COL216 Assignment 2
MIPS Assembly Program for Evaluating Postfix Expressions

Devanshi Khatsuriya 2019CS10344 | Ishita Chaudhary 2019CS10360

Approach:

The input reads a string character by character and stack is used to evaluate the postfix expression as follows:

- If the current character is an integer (from 0-9), push it onto the stack.
- If the current character is an operator (+, - or *) (say ~), pop the top two elements of the stack (say x1 followed by x2), evaluate x2~x1 and push it onto the stack.

Apart from this, error is raised whenever necessary (cases explained in design choices).
At the end, the one integer left in the stack will be the required answer.

Design Choices:

Input Format: Postfix expression with constant integer operands in the range 0 - 9 and operators +, -, and *.

- The user is prompted to enter the Postfix expression, after which the expression should be typed, and the enter key should be pressed to evaluate it. Typing any character other than 0-9 or +, -, * or enter prints the corresponding error message.
- Spaces are ignored, as an exception of the above case.
- While entering the expression, at any point, if the Postfix expression becomes invalid, an error is raised. These cases can be when an operator [+, -, *] is encountered and there aren't at least 2 integers present in the stack, or when the first two characters are not integers from 0-9. An error is also raised if after entering the complete expression, the stack doesn't simplify to contain exactly one integer.
- The inbuilt stack, and the register $sp which contains the address of the top of the stack, are used to provide the stack functionality to evaluate the expression. The register $s1 is used to maintain the current size of the stack.
- After every operation, the size of stack is decreased by one, due to two pop and one push operation.
- The code uses various loop structures in the code like readLoop, multiply, addition and subtract and makes use of conditional and unconditional control instructions to jump across them.

- An input integer [0-9] is converted to its decimal representation by adding -48 to its ASCII Code and then pushed onto the stack. Also, the result of evaluating the Postfix expression is assumed to be a 32-bit integer.

Testing Strategy and Test Cases:

We have tested out code on possible valid and invalid cases as follows:

1. No input case: If enter key is pressed without entering any characters before it, an error is raised. This is correct because an empty postfix expression is considered to be invalid and cannot be evaluated and neither can be reported as zero.

2. Test Case1: Single character input: If the postfix expression consists of only one character, then if that character is an integer from 0-9, then the result printed is that integer itself, otherwise an error is raised because of invalid expression.

3. Test Case 2: Characters outside domain: The input given is 49/31+-. An error is printed as soon as / is typed, saying it is not a valid input expression.

4. Test Case 3: The input given is 325*+. This gives the result to be 13, which is correct.

5. Test Case 4: The input given is 32 5* +. This case is same as above, except the fact there are spaces inserted between the characters. Program will ignore the spaces. The result will still evaluate to 13, which is correct.

6. Test Case 5: The input given is 3+6. An error is printed as soon as + is typed, which is correct, because the stack must contain at least 2 elements when an operator is encountered.

7. Test Case 6: The input given is 2956+-. This prints an error saying that the expression is invalid and number of integers in the expression exceeded, which is correct because there are more than 1 integers left in the stack after evaluating the expression.

8. Test Case 7: This test case is a relatively bigger input 6702+18-5*82--06*+9187299-*+-*+*+-+ evaluates to 421, which is correct.

9. Test Case 8: If the input given is -97+, an error is raised specifying the first two characters of a postfix expression must be integers.