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	**:18 .02.2021**
Date of Submission	**: 25.02.2021**

Submitted To : Ms. Moumita Choudhury & Ms. Tanjila Broti

Submitted By-

Group	**: A2**
Name	**: Minhajul Islam Jim**
Id	**: 17.01.04.001**
Section	**: A**

<u>Question No: 01</u>

<u>Question:</u> 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.
<u>Answer:</u>

```
.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 ;initialize DS
 MOV DS,AX

 MOV AH,9 ;String Output
 LEA DX,MSG1
 INT 21H
 XOR CX,CX ;clear CX
 MOV AH,1 ;single key input
 INPUT:
 INT 21H ;read a character
 CMP AL,0DH ;compare AL with CR
```

```
JE END_INPUT ;if AL=CR jump to label END_INPUT
PUSH AX ;push AX onto the STACK
INC CX ;set CX=CX+1
JMP INPUT ;jump to label 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 LOOP_1 if CX!=0

MOV AH,9 ;String Output
LEA DX,MSG2
INT 21H
MOV CX,COUNT ;set CX=COUNT
MOV COUNT,0 ;set COUNT=0
PUSH 0020H ;push 0020H onto the STACK
```

```
INC COUNT ;set COUNT=COUNT+1
OUTPUT:
XCHG BX,SP ;swap BX and SP
 POP DX ;pop a value from STACK into DX
XCHG BX,SP ;swap BX and SP
CMP DL,20H ;compare DL with 20H
JNE SKIP_PRINTING ;jump to label SKIP_PRINTING if DL!=20H
MOV AH,2 ;Single key Output
LOOP_2:
POP DX ;pop a value from STACK into DX
INT 21H ;print a character
DEC COUNT ;set COUNT=COUNT-1
JNZ LOOP_2 ;jump to label LOOP_2 if COUNT!=0
MOV DX,0020H ;set DX=0020H
SKIP_PRINTING:
PUSH DX ;push DX onto the STACK
INC COUNT ;set COUNT=COUNT+1
LOOP OUTPUT ;jump to label OUTPUT if CX!=0
MOV AH,4CH ;return 0
INT 21H
MAIN ENDP
END MAIN
```

<u>Question No: 02</u>

<u>Question:</u> 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.

<u>Answer:</u>

```
.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 ;initialize DS
 MOV DS,AX
 START:
```

```
MOV AH,9 ;String Output

LEA DX,MSG1

INT 21H

XOR CX,CX ;clear CX

MOV AH,1 ;Single key Input

INPUT:

INT 21H ;read a character

CMP AL,0DH ;compare AL with CR

JE END_INPUT ;jump to label END_INPUT if AL=CR

CMP AL,"[" ;compare AL with "["

JE PUSH_BRACKET ;jump to label PUSH_BRACKET if AL="["

CMP AL,"{" ;compare AL with "{"

JE PUSH_BRACKET ;jump to label PUSH_BRACKET if AL="{"

CMP AL,"(" ;compare AL with "("

JE PUSH_BRACKET ;jump to label PUSH_BRACKET if AL="("

CMP AL,")" ;compare AL with ")"

JE ROUND_BRACKET ;jump to label ROUND_BRACKET if AL=")"


CMP AL,"}" ;compare AL with "}"

JE CURLY_BRACKET ;jump to label CURLY_BRACKET if AL="}"

CMP AL,"]" ;compare AL with "]"

JE SQUARE_BRACKET ;jump to label SQUARE_BRACKET if AL="]"

JMP INPUT ;jump to label INPUT

PUSH_BRACKET:
```

```
PUSH AX ;push AX onto the STACK
INC CX ;set CX=CX+1
JMP INPUT ;jump to label INPUT
ROUND_BRACKET:
POP DX ;pop a value from STACK into DX
DEC CX ;set CX=CX-1
CMP CX,0 ;compare CX with 0
JL RIGHT_BRACKETS ;jump to label RIGHT_BRACKETS if CX<0
CMP DL,"(" ;compare DL with "("
JNE MISMATCH ;jump to label MISMATCH if DL!="("
JMP INPUT ;jump to label INPUT

CURLY_BRACKET:
POP DX ;pop a value from STACK into DX
DEC CX ;set CX=CX-1
CMP CX,0 ;compare CX with 0
JL RIGHT_BRACKETS ;jump to label RIGHT_BRACKETS if CX<0
CMP DL,"{" ;compare DL with "{"
JNE MISMATCH ;jump to label MISMATCH if DL!="{"
JMP INPUT ; jump to label INPUT
SQUARE_BRACKET:
POP DX ;pop a value from STACK into DX
DEC CX ;set CX=CX-1
CMP CX,0 ;compare CX with 0
```

```
JL RIGHT_BRACKETS ;jump to label RIGHT_BRACKETS if CX<0
CMP DL,"[" ;compare DL with "["
JNE MISMATCH ;jump to label MISMATCH if DL!="["
JMP INPUT ;jump to label INPUT
END_INPUT:
CMP CX,0 ;compare CX with 0
JNE LEFT_BRACKETS ;jump to label LEFT_BRACKETS if CX!=0
MOV AH,9 ;String Output
LEA DX,MSG2 ;load and print the string MSG2
INT 21H
LEA DX,MSG6 ;load and print the string MSG6
INT 21H
MOV AH,1 ;Single key Input
INT 21H
CMP AL,"Y" ;compare AL with "Y"
JNE EXIT ;jump to label EXIT if AL!="Y"
JMP START ;jump to label START
MISMATCH:
MOV AH,9
LEA DX,MSG5 ;load and display the string MSG5
INT 21H
JMP START ;jump to label START
LEFT_BRACKETS:
MOV AH,9
```

```
 LEA DX,MSG3 ;load and display the string MSG3

 INT 21H

 JMP START ;jump to label START

 RIGHT_BRACKETS:

 MOV AH,9

 LEA DX,MSG4 ;load and display the string MSG4

 INT 21H

 JMP START ;jump to label START

 EXIT:

 MOV AH,4CH ;return 0

 INT 21H

 MAIN ENDP

END MAIN
```