DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Homework Assignment No. 02:

MIPS Assignment #2

submitted to:

Professor Son Nguyen
ECE 4612: Advanced Processor Systems
Temple University
College of Engineering
1947 North 12th Street
Philadelphia, Pennsylvania 19122

October 14, 2024

prepared by:

Leo Berman
Email: leo.berman@temple.edu

A. OBJECTIVES

The purpose of this assignment is to demonstrate basic problem-solving skills and quality assurance skills in MIPS assembly language by performing the open-ended task of writing a command line tool that will check the validity of MATLAB expressions. This task requires parsing the expression character by character and checking a set of conditions to ensure that the expressions would be valid in MATLAB. It's important to note that the only characters that are valid in our MATLAB checker are the following: "(", ")", "+", "-", "*", "/", "0", "1", "2", "3", "4", "5", '6", "7", "8", "9", and " ".

B. TOOLS/EQUIPMENT

The tools used in this assignment are the MARS MIPS Simulator as the IDE and MATLAB to formulate test cases.

C. PROCEDURE

Task 1:

1. Jump to a routine to parse the input
2. Load the MATLAB style prompt for the user to enter an input into the register for print ascii strings
3. Load the instruction to print
4. Call that instruction
5. Load the instruction to take a user's command line input
6. Load the address of where to store this input and the size of the input
7. Call the instruction
8. Return to main
9. Jump to a routine to iterate through the user input
10. Load the address of where the user input is stored
11. Initialize a few registers:
    a. i = 0 loop counter
    b. integer for tracking the sets of parenthesis
    c. Boolean for whether the last character was operable
    d. Boolean for whether that last character was a number
    e. Check if the last character was a "+" or "-"
12. Save the position in main and jump to the inner loop
13. For i = 0;i < 64, i++
    a. Check for end of loop condition, if so check parenthesis balance and to make sure last character is either a closing parenthesis or number, print invalid if so
    b. Load the character
    c. Check if it's a number and if so check if last non space character was operable and not another number. Pint invalid if operable and not a number.
    d. Check if it's a left parenthesis check to make sure it's not an empty pair of parentheses, make sure the last character wasn't operable, if so print invalid
    e. Check if it's a right parenthesis, make sure that there is a corresponding left bracket, and keep track of the remaining left brackets. If either of those fail, print invalid
    f. Check for operators, if it's a plus or a minus, check if the last character is either operable, or a plus/minus, if not then print invalid. If it's a asterisk or slash, make sure last character was operable and that the next character is not an asterisk or slash. If any of those conditions fail, print invalid
    g. Check if it's a space
    h. Check if it's a new line, and if so check parenthesis counter and return if current character is operable
    i. If invalid was printed return to main
    j. Else loop
14. Free the memory used for the string
15. Jump back to the top of main

D. RESULTS

Task 1:

Invalid Symbols	>>> x Invalid input
Empty Parenthesis	>>> () Invalid input

Beginning Parenthesis Operations	>>> (-5)
	Invalid input
	>>> (+5)
	Invalid input
	>>> (*5)
	Invalid input
	>>> (/5)
	Invalid input

Ending Parenthesis Operations	>>> (5*)
	Invalid input
	>>> (5-)
	Invalid input
	>>> (5/)
	Invalid input
	>>> (5+)
	Invalid input

Parenthesis Operations	>>> (3)5
	Invalid input
	>>> 3(5)
	Invalid input

Double Operations	>>> 3-/5
	Invalid input
	>>> 3/-5
	Invalid input
	>>> 3-*5
	Invalid input
	>>> 3*-5
	Invalid input
	>>> 3*+5
	Invalid input
	>>> 3+*5
	Invalid input
	>>> 3/+5
	Invalid input
	>>> 3+/5
	Invalid input
	>>> 5/*3
	Invalid input
	>>> 5*/3
	Invalid input
	>>> 5//3
	Invalid input
	>>> 3**5
	Invalid input

End of Expression Operations	>>> 3/5*
	Invalid input
	>>> 3*5/
	Invalid input
	>>> 5-3+
	Invalid input
	>>> 3+5-
	Invalid input

Tab. 1. Results Table

E. CONCLUSION

Overall, when I started this assignment, I thought it was going to be quick and easy. I got my first implementation done based on my knowledge of the order of operations, and I thought I had cracked the code. However, after hearing about some edge cases from my classmates I realized I had missed a lot of the criteria. From there on, I tested all kinds of different cases on MATLAB and realized that there were a lot if quirks including things like “3+++----+++-+-+-++7” being valid, and “3(5)” and “5(3)” being invalid. After doing a lot of looking back and adding Boolean flags for cases like spaces between numbers I managed to create a product that I was happy with and couldn’t find any test cases I would fail.