Experiment 01

Learning Objective: Student should be able to apply Assembly Language Programing to enter and display 8 bit & 16 bits number

Tools: TASM/MASM

Theory:

Assembler Directives :-

This type of statements includes commands that are addressed to the assembler, such as:
Constant and variable definition.
Allocation of memory space and initialization of memory, and
Control of the assembly process

List of assembler directives
a. Data Allocation Directives
DB…………..define byte
DW…………define word (2 bytes)
DD…………define double word (4 bytes)
DQ…………define quadword (8 bytes)
DT…………..define tenbytes
EQU…………equate, assign numeric expression to a name

Examples:3

```
db        100 dup (?)      define 100 bytes, with no initial values for bytes
db       "Hello"       define 5 bytes, ASCII equivalent of "Hello".
maxint        equu      327673
count          equ      10 * 20       ; calculate a value (200)
```

ENDS…………used to indicate the end of the segment.
END…………. used to indicate the end of program.
PROC……......used to indicate the beginning of a procedure.
ENDP……….. .used to indicate the end of a program.
ENDM………. used to indicate the end of a program.
SEGMENT…...used to indicate the start of the segment.
TITLE…………used to indicate the title of the program.
EQU …………..used to give a name to some value or to a symbol. Each time the assembler finds the name in the program, it will replace the name with the value or symbol you given to that name.
ASSUME…….. Associates a logical segment to processor segment.
e.g. Example:
ASSUME CS:CODE ;

TASM COMMANDS:

C :/>cd foldername
C:/foldername>edit filename.asm

After this command executed in command prompt an editor window will open. Program should be typed in this window and saved. The program structure is given below.

Structure of Program:

```
.model tiny/small/medium/large
.Stack <some number>
.data
        ; Initialize data.
.code
.startup
        ; Program logic goes here.
.exit
End
```

To run the program, the following steps have to be followed:

C:/foldername>Tasm filename.asm

After this command is executed in command prompt if there are no errors in program regarding to syntax the assembler will generates an object module as discuss above.

C:/foldername>Tlink filename.obj

After verifying the program for correct syntax and the generated object files should be linked together. For this the above link command should be executed and it will give an EXE file if the model directive is small as discuss above.

C:/foldername>td filename.exe

After generating EXE file by the assembler it's the time to check the output. For this the above command is used and the execution of the program can be done in different ways. It is as shown below:

```
__ g    ; complete execution of program in single step.
__ t    ; Stepwise execution.
__d ds: starting address or ending address    ; To see data in memory locations
__p; Used to execute interrupt or procedure during stepwise execution of program
__ q    ; To quit the execution.
```

5. Procedure/ Algorithm

Program to accept 8 bit number and display 8 bit number

Explanation: Conversions from ASCII to binary usually start with keyboard data entry. If a single key is typed the conversion is accomplished by subtracting a 30H from the number. If more than one key is typed, conversion from ASCII to binary still requires that 30H be subtracted, but there is one additional step. After subtracting 30H, the prior result is first multiplied by 10, the number is added to the result. The algorithm used to convert ASCII to binary is:

1. Begin with a binary result of 0.
2. Subtract 30H from the character typed on the keyboard to convert it to BCD.
3. Multiply the binary result by 10 and add the new BCD digit.
4. Repeat steps 2 and 3 until the character typed is not an ASCII coded number of 30H-39H.

Functions and Interrupts:

1. Display message on screen.
   ```
   Mov ah,09
   Lea dx, msg
   Int 21h
   ```
2. Enter single char from user.
   ```
   Mov Ah,01
   Int 21h
   ```
   Return : AL= ASCII value
3. Display single char on screen.
   ```
   Mov Ah,02
   Int 21h
   ```

Application:

1. Conversion from ASCII to BCD
2. Conversion from BCD to ASCII

Design:

