CERTIFICATE

**Certified that this is the bonafide Record of the work done by Mr/Ms KRISHNA KUMAR Register No 143043 of Semester 4 ECE-A Class, 2014 Admission in the EC 14L1Microprocessor Laboratory during the year 2015-2016.**

**Staff-in-Charge Head of the Department**

**Internal Examiner External Examiner**

**EC14L1 MICROPROCESSOR LABORATORY(2012 scheme)**

|  |  |
| --- | --- |
| 1. | Introduction to DEBUG program commands |
|  | - Examining and modifying the contents of the memory |
|  | - Assembling 8086 instructions with the ASSEMBLER commands |
|  | - Executing 8086 instructions and programs with the Trace and GO Command. |
|  | - Debugging a program |
| 2. | Assembly language program development using IBM/PC Macro assembler |
|  | - Creating an Assembler source file |
|  | - Assembling source program with MASM |
|  | - The link program - creating a RUN module |
|  | - Typical programming examples (at least 15 no’s) |
| 3. | Familiarization of a typical 8086 microprocessor trainer kit and its operations |
| n4. | Stepper motor/DAC/ADC/Display interface to 8086 |

References:

1. Peter Abel, Niyaz Nizamuddin, *IBM ®PC Assembly language and Programming*, Prentice Hall India ,5/e ,2001, ISBN: 9788120320949

2. Lyla B.Das, *The x86 Microprocessors: Architecture, programming and Interfacing (8086 to Pentium*), Pearson Education,2010, ISBN 978-81-317-3246-5

*Note : 50 % marks is earmarked for continuous evaluation, and 50 % marks for end semester examination to be assessed by two examiners .A candidate shall secure a minimum of 50 % marks in the aggregate and 50 % minimum in the end semester examination for a pass.*

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| I | **INTRODUCTION TO ASSEMBLY LANGUAGE PROGRAMMING** | 5 |
|  | Familiarisation of DEBUG commands | 7 |
| II | **8086 ASSEMBLY LANGUAGE PROGRAMS USING MACRO ASSEMBLER (MASM)** |  |
| 1.a | Program To Input A Single Character Through Keyboard With Echo | 8 |
| 1.b | Program To Input and check a password | 9 |
| 2.a | Program to Input a string and display it in reverse. Also check whether it is a Palindrome | 11 |
| 2.b | Program to input the PIN of a user without revealing it. Also check whether key entry is a valid (numeric only) | 14 |
| 3.a | Program to count the number of words in a string input through keyboard ( using string read interrupt) | 16 |
|  |  |  |
| 4.a | Program to Star pattern display : using MACRO and Procedures | 20 |
| 1. 5a(a) With the help of a neat flow chart and necessary comments write an 8086 ALP to find the sum of 10 two digit (packed)BCD numbers(assume answer less than 99) |  |  |
| 5.a | Program to find sum of 10 nos | 22 |
| b | Program to find average of 10 nos | 23 |
| 6.a | Program to find the square of nibbles of a byte | 25 |
| B | Program to find the square of an 8 bit and 16 bit no | 26 |
| 7.a | Program to check whether the nibbles of an 8 bit no is odd or even | 28 |
| B | Program to find the no of odd nos in an array of 16 elements | 30 |
| 8.a | Program to find largest of 3 data stored in data segment | 32 |
| b | Program to find largest of n data | 34 |
| 9.a | Program to unpack the digits of a 2 digit bcd no and find its sum | 36 |
| b | Program to check whether a given no is a divisible by 9 (<99) | 38 |
| 10.a | Program to reverse the nibbles of class roll no and find their sum | 40 |
| b | Program to find the reverse of university roll no. | 42 |
| 11.a | Program to find the largest of 2 nibbles of a byte stored in data segment | 44 |
| b | Program to find the largest of 3 bytes stored in data segment | 46 |
| 12.a | Program to find the sum of bits if a byte stored in data segment  And display its parity | 48 |
| B | Program to find the modulo 2 sum of even index bits | 50 |
|  |  |  |
| 13.a | Program to find the number of ones and zeros in an 8 bit data | 52 |
| b | program to find the gray code of the given number | 54 |
| 14.a | Program to mask the higher nibble of all 10 nos stored in consecutive memory locations and to store them in another location | 56 |
| b | Program to find the decimal sum of all lower nibbles | 58 |
| 15 | Program to count and display all vowels in given sentence | 60 |
| 16 | Program to get system clock and to display a blinking message ‘Hai All!’ with a ONE sec delay in between | 63 |
| 17 | Program to read a valid alphabet from key board and to display its other case | 65 |
| 18 | Program to check whether a year is leap year or not | 67 |
| 19 | Program to print ap | 69 |
| III | **PROGRAMS USING 8086 TRAINER KIT** | 73 |
| a | Add two 16 bit numbers | 74 |
| b | Largest of two 16 bit numbers | 75 |
|  |  |  |
| IV | **INTERFACING PROGRAMS** | 76 |
| a | Stepper motor – Rotate clock wise continuously | 77 |
| b | DAC – Generate a Ramp |  |
| c | LCD – Display Hello |  |

**INTRODUCTION TO ASSEMBLY LANGUAGE PROGRAMMING:**

**LEVEL OF PROGRAMMING**

There are three levels of programming

1. Machine language

2. Assembly language

3. High level language

Machine language programs are programs that the computer can understand and execute directly. Assembly language instructions match machine language instructions, but are written using character strings so that they are more easily understood and High-level language instructions are much closer to the English language and are structured.

Ultimately, an assembly language or high level language program must be converted into machine language by programs called translators. If the program being translated is in assembly language, the translator is referred as an assembler, and if it is in a high level language the translator is referred to as a compiler or interpreter.

**ASSEMBLY LANGUAGE PROGRAM DEVELOPMENT TOOLS:**

EDITOR: An editor is a program, which allows to create a file containing the assembly language statements of the program.

ASSEMBLER: An assembler program is used to translate the assembly language Mnemonic instructions to the corresponding binary codes. The second file generated by assembler is called the assembler List file.

LINKER: A Linker is a program used to join several object files into one large object file. The linkers produce link files with the .EXE extension.

DEBUGGER: If a program requires no external hardware, then a debugger can run and debug the program. A debugger is a program, which allows to load object code program into system memory, execute the program, and troubleshoot or “debug” it.

**ASSEMBLER DIRECTIVES:**

An assembler is a program used to convert an assembly language program into the equivalent machine code modules. The assembler decides the address of each label and substitutes the values for each of the constants and variables. It then forms the machine code for mnemonics and data in assembly language program.

Assembler directives help the assembler to correctly understand assembly language programs to prepare the codes. Commonly used assembler directives are DB, DD, DW, DUP, ASSUME, BYTE, SEGMENT, MACRO, PROC, OFFSET, NEAR, FAR, EQU, STRUC, PTR, END, ENDM, ENDP etc.

**DB** Define byte directive stores bytes of data in memory.

**BYTE PTR** This directive indicates the size of data referenced by pointer.

**SEGMENT** This directive is to indicate the start of the segment.

**DUP (Duplicate)** The DUP directive reserves memory locations given by the number preceding it, but stores no specific values in any of these locations.

**ASSUME** The ASSUME statement is only used with full segment definitions.This statement tells the assembler what names have been chosen for the code, data, extra and stack segments.

**ORG** The ORG (origin) statement changes the starting offset address in a segment.

**PROC** and **ENDP** The PROC and ENDP directives indicate start and end of a procedure (Subroutine). Both the PROC and ENDP directives require a label to indicate the name of the procedure. The PROC directive, must also be followed with the NEAR or FAR. A NEAR procedure is one that resides in the same code segment as the program. A FAR procedure may reside at any location in the memory system.

A macro is a group of instructions that performs one task, just as a procedure. The difference is that a procedure is accessed via a CALL instruction, while a macro is inserted in the program at the point of usage as a new sequence of instructions.

* **MACRO** The first statement of a macro is the MACRO directive preceded with name of the macro
* **ENDM** The last statement of a macro is the ENDM instruction. Never place a label in front of the ENDM statement
* **PUBLIC & EXTRN** The public and extrn directives are very important to modular programming. PUBLIC is used to declare the labels of code, data or entire segments that are available to other program modules and EXTRN is used to declare the labels that are external to a module.

**8086 PROGRAMMING**

**TO EXECUTE 80X86 PROGRAMS USING DEBUG**

DEBUG is a utility program that allows a user to load an 80x86 programs in to memory and execute it step by step. DEBUG displays the contents of all processor registers after each instruction executes, allowing user to determine if the code is performing the desired task.

To execute the program file PROG.EXE use this command: DEBUG PROG.EXE

DEBUG uses a **hyphen** symbol as its command prompt, so you should see a “ - ” appear on display screen.

**DEBUG- Testing and edition tool help ; MS-DOS based program.**

**MS-DOS prompt:\debug [filename .exe/.com/others]**

Assemble A [address] Compare C range address Dump D [range]

Enter E address [list] Fill F range list

Go G [=address] [addresses] Hex H value1 value2

Input I port

Load L [address] [drive] [firstsector] [number] Move M range address

Name N [pathname] [arglist]

Output O port byte

Proceed P [=address] [number] Quit Q

Register R [register]

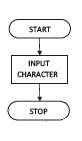
Search S range list

Trace T [=address] [value] Unassembled U [range]

Write W [address] [drive] [firstsector] [number]

**PROGRAM NO: 1.a**

**Write a program to input a single character through keyboard with echo!!!**



ASSUME CS: CODE

CODE SEGMENT

START: MOV AH, 01H ; INPUT WITH ECHO

INT 21H

MOV AH, 4CH ; RETURN TO DOS PROMPT

INT 21H

CODE ENDS

END START

Checking the o/p

**A**

**PROGRAM NO: 1.b**

**Write a program To Input and check a password!!!**

assume cs:code,ds:data

data segment

pass db "krish$"

ent db 0ah,0dh, "enter password $"

pmat db 0AH,0DH, "password match$"

pmis db 0AH,0DH, "password mismatch$"

str1 db 15 dup("$")

data ends

code segment

start:

mov ax,data

mov ds,ax

mov es,ax

lea dx,ent

mov ah,09h

int 21h

lea dx,str1

mov ah,0ah

int 21h

cmp [str1+1],05h

jne msg2

mov si,offset pass

mov di,offset [str1+2]

cld

mov cx,5h

repe cmpsb

jz msg

jnz msg2

msg:lea dx,pmat

mov ah,09h

int 21h

jmp ref

msg2:

lea dx,pmis

mov ah,09h

int 21h

ref:

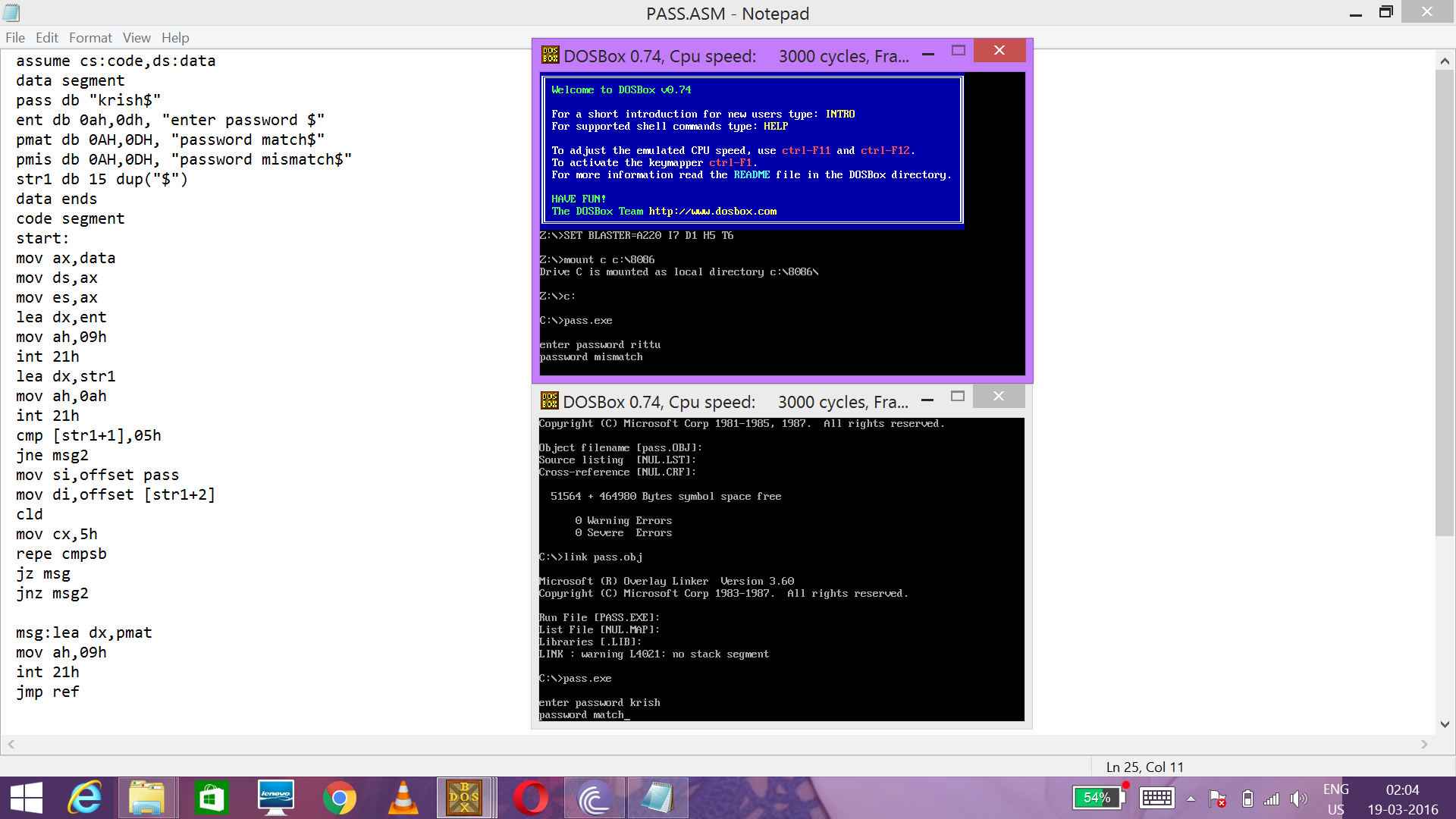
mov ah,4ch

int 21

code ends

end start

Checking the o/p



**PROGRAM NO: 2.a**

**Write a program to Input a string and display it in reverse .**

**Also check whether it is PALINDROME!!!**

ASSUME CS: CODE , DS:DATA

DATA SEGMENT

MSG1 DB 0AH,0DH, "ENTER THE STRING(LESS THAN 10 CHARACTERS) $"

