Program to add two numbers

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

main proc

mov bl,2
mov dl,1
add dl,bl
add dl,48
mov ah,2
INT 21h


mov ah,4ch
INT 21h

main endp

end main
```

Program to print A to Z in small and capital letters using loop

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

main proc

mov cx, 26

mov ah, 2

mov dl, 65

L1:

int 21h

inc dl
```

```
loop L1
mov dl, 10
mov ah,2
INT 21h

mov dl, 13
mov ah,2
INT 21h

mov cx, 26
mov dl, 97
mov ah, 2
L2:

int 21h

inc dl

loop L2

mov ah,4ch
INT 21h

main endp

end main
```

Program to print helloworld string on screen

```
.model small
.stack 100h
.data
S1 db "Helloworld$"

.code

main proc
mov ax,@data
mov ds,ax

lea dx,s1
mov ah,9
INT 21h
```

```
mov ah,4ch
INT 21h


main endp

end main
```

Program to print two different strings in two different lines

```
.MODEL SMALL
.STACK 100H
.DATA
STRING_1  DB  'Ali$'
STRING_2  DB  'Hassan Soomro $'
.CODE
MAIN PROC
MOV AX, @DATA
MOV DS, AX

LEA DX, STRING_1
MOV AH, 9
INT 21H

MOV AH, 2
MOV DL, 10
INT 21H

MOV DL, 13
INT 21H
LEA DX, STRING_2
MOV AH, 9
INT 21H
MOV AH, 4CH
INT 21H
MAIN ENDP
END MAIN
```

Program to input a capital letter from user and convert it into small letter (uppercase to lowercase)

```
.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
```

Program to input a small letter from user and convert it into capital letter (lowercase to uppercase)

```
.model small
.stack 100h
.data
.code
main proc
mov ah, 1
int 21h
mov dl, al
sub dl,32
mov ah, 2
int 21h

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

Program to input string from user and print it

```
.model small
.stack 100h
.data
var1 db 100 dup("$")
.code
main proc
mov ax,@data
mov ds,ax
mov si,offset var1
l1: mov ah,1
int 21h
cmp al,13
je printString
mov [si],al
inc si
jmp l1

printString:
mov dx, offset var1
mov ah,9
int 21h


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

Program to input String from user and print its length

```
.model small
.stack 100h
.data
var1 db 100 dup("$")
.code
main proc
mov ax,@data
mov ds,ax
mov bl, 0              ; counts the length of string
mov si,offset var1
l1: mov ah,1
int 21h
```

```
cmp al,13
je printString
mov [si],al
inc si
inc bl
jmp l1

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


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

Get an integer from user and display whether the number is 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 dx,offset od
mov ah,9
int 21h

mov ah,4ch
int 21h


IsEven:
mov dx,10
mov ah,2
int 21h

mov dx,13
mov ah,2
int 21h
mov dx,offset ev
mov ah,9
int 21h

mov ah,4ch
int 21h

main endp
end main
```

Display the sum of all odd numbers between 1 and 100

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

.code

main proc

        mov ax,@data
        mov ds,ax

        mov cx,1
        mov ax,0
        l1:
```

```
                add ax,cx
                add cl,2
                cmp cl,100
        jl l1


        mov dx,0
        mov bx,10
        mov cx,0
        l2:
                div bx
                push dx
                mov dx,0

                mov ah,0
                inc cx
                cmp ax,0
        jne l2

        mov ah,02h
        l3:
                pop dx
                add dx,48
                int 21h
        loop l3




        mov ah,4ch
        int 21h


main endp
end main
```

Display the sum of all even numbers between 1 and 100

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

.code

main proc
```

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

        mov cx,0
        mov ax,0
        l1:
                add ax,cx
                add cl,2
                cmp cl,100
        jle l1


        mov dx,0
        mov bx,10
        mov cx,0
        l2:
                div bx
                push dx
                mov dx,0

                mov ah,0
                inc cx
                cmp ax,0
        jne l2

        mov ah,02h
        l3:
                pop dx
                add dx,48
                int 21h
        loop l3



        mov ah,4ch
        int 21h


main endp
end main
```

Display the largest number in an array

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

