Department of Computer Science Engineering Data Science

Academic Year: 2022-23 **Semester: IV**
Class / Branch: S.E.D.S. **Subject: Microprocessor Lab**

Experiment No. 5

1. **Aim: Write anAssembly Language Program to sort an array of data in ascending order using 8086**
2. **Software used:** tasm,tlink ,td,dosbox
3. **Theory :-**

3.1 XCHG

XCHG can be used to swap the contents of two general purpose registers or between a general purpose register and memory location or content of accumulator and the data pointed by pointer and index registers.

Syntax:

XCHG destination,source

Example:

```
XCHG BX,CX      ;Exchange contents of two registers
XCHG CX,[SI]    ;Exchange the contents of the memory location pointed by DS:SI and
                ;the CX register.
```

3.2 LEA

Load Effective Address loads the specified register with the offset of a memory location. This instruction calculates the address of the `src` operand and loads it into the dest operand.

- **Syntax:**

  LEA reg, memory

- **Example:**

```
lea dx,aMessage
mov ah,09h
int 21h
```

3.3 INC

INC instruction adds one to the operand and sets the flag according to the result. INC instruction is treated as an unsigned binary number.

- **Syntax:**

  INC Reg

- **Example:**

  INC AX

3.4 DEC

Decrement destination register or memory DEC destination.

- **Syntax:**

  DEC Reg

- **Example:**

  DEC AX

3.5 JNE/JNZ

This instruction performs the Jump if not equal / Jump if not zero operation according to the condition, if ZF=0

- **Syntax:**

  JNZ label

- **Example:**

  JNZ label1

  label1: RET

3.6 JE/JZ

This instruction performs the Jump if equal (or) Jump if zero operations according to the condition if ZF = 1

- **Syntax:**

```
JZ label
```

- **Example:**

```
JZ label1
label1: MOV AH,4CH
        INT 21H
```

3.7 JB

This instruction performs the Jump if below (or) Jump if carry (or) Jump if not below/ equal operations according to the condition, if CF = 1

- **Syntax:**

```
JB label
```

- **Example:**

```
CMP AX, 4371H      ;Compare ( AX – 4371H )
JB  RUN_P          ;Jump to label RUN_P if AX is below 4371H
```

3.8 CMP

This instruction compares the source and destination value.

- **Syntax:**

```
CMP destination,source
```

- **Example:**

```
CMP AX, 4371H      ;Compare ( AX – 4371H )
JB RUN_P           ;Jump to label RUN_P if AX is below 4371H
```

3.9 INT 3H(Break Point Interrupt Instruction)

When a break point is inserted, the system executes the instructions upto the breakpoint, and then goes to the break point procedure. Unlike the single-Step feature which stops execution after each instruction, the breakpoint feature executes all the instructions up to the inserted breakpoint and then stops execution. The mnemonic for the instruction is **INT 3**. It is a 1 byte instruction. The execution of INT3 instruction results in the following.

1. Flag register value is pushed on to the Stack.
2. CS value of the return address and IP value of the return address are pushed onto the Stack.
3. IP is loaded from the contents of the word location 3x4 = 0000CH.
4. CS is loaded from the contents of the next word location.
5. Interrupt Flag and Trap Flag are reset to 0

3.10 INT 21H/AH-4CH

This instruction return control to the operating system (stop program).

- **Example**

  MOV AH,4CH

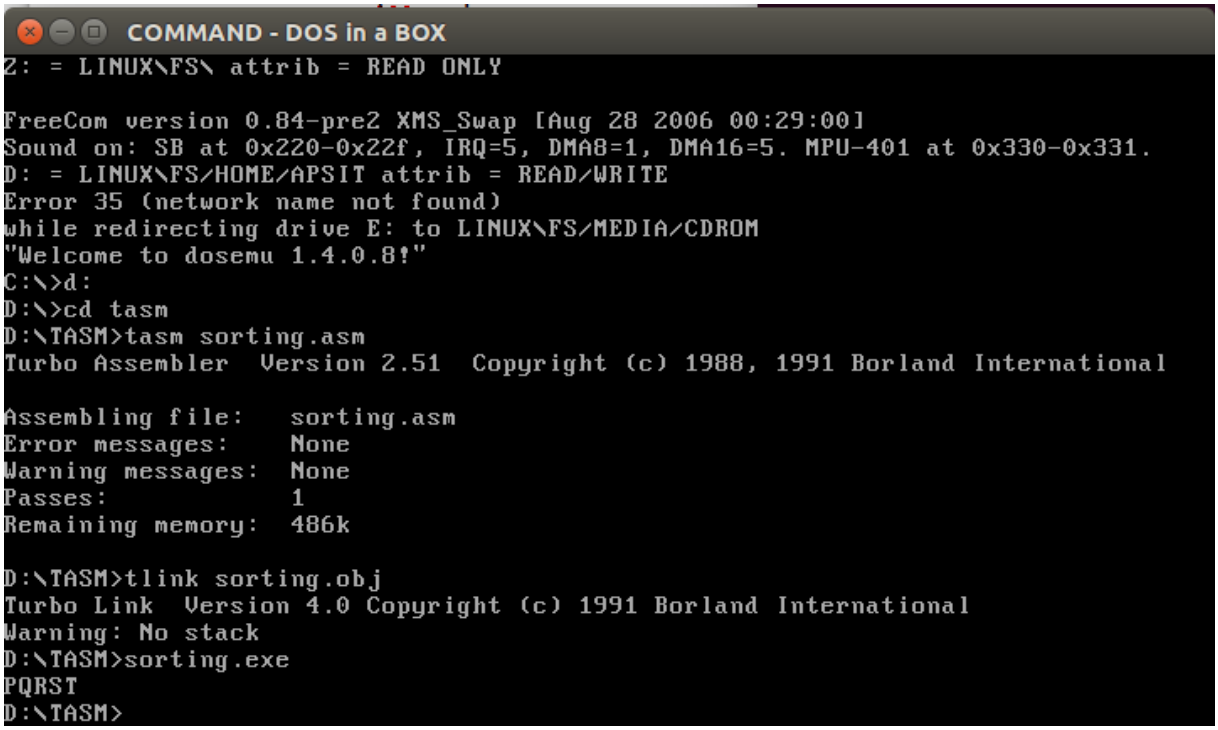
  INT 21H

4. Program

```
.model small

.data
array1 db 'd', 'b', 'e', 'a', 'c'

.code

        start: mov ax,@data
                mov ds,ax
                mov ch,04h
        up2: mov cl,04h
             lea si,array1

         up1: mov al,[si]
              mov bl,[si+1]
              cmp al,bl
              jc down
```

```
                mov dl,[si+1]
                xchg [si],dl
                mov [si+1],dl
          down: inc si
                dec cl
                jnz     up1
                dec       ch
                jnz up2
;printing array1
                mov cl,05h
                lea si,array1
     loop1:
                mov dl,[si]
                mov ah,02h
                int 21h
                inc si
                dec  cl
                jnz loop1

          mov ah,4ch
          int 21h
     end start
```

Output:



5. Conclusion :-

Exercise: Write an Assembly Language Program to sort an array of data in descending order using 8086.