FR. Conceicao Rodrigues College of Engineering Department of Computer Engineering

5.TO COUNT EVEN AND ODD NUMBERS FROM AN ARRAY OF 10 NUMBERS.

1. Course, Subject & Experiment Details

Academic Year	**2023-24**	**Estimated Time**	**Experiment No. 5– 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

Timelin e (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 counting even and odd numbers from a given array. The objective of this program is to give an overview of the string instructions of 8086

3. Software Required

TASM Assembler

4 . Brief Theoretical Description

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

Theory: A string is a series of bytes stored sequentially in the memory. string
Instruction operates on such 'strings'. The SRC element is taken from the data segment using and SI register. The destination element is in extra segment pointed by DI register. These registers are incremented or decremented after each operation depending upon the direction flag in flag register.
Some of the instructions useful for program are,

1) CLC - the instruction clears the carry flag.
2) RCR destination, count- Right shifts the bits of destination. LSB is shifted into CF. CF goes to MSB. Bits Are shifted counts no of times.
3) JC: jump to specified location.
4) INC/DEC destination: add/subtract 1 from the specified destination.
5) JMP label: The control is shifted to an instruction to which label is attached.
6) JNZ label: The control is shifted to an instruction to which label is attached if ZF = 0

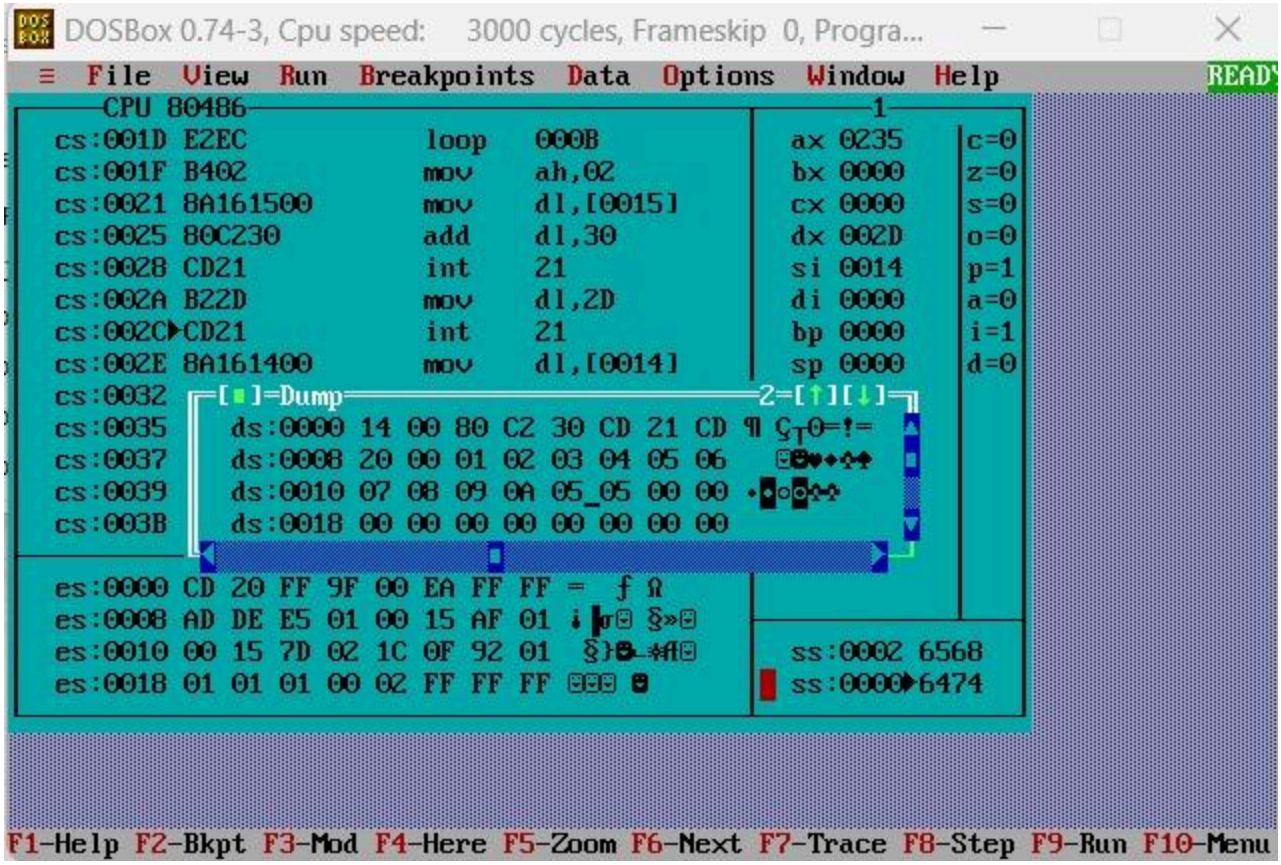
5. Algorithm:

1. . Initialize the data segment.
2. Initialize the array.
3. Load the effective address of an array in any index register.
4. Load total number of elements of the array in any register.
5. Initialize any two registers as counter for even and odd numbers to zero.
6. Load first element of an array in any general purpose register.
7. Shift/rotate the contents of loaded register to right.
8. If CF=1 increment counter for odd numbers otherwise increment counter of even numbers.
9. Store the value of even and odd counter register to two memory locations.
10. Stop.

6. Conclusion:

evodd.asm

```
.model small

.data
    array db 01H, 02H, 03H, 04H, 05H, 06H, 07H, 08H, 09H, 0AH
    odd_no db 0
    even_no db 0

.code
start:
    MOV AX, @data
    MOV DS, AX

    LEA SI, array
    MOV CX, 000AH

check:
    MOV AL, [SI]

    TEST AL, 1
    JNZ odd

    INC even_no
    JMP next

odd:
    INC odd_no

next:
    INC SI
    LOOP check

    MOV AH, 02H
    MOV DL, [even_no]
    ADD DL, 30H
    INT 21H

    MOV DL, '-'
    INT 21H

    MOV DL, [odd_no]
    ADD DL, 30H
    INT 21H

    INT 20H
end start
```



7. **Postlab:**

1. **Explain CMPSB/CMPSW , LODS/STOS instructions**

CMPSB (Compare String Byte)

- Purpose: Compare the byte at the current position in one string with the byte at the same position in another string.
- Example Use: Compare characters in two strings.

CMPSW (Compare String Word)

- Purpose: Similar to CMPSB, but compares two words (16 bits) instead of bytes.
- Example Use: Compare 16-bit values in two strings.

LODS (Load String)

- Purpose: Load a byte or a word from the memory into a register (AL or AX) and advance the source index (SI).
- Example Use: Retrieve a character or value from a string.

STOS (Store String)

- Purpose: Store a byte or a word from a register (AL or AX) into the memory at the destination index (DI) and advance DI.
- Example Use: Put a character or value into a string.