```
.data
STRING1 DB 1,2,7,5
res db ?
.code
main proc
mov ax,@data
mov ds,ax

mov cx, 4

mov bl, 0
LEA SI, STRING1
up:
mov al, [SI]
cmp al, bl
jl nxt
mov bl, al
nxt:
inc si
dec cx
jnz up

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



mov ah,4ch
int 21h

main endp
end main
```

Display the smallest number in an array

```
.model small
.stack 100h
.data
STRING1 DB 2,1,7,5
res db ?
.code
main proc
mov ax,@data
```

```
mov ds,ax

mov cx, 4

mov bl, 79h
LEA SI, STRING1
up:
mov al, [SI]
cmp al, bl
jge nxt
mov bl, al
nxt:
inc si
dec cx
jnz up

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



mov ah,4ch
int 21h

main endp
end main
```

Display the binary number of given decimal number

```
; display the binary of given decimal number
.model small
.stack 100h
.data
msg   db  'Enter a decimal number:$'
msg1  db   0dh,0ah,'Invalid entry $'
msg2  db   0dh,0ah,'Its equivalent binary is:$'
.code
main proc
mov   ax,@data
mov   ds,ax

lea   dx,msg
mov   ah,9     ;print message
```

```
int   21h

mov   ah,1
int   21h     ;read data from user

cmp   al,30h   ;check whether user enter number or something else
jnge  invalid  ;jump if any invalid entry

cmp   al,39h
jnle  invalid

lea   dx,msg2  ;print message
mov   ah,9
int   21h

and   al,0fh   ;clear upper four bits of al register
mov   cl,3     ;cl used as counter in shifting bits

mov   bl,al    ;save value in bl register
mov   bh,bl    ;move contents of bl into bh

shr   bh,cl    ;right shift bh register three times by using cl as a counter
add   bh,30h   ;make binary value visible as 0 or 1

mov   ah,2     ;print binary value
mov   dl,bh
int   21h

xor   bh,bh    ;clear bh register
mov   bh,bl

mov   cl,2     ;make cl counter value equals to two
and   bh,04h   ;clear all bits except third last bit

shr   bh,cl
add   bh,30h

mov   ah,2     ;print binary value of third last bit
mov   dl,bh
int   21h

xor   bh,bh
mov   bh,bl

and   bh,02h   ;clear all bits except second last bit
shr   bh,1

add   bh,30h
```

```
mov   ah,2    ;print second last bit
mov   dl,bh
int   21h

xor   bh,bh
mov   bh,bl

and   bh,01h   ;clear all bits except the last bit
add   bh,30h

mov   ah,2    ;print last bit in binary
mov   dl,bh
int   21h

jmp   exit

invalid:

lea   dx,msg1   ;used to print message of invalid entry
mov   ah,9
int   21h

exit:

mov   ah,4ch
int   21h

main endp
end main
```

Display the hex number of given decimal number

```
; Conversion program
;    1) Accept a decimal value (up to 5 digits), and print its hex value
;    3) Quit program.

.MODEL SMALL

.STACK 100h

.DATA
Menu    DB    10, 13, 'Enter a choice (1 or 2):'
        DB    10, 13, '1) Convert 1 to 5 Decimal values to Hex'
```

```
        DB    10, 13, '2) Quit Program', 10, 13, '$'
MenuErr    DB    10, 13, 'Choice must be a 1, 2, or 3!'
        DB    10, 13, 'Try again!', 10, 13, '$'
AskDec    DB    10, 13, 'Enter a number with 1 to 5 digits: ', 10, 13, '$'

.CODE

START    PROC
        MOV AX, @DATA             ; Startup code
        MOV DS, AX

dispMenu:
        MOV DX, OFFSET Menu          ; Display menu
        MOV AH, 09H
        INT 21H

