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:

```
db      100 dup (?)     define 100 bytes, with no initial values for bytes
db      "Hello"         define 5 bytes, ASCII equivalent of "Hello".
maxint      equu    32767
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
        ; which is used in program.
.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.

Program to accept 16-bit number and display 16-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 all 4 characters aren’t typed and processed and store it in Data segment.
5. Retrieve the processed number from memory and using the AND operation fetch individual digits, convert it to ASCII by adding 30H and display it.
6. Repeat the above for the remaining 3 digits and display the ASCII digits sequentially.

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:

Code:

16-bit

```
.MODEL SMALL
.STACK
.DATA
MSG1 DB 10,13,"ENTER 16-Bit Number (ASCII to BCD) : $"
MSG2 DB 10,13,"DISPLAY 16-Bit NUMBER (BCD to ASCII): $"
NUM DB ?
NUM2 DB ?
.CODE
 .STARTUP
 MOV AH,09
 LEA DX,MSG1
 INT 21H
 MOV AH,01
 INT 21H
 SUB AL,30H
 MOV CL,4
```

```
SHL AL,4 ; AL=50 0101 0000
MOV BH,AL
MOV AH,01
INT 21H
SUB AL,30H
ADD BH,AL
MOV NUM,BH
MOV AH,01
INT 21H
SUB AL,30H
MOV CL,4
SHL AL,4
MOV BH,AL
MOV AH,01
INT 21H
SUB AL,30H
ADD BH,AL
MOV NUM2,BH
MOV AH,09
LEA DX,MSG2
INT 21H
MOV BH,NUM
AND BH,0F0H
; AND F0 1111 0000
; BH=50
MOV CL,4
SHR BH,CL
ADD BH,30H
MOV DL,BH
MOV AH,02
INT 21H
MOV BH,NUM
AND BH,0FH
;AND 0F 0000 1111
; BH=04H
```

```
 ADD BH,30H
 MOV DL,BH
 MOV AH,02
 INT 21H
 MOV BH,NUM2
 AND BH,0F0H
 ; AND F0 1111 0000
 ; BH=50
 MOV CL,4
 SHR BH,CL
 ADD BH,30H
 MOV DL,BH
 MOV AH,02
 INT 21H
 MOV BH,NUM2
 AND BH,0FH
 ; AND 0F 0000 1111
 ; BH=04H
 ADD BH,30H
 MOV DL,BH
 MOV AH,02
 INT 21H
 .EXIT
END
```

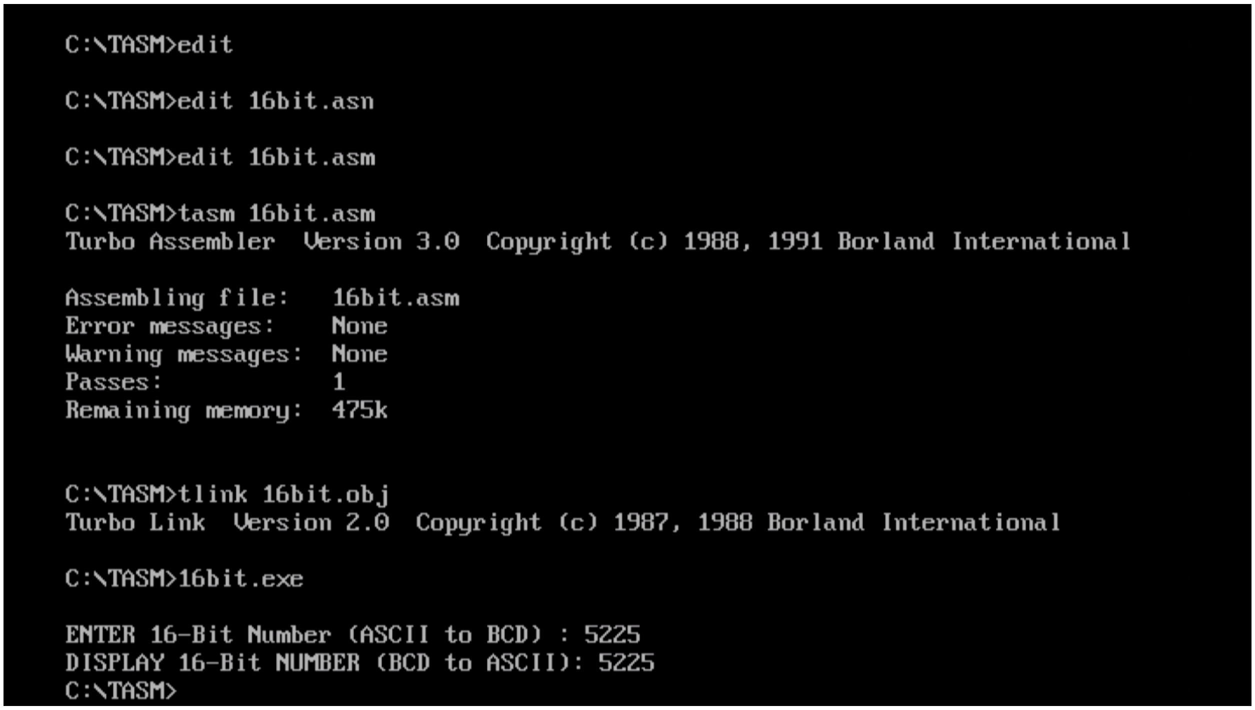
8-bit

