Write an assembly program using 8086 instructions to count the number of characters entered by the user until the "Enter" key (carriage return) is pressed. The program should then display the total count of characters.

```
.model small
.stack 100h

.data
count db 0        ; Variable to store character count

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

   mov cl, 0       ; Initialize count
   mov ah, 1       ; Read character function

while_:

    int 21h        ; Read a character
   cmp al, 13      ; Check for Enter (ASCII 13)
   je word_count   ; If Enter, go to word_count
   inc cl          ; Increment count for each character

   jmp while_      ; loop while

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

   mov ah, 2       ; Display character function
   add cl, '0'     ; Convert number to ASCII
   mov dl, cl      ; Load DL with count
   int 21h         ; Print the count

   mov ah, 4Ch     ; Terminate program
   int 21h

main endp
end main
```

Write a assembly code to display the char '*' 20 times using loop instructions or using only conditional/unconditional jumps

```
.model small
.stack 100h

.data
count db 0        ; Variable to store character count

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

   mov cx, 20
   mov ah, 2
   mov dl, '*'

print_:
   int 21h
   ;loop print_
   dec cx
   jz exit_
   jmp print_

   exit_:
   mov ah, 4Ch     ; Terminate program
   int 21h

main endp
end main
```

//printing all ascii 256 chars

```
.model small
.stack 100h

.data

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

   mov cx, 256
   mov dl, 0
   mov ah, 2

print_loop:
   int 21h
   inc dl
   dec cx
   jnz print_loop
   ;loop print_loop we can also use this line by commenint out upper two lines

mov ah, 4ch
main endp
end main
```

//Put the sum of the first 50 terms of the arithmetic sequence 1, 5, 9, 13, ... in DX. Hints: Employ LOOP instructions to do the following

```
.model small
.stack 100h

.data

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


   mov cx, 50
   mov ax, 1
   mov bx, 0

sum_loop:
   add bx, ax
   add ax, 4
   loop sum_loop

mov dx, bx


mov ah, 4ch
```

```
main endp
end main
```

//Put the sum 100 + 95 + 90 + ... + 5 in AX. Hints: Employ LOOP instructions to do the following.

```
.model small
.stack 100h

.data

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

   mov cx, 19
   mov ax, 100
   mov bx, 0


sum_loop:
   add bx, ax
   sub ax, 5
   loop sum_loop

 mov ax, bx


mov ah, 4ch
main endp
end main
```

//Read a character and display it 50 times on the next line. Hints: use LOOP instructions and put cx = 50

```
.model small
.stack 100h

.data

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

   mov ah, 1
   int 21h
   mov bl, al

   mov ah, 2
   mov dl, 10
   int 21h

   mov dl, 13
   int 21h

   mov cx, 50
   mov dl, bl
   mov ah, 2

display_char_50_times:
        int 21h
        loop display_char_50_times


mov ah, 4ch
main endp
end main
```

//Write a program to check whether a given input character is a vowel or not.

```
.model small
.stack 100h

.data

vowel db "it is a vowel$"
n db "it is not a vowel$"
newline db 0ah, 0dh, "$"
;newline db 10, 13, "$"
.code
main proc
   mov ax, @data
   mov ds, ax

   mov ah, 1
   int 21h
   mov bl, al

   mov ah, 9
   lea dx, newline
   int 21h

   cmp bl, 'a'
   je is_vowel
     cmp bl, 'e'
   je is_vowel
     cmp bl, 'i'
   je is_vowel
    cmp bl, 'o'
   je is_vowel
    cmp bl, 'u'
   je is_vowel

   jmp no_vowel

   is_vowel:

   mov ah, 9
   lea dx, vowel
   int 21h

   jmp exit_

   no_vowel:
   mov ah, 9
   lea dx, n
   int 21h
exit_:
mov ah, 4ch
main endp
end main
```

//Take an input character from user. Check it for letter and convert upper to lower or lower to upper using logical instructions

```
.model small
.stack 100h

.data

vowel db "it is a vowel$"
n db "it is not a vowel$"
newline db 0ah, 0dh, "$"
;newline db 10, 13, "$"
.code
main proc
   mov ax, @data
   mov ds, ax

   mov ah, 1
   int 21h
   mov bl, al

   mov ah, 9
   lea dx, newline
   int 21h

   cmp bl, 'a'
   jl check_upper
   cmp bl, 'z'
   jg check_upper

lower_to_upper:
   and bl, 0dfh
   mov dl, bl
   mov ah, 2
   int 21h
   jmp exit_

 check_upper:
   cmp bl, 'A'
   jl exit_

   cmp bl, 'Z'
   jg exit_

 upper_to_lower:
   or bl, 20h
   mov dl, bl
   mov ah, 2
   int 21h



exit_:
mov ah, 4ch
```

```
main endp
end main
```

