***Topic 5* : Infix and postfix expressions**

|  |  |
| --- | --- |
| Student name: | Lê Bá Vinh |
| Student ID: | 20184331 |

I. Project description

**1. Topic**

- A program to print postfix expression from infix expression inputted from user then calculate the result with postfix expression.

**2. Requirement**

- Operands (numbers) are integer in range of 0 to 99

- Operators are ‘+’, ‘-‘, ‘\*’, ‘/’

**3. Advanced idea**

- Support ‘(‘ and ‘)’

- Program can ask user to either continue inputting new expression or cancel to exit program.

II. Variables and registers used

**1. Variables**

- Infix with space 256 to store string infix expression from user

- Postfix with space 256 to be a stack of postfix elements from infix expression

- Operator with space 256 to be a stack of operator used when creating postfix stack

- Result with space 256 to be a stack to calculate the result with 2 operands scanned from postfix expression (the result will be the bottom element in the stack)

**1. Registers**

- s3 stores the first number to calculate

- s4 stores the second number to calculate

- s5 is the code of precedence of the checking-operand

- s6 is the code of precedence of the top operand

'+', '-' is 1

'\*', '/' is 2

- s7 is the digit of operand (1 or 2)

- t0, t1, t2 are indexes

- t4 is status code to check error input

0 means that previous scan is operator except '(' and ')'

1 means that previous scan is operand

- t5 stores scanning character in Infix string and Postfix stack or store temporary result when calculating

- t6 stores operator loaded from Operator stack

- t7 stores address holding character in Infix string or Postfix stack

- t8 stores address holding top operand or operator in Postfix stack

- t9 stores address holding top operand in Operator stack

II. Algorithms

**1. Main**

*\* Operators are encoded: operator := operator + 100*

- Get infix expression from user

- Initiate value for registers

- Scan infix expression (operands and operators)

+ Convert number from string to value

+ Check operator

+ If the character is neither a number nor an operator, input is error and rerun the program

+ While operator’s stack is not empty: pop operator’s stack and put it to postfix’s stack

+ If there is ‘(‘ left on the stack, input is error and rerun the program

- Print the postfix expression. After that add ‘\n’ to postfix’s stack

- Calculate the result with postfix’s stack and result’s stack

+ If the divisor (if exists) is 0, the result is error and rerun the program

- Print the result and ask user whether to continue program or not

**2. Convert number from string to value**

- Scan each byte from the infix string to t5 then check it if it was number or not

+ If it is a number, store it to s3 then continue scan (if the number has 2 digits, s3 stores 1st digit, s4 stores 2nd digit)

+ If not, then put the number to s3, push it to postfix’s stack, t4 = 1 then check the operator

**3. Check operator**

- If ‘(‘ is found: push it to operator’s stack

- If ‘)’ is found:

+ While the operator’s stack is not empty AND the top item (t9) is not ‘(‘: pop the operator’s stack and add it to postfix’s stack

+ Pop the operator’s stack (which is ‘(‘) and discard it

- If an operator is found:

+ If the operator’s stack is empty or if the top element is ‘(‘: push it to the operator’s stack

+ If not:

. While the operator’s stack is not empty AND the top of the operator’s stack is not ‘(‘ AND precedence of operator <= precedence of the top of the operator’s stack (s5 <= s6): pop the operator’s stack and put it to postfix’s stack

. Push it to the operator’s stack

**4. Calculate the result**

- Scan the postfix from the start and load it to t5 until reach ‘\n’ character

+ If it is a number (< 100): put it to result’s stack

+ If it is an operator (> 100): decode it and calculate the temporary result from result’s stack with the operator

- After that, the bottom value in result’s stack is the result of the expression and print it.

\* Result can take up to 32bits

III. Simulated Sample Result

- Input an infix expression:

A screenshot of a computer

Description automatically generated

- Show result then ask user:

A screenshot of a computer

Description automatically generated