Assembly Language Programming (ALP) in 8086

Structure of 8086 program

```
PAGE …. , …. ;Length, Width (number of lines & Character on a page)
TITLE ……. ; Text
.MODEL …… ; Model Type
.STACK …… ; Stack Size
.DATA ; Start Data Segment
…….
……. ; Data declaration here
…….
.CODE ; Start Code Segment
     MAIN PROC ; Start Procedure with procedure name main
          MOV AX, @DATA ; initialization of Data Segment
          MOV DS, AX (Set DS to point to the Data Segment)
          ……..
          ……
          ……... ; Program Statement (Mnemonics)
          ……...
          ……
          MOV AX, 4C00H ; Ends of processing/Terminates current program
          INT 21H ; Call of interrupt service
     MAIN ENDP ; Ends of Procedure
END MAIN ; Ends of Program
```

Note:

1. '**.STARTUP**' directive works similar to as **MOV AX, @DATA**, **MOV DS, AX** i.e. initialization of data segment and '**. EXIT**' or '**END**' directive works similar to 'END PROC' and 'END MAIN' functions.
2. 8086 ALP is **not case-sensitive**, so you can write commands (Instructions/Mnemonics) and assembly directives either in uppercase or lowercase.

8086 ALPs

1. Write an assembly language program for 8086 to read a character from user and display it.
2. Write an assembly language program for 8086 to display a string on the screen.
3. Write an assembly language program for 8086 to read string and print it in the reverse order.
4. write an assembly language program for 8086 to change lowercase string into uppercase.
5. write an assembly language program for 8086 to change uppercase string into lowercase.
6. Write an assembly language program for 8086 to print number of vowels in a given string.
7. Write an assembly language program for 8086 to read two strings and check whether they are same or not.
8. Write an assembly language program for 8086 to concatenate two strings.
9. Write an assembly language program for 8086 input String and Display Each Word in Next Line

Solutions:

1)

```
include 'emu8086.inc'
.stack 100h
.model small

.data
    char db ?
.code
    main proc
        mov ax, @data
        mov ds, ax

        print 'Enter a character: '

        mov ah, 01h
        int 21h

        mov char, al

        mov dl, 10     ;for newline
        mov ah, 02h
        int 21h

        mov dl, 13     ;for carriage return
        mov ah, 02h
        int 21h

        print 'Your character is: '
```

```
        mov dl, char
        mov ah, 02h
        int 21h

       mov ah, 4ch
        int 21h

    main endp
end main
```

2)

```
.model small
.stack
.data
    name1 db 'Tribhuvan University$'
    name2 db 13, 10, 'Kathmandu Nepal$'

.code
     main proc
        mov ax, @data
        mov ds, ax

        lea dx, name1
        mov ah,09h
        int 21h

        lea dx, name2
        mov ah,09h
        int 21h

       mov ah, 4ch
        int 21h

    main endp
end main
```

3)

```
include 'emu8086.inc'
.stack 100h
.model small
.data
    string db 'TEXAS CSIT SECOND SEMESTER$'
.code

    main proc
        mov ax, @data
        mov ds, ax

        print 'Original String: '
```

```
        lea dx, string
        mov ah, 09h
        int 21h

        mov dl, 10
        mov ah, 02h
        int 21h

        mov dl, 13
        mov ah, 02h
        int 21h

        mov si, offset string
        mov cx,26

        print 'string after reversal: '
        l1:
            mov bx, [si]
            push bx
            inc si
            loop l1

        mov cx, 26
        l2:
           pop dx
           mov ah, 02h
           int 21h
           loop l2

       mov ah, 4ch
        int 21h
    main endp
end main
```

4)

```
.model small
.stack
.data
    msg1 db 'Original String is:$'
    msg2 db 10, 13, 'Uppercase String is:$'
    name1 db 'tribhuvan university$'
.code
    main proc
        mov ax, @data
        mov ds, ax
        ;.startup

        lea dx, msg1
        mov ah, 09h
```

```
        int 21h

        lea dx, name1
        mov ah,09h
        int 21h

        lea dx, msg2
        mov ah, 09h
        int 21h

        mov cx,20
        lea si, name1

        uppercase:
        cmp [si], 32
        je skip
        sub [si], 20h

  skip: mov dl, [si]
        mov ah, 02h
        int 21h
        inc si

       loop uppercase

        mov ah, 4ch
        int 21h

    main endp
end main
```

5)

```
.model small
.stack
.data
    msg1 db 'Original Text is:$'
    msg2 db 10, 13, 'Lowercase Text is:$'
    string db 'TRIBHUVAN UNIVERSITY$'
.code
    main proc
        mov ax, data
        mov ds, ax
        ;.startup

        lea dx, msg1
        mov ah, 09h
        int 21h

        lea dx, string
```

```
        mov ah,09h
        int 21h

        lea dx, msg2
        mov ah, 09h
        int 21h

        mov cx,20
        lea si, string

        lowercase:
        cmp [si], 32
        je skip
        add [si], 20h

  skip: mov dl, [si]
        mov ah, 02h
        int 21h
        inc si

        loop lowercase

       mov ah, 4ch
        int 21h
    main endp
end main
```

