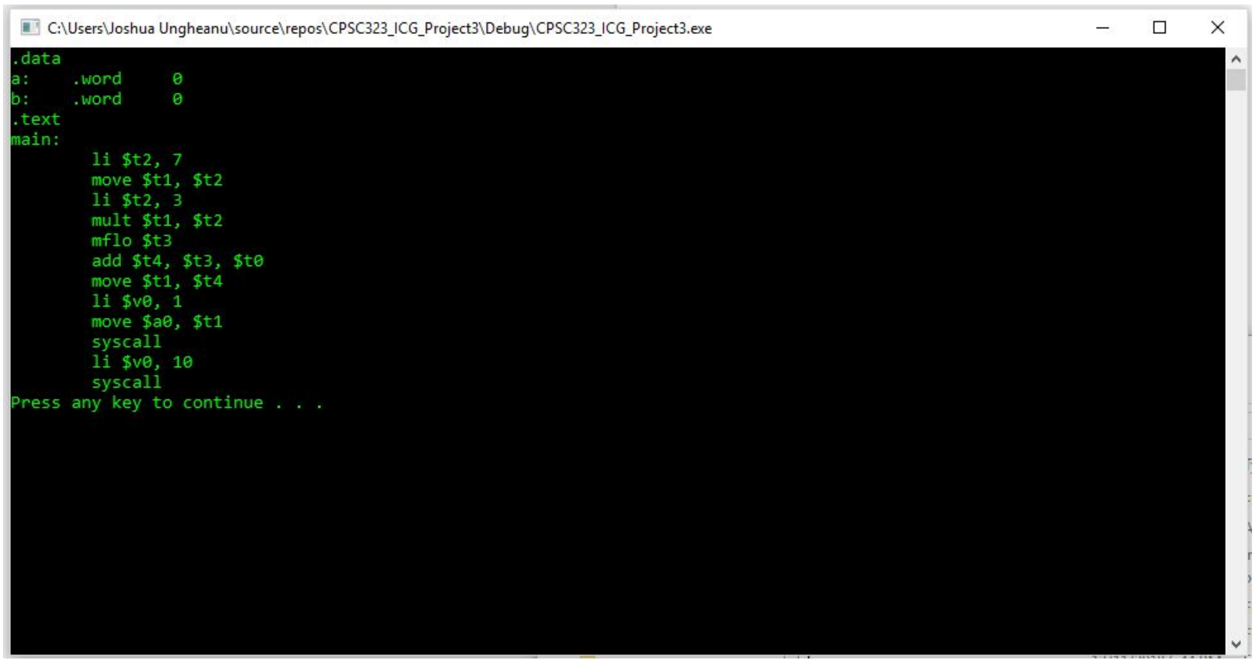
```
.data
a:     .word     0
b:     .word     0
c:     .word     0
.text
main:
        li $t3, 3
        move $t0, $t3
        li $t3, 5
        mult $t0, $t3
        mflo $t4
        add $t5, $t4, $t0
        move $t1, $t5
        li $t3, 2
        div $t1, $t3
        mflo $t4
        sub $t5, $t0, $t4
        move $t2, $t5
        li $v0, 10
        syscall
Press any key to continue . . .
```

Case - 2

Input:

```
program
        int a, b;
 begin
        b := 7;
        b := b * 3 + a;
        write(b);
 end.
```

Output:



C:\Users\Joshua Ungheanu\source\repos\CPSC323_ICG_Project3\Debug\CPSC323_ICG_Project3.exe

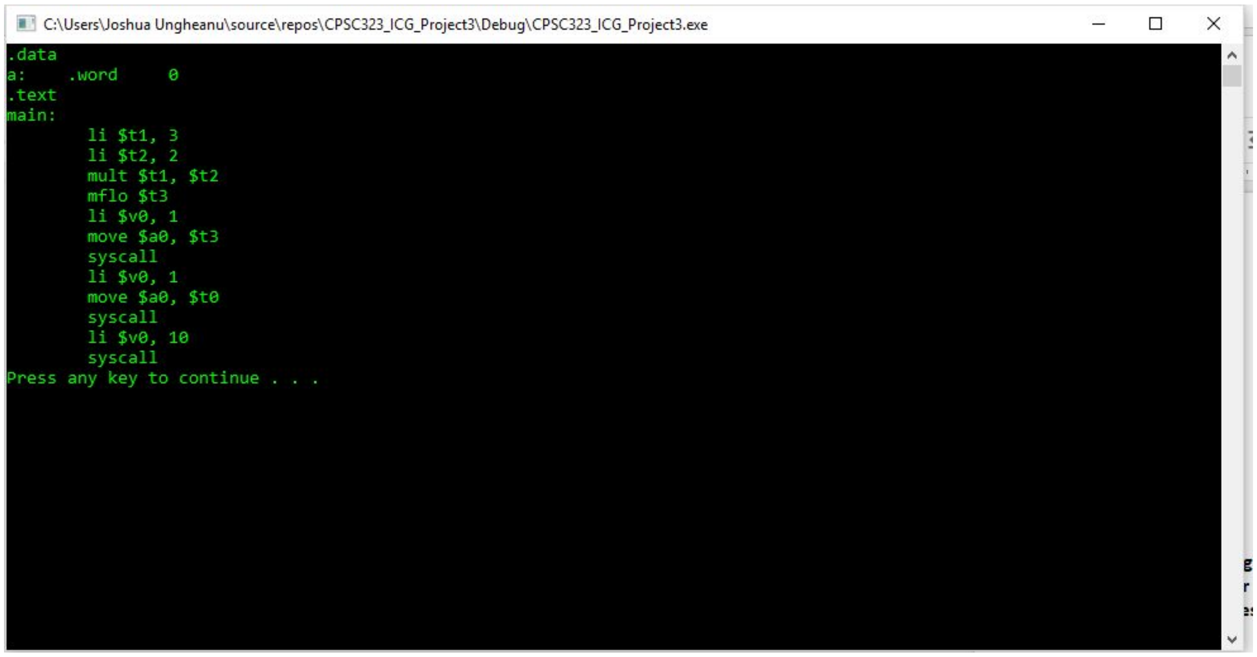
```
.data
a:     .word     0
b:     .word     0
.text
main:
        li $t2, 7
        move $t1, $t2
        li $t2, 3
        mult $t1, $t2
        mflo $t3
        add $t4, $t3, $t0
        move $t1, $t4
        li $v0, 1
        move $a0, $t1
        syscall
        li $v0, 10
        syscall
Press any key to continue . . .
```

Case - 3

Input:

```
program
        int a;
 begin
        write(3*2);
        write(a);
 end.
```

Output:



C:\Users\Joshua Ungheanu\source\repos\CPSC323_ICG_Project3\Debug\CPSC323_ICG_Project3.exe

```
.data
a:     .word     0
.text
main:
        li $t1, 3
        li $t2, 2
        mult $t1, $t2
        mflo $t3
        li $v0, 1
        move $a0, $t3
        syscall
        li $v0, 1
        move $a0, $t0
        syscall
        li $v0, 10
        syscall
Press any key to continue . . .
```