

Rajshahi University Of Engineering And Technology



Department Of Computer Science And Engineering

<i>Submitted By</i>	<i>Submitted To</i>
Name: Ashadullah Shawon	Dr. Md. Rabiul Islam
Class : 3 rd Year, Odd Semester	Professor
Roll No: 133009	Department Of CSE
Subject: Solution of Exercise Problems	RUET
Course Title: Microprocessor And Assembly Language	Rajshahi
Course No: CSE 3109	

Chapter 4

8. Write a program to (a) display a "?", (b) read two decimal digits whose sum is less than 10, (c) display them and their sum on the next line, with an appropriate message.

Sample execution:

?27

THE SUM OF 2 AND 7 IS 9

```
.model small
.stack 100h
.data
msg db 'THE SUM OF$'
msg1 db ' '$
msg2 db 'AND$'
msg3 db 'IS$'
```

```
.code
main proc
```

```
    mov dl,'?'
    mov ah,2
    int 21h
```

```
    mov ah,1
    int 21h
```

```
    mov bl,al
```

```
    mov ah,1
    int 21h
```

```
mov cl,al
```

```
mov ah,2  
mov dl,0dh      ; carriage return  
int 21h  
mov dl,0ah      ;new line  
int 21h
```

```
mov ax,@data ;initialize data segmet  
mov ds,ax  
lea dx,msg  
mov ah,9  
int 21h
```

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

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

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

```
lea dx,msg2  
mov ah,9  
int 21h
```

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

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

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

```
int 21h
```

```
lea dx,msg3  
mov ah,9  
int 21h
```

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

```
add bl,cl ;sum
```

```
sub bl,48d ;convert to ascii number
```

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

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

```
main endp
```

```
end main
```

Output:



The screenshot shows a window titled "emulator screen (80x25 chars)". The output displayed on the screen is:

```
?34  
THE SUM OF 3 AND 4 IS 7
```

9. Write a program to (a) prompt the user, (b) read first, middle, and last initials of a person's name, and (c) display them down the left margin.

Sample execution:

ENTER THREE INITIALS: JFK

J

F

K

```
.model small
.stack 100h
.data
msg db 'ENTER THREE INITIALS: $'
.code
main proc

    mov ax, @data ;initialize data segment
    mov ds, ax
    lea dx, msg
    mov ah, 9
    int 21h

    mov ah, 1
    int 21h
    mov bl, al

    mov ah, 1
    int 21h
    mov cl, al

    mov ah, 1
    int 21h
    mov bh, al

    mov ah, 2
    mov dl, 0dh ; carriage return
    int 21h
    mov dl, 0ah ; new line
    int 21h
```

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

```
mov dl,0dh  
int 21h  
mov dl,0ah  
int 21h
```

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

```
mov dl,0dh  
int 21h  
mov dl,0ah  
int 21h
```

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

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

```
main endp  
end main
```

Output:



10. Write a program to read one of the hex digits *A-F*, and display it on the next line in decimal.

Sample execution:

**ENTER A HEX DIGIT: C
IN DECIMAL IT IS 12**

```
.model small
.stack 100h
.data
msg1 db 'Enter a hex digit: $'
msg2 db 'In decimal it is: $'
.code
main proc

    mov ax,@data
    mov ds,ax
    lea dx,msg1
    mov ah,9
    int 21h

    mov ah,1
    int 21h
    mov bl,al
    sub bl,17d ; convert to corresponding hex value as C=67. So 67-17=50='2'
```

```
mov ah,2
mov dl,0dh
int 21h
mov dl,0ah
int 21h
```

```
lea dx,msg2
mov ah,9
int 21h
```

```
mov dl,49d ;print 1 at first
mov ah,2
int 21h
```

```
mov dl,bl
mov ah,2 ; print next value of hex after 1
int 21h
```

```
main endp
end main
```

Output:

A screenshot of a Windows-style window titled "emulator screen (80x25 chars)". The window has a black background with white text. The text displayed is "Enter a hex digit: C" on the first line and "In decimal it is: 12" on the second line. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
emulator screen (80x25 chars)
Enter a hex digit: C
In decimal it is: 12
```


11. Write a program to display a 10 x 10 solid box of asterisks.

***Hint:* declare a string in the data segment that specifies the box, and display it with INT 21h, function 9h.**

```
.model small
.stack 100h
.data
msg1 db '*****$'

.code
main proc
    mov ax,@data
    mov ds,ax
    lea dx,msg1
    mov ah,9           ;print 10 times
    int 21h
    mov ah,2
    mov dl,0dh
    int 21h
    mov dl,0ah
    int 21h
    lea dx,msg1
    mov ah,9
    int 21h
    mov ah,2
    mov dl,0dh
    int 21h
    mov dl,0ah
    int 21h
    lea dx,msg1
    mov ah,9
    int 21h

    mov ah,2
    mov dl,0dh
    int 21h
    mov dl,0ah
    int 21h
    lea dx,msg1
    mov ah,9
    int 21h

    mov ah,2
    mov dl,0dh
    int 21h
```

```
mov dl,0ah  
int 21h  
lea dx,msg1  
mov ah,9  
int 21h
```

```
mov ah,2  
mov dl,0dh  
int 21h  
mov dl,0ah  
int 21h  
lea dx,msg1  
mov ah,9  
int 21h
```

```
mov ah,2  
mov dl,0dh  
int 21h  
mov dl,0ah  
int 21h  
lea dx,msg1  
mov ah,9  
int 21h
```

```
mov ah,2  
mov dl,0dh  
int 21h  
mov dl,0ah  
int 21h  
lea dx,msg1  
mov ah,9  
int 21h
```

```
mov ah,2  
mov dl,0dh  
int 21h  
mov dl,0ah  
int 21h  
lea dx,msg1  
mov ah,9  
int 21h
```

```
mov ah,2  
mov dl,0dh  
int 21h  
mov dl,0ah
```

```

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

mov ah,4ch
int 21h

main endp
end main

```

Output:



12. Write a program to (a) display "?", (b) read three initials,(<;) display them in the middle of an 11 x 11 box of asterix, and (d) beep the computer.

