ASM

# Asm code

## Triangle code

```
.model small
.stack 100h

.data
    newline db 0ah, 0dh, '$'
    asterisk db '*'

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

    mov cx, 3     ; Number of rows

    mov si, 1     ; Counter for the number of rows

outer_loop:
    mov di, 1     ; Counter for printing asterisks in each ro
w

inner_loop:
    mov ah, 02h   ; Function to print character
    mov dl, [asterisk]
    int 21h

    inc di
    cmp di, si    ; Compare current column with the row count
er
    jle inner_loop

    mov ah, 09h   ; Function to print newline
    mov dx, offset newline
    int 21h

    inc si
    cmp si, cx    ; Compare current row with the total number
of rows
    jle outer_loop

    mov ah, 4ch   ; Function to exit program
    int 21h

main endp
end main
```

# new line print korar jonno

```
mov dx,13
  mov ah,2
  int 21h
  mov dx,10
  mov ah,2
  int 21h
```

## Character print

```


;program a single character
.model
.stack 100h
```

```
.data
.code
main proc
    mov dl ,'a'

    mov ah,2
    int 21h



     mov ah,4ch
     int 21h



main endp
end main
```

## program to print a name with character

```
;program to print a name with character

.model
.stack 100h
.data
.code

main proc
 mov dl ,'e'
 mov ah,2
 int 21h



 mov dl,'m'

 mov ah,2
 int 21h

 mov dl,'o'
 mov ah,2
 int 21h

 mov dl,'n'
 mov ah,2
 int 21h

 mov ah,4ch
 int 21h


main endp
end main
```

## program to input a character from user and output

```
;program to input char from user
.model small

.stack 100h
.data
.code

main proc

  mov ah,1;

  int 21h
  mov dl,al

  mov ah,2
  int 21h





main endp
end main
```

## Substract 2 numbers

```
.model small
.stack 100h
.data
a db 10,13,'enter 1st no :$'
b db 10,13,'enter 2nd no : $'
  c db 10,13,'your result is : $'



.code



main proc
   mov ax,@data
   mov ds,ax

   mov ah,9
```

```
lea dx,a
int 21h


mov ah,1 ;input
int 21h

  mov bl,al

  mov ah,9
  lea dx,b
  int 21h

  mov  ah,1  ;input

  int 21h

  mov bh,al

  mov ah,9
lea dx,c
int 21h


  sub bl,bh

  add bl,48

  mov dl,bl




  mov ah,2
  int 21h
```

```
     main endp
end
```

## sub 3 numbers

```
.model small
.stack 100h
.data
 m db ?



.code


main proc

 mov ah,1
 int 21h

 mov bl,al

 mov ah,1
 int 21h
 mov bh,al
 mov ah,1
 int 21h
 mov cl,al


 sub bl,bh
 add bl,48


sub bl,cl

add bl,48

mov dl,bl

mov ah,2
int 21h










     main endp
end
```

## input output

```
.model
.stack 100h
.code

main proc
    mov ah,1
    int 21h
    mov bl,al
    mov ah,1
    int 21h
    mov bl,al

    mov ah,1
    int 21h
    mov bh,al

    ;display

    mov ah,2

    mov dl,bl
    int 21h
    mov ah,2
    mov dl,bh
    int 21h



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

## Add two numbers

```
;Add 2 numbers
.model small
.stack 100h
.data
.code
 main proc

   mov ah,1
   int 21h
   mov bl,al


      mov al,1
      int 21h

      add bl,al


        sub bl,48
      mov dl,bl




     mov ah,2
     int 21h
```

```
    main endp
  end main
```

## Add  3 numbers

```
.model small
.stack 100h
.data
 m db ?



.code


main proc

 mov ah,1
 int 21h

 mov bl,al

 mov ah,1
 int 21h
 mov bh,al


 mov ah,1
 int 21h
 mov cl,al
 add bl,bh
 sub bl,48


add bl,cl

sub bl,48

mov dl,bl

mov ah,2
int 21h











     main endp
end
```

## Add 2 numbers double digit

```
.model small
.stack 100h
.data
 msg1  db 10,13,  'enter first number: $'

    msg2  db 10,13,  'enter second number: $'
```

```
      msg3 db 10,13,'result: $'

      a db ?
      b db ?
.code


main proc

    mov ax,@data
    mov ds,ax


    mov ah,9
    lea dx,msg1
    int 21h

    mov ah,1
    int 21h
    mov a,al




      mov ah,9
    lea dx,msg2
    int 21h


    mov ah,1
    int 21h
    mov b,al


    add al,a
```

```
mov ah,0 ;ei line

aaa ; ei line likhbo double digit er jonno

  add al,48
  add ah,48

  mov bx,ax

  mov ah,9
  lea dx,msg3
  int 21h

  mov ah,2
  mov dl,bh
  int 21h

  mov ah,2
  mov dl,bl
  int 21h


  mov ah,4ch

 int 21h
```

```
     main endp
end
```

## program to convert capital lettter to small letter

```
;program to convert capital lettter to small letter



.model small
.stack 100h
.data
.code
 main proc
      mov ah,1
      int 21h

      mov dl,al

      add dl,32

      mov ah,2

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

## Multipication of single base

```
;multipication for single ans



.model small
.stack 100h
.data
.code
 main proc
    mov al,3
    mov bl,4

    mul bl

    mov dx,ax
    add dx ,48

   mov ah,2
   int 21h



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

## Mul for boro digit

```
;multipication for boro  ans  3*5



.model small
.stack 100h
.data
a db 10,13,'hello$ '
b db 10,13, 'mul is$'

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

    mov ah,9
    lea dx,a
    int 21h



   mov al,3
   mov bl,5

   mul bl
   AAM
   mov ch,ah
   mov cl,al

   mov dl,ch
   add dl,48
```

```
        mov ah,9
     lea dx,b
     int 21h


    mov ah,2
    int 21h
    mov dl,cl
    add dl,48
      mov ah,2
      int 21h


     main endp
  end main
```

## multiplication from input from user

```
.model small
.stack 100h
.data
a db ?
b db ?
c db 'input first digit$'
d db 10,13,  'input secound digit$'

e db 10,13,'result$'

.code

main proc
```

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

mov ah,9
lea dx,c
int 21h

mov ah,1
int 21h

 sub al,48
 mov a,al


 mov ah,9
 lea dx,d
 int 21h

 mov ah,1
 int 21h
 sub al,48
 mov b,al



 mov ah,9
 lea dx,e
 int 21h


 mov al,a

mul b
AAM
```

```
    mov ch,ah
    mov cl ,al

    mov dl,ch
    add dl,48

    mov ah,2
    int 21h
    mov dl,cl
    add dl,48
    mov ah,2
    int 21h


     mov ah,4ch
     int 21h

     main endp
end
```

## Division

```
;division



.model small
.stack 100h
.data
a db 10,13, 'result is :$'
b db 10,13, 'remainder is$ '
.code
 main proc
mov ax,@data
mov ds,ax
```

```
  mov ax,26
  mov bl,5

  div bl

  mov cl,al
  mov ch,ah

  mov ah,9
 int 21h
 lea dx,a


  mov dl,cl
  add dl,48
  mov ah,2
  int 21h

  mov ah,9
 int 21h
 lea dx,b


  mov dl,ch
  add dl,48

  mov ah,2
  int 21h







     main endp
  end main
```