        MOV AH, 01H                ; Get keyboard input
        INT 21H

        CMP AL, '1'
        JL dispErr

        CMP AL, '3'
        JG dispErr

        CMP AL, '1'
        JE goDec


        CMP AL, '2'
        JE goQuit

dispErr:
        MOV DX, OFFSET MenuErr       ; Display menu error.
        MOV AH, 09H
        INT 21H
        JMP dispMenu

goDec:
        CALL DEC2HEX             ; Call DEC2HEX procedure.
        JMP dispMenu


goQuit:

        MOV AL, 0                ; Exit program.
        MOV AH, 4CH
        INT 21H
```

```
START ENDP

DEC2HEX PROC                ; *** Accept a decimal value (up to 5 digits) > print it's hex value.

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

        MOV AX, 0           ; Clear AX
        PUSH AX             ; Save AX to stack (else overwritten when 0Dh is pressed)

Again:

        MOV AH, 01H         ; Get keyboard input
        INT 21H

        CMP AL, 0Dh         ; If Return is entered, start division.
        JE SDiv1

        CMP AL, '0'
        JL Again

        CMP AL, '9'
        JG Again

        MOV AH, 0           ; Change to a digit.
        SUB AL, 30h
        MOV CX, AX          ; Save digit in CX
        pop ax

        MOV BX, 10          ; Division by 10.
        MUL BX

        ADD AX, CX          ; Add CX (original number) to AX (number after multiplication).
        PUSH AX             ; Save on stack.
        JMP Again           ; Repeat.

SDiv1:

        mov cx, 0
        MOV BX, 16
        pop ax

Div1:

        DIV BX              ; Divide (Word-sized).
        PUSH DX             ; Save remainder.
```

```
        ADD CX, 1               ; Add one to counter
        MOV DX, 0               ; Clear Remainder (DX)
        CMP AX, 0               ; Compare Quotient (AX) to zero
        JNE Div1                ; If AX not 0, go to "Div1:"

getHex:                 ; Get hex number.
        MOV DX, 0               ; Clear DX.
        POP DX                  ; Put top of stack into DX.
        ADD DL, 30h             ; Conv to character.

        CMP DL, 39h             ; If DL > 39h (character 9)...
        JG MoreHex

HexRet:                 ; Display hex number
        MOV AH, 02h             ; 02h to display DL

        INT 21H                 ; Send to DOS

        LOOP getHex             ; LOOP subtracts 1 from CX. If non-zero, loop.
        JMP Skip
MoreHex:                ; Add 7h if DL > 39h (10-15)
        ADD DL, 7h              ; Add another 7h to DL to get into the A-F hex range.
        JMP HexRet              ; Return to where it left off before adding 7h.
Skip:                   ; Skip addition of 7h if it is not needed.
        RET
DEC2HEX ENDP


END START
```

Display the octal number of given decimal number

```
; display the octal number of given decimal number
.model small
.stack 90h

.data
counter db 0
curValue db 0
prevValue db 0
octal db 0
msg db "Enter a decimal number and press Enter: $"
octmsg db 13,10,"In octall: $"
```

```
.code
main proc
mov ax, @data          ;initialize DS
mov ds, ax

                       ;load and display the string msg
mov ah, 09h
lea dx, msg
int 21h

accept:
mov ah, 01
int 21h              ;read a digit

cmp al, 13           ;compare al with 13
je exit              ;jump to label exit if input is 13

sub al, 48           ;subract al with 48
mov curValue, al       ;move al to curValue

cmp counter, 1         ;compare counter with 1
jl inc_ctr           ;jump to label inc_ctr if al<1

mov al, prevValue      ;move prevValue to al
mov bl, 10
mul bl

add al, curValue       ;add curValue to al

mov prevValue, al      ;move al tp prevValue

inc counter          ;inc_ctr counter
jmp accept           ;jump to label accept

inc_ctr:
mov prevValue, al       ;move al to prevValue

inc counter              ;inc_ctr counter
jmp accept           ;jump to label accept