MSG2 DB 0AH,0DH, "REVERSED STRING IS $"

MSG3 DB 0AH,0DH, "PALINDROME$"

MSG4 DB 0AH,0DH, "NOT PALINDROME$"

N EQU 10

STR1 DB N,00, N+1 DUP("$")

STR2 DB N+1 DUP("$")

DATA ENDS

CODE SEGMENT

START: MOV AX,DATA

MOV DS,AX

MOV ES,AX

LEA DX,MSG1

MOV AH,09H

INT 21H

LEA DX,STR1

MOV AH,0AH

INT 21H

MOV CH,00H

MOV CL,STR1+1

MOV SI,OFFSET STR1+2

LEA DI,STR2

ADD DI,CX

DEC DI

APPLE:

MOV BH,[SI]

MOV [DI],BH

INC SI

DEC DI

LOOP APPLE

LEA DX,MSG2

MOV AH,09H

INT 21H

LEA DX,STR2

MOV AH,09H

INT 21H

;;; PALINDROME CHECK

;MOV AX,DATA

;MOV DS,AX

;MOV ES,AX ; GIVE AT THE TOP ITSELF

LEA SI, STR1+2

LEA DI, STR2

MOV CH,00H

MOV CL,STR1+1

CLD ; GO UP IN STRING COMPARISON

REPE CMPSB

LEA DX, MSG3

JCXZ YES

LEA DX, MSG4

YES: MOV AH,09H

INT 21H

MOV AH,4CH

INT 21H

CODE ENDS

END START

Checking the o/p

ENTER THE STRING(LESS THAN 10 CHARACTERS) apple

REVERSED STRING IS elppa

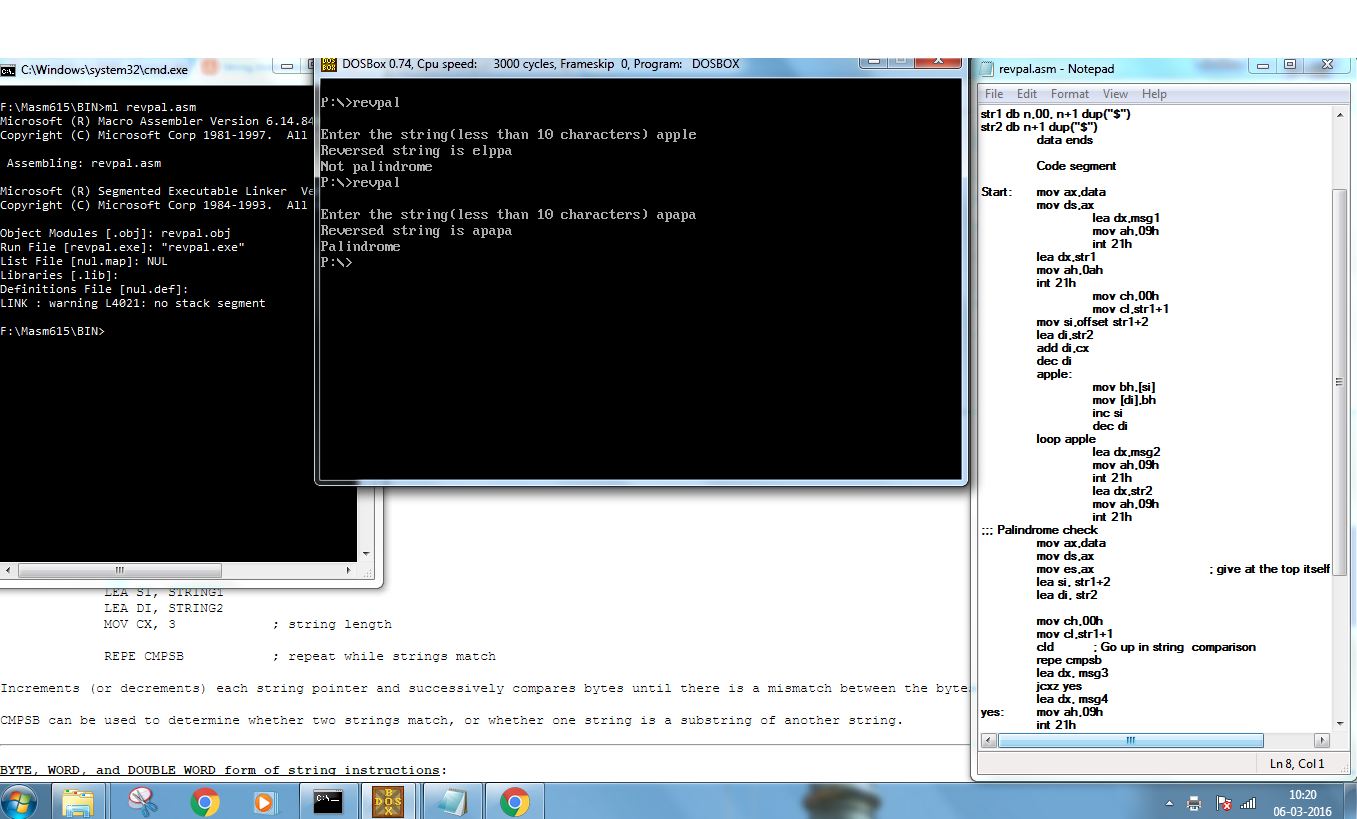
NOT PALINDROME

ENTER THE STRING(LESS THAN 10 CHARACTERS) apapa

REVERSED STRING IS apapa

PALINDROME

**Sample screen shot:**

****

**PROGRAM NO: 2.b**

**Write a program to input the PIN of a user without revealing it.**

**Also check whether key entry is a valid (numeric only) !!!**

;;READ USER PIN WITHOUT ECHO

ASSUME CS: CODE,DS:DATA

DATA SEGMENT

MSG1 DB 'ENTER YOUR PIN: $'

PIN DB 00H,00H,00H,00H

WELCOME\_MSG DB ' WELCOME $'

WRONG\_MSG DB ' SORRY, INVALID KEY PRESSED$'

DATA ENDS

CODE SEGMENT

START: MOV AX,DATA

MOV DS,AX

LEA DX, MSG1 ; PROMPT USER I/P

MOV AH,09H

INT 21H

; READ WITHOUT ECHO FOUR TIMES ( VALID PIN)

LEA BX,PIN

MOV CX,04H

FOR\_PIN: MOV AH,08H ; READ W/O ECHO

INT 21H

MOV DH,AL

MOV DL,'\*' ; TO HIDE ON SCREEN

MOV AH,02h

INT 21h

CMP DH,39H ; CHECK VALIDITY OF KEY

JG WRONG\_KEY

CMP DH,30H

JL WRONG\_KEY

MOV [BX],DH

INC BX

LOOP FOR\_PIN

LEA DX, WELCOME\_MSG

JMP WELCOME

WRONG\_KEY: LEA DX, WRONG\_MSG

WELCOME: MOV AH,09H

INT 21H

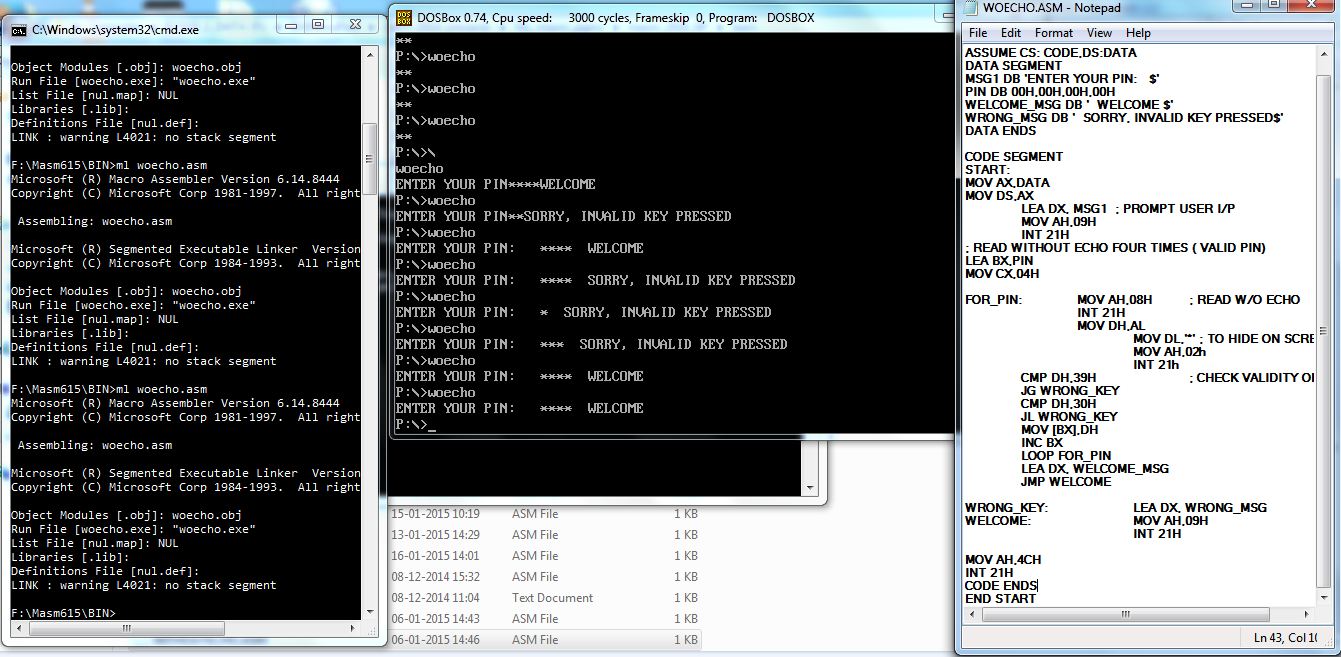
MOV AH,4CH

INT 21H

CODE ENDS

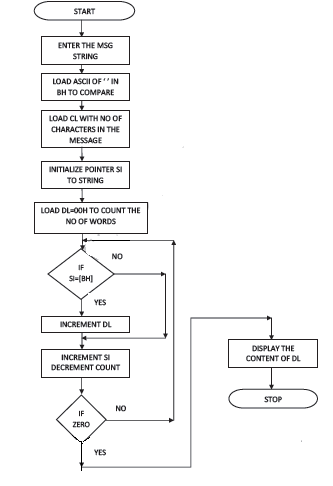
END START

Checking the o/p

****

**PROGRAM NO: 3.a**

**Write a program to count the number of words in a string input through keyboard ( using string read interrupt)!!!**



ASSUME CS: CODE, DS: DATA

DATA SEGMENT

M1 DB " ENTER THE MESSAGE: $"

M2 DB 0AH, 0DH,"THE NUMBER OF WORDS ARE : $"

M3 DB 65,00,67 DUP ("$")

M4 DB 20H DUP (' ')

ANS DB 00H

ASCIICODE DB ‘0123456789ABCDEF’

DATA ENDS

CODE SEGMENT

START:

MOV AX, DATA

MOV DS, AX ; INITIALISE DATA SEGMENT

LEA DX, M1

MOV AH, 09H ; TO PRINT MESSAGE M1

INT 21H

LEA DX, M3

MOV AH, 0AH ; READ IN THE STRING TYPED THROUGH KEYBOARD. UPTO 64 CHARACTERS

INT 21H

LEA SI, M4

MOV BH,' ' ; LOAD ASCII OF SPACE IN 'BL' TO COMPARE

LEA SI, M3

INC SI

MOV CL, [SI] ;TO LOAD THE NO OF CHARACTERS ENTERD IN 'CL'

