**Lab Assignment #3**

**CSCI 4335 Computer Architecture**

**Stack Architecture**

**Dr. Abraham**

**Computers can be designed with stack architecture (stack machines) in which registers are arranged in a stack (there are no named registers in this case except top-of-the-stack). Today’s computers use stack extensively, but use RAM for the stack frames. Several registers are set aside to manage stack. In order to understand stack I am giving you this assignment. The best way to understand stack is to write a program to handle postfix arithmetic. This is going to be a fun program, so enjoy. You can choose any language and any implementation of stack. I used array based stack.**

Write a program to solve a mathematical problem given in postfix notation. Postfix notation is a notation for writing arithmetic expressions in which the operands appear before their operators. The advantage is that there are no precedence rules and parentheses are never needed. We evaluate the expression by scanning from left to right. Consider the problem 6 2 / 5 +. Look for the first operator beginning from left. The division operator is applied to the immediate previous operands. The divisor would be the later one, in this case 2. Now the problem has been reduced to 3 5 +. Continue processing until the end of the problem statement.

Here are some problems with results.

postfix expression result

 \* this description and the samples are

4 5 7 2 + - \* -16 taken from Nell Dale's Book.

3 4 + 2 \* 7 / 2

5 7 + 6 2 - \* 48

Technical notes: The program should accept real numbers as well as integers. You need to enter the problem as a string. The string should be parsed character at a time. Determine if the characters make operands or operators. If operands, convert them to numbers and push into a stack. If an operator, pop and do the required operation. Show all pushes and pops. Write a warning message if there is an error in the problem. Give the result at the end. Here is an example of a program run.



|  |
| --- |
| **Postfix Problem: 4.1 2.2 + 3.9 5.3 1 - \* +** |
| pushed 4.1 |
| pushed 2.2 |
| popped 2.2 4.1 + |
| pushed 6.3 |
| pushed 3.9 |
| pushed 5.3 |
| pushed 1 |
| popped 1 5.3 - |
| pushed 4.3 |
| popped 4.3 3.9 \* |
| pushed 16.77 |
| popped 16.77 6.3 + |
| pushed 23.07 |
| **The result of the operation is--> 23.07** |

**I will be demonstrating a visual stack. If you write the visual part, you will be given 30 extra points (total of 130 points instead of 100).**