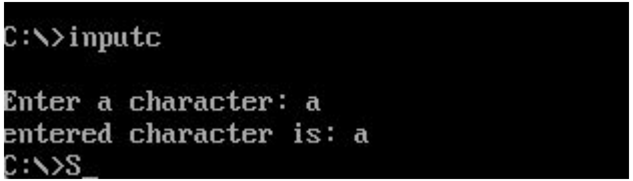
Tutorial 4

MUPI

Taking single character user input in masm

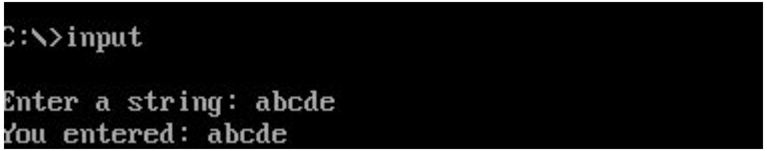
```
.model small
.stack 100h

.data
   msg db 0ah, 0dh, "Enter a character: $"
   msg2 db 0ah, 0dh, "entered character is: $"
   char db '$'

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

   mov ah, 09h        ; 09h for printing string
   lea dx, msg
   int 21h

   mov ah, 01h        ; 01h for taking character input
   int 21h
   mov char, al

   mov ah, 09h
   lea dx, msg2
   int 21h

   mov ah, 02h        ; for printing character
   mov dl, char
   int 21h

   mov ah, 4Ch        ; for termination
   int 21h
end main
```



```
C:\>inputc

Enter a character: a
entered character is: a
C:\>S_
```

String input/output

```
.model small
.stack 100h

.data
input_buffer    db  100, ?, 100 dup('$')   ; Buffer to store input string
prompt         db  13, 10, "Enter a string: $"
output_msg     db  13, 10, "You entered: $"

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

   mov ah, 09h
   lea dx, prompt
   int 21h

   mov ah, 0ah                            ; 0ah for taking string i/p
   lea dx, input_buffer
   int 21h

   mov ah, 09h
   lea dx, output_msg
   int 21h

   mov ah, 09h
   lea dx, input_buffer+2  ; Skip the length byte,
   int 21h

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



```
C:\>input

Enter a string: abcde
You entered: abcde
```

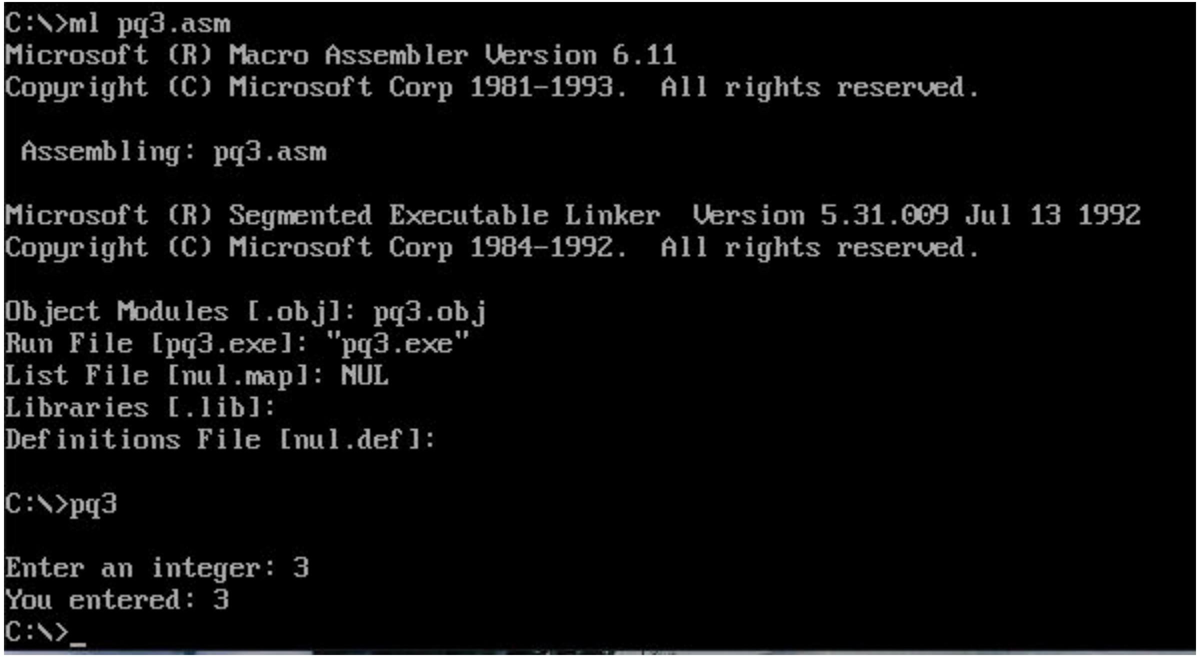
Taking An integer Input

