**Microprocessor Lab  
Lab Experiment No. 4**

Name: Ninad Rao Roll No. 53

**Aim**: 8086 Assembly Language Programming.

**Instructions on how to use TASM**:

Steps for creating the program:

1. TASM is loaded
2. TASM < Edit - We will get an edit window
3. Type the program here
4. Save the file as <filename>.asm

Steps for running the program:

1. c:\tasm> Type here tasm filename

**c:\tasm> tasm <filename>.asm**

1. c:\tasm> Linking the program

**c:\tasm> tlink <filename>.obj**

1. c:\tasm> Now to execute the program and get to the result window

**c:\tasm> td <filename>.exe**

**Programs for 8086 Assembly Language Programming**:

1. **Program for 16 bit BCD addition**:

**Code**:

.model small

.data

a dw 3629H

b dw 4738H

.code

mov ax, @data ; Initialize data section

mov ds, ax

mov ax, a ; Load number1 in ax

mov bx, b ; Load number2 in bx

add al, bl ; Add lower two digits. Result in al

daa ; Adjust result to valid bcd

mov bl, al ; Store result in bl

adc ah, bh ; Add upper two digits. Result in ah

mov al, ah ; al=ah as daa works on al only

daa ; Adjust result to valid BCD

mov bh, al ; Store result in bh

mov ch, 04h ; Count of digits to be displayed

mov cl, 04h ; Count to roll by 4 bits

l2: rol bx, cl ; Roll bl so that msb comes to lsb

mov dl, bl ; Load dl with data to be displayed

and dl, 0fH ; Get only lsb

cmp dl, 09 ; Check if digit is 0-9 or letter A-F

jbe l4

add dl, 07 ; If letter add 37H else only add 30H

l4: add dl, 30H

mov ah, 02 ; Function 2 under INT 21H (Display character)

int 21H

dec ch ; Decrement Count

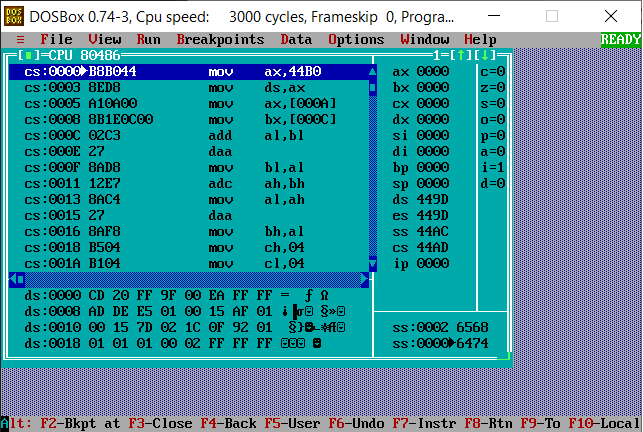
jnz l2

mov ah, 4cH ; Terminate Program

int 21H

end

**Output**:



1. **Conversion of 2 digits packed BCD to unpacked BCD**:

**Code**:

data segment

unpackedbcd1 db 05h

unpackedbcd2 db 02h

ascii1 db ?

ascii2 db ?

packedbcd db ?

data ends

code segment

assume ds:data, cs:code

start:

mov ax, @data

mov ds, ax

mov al, unpackedbcd1

mov bl, unpackedbcd2

mov ah, al

mov bh, bl

add ah, 30h

add bh, 30h

mov ascii1, ah

mov ascii2, bh

mov cl, 04h

rol al, cl

or al, bl

mov bh, al

mov packedbcd, bh

mov dl, ascii1

mov ah, 02

int 21h

mov dl, ascii2

mov ah, 02

int 21h

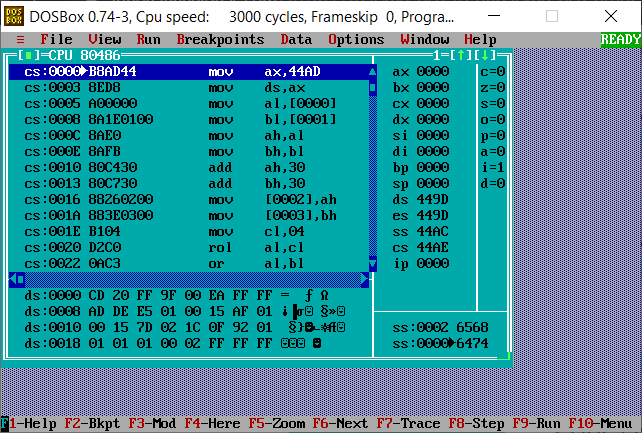
mov ah, 4ch

int 21h

code ends

end start

**Output**:



1. **Program for evaluating given logical expression (A=B+C-D\*E):**

**Code:**

.model small

assume cs: code, ds: data

data segment

num\_b db 30h

num\_c db 20h

num\_d db 08h

num\_e db 05h

data ends

code segment

start:

mov ax, data

mov ds, ax

mov ax, 0000h

mov bx, 0000h

xor cx, cx

xor dx,dx

mov al, num\_b

mov bl, num\_c

add al, bl

mov cx, ax

mov al, num\_d

mov dl, num\_e

mul dl

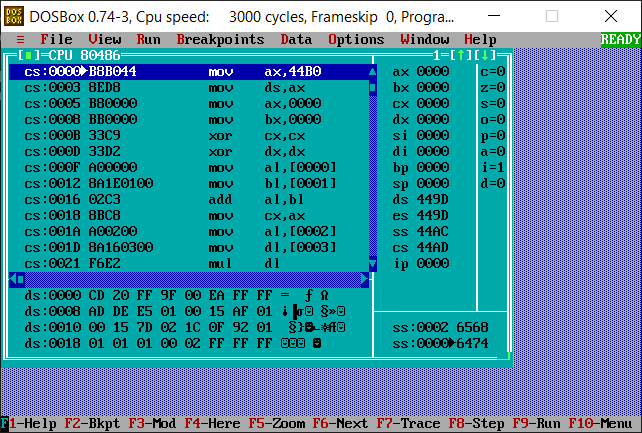
sub al, cl

int 03h

code ends

end start

**Output:**

****

1. **Program to move a set of numbers from one block to another**:

**Code**:

data segment

blk dw 1111h, 2222h, 3333h, 4444h, 5555h, 6666h, 7777h, 8888h, 9999h, 0000h

data ends

code segment

assume ds:data, cs:code

start:

mov ax, data

mov ds, ax

mov es, ax

lea si, blk

mov di, si

add di, 0020h

mov cx, 000ah

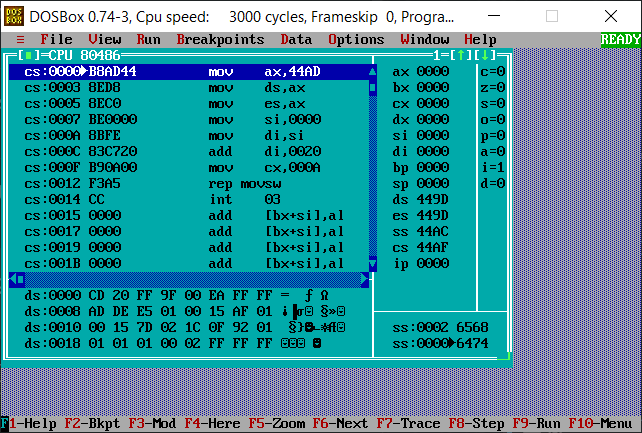
rep movsw

int 03h

code ends

end start

**Output**:



1. **Program to count number of 1’s and 0’s in a given 8 bit number**:

**Code**:

data segment

no dw 5648h

z dw ?

o dw ?

data ends

code segment

assume cs:code, ds:data

start:

mov ax, @data ; Initialize Data Segment

mov ds, ax

mov ax, no

mov bx, 00h ; Initially ZERO count = 00H

mov cx, 10h ; Initialize bit counter = 10H

mov dx, 00h ; Initially ONES count = 00H

up:

rol ax,1 ; Rotate left 'no'

jc one ; IFCY = 1 go to label ONE

inc bx ; INC O'S Count

jmp nxt

one:

inc dx ; Inc i's count

nxt:

