Assignment-2 Submission

Indian Institute of Technology Delhi

COL216: Computer Architecture

Name: Deepanshu **Entry Number: 2019CS50427**
Name: Prashant Mishra **Entry Number: 2019CS50506**

1 Introduction

The assignment is about writing a MIPS Assembly Program for evaluating an expression in postfix format.

2 Approach taken to solve the problem

Following is the step wise description of the approach taken to solve the problem.

2.1 Writing pseudocode

We began by writing a pseudo code which is more inclined towards human understanding and high level programming.
We covered all the possibilities in if-else branching (nesting wherever necessary).
Note: We have provided the pseudocode in the zip file.

2.2 Breakdown of complex expressions

The next step involved breakdown of complex mathematical expressions into multiple line of simpler commands.
For example, consider the following if-else branching:

1. if(s[i] = '+') do
   statements for addition

2. else if(s[i] = '-') do
   statements for subtraction

3. else if(s[i] = '*') do
   statements for multiplication

Wrote these statements as conditional jump/branching instructions. For following statement, assume that $t1 corresponds to s[i] of pseudo code. And addition, subtraction and multiplication points to location in memory where all the corresponding instructions are stored.

1. beq $t1, $s2, addition
2. beq $t1, $s3, subtraction
3. beq $t1, $s4, multiplication

2.3 Error reporting

We threw error in following scenarios:

1. If the char is not an operator('+', '-' or '*') and is also not a numeric character, we throw the invalid character error.
   We are checking this using the ASCII value of digits and these operators. All other ASCII values will give rise to errors. (See point 7 of test cases)
2. Empty expression as input.
   The code gives the message that given postfix expression is wrong.
3. If the size of stack $\neq 1$ in the end of while loop, then the given expression is not a valid postfix expressions.
4. Whenever we encounter an operator and the size of stack is less than 2, we throw an error for invalid postfix expression.
5. If we try to input negative values, for example for the expression (-2 + 3) the C++ code will give postfix as 2-3+. But we are not allowing these kind of responses because only values ranging from 0 to 9 are allowed.
6. If the input contains white spaces, it will also throw error of invalid input.
7. Note that we are considering only binary operators. Uniary operators like square-root and factorial are not allowed. Writing them will also cause error as it will disturb the semantics of the stack.

2.4 Programming in MIPS

After modifying the pseudo code as stated above, we started the actual coding part where we converted statements from above simplified pseudo code into MIPS commands of loading, writing, branching, error reporting etc.

3 Testing Strategy

The testing strategy had 3 layers. Note that we first used the C++ code provided to convert infix expression to postfix expression. We used that postfix expression as an input to further code.

3.1 Testing the pseudo code

In this, we took the pseudo code and programmed it in a high level language (C++). We then tested it against various input and confirmed its correctness. We also stored the values corresponding to some test cases.

3.2 Testing the MIPS code

Here, we took the same test cases that we used in part 1 and matched the output with the output from part 1. This ensured that the code is converted correctly into MIPS language.

3.3 Choosing the test cases

We tested against following possibilities:

1. Addition/Subtraction followed by Multiplication followed by Addition/Subtraction. This case is considered to verify that Multiplication is carried before Addition/Subtraction or not i.e BODMAS rule is being followed or not.

   (a) For Example:
   Postfix Expression: 123*+4+
   Infix Expression: 1+2*3+4   Ans: 11

   (b) For Example:
   Postfix Expression: 232*+
   Infix Expression: 2+3*2   Ans: 8

   (c) For Example:
   Postfix Expression: 24+22*-4+22*-1+3-
   Infix Expression: 2+4-2*2+4-2*2+1-3   Ans: 0

   (d) For Example:
   Postfix Expression: 24-22*+4-22*+1-3+
   Infix Expression: 2-4+2*2-4+2*2-1+3   Ans: 4

2. Multiplication with 0 in between expression.

   (a) For Example:
   Postfix Expression: 20*
   Infix Expression: 2*0   Ans: 0

   (b) For Example:
   Postfix Expression: 23-40*+2-8+
   Infix Expression: 2-3+4*0-2+8   Ans: 5

   (c) For Example:
   Postfix Expression: 23+40*-2+8-
   Infix Expression: 2+3-4*0+2-8   Ans: -1

3. Combining the above two cases:

   (a) For Example:
   Postfix Expression: 23+43*-70*-3+5-7+
   Infix Expression: 2+3-4*3-7*0+3-5+7   Ans: -2

   (b) For Example:
   Postfix Expression: 69+16*-20*-4-2+8-
   Infix Expression: 6+9-1*6-2*0-4+2-8   Ans: -1

4. Case when there are most of the operators are at the end in postfix expression.

   (a) For Example: Postfix Expression: 3456+-*
   Ans: -21

   (b) For Example: Postfix Expression: 0123456789+-*+-*+-*
   Ans: 0

   (c) For Example: Postfix Expression: 1234567890+-*+-*+-*
   Ans: -25

5. Case when the operator is present after every operands except the first one in the postfix expression.

   (a) For Example: Postfix Expression: 12+3-4*5+6-7*8+9-
   Ans: -8

   (b) For Example: Postfix Expression: 01+2-3+4-5*
   Ans: -10

6. Some special corner cases:

   (a) Single operator as input. Eg: Postfix Expression: +
   The code gives the error that given postfix expression is wrong.

   (b) Single digit as input. Eg: Postfix Expression: 1
   The code gives that digit as the output.

   (c) Empty string as input.
   The code gives the error that given postfix expression is wrong.

7. Other cases involving error:

   (a) If count of operand is n, then there should be exactly n-1 operator. In other scenarios, following cases will throw error.

      - Excess count of operator.
        For Example: 23+-+

- Lower count of operator.
  For Example: 234-+2344-

(b) Inserting white spaces in the expression
For Example: Postfix Expression: 2 3 +

(c) Expression involving other operators like division and raise to power operator.
For Example: Postfix Expression: 42/   (/ denotes division operator)

(d) Inserting other characters instead of digits from 0-9.
For Example: Postfix Expression: 1b+

(e) The expression should not start with an operator.
For Example: -23