```
.model small
.stack 100h
.data
   msg1 db 10, 13, "Enter an integer: $"
   msg2 db 10, 13, "You entered: $"
   num dw ?
.code
main proc
   mov ax, @data
   mov ds, ax

   mov ah, 9
   lea dx, msg1
   int 21h

   mov ah, 01h
   int 21h

   sub al, 30h
   mov bl, al

   mov ah, 9
   lea dx, msg2
   int 21h

   add bl, 30h
   mov dl, bl

   mov ah, 02h
   int 21h

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

Taking An integer Input

```
.model small
.stack 100h
.data
   msg1 db 10, 13, "Enter an integer: $"
   msg2 db 10, 13, "You entered: $"
   num dw ?
.code
main proc
   mov ax, @data
   mov ds, ax

   mov ah, 9
   lea dx, msg1
   int 21h

   mov ah, 01h    ; Function to read a character from STDIN
   int 21h        ; DOS interrupt

   sub al, 30h    ; Convert ASCII character to integer
   mov bl, al     ; Move the entered integer to bl

   mov ah, 9
   lea dx, msg2
   int 21h

   add bl, 30h    ; Convert integer back to ASCII character
   mov dl, bl     ; Move the ASCII character to dl

   mov ah, 02h    ; Function to print character to STDOUT
   int 21h        ; DOS interrupt

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

Output



```
C:\>ml pq3.asm
Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993.  All rights reserved.

 Assembling: pq3.asm

Microsoft (R) Segmented Executable Linker  Version 5.31.009 Jul 13 1992
Copyright (C) Microsoft Corp 1984-1992.  All rights reserved.

Object Modules [.obj]: pq3.obj
Run File [pq3.exe]: "pq3.exe"
List File [nul.map]: NUL
Libraries [.lib]:
Definitions File [nul.def]:

C:\>pq3

Enter an integer: 3
You entered: 3
C:\>_
```

Practice problem 1

Q1. Design a MASM program that takes user input for two hexadecimal numbers (assume that each number is a two-digit hexadecimal), computes their sum, and prints whether the sum is even or odd.

Data

inp1 that holds the message $ denotes the end of the string.

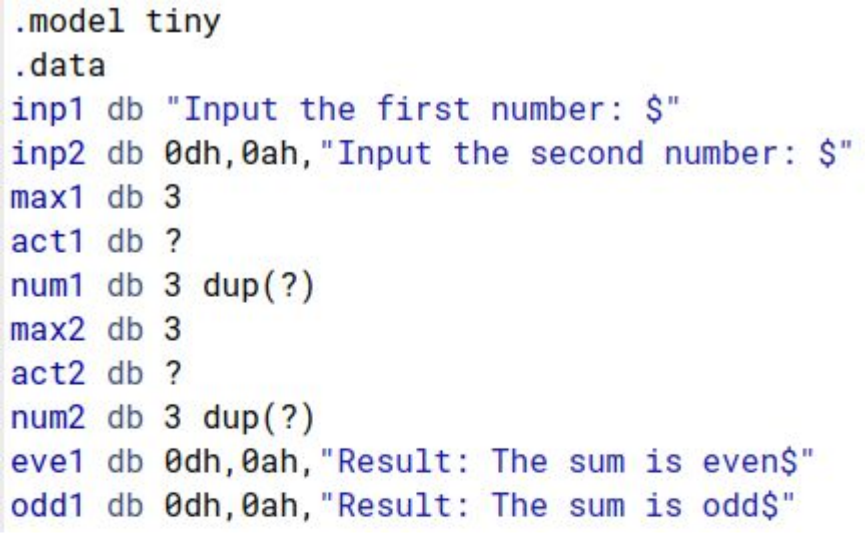
max1 db 3 maximum number of digits allowed

act1 db store the actual number of digits

num1 db 3 dup(?): declares an array num1 of 3 bytes

These lines declare two strings

indicating that the sum is even and the sum is odd



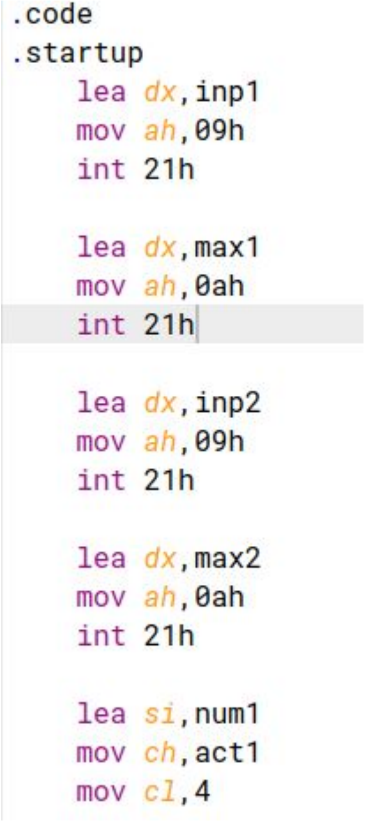
```
.model tiny
.data
inp1 db "Input the first number: $"
inp2 db 0dh,0ah,"Input the second number: $"
max1 db 3
act1 db ?
num1 db 3 dup(?)
max2 db 3
act2 db ?
num2 db 3 dup(?)
eve1 db 0dh,0ah,"Result: The sum is even$"
odd1 db 0dh,0ah,"Result: The sum is odd$"
```

Code

int 21h DOS interrupt

mov ah , 09h displays the string whose offset is in dx.

This segment loads offset address into the resp registers



```
.code
.startup
    lea dx,inp1
    mov ah,09h
    int 21h

    lea dx,max1
    mov ah,0ah
    int 21h

    lea dx,inp2
    mov ah,09h
    int 21h

    lea dx,max2
    mov ah,0ah
    int 21h

    lea si,num1
    mov ch,act1
    mov cl,4
```

Conversion

Shifts the bits in bl to the left.

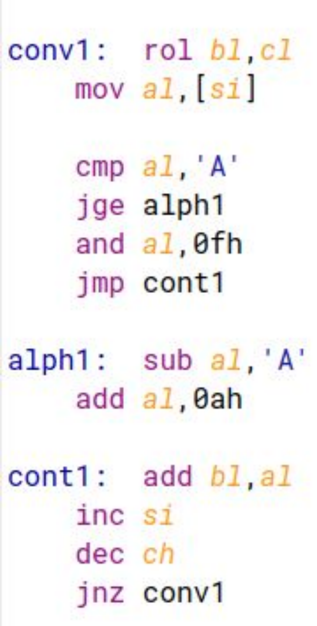
Compares the value in al with the ASCII value of the character 'A'

Subtracts the ASCII value of 'A' from the value in al, to get hex from A'-'F'

Adds the ASCII value of '0ah' (10 in decimal)

repeats the conversion process until ch becomes zero.

Similarly for num 2 : conv2 , alph2 and cont2



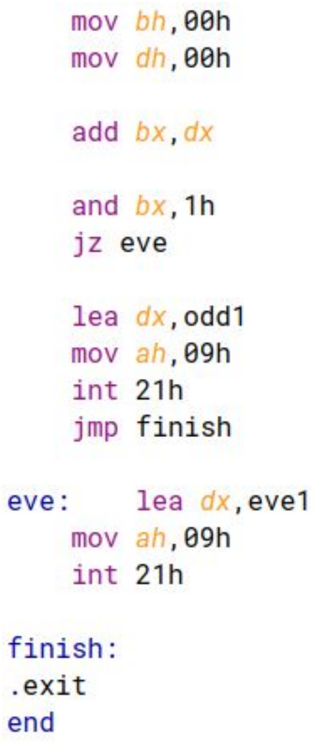
```
conv1:  rol bl,cl
    mov al,[si]