/more example div

```
.model small
.stack 100h
.data
a db 10,13, 'result is :$'
b db 10,13, 'remainder is$ '
.code
 main proc
    mov ax,@data
mov ds,ax


 mov ax,26
 mov bl,5

 div bl

 mov cl,al
 mov ch,ah


 mov ah,9
int 21h
lea dx,a

 mov dl,cl
 add dl,48

 mov ah,2
 int 21h


 mov ah,9
int 21h
lea dx,b
```

```
mov dl,ch
add dl,48
mov ah,2
int 21h






   main endp
end main
```

## print string

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

 a db 'bangladesh is my country$'

 .code

 main proc
    mov ax,@data
    mov ds,ax

    mov ah,9
    lea dx,a
    int 21h

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

## input a number and print it on new line

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

 a db 'enter a number :$'
 b db 'the number is : $'



 .code

 main proc
    mov ax,@data
    mov ds,ax

    mov ah,9
    lea dx,a
    int 21h

    mov ah,1
    int 21h
    mov bl,al



      mov ah,2   ;
      mov dl,10
      int 21h
       mov dl,13     ;new line print korbe
       int 21h  ;




    mov ah,9
    lea dx,b
    int 21h




       mov dl,bl



      mov ah,2
      int 21h

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

## print string in reverse order

```
.model small
 .stack 100h

 .data
  string db "emon"
```

```
.code

main proc
        mov ax,@data
        mov ds,ax


   mov cx,4
   mov si,offset string

   l1:
      mov bx,[si]

      push bx

      inc si
      loop l1



    mov cx,4

    l2:

    pop dx
      mov ah,2
      int 21h

      loop l2


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

## How to convert celsius to fahrenheit using assembly language

```
;


org 100h
.data
F DW ? ; 16 bit er data rakhbo
.code
mov ax,@data
mov ds,ax
 main proc





    mov ax,260

    mov bx,9

    mul bx

    mov bx,5
```

```
   div bx

   mov bx,32
   add ax,bx

   mov bx,1
   sub ax,bx

   mov F,ax













   main endp

end main

ret
```

## Fahrenheit to Celsius

```
org 100h

;add your code here
```

```
.DATA
C DW ?
.CODE
MOV AX, @DATA
MOV DS,AX
MAIN PROC
    MOV AX,1000
    MOV BX,32
    SUB AX,BX
    MOV BX,5
    MUL BX
    MOV BX,9
    DIV BX
    ;ADD AX,BX
    MOV BX,1
    SUB AX,BX







    MOV C,AX
    MAIN ENDP
END MAIN

ret
```

## Input double digit

```
.model small
.data
```

```
.code

mov ax, @data
mov ds, ax

mov dl, 10
mov bl, 0

scanNum:

      mov ah, 01h
      int 21h

      cmp al, 13
      je  exit

      mov ah, 0
      sub al, 48

      mov cl, al
      mov al, bl

      mul dl
      mov bl, al

      jmp scanNum

exit:

      mov ah, 04ch
      int 21h

 end
```

## Variable input and output

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

a db 5

b db ?


.code

main proc
      mov ax,@data
      mov ds,ax

      mov ah,1 ; input nilam ebong b te rakhlam
      int 21h
      mov b,al

      mov ah,2
      mov dl,a
      add dl,48
      int 21h



        mov ah,2
        mov dl,b
        int 21h
```

```
       mov ah,4ch
       int 21h


     main endp
end main
```

## Loop

### print 1 to 9 using loop

```
; print 1 to 9
.model small
.stack 100h
.data

.code

main proc



     mov cx,10
 mov bl,1

 start:
 mov dl,bl
 add dl,48
```

```
  mov ah,02h
  int 21h

  mov dl,10
  mov ah,02
  int 21h
  mov dl,13


    inc bl
  mov ah,02
  int 21h

  loop start:





     main endp
 end main
```

## Print two characters in reverse alphabetic order using assembly language.

```
 .STACK 100H

  .DATA
  msg_1 DB ’Enter the first capital letter : $’;message 1
  msg_2 DB ’Enter the second capital letter : $’;message 2
  msg_3 DB ’The given capital letters in alphabetical order are
```

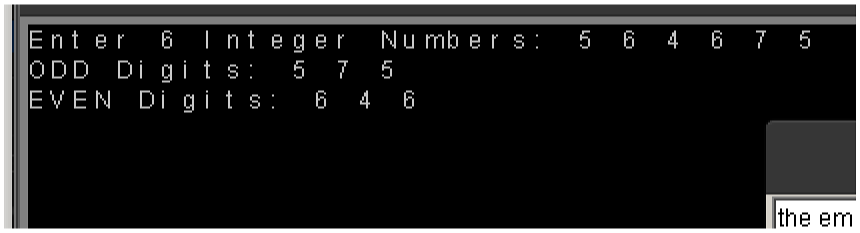
```
NEXT DB 0DH,0AH,"$"

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


MOV AH, 9 ; set string output function

LEA DX, NEXT ; Next line
INT 21H

LEA DX, msg_1 ; display message 1
INT 21H

MOV AH, 1 ; set input function
INT 21H ; read first character

MOV BL, AL ; save first character into BL

MOV AH, 9 ; set string output function

LEA DX, NEXT ; new line
INT 21H

LEA DX, msg_2 ; message 2
INT 21H

MOV AH, 1 ; set input function
INT 21H ; read second character

MOV BH, AL ; save second character into BH
MOV AH, 9 ; set string output function

LEA DX, NEXT ; next line
INT 21H

LEA DX, msg_3 ; message3
INT 21H

MOV AH, 2 ; set output function

CMP BL, BH

JAE Larger_
MOV DL, BH
INT 21H

MOV DL,  BL
INT 21H

JMP END

Larger_:
MOV BH, BL
INT 21H

MOV DL,BH
INT 21H
 END:

MOV AH, 4CH



INT 21H
MAIN ENDP
END MAIN
```

## Even or Odd

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

ev db 'Even$'
od db 'Odd$'
.code
main proc
mov ax,@data
mov ds,ax

mov ah,1
int 21h

mov bl,2
div bl

cmp ah,0
je IsEven

mov dx,10
mov ah,2
int 21h
mov dx,13
mov ah,2
int 21h


mov ah,9
LEA DX,od
```

```
int 21h
mov ah,4ch
int 21h

IsEven:
mov ah,2
int 21h
mov dx,13
mov ah,2
int 21h

 mov ah,9
LEA DX,ev
int 21h
mov ah,4ch
int 21h

int 21h
mov ah,4ch
int 21h

main endp
end main
```

## separate even and odd



```
Enter 6 Integer Numbers: 5 6 4 6 7 5
ODD Digits: 5 7 5
EVEN Digits: 6 4 6
```

```
ORG 100h

.DATA
    PROMPT_1 DB 'Enter 6 Integer Numbers: ', '$'
    PROMPT_2 DB 0Dh, 0Ah, 'ODD Digits: ', '$'
    PROMPT_3 DB 0Dh, 0Ah, 'EVEN Digits: ', '$'
    ARRAY DB 10 DUP(0)
    odd_sum DB ?
    even_sum DB ?
.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX

    MOV AH, 9
    LEA DX, PROMPT_1
    INT 21H

    MOV CX, 6                   ; because we will input 6 int
egers
    LEA SI, ARRAY

INPUTS:
    MOV AH, 1
    INT 21h

    MOV [SI], AL                ; Load the inputs in array on
e by one
    INC SI

    MOV AH, 2
    MOV DX, ' '
    INT 21h
    LOOP INPUTS
    CALL Odd_Numbers
    CALL Even_Numbers

MAIN ENDP

Odd_Numbers PROC

    MOV AH, 9
    LEA DX, PROMPT_2
    INT 21H

    MOV CX, 6
    LEA SI, ARRAY
    XOR BH,BH

Loop_1:
    XOR AX, AX
    MOV AL, [SI]
    SUB AL, 48

    MOV BL, 2                   ; compare the integer with al
l elements of the one by one
    DIV BL

    CMP AH, 1
    JE Print1
    JL noPrint1

    Print1:
        MOV AH, 2
        MOV DX, [SI]
        INT 21h

        MOV DX, ' '
        INT 21h
        add bh, [SI]
        sub bh,48
    noPrint1:
        INC SI

    LOOP Loop_1

mov odd_sum,bh
Odd_Numbers ENDP

Even_Numbers PROC

    MOV AH, 9
    LEA DX, PROMPT_3
    INT 21H

    MOV CX, 6
    LEA SI, ARRAY
    XOR BH,BH
Loop_2:
    XOR AX, AX
    MOV AL, [SI]
    SUB AL, 48

    MOV BL, 2                  ; compare the integer with al
l elements of the one by one
    DIV BL

    CMP AH, 0
    JE Print2
    JG noPrint2

    Print2:
        MOV AH, 2
        MOV DX, [SI]
        INT 21h
        MOV DX, ' '
        INT 21h
        add bh, [SI]
        sub bh,48
    noPrint2:
        INC SI
        LOOP Loop_2

mov even_sum,bh
Even_Numbers ENDP

END MAIN
RET
```

## Print Letters in alphabetical order

```
.model
.stack 100h


;add your code here
.DATA

msg1 db 10,13, 'enter two capital letters:$'
msg2 db 10,13,'please enter capital letters$'
msg3 db 10,13,'in alpharbatical order it is : $'

.code





MAIN PROC
       MOV AX, @DATA
        MOV DS,AX
```

```
mov ah,9
lea dx,msg1
int 21h

mov ah,1
int 21h

mov bl,al


  mov ah,1
int 21h

mov bh,al


cmp bl,bh
jge   LEVEL1

cmp bh,bl
jge LEVEL2


LEVEL1:
mov ah,9
lea dx,msg3
int 21h

  mov ah,2
  mov  dl,bh
  int 21h
  mov ah,2
  mov dl,bl
  int 21h
```

```
      JMP EXIT

            LEVEL2:
        mov ah,9
        lea dx,msg3
        int 21h

          mov ah,2
          mov  dl,bl
          int 21h
          mov ah,2
          mov dl,bh
          int 21h

           EXIT:
           mov ah,4ch
           int 21h



    MAIN ENDP
END MAIN
```

## Print two characters in reverse alphabetic order using assembly language.

```
.model
.stack 100h


;add your code here
```

```
.DATA

msg1 db 10,13, 'enter two capital letters:$'
msg2 db 10,13,'please enter capital letters$'
msg3 db 10,13,'in alpharbatical order it is : $'

.code





MAIN PROC
       MOV AX, @DATA
        MOV DS,AX

        mov ah,9
        lea dx,msg1
        int 21h

        mov ah,1
        int 21h

        mov bl,al


          mov ah,1
        int 21h

        mov bh,al






        LEVEL1:
        mov ah,9
```

```
         lea dx,msg3
         int 21h

           mov ah,2
           mov  dl,bh
           int 21h
           mov ah,2
           mov dl,bl
           int 21h





            EXIT:
            mov ah,4ch
            int 21h




     MAIN ENDP
 END MAIN
```

## jmp and cmp statement

### Write some code to compare two register ax and bx then greater put to the cx register

```
;suppose AX and BX contain signed numbers
 ; write some code to put the biggest one in CX

 .model small
 .stack 100h
 .data
 .code

 main proc



    mov ax,5
    mov bx,7

     cmp ax,bx
     jg label1:    ;ax>bx  // jl ax<bx ki na

      mov cx,bx ; uporer ta sotti na hole etate jabe
       jmp return

       label1:
       mov cx,ax

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

## Find out the largest number between two numbers using assembly language.

```
.model small
.stack 100h
.data
msg1 db 10,13, 'enter 1st number: $'
msg2 db 10,13,'enter 2nd number : $'

msg3 db 10,13,'1st greater$'

 msg4 db 10,13,'2nd greater$'
 msg5 db 10,13,'equal$'

 num1 db ?
 num2 db ?
 .code
 main proc
  mov ax,@data
  mov ds,ax

     mov ah,9
     lea dx,msg1
     int 21h


     mov ah,1
     int 21h

     mov num1,al


         mov ah,9
     lea dx,msg2
     int 21h


     mov ah,1
```

```
int 21h

mov num2,al

;compare

mov bl,num1

cmp bl,num2

jg first
je equal
jl less


first:
mov ah,9
lea dx,msg3
int 21h
jmp exit


equal:
mov ah,9
lea dx,msg5
int 21h
jmp exit


less:
mov ah,9
lea dx,msg4
int 21h
jmp exit
```

```
          exit:
          mov ah,4ch
          int 21h

     main endp
  end
```

## find character is vowel or consonant

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


  msg db 10,13,'enter the character$'
  msg1 db 10,13,' char is vowel$'
  msg2 db 10,13,'char is consonant$'


  .code

  main proc
    mov ax,@data
    mov ds,ax


    mov ah,09
    lea dx,msg
```

```
int 21h

mov ah,01h
int 21h



cmp al,'a'
je vowel

 cmp al,'e'
je vowel


 cmp al,'i'
je vowel

 cmp al,'o'
je vowel

 cmp al,'u'
je vowel




  mov ah,9
  lea dx, msg2
  int 21h
   jmp exit




vowel:
mov ah,9
lea dx,msg1
```

```
    int 21h

      exit:
      mov ah,4ch
      int 21h


        main endp
```

## Take a number input from user, check whether the given number is divisible by 5 or not.

```
.model small
.stack 100h

.data
    prompt db 10,13 ,'Enter a number: $'
    divisible db 10,13, 'The number is divisible by 5. $'
    not_divisible db 10,13, 'The number is not divisible by 5.

