Microprocessor

1. **Write an assembly language program to transfer block of 8-bit data from one memory location to another.**
   Solution:

```
.model small
.stack 100h
.data
list1 db 10h,20h,30h,40h,50h
list2 db 5 dup(?)

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

    mov si,offset list1 ;move the offset address of list1 to si
    mov di,offset list2 ;move the offset address of list2 to di
    mov cx,0005h    ;cx is always used as counter

again:
    mov al,[si] ;mov the first element of list1 to al i.e content of ds:si
    mov [di],al ;transfer the first element of list1 to list2
    inc si
    inc di

loop again  ;auto decrements cx and the loop continues till cx=0000h

mov ax,4c00h
int 21h
main endp
end main
```

2. **Write an assembly language program to add all the elements of list1 and store in variable.**
   Solution:

```
.model small
.stack 100h
.data
list1 db 10h,20h,30h,40h,50h
list2 db 5 dup(?)

.code
main proc far
    mov ax,@data
    mov ds,ax
    mov si,offset list1 ;move the offset address of list1 to si
    mov di,offset list2 ;move the offset address of list2 to di
    mov cx,0004h    ;cx is always used as counter

    mov al,[si]
again:

    inc si
    add al,[si]
    mov list2,al
loop again  ;auto decrements cx and the loop continues till cx=0000h
mov ax,4c00h
int 21h
main endp
end main
```

3. **There are two tables having ten 16-bit data in each. Write an assembly language program to generate the third table which contains the sum of corresponding element of 1st and 2nd table.**

   Solution:

```
title addition of two table
.model small
.stack 100h
.data
    array1 dw 1111h,2222h,3333h,4444h,5555h,11h,22h,33h,44h,55h
    array2 dw 1111h,2222h,3333h,4444h,5555h,55h,44h,33h,22h,11h
    arraysum dw 10 dup(?)
.code
main proc
    mov ax,@data
    mov ds,ax

    mov cx,000ah
    mov bx,0000h
    start:
    mov ax,array1[bx]
    add ax,array2[bx]
    mov arraysum[bx],ax
    inc bx
    inc bx
    loop start

    mov ax,4c00h
    int 21h
    main endp
    end main
```

4. **Two tables contain ten 16-bit data each. Write an assembly language program to generate the 3rd table which contains 1FFFh if the corresponding data in the 1st table is less than that of 2nd table, else store 0000h.**
   Solution:

```
.model small
.stack 100h
.data
    array1 dw 0111h,0222h,0333h,0444h,0555h,732h,22h,33h,0aaah,0bbbh
    array2 dw 0222h,0111h,0132h,4444h,5555h,55h,44h,33h,22h,11h
    arraysum dw 10 dup(?)

.code
main proc far
    mov ax,@data
    mov ds,ax
    mov cx,0ah
    mov bx,00h
    start:
    mov dx,0000h
    mov ax,array1[bx]
    cmp ax,array2[bx]
    jae condition   ;jump if above or equal
    mov dx,1fffh
    condition:
    mov arraysum[bx],dx
    inc bx
    inc bx
    loop start

mov ax,4c00h
int 21h
main endp
end main
```

5. **Write an assembly language program to find the largest number in the list of array of 5 elements.**
   Solution:

```
title find largest number
.model small
.stack 100h
.data
    list db 10h,20h,30h,40h,09h,60h
    large db 00h
.code
```

```
main proc
    mov ax,@data
    mov ds,ax

    mov si,offset list
    mov bl,large
    mov cx,0006h
    mov bl,[si]
    again:

    cmp bl,[si]
    jnc nochange
    mov bl,[si]

    nochange:
        inc si
        loop again

        mov large,bl
    mov ax,4c00h
    int 21h
    main endp
end main
```

6. **Write an assembly language program to find the smallest number in the list of array of 5 elements.**
   Solution:

```
title find smallest number
.model small
.stack 100h
.data
    list db 10h,20h,30h,40h,09h,60h
    small db 00h
.code
main proc
    mov ax,@data
    mov ds,ax

    mov si,offset list
    mov bl,small
    mov cx,0006h
    mov bl,[si]
    again:

    cmp bl,[si]
    jc nochange
```

```
        mov bl,[si]

        nochange:
           inc si
           loop again

           mov small,bl
        mov ax,4c00h
        int 21h
        main endp
     end main
```