    cmp al,'A'
    jge alph1
    and al,0fh
    jmp cont1

alph1:  sub al,'A'
    add al,0ah

cont1:  add bl,al
    inc si
    dec ch
    jnz conv1
```

Addition and output

Clears bh and dh for junk values

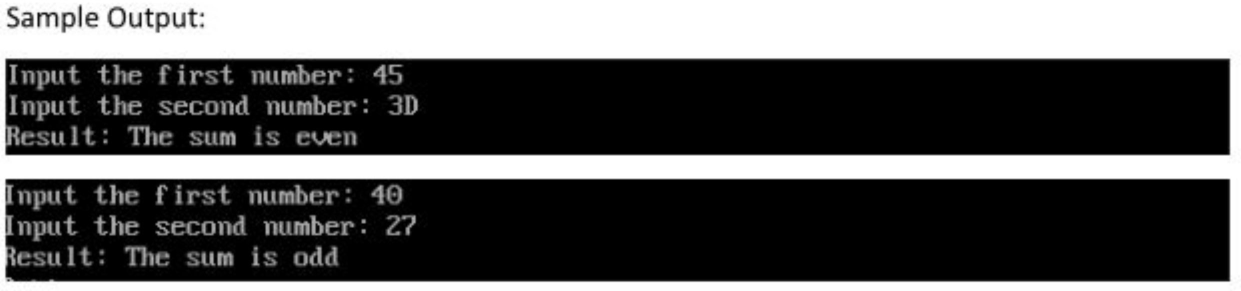
and bx,1h performs a bitwise AND operation

Prints even if 0 or odd if set



```
    mov bh,00h
    mov dh,00h

    add bx,dx

    and bx,1h
    jz eve

    lea dx,odd1
    mov ah,09h
    int 21h
    jmp finish

eve:    lea dx,eve1
    mov ah,09h
    int 21h

finish:
.exit
end
```

Expected output



Sample Output:

```
Input the first number: 45
Input the second number: 3D
Result: The sum is even
```

```
Input the first number: 40
Input the second number: 27
Result: The sum is odd
```

Practice problem 2

Q2. Write a MASM program which takes input for a decimal number in the range of 0-9999,
converts it into its hexadecimal equivalent and displays the result on the screen.

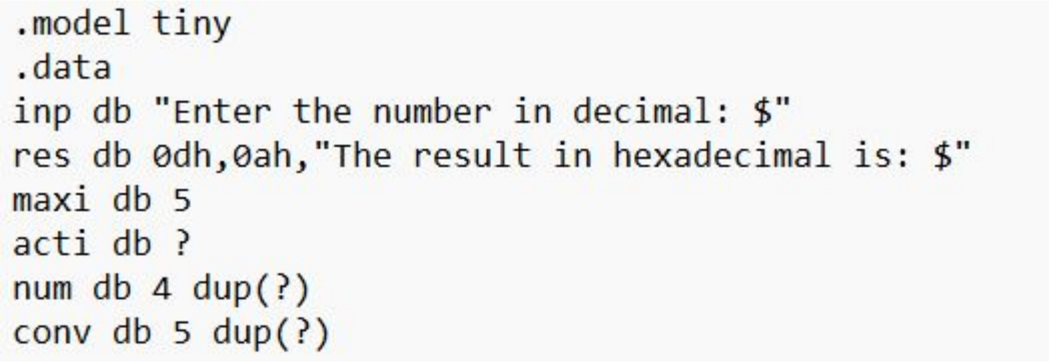
Data

Maxi - maximum number of digits for decimal input

Acti - actual number of digits entered by user

Num - array to store numbers typed in by user

Conv - array to store converted hexadecimal values