.code



main proc
    mov ax,@data
    mov ds,ax

    mov ah, 9
    lea dx, prompt
    int 21h
    ; read a number from input
    mov ah, 1
    int 21h
    sub al, 48 ; convert the ASCII digit to a number
    mov bl, al ; store the number in bl

    ; check if the number is divisible by 5
    mov ax, 0
    mov al, bl
    mov dl, 5
    div dl ; divide the number by 5
    cmp ah, 0 ; check if the remainder is 0
    jne not_divisible1 ; jump to "not_divisible" if the remainde

    ; display "divisible" message
    mov ah, 9
    lea dx, divisible
    int 21h
    jmp exit

not_divisible1:

    mov ah, 9
    lea dx, not_divisible
    int 21h

exit:
    ; exit program
    mov ah, 4ch
    int 21h

   main endp
```

## Array summation

```
org 100h

.DATA ; Data segment starts
 A db 3, 1, 2, 2, 1 ;1-D array for number
 B db 00h
 message db 'Enter the value of N:$' ;1-D array for string

 .CODE ; Code segment starts
 MAIN PROC
 mov ax, @DATA
 mov ds, ax

 xor ax,ax
 mov si, OFFSET A
 mov di, OFFSET B

 mov dx, OFFSET message ; Load Effective Address of the message
 ; lea dx, message ; (similar meaning that Load Effective Addres
 mov ah, 09h ;display string function
 int 21h ;display message

 mov ah, 01h
 int 21h
 mov cl, al
 sub cl, 48 ; to convert the ascii value of 3 to decimal 3

 xor al, al

 Loop_1:
 add al, [Si]
 inc Si
 loop Loop_1

 mov bl, al
 add bl, 48 ; to convert the ascii value of the output to decima
 mov ah, 02h
 mov dl, 0Dh ; Clear Buffer
 int 21h
 mov dl, 0Ah ; for newline
 int 21h

 mov dl, bl
 int 21h

 mov ah, 4ch
 int 21h

 MAIN ENDP
 END MAIN
 RET
```

## array duplication of summetion

```
org 100h

.DATA ; Data segment starts
 A db 3  2 DUP(2),5,7,8;1-D array for number

 B db 00h
 message db 'Enter the value of N:$' ;1-D array for string

 .CODE ; Code segment starts
 MAIN PROC
 mov ax, @DATA
 mov ds, ax
```

```
xor ax,ax
mov si, OFFSET A
mov di, OFFSET B

mov dx, OFFSET message ; Load Effective Address of the message
; lea dx, message ; (similar meaning that Load Effective Addres
mov ah, 09h ;display string function
int 21h ;display message

mov ah, 01h
int 21h
mov cl, al
sub cl, 48 ; to convert the ascii value of 3 to decimal 3

xor al, al

Loop_1:
add al, [Si]
inc Si
loop Loop_1

mov bl, al
add bl, 48 ; to convert the ascii value of the output to decima

mov ah, 02h
mov dl, 0Dh ; Clear Buffer
int 21h
mov dl, 0Ah ; for newline
int 21h

mov dl, bl
int 21h

mov ah, 4ch
int 21h
```

```
  MAIN ENDP
  END MAIN
  RET
```

## input 3 char and print it new line

```
.model small
.stack 100h
.data
a db 'enter three initials: $ '
.code

main proc
    mov ax,@data
    mov ds,ax

    mov ah,9
    lea dx,a
    int 21h

    ;input 1
    mov ah,1
    int 21h
    mov bl,al
          ;input 2
    mov ah,1
    int 21h
    mov cl,al
           ;input 3
    mov ah,1
```

```
int 21h
mov bh,al



mov ah,2
mov dl,10
int 21h
mov dl,13
int 21h


  mov ah,2
   mov dl,bl
  int 21h

 mov ah,2
mov dl,10
int 21h
mov dl,13
int 21h


      mov ah,2
   mov dl,cl
  int 21h

        mov ah,2
mov dl,10
int 21h
mov dl,13
int 21h


   mov ah,2
   mov dl, bh
```

```
      int 21h

      mov ah,2
    mov dl,10
    int 21h
    mov dl,13
    int 21h



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

## loop

### normal loop print star

```
.model small
.stack 100h

.data


.code


main proc

   mov cx,80 ; 80 bar print korbo tai 80
   mov ah,2
   mov dl,'*'

   top:
   int 21h

   loop top


   main endp
```

## Array

### print array using loops

```
;print array using loops
.model
.stack 100h
.data

arr1 db 1,2,3,4

.code

main proc
    mov ax,@data
    mov ds,ax
    mov si,offset arr1 ; arr1 er first address jabe

mov cx,4

print:

mov dl,[si]

add dl,48

mov ah,2
int 21h
  inc si
loop print



    main endp
end main
```

## summation 1 to n

```
.model
.stack 100h
.data

   i dw 1

   adds dw ?
.code
```

```
main proc

    mov ax,@data
    mov ds,ax

    mov ah,1
    int 21h

    sub al,48
      mov cx,0
    mov cl,al

    mov bx,0
    sum:
    add bx,i

      inc i
      loop sum

      mov adds,bx

    main endp
end main
```

## single digit summation

```
;single digit summation
.model
.stack 100h
.data
```

```
   i db 1



main proc

    mov ax,@data
    mov ds,ax

    mov ah,1
    int 21h

    sub al,48
      mov cx,0
    mov cl,al

    mov bl,0
    sum:
    add bl,i

      inc i
      loop sum

    mov dl,bl
     add dl,48
    mov ah,2
    int 21h


    main endp
end main
```

## Summation of first N numbers using assembly language.

```
.model small
.stack 100h

.data
    prompt db 'Enter a number: $'
    result db 'The sum of the first N numbers is: $'

.code



main proc
    mov ax,@data
    mov ds,ax

    mov ah, 9
    lea dx, prompt
    int 21h

    ; read a number from input
    mov ah, 1
    int 21h
    sub al, '0' ; convert the ASCII digit to a number
    mov cl, al ; store the number in cl

    ; calculate the sum of the first N numbers
    mov dl, 0 ; initialize the counter to 0
    mov bl, 0 ; initialize the sum to 0
start:
    inc dl ; increment the counter
    add bl, dl ; add the counter to the sum
    cmp dl, cl ; compare the counter with N
    jne start ; jump to "start" if the counter is not equal to

    ; display the result
    mov ah, 9
    lea dx, result
    int 21h

    ; print the sum
    mov ah, 2
    mov dl, bl
    add dl, '0' ; convert the sum to ASCII
    int 21h

    ; exit program
    mov ah, 4ch
    int 21h
main endp
```

## Implement a loop to find out the summation of 1+2+3+.....+100, also try to implement it without loop.(Use formula).

```
.model small
.stack 100h

.data
    prompt db 'Enter a number: $'
    result db 'The sum of the first N numbers is: $'
    result_100 db 'The sum of the first 100 numbers is: $'

.code


main proc

    mov ax,@data
    mov ds,ax



    ; calculate the sum of the first 100 numbers using a formula
    mov ax, 100 ; load the value of 100 into ax
    add ax, 1 ; add 1 to the value of ax
    mov bx, 100 ; load the value of 100 into bx
    mul bx ; multiply ax by bx
    mov cx, 2 ; load the value of 2 into cx
    div cx ; divide ax by cx
    sub ax, 101 ; subtract 101 from ax
    mov bx, ax ; copy the result to bx

    ; display the result
    mov ah, 9
    lea dx, result_100
    int 21h

    ; print the sum
    mov ah, 2
    mov dl, bl
    add dl, 48 ; convert the sum to ASCII
    int 21h

    ; exit program
    mov ah, 4ch
    int 21h
main endp
```