6)

```
.model small
.stack
.data
    msg db 'Number of vowels in the given string is: $'
    string db 'mIcroprocEssOr$'
.code
    main proc
        mov ax, @data
        mov ds, ax
        ;.startup
        mov bl, 0 ;to count no. of vowels

        mov cx, 14 ;length of string
        mov si,offset string
        AGAIN:
        mov al,[si]
        cmp al, 61h ;ASCII of ‘a’ in HEX
        je COUNT    ;jump if equal

        cmp al, 41h ;ASCII of ‘A’ in HEX
        je COUNT    ;jump if equal

        cmp al, 65h ;ASCII of ‘e’ in HEX
        je COUNT    ;jump if equal

        cmp al, 45h ;ASCII of ‘E’ in HEX
        je COUNT    ;jump if equal

        cmp al, 69h ;ASCII of ‘i’ in HEX
        je COUNT    ;jump if equal

        cmp al, 49h ;ASCII of ‘I’ in HEX
        je COUNT    ;jump if equal

        cmp al, 6fh ;ASCII of ‘o’ in HEX
        je COUNT    ;jump if equal

        cmp al, 4fh ;ASCII of ‘O’ in HEX
        je COUNT    ;jump if equal

        cmp al, 75h ;ASCII of ‘u’ in HEX
        je COUNT    ;jump if not equal

        cmp al, 55h ;ASCII of ‘U’ in HEX
        je COUNT
        jmp next_char ;If not a vowel, move to the next character
```

```
        COUNT: inc bl

        next_char:
        inc si
        loop AGAIN

        lea dx, msg ;Display output msg
        mov ah, 09h
        int 21h

        add bl,30h  ;Convert the vowel count to ASCII and display it
        mov dl,bl
        mov ah,02h
        int 21h

        mov ah, 4ch
        int 21h
    main endp
end main
```

7)

```
.model small
.stack 100h
.data
    cr equ 13
    nl equ 10
    inmsg1 db 'Enter the first string: $'
    inmsg2 db cr,nl,nl, 'Enter the second string: $'
    strng1 db 0ah, 100 dup(?)   ;reserves space for strng1, starting
                                ;with a byte 0ah, followed by 100
                                ;uninitialized bytes
    strng2 db 0ah, 100 dup(?)   ;reserves space for strng2, starting
                                ;with a byte 0ah, followed by 100
                                ;uninitialized bytes
    sucmsg db cr,nl,nl,'Both are same $'
    falmsg db cr,nl,nl,'Different strings $'
.code
    main proc
        mov ax, @data
        mov ds, ax
        mov es, ax

        lea dx, inmsg1
        mov ah,09
        int 21h

        mov dx, offset strng1
        mov ah, 0ah
        int 21h
```

```
        lea dx, inmsg2
        mov ah, 09h
        int 21h

        mov dx, offset strng2
        mov ah,0ah
        int 21h

        mov si, offset strng1
        mov di, offset strng2
        cld
        mov cx, 6h   ;maximum characters that are compared from the
                     ;both strings
        repe cmpsb
        jz success
        lea dx, falmsg
        jmp display

        success: lea dx, sucmsg
        display: mov ah,09h
        int 21h

        mov ah,4ch
        int 21h
    main endp
end main
```

8)

```
.model small
.stack
.data
    string1 db 'Microprocessor is an$'
    string2 db 'Assembly Language.$'
    string3 db ?
    spc equ 32

.code
    main proc
        mov ax, @data
        mov ds, ax

        mov di,offset string3
        mov si,offset string1

        mov cx,20

        l1:
        mov bx,[si]
```

```
        mov [di], bx
        inc si
        inc di
        loop l1

        mov [di], spc    ;To print space between two strings
        inc si
        inc di

        mov si, offset string2
        mov cx,18
        l2:
        mov bx, [si]
        mov [di], bx
        inc si
        inc di
        loop l2

        mov dx, offset string3   ;or lea dx, string3
        mov ah, 09h
       int 21h

       mov ah, 4ch
       int 21h
      main endp
end
```

9)

```
include 'emu8086.inc'
.model small
.stack 100
.data
    msg db 60 dup(?)

.code
main proc
    mov ax,@data
    mov ds,ax

    lea si, msg
    print 'Enter your string: '

    input:
        mov ah, 01h
        int 21h
        cmp al, 13
        je display
        mov [si], al
        inc si
```

```
        jmp input

    display:
        mov [si], '$'
        lea di, msg
        mov dl, 10
        mov ah, 2
        int 21h
        mov dl, 13
        mov ah, 2
        int 21h

    again:
        cmp [di], '$'
        je last
        cmp [di], 32
        je next
        mov dl, [di]
        mov ah, 02h
        int 21h
        inc di
        jmp again

    next:
        mov dl,10
        mov ah,2
        int 21h

        mov dl,13
        mov ah,2
        int 21h

        inc di
        jmp again

    last:
       mov ah,4ch
       int 21h

   main endp
end
```

10. Write an assembly language program which take string input from user and display that string on the console.

ALP:

```
.model small
.stack 100h
.data
    msg1 db "Enter a string: $"
    msg2 db 10, 13, "Your string is: $"
    str db 100 dup('$')

.code
    main proc
        mov ax, @data
        mov ds, ax

        lea dx, msg1
        mov ah, 09h
        int 21h

        mov ah, 0ah
        lea dx, str
        mov str, 40 ;Sets the first byte of str to 40h (64 in decimal),
                    ;which defines the maximum number of characters
                    ;the user can input.
        int 21h

        lea dx, msg2
        mov ah, 09h
        int 21h

        lea dx, str+2   ; Loads the address of the user input (skipping
                        ;the first two bytes,
                        ; which contain the maximum length and actual
length) into DX.
        mov ah, 09h
        int 21h

        mov ah, 4ch ;mov ax, 4c00h
        int 21h

       main endp
 end main
```

Assignments

1. Write an assembly language program for 8086 to calculate factorial of a given number.
2. Write an assembly language program for 8086 to find the sum of Natural numbers from 1 to 10. [1+2+3+ …. +10]
3. Write an assembly language program for 8086 to print the multiplication table of a given number.