7. **Write a program to generate the multiplication table of a given number.**
   Solution:

```
     .model small
     .stack 100h
     .data
     list db 10 dup(?)
     num db 03h
     .code
     main proc
        mov ax,@data
        mov ds,ax

        mov si,offset list
        mov cx,0ah

        mov al,num

        mov bl,al

        mov dl,01h

        back:
        mul dl
        mov [si],al
        inc si
        inc dl
        mov al,bl
        loop back

        mov ax,4c00h
        int 21h

        main endp
     end main
```

8. **Write an assembly language program to arrange the given set of data in descending order.**
   Solution:

```
title sorting numbers
.model small
.stack 100h
.data
   list db 10h,42h,11h,05h,01h,79h,34h,67h,02h,12h
.code
   main proc far
      mov ax,@data
      mov ds,ax
   sort:
      mov si,offset list
      mov bl,00h
      mov cx,000ah
   back:
      mov al,[si] ;get kth element
      inc si
      cmp al,[si] ;compare with (k+1)th element
      jnc ahead   ;not interchange if kth<=(k-1)th
      mov dl,[si]
      mov [si],al
      dec si
      mov [si],dl
      inc si
      mov bl,01   ;interchange flag =1
   ahead:
      loop back   ;is interchange flag=1
      dec bl
      jz sort     ;yes, do another pass

      mov ax,4c00h
      int 21h
   main endp
   end main
```

9. **Write a program to generate the Fibonacci series up-to 10 numbers.**
   Solution:

```
.model small
.stack 100h
.data
   list db 10 dup (?)
.code
   main proc
```

```
    mov ax,@data
    mov ds,ax

    mov si,offset list
    mov bh,00h
    mov bl,01h
    mov cx,000ah

again:
    mov [si],bh
    add bh,bl
    mov dh,bh
    mov bh,bl
    mov bl,dh
    inc si

    loop again


    mov ax,4c00h
    int 21h

    main endp
    end main
```

10. **Write a program to generate the multiplication table of a number given by the user.**
    Solution:

```
.model small
.stack 100h
.data
 .code
main proc
mov ax,@data
mov ds,ax

mov ah,08h     ;number entered by the user
int 21h

and al,0fh  ;taking only LSB
mov dh,al
mov bl, 01h
mov cx,10   ;counter
again:
mov al,dh
mul bl
aam
mov bh,al
```

```
cmp ah,00h ; not showing 0 in output
je label

add ah,30h    ; adding 30 converts the contents of ah to decimal.
mov dl,ah
mov ah,02h
int 21h

label:
mov al,bh
add al,30h
mov dl,al
mov ah,02h
int 21h


mov dl,20h
mov ah,02h
int 21h

inc bl
loop again

mov ax,4c00h
int 21h
main endp
end main
```

11. Write an assembly language program to arrange the given set of data in descending order.

Solution:

```
title sorting numbers
.model small
.stack 100h
.data
   list db 10h,42h,11h,05h,01h,79h,34h,67h,02h,12h
.code
   main proc far
      mov ax,@data
      mov ds,ax
   sort:
      mov si,offset list
      mov bl,00h
      mov cx,000ah
   back:
      mov al,[si] ;get kth element
      inc si
```

```
    cmp al,[si] ;compare with (k+1)th element
    jnc ahead   ;not interchange if kth<=(k-1)th
    mov dl,[si]
    mov [si],al
    dec si
    mov [si],dl
    inc si
    mov bl,01   ;interchange flag =1
ahead:
    loop back   ;is interchange flag=1
    dec bl
    jz sort     ;yes, do another pass

    mov ax,4c00h
    int 21h
main endp
end main
```

12. Write a program to generate multiplication table of five numbers stored in memory as array, store the result and display in following format

```
2    4    6    8    10   12   14   16   18   20
3    6    9    12   15   18   21   24   27   30
4    8    12   16   20   24   28   32   36   40
5    10   15   20   25   30   35   40   45   50
6    12   18   24   30   36   42   48   54   60
```