dec cx ; Dec cx and if cx = 0 go to up

jnz up

mov z, bx ; ZEROS = no. of 0's

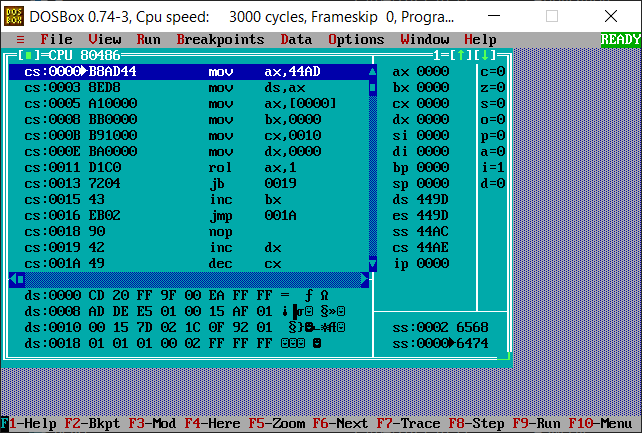
mov o, dx ; ONES = no. of 1's

int 3 ; Terminate program by break point.

code ends

end start

**Output**:



1. **Program to find even and odd numbers from a given list**:

**Code**:

data segment

a dw 1,2,3,4,5,6,7,8,9,10

data ends

code segment

assume ds:data, cs:code

start:

mov ax, @data

mov ds, ax

lea si, a

mov dx, 0000

mov bl, 02

mov cl, 10

l1:

mov ax, word ptr[si]

div bl

cmp ah, 00

jnz l2

inc dh

jmp l3

l2:

inc dl

l3:

add si, 2

dec cl

cmp cl, 00

jnz l1

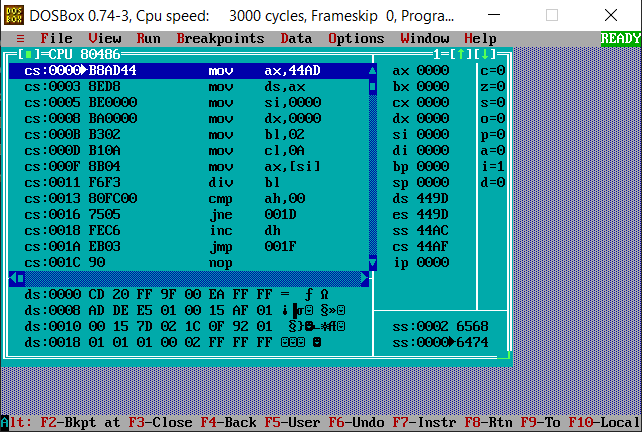
mov ah, 4ch

int 21h

code ends

end start

**Output**:



1. **Program to search a number from given list**:

**Code**:

data segment

string1 db 11h,22h,33h,44h,55h

msg1 db "found$"

msg2 db "not found$"

se db 33h

data ends

print macro msg

mov ah, 09h

lea dx, msg

int 21h

int 3

endm

code segment

assume cs:code, ds:data

start:

mov ax, data

mov ds, ax

mov al, se

lea si, string1

mov cx, 04h

up:

mov bl,[si]

cmp al, bl

jz fo

inc si

dec cx

jnz up

print msg2

jmp end1

fo:

print msg1

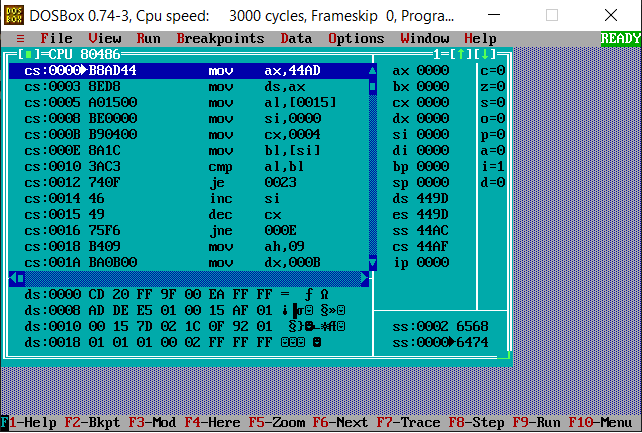
end1:

int 3

code ends

end start

**Output**:



**Conclusion**: Thus, we have studied and understood 8086 Assembly Language Programming.