MOV CH, 00H

MOV DL, 00H

LOOP1:

INC SI

MOV BL, [SI]

CMP BL, BH

JE LOOP2

JMP LOOP3

LOOP2:

INC DL ; IF THERE IS SPACE INCREMENT THE COUNT

LOOP3: LOOP LOOP1

MOV ANS, DL

LEA DX, M2

MOV AH, 09H ; DISPLAY MSG1

INT 21H

**;; TO Print no of words**

LEA BX, ACSIICODE

MOV AL, ANS

AND AL,0FOH

MOV CL,04H

SHR AL,CL

XLATB

MOV DL,AL

MOV AH, 02H

INT 21H

MOV AL, ANS

AND AL,0FH

XLATB

MOV DL,AL

MOV AH, 02H

INT 21H

MOV AH, 4CH ; RETURN TO DOS

INT 21H

CODE ENDS

END START

Checking the o/p

**ENTER THE MESSAGE**:MILES TO GO BEFORE I SLEEP

**THE NO OF WORDS ARE: 6**

**PROGRAM NO 4A**

**WAP TO PRINT TRIANGULAR PATTERN**

star macro n

mov cl,n

mov ch,00h

apple : mov dl,'\*'

mov ah,02h

int 21h

mov dl,' '

mov ah,02h

int 21h

loop apple

endm

newline macro

mov dl,0dh

mov ah,02h

int 21h

mov dl,0ah

mov ah,02h

int 21h

mov cx,si

s : mov dl,' '

mov ah,02h

int 21h

dec cx

jcxz k

jmp s

k : inc si

endm

data segment

no db 05h

orange db 00h

data ends

code segment

assume cs: code,ds: data

start : mov ax,data

mov ds,ax

mov bl,no

mov si,0001h

grape : mov orange,bl

star orange

newline

dec bl

jnz grape

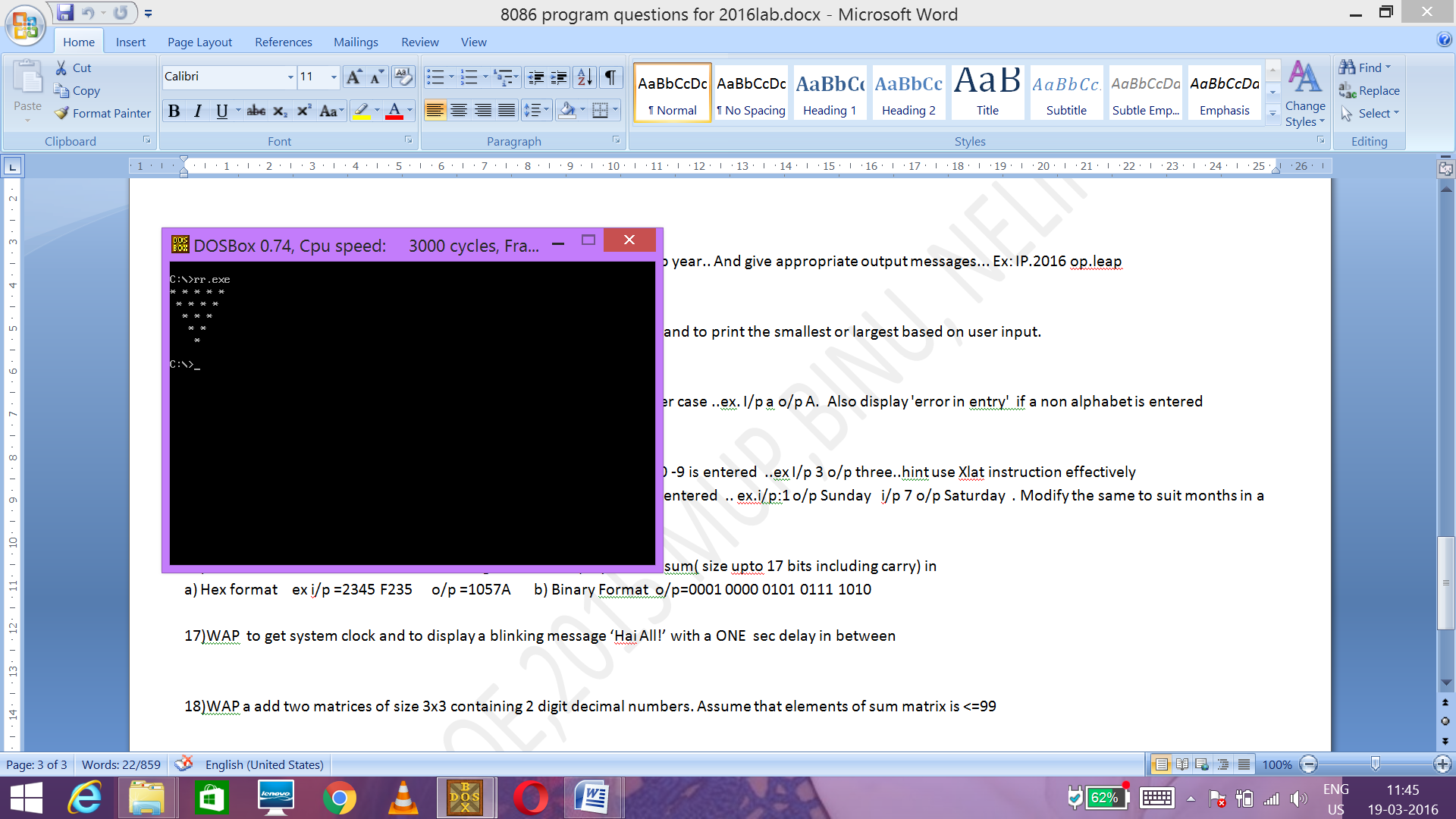
mov ah,04ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**



**PROGRAM NO.5(A)**

**With the help of a neat flow chart and necessary comments write an 8086 ALP to find the sum of 10 two digit (packed)BCD numbers(assume answer less than 99)**

**Ex i/p: 11h,01h,02h,03h,04h,05h,06h,07h,08h,09h o/p :56) ( hint use DAA)**

ASSUME CS:CODE,DS:DATA

DATA SEGMENT

NO DB 11H,01H,02H,03H,04H,05H,06H,07H,08H,09H,00H

DATA ENDS

CODE SEGMENT

START:MOV AX,DATA

MOV DS,AX

LEA BX,NO ;LOADING THE BASE ADDRESS OF NOS

MOV CX,10 ;LOADING THE COUNTER

MOV AL,NO

AGAIN:INC BX ;LOOP TO CALCULATE THE SUM

MOV AH,[BX]

ADD AL,AH

DAA

DEC CX

JCXZ NEXT

JMP AGAIN

NEXT:INC BX

MOV [BX],AL ;STORES THE SUM IN DATA SEGMENT

DISPLAY:MOV DH,AL ;DISPLAYS THE SUM

MOV CX,4H

SHR DH,CL

MOV DL,DH

ADD DL,30H

MOV AH,02H

INT 21H

MOV DL,[BX]

AND DL,0FH

ADD DL,30H

MOV AH,02H

INT 21H

STOP:MOV AH,4CH

INT 21H

CODE ENDS

END START

**Checking the o/p**

INPUT 11H,1H,2H,3H,4H,5H,6H,7H,8H,9H,0H

OTPUT SUM=56

**PROGRAM NO.5(B)**

**Find the average**

ASSUME CS:CODE,DS:DATA

DATA SEGMENT

NO DB 11H,01H,02H,03H,04H,05H,06H,07H,08H,09H,00H

average1 db 0ah,0dh,"average $"

sum1 db 0ah,0dh,"sum $"

DATA ENDS

CODE SEGMENT

START:MOV AX,DATA

MOV DS,AX

lea dx,sum1

mov ah,09h

int 21h

LEA BX,NO ;LOADING THE BASE ADDRESS OF NO

MOV CX,10 ;LOADING THE COUNTER

MOV AL,NO

AGAIN:INC BX ;CALCULATING THE SUM

MOV AH,[BX]

ADD AL,AH

DAA

DEC CX

JCXZ NEXT

JMP AGAIN

NEXT:INC BX

MOV [BX],AL

DISPLAY:MOV DH,AL ;DISPLAYING THE SUM

MOV CX,4H

SHR DH,CL

MOV DL,DH

ADD DL,30H

MOV AH,02H

INT 21H

MOV DL,[BX]

AND DL,0FH

ADD DL,30H

MOV AH,02H

INT 21H

lea dx,average1 ;DISPLAYING THE AVERAGE

mov ah,09h

int 21h

mov ah,[bx]

mov cx,4h

shr ah,cl

mov dl,ah

add dl,30h

mov ah,02h

int 21h

mov dl,"."

mov ah,02h

int 21h

mov ah,[bx]

and ah,0fh

mov dl,ah

add dl,30h

mov ah,02h

int 21h

STOP:MOV AH,4CH

INT 21H

CODE ENDS

END START

**Checking the o/p**

INPUT 11H,1H,2H,3H,4H,5H,6H,7H,8H,9H,0H

OTPUT SUM=56

AVERAGE=5.6

**PROGRAM NO.6(A)**

**With the help of a neat flow chart and necessary comments write an 8086 ALP to find the square of nibbles of a byte. Store the 8 bit results in data segment**

**Ex: i/p : 34h o/p 09h,10h**

ASSUME CS:CODE,DS:DATA

DATA SEGMENT

NO DB 34H,00H,00H

DATA ENDS

CODE SEGMENT

START:MOV AX,DATA

MOV DS,AX

MOV DH,NO ;MOVING THE NO IN DH

AND DH,0FH ;SEPARATING OUT THE NIBBLES;

MOV AL,DH

MUL DH ;OBTAINING THE SQUARE OF NIBBLES

LEA BX,NO

INC BX

MOV [BX],AX ;STORING THE RESULT

MOV DH,NO

MOV CX,4H

SHR DH,CL

MOV AL,DH

MUL DH

INC BX

MOV [BX],AX

MOV AH,4CH

INT 21H

CODE ENDS

END START

**Checking the o/p**

INPUT 34H

OUTPUT 09H,10H

**PROGRAM NO 6(B)**

**Modify the program to store square of an 8 and 16 bit number**

ASSUME CS:CODE,DS:DATA

DATA SEGMENT

NO DB 25H

RES DW 0010H

NO2 DW 9876H

RES2 DW 0000H,0000H

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

MOV AH,NO ;FINDING THE SQUARE OF 8 BIT NO

MOV AL,NO

LEA BX,RES

MUL AH

MOV [BX],AH

INC BX

MOV [BX],AL ;STORING THE RESULT

LEA BX,NO2 ;ACCESSING 16 BIT NO

MOV CL,[BX]

INC BX

MOV CH,[BX]

