Indian Institute of Technology, Delhi

Department of Computer Science and Engineering

COL216: Computer Architecture

Semester ll, 2020-2021

Prof. Preeti Ranjan Panda

Assignment 2

Submitted by:

Kshitiz Bansal, 2019CS50438

Date: *28/2/21*

Problem Statement

Write a MIPS Assembly Program for evaluating an expression in postfix format. Postfix expression with constant integer operands in the range 0-9 and operators +, -, and * ***only***.

Approach:

A postfix expression can be evaluated using an easy to implement algorithm which uses a stack. Pseudocode:

```
Read token-
    If token is an operand, push it onto the stack.
    If token is an operator,
        Pop 2 elements from the stack,
        Use them as operands for the operator,
        Push back the result onto the stack.
```

Design Choices:

1. Input: The input is a postfix expression composed of digits 0-9, and operators +, - and * only.
2. Wrong input: I terminate the program whenever some wrong input is received. Wrong input can be like:
    a. Expression contains operators other than +, - and * like / or %.
    b. Postfix expression is invalid i.e. it is not mathematically correct.

Notable edge cases (Testing):

1. Having 0 as an operand: This case would have been very significant had the operators included / too. For +, - and * only, this case does not cause any major problems. Eg: 3250*-+

2. Wrong input (Extra characters): User may enter wrong input expressions which contain characters like %, ^ etc. So, I terminate the execution whenever such a character is encountered. Eg: 325&+7*

3. Wrong input (Mathematically invalid postfix expression): User may enter a mathematically wrong postfix expression. An expression may be mathematically wrong because it contains too many operands or too many operators. So, I have kept a

track of that, and the execution is terminated whenever it can be deduced that the expression is wrong. Eg: 5487/*916-

4. Overflow: As the input expression will have operands as single digits only, overflowing is very unlikely.
   Will have to input something like 99…{say 10^8 times}..9***…{10^8 – 1 times}..**
   Nonetheless, the execution will end if the values overflow.