1) Explanation for displaying 8 bit number:

```
; 8086 Assembly Program to read an 8-bit number and display it
; Only supports digits 0–9, not A–F for hex input
.model small
.data
msg1 db 10,13,"Enter 8 bit no:$"    ; Message 1
msg2 db 10,13,"8 bit no is:$"       ; Message 2
.code
.startup                            ; Initializes DS = Data Segment

; Display msg1: "Enter 8 bit no:"
mov ah,09h
lea dx,msg1
int 21h

; Input first digit (high nibble)
mov ah,01h
int 21h           ; AL = ASCII input
sub al,30h         ; Convert ASCII to number (0–9)
mov cl,04h         ; Shift amount = 4 bits
shl al,cl          ; Shift left to make it high nibble
mov bl,al          ; Store in BL
; Input second digit (low nibble)
mov ah,01h
int 21h
sub al,30h         ; Convert ASCII to number
add al,bl          ; Combine high and low nibble
mov bh,al          ; Store full 8-bit value in BH
```

```
; Display msg2: "8 bit no is:"
mov ah,09h
lea dx,msg2
int 21h

; Extract and print high nibble
mov bl,bh
and bl,0f0h       ; Mask low nibble
shr bl,cl         ; Shift high nibble to lower bits
add bl,30h        ; Convert to ASCII ('0'-'9')
mov dl,bl
mov ah,02h
int 21h           ; Print high nibble

; Extract and print low nibble
mov bl,bh
and bl,0fh        ; Mask high nibble
add bl,30h        ; Convert to ASCII ('0'-'9')
mov dl,bl
mov ah,02h
int 21h           ; Print low nibble
.exit             ; Terminate program
End
```

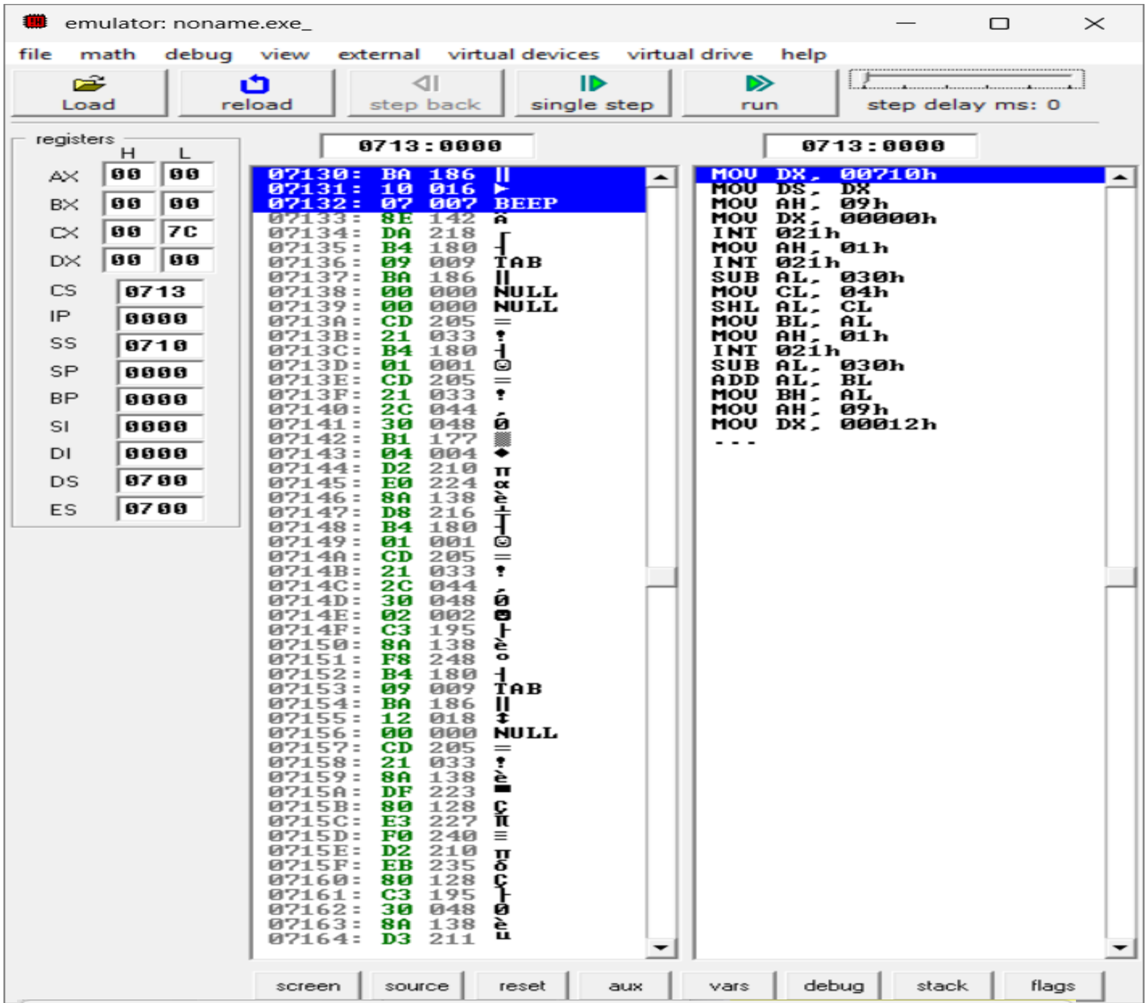
2) Explanation for displaying 16 bit number:

```
; 8086 Assembly: Read a 16-bit hex number (only digits 0–9) and display it
.model small
.data
msg1 dw 10,13,"Enter 16 bit no:$"   ; Prompt message
msg2 dw 10,13,"16 bit no is:$"      ; Output message
.code
.startup                            ; Sets DS = @data automatically
