

# Green University of Bangladesh

## Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year: 2021), B.Sc. in CSE (Day)

Lab Report No: 06

Course Title: Microprocessors and Microcontrollers Lab

Course Code: CSE 304      Section: PC-DD

**Lab Experiment Name:** Introduction of understanding the use of MACRO.

### Student Details

Name	Id
Md. Romzan Alom	201902144

Lab Date: 13.12. 2021

Submission Date: 20.12.2021

Course Teacher's Name: Rusmita Halim Chaity

**[For Teachers use only: Don't Write Anything inside this box]**

Lab Report Status	
Marks: .....	Signature:.....
Comments:.....	Date:.....

**Title of the Lab Experiment:** Introduction of understanding the use of MACRO.

### **Objectives / Aim:**

We learn about MACRO, Array and Procedure from this experiment. We can take user input as elements of array index and we can perform various operations on this input to use MACRO and Procedure and show the result as output.

### **Procedure / Analysis / Design:**

Instruction MOV AX, BX is used moves data BX to AX and data is stored in AX.

Instruction DIV BX is used division. Here, AX is divided by BX and Quotient is stored in AL and Remainder is stored in AH.

Instruction ADD AX, BX is used addition. Here, add to BX and AX and this data is stored in AX.

Instruction SUB AX, BX is used subtraction. Here, BX and AX are subtracted and this data is stored in AX.

Instruction MOV AH, 1 and INT 21H are used for user input and it stored AX.

Instruction MOV DX, AX and MOV AH, 2 and INT 21H are used for show output. Here, DX store AX data and compiler understand DX data then it show DX data as output.

Instruction LEA DX, R and MOV AH, 9 and INT 21H are used for call variable and show output the variable data.

Instruction MOV AH,2 and MOV DL,0DH and INT 21H and MOV DL,0AH and INT 21H are used for newline.

Instruction MOV AH,2 and MOV DL,0DH and INT 21H and MOV DL,0AH and INT 21H are used for newline.

Instruction Array DB N (?) is used for initialize an array.

Instruction CMP AX, BX are used for compare AX and BX.

Instruction JL and JGE are used for CMP condition.

Instruction RESULT\_1: are created a level.

Instruction LOOP START is used for looping. It goes to START level and this level decrease with CX.

Instruction R PROC to R ENDP is used for procedure.

Instruction call is used for call procedure.

Instruction R MICRO X to ENDM is used for creating a MICRO. When we need MICRO then we write only R and variable name.

### **Problem-01**

1. Write an Assembly Language code that takes an input ARRAY and passes the array values and address to a MACRO. Using the array, address and one procedure separate out the ODD digits and EVEN digits.

Input: 2 0 4 7 1 9 Output: ODD Digits: 7 1 9 EVEN Digits: 2 0 4
---

#### **Pseudo-code:**

Romzan macro R

mov al,R

mov ah,2

mov dl,al

add dl,30H

int 21H

endm

.MODEL SMALL

.STACK 100H

.DATA

n db ?

odd db ?

even db ?

z db ?

A db n dup (?)

R1 DB "Enter the number of input: \$"

R2 DB 0AH,0DH,"Enter the all element of index: \$"

R3 DB 0AH,0DH,"The result of our program: \$"

R4 DB 0AH,0DH,"\$"

R5 DB 0AH,0DH,"All even numbers of array are: \$"

R6 DB 0AH,0DH,"All odd numbers of array are: \$"

.CODE

MAIN PROC

mov ax, @DATA

mov ds, ax

mov odd,0

mov even, 0

mov bx,0

lea dx,R1

mov ah,9

int 21H

mov ah,1

int 21H

sub al,30H

```
mov n,al
```

```
lea dx,R4
```

```
mov ah,9
```

```
int 21H
```

```
lea dx,R2
```

```
mov ah,9
```

```
int 21H
```

```
xor cx,cx
```

```
mov cl,n
```

```
mov si,0
```

```
Loop_1:
```

```
mov ah,1
```

```
int 21H
```

```
sub al,30H
```

```
mov A[si],al
```

```
inc si
```

```
loop Loop_1
```

```
lea dx,R4
```

```
mov ah,9
```

```
int 21H
```

```
lea dx,R3
```

```
mov ah,9
```

```
int 21H
```

```
lea dx,R4
```

```
mov ah,9
```

```
int 21H
```

```
lea dx,R5
```

```
mov ah,9
```

```
int 21H
```

```
xor cx,cx
```

```
mov cl,n
```

```
mov z,2
```

```
mov si,0
```

```
Loop_2:
```

```
call F_even
```

```
loop Loop_2
```

```
lea dx,R4
```

```
mov ah,9
```

```
int 21H
```



lea dx,R6

mov ah,9

int 21H

xor cx,cx

mov cl,n

mov si,0

Loop\_3:

call F\_odd

loop Loop\_3

mov ah, 4ch

int 21h

MAIN ENDP

F\_even proc

MOV AX,00

MOV AL,A[si]

DIV z

mov di,si

inc si

CMP AH,00

JZ RESULT\_1

JNZ RESULT\_2

RESULT\_1:

Romzan A[di]

RESULT\_2:

ret

F\_even ENDP

F\_odd proc

MOV AX,00

MOV AL,A[si]

DIV z

mov di,si

inc si

CMP AH,00

JNZ RESULT\_3

JZ RESULT\_4

RESULT\_3:

Romzan A[di]

RESULT\_4:

ret

F\_odd ENDP

END MAIN

## **Problem-02**

- Write an Assembly Language code that takes an input ARRAY and passes the array values and address to a MACRO. Now produce the summation of odd digits and even digits as output.