```

.model small
.stack 100h
.data
msg1 db '*****$'
msg2 db '****$'

.code
main proc

```

```
mov dl,'?'  
mov ah,2  
int 21h
```

```
mov ah,1  
int 21h  
mov bl,al
```

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

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

```
mov ah,2  
mov dl,0dh  
int 21h  
mov dl,0ah  
int 21h
```

```
mov ax,@data  
mov ds,ax  
lea dx,msg1  
mov ah,9  
int 21h
```

```
mov ah,2  
mov dl,0dh  
int 21h  
mov dl,0ah  
int 21h  
lea dx,msg1  
mov ah,9  
int 21h
```

```
mov ah,2  
mov dl,0dh  
int 21h  
mov dl,0ah  
int 21h  
lea dx,msg1
```

```
mov ah,9  
int 21h
```

```
mov ah,2  
mov dl,0dh  
int 21h  
mov dl,0ah  
int 21h  
lea dx,msg1  
mov ah,9  
int 21h
```

```
mov ah,2  
mov dl,0dh  
int 21h  
mov dl,0ah  
int 21h  
lea dx,msg1  
mov ah,9  
int 21h
```

```
mov ah,2  
mov dl,0dh  
int 21h  
mov dl,0ah  
int 21h
```

```
lea dx,msg2 ; printing less star to put the scanned value  
mov ah,9  
int 21h
```

```
mov dl,bl  
mov ah,2 ;printing scanned value  
int 21h
```

```
mov dl,cl  
int 21h ;printing scanned value
```

```
mov dl,bh ;printing scanned value
```

int 21h

lea dx,msg2
mov ah,9
int 21h

mov ah,2
mov dl,0dh
int 21h
mov dl,0ah
int 21h
lea dx,msg1
mov ah,9
int 21h

mov ah,2
mov dl,0dh
int 21h
mov dl,0ah
int 21h
lea dx,msg1
mov ah,9
int 21h

mov ah,2
mov dl,0dh
int 21h
mov dl,0ah
int 21h
lea dx,msg1
mov ah,9
int 21h

mov ah,2
mov dl,0dh
int 21h
mov dl,0ah
int 21h
lea dx,msg1
mov ah,9
int 21h

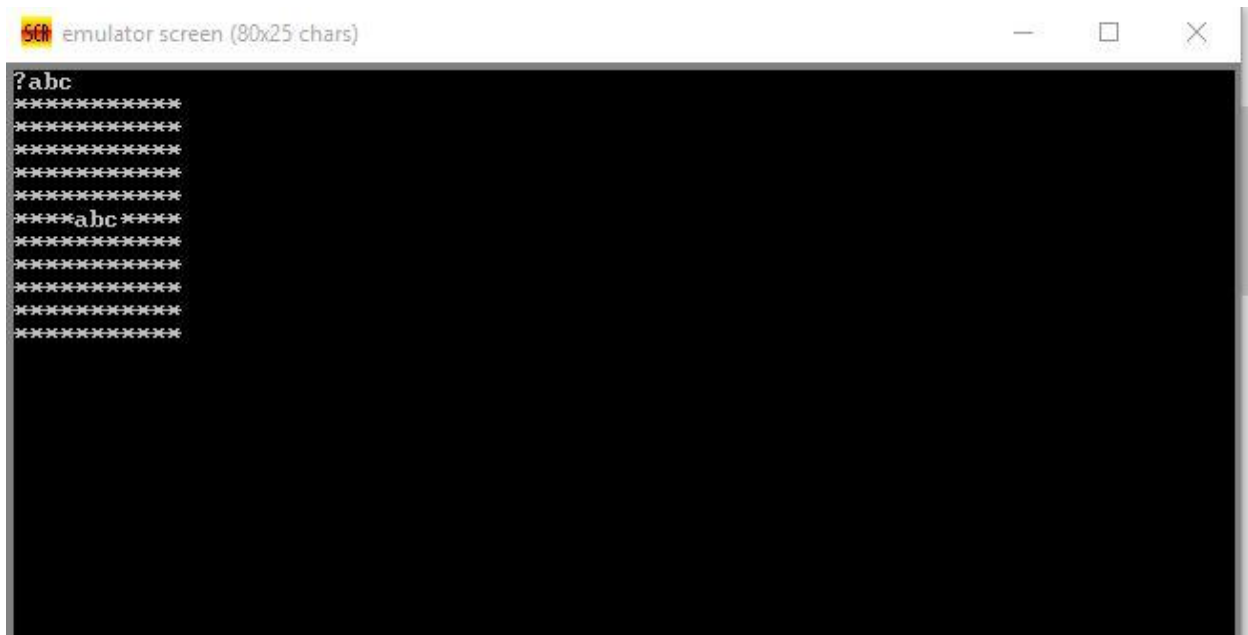
```
    mov ah,2
    mov dl,0dh
    int 21h
    mov dl,0ah
    int 21h
    lea dx,msg1
    mov ah,9
    int 21h
```

```
mov ah,2
mov dl,07h
int 21h
```

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

```
main endp
end main
```

Output:



Chapter 6

8. Write a program to display a "?", read two capital letters, and display them on the next line In alphabetical order.

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

```
msg db 10,13,'? '$  
msg1 db 10,13,'Enter two capital letter=$'  
msg2 db 10,13,'Output is=$'
```

```
.code
```

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

```
    lea dx,msg
```

```
    mov ah,9  
    int 21h
```

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

```
    mov ah,1  
    int 21h  
    mov bl,al
```

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



```
lea dx,msg2
mov ah,9
int 21h
```

```
cmp bl,cl
ja go      ; if bl>cl
```

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

```
mov dl,cl
int 21h
```

```
jmp exit:
```

```
go:
```

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

```
mov dl,bl
int 21h
```

```
jmp exit:
```

```
exit:
```

Output:



9. Write a program to display the extended ASCII characters (ASCII codes 80h to FF_h). Display 10 characters per line, separated by blanks. Stop after the extended characters have been displayed once.

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

.code

main proc

    mov cx,127 ;initialize number of character
    mov bl,0

print:

    mov ah,2
    inc cx
    cmp cx,255
    ja exit
    mov dx,cx
    int 21h
    mov dx,32d ; giving space
    int 21h
```

```
jmp go
```

go:

```
inc bl
cmp bl,10 ; 10 char per line
je nl
jmp print
```

nl:

```
mov ah,2
mov dl,0dh
int 21h
mov dl,0ah
int 21h
mov bl,0
jmp print
```

exit:

Output



10. Write a program that will prompt the user to enter a hex digit character ("0" ... "9" or "A" ... "F"), display it on the next line

in decimal, and ask the user if he or she wants to do it again. If the user types "y" or "Y", the program repeats; If the user types anything else, the program terminates. If the user enters an illegal character, prompt the user to try again.

```
.model small
.stack 100h
.data
msg1 db 10,13,'ENTER A HEX DIGIT:$'
msg2 db 10,13,'IN DECIMAL IS IT:$'
msg3 db 10,13,'DO YOU WANT TO DO IT AGAIN?$(
msg4 db 10,13,'ILLEGAL CHARACTER- ENTER 0-9 OR A-F:$'
```

```
.code
```

```
again:
```

```
    mov ax,@data
    mov ds,ax
    lea dx,msg1
    mov ah,9
    int 21h
```

```
    mov ah,1
    int 21h
```

```
    mov bl,al
```

```
    jmp go
```

```
go:
```

```
    cmp bl,'9'
    ja hex    ;if bl>9 go to hex label
```

```
jb num  
je num
```

hex:

```
cmp bl,'F'  
ja illegal ;if bl>F illegal
```

```
lea dx,msg2  
mov ah,9  
int 21h
```

```
mov dl,49d  
mov ah,2  
int 21h
```

```
sub bl,17d ; convert to letter  
mov dl,bl  
mov ah,2  
int 21h
```

```
jmp inp
```

inp:

```
lea dx,msg3  
mov ah,9  
int 21h
```

```
mov ah,1  
int 21h
```

```
mov cl,al  
cmp cl,'y'  
je again  
cmp cl,'Y'
```

```
je again  
jmp exit
```

num:

```
cmp bl,'0'  
jb illegal
```

```
lea dx,msg2  
mov ah,9  
int 21h
```

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

```
jmp inp
```

illegal:

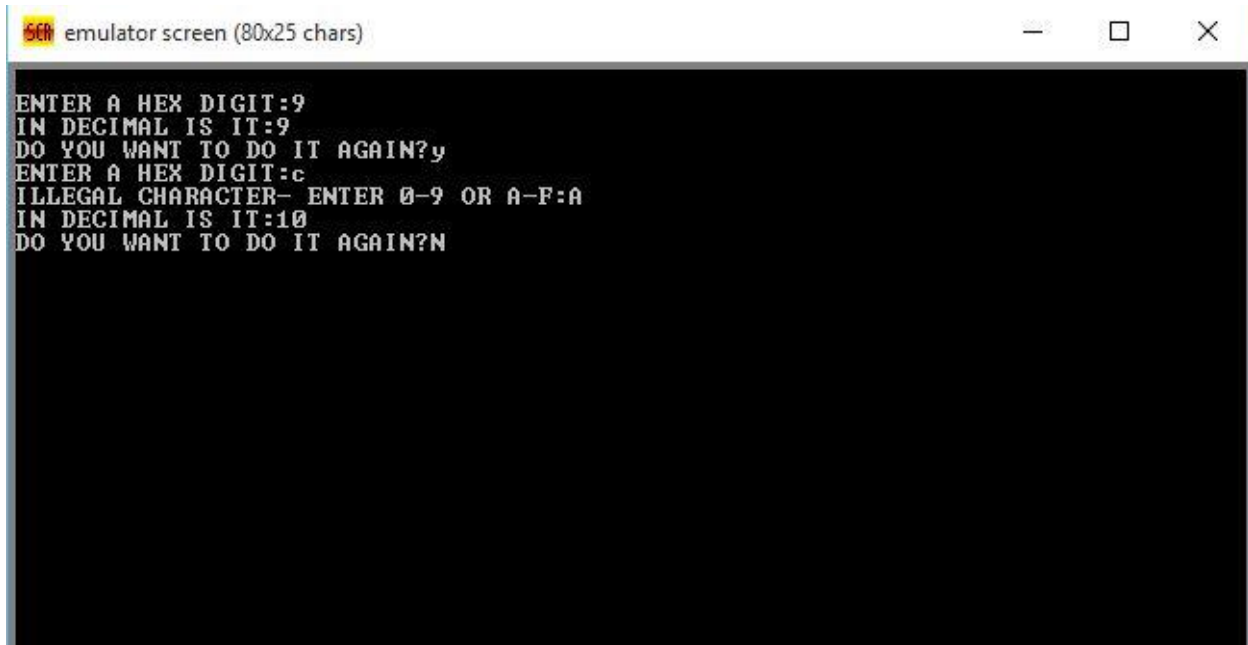
```
lea dx,msg4  
mov ah,9  
int 21h
```

```
mov ah,1  
int 21h
```

```
mov bl,al
```

```
jmp go
```

exit:

A screenshot of a Windows-style window titled "emulator screen (80x25 chars)". The window contains a black terminal area with white text. The text shows a program that prompts for a hex digit, converts it to decimal, and asks if the user wants to try again. The first attempt uses '9' and succeeds. The second attempt uses 'c' and fails with an "ILLEGAL CHARACTER" message. The third attempt uses '10' and succeeds.

```
ENTER A HEX DIGIT:9
IN DECIMAL IS IT:9
DO YOU WANT TO DO IT AGAIN?y
ENTER A HEX DIGIT:c
ILLEGAL CHARACTER- ENTER 0-9 OR A-F:A
IN DECIMAL IS IT:10
DO YOU WANT TO DO IT AGAIN?N
```

11. Do programming exercise 10, except that if the user fails to enter a hex-digit character In three tries, display a message and terminate the program.

```
.model small
.stack 100h
.data
msg1 db 10,13,'ENTER A HEX DIGIT:$'
msg2 db 10,13,'IN DECIMAL IS IT:$'
msg3 db 10,13,'DO YOU WANT TO DO IT AGAIN?$'
msg4 db 10,13,'ILLEGAL CHARACTER- ENTER 0-9 OR A-F:$'
msg5 db 10,13,'Too Many Try$'
```

```
.code
```

```
again:
```

```
    mov cx,0
    mov ax,@data
    mov ds,ax
    lea dx,msg1
    mov ah,9
```

int 21h

mov ah,1
int 21h

mov bl,al

jmp go

go:

cmp bl,'9'
ja hex
jb num
je num

hex:

cmp bl,'F'
ja illegal

lea dx,msg2
mov ah,9
int 21h

mov dl,49d
mov ah,2
int 21h

sub bl,17d
mov dl,bl
mov ah,2
int 21h

jmp inp

inp:


```
lea dx,msg3
mov ah,9
int 21h
```

```
mov ah,1
int 21h
```

```
mov cl,al
cmp cl,'y'
je again
cmp cl,'Y'
je again
jmp exit
```

num:

```
cmp bl,'0'
jb illegal
```

```
lea dx,msg2
mov ah,9
int 21h
```

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

```
jmp inp
```

illegal:

```
inc cx
cmp cx,3
je i2
```

```
lea dx,msg4
mov ah,9
int 21h
```

```
mov ah,1
```

```
int 21h
```

```
mov bl,al
```

```
jmp go
```

```
i2:
```