exit:
mov bl,prevValue        ;move prevValue to bl



mov octal, bl          ;move bl to octal
xor bx, bx           ;clear bx
```

```
jmp convertTooctall     ;jump to convertTooctall

convertTooctall:

mov ah, 09h           ;load and display the string ctmsg
lea dx, octmsg
int 21h

mov bh, octal         ;move octal to bh

and bh, 192           ;multiply 192 to bh
mov cl, 2             ;move 2 to cl
rol bh, cl           ;rotate bh 2x

add bh, 48            ;add 48 to bh
mov ah, 02            ;set the output function
mov dl, bh            ;move bh to dl
int 21h              ;print the character

mov bh, octal         ;move octal to bh
and bh, 56            ;add 56 to bh
mov cl, 5             ;move 5 to cl
rol bh, cl           ;rotate bh 5x
add bh, 48            ;add 48 to bh
mov ah, 02            ;set the output function
mov dl, bh            ;move bh to dl
int 21h              ;print the character

mov bh, octal         ;move octal to bh
and bh, 7             ;mulptiply by 7

add bh, 48            ;add 48 to bh
mov ah, 02            ;set the output function
mov dl, bh            ;move bh to dl
int 21h              ;print the character

mov ah, 04ch          ;return control to DOS
int 21h

main endp
end main
```

Display which is divisible by 2 and 3.

```
.model small
.stack 100h
.data
ev db 'divisible by 2$'
od db 'divisible by 3$'
.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 dx,offset od
mov ah,9
int 21h

mov ah,4ch
int 21h


IsEven:
mov dx,10
mov ah,2
int 21h

mov dx,13
mov ah,2
```

```
int 21h
mov dx,offset ev
mov ah,9
int 21h

mov ah,4ch
int 21h

main endp
end main
```

Get 10 numbers from user and display after placing them in an array.

```
.model small
.stack 100h
.data
msg db 'Please 10 Digits: $'
array db 11 dup('$')

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

mov dx,offset msg
mov ah,9
int 21h

lea si,array
l1:
mov ah,1
int 21h
cmp al,13
je Print
mov [si],al  ; placing in array
inc si
jmp l1

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

```
mov ah,2
int 21h


mov dx,offset array  ; displaying array to get confirm numbers are placed
mov ah,9
int 21h

mov ah,4ch
int 21h

main endp
end main
```

Get number in the form of array and display it.

```
dosseg
.model small
.stack 100h
.data
msg db 'Please 5 Digits in terms of array: $'
array db 11 dup('$')

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

mov dx,offset msg
mov ah,9
int 21h
mov bl,','

lea si,array
l1:
mov ah,1
int 21h
cmp al,13
je Print
cmp al,bl
je l1
mov [si],al  ; placing in array
inc si
jmp l1

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

```
int 21h


mov dx,offset array  ; displaying array to get confirm numbers are placed
mov ah,9
int 21h

mov ah,4ch
int 21h

main endp
end main
```

Program to reverse the hard coded string

```
;dosseg
.model small
.stack 100h
.data
arr1 db 'k','c','n','g'

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

mov si, offset arr1
mov cx, 4

L1:
mov ax, [si]
push ax
inc si
loop L1

mov cx, 4

L2:
pop dx
mov ah, 2
int 21h
inc si
loop L2


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

```
end main
```

Program to Input string from user and reverse it

```
;program to input string from user reverse it
.model small
.stack 100h
.data
var1 db 100 dup("$")
.code
inputString proc
mov ax,@data
mov ds,ax
mov bl, 0            ; counts the length of string
mov si,offset var1
l1: mov ah,1
int 21h
cmp al,13
je printString
mov [si],al
inc si
inc bl
jmp l1

printString:
mov cl, bl
print:
dec si
mov dx,[si]
mov ah,2
int 21h
loop print


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

Program to input two numbers and check if they are equal, unequal, greater or lesser