Input: 3 1 4 5 1 6 8 7 Output: ODD Digits: 17 EVEN Digits: 18
---

### **Pseudo-code:**

Romzan macro R1,E,O,Arr

mov ah,R1

mov al,E

mov bl,O

mov BH,Arr

CMP AH,00

JZ RESULT\_1

JNZ RESULT\_2

RESULT\_1:

call Function

mov E,al

jmp Level

RESULT\_2:

call Function1

mov O,bl

```
    jmp Level  
endm
```

```
.MODEL SMALL
```

```
.STACK 100H
```

```
.DATA
```

```
n db ?
```

```
odd db ?
```

```
even db ?
```

```
R db ?
```

```
z db ?
```

```
B db ?
```

```
A db n dup (?)
```

```
R1 DB "Enter the number of input: $"
```

```
R2 DB 0AH,0DH,"Enter the all element of index: $"
```

R3 DB 0AH,0DH,"The result of our program: \$"

R4 DB 0AH,0DH,"\$"

R5 DB 0AH,0DH,"The summation of odd: \$"

R6 DB 0AH,0DH,"The summation of even: \$"

.CODE

MAIN PROC

mov ax, @DATA

mov ds, ax

mov odd,0

mov even, 0

mov bx,0

lea dx,R1

mov ah,9

int 21H

mov ah,1

int 21H

sub al,30H

mov n,al

lea dx,R4

mov ah,9

int 21H

lea dx,R2

mov ah,9

int 21H

xor cx,cx

mov cl,n

mov si,0



Loop\_1:

mov ah,1

int 21H

sub al,30H

mov A[si],al

inc si

loop Loop\_1

lea dx,R4

mov ah,9

int 21H

lea dx,R3

mov ah,9

int 21H

xor cx,cx

mov cl,n

mov z,2

mov si,0

Loop\_2:

MOV AX,00

MOV AL,A[si]

DIV z

mov di,si

inc si

mov B,ah

Romzan B,even,odd,A[di]

Level:

loop Loop\_2

Final:

;sum of odd

lea dx,R4

mov ah,9

int 21H

lea dx,R5

mov ah,9

int 21H

mov R,10

xor ax,ax

mov al,odd

div R

mov bh,ah

```
mov ah,2  
mov dl,al  
add dl,30H  
int 21H
```

```
mov ah,2  
mov dl,bh  
add dl,30H  
int 21H
```

```
lea dx,R4  
mov ah,9  
int 21H
```

```
lea dx,R6  
mov ah,9  
int 21H
```

xor ax,ax ;sum of even

mov al,even

div R

mov bh,ah

mov ah,2

mov dl,al

add dl,30H

int 21H

mov ah,2

mov dl,bh

add dl,30H

int 21H

```
mov ah, 4ch
```

```
int 21h
```

```
MAIN ENDP
```

```
Function proc
```

```
    add al,BH
```

```
ret
```

```
Function ENDP
```

```
Function1 proc
```

```
    add bl,BH
```

```
ret
```

```
Function1 ENDP
```