## factorial

```
.MODEL SMALL
.STACK 100H
.DATA

msg db "Enter the number: $"
msg1 db 0AH,0DH,"Factorial is: $"
n db ?
result DW ?
.CODE
MAIN PROC

MOV AX,@DATA
MOV DS,AX

LEA DX,msg
MOV AH,09
INT 21H

MOV AH,1
INT 21H
SUB AL,30H
MOV N,AL

XOR CX,CX
MOV CL,N
MOV AX,CX

MOV BX,CX
DEC BX

FACT:

MUL BX
DEC BX
```

```
CMP BX,0
JE STOP

LOOP FACT

STOP:
XOR BX,BX
MOV RESULT,AX
JMP PRINT
MOV AH,4CH
INT 21H

PRINT:
LEA DX,msg1
MOV AH,09
INT 21H

XOR AX,AX

MOV AX,RESULT
MOV BL,100
DIV BL
MOV BL,AL
MOV BH,AH
XOR AX,AX
MOV AL,BH
MOV BH,10
DIV BH
MOV BH,AL
MOV Cl,AH

MOV DL,BL
ADD DL,30H
MOV AH,2
INT 21H
MOV DL,BH
```

```
ADD DL,30H
INT 21H
MOV DL,CL
ADD DL,30H
INT 21H



MAIN ENDP
END MAIN
```

## clp1

## oddd and even

```
.model small
.stack 100h
.data
ev db 10,13, ' the number is Even$'
od db 10,13, ' the number is Odd$'
.code
main proc
mov ax,@data
mov ds,ax
mov ah,1
int 21h

mov bl,2
div bl
cmp ah,0

je IsEven
```

```



mov ah,9
LEA DX,od
int 21h
mov ah,4ch
int 21h

IsEven:



mov ah,9
LEA dx,ev
int 21h

mov ah,4ch
int 21h






main endp
end main
```

## avg 3 num

```
.model small
.stack 100h

.data
num1 db ?
num2 db ?
num3 db ?
result  db  ?

.code
main proc



    mov ah, 1
    int 21h
    add al, 48
    mov num1, al


    mov ah, 1
    int 21h
    add al, 48
    mov num2, al


    mov ah, 1
    int 21h
    add al, 48
    mov num3, al
    mov al, num1
    add al, num2
    add al, num3
    mov bl, 3
    div bl
    mov result, al


 mov dl, 10
mov ah, 02h
int 21h
mov dl, 13
mov ah, 02h
int 21h

    mov ah, 2
    mov dl, result
    sub dl, 48
    int 21h

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

## print asci 1 to `18

```
.STACK 100H

 .DATA
```

```
    e  DB  'The 128 ASCII Characters are : $'

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

    LEA DX,e
    MOV AH, 9
    INT 21H

    MOV CX, 128

    MOV AH, 2
    MOV DL, 0
    emon:
      INT 21H

      INC DL
    LOOP emon

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

## Procedure

example

```
.model small

   .stack 100h
   .data
   str1 db "hello$"
   str2 db "how are you$"
   str3 db "good to see you$"

   .code

   main proc

    mov ax ,@data
    mov ds,ax

    mov ah,9
    lea dx,str1
       int 21h

    call enterkey


        mov ah,9
    lea dx,str2
    int 21h

    call enterkey

      mov ah,9
```

```
     lea dx,str3
     int 21h

     call enterkey
     ;nicher ei code na likhle unlimited cholbe

        mov ah,4ch
        int 21h


     main endp




            enterkey proc

             mov dx,10
             mov ah,2
             int 21h

             mov dx,13
             mov ah,2
             int 21h

             ret
             enterkey endp
    end main
```

## Macro

example

```
print macro p1
    mov ah,9
    lea dx,p1
    int 21h

    mov dx,10
    mov ah,2
    int 21h
    mov dx,13
    mov ah,2
    int 21h
endm





.model small
.stack 100h

.data
str1 db "hello$"
str2 db "it is a test program$"

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


    print str1
    print str2
    mov ah,4ch
    int 21h
    main endp
end main
```

## Write an Assembly Language code that takes an input ARRAY and passes the array values and address to a MACRO. Using the array, address and one procedure separate out the ODD digits and EVEN digits.

```
ORG 100h
MDSPLY_STRING MACRO STRING

    MOV AH, 9
    LEA DX, STRING
    INT 21H

    MOV CX, 6

    mov si,offset ARRAY
ENDM

CMP_EVENODD MACRO
    XOR AX, AX
    MOV AL, [SI]
    SUB AL, 48

    MOV BL, 2
    DIV BL
ENDM
```

```
.DATA
    PROMPT_1 DB 'Enter 6 Integer Numbers: ', '$'
    PROMPT_2 DB 0Dh, 0Ah, 'ODD Digits: ', '$'
    PROMPT_3 DB 0Dh, 0Ah, 'EVEN Digits: ', '$'
    ARRAY DB 10 DUP(0)

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

    MDSPLY_STRING PROMPT_1      ; 1st Call of the MACRO

INPUTS:
    MOV AH, 1
    INT 21h

    MOV [SI], AL                ; Load the inputs in array one
    INC SI

    MOV AH, 2
    MOV DX, ' '
    INT 21h
    LOOP INPUTS

    CALL Odd_Numbers
    CALL Even_Numbers

MAIN ENDP

Odd_Numbers PROC
    MDSPLY_STRING PROMPT_2      ; 2nd Call of the MACRO

Loop_1:
    CMP_EVENODD
    CMP AH, 1
    JE Print1
    JNE noPrint1

    Print1:
        MOV AH, 2
        MOV DX, [SI]
        INT 21h

        MOV DX, ' '
        INT 21h

    noPrint1:
        INC SI
        LOOP Loop_1

Odd_Numbers ENDP

Even_Numbers PROC
    MDSPLY_STRING PROMPT_3      ; 3rd Call of the MACRO

Loop_2:
    CMP_EVENODD

    CMP AH, 0
    JE Print2
    JNE noPrint2

    Print2:
        MOV AH, 2
        MOV DX, [SI]
        INT 21h

        MOV DX, ' '
        INT 21h
    noPrint2:
        INC SI
        LOOP Loop_2

Even_Numbers ENDP



END MAIN
RET
```

