```
#addition
.model small
.stack 100h
.data
msg db "enter number:$"
var1 db ?
var2 db ?

.code

main proc
    mov ax,@data
    mov ds,ax

    mov ah,9
    lea dx,msg
    int 21h

    mov ah,1
    int 21h
    mov var1,al

    mov ah,1
    int 21h

    ;sub var1,'0'
    ;sub al,'0'

    add al,var1
```

```
    mov ah,0
    aaa
    add al,'0'
    add ah,'0'
    mov var2,ah

    mov ah,2
    mov dl,var2
    int 21h

    mov ah,2
    mov dl,al
    int 21h



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

```
#input output
.model small
.stack 100h

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

    mov ah,1
    int 21h
    mov bh,al


    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
```

```
#multiplication

.model small
.stack 100h
.data

var1 db ?



.code
main proc
    mov ah,1
    int 21h
    mov var1,al

    mov ah,1
    int 21h

    sub al,'0'
    sub var1,'0'

    mov ah,0
    mul var1
    aam
    mov ch,ah
    mov cl,al

    mov ah,2
```

```
    add ch,'0'
    mov dl,ch
    int 21h

    mov ah,2
    add cl,'0'
    mov dl,cl
    int 21h




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

```
#print 1 to 9
.model small
.stack 100h

.data

msg dw "Output all integer $"
counter db 10

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

    mov ah,9
    lea dx,msg
    int 21h

    mov cx,48

    position:

    mov ah,2
    mov dx,cx
    int 21h
    add cx,1

    mov ah,2
    mov dl,10
```

```
    int 21h
    mov dl,13
    int 21h

    sub counter,1
    cmp counter,0
    jg position

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

```
.model small
.stack 100h

.data

msg dw "Output a to z $"
counter db 26

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

    mov ah,9
    lea dx,msg
    int 21h

    mov cx,"A"

    position:

    mov ah,2
    mov dx,cx
    int 21h
    add cx,1

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

```
    mov dl,13
    int 21h

    sub counter,1
    cmp counter,0
    jg position

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

```
#palindrome/reverse
.model small
.stack 100h

.data

string db 20 dup('$')

line db 10,13,"$"
not_equal db 10,13,"Not equal$"
equal db 10,13,"equal$"



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

   mov ah,10       ;input command 10
   lea dx,string    ;load offset of string
   mov string,19     ;number of characters to input
   int 21h



   lea dx,line      ;don't use int 21h randomly only while performing interrupt operation
   mov ah,9
   int 21h
```

```
    ;lea dx,string+2
    ;mov ah,9
    ;int 21h



    mov si,offset string
    add si,2

    mov cl,string[1]      ;str[1]=number_of_input_characters
str[0]=number_of_defined_input_characters

    mov ch,0
    ;int 21h

    loop1:          ;pushing loop
       mov ah,2
       mov dx,[si]
       push dx
       inc si
       int 21h
    loop loop1

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

```
mov cl,string[1]
mov ch,0


mov si,offset string
add si,2

loop2:
   mov ah,2
   pop dx
   cmp dl,[si]
   jne n_equal
   inc si
   int 21h
loop loop2

mov ah,9
lea dx,equal
int 21h
jmp go


n_equal:
   mov ah,9
   lea dx,not_equal
   int 21h

 go:
```

```
    mov ah,4ch
    int 21h




    main endp
end main
```