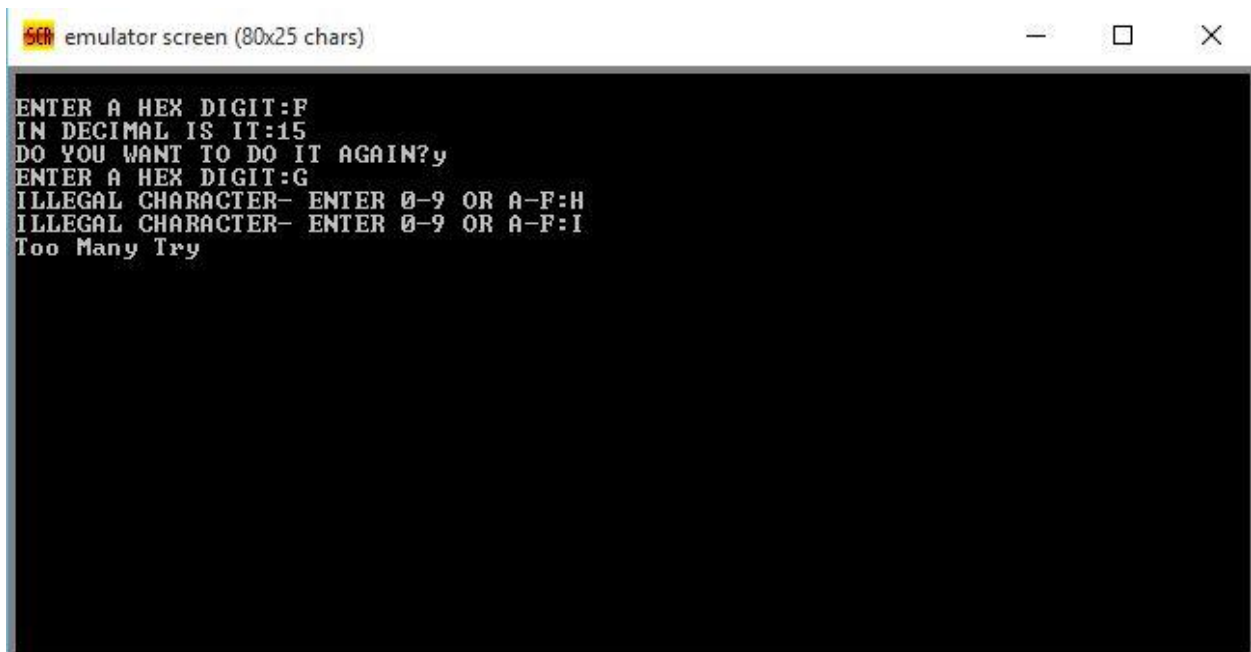
```
lea dx,msg5
```

```
mov ah,9
```

```
int 21h
```

```
jmp exit
```

```
exit:
```



The screenshot shows a window titled "emulator screen (80x25 chars)" with a black background and white text. The text displays the output of an assembly program. It starts with "ENTER A HEX DIGIT:F", followed by "IN DECIMAL IS IT:15", and "DO YOU WANT TO DO IT AGAIN?y". Then it prompts "ENTER A HEX DIGIT:G", which is followed by two lines of error messages: "ILLEGAL CHARACTER- ENTER 0-9 OR a-f:h" and "ILLEGAL CHARACTER- ENTER 0-9 OR a-f:i". The final line of output is "Too Many Try".

```
ENTER A HEX DIGIT:F
IN DECIMAL IS IT:15
DO YOU WANT TO DO IT AGAIN?y
ENTER A HEX DIGIT:G
ILLEGAL CHARACTER- ENTER 0-9 OR a-f:h
ILLEGAL CHARACTER- ENTER 0-9 OR a-f:i
Too Many Try
```

Chapter 7

8. Write a program that prompts the user to enter a character, and on subsequent lines prints its ASCII code in binary, and the number of 1 bits in its ASCII code.

Sample execution:

TYPE A CHARACTER: A

THE ASCII CODE OF A IN BINARY IS 01000001

THE NUMBER OF 1 BITS IS 2

```
.model small
```

```
.stack 100h
```

```
.data
```

```
msg1 DB 'TYPE A CHARACTER:$'
```

```
msg2 DB 0DH,0AH,'THE ASCII CODE OF $'
```

```
msg3 DB ' IN BINARY IS $'
```

```
msg4 DB 0DH,0AH,'THE NUMBER OF 1 BITS IS $'
```

```
.code
```

```
main proc
```

```
mov ax,@data
```

```
mov ds,ax
```

```
lea dx,msg1
```

```
mov ah,9
```

```
int 21h
```

```
mov ah,1
```

```
int 21h
```

```
xor bx,bx
```

```
mov bl,al
```

```
lea dx,msg2
```

```
mov ah,9
```

```
int 21h
```

```
mov dl,bl
```

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

```
lea dx,msg3  
mov ah,9  
int 21h
```

```
mov cx,8 ; limit for loop i<=8 for 8 bit  
mov bh,0
```

binary:

```
shl bl,1  
jnc zero; CF=0  
inc bh  
mov dl,31h  
jmp display
```

zero:

```
mov dl,30h
```

display:

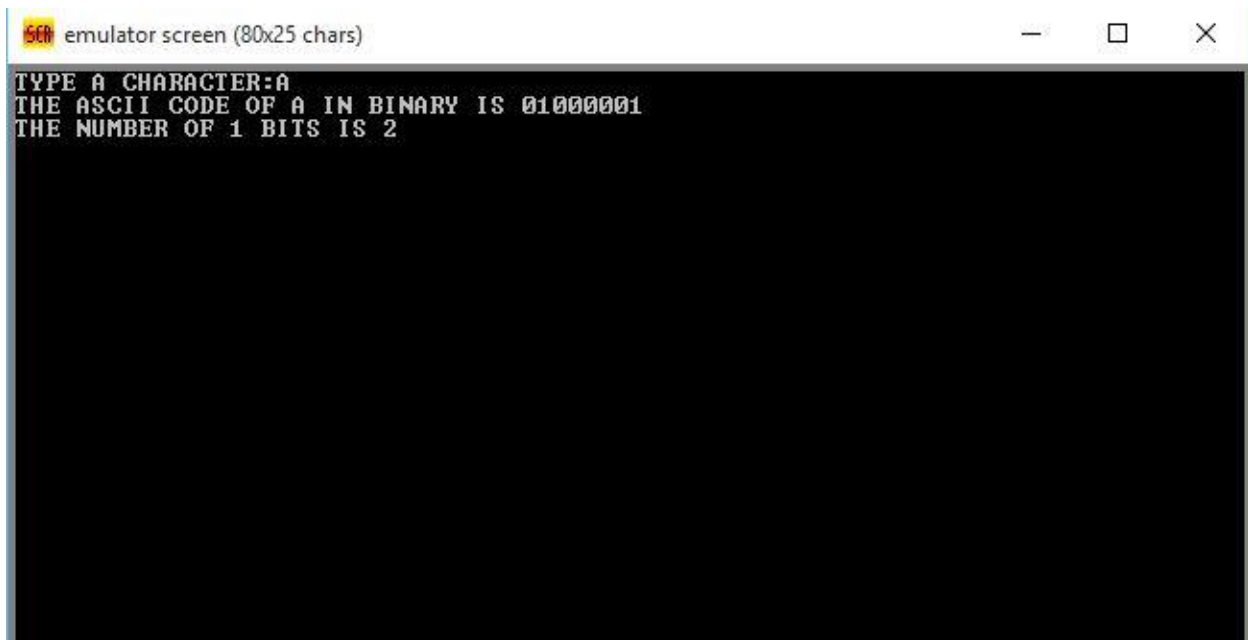
```
mov ah,2  
int 21h
```

loop binary ;loop will be terminated while cx>8

```
ADD bh,30h
```

```
lea dx,msg4  
mov ah,9  
int 21h
```

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



```
emulator screen (80x25 chars)
TYPE A CHARACTER:A
THE ASCII CODE OF A IN BINARY IS 01000001
THE NUMBER OF 1 BITS IS 2
```

9. Write a program that prompts the user to enter a character and prints the ASCII code of the character in hex on the next line. Repeat this process until the user types a carriage return.