```
.model tiny
.data
inp db "Enter the number in decimal: $"
res db 0dh,0ah,"The result in hexadecimal is: $"
maxi db 5
acti db ?
num db 4 dup(?)
conv db 5 dup(?)
```

The Logic

Retrieve each decimal digit from the array

Convert to hexadecimal representation.

Store it in the conv array.

If number is greater than or equal to 10 , subtract 10 and add 65 (ASCII for A) else add 0(Since numbers will be within 0-9). [to get the numbers in hexadecimal range]

Print the digits of hexadecimal number equivalent.

Example

44 - 0000 0000 0010 1100

2 C

To perform this in MASM , we use ASCII values.

So , 44 would be broken down as 40 + 4 (this is done by multiplying the position value with the digit)-

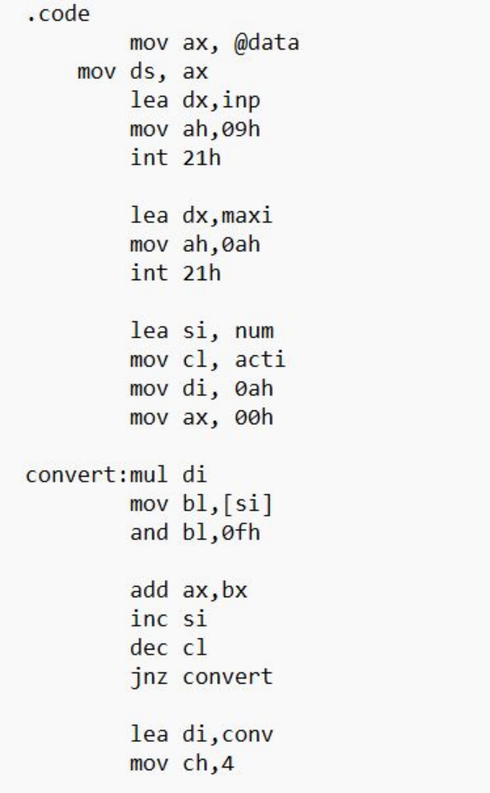
4*10 +4*1

4 in ASCII table (hexadecimal) is 34 -> retrieve last 4 bits = 4

40 in ASCII table(hexadecimal) is 28 -> Now we add both numbers = we get 2C which is the ASCII value for 44.

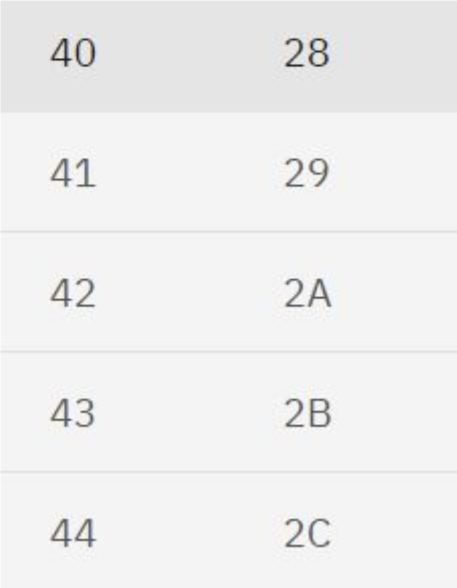
Conversion to hexadecimal.

Steps mentioned in previous slide are being performed in these code sections.



