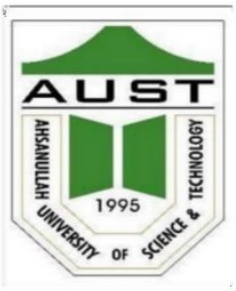

Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No : CSE2214

Course Title : Assembly Language Programming Sessional
Assignment No : 08

Date of Performance : 14-02-2021

Date of Submission : 28-02-2021

Submitted To : Ms. Tahsin Aziz & Mr. A.K.M. Amanat Ullah

Submitted By-

Group : B_2

Name : S. M Tasnimul Hasan

Id : 180204142

Section : B

Question No : 01

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

Answer :

```
.MODEL SMALL
.STACK 100H
.DATA
    MSG1  DB  'Enter the string : $'
    MSG2  DB  0DH, 0AH, 'The string with words in reverse
                        order : $'

    COUNT  DW  0

.CODE
    MAIN  PROC

        MOV  AX, @DATA
        MOV  DS, AX
```

```
MOV  AH, 9
LEA  DX, MSG1
INT  21H

XOR  CX, CX      ; clear CX
MOV  AH, 1

INPUT:
    INT 21H      ; read a character
    CMP AL, 0DH  ; compare AL with CR
    JE  END_INPUT ; jump to label if AL = CR

    PUSH AX      ; push AX onto the STACK
    INC  CX      ; set CX = CX+1

    JMP INPUT

    END_INPUT:
        MOV  BX, 50H   ; set BX = 50H
        XCHG BX, SP    ; swap BX and SP
        PUSH 0020H     ; push 0020H onto the STACK
        XCHG BX, SP    ; swap BX and SP
        INC  COUNT     ; set COUNT = COUNT+1
```

```
LOOP_1 :

    POP DX      ; pop a value from STACK into DX

    XCHG BX, SP ; swap BX and SP

    PUSH DX     ; push DX onto the STACK

    XCHG BX, SP ; swap BX and SP

    INC COUNT   ; set COUNT = COUNT+1

    LOOP LOOP_1 ; jump to label if CX != 0

    MOV AH, 9
    LEA DX, MSG2
    INT 21H

    MOV CX, COUNT
    MOV COUNT, 0    ; set COUNT = 0

    PUSH 0020H      ; push 0020H onto the STACK

    INC COUNT

    OUTPUT :

      XCHG BX, SP

      POP DX

      XCHG BX, SP
```

```
CMP  DL, 20H
JNE  SKIP_PRINTING   ; jump to label if DL != 20H

MOV  AH, 2

LOOP_2 :
    POP  DX      ; pop a value from STACK into DX
    INT  21H     ; print a character
    DEC  COUNT   ; set COUNT = COUNT - 1
    JNI  LOOP_2  ; jump to label if COUNT != 0

    MOV  DX, 0020H   ; set DX = 0020H

    SKIP_PRINTING :

        PUSH  DX
        INC   COUNT
        LOOP  OUTPUT   ; jump to label OUTPUT if CX != 0

        MOV  AH, 4CH   ; return 0
        INT  21H
        MAIN ENDP

END  MAIN
```

Question No : 02

Question : Write a program that lets the user type in an algebraic expression, ending with a carriage return, that contains round, 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.

```
.MODEL SMALL
.STACK 100H
.DATA
    MSG1 DB 0DH,0AH,'Enter an Algebraic Expression :',0DH,0AH,'$'
    MSG2 DB 0DH,0AH,'Expression is correct. $'
    MSG3 DB 0DH,0AH,'Too many Left Brackets. $'
    MSG4 DB 0DH,0AH,'Too many Right Brackets. $'
    MSG5 DB 0DH,0AH,'Bracket Mismatch. Begin Again. $'
    MSG6 DB 0DH,0AH,'Type Y if you want to continue : $'

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

    START :
      MOV AH,9
      LEA DX,MSG1
      INT 21H

      XOR CX,CX   ; clear CX
      MOV AH,1
```

```
INPUT :
    INT 21H         ; read a character
    CMP AL, 0DH     ; compare AL with CR
    JE END_INPUT    ; jump to label if AL = CR

    CMP AL, "["     ; compare AL with "["
    JE PUSH_BRACKET ; jump to label if AL = "["

    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       ; jump to label INPUT
```

```
PUSH_BRACKET :
    PUSH AX      ; push AX onto the STACK
    INC CX
    JMP INPUT

ROUND_BRACKET :
    POP DX       ; Pop a value from STACK into DX
    DEC CX
    CMP CX, 0    ; compare CX with 0
    JL RIGHT_BRACKETS ; Jump to label if CX<0

    CMP DL, "("
    JNE MISMATCH     ; Jump to label if DL!="("

    JMP INPUT

CURLY_BRACKET :
    POP DX
    DEC CX
    CMP CX, 0
    JL RIGHT_BRACKETS
```

```
    CMP DL, "{"
    JNE MISMATCH    ; jump to label if DL != "{"
    JMP INPUT

SQUARE_BRACKET :

    POP DX
    DEC CX

    CMP CX, 0
    JL RIGHT_BRACKETS   ; jump to label if CX < 0

    CMP DL, "["
    JNE MISMATCH
    JMP INPUT

END INPUT :

    CMP CX, 0
    JNE LEFT_BRACKETS   ; jump to label if CX != 0

    MOV AH, 9
    LEA DX, MSG2
    INT 21H
```

```
LEA DX, MSG6
INT 21H

MOV AH, 1
INT 21H

CMP AL, "Y"
JNE EXIT    ; jump to label EXIT if AL!="Y"
JMP START

MISMATCH :
    MOV AH, 9
    LEA DX, MSG5
    INT 21H

    JMP START

- LEFT BRACKETS :
    MOV AH, 9
    LEA DX, MSG3
    INT 21H

    JMP START
```

```
RIGHT_BRACKETS :

    MOV AH,9
    LEA DX, MSG4
    INT 21H

    JMP START

EXIT :

    MOV AH, 4CH    ; return 0
    INT 21H

    MAIN ENDP

END MAIN
```