Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year:2025), B.Sc. in CSE (Day)

Lab Report NO : 3

Course Title: Microprocessor and Microcontroller lab

Course Code: CSE304 Section: 231D4

Lab Experiment Name: Implement Array and String in Assembly Language Programming

<u>Student Details</u>

Name	ID
Promod Chandra Das	231002005

Lab Date : 21/04/2025

Submission Date : 28/04/2025

Course Teacher's Name: Md. Romzan Alom

<u>Lab Report Status</u>

Marks: …………………………………

Signature:.....................

Comments:...

Date:..............................

- **Introduction:**

  In this lab session, we focused on understanding how arrays are implemented in the 8086 Assembly Language.
  Using the EMU 8086 simulator, we practically explored how arrays function, including how to input values into an array and manipulate them through various tasks.
  The exercises allowed us to interact directly with the system memory, manage string input/output operations, and carry out basic arithmetic operations at a low level.
  These activities provided deeper insights into how assembly language interacts with processor registers, memory addressing, and logical operations

- **Objective:**

  To learn how arrays are utilized in Assembly Language programming.

  To understand the handling of strings within Assembly Language programs.

- **Procedure:**

Problem 1: *Display the Elements of an Array in Reverse Order*An array of numbers was created.Accessed each element starting from the end and printed them in reverse.Instructions such as MOV, LEA, and INT 21H were used for memory handling and output.A loop was implemented to traverse and display elements backward.

Problem 2: *Input Multiple Numbers into an Array and Display Them*Developed a program to take several user inputs and store them into an array.After collecting the inputs, the program printed all elements sequentially.DOS interrupts were utilized for input and output handling

.**Problem 3:** *Input a Series of Natural Numbers and Output the Sum of Odd and Even Numbers Separately*The user was prompted to input a series of natural numbers.Each number was checked if it was odd or even.Two separate sums were calculated for odd and even numbers by iterating through the array.

❖ **Problem 1: Write an assembly language code to print out the elements in an array in reverse order.**

Code:

```
.model small
.stack 100h
.data
   arr db 1,2,3,4,5
   arr_size dw 5
.code
main:
   mov ax, @data
   mov ds, ax

   mov cx, arr_size
   dec cx
   mov si, offset arr
   add si, cx
```

```
print_loop:
    mov dl, [si]
    add dl, 30h
    mov ah, 02h
    int 21h
    dec si
    loop print_loop

    mov ah, 4Ch
    int 21h
end main
```

❖ **Problem 2:**

Write an assembly language code to: Take any number of inputs in an array. Print out the elements in an array.

Code:

```
.model small
.stack 100h
.data
   msg_in db 'Enter number:$'
   msg_out db 'Array:$'
   arr db 10 dup(?)
.code
main:
   mov ax, @data
   mov ds, ax

   mov cx, 5
   mov si, 0

input_loop:
   lea dx, msg_in
   mov ah, 09h
   int 21h
   mov ah, 01h
   int 21h
   sub al, 30h
   mov arr[si], al
   inc si
   loop input_loop

   mov cx, 5
   mov si, 0

print_loop:
   mov dl, arr[si]
   add dl, 30h
   mov ah, 02h
   int 21h
   inc si
   loop print_loop

   mov ah, 4Ch
   int 21h
```

```
end main
```

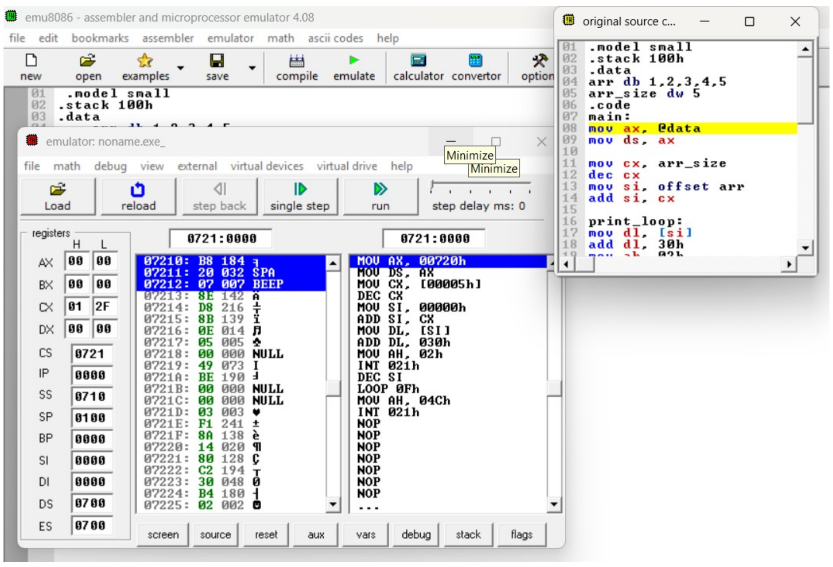
❖ **Problem 03: • Write an assembly language code to take natural number series as input and as output, show: a. The summation of odd numbers. b. The summation of even numbers. it wors successfully.**