```
.code
        mov ax, @data
    mov ds, ax
        lea dx,inp
        mov ah,09h
        int 21h

        lea dx,maxi
        mov ah,0ah
        int 21h

        lea si, num
        mov cl, acti
        mov di, 0ah
        mov ax, 00h

convert:mul di
        mov bl,[si]
        and bl,0fh

        add ax,bx
        inc si
        dec cl
        jnz convert

        lea di,conv
        mov ch,4
```

ASCII table with decimal and hexadecimal equivalents.



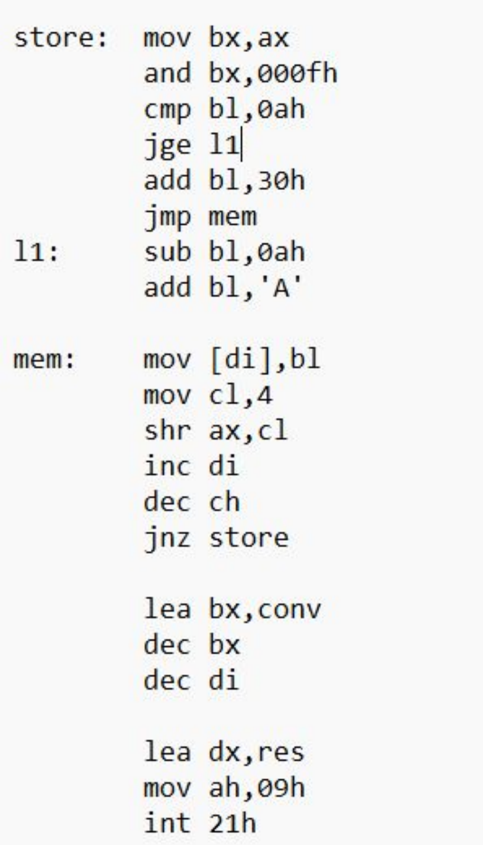
40	28
41	29
42	2A
43	2B
44	2C

Storing numbers in memory

If number is greater than or equal to 10 , subtract 10 and add 65

(ASCII for A) else add 0(Since numbers will be within 0-9).

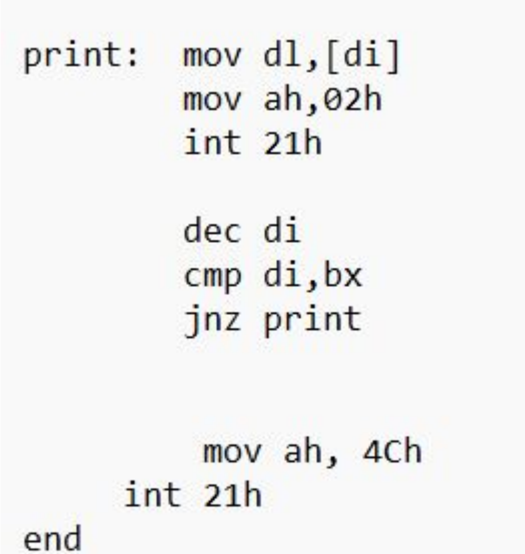
[to get the numbers in hexadecimal range]



```
store:  mov bx,ax
        and bx,000fh
        cmp bl,0ah
        jge l1
        add bl,30h
        jmp mem
l1:     sub bl,0ah
        add bl,'A'

mem:    mov [di],bl
        mov cl,4
        shr ax,cl
        inc di
        dec ch
        jnz store

        lea bx,conv
        dec bx
        dec di

        lea dx,res
        mov ah,09h
        int 21h
```

Printing the numbers

Print each character one by one from memory.



```
print:  mov dl,[di]
        mov ah,02h
        int 21h

        dec di
        cmp di,bx
        jnz print


         mov ah, 4Ch
     int 21h
end
```