COA – Lab Assignment 5

Name: Bhavin Patil

Roll No- 78

GR No : 12120056

Div : CS- D

Batch : B3

Problem Statement : Write an assembly language program to accept 5 numbers from user and arrange numbers in ascending order , descending order and display them.

Instructions –

- MOV: This instruction is used to move data from one location to another.
  Syntax – mov destination, source

- LEA (Load Effective Address): It loads the specified register with the offset of a memory location.

- JNZ: (conditional jump) The program sequence is transferred to a particular level or a 16-bit address if Z=0 (or zero flag is 0)

- JC: The JC instruction branches to the specified address if the carry flag is set. Otherwise, execution continues with the next instruction. No flags are affected by this instruction.

- SWAP: The SWAP instruction exchanges the low-order and high-order nibbles within the accumulator. No flags are affected by this instruction.

- JNC: The JNC instruction transfers program control to the specified address if the carry flag is 0. Otherwise, execution continues with the next instruction. No flags are affected by this instruction.

- DIV: The DIV (unsigned divide) instruction performs 8-bit, 16-bit, and 32-bit division on unsigned integers.

- CALL: CALL instruction is used to call a subroutine. Subroutines are often used to perform tasks that need to be performed frequently.

- JLE: It performs a signed comparison jump after a cmp if the destination operand is less than or equal to the source operand.
- CMP: The CMP instruction compares two operands. It is generally used in conditional execution. This instruction basically subtracts one operand from the other for comparing whether the operands are equal or not.
- SHL: It is a bitwise rotation, also known as a circular shift, is a bitwise operation that shifts all bits of its operand.

Commands –

1. **01h :** It is used to read character from standard input, with echo, result is stored in AL.
2. **02h :** It is used to display single character
3. **09h**: Displays the string until "$" is reached.
4. **Int 21h**: Interrupt used to exit the program.
5. **.data**: This Command is used only when we want to store in Data Segment, basically, it is the memory access of the Data Segment. Whatever we want to print must be written here. Also, the variables are declared here.
6. **10, 13**: They work as Escape Sequence Character
7. **$**: It states the end of a Statement
8. **Db (Define Byte)**: It acts as an Assembler Directive
9. **. code**: Full Logical Program is written here
10. **Tasm –** Used for Compilation
11. **tlink**– Perform linking operation

Screenshots of Source Code and Output:

Source Code –

1. Ascending

File Edit Search View Options Help

C:\TASM\EXP5.ASM

```
.model small
.data
m1 db 10,13, "Enter 5 number: $"
m2 db 10,13, "After sorting Number: $"
arr db 2 dup(?)

.code
mov ax, @data
mov ds, ax

mov cx, 05h
mov bx, offset arr

mov ah, 09h
lea dx, m1
int 21h

inputs:
mov ah, 01h
int 21h
mov [bx], al
inc bx
```