```
.MODEL small
.STACK
.DATA
MSG1 DB 10,13,"ENTER 8-Bit Number (ASCII to BCD) : $"
MSG2 DB 10,13,"DISPLAY 8-Bit NUMBER (BCD to ASCII): $"
NO DB ?
.CODE
.STARTUP
MOV AH,09 ; DISPLAY MSG1
```

```
LEA DX, MSG1
INT 21H
MOV AH,01 ;ENTER 1ST DIGIT
INT 21H
SUB AL,30H  ; 5, AL=35-30=05 0000 0101
MOV CL,4
SHL AL,CL   ; AL=50 0101 0000
MOV BL,AL
MOV AH,01 ; ENTER 2ND DIGIT
INT 21H   ; RETURN ASCII VALUE IN AL E.G. 4, AL=34
SUB AL,30H  ; AL-30H; AL=04
ADD BL,AL   ; BL=54H
MOV NO, BL
MOV AH,09   ;DISPLAY MSG2
LEA DX, MSG2
INT 21H
MOV BH, NO
AND BH,0F0H
MOV CL,4
SHR BH,CL
ADD BH,30H
MOV DL,BH
MOV AH,02
INT 21H
MOV BH, NO
AND BH,0FH
ADD BH,30H
MOV DL,BH
MOV AH,02
INT 21H
.EXIT
END
```

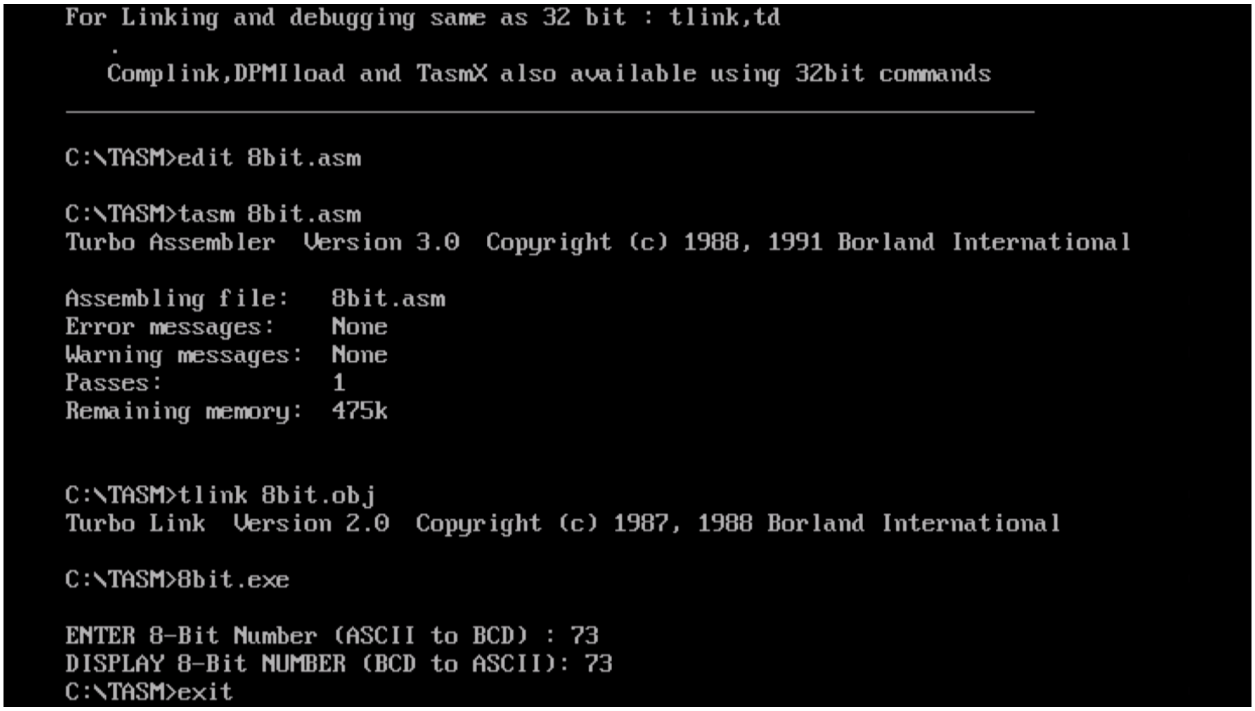
Output:

16-bit



```
C:\TASM>edit

C:\TASM>edit 16bit.asn

C:\TASM>edit 16bit.asm

C:\TASM>tasm 16bit.asm
Turbo Assembler  Version 3.0  Copyright (c) 1988, 1991 Borland International

Assembling file:   16bit.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  475k


C:\TASM>tlink 16bit.obj
Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International

C:\TASM>16bit.exe

ENTER 16-Bit Number (ASCII to BCD) : 5225
DISPLAY 16-Bit NUMBER (BCD to ASCII): 5225
C:\TASM>
```

8-bit



```
For Linking and debugging same as 32 bit : tlink,td
   .
   Complink,DPMIload and TasmX also available using 32bit commands
_____________________________________________________________________

C:\TASM>edit 8bit.asm

C:\TASM>tasm 8bit.asm
Turbo Assembler  Version 3.0  Copyright (c) 1988, 1991 Borland International

Assembling file:   8bit.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  475k


C:\TASM>tlink 8bit.obj
Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International

C:\TASM>8bit.exe

ENTER 8-Bit Number (ASCII to BCD) : 73
DISPLAY 8-Bit NUMBER (BCD to ASCII): 73
C:\TASM>exit
```

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:

Through this experiment, we’ve successfully grasped the concept of using Assembly language programming to take input and display 8-bit and 16-bit numbers.

For Faculty Use

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