//Take an input character from user. Check it for number and find whether it is odd or even using TEST instruction.

```
.model small
.stack 100h

.data

error db "it is not a number$"
even db "even number $"
odd db "odd number $"
newline db 0ah, 0dh, "$"
;newline db 10, 13, "$"
.code
main proc
   mov ax, @data
   mov ds, ax

   mov ah, 1
   int 21h
   mov bl, al

   mov ah, 9
   lea dx, newline
   int 21h

   cmp bl, '0'
   jl error_
   cmp bl, '9'
   jg error_


ascii_to_number:
   and bl, 0fh  ; same as sub al, 48 ; sub al, 30h, sub al, '0'

check_even_odd:
   test bl, 1
   jz print_even


print_odd:
   mov ah, 9
    lea dx, odd
    int 21h
    jmp exit_

print_even:
      mov ah, 9
    lea dx, even
```

```
    int 21h
    jmp exit_

error_:

   mov ah, 9
    lea dx, error
    int 21h
exit_:
mov ah, 4ch
main endp
end main
```

//Write an assembly language program for Binary Input and Output

```
.model small
.stack 100h

.data

error db "it is not a number$"
even db "even number $"
odd db "odd number $"
newline db 0ah, 0dh, "$"
;newline db 10, 13, "$"
.code
main proc
   mov ax, @data
   mov ds, ax

  xor bx, bx
  mov cx, 8

input_binary:
   mov ah, 1
   int 21h
   and al, 0fh
   shl bl, 1
   or bl, al
   loop input_binary

 mov ah, 9
 lea dx, newline
 int 21h

 mov cx, 8

 output_binary:
   mov ah, 2
```

```
   shl bl, 1
   mov dl, '0'
   jnc print_binary
   mov dl, '1'

 print_binary:
   int 21h
   loop output_binary

exit_:
mov ah, 4ch
main endp
end main
```

//**binary input and reverse binary output**

```
.model small
.stack 100h

.data

error db "it is not a number$"
even db "even number $"
odd db "odd number $"
newline db 0ah, 0dh, "$"
;newline db 10, 13, "$"
.code
main proc
   mov ax, @data
   mov ds, ax

   xor bl, bl
   mov cx, 8

 input_binary:
   mov ah, 1
   int 21h
   and al, 0fh ; same as sub al, 48
   shl bl, 1
   or bl, al

   loop input_binary

mov ah, 9
lea dx, newline
int 21h

mov cx, 8
reverse_binary:
   mov ah, 2
   shr bl, 1
   mov dl, '0'
```

```
   jnc print_reverse
   mov dl, '1'


print_reverse:
   int 21h
   loop reverse_binary


exit_:
mov ah, 4ch
main endp
end main
```

//count the number of characters in the input until user press enter

```
.model small
.stack 100h

.data

newline db 0ah, 0dh, "$"
;newline db 10, 13, "$"
.code
main proc
   mov ax, @data
   mov ds, ax

mov bl, 0
mov ah, 1
int 21h

count_char:
   cmp al, 0dh
   je display_char_count
   inc bl
   int 21h
   jmp count_char

display_char_count:
   mov ah, 9
   lea dx, newline
   int 21h

   mov ah, 2
   ;add bl, '0'
   or bl, 30h

   mov dl, bl
   int 21h

exit_:
mov ah, 4ch
```

```
main endp
end main
```

//enter a char and display corresponding binary with the count of bit 1 of output binary

```
;binary input and reverse binary output(using shit and rotate)
.model small
.stack 100h

.data


input db "Pleae enter a binary number(8 bit):$"
output db "the number of 1 bits is : $"
newline db 0ah, 0dh, "$"
;newline db 10, 13, "$"
.code
main proc
   mov ax, @data
   mov ds, ax

   mov ah, 9
   lea dx, input
   int 21h

   mov ah, 1
   int 21h
   mov bl, al

   mov ah, 9
   lea dx, newline
   int 21h

mov cx, 8
mov bh, 0
print_binary:
       mov ah, 2
       shl bl, 1
       mov dl, '0'
       jnc print
       mov dl, '1'
       inc bh

 print:
 int 21h
 loop print_binary

count_bit_number:
 mov ah, 9
   lea dx, newline
   int 21h

    mov ah, 9
   lea dx, output
```

```
  int 21h
 mov ah,2
 or bh, 30h
 mov dl, bh
 int 21h


exit_:
mov ah, 4ch
main endp
end main
```