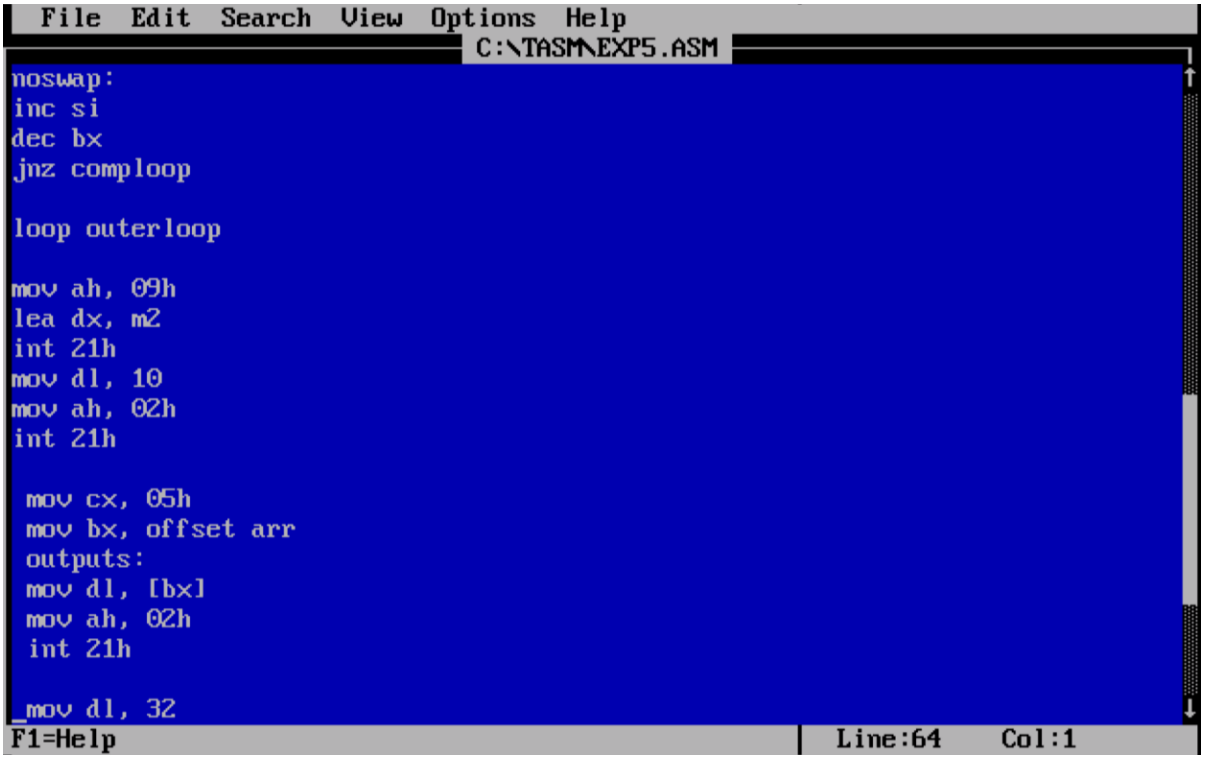
F1=Help | Line:1 Col:1

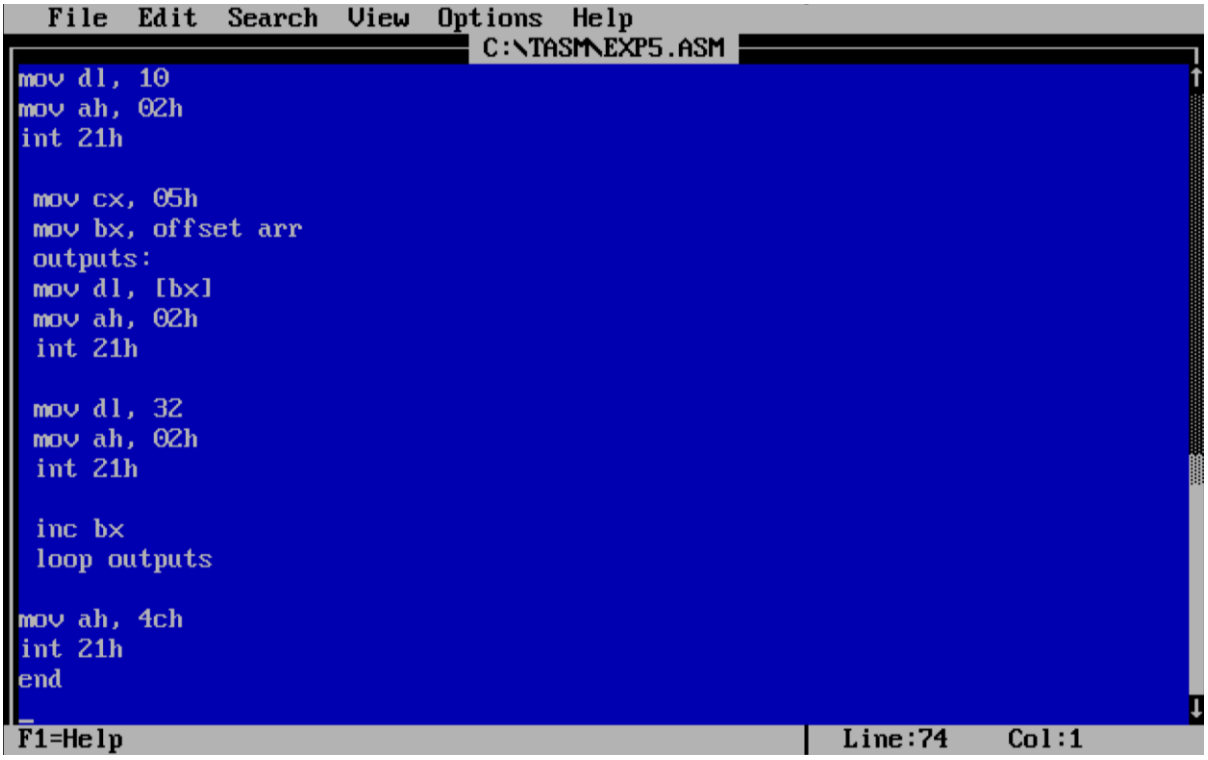
File Edit Search View Options Help

C:\TASM\EXP5.ASM

```
inc bx

Loop inputs

mov cx, 05h
dec cx

outerloop:
mov bx, cx
mov si, 0

comploop:
mov al, arr[si]
mov dl, arr[si+1]
cmp al, dl

jc noswap

mov arr[si], dl
mov arr[si+1], al

noswap:
```

F1=Help | Line:43 Col:1


File Edit Search View Options Help
C:\TASM\EXP5.ASM
noswap:
inc si
dec bx
jnz comploop
loop outerloop
mov ah, 09h
lea dx, m2
int 21h
mov dl, 10
mov ah, 02h
int 21h
mov cx, 05h
mov bx, offset arr
outputs:
mov dl, [bx]
mov ah, 02h
int 21h
mov dl, 32
F1=Help
Line:64 Col:1


File Edit Search View Options Help
C:\TASM\EXP5.ASM
mov dl, 10
mov ah, 02h
int 21h
mov cx, 05h
mov bx, offset arr
outputs:
mov dl, [bx]
mov ah, 02h
int 21h
mov dl, 32
mov ah, 02h
int 21h
inc bx
loop outputs
mov ah, 4ch
int 21h
end
F1=Help
Line:74 Col:1

2. Descending

File Edit Search View Options Help

C:\TASM\EXP5.ASM

```
.model small
.data
m1 db 10,13, "Enter 5 number: $"
m2 db 10,13, "After sorting Number: $"
arr db 2 dup(?)

.code
mov ax, @data
mov ds, ax

mov cx, 05h
mov bx, offset arr

mov ah, 09h
lea dx, m1
int 21h

inputs:
mov ah, 01h
int 21h
mov [bx], al
inc bx
```