MOV AL,CL

MOV AH,CH

MUL CX ;FINDING ITS SQUARE

LEA BX,RES2 ;STORING THE RESULT

MOV [BX],DH

INC BX

MOV [BX],DL

INC BX

MOV [BX],AH

INC BX

MOV [BX],AL

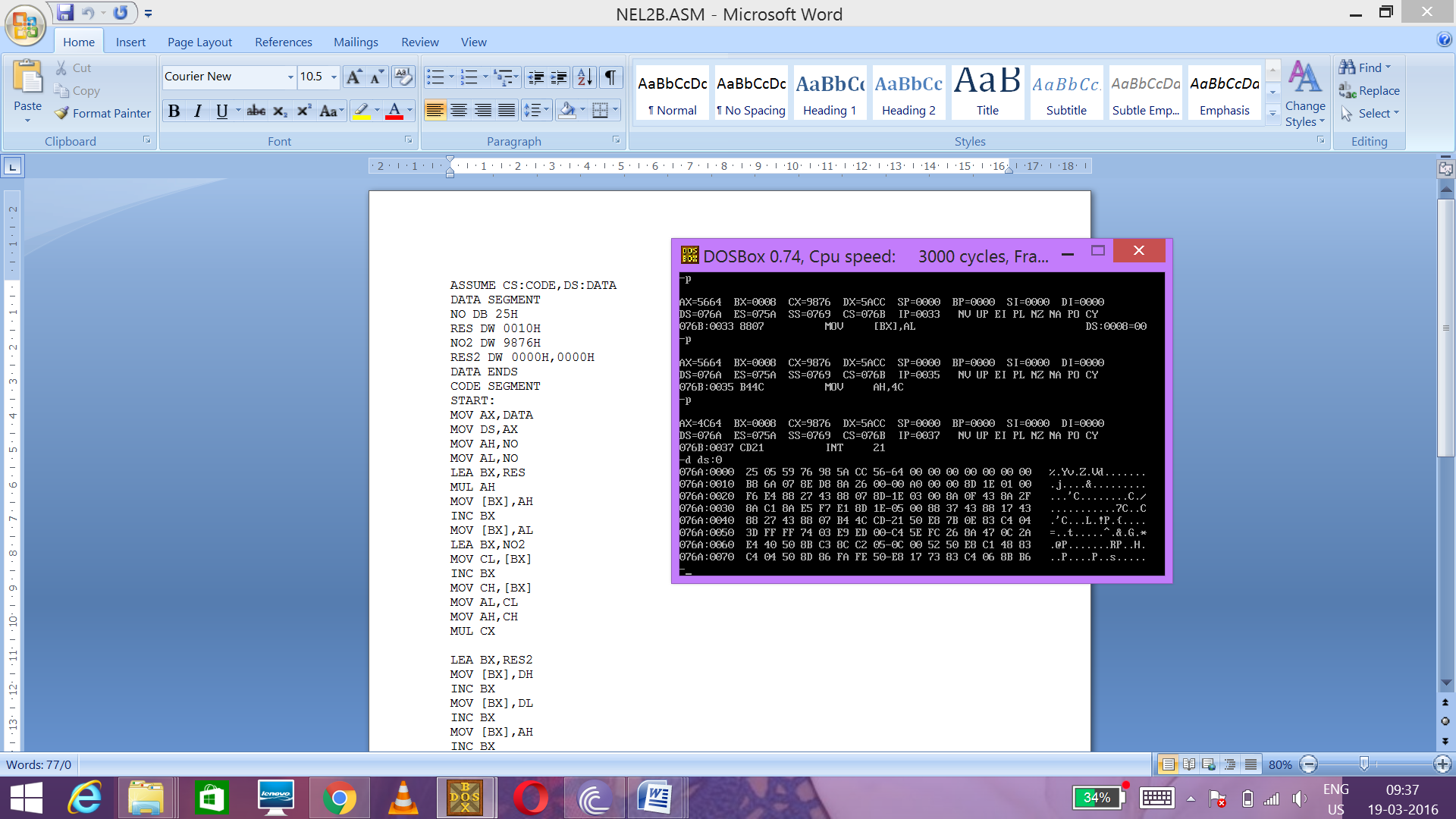
MOV AH,4CH

INT 21H

CODE ENDS

END START

**Checking the o/p**



**PROGRAM NO.7(A)**

**With the help of a neat flow chart and necessary comments write an 8086 ALP to find whether the nibbles of given 8 bit number is odd or not.**

**Ex: i/p :56 o/p Upper nibble is odd Lower nibble is even**

ASSUME CS:CODE,DS:DATA

DATA SEGMENT

NUM1 DB 87H

YN DB 0AH,0DH, "YOUR NO $"

UO DB 0AH,0DH, "UPPER NIBBLE IS ODD$"

UE DB 0AH,0DH, "UPPER NIBBLE IS EVEN$"

LOO DB 0AH,0DH, "LOWER NIBBLE IS ODD$"

LOE DB 0AH,0DH, "LOWER NIBBLE IS EVEN$"

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

LEA DX,YN

MOV AH,09H

INT 21H

MOV DL,NUM1 ;ACCESSING AND DISPLAYING THE NUMBER

MOV CX,4H

SHR DL,CL

ADD DL,30H

MOV AH,02H

INT 21H

MOV DL,NUM1

AND DL,0FH

ADD DL,30H

MOV AH,02H

INT 21H

MOV CX,02H ;CHECKING THE UPPER NIBBLE FOR ODD OR EVEN

MOV AH,NUM1

AND AH,10H

CMP AH,00H

JE EVN

JMP ODD

NN: ;CHECKING THE LOWER NIBBLE FOR ODD OR EVEN

MOV AL,NUM1

AND AL,01H

CMP AL,00H

JE EVN

JMP ODD

EVN: ;DISPLAYING THE NIBBLE EVEN

CMP CX,1H

JE LL

LEA DX,UE

JMP PP

LL:

LEA DX,LOE

PP:

MOV AH,09H

INT 21H

LOOP NN

JMP STOP

ODD: ;DISPLAYING THE NIBBLE IS ODD

CMP CX,1H

JE UU

LEA DX,UO

JMP SSD

UU:

LEA DX,LOO

SSD:

MOV AH,09H

INT 21H

LOOP NN

JMP STOP

STOP:

MOV AH,4CH

INT 21H

CODE ENDS

END START

**CHECKING THE OUTPUT**

INPUT 87H

OUTPUT UPPER NIBBLE IS EVEN

LOWER NIBBLE IS ODD

**PROGRAM NO.7(B)**

**Modify the program to find the number of odd numbers in an array of 16 elements.**

assume cs:code,ds:data

data segment

no db 00h,01h,02h,03h,04h,05h,03h,07h,08h,09h,10h,11h,12h,13h,14h,15h

dds db 0ah,0dh, "no of odd nos $"

data ends

code segment

start:

mov ax,data

mov ds,ax

lea dx,dds

mov ah,09h

int 21h

mov cx,10h ;LOADING THE COUNTER

lea bx,no ;LOADING THE BASE ADDRESS OF NO

dec bx

mov si,00h

again:inc bx ;CHECKING FOR ODD OR EVEN

mov al,[bx]

shr al,1h

jc label1

jmp label2

label1:inc si ;COUNTER FOR ODD NOS

loop again

jmp next

label2:loop again

next:MOV ax,si ;DISPLAYING THE ANSWER

add al,00h

daa

mov dl,al

MOV CX,4H

SHR DL,CL

ADD DL,30H

MOV AH,02H

INT 21H

MOV ax,si

add al,00h

daa

mov dl,al

AND DL,0FH

ADD DL,30H

MOV AH,02H

INT 21H

mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**

INPUT 00h,01h,02h,03h,04h,05h,03h,07h,08h,09h,10h,11h,12h,13h,14h,15h

OUTPUT 9

**PROGRAM NO.8(A)**

With the help of a neat flow chart and necessary comments write an 8086 ALP to find the largest of 3 data stored in memory. Store the answer in data segment

Ex: i/p 23h,88h,45h o/p :88h

assume cs:code,ds:data

data segment

no db 23h,88h,45h

res db 00h

data ends

code segment

start:

mov ax,data

mov ds,ax

LEA BX,NO ;LOADING THE BASE ADDRESS OF NO

mov cl,[bx] ;MOVING ALL THE NOS INTO THE REGISTERS

inc bx

mov ch,[bx]

inc bx

mov al,[bx]

cmp cl,ch ;COMPARING THE NOS.

jb label2

cmp cl,al

jb label4

mov res,cl ;STORING THE RESULT IF NO IN CL IS GREATEST

jmp stop

label4:

mov res,al ;STORING THE RESULT IF NO IN AL IS GREATEST

jmp stop

label2:

cmp ch,al

jb label3

mov res,ch ;STORING THE RESULT IF NO IN CH IS GREATEST

jmp stop

label3:

mov res,al

stop:

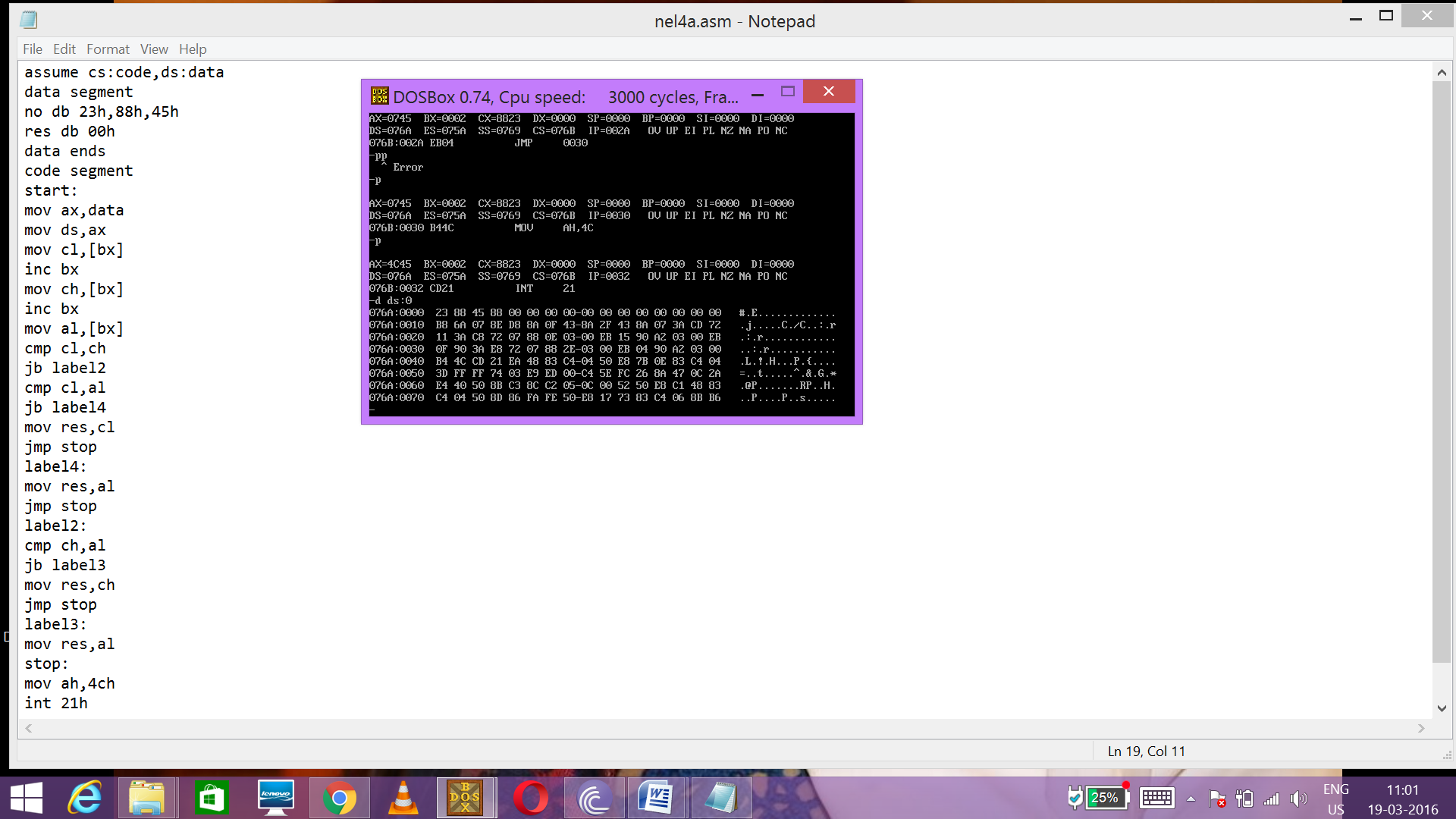
mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**



**Program 8(b)**

**Wap to find the largest of n data**

assume cs:code,ds:data

data segment

no db 10 dup(0)

msg db "how many nos ??$"

msg2 db 0ah,0dh, "enter the nos$"

no3 db 00h

res db 0ah,0dh, "largest no is$"

data ends

code segment

start:

mov ax,data

mov ds,ax

mov cx,00h

lea dx,msg

mov ah,09h

int 21h

mov ah,01h

int 21h

and al,0fh

mov cl,al

mov no3,al

lea dx,msg2

mov ah,09h

int 21h

lea bx,no

inn:

mov ah,01h

int 21h

mov [bx],al

inc bx

dec cx

jcxz apple

jmp inn

apple:

lea bx,no

mov al,[bx]

mov cl,no3

nextno:

inc bx

dec cx

cmp al,[bx]

jb next

loop nextno

jmp print

next:

mov al,[bx]

jcxz print

jmp nextno

print:

lea dx,res

mov ah,09h

int 21h

mov dl,al

mov ah,02h

int 21h

stop:

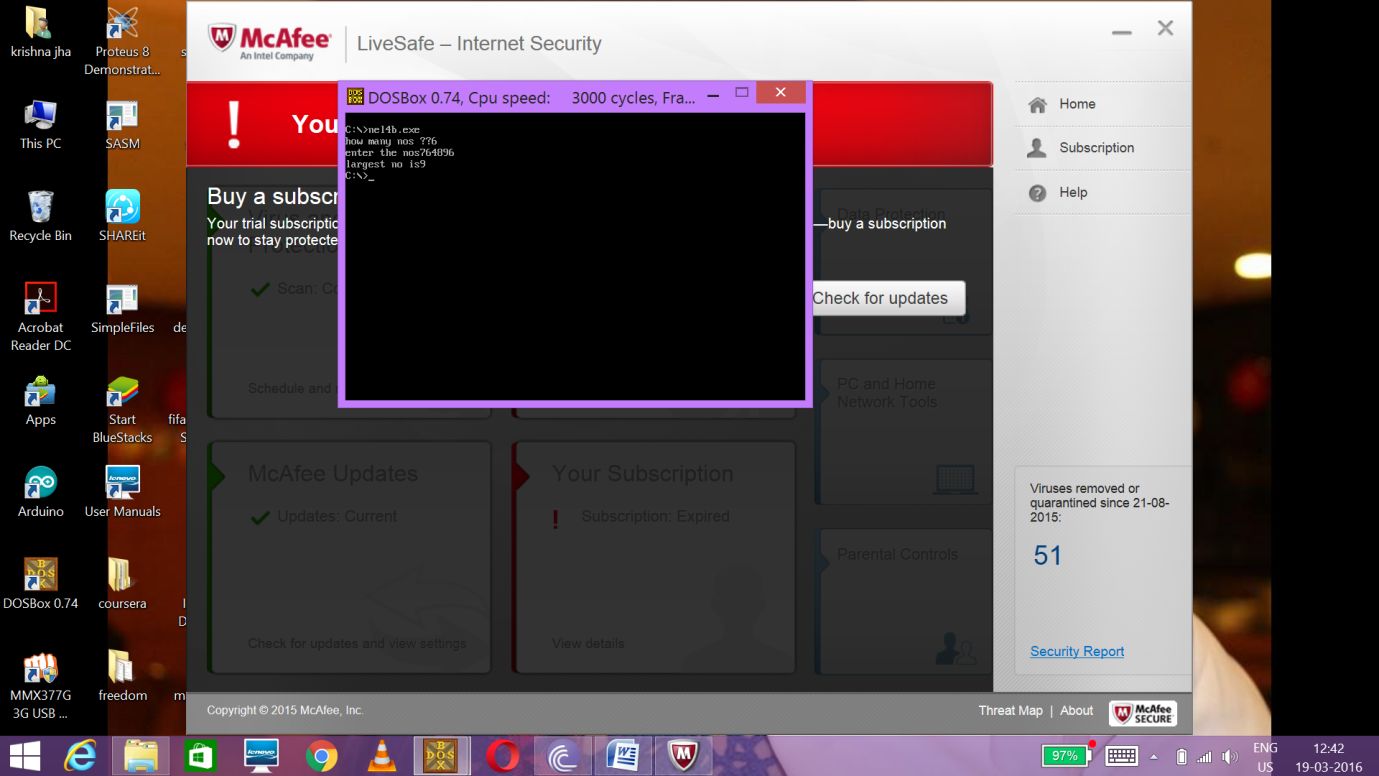
mov ah,4ch

int 21h

code ends

end start

**checking the output**



**PROGRAM NO 9(A)**

With the help of a neat flow chart and necessary comments write an 8086 ALP to unpack the digits of a 2 digit BCD number and to find the sum of digits. Store all answers in data segment. Display them also.

Ex: i/p 56h o/p 05 06 11)

assume cs:code,ds:data

data segment

no db 56h

un db 0ah,0dh,"upper nibble $"

ln db 0ah,0dh,"lower nibble $"

sum db 0ah,0dh, "sum $"

ans db 00,00h,00h

data ends

code segment

start:

mov ax,data

mov ds,ax

lea dx,un

mov ah,09h

int 21h

mov dl,no ;DISPLAYING THE UPPER NIBBLE

mov cl,4h

shr dl,cl

mov ans,dl

add dl,30h

mov ah,02h

int 21h

lea dx,ln

mov ah,09h

int 21h

mov dl,no ;DISPLAYING THE LOWER NIBBLE

and dl,0fh

mov ans+1,dl

add dl,30h

mov ah,02h

int 21h

lea dx,sum

mov ah,09h

int 21h

mov bh,ans

mov al,ans+1

add al,bh ;ADDING THE NIBBLES

daa

mov ans+2,al

mov dl,ans+2 ;DISPLAYING THE SUM

shr dl,cl

add dl,30h

mov ah,02h

int 21h

mov dl,ans+2

and dl,0fh

add dl,30h

mov ah,02h

int 21h

mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**

INPUT 56H

OUTPUT UPPER NIBBLE:5

LOWER NIBBLE:6

SUM:11

**PROGRAM NO.9(b)**

**Find whether the given 2 digit BCD number (<=90) is a multiple of 9. If yes store FFH at 5000H else store 00H**

assume cs:code,ds:data

data segment

no db 80h

data ends

code segment

start:

mov ax,data

mov ds,ax

mov bh,no

and bh,0fh

mov bl,no

mov cl,4h

shr bl,cl

add bh,bl

cmp bh,9h

jE label1

jnE label2

label1:

mov SI,5000h

MOV BH,0FFH

mov [SI],BH

jmp stop

label2:

mov SI,5000h

MOV BH,00H

mov [SI],BH

stop:mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**



**PROGRAM NO.10(A)**

With the help of a neat flow chart and necessary comments write an 8086 ALP to reverse the nibbles of your class roll number and calculate the sum of the resulting numbers.

Ex: i/p 06 o/p Reversed number is 60 and sum is 66)

assume cs:code,ds:data

data segment

MRN DB 0AH,0DH,"MY ROLL NUMBER$"

REV DB 0AH,0DH,"REV ROLL NUMBER$"

SUM DB 0AH,0DH, "SUM=$"

no db 20h,00h,00h

data ends

code segment

start:

mov ax,data

mov ds,ax

LEA DX,MRN

MOV AH,09H

INT 21H

MOV DL,NO ;PRINTING THE ROLL NO

MOV CX,4H

SHR DL,CL

ADD DL,30H

MOV AH,02H

INT 21H

MOV DL,NO

AND DL,0FH

ADD DL,30H

MOV AH,02H

INT 21H

LEA DX,REV

MOV AH,09H

INT 21H

mov dl,no ;REVERSING THE DIGITS

mov cl,4h

ror dl,cl

mov no+1,dl

mov al,no

add al,dl ;FINDING THE SUM

daa

mov no+2,al

MOV DL,NO+1 ;DISPLAYING THE REVERSED NO

MOV CX,4H

SHR DL,CL

ADD DL,30H

MOV AH,02H

INT 21H

MOV DL,NO+1

AND DL,0FH

ADD DL,30H

MOV AH,02H

INT 21H

LEA DX,SUM ;DISPLAYING THE SUM

MOV AH,09H

INT 21H

MOV DL,NO+2

MOV CX,4H

SHR DL,CL

ADD DL,30H

MOV AH,02H

INT 21H

MOV DL,NO+2

AND DL,0FH

ADD DL,30H

MOV AH,02H

INT 21H

mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**

INPUT:20

OUTPUT:MY ROLL NO 20

REV ROLL NO 02

SUM 22

PROGRAM NO 10(b)

Modify the same to suit your university reg no stored in data segment

assume cs:code,ds:data

data segment

myroll db 0ah,0dh,"my roll$"

str db "13150027$"

rev db 10 dup("$")

revstr db 0ah,0dh, "rev roll$"

data ends

code segment

start:

mov ax,data

mov ds,ax

lea dx,myroll

mov ah,09h

int 21h

lea bx,str ;DISPLAYING ORIGINAL ROLL NO

mov cx,8h

print:

mov dx,[bx]

mov ah,02h

int 21h

inc bx

loop print

lea dx,revstr ;REVERSING AND DISPLAYING THE REVERSED ROLL NO

mov ah,09h

int 21h

lea bx,str+7 ;DISPLAYING THE REVERSED ROLL NO.,ONE DIGIT AT A TIME

lea si,rev

mov cx,8h

again:

mov dx,[bx]

mov ah,02h

int 21h

mov [si],dx

inc si

dec bx

loop again

mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**

MY ROLL 13150027

REV ROLL 72005131

**PROGRAM NO 11(a)**

**With the help of a neat flow chart and necessary comments write an 8086 ALP to find the largest of 2 nibbles of a byte stored in a memory location**

**Ex: i/p = 45; o/p Largest nibble is 4)**

assume cs:code,ds:data

data segment

no db 54h

YN DB 0AH,0DH, "YOUR NUMBER$"

msg db 0AH,0DH, "largest nibble is$"

data ends

code segment

start:

mov ax,data

mov ds,ax

LEA DX,YN

MOV AH,09H

INT 21H

MOV DL,NO ;DISPLAYING THE ORIGINAL NO

MOV CX,4H

SHR DL,CL

ADD DL,30H

MOV AH,02H

INT 21H

MOV DL,NO

AND DL,0FH

ADD DL,30H

MOV AH,02H

INT 21H

mov bl,no

mov cl,4h

shr bl,cl

mov bh,no ;STORING THE UPPER NIBBLE

and bh,0fh ;STORING THE LOWER NIBBLE

cmp bh,dl ;COMPARING THE TWO NIBBLES

ja lsb

jmp msb

lsb:lea dx,msg ;DISPLAYING THE LARGEST NO.

mov ah,09h

int 21h

mov dl,bh

add dl,30h

mov ah,02h

int 21h

jmp stop

msb:lea dx,msg

mov ah,09h

int 21h

mov dl,bl

add dl,30h

mov ah,02h

int 21h

stop:mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**

YOUR NUMBER 54H

LARGEST NUMBER IS 5

**PROGRAM NO. 11(B)**

**Modify the program to find the largest of 3 bytes stored in data segment**

assume cs:code,ds:data

data segment

no db 98h,99h,22h

res db 00h

data ends

code segment

start:

mov ax,data

mov ds,ax

mov cl,[bx] ; STORING THE FIRST NO IN REGISTER

inc bx

mov ch,[bx] ;STORING THE SECOND NO IN REGISTER

inc bx

mov al,[bx] ;STORING THE SECONGD NO IN REGISTER

cmp cl,ch

jb label2

cmp cl,al ; COMPARING THE NOS

jb label4

mov res,cl

jmp stop

label4:

mov res,al

jmp stop

label2:

cmp ch,al

jb label3

mov res,ch

jmp stop

label3:

mov res,al

stop:

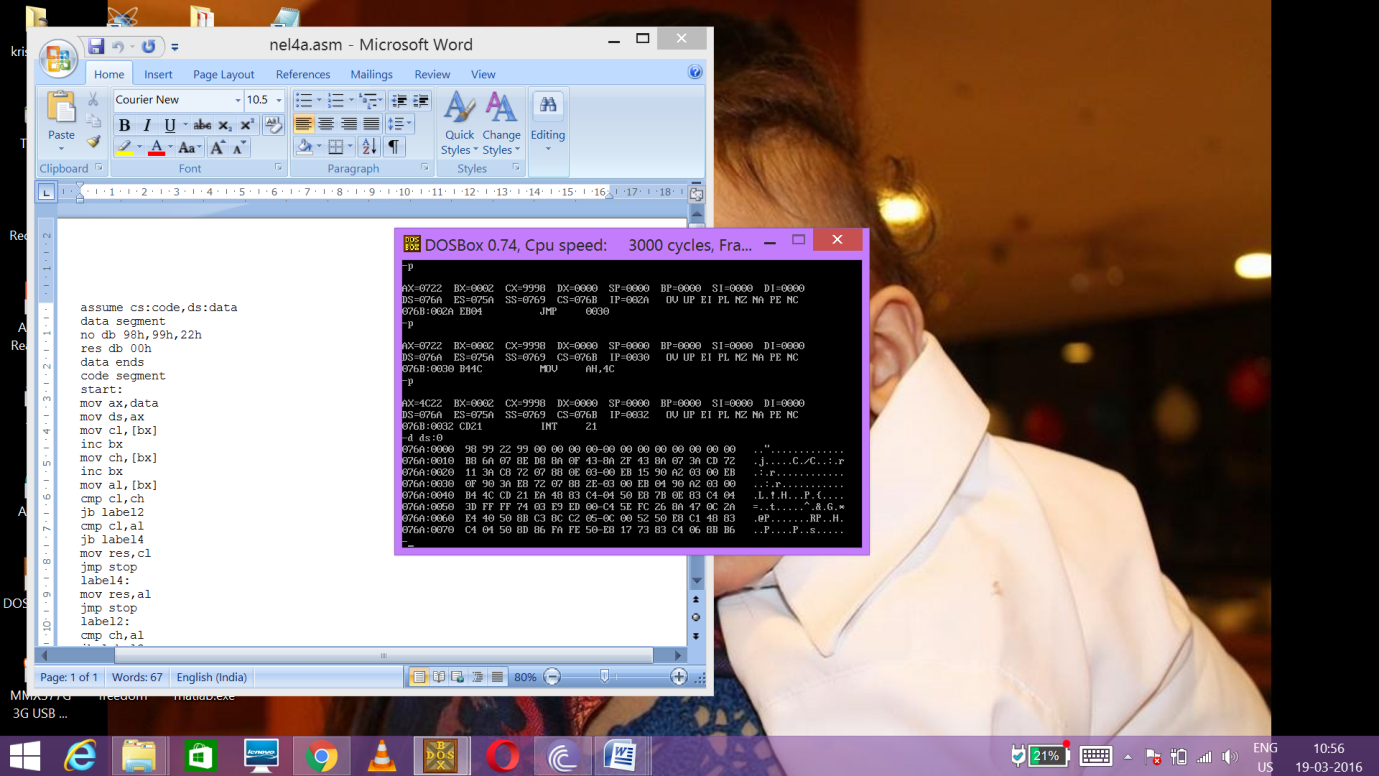
mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**



PROGRAM N0 12(a)

With the help of a neat flow chart and necessary comments write an 8086 ALP to find the sum of bits in a byte stored in memory. Also display its parity.