**Write an Assembly Language code that takes an input ARRAY and passes the array values and address to a MACRO. Now produce the summation of odd digits and even digits as output.**

```



ORG 100h
MDSPLY_STRING MACRO STRING

    MOV AH, 9
    LEA DX, STRING
    INT 21H

    MOV CX, 6
    LEA SI, ARRAY
ENDM

CMP_EVENODD MACRO
    XOR AX, AX
    MOV AL, [SI]
    SUB AL, 48
    MOV BL, 2
    DIV BL
ENDM

SUM MACRO
    ADD BH, [SI]
    SUB BH, 48
ENDM



.DATA
    PROMPT_1 DB 'Enter 6 Integer Numbers: ', '$'
    PROMPT_2 DB 0Dh, 0Ah, 'ODD Digits: ', '$'
    PROMPT_3 DB 0Dh, 0Ah, 'EVEN Digits: ', '$'
    ARRAY DB 10 DUP(0)
    Odd_sum  db ?
    Even_sum  db ?



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

    MDSPLY_STRING PROMPT_1      ; 1st Call of the MACRO

INPUTS:
    MOV AH, 1
    INT 21h

    MOV [SI], AL                ; Load the inputs in array one 
    INC SI

    MOV AH, 2
    MOV DX, ' '
    INT 21h
    LOOP INPUTS

    CALL Odd_Numbers
    CALL Even_Numbers





MAIN ENDP

Odd_Numbers PROC
    MDSPLY_STRING PROMPT_2      ; 2nd Call of the MACRO

    XOR BH, BH

Loop_1:
    CMP_EVENODD

    CMP AH, 1
    JE Print1
    JNE noPrint1

    Print1:
        MOV AH, 2
        MOV DX, [SI]
        INT 21h

        MOV DX, ' '
        INT 21h

        SUM


    noPrint1:
        INC SI
        LOOP Loop_1

MOV Odd_sum, BH
Odd_Numbers ENDP

Even_Numbers PROC
    MDSPLY_STRING PROMPT_3      ; 3rd Call of the MACRO

    XOR BH, BH

Loop_2:
    CMP_EVENODD

    CMP AH, 0
    JE Print2
    JNE noPrint2

    Print2:
        MOV AH, 2
        MOV DX, [SI]
        INT 21h

        MOV DX, ' '
        INT 21h

        SUM



    noPrint2:
        INC SI
        LOOP Loop_2

MOV Even_sum, BH
Even_Numbers ENDP
```

```
END MAIN
RET
```

```
MDSPLY_STRING MACRO STRING

MOV DX, OFFSET STRING

MOV AH, 09H
INT 21H

ENDM

.MODEL SMALL
.STACK 100H
.DATA

n db ?
odd db ?
even db ?

R db ?
z db ?

A db n dup (?)




msg1 DB "Enter the number of input: $"
msg2 DB 0AH,0DH,"Enter the all element : $"

msg3 DB "  $"
```

```
msg5 DB 0AH,0DH,"The summation of odd: $"
msg6 DB 0AH,0DH,"The summation of even: $"

.CODE
MAIN PROC




mov ax, @DATA
mov ds, ax

mov odd,0
mov even, 0
mov bx,0

MDSPLY_STRING  msg1

mov ah,1
int 21H
sub al,30H
mov n,al

MDSPLY_STRING msg2


xor cx,cx
mov cl,n
mov si,0



Loop_1:

mov ah,1
int 21H
sub al,30H
```

```
mov A[si],al


inc si

MDSPLY_STRING  msg3


loop Loop_1


xor cx,cx
mov cl,n
mov z,2
mov si,0

Loop_2:


MOV AX,00
MOV AL,A[si]
DIV z
mov di,si
inc si

CMP AH,00

JZ RESULT_1

JNZ RESULT_2




RESULT_1:
mov bl,even
```

```
add bl,A[di]

mov even,bl

RESULT_2:



loop Loop_2

xor cx,cx
mov cl,n
mov si,0

Loop_3:



MOV AX,00
MOV AL,A[si]
DIV z
mov di,si
inc si

CMP AH,00

JNZ RESULT_3

JZ RESULT_4




RESULT_3:

mov bl,odd
add bl,A[di]
mov odd,bl
```

```
RESULT_4:

loop Loop_3

MDSPLY_STRING  msg5

mov R,10

xor ax,ax

mov al,odd

div R
mov bh,ah

mov ah,2
mov dl,al
add dl,30H
int 21H

mov ah,2
mov dl,bh
add dl,30H
int 21H

MDSPLY_STRING  msg6



xor ax,ax

mov al,even

div R
mov bh,ah
mov ah,2
mov dl,al
add dl,30H
int 21H

mov ah,2
mov dl,bh
add dl,30H
int 21H









mov ah, 4ch
int 21h

MAIN ENDP
END MAIN




; [SOURCE]: C:\emu8086\MySource\lab 6 &7.asm
```

## sum of squared numbers

```
org 100h

.DATA
A db 1,2,3,4,5,6      ;1-D array for number
output db ?
```

```
.CODE
MAIN PROC
    mov ax, @DATA
    mov ds, ax

    xor ax,ax
    mov si, OFFSET A


    mov CX, 6

    Loop_1:

        MOV AL, [SI]
        XOR AH, AH  ; to clear ah

        MOV BL, [SI]
        XOR BH, BH

        MUL BL
        ADD DL, AL

        inc Si
        loop Loop_1

    mov output, dl

    sub output,48

    mov dl,output

    mov ah,2
    int 21h
    mov ah, 4ch
    int 21h

    MAIN ENDP
    END MAIN
RET
```

## Write an Assembly Language Program (ALP) to put the sum 1+4+7+...+25 in AX.

```
.model small
.stack 100h

.data
    sum dw 0
    c dw 1

.code
    mov ax, @data
    mov ds, ax

    mov ax, 0
    mov cx, 1

loop_start:
    add ax, cx
    add cx, 3

    cmp cx, 25
    jle loop_start



    mov sum, ax




    mov ax, sum
    call print_num


    mov ax,4ch
    int 21h

print_num proc
    push ax
    push bx

    mov bx, 10
    xor cx, cx

print_num_loop:
    xor dx, dx
    div bx
    push dx

    inc cx

    cmp ax, 0
    jnz print_num_loop
```

```
display:
    pop dx
    add dl, '0'
    mov ah, 02h
    int 21h

    loop display

    pop bx
    pop ax

    ret
print_num endp

endp
```

## Write an Assembly Language Program to print the following structure with a procedure.

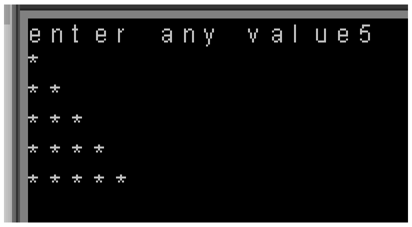
```
Input: A
     ***
     *A*
     ***
```

```
.MODEL SMALL
.STACK 100H

.DATA
A DB ?
```

```
.CODE

MAIN PROC

    MOV AH, 01H
    INT 21H
    MOV A, AL
      MOV DL, 10
    INT 21H
    CALL PRINT_STRUCTURE

    MOV AH, 4CH
    INT 21H

MAIN ENDP

PRINT_STRUCTURE PROC

    MOV AH, 02H
    MOV DL, '*'
    INT 21H
    MOV DL, '*'
    INT 21H
    MOV DL, '*'
    INT 21H

    MOV DL, 10
    INT 21H

    MOV DL, '*'
    INT 21H
    MOV DL, A
    INT 21H
    MOV DL, '*'
    INT 21H
    MOV DL, 10
    INT 21H

    MOV AH, 02H
    MOV DL, '*'
    INT 21H
    MOV DL, '*'
    INT 21H
    MOV DL, '*'
    INT 21H

    RET

PRINT_STRUCTURE ENDP
```