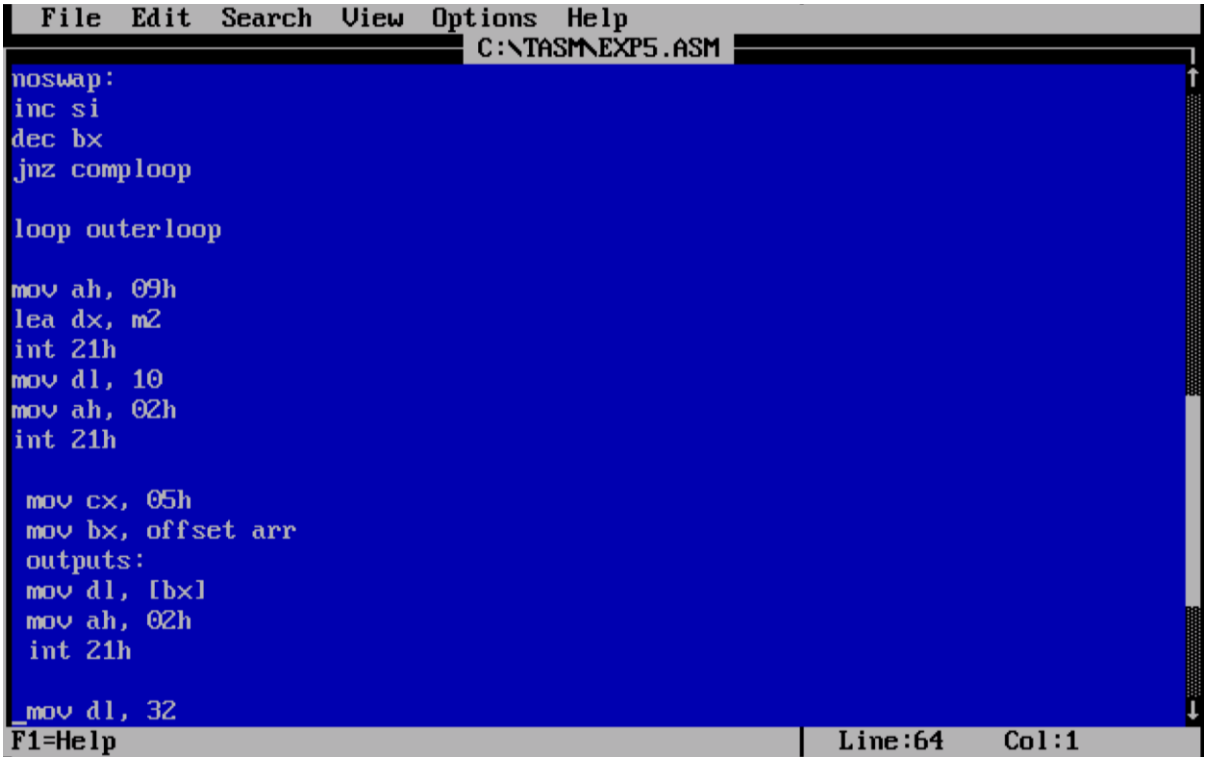
F1=Help | Line:1 Col:1

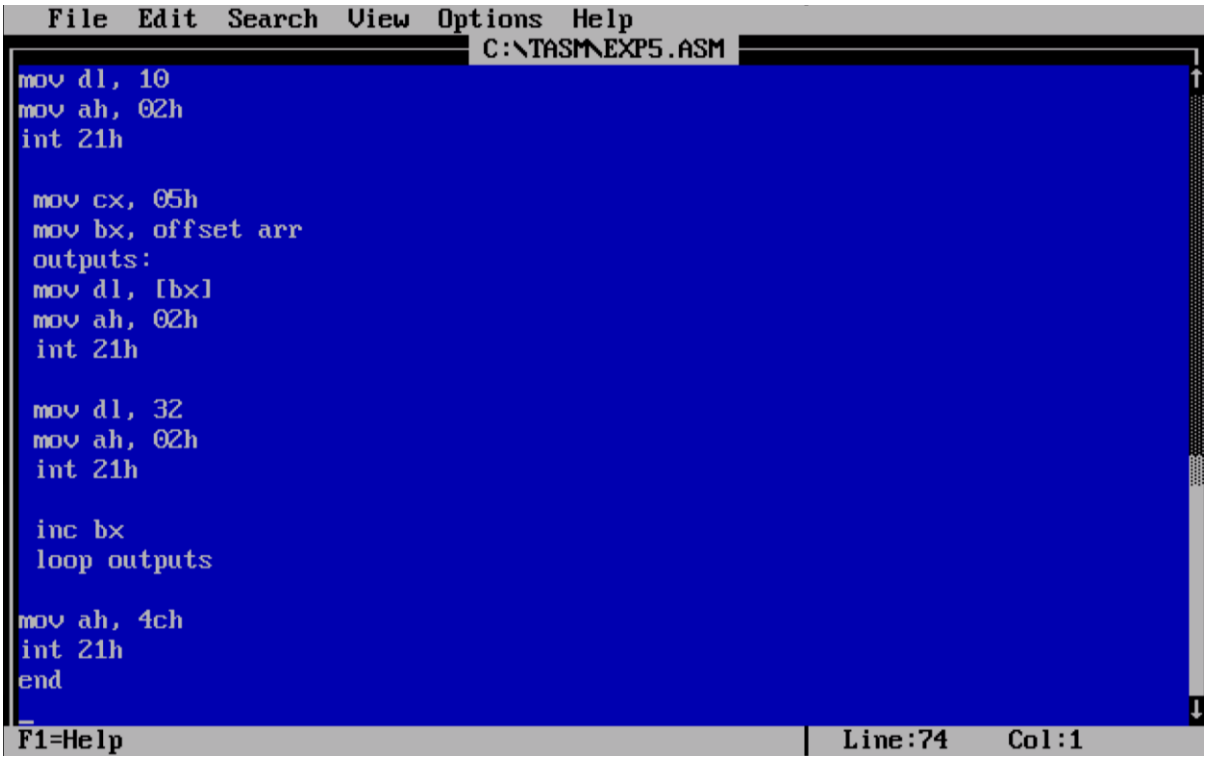
File Edit Search View Options Help

C:\TASM\EXPE5.ASM

```
inc bx

Loop inputs

mov cx, 05h
dec cx

outerloop:
mov bx, cx
mov si, 0

comploop:
mov al, arr[si]
mov dl, arr[si+1]
cmp al, dl

jnc noswap

mov arr[si], dl
mov arr[si+1], al

noswap:
```

F1=Help | Line:43 Col:1


File Edit Search View Options Help
C:\TASM\EXP5.ASM
noswap:
inc si
dec bx
jnz comploop
loop outerloop
mov ah, 09h
lea dx, m2
int 21h
mov dl, 10
mov ah, 02h
int 21h
mov cx, 05h
mov bx, offset arr
outputs:
mov dl, [bx]
mov ah, 02h
int 21h
mov dl, 32
F1=Help
Line:64 Col:1


File Edit Search View Options Help
C:\TASM\EXP5.ASM
mov dl, 10
mov ah, 02h
int 21h
mov cx, 05h
mov bx, offset arr
outputs:
mov dl, [bx]
mov ah, 02h
int 21h
mov dl, 32
mov ah, 02h
int 21h
inc bx
loop outputs
mov ah, 4ch
int 21h
end
F1=Help
Line:74 Col:1

Output:

1. Ascending

```
   Complink,DPMIload and TasmX also available using 32bit commands
_____________________________________________________________________

C:\TASM>edit exp5.asm

C:\TASM>tasm exp5.asm
Turbo Assembler  Version 3.0  Copyright (c) 1988, 1991 Borland International

Assembling file:   exp5.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  475k


C:\TASM>tlink exp5
Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International
Warning: no stack

C:\TASM>exp5.exe

Enter 5 number: 24583
After sorting Number:
2 3 4 5 8
C:\TASM>_
```

2. Descending

```
C:\TASM>tasm expe5.asm
Turbo Assembler  Version 3.0  Copyright (c) 1988, 1991 Borland International

Assembling file:   expe5.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  475k


C:\TASM>tlink expe5
Turbo Link  Version 2.0  Copyright (c) 1987, 1988 Borland International
Warning: no stack

C:\TASM>expe5.exe

Enter 5 number: 36925
After sorting Number:
9 6 5 3 2
C:\TASM>_
```