Sample execution:

**TYPE A CHARACTER: Z
THE ASCII CODE OF Z IN HEX IS 5A
TYPE A CHARACTER:**

```
.model small
```

```
.stack 100h
```

```
.data
```

```
msg1 db 10,13,'Type a character:$'  
msg2 db 10,13,'The Ascii code of $'  
msg3 db ' in hex is:$'
```

```
.code
```

```
main proc
```

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

input:

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

```
mov ah,1
int 21h
```

```
mov bl,al
```

```
cmp bl,0dh
je end
```

```
lea dx,msg2
mov ah,9
int 21h
```

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

```
lea dx,msg3
mov ah,9
int 21h
```

```
mov cx,4
```

convert:

```
mov dl,bh
shr dl,1    ;shift left 4 times
shr dl,1
shr dl,1
shr dl,1
```

```
cmp dl,9
jbe num
```

```
add dl,55d
```

jmp display

num:

 add dl,30h

display:

 mov ah,2

 int 21h

 rcl bx,1 ;rotate carry left 4 times

 rcl bx,1

 rcl bx,1

 rcl bx,1

loop convert

jmp input

end:

 MOV AH,4CH

 INT 21H

MAIN ENDP

END MAIN

Output:



10. Write a program that prompts the user to type a hex number of four hex digits or less, and outputs it in binary on the next line. If the user enters an illegal character, he or she should be prompted to begin again. Accept only uppercase letters.

Sample execution:

**TYPE A HEX NUMBER (0 TO FFFF): la
ILLEGAL HEX DIGIT, TRY AGAIN: IABC
IN BINARY IT IS 0001101010111100**

```
.model small
```

```
.stack 100h
```

```
.data
```

```
msg1 db 10,13,' Type a hex number (0 to FFFF):$'  
msg2 db 10,13,'Illegal hex digit,try again:$'  
msg3 db 10,13,'In Binary it is $'
```

```
.code
```

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


jmp input

invalid:

```
lea dx, msg2
mov ah, 9
int 21h
```

input:

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

```
xor bx, bx
mov cl, 4
```

```
mov ah, 1
int 21h
```

convert:

```
cmp al, 0dh
je end_input
```

```
cmp al, '0'
jb invalid
```

```
cmp al, 'F'
ja invalid
```

```
cmp al, 39h
ja letter
```

```
and al, 0fh
jmp left
```

letter:

```
sub al,55d ;convert char to binary
```

left:

```
shl bx,cl  
or bl,al
```

```
mov ah,1  
int 21h  
jmp convert
```

end_input:

```
lea dx,msg3  
mov ah,9  
int 21h
```

```
xor dx,dx  
mov cx,16
```

print_binary:

```
shl bx,1 ;catch bx bit
```

```
jc one ;cf=1
```

```
mov dl,30h  
jmp display
```

one:

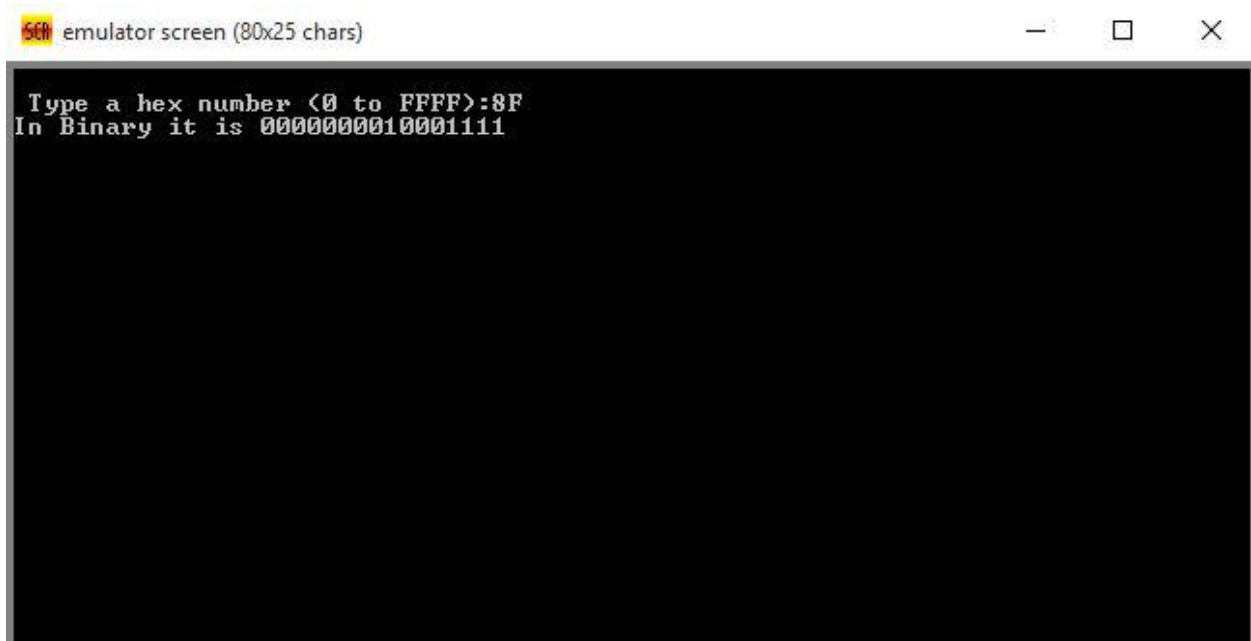
```
mov dl,31h
```

display:

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

loop print_binary

```
main endp
end main
```



11. Write a program that prompts the user to type a binary number of 16 digits or less, and outputs It In hex on the next line. If the user enters an illegal character, he or she should be prompted to begin again .

... ..

**TYPE A BINARY NUMBER, UP TO 16 DIGITS: 11100001
IN HEX IT IS E1**

```
.model small
```

```
.stack 100h
```

```
.data
```

```
msg1 db 'Type a binary number upto 16 digits:$'
```

```
msg2 db 10,13,'in hex it is:$'
```

```
.code
```

```
main proc
```

```
mov ax,@data
mov ds,ax
lea dx,msg1
mov ah,9
int 21h
```

```
xor bx,bx
mov ah,1
int 21h
```

input:

```
cmp al,0dh
je exit
```

```
and al,0fh
shl bx,1
or bl,al
```

```
int 21h
```

```
jmp input
```

exit:

```
lea dx,msg2
mov ah,9
int 21h
```

```
mov cx,4
```

convert:

```
mov dl,bh
shr dl,1
shr dl,1
shr dl,1
shr dl,1
```

```
cmp dl,9
```

```
jbe num
add dl,55d
jmp display
```

num:

```
add dl,30h
```

display:

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

```
rcl bx,1
rcl bx,1
rcl bx,1
rcl bx,1
```

loop convert

```
main endp
end main
```

56H emulator screen (80x25 chars)

```
Type a binary number upto 16 digits:10101111
in hex it is:00AF
```

12. Write a program that prompts the user to enter two binary numbers of up to 8 digits each, and prints their sum on the next line in binary. If the user enters an illegal character, he or she should be prompted to begin again. Each input ends with a carriage return.