; ----- Print prompt message -----
mov ah,09h
lea dx,msg1
int 21h
; ----- Read first hex digit (high nibble of high byte) -----
mov ah,01h
int 21h
sub al,30h          ; Convert ASCII to numeric
mov cl,04h
shl al,cl           ; Shift left to make high nibble
mov bh,al           ; Store in BH temporarily
; ----- Read second hex digit (low nibble of high byte) -----
mov ah,01h
int 21h
```

```
sub al,30h
add bh,al              ; Combine to form full high byte (BH)
; ----- Read third hex digit (high nibble of low byte) -----
mov ah,01h
int 21h
sub al,30h
mov cl,04h
shl al,cl              ; Shift left to make high nibble
mov bl,al              ; Store in BL temporarily
; ----- Read fourth hex digit (low nibble of low byte) -----
mov ah,01h
int 21h
sub al,30h
add bl,al              ; Combine to form full low byte (BL)
; ----- Print output message -----
mov ah,09h
lea dx,msg2
int 21h
; ----- Display high byte (BH) -----
mov ch,bh
and ch,0f0h            ; Mask low nibble
mov cl,04h
shr ch,cl              ; Shift high nibble to right
add ch,30h             ; Convert to ASCII ('0'-'9')
mov dl,ch
mov ah,02h
int 21h                ; Display high nibble of BH
mov ch,bh
and ch,0fh             ; Mask high nibble
add ch,30h             ; Convert to ASCII
mov dl,ch
mov ah,02h
int 21h                ; Display low nibble of BH
; ----- Display low byte (BL) -----
mov dh,bl
and dh,0f0h            ; Mask low nibble
mov cl,04h
shr dh,cl              ; Shift high nibble to right
add dh,30h             ; Convert to ASCII
mov dl,dh
mov ah,02h
int 21h                ; Display high nibble of BL
mov dh,bl
and dh,0fh             ; Mask high nibble
add dh,30h             ; Convert to ASCII
mov dl,dh
mov ah,02h
int 21h            ; Display low nibble of BL
.exit              ; Exit to DOS
End
```