Code:

```
.model small
.stack 100h
.data
  msg_in db 'Enter a number:$'
  msg_odd db 13,10,'Sum of odd numbers:$'
  msg_even db 13,10,'Sum of even numbers:$'
  arr db 10 dup(?)
  sum_odd db 0
  sum_even db 0
.code
main:
  mov ax, @data
  mov ds, ax

  mov cx, 5
  mov si, 0

input_loop:
  lea dx, msg_in
  mov ah, 09h
  int 21h
  mov ah, 01h
  int 21h
  sub al, 30h
  mov arr[si], al
  inc si
  loop input_loop
```

```
    mov cx, 5
    mov si, 0

process_loop:
    mov al, arr[si]
    and al, 1
    jz even_number

    ; If odd
    mov al, arr[si]
    add sum_odd, al
    jmp next_number

even_number:
    mov al, arr[si]
    add sum_even, al

next_number:
    inc si
    loop process_loop

    ; Display sum of odd numbers
    lea dx, msg_odd
    mov ah, 09h
    int 21h
    mov dl, sum_odd
    add dl, 30h
    mov ah, 02h
    int 21h

    ; Display sum of even numbers
    lea dx, msg_even
    mov ah, 09h

    int 21h

    mov dl, sum_even

    add dl, 30h

    mov ah, 02h

    int 21h



    mov ah, 4Ch

    int 21h

end main
```

- **<u>Output(1):</u>**


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor option
emulator: noname.exe_
file math debug view external virtual devices virtual drive help
Minimize
Minimize
Load reload step back single step run
step delay ms: 0
registers
screen source reset aux vars debug stack flags
original source c...

```
.model small
.stack 100h
.data
```

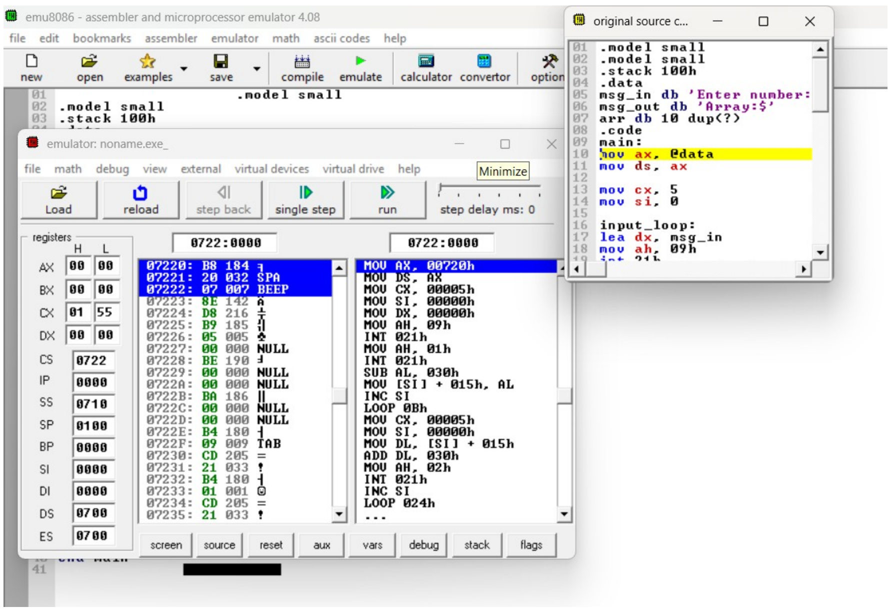
```
registers
     H   L
AX   00  00
BX   00  00
CX   01  2F
DX   00  00
CS   0721
IP   0000
SS   0710
SP   0100
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