**TYPE 'A BINARY NUMBER, UP TO 8 DIGITS: 11001010
TYPE 'A BINARY NUMBER, UP TO 8 DIGITS: 10011100
THE BINARY SUM IS 101100110**

```
.model small
```

```
.stack 100h
```

```
.data
```

```
msg1 db 10,13,'Type a binary number upto 8 digits:$'  
msg2 db 10,13,'The binary sum is:$'
```

```
.code
```

```
main proc
```

```
    mov ax,@data  
    mov ds,ax  
    lea dx,msg1  
    mov ah,9  
    int 21h
```

```
    mov ah,1  
    int 21h
```

```
    xor bx,bx  
    mov cx,8
```

```
input1:
```

```
    mov ah,1  
    int 21h
```

```
cmp al,0dh
je break
```

```
and al,0fh ;convert to binary
shl bl,1 ;make room for new value
```

```
or bl,al ;insert value
```

```
loop input1
```

```
break:
```

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

```
mov cx,8
```

```
input2:
```

```
mov ah,1
int 21h
```

```
cmp al,0dh
je break2
```

```
and al,0fh ;convert to binary
shl bh,1 ;make room for new value
```

```
or bh,al ;insert value
```

```
loop input2
```

```
break2:
```

```
lea dx,msg2
mov ah,9
int 21h
```

sum:

```
add bl,bh ;sum binary 00000011+00000010, bl=000000101
jnc zero ;if sum has no carry then no need to print zero
mov dl,31h
mov ah,2
int 21h ;if sum has carry 1 then need to print 1
```

zero:

```
mov dl,30h
```

```
mov cx,8
```

print:

```
shl bl,1 ;sending one by one bit to print 000000101
jnc z
mov dl,31h
jmp display
```

z:

```
mov dl,30h
```

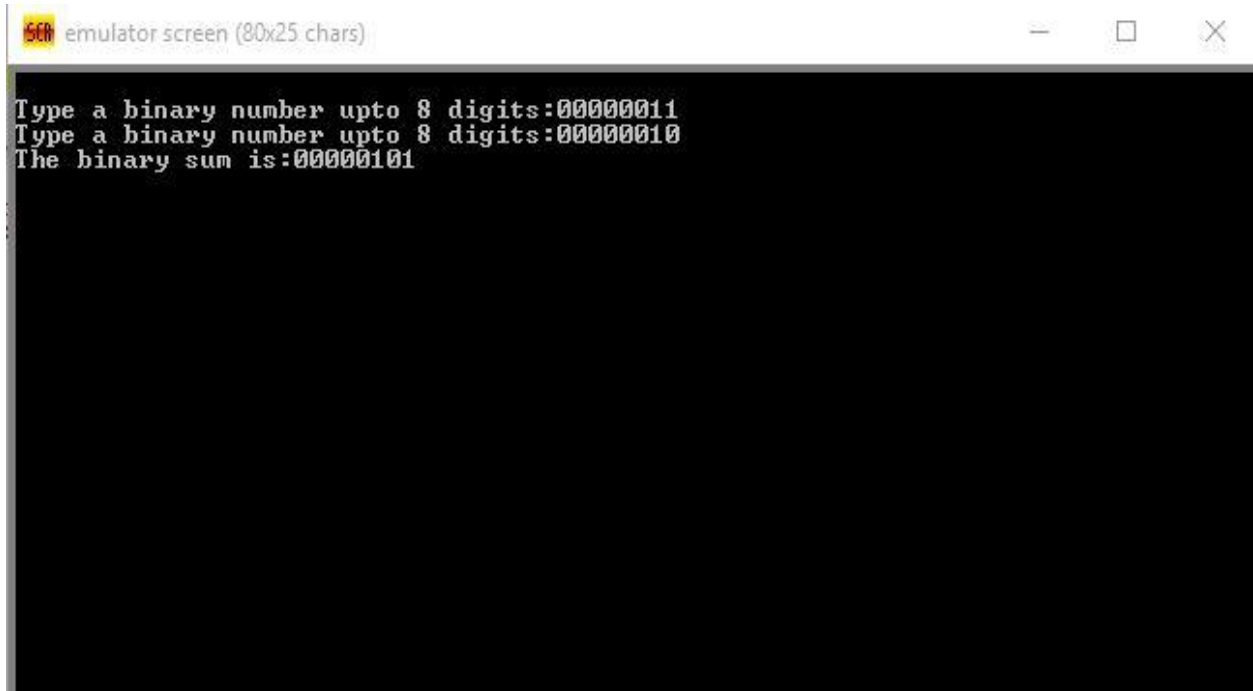
display:

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

```
loop print
```

```
main endp
end main
```

Output



```
emulator screen (80x25 chars)
Type a binary number upto 8 digits:00000011
Type a binary number upto 8 digits:00000010
The binary sum is:00000101
```

Chapter 8

8. Write a program that lets the user type some text, consisting of words separated by blanks, ending with a carriage return, and displays the text in the same word order as entered, but with the letters

ters in each word reversed. For example, "this is a test" becomes "siht si a tset".

.model small

.stack 100h

.data

.code

msg1 db 'Enter String:\$'

msg2 db 'Reverse String:\$'

count DW 0

main proc

mov ax,@data ;initialize data segment

mov ds,ax

lea dx,msg1

mov ah,9

int 21h

xor cx,cx

mov ah,1

int 21h

input:

cmp al,0dh

je end_input

push ax

inc cx

int 21h

jmp input

end_input:

mov dx,0dh ;carriage and new line

mov ah,2

int 21h

mov dx,0ah

mov ah,2

int 21h

mov bx,50h

xchg bx,sp ;swap sp pointer

push 0020h

xchg bx,sp

inc count

loop1:

```
pop dx          ;swap bx and sp
xchg bx,sp
push dx         ; push to new stack
xchg bx,sp
inc count
```

loop loop1

```
lea dx, msg2
mov ah,9
int 21h
```

```
mov cx,count
mov count,0
push 0020h      ; push 0020H onto the STACK
inc count
```

reverse:

```
xchg bx, sp     ; swap bx and sp
pop dx          ; pop a value from stack into dx
xchg bx, sp     ; swap bx and sp
cmp dl, 20h
jne skip
```

```
mov ah, 2
```

```
loop2:
pop dx          ; pop and show output
int 21h
```

```
dec count
jnz loop2
```

```
mov dx, 0020h
```

skip:

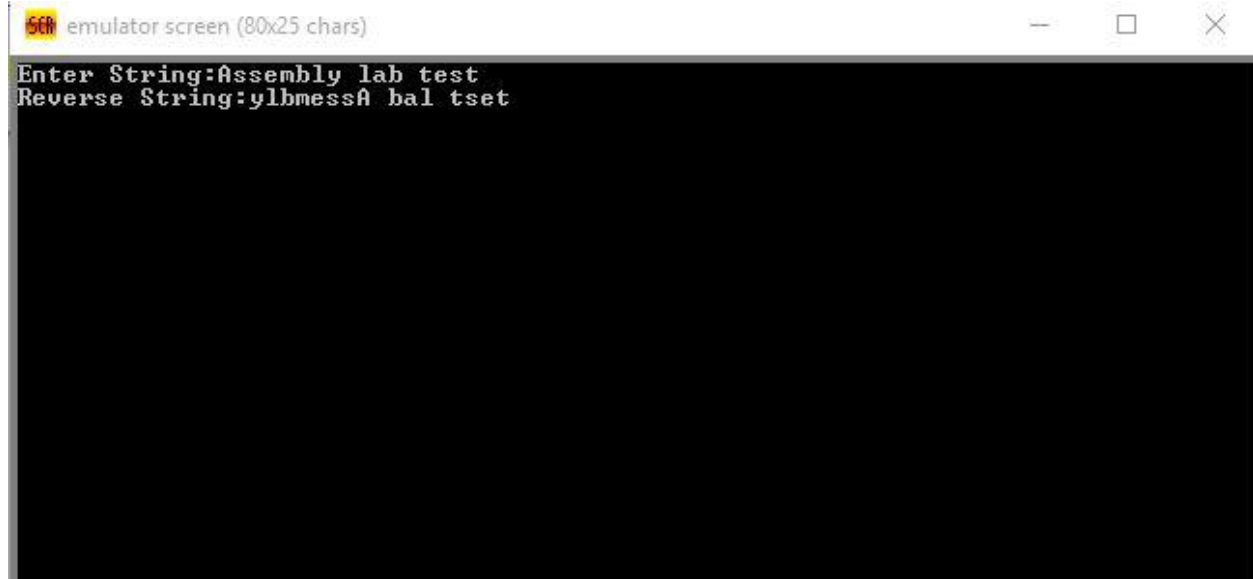
```
push dx
inc count
```

```

loop reverse

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

```



emulator screen (80x25 chars)

```

Enter String:Assembly lab test
Reverse String:ylbmessA bal tset

```

10. The following method can be used to generate random numbers in the range 1 to 32767.

Start with any number in this range.

Shift left once.

Replace bit 0 by the XOR of bits 14 and 15.

Clear bit 15.

Write the following procedures:

a. A procedure READ that lets the user enter a binary number and stores it in AX. You may use the code for binary input given in section 7.4.

b. A procedure RANDOM that receives a number in AX and returns a random number in AX.

c. A procedure WRITE that displays AX in binary. You may use the algorithm given in section 7.4.

Write a program that displays a '?', calls READ to read a binary number, calls RANDOM and WRITE to compute and display 100 random numbers. The numbers should be displayed four per line, with four blanks separating the numbers.

.model small

```
.stack 100h
```

```
.data
```

```
msg1 db 'Enter a binary number:$'
```

```
msg2 db 10,13,'Random numbers are:$'
```

```
msg3 db 10,13,'illegal entry: try again:$'
```

```
va dw ?
```

```
vb dw ?
```

```
vc dw ?
```

```
.code
```

```
main proc
```

```
mov ax,@data
```

```
mov ds,ax
```

```
mov vb,1 ; mov 1 to word vb
```

```
call rproc ; calling new procedure
```

```
mov ah,9
```

```
lea dx,msg2
```

```
int 21h
```

```
star:
```

```
mov vc,1
```

```
cmp vb,25
```

```
jg las
```

```
inc vb
```

```
mov ah,2
```

```
mov dl,0ah
```

```
int 21h
```

```
start:
```

```
call random ; calling random proc
```

```
call write
```

```
mov ah,2
```

```
mov dl,' '
```

```
int 21h
```

```
inc vc
```

```
cmp vc,4  
jle start  
jmp star
```

```
las:  
mov ah,1  
int 21h  
mov ah,4ch  
int 21h  
main endp
```

```
rproc proc
```

```
mov ah,9  
lea dx,msg1  
int 21h  
soo:  
mov cl,16  
xor bx,bx
```

```
top:  
mov ah,1  
int 21h
```

```
cmp al,0dh  
je last  
cmp al,'0'  
jne as  
jmp asd
```

```
as:  
cmp al,'1'  
jne error
```

```
asd:  
and al,0fh  
shl bx,1  
or bl,al  
loop top  
jmp last
```

```
error:  
mov ah,9  
lea dx,msg3
```

int 21h

jmp soo

last:

ret

rproc endp

random proc

shl bx,1

xor dx,dx

mov a,bx

mov dx,bx

shl va,1

xor dx,a

rol dx,1

jc setbx

and bx,0fffeh

jmp go

setbx:

or bx,0001h

go:

ret

random endp

write proc

mov cl,16

top1:

rol bx,1

jc one

mov ah,2

mov dl,'0'

```

int 21h
jmp en
one:

mov ah,2
mov dl,'1'
int 21h

en:
loop top1

ret          ; returning to procedure

write endp

end main

```

Output:

The screenshot shows a window titled "emulator screen (80x25 chars)" containing a 25x80 grid of binary digits. The digits are arranged in a pattern that appears to be a stylized representation of the word "11111111" repeated multiple times, with some variations in the spacing and alignment of the digits. The grid is filled with 0s and 1s, creating a complex, repeating pattern.

Chapter 9

8. Modify procedure INDEC so that it will check for overflow.

Modified Decimal Input Code (INDEC) For Overflow :

```
indec proc
    push bx
    push cx
    push dx

start:
    mov ah,2
    mov dl,0dh
    int 21h
    mov dl,0ah
    int 21h

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

    xor bx,bx ; bx is for total=0
    xor cx,cx ; cx is for detecting sign
    mov ah,1
    int 21h
    cmp al,'-' ; if minus sign set it
    je min
    cmp al,'+' ; if plus sign set it
    je pl
    jmp convert ; else convert to decimal
min:
    mov cx,1 ; negative is true

pl:
    int 21h
convert:
    cmp al,'0'
```

```
jl illegal
cmp al,'9'
jg illegal
```

```
and ax,000Fh ; convert ascii char to decimal
push ax      ; save on stack
```

```
mov ax,10    ; set value 10 for multiply
imul bx      ; multiply by 10, ax=total*10
```

```
jo of1       ; jump if overflow
```

```
pop bx       ; bx= converted decimal digit
add bx,ax    ; total=total*10+ decimal digit
jo of2       ; jump if overflow
mov ah,1
int 21h
cmp al,0dh
je cr
jmp convert
cr:
```

```
mov ax,bx    ; move decimal number to ax
or cx,cx     ; negative sign
je exit      ; if not negative
neg ax
```

```
exit:
```

```
pop dx
pop cx       ; restore registers
pop bx
```

```
ret
```

```
of1:
```

```
pop bx
mov ah,9
lea dx,msg1
int 21h
jmp start
```

```
of2:
    mov ah,9
    lea dx,msg1
    int 21h
    jmp start
illegal:
    mov ah,2
    mov dl,0ah
    int 21h
    jmp start
indec endp
```

