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

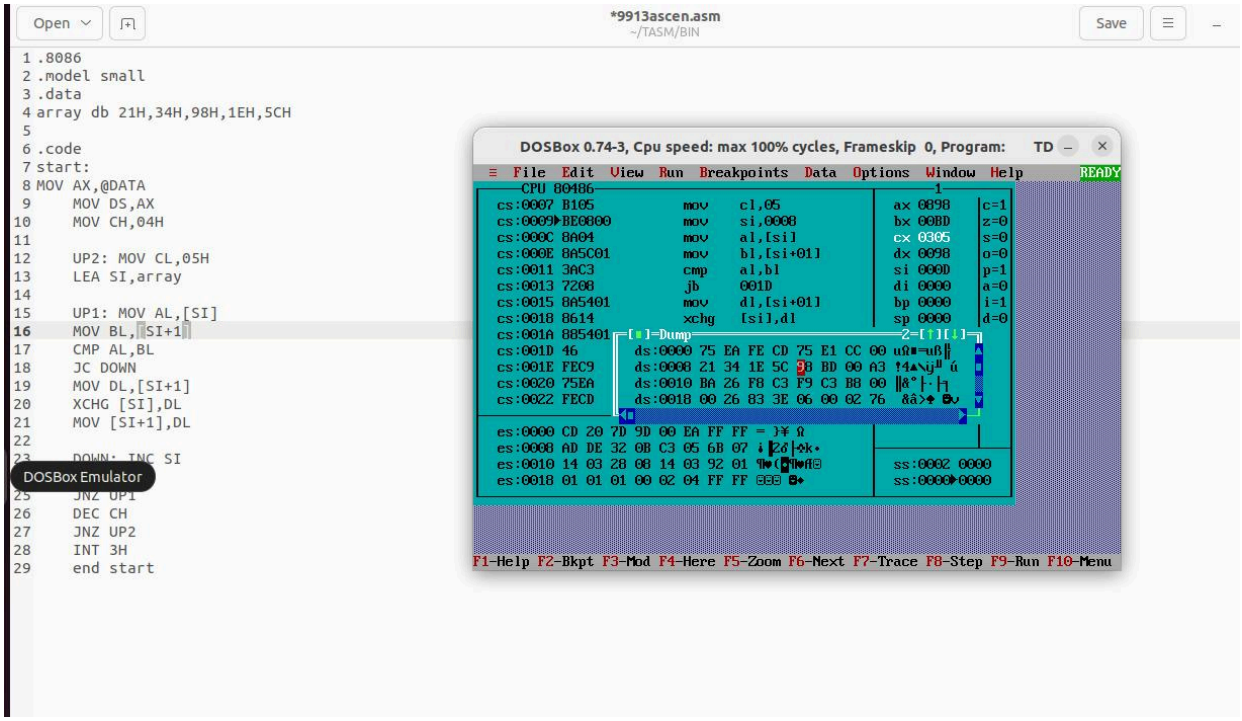
4 . Brief Theoretical Description

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:



Postlab:

1. **Compare JMP and CMP instruction**

JMP and CMP are both assembly language instructions used in various processor architectures, but they serve different purposes:

JMP (Jump):

- Function: Performs an unconditional jump to a specific memory address.expand_more
- Effect: Transfers program execution control to the instruction located at the specified address.expand_more
- No comparison involved: JMP doesn't perform any comparison; it simply jumps regardless of the processor's state.exclamation

Prepared by : Prof. Heenakausar Pendhari

- Example: `JMP 0x1000` would jump to the instruction located at memory address 0x1000.

CMP (Compare):

- Function: Compares two operands (values).
- Effect: Does not modify the operands themselves.expand_more Instead, it sets the flags register in the processor based on the comparison result (equal, greater than, less than, etc.).expand_more
- Used for conditional execution: The CMP instruction is usually followed by a conditional jump instruction (e.g., JE - jump if equal, JNE - jump if not equal, etc.) to alter program flow based on the comparison outcome.expand_more
- Example: `CMP register1, register2` would compare the values in register1 and register2, setting the flags register accordingly.expand_more You might then follow this with `JE label` to jump to a specific label (`label`) if the values were equal.