# FR. Conceicao Rodrigues College of Engineering
# Department of Computer Engineering

## 4. ARRANGING NUMBER IN ASCENDING / DESCENDING ORDER.

### 1. Course, Subject & Experiment Details

| | | | |
|---|---|---|---|
| **Academic Year** | **2023-24** | **Estimated Time** | **Experiment No. 4– 02 Hours** |
| **Course & Semester** | **S.E. (Comps) – Sem. IV** | **Subject Name** | **Microprocessor** |
| **Chapter No.** | **2** | **Chapter Title** | **Instruction Set and Programming** |
| **Experiment Type** | **Software** | **Subject Code** | **CSC405** |

**Rubrics**

| Timeline (2) | Practical Skill & Applied Knowledge (2) | Output (3) | Postlab (3) | Total (10) | Sign |
|---|---|---|---|---|---|
| | | | | | |

### 2. Aim & Objective of Experiment

**Arrange the given numbers in Ascending/ Descending order.**

**Objective :** Program involves sorting an array in ascending order using Bubble sort algorithm. The objective of this program is to give an overview of the Compare and Jump instructions. Use of Indirect Addressing mode for array addressing is expected

### 3. Software Required

TASM Assembler

**Prepared by : Prof. Heenakausar Pendhari**

## 4 . Brief Theoretical Description

**Pre-Requisites:** 1. Knowledge of TASM directories.
2. Knowledge of CMP and Jump Instructions of 8086.

## 5. Algorithm:

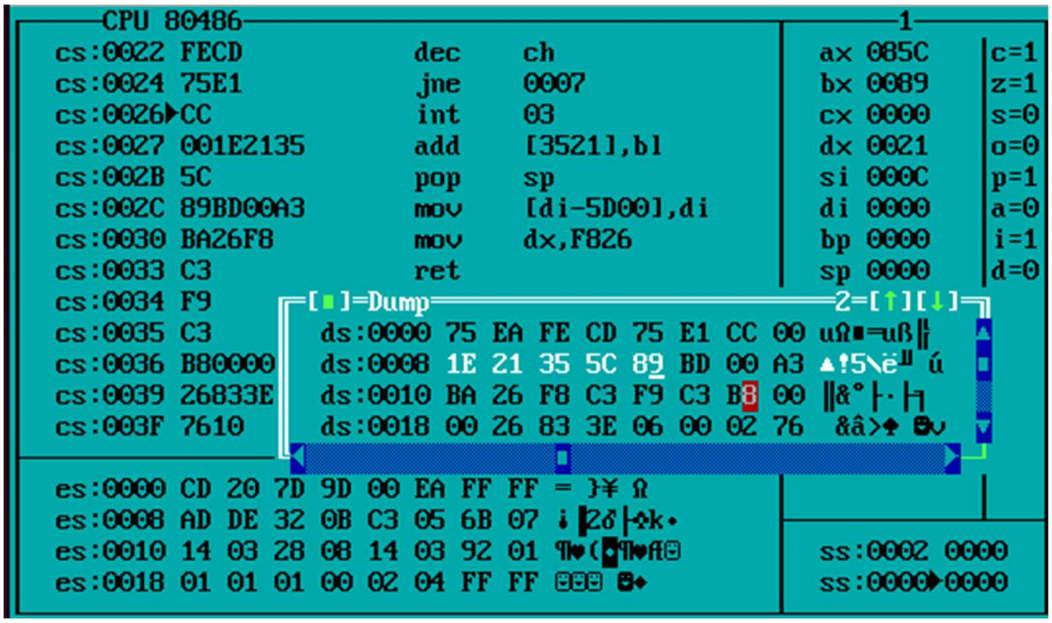
1. Initialize the data segment.
2. Initialize the array to be sorted.
3. Store the count of numbers in a register.
4. Store count-1 in another register.
5. Load the effective address of array in any general purpose register.
6. Load the first element of the array in a register.
7. Compare with the next element of the array.
8. Check for carry flag.
9. If carry=0 first number > second number. Swap the 2 numbers.
10. Increment to the contents of the SI register so that it points to the next element of the array.
11. Decrement (count-1) by 1.
12. Check if (count-1) =0. If no then repeat steps 7 to 11.
13. Decrement count by 1.
14. Check if count = 0.If no then repeat steps 6 through
    Stop.

## 6. Conclusion:

CODE:

```
.8086

.model small

.data

STRING1 db 21H, 35H, 89H, 1EH, 5CH


.code

start:
```

```
MOV AX,@data
      MOV DS,AX
      MOV CH,04H


      UP2: MOV CL,04H
      LEA SI,STRING1

      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
      INT 3
      end start
```




**Postlab:**

1. **Compare JMP and CMP instruction**

1) The CMP (compare) and JMP (unconditional jump) instructions in assembly language serve different purposes. Here's a comparison of the two instructions:

1. CMP (Compare) Instruction:

- Compares two operands by subtracting one from the other.
- Adjusts the flags (such as zero, sign, carry, overflow) based on the result of the subtraction.
- Does not alter the destination or source operands.
- Used in conditional execution and decision making, often in combination with conditional jump instructions.
- Syntax: CMP destination, source

2. JMP (Unconditional Jump) Instruction:

- Transfers the flow of control to a specified label or memory address unconditionally.
- Does not perform any comparison or evaluation of operands.
- Used to implement loops, switch statements, and to transfer control to other parts of the program.
- Syntax: JMP label or JMP memory_address

**Prepared by : Prof. Heenakausar Pendhari**