Solution:

```
.model small
.stack 100h
.data
    array db 02h,03h,04h,05h,06h
.code
main proc
mov ax,@data
mov ds,ax
mov bx,0000h
mov cx,0005h
push cx
push bx

No_of_table:
push cx
mov al,array[bx]
and al,0fh  ;taking only LSB
mov dh,al
mov bl, 01h
mov cx,10   ;counter
again:
mov al,dh
mul bl
aam
mov bh,al
cmp ah,00h ; not showing 0 in output
je label

add ah,30h    ; adding 30 converts the contents of ah to decimal.
mov dl,ah
mov ah,02h
int 21h

label:
mov al,bh
add al,30h
mov dl,al
mov ah,02h
int 21h

mov dl,20h
mov ah,02h
int 21h

inc bl
loop again

mov dl,0dh                ;carriage return
mov ah,02h
int 21h

mov dl,0ah                ;next line
mov ah,02h
int 21h

pop cx
pop bx
inc bx
push bx
loop No_of_table

mov ax,4c00h
int 21h
main endp
end main
```

13. **Write a program that finds the sum of the following series up-to 10th term and store the result in a variable. Series -> 2*3+4*5+6*7+…..+ up-to 10th term.**
    Solution:

```
.model small
.stack 100h
.data
    sum dw ?
.code
    main proc
        mov ax,@data
        mov ds,ax

        mov cx,0ah
        mov ah,00h
        mov dx,00h
        mov bl,03h
        mov bh,02h

        again:
            mov al,bh
            mul bl
            add dx,ax

            add bl,02
            add bh,02
            loop again

            mov sum,dx

            mov ax,4c00h
            int 21h

            main endp
    end main
```

14. **Write an Assembly language program to print the given line word wise into next line.**
    Solution:

```
.model small
.stack 100h
.data

string db 'I would love to program in 8086 assembly language$'
;count dw $-string
.code
main proc
```

```
            mov ax,@data
            mov ds,ax




            lea si,string
again:
            mov dl,[si]
            cmp dl,'$'
            jz finish
            cmp dl,32
            jz nextline
            jmp print
nextline:
            mov dl,0dh
            mov ah,02h
            int 21h
            mov dl,0ah
            mov ah,02h
            int 21h
print:
            inc si
mov ah,02h
int 21h

jmp again

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

15. **Write a program to convert from lowercase to uppercase entered by the user.**
    Solution:

```
.model small
.stack 100h
.data
    string db ''
.code
main proc
            mov ax,@data
            mov ds,ax
            mov di,offset string
a1:
```

```
            mov ah,08h ; reading character without echo
            int 21h
            cmp al,0dh
            je a3
            cmp al,'a'
            jb a2
            cmp al,'z'
            ja a2
            sub al,32
    a2:
            mov ah,02h
            mov dl,al
            int 21h
            mov [di],al
            inc di
            jmp a1
    a3:
            inc di
            mov dl,'$'
            mov [di],dl
            mov dx,offset string
            mov ah,09h
            int 21h
            mov ah,4ch
            int 21h
            main endp
            end main
```

16. Write a program to count the number of vowel in given sentence.
Solution:

```
.model small
.stack 100h
.data
   list db 'the quick brown fox jumped over lazy sleeping dog'
   len dw $-list
   vow db ?
.code
   main proc far
      mov ax,@data
               mov ds,ax
               mov si,offset list
               mov cx,len
               mov ch,00h
               mov bl,00
   back:
         cmp [si],'a'
```

```
            jb vowel
            cmp [si],'z'
            ja vowel
vowel:
        cmp [si],'a'
        jnz a3
        inc bl
        jmp a2
    a3:
        cmp [si],'e'
        jnz a4
        inc bl
        jmp a2
    a4:
        cmp [si],'i'
        jnz a5
        inc bl
        jmp a2
    a5:
        cmp [si],'o'
        jnz a6
        inc bl
        jmp a2
    a6:
        cmp [si],'u'
        jnz a2
        inc bl
    a2:
        inc si
        loop back
        mov vow,bl

        mov ax,4c00h
        int 21h
    main endp
    end main
```

17. **Write a program in 8086 to read a string and count the number of vowels, consonants, numerals and other characters and display the count.**
    Solution:

```
title counting different elements in a sentence
.model small
.stack 100h
.data
    vowels      db 00h
    consonents  db 00h
```

```
        numbers     db 00h
        others      db 00h

        para label byte
        ml db 45h
        len db ?
        msg db 45 dup(?)