Ex: i/p 89 o/p sum is 3, parity is odd)

assume cs:code,ds:data

data segment

no db 55h

msg db 0ah,0dh, "even parity$"

msg2 db 0ah,0dh, "odd parity$"

SUMD DB 0AH,0DH, "SUM OF BITS$"

data ends

code segment

start:

mov ax,data

mov ds,ax

mov dh,no ;CHECKING THE PARITY

add dh,00h

jpe even1

jmp odd

even1:lea dx,msg

mov ah,09h

int 21h

jmp proceed

odd:lea dx,msg2

mov ah,09h

int 21h

proceed:xor ah,ah

mov al,no

mov cx,8h

again:ror al,1h ;FINDING THE SUM OF NIBBLES

adc ah,00h

loop again

MOV CL,AH

LEA DX,SUMD

MOV AH,09H

INT 21H

mov dl,CL ;DISPLAYING THE SUM

add dl,30h

mov ah,02h

int 21h

mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**

INPUT 55H

OUTPUT

ODD PARITY

SUM OF NIBBLES 4

PROGRAM N0 12(b)

Modify the program to find the modulo 2 sum of even index bits.

assume cs:code,ds:data

data segment

yn db 0ah,0dh, "your number$"

res db 0ah,0dh, "result$"

no db 50h

data ends

code segment

start:mov ax,data

mov ds,ax

lea dx,yn

mov ah,09h

int 21h

mov dl,no

mov cx,4h

shr dl,cl

add dl,30h

mov ah,02h

int 21h

mov dl,no

and dl,0fh

add dl,30h

mov ah,02h

int 21h

lea dx,res

mov ah,09h

int 21h

mov al,no

xor ah,ah

mov cx,4h

again:ror al,1h

adc ah,00h

ror al,1h

loop again

mov dl,ah

add dl,30h

mov ah,02h

int 21h

MOV AH,4CH

INT 21H

code ends

end start

**CHECKING THE OUTPUT**

INPUT 50H

OUTPUT

RESULT 2

PROGRAM NO 13 (a)

With the help of a neat flow chart and necessary comments write an 8086 ALP to find the number of ones and zeros in an 8 bit data

Ex i/p : AB o/p 5 ones 3 zeros)

assume cs:code,ds:data

data segment

no db 55h

YN DB 0AH,0DH, "YOUR NO$"

ODDC DB 0AH,0DH, "ODD COUNT$"

EVENC DB 0AH,0DH, "EVEN COUNT$"

data ends

code segment

start:

mov ax,data

mov ds,ax

LEA DX,YN

MOV AH,09H

INT 21H

MOV DL,NO

MOV CX,4H

SHR DL,CL

ADD DL,30H

MOV AH,02H

INT 21H

MOV DL,NO

AND DL,0FH

ADD DL,30H

MOV AH,02H

INT 21H

mov dh,no

mov cx,8h

mov bx,0h

firse:

shr dh,1h

jc even1

jnc odd1

even1:

inc bh

LOOP FIRSE

JMP BAHAR

odd1:

inc bl

LOOP FIRSE

bahar:

LEA DX,ODDC

MOV AH,09H

INT 21H

MOV DL,BH

ADD DL,30H

MOV AH,02H

INT 21H

LEA DX,EVENC

MOV AH,09H

INT 21H

MOV DL,BL

ADD DL,30H

MOV AH,02H

INT 21H

mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**

INPUT 55H

OUTPUT

NO OF 1’S 4

NO OF 0’S 4

PROGRAM NO 13(b)

Modify the program to find the gray code of the given number

assume cs:code,ds:data

data segment

no db 76h

linefeed db 13,10,("$")

data ends

code segment

START:

mov ax,data

mov ds,ax

mov dh,no

mov cx,8h

again:shl dh,1h

jc print1

jnc print0

print1:mov dl,31h

mov ah,02h

int 21h

dec cx

jcxz newline

jmp again

print0:mov dl,30h

mov ah,02h

int 21h

dec cx

jcxz newline

jmp again

newline:mov dx,offset linefeed

mov ah,09h

int 21h

mov cx,8h

mov dh,no

shl dh,1h

jc apple

jmp grape

apple:

mov bh,1h

jmp gray

grape:

mov bh,00h

gray:mov bl,no

xor bl,dh

cmp bh,1h

jz hi

jmp bye

hi:stc

rcr bl,1h

jmp pp

bye:clc

rcr bl,1h

pp:shl bl,1h

jc prin1

jmp prin0

prin1:mov dl,31h

mov ah,02h

int 21h

dec cx

jcxz stop

jmp pp

prin0:mov dl,30h

mov ah,02h

int 21h

dec cx

jcxz stop

jmp pp

stop:mov ah,4ch

int 21h

code ends

end start

**CHEKING THE OUTPUT**

INPUT 76H

OUTPUT

NO 0111 0110

GRAY CODE 0100 1101

PROGRAM NO.14 (a)

With the help of a flow chart and necessary comments write an 8086 ALP to mask the higher nibble of all 10 nos stored in consecutive memory locations and to store them in another location

assume cs:code,ds:data

data segment

nos db 11h,23h,44h,55h,77h,88h,44h,65h,66h,77h

nib db 10 dup(0)

COUNT DB 00B

data ends

code segment

start:

mov ax,data

mov ds,ax

lea bx,nos

lea si,nib

mov CX,0Ah

next:

MOV COUNT,CL

MOV CL,4H

mov dl,[bx]

shr dl,CL

add dl,30h

mov [si],dl

inc bx

inc si

MOV CL,COUNT

LOOP NEXT

STOP:

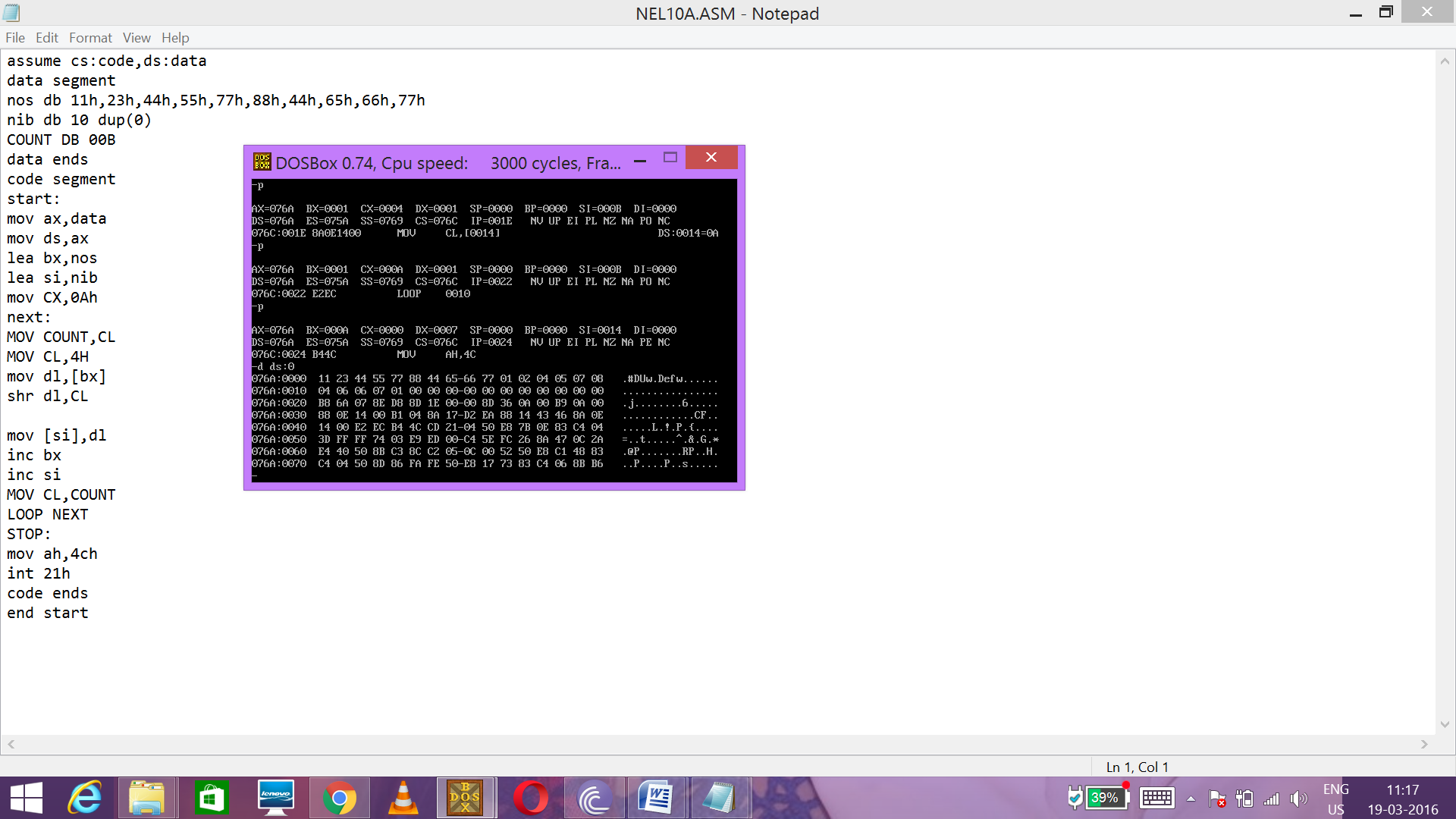
mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**



PROGRAM N0 14(b)

Find the decimal sum of all lower nibbles

assume cs:code,ds:data

data segment

nos db 44h,33h,66h,76h,88h,98h,21h,65h,65h,32h

sum db 00h

data ends

code segment

start:

mov ax,data

mov dl,00h

mov cx,10h

mov ds,ax

lea bx,nos

MOV AL,00H

again:

mov dh,[bx]

and dh,0fh

add Al,dh

DAA

MOV DL,AL

inc bx

LOOP AGAIN

NEXT:

mov sum,dl

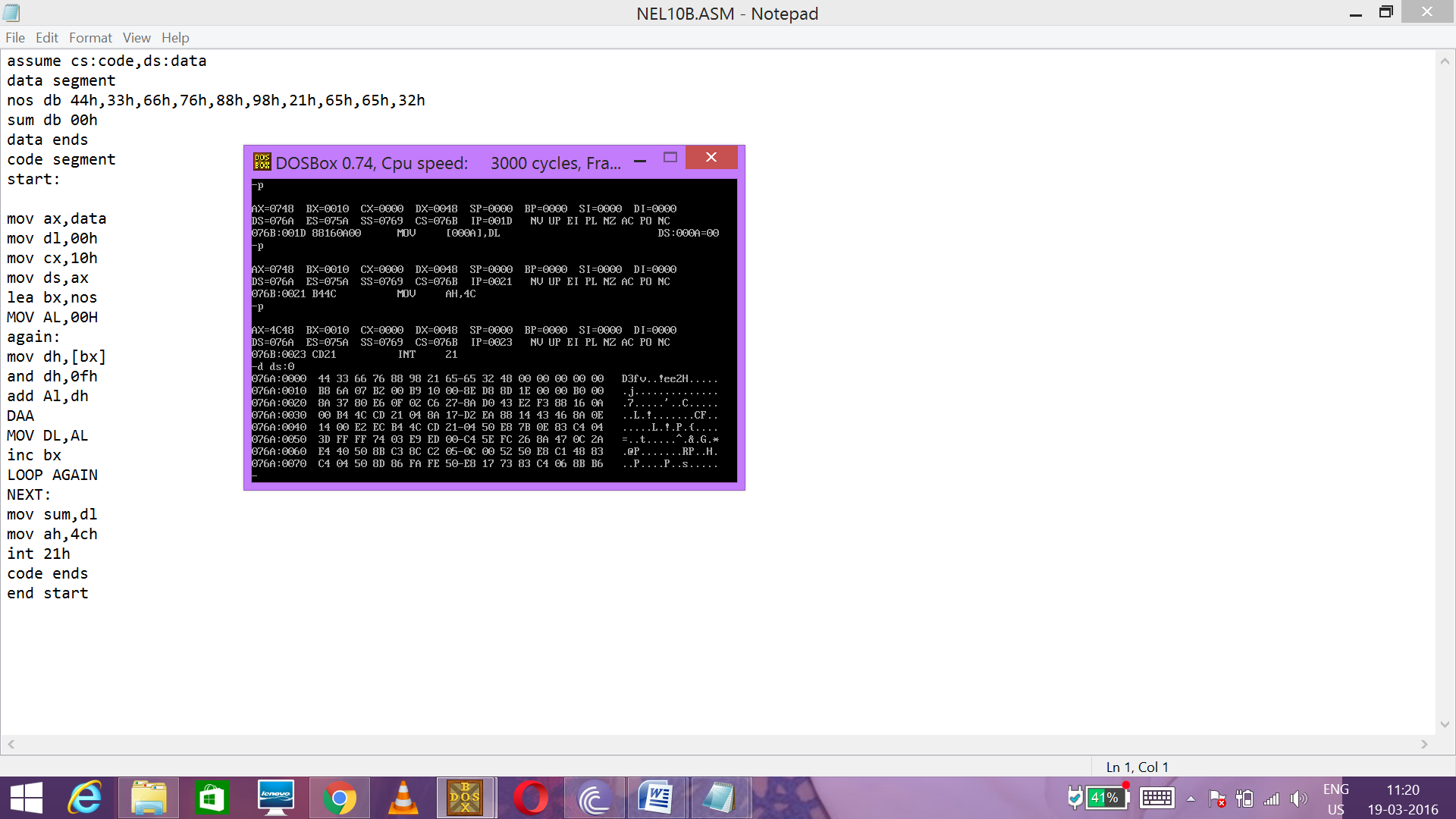
mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**