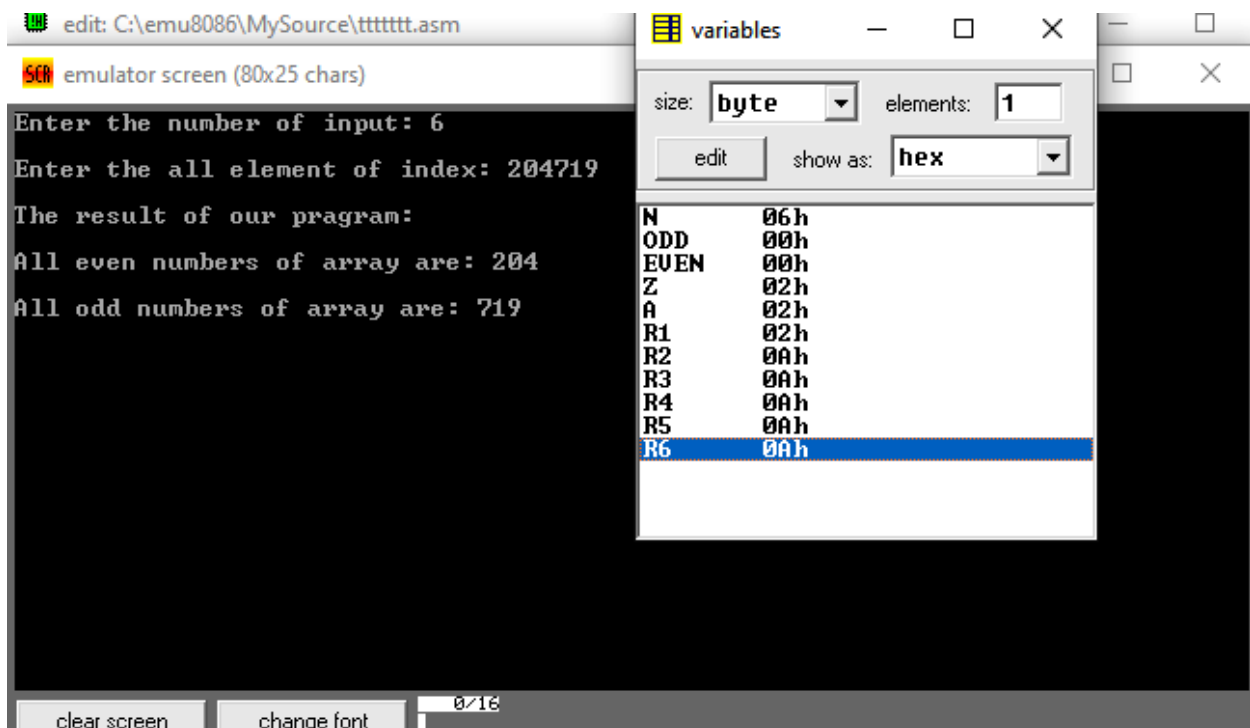
```
END MAIN
```

## Calculation:

We will try to find odd and even number from an array and also find the summation of odd and even. For find summation, we use a law. That is, summation = sum of all elements values. On the other hand when we check a number odd or even then we use a condition that is if number % 2 is 0 that means number is even otherwise it is odd.

## Test Result / Output:

### Problem-1 output:



The screenshot shows an 8086 emulator interface. The main window displays the following text:

```
Enter the number of input: 6
Enter the all element of index: 204719
The result of our program:
All even numbers of array are: 204
All odd numbers of array are: 719
```

At the bottom of the main window are buttons for "clear screen" and "change font", and a status bar showing "0/16".

Overlaid on the right is a "variables" window. It has a "size" dropdown set to "byte", an "elements" input set to "1", an "edit" button, and a "show as" dropdown set to "hex". Below these controls is a table of variables:

N	06h
ODD	00h
EVEN	00h
Z	02h
A	02h
R1	02h
R2	0Ah
R3	0Ah
R4	0Ah
R5	0Ah
R6	0Ah

The serial numbers are 2, 0, 4, 7, 1 and 9.

So, all even numbers of serial are 2, 0 and 4.

Again, all odd numbers of serial are 7, 1 and 9.

We see our output and calculation output are same. So, it is 100% right outputs.

## Problem-2 output:

The screenshot shows an 8086 emulator window titled "edit: C:\emu8086\MySource\odd even using macro.asm". The main window is titled "emulator screen (80x25 chars)" and displays the following text:

```
Enter the number of input: 8
Enter the all element of index: 31451678
The result of our program:
The summation of odd: 17
The summation of even: 18
```

At the bottom of the emulator window are buttons for "clear screen" and "change font", and a status bar showing "0/16".

Overlaid on the right is a "variables" window. It has a "size" dropdown set to "byte", an "elements" input set to "1", an "edit" button, and a "show as" dropdown set to "signed". Below this is a table of variables:

N	8
ODD	17
EVEN	18
R	10
Z	2
B	0
A	3
R1	3
R2	10
R3	10
R4	10
R5	10
R6	10

The serial numbers are 2, 0, 4, 7, 1 and 9.

So, the summation of even number is  $(4 + 6 + 8) = 18$ .

Again, the summation of odd number is  $(3 + 1 + 5 + 1 + 7) = 17$ .

We see our output and calculation output are same. So, it is 100% right outputs.

## Analysis and Discussion:

1. Due to Covid-19 situation, we can't do this experiment directly. So, it is completely based on software.
2. Since, it is done with Software. So it may have some Software and Mechanical errors.



3. To emulate those codes I am facing so many problems for hexadecimal number.
4. From those problems, we use MICRO. When we operate that MICRO, we facing some problem. Like Initialize and put on variable's value.
5. We can use loop but loop decreases with CX register. That is really confusing to think when loop will stop.
6. To find summation of even and odd from an array I am facing too many problems for using two procedure and MICRO. Like bring variable to MICRO.
7. When we use MICRO in code then it stop that line and works in MICRO. That why sometimes I can't understand its work.
8. From our compile show only char value of hexadecimal number. That why we face some problem to sort array index.
9. Network problem. Cause of the ups and downs of the internet we could not attend the class properly. That why I saw the class recording but this video was very bad quality. This reason I have so many confusion.

## **SUMMARY:**

From this problem, we can take so many decimal numbers from user as input and also take an array from user and print the summation of even or odd and check a number that is even or odd of an array or perform various operations on this input and show the result as output very easily which will help the user to change the program's output. We can also use MICRO and Procedure that help to make a program easier and smaller than before. Those are very important to complete this course.