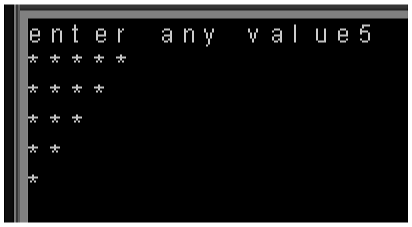
## Final topic

## Print Triange



```
enter any value5
*
* *
* * *
* * * *
* * * * *
```

```
.model small
 .stack 100h

 .data
 msg db "enter any value$"
var db ?
 .code

 main proc

    mov ax,@data
    mov ds,ax


 lea dx,msg
 mov ah,9
 int 21h


 mov ah,1
 int 21h

 sub al,48

 mov var,al

mov dx,13
  mov ah,2
  int 21h
  mov dx,10
  mov ah,2
  int 21h


   mov cx,0
   mov cl,var

   mov bl,1


   TOP:

   mov cx,bx

   level1:
   cmp bl,var
   JG exit
   mov ah,2
   mov dl,'*'
   int 21h
   loop level1
   inc bl

       mov dx,13
      mov ah,2
      int 21h
      mov dx,10
      mov ah,2
      int 21h
  loop TOP




    exit:
    mov ah,4ch
    int 21h

    main endp
 end main
```



```
enter any value5
* * * * *
* * * *
* * *
* *
*
```

```
.model small
 .stack 100h

 .data
 msg db "enter any value$"
var db ?
 .code

 main proc

    mov ax,@data
    mov ds,ax



 lea dx,msg
 mov ah,9
 int 21h



 mov ah,1
 int 21h

 sub al,48
 mov var,al

mov dx,13
  mov ah,2
  int 21h
  mov dx,10
  mov ah,2
  int 21h



   mov cx,0

   mov cl,var

   mov bl,var



   TOP:

   mov cx,bx

   level1:
   cmp bl,0
   Je exit
   mov ah,2
   mov dl,'*'
   int 21h
   loop level1
   dec bl

       mov dx,13
      mov ah,2
      int 21h
      mov dx,10
      mov ah,2
      int 21h
   loop TOP




    exit:
    mov ah,4ch
    int 21h

    main endp

  end main
```

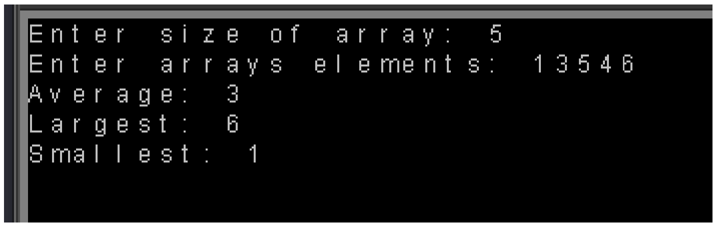
## summation 1+3+5+7+n

```
ADDER MACRO NUM
ADD AX,NUM
MOV RESULT,AX
ENDM

.MODEL SMALL
.STACK 100H
.DATA

msg db "Enter the number: $"
msg1 db 0AH,0DH,"Summition is: $"
n db ?
M dw ?
result DW ?
.CODE
MAIN PROC
```

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

LEA DX,msg
MOV AH,09
INT 21H

MOV AH,1
INT 21H
SUB AL,48
MOV N,AL

XOR CX,CX
XOR AX,AX
XOR BX,BX
MOV CL,N
MOV BX,1




LABEL:

MOV M,BX
ADDER M
INC BX
INC BX
MOV M,BX

LOOP LABEL

LEA DX,msg1
MOV AH,09
INT 21H

OUTPUT:
```

```
XOR AX,AX
MOV AX,RESULT
MOV BL,100
DIV BL
MOV BL,AL
MOV BH,AH
XOR AX,AX
MOV AL,BH
MOV BH,10
DIV BH
MOV BH,AL
MOV Cl,AH

MOV DL,BL
ADD DL,30H
MOV AH,2
INT 21H
MOV DL,BH
ADD DL,30H
INT 21H
MOV DL,CL
ADD DL,30H
INT 21H

MAIN ENDP
END MAIN
```

## write a code that store array and ans me to avarage ,largest and smallest


```
Enter size of array: 5
Enter arrays elements: 13546
Average: 3
Largest: 6
Smallest: 1
```

```
.MODEL SMALL
.STACK 100H
.DATA
n db ?
arr db n dup(?)
msg db "Enter size of array: $"
msg1 db 0AH,0DH,"Enter arrays elements: $"
msg2 db 0aH,0DH,"Average: $"
msg3 db 0AH,0DH,"Largest: $"
msg4 db 0AH,0DH,"Smallest: $"

total dw ?
average_ans db ?
large db ?
small db ?
.CODE
MAIN PROC
MOV AX,@DATA
MOV DS,AX

LEA DX,msg
MOV AH,09
INT 21H

MOV AH,1
INT 21H
SUB AL,30H
MOV N,AL
```

```
LEA DX,msg1
MOV AH,09
INT 21H

XOR CX,CX
MOV CL,N
MOV SI,0

INPUT:
MOV AH,1
INT 21H
SUB AL,30H
MOV ARR[SI],AL
INC SI
LOOP INPUT

CALL AVERAGE
CALL LARGEST
CALL SMALLEST
CALL OUTPUT
MAIN ENDP

AVERAGE PROC
XOR CX,CX
XOR AX,AX
MOV BX,BX
MOV CL,N
MOV SI,0
MOV BX,0

AVG:
ADD BL,ARR[SI]
INC SI
LOOP AVG

MOV TOTAL,BX
MOV AX,TOTAL
MOV BL,N
DIV N
MOV average_ans,AL
RET
AVERAGE ENDP

LARGEST PROC
XOR CX,CX
XOR AX,AX
MOV BX,BX
MOV CL,N
DEC CL

MOV SI,0
MOV BL,ARR[SI]
MOV LARGE,BL
MOV SI,1

COMPARE:
CMP BL,ARR[SI]
JL SWAP
JG SKIP
SWAP:
MOV BL,ARR[SI]
MOV LARGE,BL
SKIP:
INC SI
LOOP COMPARE
RET

LARGEST ENDP

SMALLEST PROC
XOR CX,CX
XOR AX,AX
```

```
MOV BX,BX
MOV CL,N
DEC CL

MOV SI,0
MOV BL,ARR[SI]
MOV SMALL,BL
MOV SI,1

COMPARE1:
CMP BL,ARR[SI]
JG SWAP1
JL SKIP1
SWAP1:
MOV BL,ARR[SI]
MOV SMALL,BL
SKIP1:
INC SI
LOOP COMPARE1
RET

SMALLEST ENDP

OUTPUT PROC
LEA DX,msg2
MOV AH,09
INT 21H

MOV DL,average_ans
ADD DL,30H
MOV AH,2
INT 21H

LEA DX,msg3
MOV AH,09
INT 21H
```

```
MOV DL,large
ADD DL,30H
MOV AH,2
INT 21H

LEA DX,msg4
MOV AH,09
INT 21H

MOV DL,small
ADD DL,30H
MOV AH,2
INT 21H
OUTPUT ENDP
END MAIN
```

## find array smallest and avarage value

```
.MODEL SMALL
.STACK 100H
.DATA
N1 DB ?
SM DB ?
LR DB 0
AV DW ?


M1 DB "Enter array size : $"
M2 DB 0DH,0AH,"Enter array element : $"
M3 DB 0DH,0AH,"The smallest value is : $"
```

```
M4 DB 0DH,0AH,"The largest value is : $"
M5 DB 0DH,0AH,"The average value is : $"


ARR1 DB N1 DUP (?)