Result and Discussion:

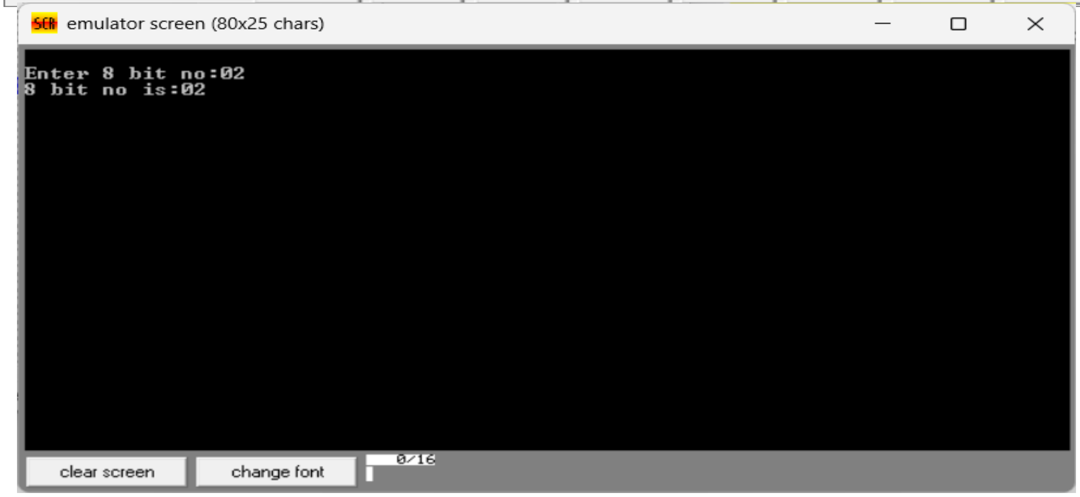
1) For 8 bit no:



```
registers
      H   L
AX   00  00
BX   00  00
CX   00  7C
DX   00  00
CS   0713
IP   0000
SS   0710
SP   0000
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

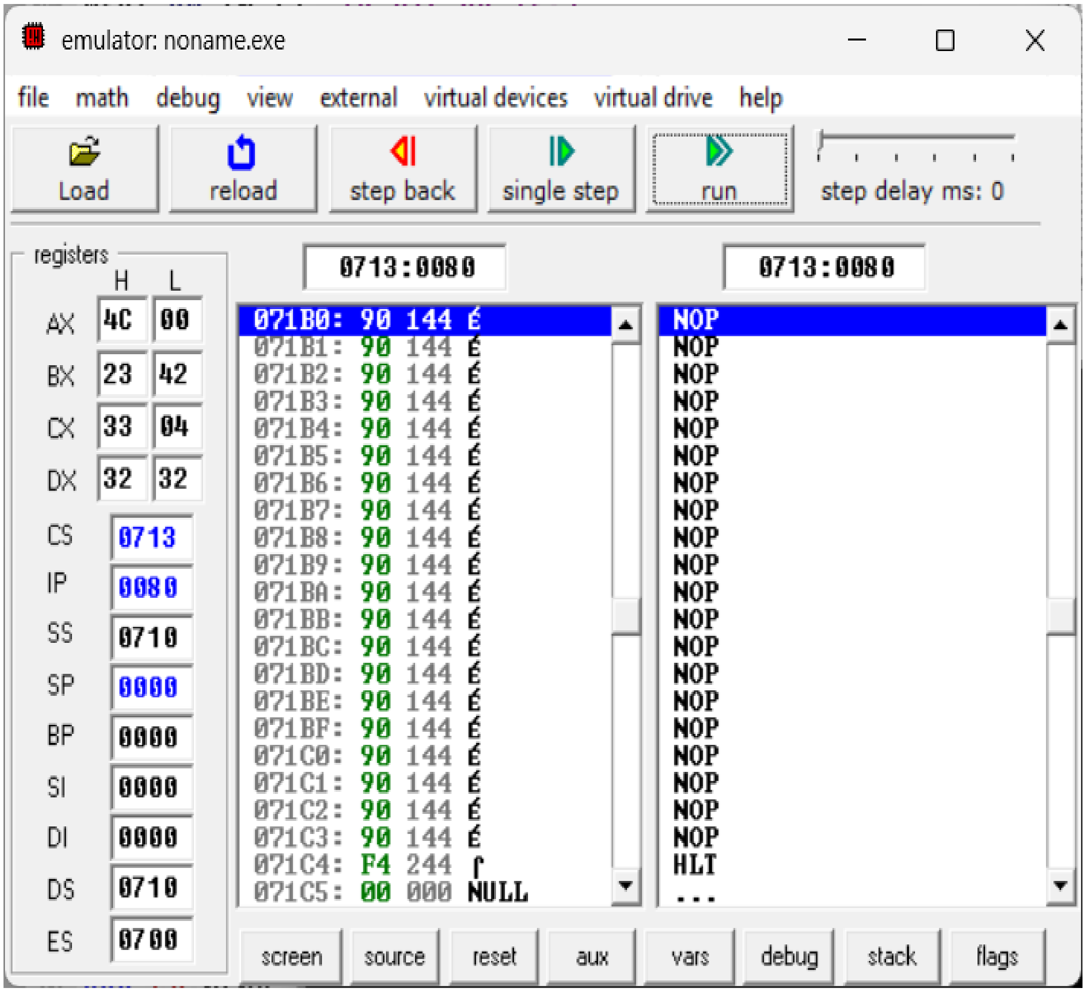
```
0713:0000
07130:  BA 186 ║
07131:  10 016 ►
07132:  07 007 BEEP
07133:  8E 142 Ä
07134:  DA 218 ┌
07135:  B4 180 ┤
07136:  09 009 TAB
07137:  BA 186 ║
07138:  00 000 NULL
07139:  00 000 NULL
0713A:  CD 205 ═
0713B:  21 033 !
0713C:  B4 180 ┤
0713D:  01 001 ☺
0713E:  CD 205 ═
0713F:  21 033 !
07140:  2C 044 ,
07141:  30 048 0
07142:  B1 177 ▒
07143:  04 004 ♦
07144:  D2 210 π
07145:  E0 224 α
07146:  8A 138 è
07147:  D8 216 ╪
07148:  B4 180 ┤
07149:  01 001 ☺
0714A:  CD 205 ═
0714B:  21 033 !
0714C:  2C 044 ,
0714D:  30 048 0
0714E:  02 002 ☻
0714F:  C3 195 ├
07150:  8A 138 è
07151:  F8 248 °
07152:  B4 180 ┤
07153:  09 009 TAB
07154:  BA 186 ║
07155:  12 018 ↕
07156:  00 000 NULL
07157:  CD 205 ═
07158:  21 033 !
07159:  8A 138 è
0715A:  DF 223 ▀
0715B:  80 128 Ç
0715C:  E3 227 π
0715D:  F0 240 ≡
0715E:  D2 210 π
0715F:  EB 235 δ
07160:  80 128 Ç
07161:  C3 195 ├
07162:  30 048 0
07163:  8A 138 è
07164:  D3 211 ╙
```