```
0721:0000

07210: B8 184 ╕
07211: 20 032 SPA
07212: 07 007 BEEP
07213: 8E 142 Ä
07214: D8 216 ╪
07215: 8B 139 ï
07216: 0E 014 ♫
07217: 05 005 ♣
07218: 00 000 NULL
07219: 49 073 I
0721A: BE 190 ╛
0721B: 00 000 NULL
0721C: 00 000 NULL
0721D: 03 003 ♥
0721E: F1 241 ±
0721F: 8A 138 è
07220: 14 020 ¶
07221: 80 128 Ç
07222: C2 194 ┬
07223: 30 048 0
07224: B4 180 ┤
07225: 02 002 ☻
```

```
0721:0000

MOV AX, 00720h
MOV DS, AX
MOV CX, [00005h]
DEC CX
MOV SI, 00000h
ADD SI, CX
MOV DL, [SI]
ADD DL, 030h
MOV AH, 02h
INT 021h
DEC SI
LOOP 0Fh
MOV AH, 04Ch
INT 021h
NOP
NOP
NOP
NOP
NOP
NOP
NOP
...
```

```
.model small
.stack 100h
.data
arr db 1,2,3,4,5
arr_size dw 5
.code
main:
mov ax, @data
mov ds, ax

mov cx, arr_size
dec cx
mov si, offset arr
add si, cx

print_loop:
mov dl, [si]
add dl, 30h
```

- **<u>Output(2):</u>**


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor option
emulator: noname.exe_
file math debug view external virtual devices virtual drive help
Minimize
Load reload step back single step run
step delay ms: 0
registers
screen source reset aux vars debug stack flags
original source c...

```
                    .model small
.model small
.stack 100h
```

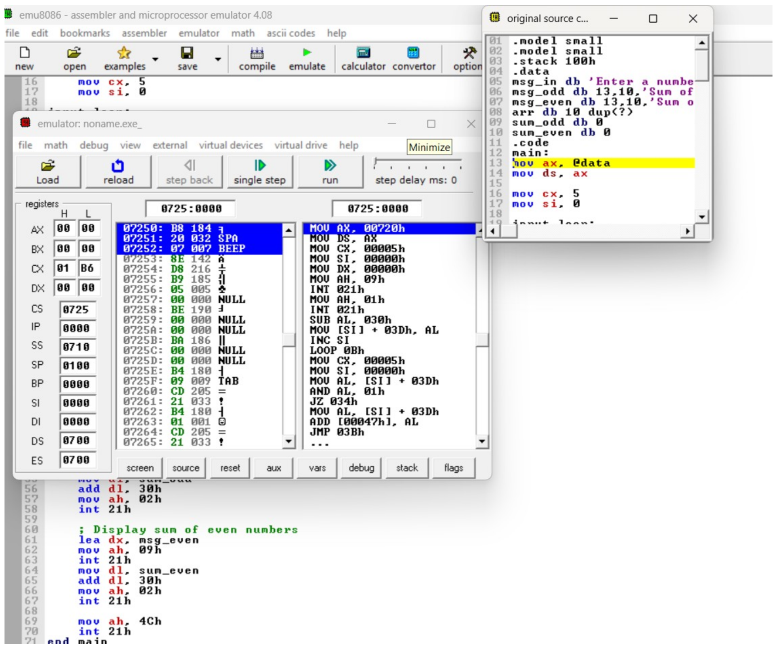
```
registers
     H   L
AX   00  00
BX   00  00
CX   01  55
DX   00  00
CS   0722
IP   0000
SS   0710
SP   0100
BP   0000
SI   0000
DI   0000
DS   0700
ES   0700
```