    .code
    main proc far
        mov ax,@data                ;initializing data segment
        mov ds,ax

        mov ah,0ah                  ;taking input from user
        mov dx,offset para
        int 21h

        mov ch,00h
        mov cl,len                  ;count of the input string
        mov si,offset msg
        again:
            cmp [si],'A'            ;compare input character with 'A'
            jb number               ;if it is below A, jump to number

            cmp [si],'Z'            ;else compare it with 'Z'
            ja comp                 ;if it is above A, jump to comp

            add [si],20h            ;if it is between A and Z, convert it to
                                       small letters
            comp:
                cmp [si],'a'        ;compare it with 'a'
                jb number           ;if it is below a, jump to number
                cmp [si],'z'        ;else compare it with 'z'
                ja number           ;if it is above z, jump to number
                cmp [si],'a'        ;if it is between 'a' and 'z', check if
                                       it is a vowel and jump to inc_vowels
                je inc_vowels:
                cmp [si],'e'
                je inc_vowels:
                cmp [si],'i'
                je inc_vowels:
                cmp [si],'o'
                je inc_vowels:
                cmp [si],'u'
                je inc_vowels:

                inc consonents      ;if not a vowel, increment count of
                                       consonents
                jmp update          ;jump to update to take next character
            inc_vowels:
                inc vowels              ;increment vowel counter
                jmp update              ;jump to update to take next character

            number:
                cmp [si],'0'            ;check if it is a number, if not jump to
                                       inc_others
                jb inc_others
                cmp [si],'9'
                ja inc_others

                inc numbers             ;if it is a number, increment the number
                                                    counter
                jmp update              ;jump to update to take next character

            inc_others:
                inc others              ;increment other counter

            update:
                inc si
                loop again

                mov ax,4c00h
                int 21h

            main endp
    end main
```

18. **Write a program to scroll the text from right to left.**
    Solution:

```
.model small
.stack 100h
.data
    msg db 'I am scrolling... and its fun $'
    len dw $-msg
.code
    main proc

        mov ax,@data
        mov ds,ax
        mov ah,00     ;defining vedio mode
        mov al,03     ;80*25
        int 10h

        mov cx,80-len ;value need to scroll(move) from right to left
        mov bl,cl
        again:

        mov ah,02      ;setting cursor position
        mov dh,12      ;row 12th
        mov dl,bl      ;variable column (decreasing fashion)
        int 10h

        mov ah,09      ;displaying the messege
        lea dx,msg
        int 21h


        dec bl

        mov bh,00h
        mov ah,06h     ;clearing the window
        mov al,00
        int 10h

        loop again


        mov ax,4c00h
        int 21h

        main endp
end main
```

19. **Write an assembly language program to take name and address from the user and display at the center of the screen.**
    Solution:

```
.model small
.stack 100h

new_line macro        ;macro definition
mov ah,02h
mov dl,0ah
int 21h

mov ah,02h
mov dl,0dh
int 21h
endm

.data
paralist1 label byte       ;Giving 1st byte the Label 'paralist1'
```

```
max1 db 20
act1 db ?
name1 db 20 dup(0),'$'

paralist2 label byte
max2 db 20
act2 db ?
address db 20 dup(0),'$'
.code
main proc

mov ax,@data
mov ds,ax

mov ah,0ah
lea dx, paralist1
int 21h
new_line        ;macro

mov ah,0ah
lea dx, paralist2
int 21h

mov ah,02
mov dh,12
mov dl,40
int 10h

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


mov ah,02
mov dh,13
mov dl,40
int 10h

mov ah,09
lea dx,address
int 21h

main endp
end main
```

20. **Write a program to display your name at center of the screen with green background and red foreground.**
   Solution:

```
.model small
.stack 100h
.data
    paralist1 label byte       ;Giving 1st byte the Label 'paralist1'
    max1 db 20
    act1 db ?
    name1 db 20 dup(0),'$'
.code
    main proc
        mov ax,@data
        mov ds,ax

        mov ah,0ah
        lea dx, paralist1
        int 21h

        lea si,name1
        mov ah,02
        mov dh,12
        mov dl,40
        int 10h
        again:
        mov ah,02
        int 10h

        mov ah,09
        mov al,[si]
        cmp al,0dh    ;comparing the character with 'enter' key.
        je finish
        mov bl,2ch    ;green background and red foreground
        inc si        ;getting next character
        inc dx        ;next colum of screen
        mov cx,1h     ;number of times the character is to display
        int 10h
        jmp again

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