**PROGRAM NO 15**

**WAP to count and display all vowels in given sentence. The program must be case insensitive. Also give the total number of vowels(<=99d)**

assume cs:code,ds:data

data segment

str db 0ah,0dh,100 dup("$")

vow db "AEIOU$"

ip db 0ah,0dh,"enter the string $"

at db 0ah,0dh, "no of $"

c1 db 00h

d1 db 00h

TOT DB 0AH,0DH, "TOTAL NO OF VOWELS$"

TOTAL DB 00H

data ends

code segment

start:

mov ax,data

mov ds,ax

lea dx,ip

mov ah,09h

int 21h

lea dx,str

mov ah,0ah

int 21h

MOV DI,00H

mov bx,offset vow

next:

lea si,str+1

mov cx,00h

again:

inc si

mov al,00h

mov ah,[si]

AND AH,11011111B

xlatb

mov d1,al

cmp d1,"$"

je stop

cmp al,ah

jz count

cmp ah,0dh

jz display

jmp again

count:

inc cl

jmp again

display:

lea dx,at

mov ah,09h

int 21h

mov c1,cl

ADD CL,TOTAL

MOV TOTAL,CL

mov dl,al

mov ah,02h

int 21h

MOV DL,C1

MOV CL,4H

SHR DL,CL

ADD DL,30H

MOV AH,02H

INT 21H

MOV DL,C1

AND DL,0FH

ADD DL,30H

MOV AH,02H

INT 21H

inc bx

jmp next

stop:

LEA DX,TOT

MOV AH,09H

INT 21H

MOV DL,TOTAL

MOV CL,4H

SHR DL,CL

ADD DL,30H

MOV AH,02H

INT 21H

MOV DL,TOTAL

AND DL,0FH

ADD DL,30H

MOV AH,02H

INT 21H

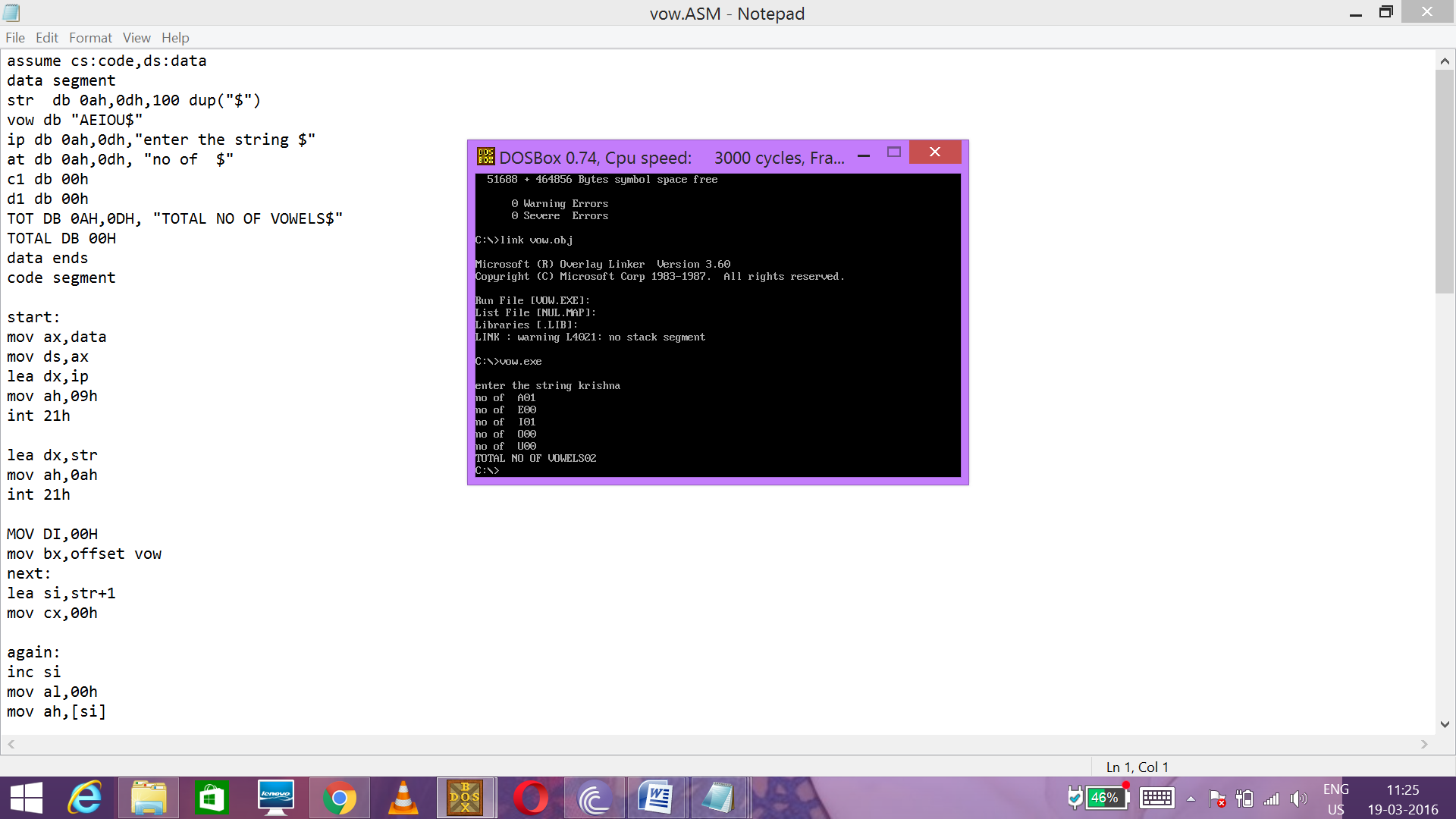
mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**



**PROGRAM NO.16**

**WAP to get system clock and to display a blinking message ‘Hai All!’ with a ONE sec delay in between**

data segment

msg1 db 0dh,'Hai! ALL$'

msg2 db 0dh,' \*$'

data ends

code segment

assume cs:code,ds:data

start:mov ax,data

mov ds,ax

mov cx, 9h

apple:

;; show message

lea dx,msg1

mov ah,09h

int 21h

;; get clock

push cx ; since

mov ah,2ch

int 21h

mov bl,dh

inc bl

;; delay

gettime1: mov ah,2ch

int 21h

cmp dh,bl

jnz gettime1

;; Blank the line

lea dx,msg2

mov ah,09h

int 21h

;; delay

mov ah,2ch

int 21h

mov bl,dh

inc bl

gettime2: mov ah,2ch

int 21h

cmp dh,bl

jnz gettime2

pop cx

loop apple

mov ah,4ch

int 21h

code ends

end start

**CHECKING THE OUTPUT**

Op: “ HAI ALL “ ;AFTER 1 SEC

Op: “ “

**PROGRAM NO. 17**

**WAP to read a valid alphabet from key board and to display its other case ..ex. I/p a o/p A. Also display 'error in entry' if a non alphabet is entered**

ASSUME CS:CODE,DS:DATA

DATA SEGMENT

IP DB 0AH,0DH, "ENTER A CHARACTER $"

IPE DB 00H

op db 0ah,0dh, "char $"

ERROR DB 0AH,0DH, "THAT'S NOT AN ALPHABET$"

DATA ENDS

CODE SEGMENT

START:

MOV AX,DATA

MOV DS,AX

LEA DX,IP

MOV AH,09H

INT 21H

MOV AH,01H

INT 21H

lea dx,op

mov ah,09h

int 21h

MOV IPE,AL

and al,01000000b

cmp al,40h

jne disp

mov al,ipe

AND AL,00100000B

CMP AL,20H

JE CUPPER

jmp clower

Cupper:

MOV DL,IPE

AND DL,11011111B

MOV AH,02H

INT 21H

JMP STOP

Clower:

MOV DL,IPE

ADD DL,20H

MOV AH,2H

INT 21H

jmp stop

disp:

lea dx,error

mov ah,09h

int 21h

STOP:

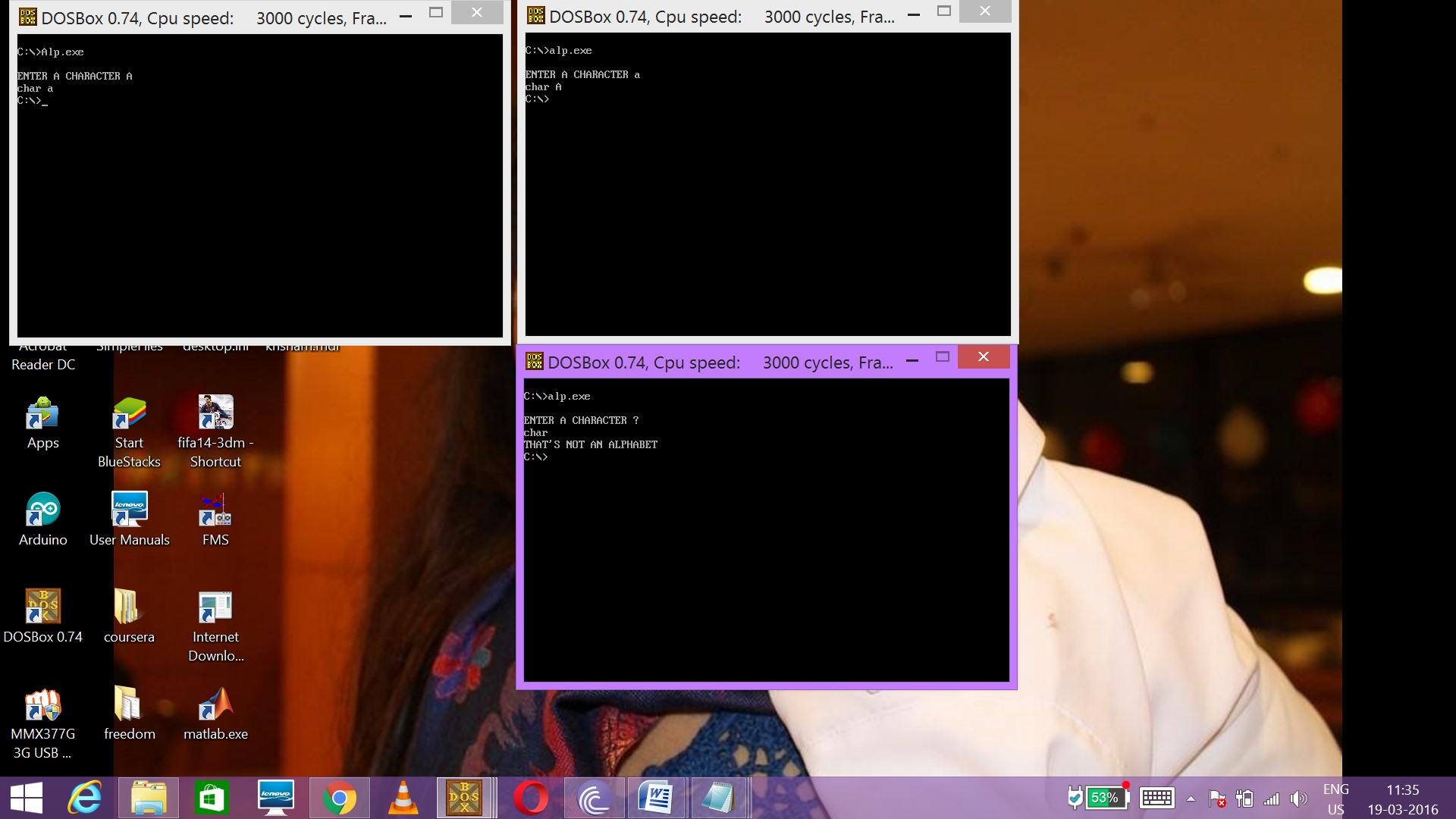
MOV AH,4CH

INT 21H

CODE ENDS

END START

**CHECKING THE OUTPUT**



**PROGRAM NO.18**

**WAP to check whether the year entered in the form yyyy is a leap year.. And give appropriate output messages... Ex: IP.2016 op.leap**

assume cs:code,ds:data

data segment

IP DB 0AH,0DH,"ENTER THE YEAR$"

year db 10 dup("$")

leap db 0ah, 0dh, "leap year$"

noleap db 0ah, 0dh, "not a leap year$"

data ends

code segment

start:

mov ax,data

mov ds,ax

LEA DX,IP

MOV AH,09H

INT 21H

lea dx,year

mov ah,0ah

int 21h

mov cl,year+5

mov ch,year+4

sub ch,30h

sub cl,30h

mov ax,cx

aad

jz msb

lsb:

mov bx,4h

div bl

cmp ah,00h

jz printl

jmp printn

msb:

mov ch,year+2

mov cl,year+3

sub ch,30h

sub cl,30h

mov ax,cx

aad

mov bx,4h

div bl

cmp ah,00h

jz printl

jmp printn

printl:

lea dx,leap

mov ah,09h

int 21h

jmp stop

printn:

lea dx,noleap

mov ah,09h

int 21h

stop:

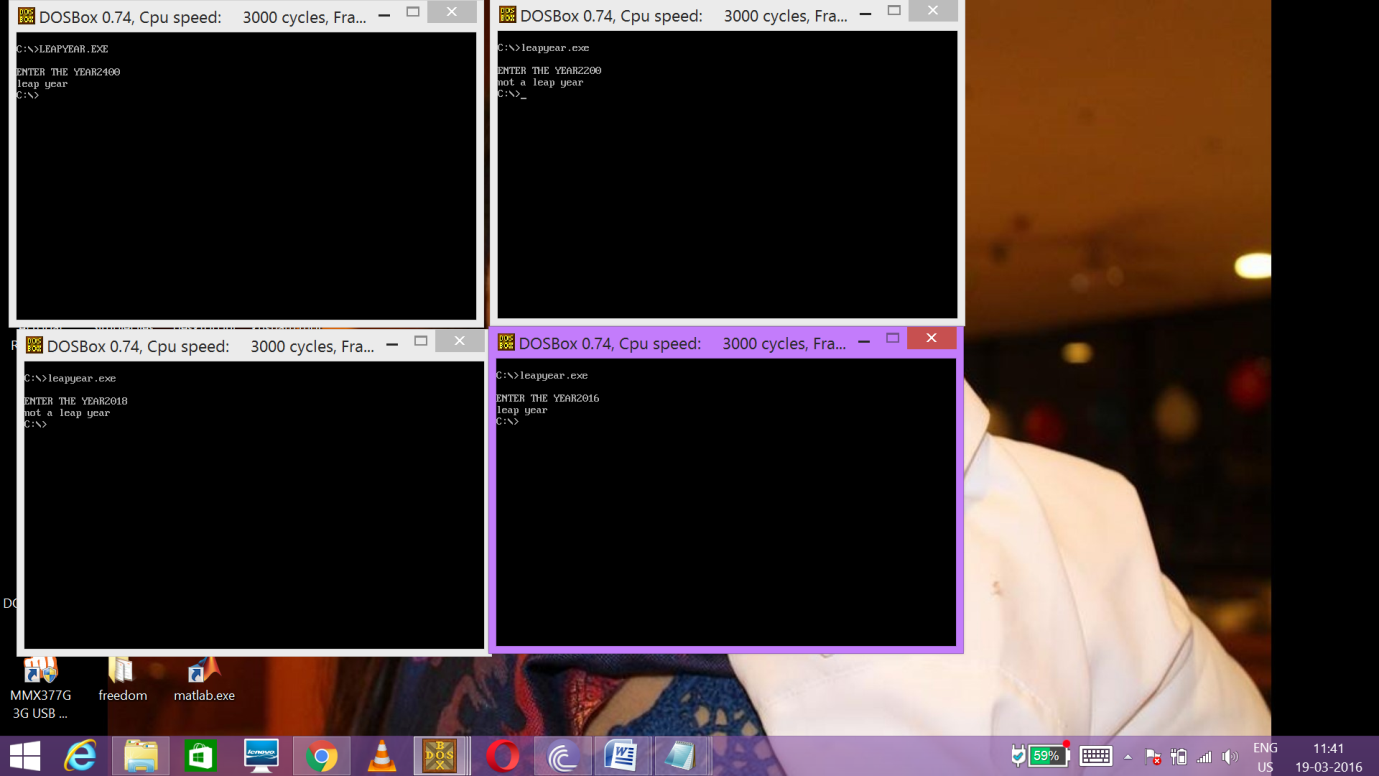
mov ah,4ch

int 21h

code ends

end start

CHECKING THE OUTPUT



**Program no 19**

**Wap to print the ap with first term, common difference and n taken as input from the user**

star macro n

mov ax,0000h

mov cx,0000h

lea si,str1+2

mov al,[si]

sub al,30h

mov cl,04h

ror al,cl

inc si

mov ch,[si]

sub ch,30h

or al,ch

mov n,al

endm

newline macro

mov dl,0dh

mov ah,02h

int 21h

mov dl,0ah

mov ah,02h

int 21h

endm

data segment

a1 db 00h

d1 db 00h

n1 db 00h

n2 db 36 dup(00h)

msg1 db "Enter your two digit first number",0dh,0ah,"$"

msg2 db "Enter the two digit common ratio",0dh,0ah,"$"

msg3 db "Enter the count to print ap",0dh,0ah,"$"

msg4 db "Your count is",0dh,0ah,"$"

str1 db 36 dup("$")

data ends

code segment

assume cs: code, ds: data

start : mov ax,data

mov ds,ax

lea dx,msg1

mov ah,09h

int 21h

lea dx,str1

mov ah,0ah

int 21h

newline

star a1

lea dx,msg2

mov ah,09h

int 21h

lea dx,str1

mov ah,0ah

int 21h

newline

star d1

lea dx,msg3

mov ah,09h

int 21h

lea dx,str1

mov ah,0ah

int 21h

newline

star n1

mov ax,0000h

mov al,n1

mov bx,0000h

mov bl,n1

mov cx,0004h

and al,0f0h

ror al,cl

mov cl,0ah

mul cl

and bl,0fh

add al,bl

mov n1,al

mov cx,0000h

mov cl,n1

mov ax,0000h

lea bx,n2

mov al,[a1]

mov [bx],al

again :

inc bx

add al,d1

daa

mov [bx],al

dec cx

jcxz stop

jmp again

stop :

newline

lea dx,msg4

mov ah,09h

int 21h

newline

lea bx,n2

mov cx,0000h

mov cl,n1

l1 :

mov ax,0000h

mov al,[bx]

and al,0f0h

ror al,01h

ror al,01h

ror al,01h

ror al,01h

add al,30h

mov dl,al

mov ah,02h

int 21h

mov al,[bx]

and al,0fh

add al,30h

mov dl,al

mov ah,02h

int 21h

newline

inc bx

dec cx

jcxz l2

jmp l1

l2 :

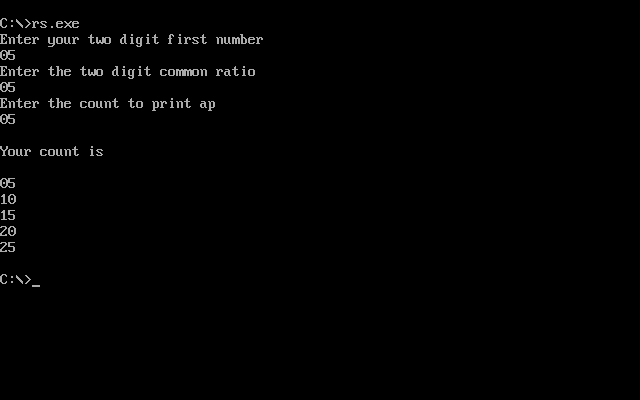
mov ah,04ch

int 21h

code ends

end start

**checking the output**



**III 8086 PROGRAM USING MICROPROCESSOR TRAINER KIT**

**PROGRAM :1**

**WRITE A PROGRAM TO ADD TWO 8 BIT NUMBERS**

|  |  |  |
| --- | --- | --- |
| **MEMORY ADDRESS** | **OPCODE** | **MNEMONICS** |
| **0400** | **BE** | **MOV SI,0500** |
| **0401** | **00** |  |
| **0402** | **05** |  |
| **0403** | **8A** | **MOV AL,[SI]** |
| **0404** | **04** |  |
| **0405** | **46** | **INC SI** |
| **0406** | **02** | **ADD AL,[SI]** |
| **0407** | **04** |  |
| **0408** | **46** | **INC SI** |
| **0409** | **88** | **MOV [SI],AL** |
| **040A** | **04** |  |
| **040B** | **F4** | **HLT** |

Checking the o/p

**INPUT OUTPUT**

**0500 – 85 0502 - 8C**

**0501 - 07**

**PROGRAM NO:2**

**WRITE A PROGRAM TO ADD TWO 16 BIT NUMBERS**

|  |  |  |
| --- | --- | --- |
| **MEMORY ADDRESS** | **OPCODE** | **MNEMONICS** |
| **0400** | **25** | **AND AX,0000** |
| **0401** | **00** |  |
| **0402** | **00** |  |
| **0403** | **BB** | **MOV BX,0600** |
| **0404** | **00** |  |
| **0405** | **06** |  |
| **0406** | **BE** | **MOV SI,0500** |
| **0407** | **00** |  |
| **0408** | **05** |  |
| **0409** | **BF** | **MOV DI,0550** |
| **040A** | **50** |  |
| **040B** | **05** |  |
| **040C** | **8B** | **MOV AX.[SI]** |
| **040D** | **04** |  |
| **040E** | **03** | **ADD AX,[DI]** |
| **040F** | **05** |  |
| **0410** | **89** | **MOV [BX],AX** |
| **0411** | **07** |  |
| **0412** | **B8** | **MOV AX,0000** |
| **0413** | **00** |  |
| **0414** | **00** |  |
| **0415** | **15** | **ADC AX,0000** |
| **0416** | **00** |  |
| **0417** | **00** |  |
| **0418** | **89** | **MOV[BX+2],AX** |
| **0419** | **47** |  |
| **041A** | **02** |  |
| **041B** | **F4** | **HLT** |

Checking the o/p

INPUT OUTPUT

0500-B5 0600-DF

0501-7A 0601-5F

0550-2A 0602-01

0551-E5

The program adds 7AB5+E52A and stores sum 015FDF in memory with LSB starting at location 0600

**PROGRAM NO:3**

**WRITE A PROGRAM TO SUBTRACT TWO 16 BIT NUMBERS**

|  |  |  |
| --- | --- | --- |
| **MEMORY**  **ADDRESS** | **OPCODE** | **MNEMONICS** |
| **0400** | **F8** | **CLC** |
| **0401** | **BB** | **MOV BX,0900** |
| **0402** | **00** |  |
| **0403** | **09** |  |
| **0404** | **BE** | **MOV SI,0700** |
| **0405** | **00** |  |
| **0406** | **07** |  |
| **0407** | **BF** | **MOV DI,0800** |
| **0408** | **00** |  |
| **0409** | **08** |  |
| **040A** | **8B** | **MOV AX.[SI]** |
| **040B** | **04** |  |
| **040C** | **1B** | **SBB AX,[DI]** |
| **040D** | **05** |  |
| **040E** | **89** | **MOV [BX],AX** |
| **040F** | **07** |  |
| **0410** | **F4** | **HLT** |

Checking the o/p

INPUT OUTPUT

0700 –18 0900- D8

0701 -08 0901- F7

0800 -40

0801 -10

The program performs [0700]-[0800] and stores as [ 0900];

0818-1040=F7D8

**IV. INTERFACING PROGRAMS**

**PROGRAM NO: 1**

**STEPPER MOTOR CONTROL INTERFACE**

|  |  |  |
| --- | --- | --- |
|  | MOV AL,80H | ;INITIALIZE PORT |
| OUT 46H,AL |  |
| LOOP2: | MOV CX,04 | ;LOOP COUNT |
|  | MOV BX,0500 | ;TABLE LOCATION |
| LOOP1: | MOV AL,[BX] |  |
|  | OUT 40H,AL | ;PLACE TO PORT |
|  | CALL DELAY | ;ROTATION DATA |
|  | INC BX | ;INC POINTER |
|  | LOOPNZ LOOP1 |  |
|  | JMP LOOP2 | ;CONTINUE |
| DELAY: | PUSH CX | ;SAVE CX |
|  | MOV CX, FFFFH | ;DELAY LOOP COUNT |
| HERE: | LOOPNZ HERE |  |
|  | POP CX | ;RETRIEVE CX |
|  | RET |  |

FORWARD

0500 : 01 02 04 08 ( CLOCKWISE PATTERN) REVERSE

0500: 08 04 02 01 (ANTICLOCKWISE PATTERN

Checking the o/p

The Stepper rotated continuously in clock wise direction

**PROGRAM NO:2**

1. **TO GENERATE RAMP WAVE USING DAC**

MOV AL, 80H

OUT 46H,AL; write control word

MOV AL,FFH

LOOP1: DEC AL

OUT 40H,AL ; send to both Port A and Port B

OUT 42H,AL

JMP LOOP1

1. **TO GENERATE SQUARE WAVE USING DAC**

MOV AL, 80H

OUT 46H,AL ; write control word

MOV AL,00H ; Output for 0volts

OUT 40H,AL

OUT 42H,AL

CALL DELAY ; hold 0V

MOV AL,FFH ; Output for 8volts

OUT 40H,AL

OUT 42H,AL

CALL DELAY ; hold 8V

JMP LOOP1 ; repeat infinite number of times

**Subroutine:**

|  |  |  |
| --- | --- | --- |
| DELAY: | PUSH CX | ;SAVE CX |
|  | MOV CX, FFFFH | ;DELAY LOOP COUNT |
| HERE: | LOOPNZ HERE |  |
|  | POP CX | ;RETRIEVE |
|  | RET |  |

Checking the o/p

Observed the following waveforms:

1. B)