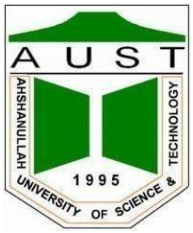

Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No : **CSE2214**
Course Title : **Assembly Language Programming Sessional**
Assignment No : **08**

Date of Performance : **19.08.2020**

Date of Submission : **25.08.2020**

Submitted To : **Ms. Tahsin Aziz & Md. Siam Ansary**

Submitted By-

Group : **A1**
Name : **Mustofa Ahmed**
Id : **18.01.04.005**
Section : **A**

Question No: 01

Write a program that lets the user type some text, consisting of words separated by blanks, ending with a carriage return, and displays the text in the same word order as entered, but with the letters in each word reversed

Answer:

```
.MODEL SMALL
.STACK 100H
.DATA
MSG1 DB 0AH,0DH,'Enter a string : ','$'
MSG2 DB 0AH,0DH,'The reversed string is : ','$'

.CODE
MAIN PROC
   MOV AX,@DATA
   MOV DS,AX

   MOV AH,9
   LEA DX,MSG1
   INT 21H

   XOR CX,CX

   MOV AH,1
   INT 21H

   WHILE_:
   CMP AL,0DH
   JE END_WHILE


   PUSH AX
```

```
    INC CX

    INT 21H    ;read a character
    JMP WHILE_

    END_WHILE:

    MOV AH,9
    LEA DX,MSG2
    INT 21H

    JCXZ EXIT  ; if CX register is 0

    MOV AH,2
TOP:

    POP DX

    INT 21H
    LOOP TOP

    EXIT:

    MOV AH,4CH
    INT 21H
    MAIN ENDP

END MAIN
```

Question No: 02

Write a program that lets the user type in an algebraic expression, ending with a carriage return, that contains round (parentheses), square, and curly brackets. As the expression is being typed in, the program evaluates each character. If at any point the expression is incorrectly bracketed (too many right brackets or a mismatch between left and right brackets), the program tells the user to start over. After the carriage return is typed, if the expression is correct, the program displays "expression is correct." If not, the program displays "too many left brackets". In both cases, the program asks the user if he or she wants to continue. If the user types 'Y', the program runs again. Your program does not need to store the input string, only check it for correctness.

Answer:

```
.MODEL SMALL
 .STACK 100H

 .DATA
  PROMPT        DB  0DH,0AH,'Enter an Algebraic Expression : ','$'
  PROMPT2       DB  0DH,0AH,'Enter an Algebraic Expression : ','$'
  CORRECT       DB  0DH,0AH,'Expression is Correct.$'
  LEFT_BRACKETS_ERROR   DB  0DH,0AH,'Too many Left Brackets.$'
  RIGHT_BRACKETS_ERROR  DB  0DH,0AH,'Too many Right Brackets.$'
  MISMATCH_ERROR       DB  0DH,0AH,'Bracket Mismatch. Begin Again.$'
CONTINUE       DB  0DH,0AH,'Type Y if you want to Continue : $'

 .CODE
  MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    START:
```

```
LEA DX, PROMPT2
MOV AH, 9
INT 21H

XOR CX, CX

MOV AH, 1

INPUT:
 INT 21H

 CMP AL, 0DH
 JE END_INPUT

 CMP AL, "["
 JE PUSH_BRACKET

 CMP AL, "{"
 JE PUSH_BRACKET

 CMP AL, "("
 JE PUSH_BRACKET

 CMP AL, ")"
 JE ROUND_BRACKET

 CMP AL, "}"
 JE CURLY_BRACKET

 CMP AL, "]"
 JE SQUARE_BRACKET          JMP INPUT
```

```
PUSH_BRACKET:

PUSH AX
INC CX
JMP INPUT

ROUND_BRACKET:

POP DX
DEC CX

CMP CX, 0
JL RIGHT_BRACKETS

CMP DL, "("
JNE MISMATCH
JMP INPUT

CURLY_BRACKET:

POP DX
DEC CX

CMP CX, 0
JL RIGHT_BRACKETS

CMP DL, "{"
JNE MISMATCH
JMP INPUT          SQUARE_BRACKET:

POP DX
DEC CX
```

```
  CMP CX, 0
  JL RIGHT_BRACKETS

  CMP DL, "["
  JNE MISMATCH
JMP INPUT

END_INPUT:

CMP CX, 0
JNE LEFT_BRACKETS

MOV AH, 9

LEA DX, CORRECT
INT 21H

CONTINUE_MESSAGE:


LEA DX, CONTINUE
INT 21H

MOV AH, 1
INT 21H

CMP AL, "Y"
JNE EXIT
JMP START

MISMATCH:
```

```
    LEA DX, MISMATCH_ERROR
    MOV AH, 9
    INT 21H

    JMP START:

    LEFT_BRACKETS:

    LEA DX, LEFT_BRACKETS_ERROR
    MOV AH, 9
    INT 21H

    JMP CONTINUE_MESSAGE:

    RIGHT_BRACKETS:

    LEA DX, RIGHT_BRACKETS_ERROR
    MOV AH, 9
    INT 21H

    JMP START

    EXIT:
    MOV AH, 4CH
    INT 21H
  MAIN ENDP
END MAIN
```