```
0713:0000
MOV DX, 00710h
MOV DS, DX
MOV AH, 09h
MOV DX, 00000h
INT 021h
MOV AH, 01h
INT 021h
SUB AL, 030h
MOV CL, 04h
SHL AL, CL
MOV BL, AL
MOV AH, 01h
INT 021h
SUB AL, 030h
ADD AL, BL
MOV BH, AL
MOV AH, 09h
MOV DX, 00012h
---
```



```
Enter 8 bit no:02
8 bit no is:02
```

2) For 16 bit number:

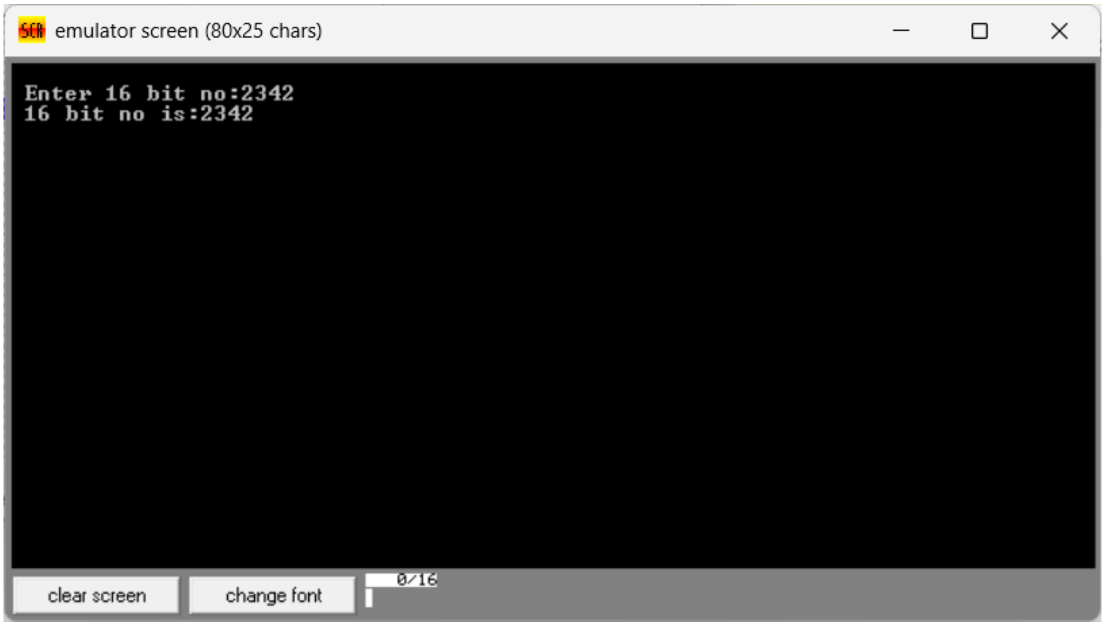


	H	L
AX	4C	00
BX	23	42
CX	33	04
DX	32	32

CS	0713
IP	0080
SS	0710
SP	0000
BP	0000
SI	0000
DI	0000
DS	0710
ES	0700

```
0713:0080
071B0: 90 144 É
071B1: 90 144 É
071B2: 90 144 É
071B3: 90 144 É
071B4: 90 144 É
071B5: 90 144 É
071B6: 90 144 É
071B7: 90 144 É
071B8: 90 144 É
071B9: 90 144 É
071BA: 90 144 É
071BB: 90 144 É
071BC: 90 144 É
071BD: 90 144 É
071BE: 90 144 É
071BF: 90 144 É
071C0: 90 144 É
071C1: 90 144 É
071C2: 90 144 É
071C3: 90 144 É
071C4: F4 244 ⌠
071C5: 00 000 NULL
```

```
0713:0080
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
HLT
...
```



```
Enter 16 bit no:2342
16 bit no is:2342
```

Learning Outcomes:

The student should have the ability to

LO1 List the features of Assembly language.
LO2 Identify the role of translator in programming language.
LO3 List and define the assemble directives.
LO4 Implement a basic program using assembly language features.

Course Outcomes: Upon completion of the course students will be able to make use of instructions of 8086 to build assembly and Mixed language programs.

Conclusion:

For Faculty Use

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance /Learning Attitude [20%]**
Marks Obtained			