Decimal Output Code (OutDEC):

```
outdec proc
```

```
    push ax
    push bx
    push cx
    push dx
```

```
    or ax,ax
    jge valid ; if ax>=0
```

```
    push ax
    mov dl,'-'
    mov ah,2
    int 21h
```

```
    pop ax
    neg ax
```

```
valid:
```

```
    xor cx,cx
    mov bx,10d ; bx is divisor
```

divide:

```
xor dx,dx ; dx is remainder
```

```
div bx ; divide ax by bx, ax =quotient, dx=remainder
```

```
push dx ; push remainder
```

```
inc cx
```

```
or ax,ax ; untill ax is not zero
```

```
jne divide
```

```
mov ah,2
```

```
print:
```

```
pop dx
```

```
or dl,30h ;print digits
```

```
int 21h
```

```
loop print
```

```
pop dx
```

```
pop cx
```

```
pop bx
```

```
pop ax
```

```
ret
```

```
outdec endp
```

Input And Output Decimal:

```
.model small
```

```
.stack 100h
```

```
.data
```

```
msg1 db 10,13,'Invalid!! Overflow',0AH,'$'
```

```
.code
```

```
main proc
```

```
mov ax,@data
```

```
mov ds,ax
```

```
call indec
```

```
push ax
```

```
mov ah,2
```

```
mov dl,0dh
```

```
int 21h
```

```

mov dl,0ah
int 21h
pop ax
call outdec
mov ah,4ch
int 21h
main endp
include outdec.asm
include indec.asm
end main

```

Output:



9. Write a program that lets the user enter time in seconds, up to 65535, and outputs the time as hours, minutes, and seconds. Use INDEC and OUDEC to do the I/O.

```

.model small
.stack 100h
.data
msg1 db 'Enter time in seconds:$'
msg2 db 10,13,'Time in Hour Miniute And Second Is:$'

.code
main proc
    mov ax,@data
    mov ds,ax
    lea dx,msg1
    mov ah,9
    int 21h

    call indec
    push ax

```

```
xor cx,cx
mov cl,3
pop ax
```

l1:

```
xor dx,dx    ; dx is remainder
mov bx,60d   ; bx is divisor
div bx       ; ax =quotient
push dx      ; push remainder
loop l1
```

```
mov ah,9
lea dx,msg2
int 21h
```

```
mov cl,3
```

time:

```
pop ax       ; get remainder
call outdec
mov ah,2
mov dl,':'
int 21h
loop time
```

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

```
include outdec.asm
include indec.asm
end main
```

Output:

emulator screen (80x25 chars)

```
Enter time in seconds:
?5325
Time in Hour Miniute And Second Is:1:28:45:
```

12. Write a program to find the greatest common divisor (GCD) of two integers M and N , according to the following algorithm:

- 1. Divide M by N , getting quotient O and remainder R .**
- 2. If $R = 0$, stop. N is the GCD of M and N .**
- 3. If $R \neq 0$, replace M by N , N by R , and repeat step 1.**

Use INDEC to enter M and N and OUTDEC to print the GCD.

```
.model small
.stack 100h
.data
msg1 db 'Enter M=$'
msg2 db 10,13,'Enter N=$'
msg3 db 10,13,'GCD is=$'

a dw ?
b dw ?

.code
main proc
    mov ax,@data
    mov ds,ax
    lea dx,msg1
    mov ah,9
    int 21h
```

```
call indec    ;input
push ax
lea dx,msg2
mov ah,9
int 21h
```

```
call indec    ;input
push ax
xor bx,bx
pop bx
pop ax
```

```
cmp ax,bx     ; if ax is small then swap
jl swap
jmp gcd
```

```
swap:
  mov a,ax     ;swap ax and bx
  mov ax,bx
  mov bx,a
```

```
gcd:
  xor dx,dx    ;remainder dx
  mov b,bx
  div b
```

```
cmp dx,0      ; if remainder is zero print gcd
je go
mov ax,bx     ; replace dividend by divisor
mov bx,dx     ; replace divisor by remainder
jmp gcd
```

```
go:
  lea dx,msg3  ;print gcd
  mov ah,9
  int 21h
  mov ax,b
  call outdec
```

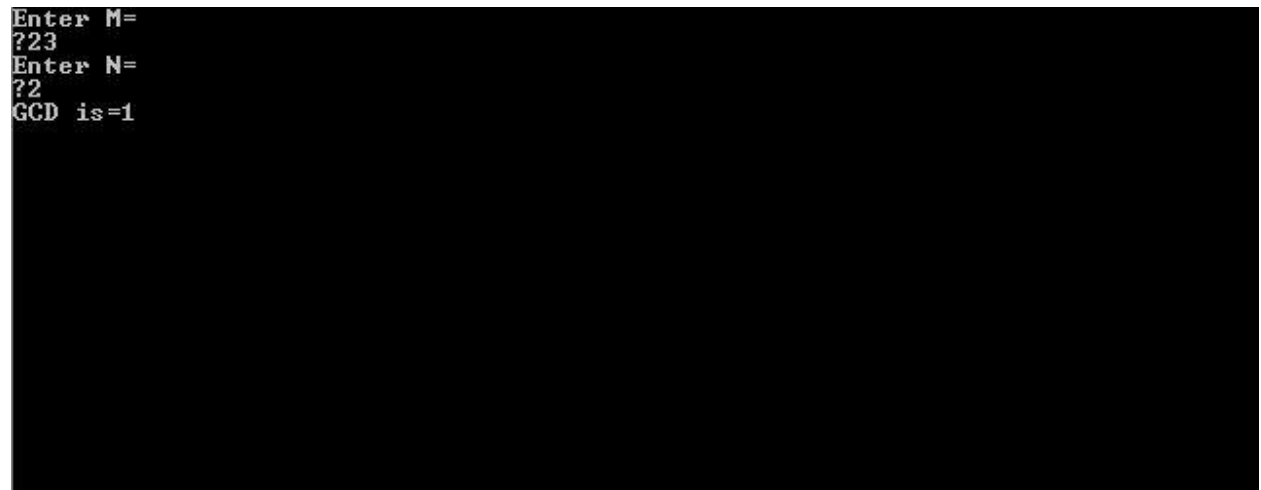
```
mov ah,4ch
int 21h
```

```
main endp
```



```
include outdec.asm  
include indec.asm  
  
end main
```

Output:

A terminal window with a black background and white text. The text shows the program's execution flow: it prompts for 'M' and receives '23', then prompts for 'N' and receives '2', and finally outputs 'GCD is=1'.

```
Enter M=  
?23  
Enter N=  
?2  
GCD is=1
```

For More Programming Tutorials
Visit

<http://shawonruet.blogspot.com>