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

MOV DX, OFFSET M1
MOV AH,09H
INT 21H

MOV AH,1
INT 21H
MOV N1,AL
SUB N1,48

MOV AH,09H
LEA DX,M2
INT 21H

XOR CX,CX
MOV CL,N1
MOV SI,0

LOOP_INPUT:
MOV AH,1
INT 21H
MOV ARR1[SI],AL
SUB AL,48
INC SI
```

```
LOOP LOOP_INPUT

CALL AVERAGE

CALL SMALL1

CALL  LARGEST







MOV AH,4CH
INT 21H
MAIN ENDP

LARGEST PROC

XOR DX,DX
XOR CX,CX
XOR BX,BX

MOV SI,0
MOV CL,N1
DEC CL
MOV BL,ARR1[SI]
SUB BL,30H
MOV LR,BL
MOV SI,1

L_LARGE:

CMP BL,ARR1[SI]
JL LARGE
JG U_J
```

```
LARGE:
MOV BL,ARR1[SI]
MOV LR,BL
U_J:
INC SI

LOOP L_LARGE



MOV AH,09H
LEA DX,M4
INT 21H

MOV DL,LR
ADD DL,30H
MOV AH,02
INT 21H

LARGEST ENDP
AVERAGE PROC

XOR AX,AX
XOR BX,BX
XOR CX,CX
XOR DX,DX
MOV SI,0
MOV CL,N1
DEC CL

MOV AL,ARR1[SI]
SUB AL,30H
INC SI

SUM:
MOV BL,ARR1[SI]
```

```
ADD AL,BL
INC SI

LOOP SUM

MOV AV,AX

MOV AH,09H
LEA DX,M5
INT 21H

MOV AX,AV
XOR BX,BX
MOV BL,N1
DIV BL


MOV DL,AL
ADD DL,48
MOV AH,2
INT 21H




AVERAGE ENDP


SMALL1 PROC

XOR DX,DX
XOR CX,CX
XOR BX,BX

MOV SI,0
MOV CL,N1
DEC CL
MOV BL,ARR1[SI]
SUB BL,30H
MOV SM,BL
MOV SI,1

L_SMALL:

SUB ARR1[SI],48
CMP BL,ARR1[SI]
JL SKIP
JG SMALL

SMALL:
MOV BL,ARR1[SI]
MOV SM,BL
SKIP:
INC SI

LOOP L_SMALL



MOV AH,09H
LEA DX,M3
INT 21H

MOV DL,SM
ADD DL,48
MOV AH,02
INT 21H



SMALL1 ENDP
```

```
END MAIN
```

## print value in assending order

```
org 100h

.data

str db 10,13,"Enter Values: $"
str1 db 0dh,0ah,"Assending Order: $"
array db 10dup(0)

.code

mov ah,9
lea dx,str
int 21h

mov cx,7
mov bx,offset array
mov ah,1

inputs:
int 21h
```

```
mov [bx],al
inc bx
Loop inputs

mov cx,10
dec cx

nextscan:
mov bx,cx
mov si,0

nextcomp:
mov al,array[si]
mov dl,array[si+1]
cmp al,dl

jc noswap

mov array[si],dl
mov array[si+1],al

noswap:
inc si
dec bx
jnz nextcomp

loop nextscan

mov ah,9
lea dx,str1
int 21h

mov cx,10
mov bx,offset array

; this loop to display elements on the screen
```

```
print:
mov ah,2
mov dl,[bx]
int 21h
inc bx
loop print

ret
```

### even sum on array

```


org 100h

.DATA
A db 1,2,3,4,5,6,7,8,9,10      ;1-D array for number
even_sum db ?



.CODE
MAIN PROC
    mov ax, @DATA
    mov ds, ax

    xor ax,ax
    mov si, OFFSET A
    XOR BH,BH

    mov CX, 10

    Loop_1:
        XOR AX, AX
        MOV AL, [SI]
        MOV BL, 2                  ; compare the integer wit
h all elements of the one by one
        DIV BL

        CMP AH, 0
        JE Print1
        JG noPrint1

    Print1:
        add bh, [Si]
    noPrint1:
        INC Si
     loop Loop_1
    mov even_sum, bh

    mov ah, 4ch
    int 21h

    MAIN ENDP
    END MAIN
RET
```

## odd sum using array

```
org 100h

.DATA
A db 1,2,3,4,5,6,7,8,9,10      ;1-D array for number
odd_sum db ?



.CODE
```

```
MAIN PROC
    mov ax, @DATA
    mov ds, ax

    xor ax,ax
    mov si, OFFSET A
    XOR BH,BH

    mov CX, 10

    Loop_1:
        XOR AX, AX
        MOV AL, [SI]

        MOV BL, 2                   ; compare the integer wit
h all elements of the one by one
        DIV BL

        CMP AH, 1
        JE Print1
        JL noPrint1

    Print1:
        add bh, [Si]
    noPrint1:
        INC Si
     loop Loop_1
    mov odd_sum, bh

    mov ah, 4ch
    int 21h

    MAIN ENDP
    END MAIN
```

## print asending and desending order

```
ORG 100h

.data
PROMPT_1 DB 'Enter 10 Integer Values: ', '$'
PROMPT_2 DB 0Dh, 0Ah, 'Ascending Order: ', '$'
PROMPT_3 DB 0Dh, 0Ah, 'Descending Order: ', '$'
ARRAY DB 10 DUP(0)

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

    MOV AH, 9
    LEA DX, PROMPT_1
    INT 21H

    MOV CX, 10          ; because we will input 10 integers
    LEA SI, ARRAY

INPUTS:
    MOV AH, 1
    INT 21h

    MOV [SI], AL        ; Load the inputs in array one by one
    INC SI

    MOV AH, 2
    MOV DX, ' '
    INT 21h
    LOOP INPUTS




    MOV CX, 10
    DEC CX              ; because n - 1 elements need to be c
hecked

First:
    MOV SI, CX
    MOV BX, 0

Second:
    MOV AL, ARRAY[BX]
    MOV DL, ARRAY[BX+1]
    CMP AL, DL

    JL noSwap

    MOV ARRAY[BX], DL
    MOV ARRAY[BX+1], AL

noSwap:
    INC BX
    DEC SI
    JNZ Second

    LOOP First

    CALL Ascending_Sort
    CALL Descending_Sort

MAIN ENDP

Ascending_Sort PROC
```

```
    MOV AH, 9
    LEA DX, PROMPT_2
    INT 21H

    MOV CX, 10
    LEA SI, ARRAY

Print1:                 ; This loop prints the ARRAY elements
on the screen
    MOV AH, 2
    MOV DL, [SI]
    INT 21H
    INC SI

    MOV DX, ' '
    INT 21h
    LOOP Print1

Ascending_Sort ENDP

Descending_Sort PROC

    MOV AH, 9
    LEA DX, PROMPT_3
    INT 21H

    MOV CX, 10
    LEA SI, ARRAY
    ADD SI, 9           ; to get the last address of the 10 s
ized array, we need to add 9 with the first address

Print2:                 ; This loop prints the ARRAY elements
on the screen
    MOV AH, 2
    MOV DL, [SI]
    INT 21H
    DEC SI

    MOV DX, ' '
    INT 21h
    LOOP Print2

Descending_Sort ENDP

END MAIN
RET
```