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

3. TO IMPLEMENT BLOCK TRANSFER

## Course, Subject & Experiment Details

| **Academic Year** | **2023-24** | **Estimated Time** | **Experiment No. 3– 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** |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |

## Aim & Objective of Experiment

### **Aim:** Write a program to transfer a block of data from one location to another.

**Objective :** Program involves transferring source string from a particular location in source segment (Data Segment) to the desired location in destination segment (Extra Segment). The objective of this program is to give an overview of the String instructions of 8086.

## Software Required

### TASM Simulator

1. **Pre-Requisites:** 1. Knowledge of TASM directives.

### 2. Knowledge of String Instructions of 8086.

1. **Algorithm:** 1. Initialize the data segment.
   1. Store the source string in consecutive memory location
   2. Initialize the extra segment.
   3. Allocate consecutive memory locations for transfer.
   4. Load the effective address of source string in SI register.
   5. Load the effective address of destination string in DI register.
   6. Initialize the Direction flag for Auto increment or Auto Decrement.
   7. Store number of bytes to be transferred in any of the general Purpose registers.
   8. Transfer the source string using appropriate string instructions (MOVSB / MOVSW)
   9. Decrement count
   10. Check if count = 0.If yes then stop else repeat steps 9 - 11.
   11. Stop

# Conclusion:

CODE:

.8086

.model small

.data

loc1 db 1AH, 1BH, 21H, 12H, 11H, 24H, 1BH, 1CH, 17H, 10H, 1FH, 1AH

loc2 db ?

.code

start:

MOV AX, @data

MOV DS, AX

MOV ES, AX

LEA SI, loc1

LEA DI, loc2

CLD

MOV CX, 000AH

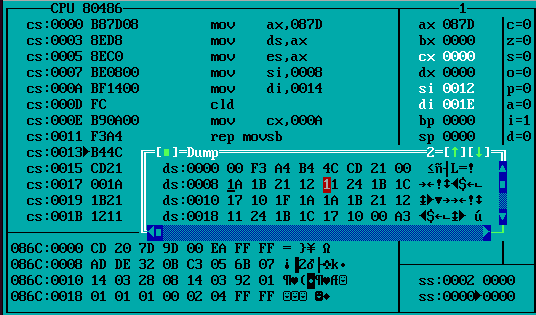
REP MOVSB

MOV AH, 4CH

INT 21H

end start

OUTPUT:



**Postlab:**

### What is the advantage of segmentation?

⇒ The advantages of segmentation with respect to block transfer on 8086 using assembly language are as follows:

1. **Memory Addressing Capacity**: Segmentation allows the memory addressing capacity to be 1 MB even though the address associated with individual instructions is only 16-bit.
2. **Memory Management**: It provides a powerful memory management mechanism, allowing data, code, and stack related operations to be performed in different segments.
3. **Code, Data, and Stack Separation**: Segmentation facilitates the use of separate memory areas for program, data, and stack, allowing code-related operations to be done in separate code segments.
4. **Sharing Data**: It allows processes to easily share data by enabling different segments to access the same.
5. **Addressing Capability**: Segmentation allows the use of 16-bit registers to give an addressing capability of 1 MB, enhancing the addressability of the processor.
6. Explain the significance of REP Prefix

⇒ The REP(repeat) prefix in assembly language is used to repeat certain string operations a specified number of times or until a particular condition is met. It is typically used with string instructions such as MOVS, STOS, SCAS, and CMPS. When a string instruction is prefixed with REP, the CX register is decremented by 1 for each iteration, and the operation is repeated until CX is 0. The REPE/REPZ (repeat while equal/zero) and REPNE/REPNZ (repeat while not equal/zero) prefixes are used with string comparison instructions to repeat the operation based on the state of the zero flag (ZF). When the specified condition is met or CX is decremented to 0, the operation terminates.