```
Dosseg
.model small
```

```
.stack 100h
.data
MsgEq db 'Numbers are Equal $'
MsgUneq db 'Numbers are Unequal and $'
MsgGr db ' First Number is greater than second number $'
MsgLs db ' First Number is lesser than second number $'
.code
main proc
mov ax, @data
mov ds, ax

mov ah, 1     ; input first number
int 21h
mov bl, al     ; saving first number to bl from al

mov al, 1      ; input second number
int 21h
mov cl, al    ; saving second number to cl from al

L1:
cmp bl,cl      ; Comparing whether they are equal or not
je EQUAL        ; Jump to Equal box, where we print the equal msg

mov dl, 10     ; for next line feed
mov ah, 2
int 21h
mov dl, 13      ; for carriage return
mov ah, 2
int 21h

mov dx, offset MsgUneq   ; but if not equal, then print msg they are not equal
mov ah, 9
int 21h

cmp bl, cl         ; again compare to check the first is greater or lesser
jg Greater         ; if greater, jump to greater to print a greater msg

mov dx, offset MsgLs  ; but if not greater, print lesser msg
mov ah, 9
int 21h
jmp PRINT



Greater:
mov dx, offset MsgGr
mov ah, 9
int 21h
```

```
jmp PRINT


EQUAL:
mov dl, 10     ; for next line feed
mov ah, 2
int 21h
mov dl, 13     ; for carriage return
mov ah, 2
int 21h

mov dx, offset MsgEq
mov ah, 9
int 21h
jmp PRINT



PRINT:
mov ah,4ch
int 21h
main endp

end main
```

Program to print the following pattern;

```
*
**
***
****
*****
```

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

```
    mov bx, 1

    mov cx, 5
    L1:
    push cx
    mov cx, bx

    L2:
    Mov dl, '*'
    mov ah,2
    int 21h
    loop L2


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

    pop cx
    loop L1

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

Program to print the following pattern;

1
22
333
4444
55555

When user input 5

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

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


mov ah, 1
int 21h

mov cl, al
sub cl, 48

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

mov bx, 1

L1:
push cx
mov cx, bx

L2:

Mov dx, bx
add dx, 48
mov ah,2
int 21h
loop L2


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

pop cx
```

```
loop L1

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

Program to read a character and display that character 100 times on next line using loop

```
;program to read a character and display that character 100 times on next line using loop
dosseg
.model small
.stack 100h
.data
.code
main proc
mov ah, 1
int 21h
mov bl,al
mov cx, 100
l1:
mov dl, bl
mov ah, 2
int 21h

mov dl, 10
mov ah, 2
int 21h

mov dl,13
mov ah, 2
int 21h

loop l1


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

Program to take input two numbers and multiply them

```
.MODEL SMALL
.STACK 100H

.DATA
iMsg1 db 10,13,"Enter First Number:$"
iMsg2 db 10,13,"Enter Second Number:$"
outputMsg db 10,13,"Their Product is = :$"
n1 db ?
n2 db ?

.CODE
MAIN PROC

MOV AX, @data
MOV DS, AX
LEA DX, iMsg1
MOV AH, 09h
INT 21H
MOV AH, 01H
INT 21H
SUB AL,48
MOV n1,AL
LEA DX, iMsg2
MOV AH, 09h
INT 21H
MOV AH, 01H
INT 21H
SUB AL,48
;MOV n2,AL
MUL n1

LEA DX, outputMsg
MOV AH, 09h

INT 21H
ADD AX,48
MOV DL,AL
MOV AH, 02H
INT 21H
MOV AH, 4CH
INT 21H
MAIN ENDP
END MAIN
```

Take a character as input from user and find the occurrence of that character in hard coded string