```
0722:0000

07220: B8 184 ╕
07221: 20 032 SPA
07222: 07 007 BEEP
07223: 8E 142 Ä
07224: D8 216 ╪
07225: B9 185 ╣
07226: 05 005 ♣
07227: 00 000 NULL
07228: BE 190 ╛
07229: 00 000 NULL
0722A: 00 000 NULL
0722B: BA 186 ║
0722C: 00 000 NULL
0722D: 00 000 NULL
0722E: B4 180 ┤
0722F: 09 009 TAB
07230: CD 205 ═
07231: 21 033 !
07232: B4 180 ┤
07233: 01 001 ☺
07234: CD 205 ═
07235: 21 033 !
```

```
0722:0000

MOV AX, 00720h
MOV DS, AX
MOV CX, 00005h
MOV SI, 00000h
MOV DX, 00000h
MOV AH, 09h
INT 021h
MOV AH, 01h
INT 021h
SUB AL, 030h
MOV [SI] + 015h, AL
INC SI
LOOP 0Bh
MOV CX, 00005h
MOV SI, 00000h
MOV DL, [SI] + 015h
ADD DL, 030h
MOV AH, 02h
INT 021h
INC SI
LOOP 024h
...
```

```
.model small
.model small
.stack 100h
.data
msg_in db 'Enter number:
msg_out db 'Array:$'
arr db 10 dup(?)
.code
main:
mov ax, @data
mov ds, ax

mov cx, 5
mov si, 0

input_loop:
lea dx, msg_in
mov ah, 09h
```

- **Output(3):**


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options
emulator: noname.exe_
file math debug view external virtual devices virtual drive help
Minimize
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags
original source c...

```
        mov cx, 5
        mov si, 0
```

```
registers
     H   L
AX  00  00
BX  00  00
CX  01  B6
DX  00  00
CS  0725
IP  0000
SS  0710
SP  0100
BP  0000
SI  0000
DI  0000
DS  0700
ES  0700
```

```
0725:0000
07250: B8 184 ╕
07251: 20 032 SPA
07252: 07 007 BEEP
07253: 8E 142 Ä
07254: D8 216 ╪
07255: B9 185 ╣
07256: 05 005 ♣
07257: 00 000 NULL
07258: BE 190 ╛
07259: 00 000 NULL
0725A: 00 000 NULL
0725B: BA 186 ║
0725C: 00 000 NULL
0725D: 00 000 NULL
0725E: B4 180 ┤
0725F: 09 009 TAB
07260: CD 205 ═
07261: 21 033 !
07262: B4 180 ┤
07263: 01 001 ☺
07264: CD 205 ═
07265: 21 033 !
```

```
0725:0000
MOV AX, 00720h
MOV DS, AX
MOV CX, 00005h
MOV SI, 00000h
MOV DX, 00000h
MOV AH, 09h
INT 021h
MOV AH, 01h
INT 021h
SUB AL, 030h
MOV [SI] + 03Dh, AL
INC SI
LOOP 0Bh
MOV CX, 00005h
MOV SI, 00000h
MOV AL, [SI] + 03Dh
AND AL, 01h
JZ 034h
MOV AL, [SI] + 03Dh
ADD [00047h], AL
JMP 03Bh
...
```

```
.model small
.model small
.stack 100h
.data
msg_in db 'Enter a numbe
msg_odd db 13,10,'Sum of
msg_even db 13,10,'Sum o
arr db 10 dup(?)
sum_odd db 0
sum_even db 0
.code
main:
mov ax, @data
mov ds, ax

mov cx, 5
mov si, 0

```

```
        add dl, 30h
        mov ah, 02h
        int 21h

        ; Display sum of even numbers
        lea dx, msg_even
        mov ah, 09h
        int 21h
        mov dl, sum_even
        add dl, 30h
        mov ah, 02h
        int 21h

        mov ah, 4Ch
        int 21h
end main
```

Discussion:

Throughout this lab, we practically learned the implementation of arrays using EMU 8086.
We tackled different tasks, such as printing array elements in reverse, taking dynamic input, and calculating separate sums for odd and even numbers.
Initially, understanding the use of SI register and offset addressing was a bit challenging, but with the instructor's detailed explanation, these concepts became much clearer.
Working through the problems helped us not only understand the theoretical concepts but also provided a solid hands-on experience in managing memory and working with low-level logic in assembly programming.