

By: Ali Hassan Soomro

Facebook: <https://web.facebook.com/AlilHassanSoomro>

Gmail: alisoomro666@gmail.com

Youtube: <https://www.youtube.com/programology>

1. Get an integer from user and display whether the number is even or odd.

```
; get an integer display the even or odd.
```

```
dosseg
```

```
.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
```

2. Display the sum of all odd numbers between 1 and 100

```
dosseg
.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

```

3. Display the sum of all even numbers between 1 and 100

```

dosseg
.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

```

4. Display the largest number in an array

```

;display the largest in an array
dosseg
.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
```

5. Display the smallest number in an array

```
;display the smallest in an array
```

```
dosseg
```

```
.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

```

6. 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

```

7. 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

```

8. 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      ;subtract 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 to 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 convertTooctal ;jump to convertTooctal

convertTooctal:

mov ah, 09h      ;load and display the string ctmmsg
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       ;multiply 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

```

9. Display which is divisible by 2,3,4.

```

dosseg
.model small
.stack 100h
.data
ev db 'divisible by 2 and 4$'
od db 'divisible by 3$'
.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
```

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

```
dosseg
.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

```

11. 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
```