```
.model small
.stack 100h
.data
String db 'helloworld$'
.code
main proc
mov ax,@data
mov ds,ax

mov ah,1   ;Input
INT 21h

mov dl,0            ; using for counting and printing in end

mov bl,'$'     ;for comparing the end of string

mov si, offset String

L1:
cmp bl,[si]    ; comparing $ and letter of string
je ToEnd

cmp al,[si]      ; comparing input letter with letter of string
je Counter
inc si
jmp L1


Counter:
add dl,1  ; increment to dl register for counting occurrence
inc si
jmp L1


ToEnd:
add dl,48
mov ah,2    ; printing the counter
INT 21h
mov ah,4ch
INT 21h
```

```
main endp

end main
```

Take two numbers from the user, divide those numbers and print quotient and reminder with a proper message

```
.model small
.stack 100h
.data
Num db "Numerator = $"
Dino db "Denominator = $"
Quo db "Quotient = $"
Rem db "Reminder = $"
N db ?
D db ?

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

mov ah,1
int 21h
sub al,48
mov N,al  ; input numerator

mov ah,1 ; input dinominator
int 21h
sub al,48
mov D,al

mov ah,0  ; division
mov al,N
mov bl,D
DIV bl

push ax  ; push content from ax to stack


mov dl,10
```

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



mov dx, offset Num   ;print numerator string
mov ah,9
int 21h
mov dl,N
add dl,48
mov ah,2
int 21h


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

mov dx, offset Dino   ;print dinominator string
mov ah,9
int 21h
mov dl,D
add dl,48
mov ah,2
int 21h


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

pop cx ; pop content from stack to cx

mov dx, offset Quo   ;print quotient string
mov ah,9
```

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

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

mov dx, offset Rem   ;print reminder string
mov ah,9
int 21h
mov dl,ch
add dl,48
mov ah,2
int 21h



mov ah,4ch
int 21h


Main endp
end Main
```

Program to convert lower case string to Upper case string

```
.model small
.stack 100h
.data
var1 db 100 dup("$")
.code
inputString proc
mov ax,@data
mov ds,ax
mov bl, 0            ; counts the length of string
mov si,offset var1
```

```
l1: mov ah,1
int 21h
cmp al,13
je printString
mov [si],al
inc si
inc bl
jmp l1
printString:

mov si,offset var1
mov cl, bl
uppercase:
mov dx,[si]
sub dx, 32
mov ah,2
int 21h
inc si
loop uppercase


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

Program to convert upper case string to lower case string

```
.model small
.stack 100h
.data
var1 db 100 dup("$")
.code
inputString proc
mov ax,@data
mov ds,ax
mov bl, 0            ; counts the length of string
mov si,offset var1
l1: mov ah,1
int 21h
cmp al,13
je printString
```

```
mov [si],al
inc si
inc bl
jmp l1

printString:
mov si,offset var1
mov cl, bl
lowercase:
mov dx,[si]
add dx, 32
mov ah,2
int 21h
inc si
loop lowercase

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

Program to print carry while shifting left 8 times

```
.model small
.stack 100h
.data
.code
main proc


mov ah, 1
int 21h

mov cl, 8

mov bl, al
l1:
shl bl, 1
mov ah, 2
mov dl, 0
adc dl, 48
int 21h
```

```
loop l1


mov ah, 4ch
int 21h

main endp
end main
```

Program to check the input number is Negative or Positive

```
.model small
.stack 100h
.data
num db 10 dup('$')
msgNeg db 'Given Number is Negative. $'
msgPos db 'Given Number is Positive. $'

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

mov si, offset num


inputString:
mov ah, 1
int 21h
cmp al, 13
JE CheckNum
mov [si],al
inc si
jmp inputString

CheckNum:
```

```
cmp num,'-'
JE PrintNeg

mov dx, offset msgPos
mov ah, 9
int 21h

mov ah,4ch
int 21h


PrintNeg:
mov dx, offset msgNeg
mov ah, 9
int 21h


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