



Date:.....

1. TITLE OF THE LAB EXPERIMENT

Implementation of loop using assembly language.

2. OBJECTIVES/AIM

To reinforce fundamental concepts in assembly language programming, such as loops, input/output operations, branching, and conditional checks.

3. PROCEDURE / ANALYSIS / DESIGN

Problem 1: Use loop to find out the summation of $1+3+5+7+\dots+99$. Also try to find out the summation using formula.

Step 1: Initialize the program and data section.

- Set the program's origin to 100h.
- Define the data section.

Step 2: Start of the code section.

- Begin the 'code' section.
- Define the 'main' procedure.

Step 3: Initialize data segment and registers.

- Load the data segment address into AX from '@data'.
- Set DS (data segment) to the value in AX.

Step 4: Initialize variables and registers.

- Set AX to 1: Initialize AX with the value 1.
- Set CX to 50: Initialize CX with the value 50.
- Clear DX: Set DX to 0 (zero).

Step 5: Loop processing.

Label 'Label1' is used to mark the start of a loop.

Inside the loop:

- Add AX to DX: Add the value in AX to DX.
- Add 2 to AX: Increase the value in AX by 2.

Use the 'loop' instruction to decrement CX and repeat the loop until CX is not zero.

If CX is not zero, jump back to 'Label1' for the next iteration.

Step 6: End of the program.

- Mark the end of the 'main' procedure using 'main endp.'
- End the 'code' section and the program using 'end main.'

Problem 2: Take a number n from user. After that find out the factorial of that number n . (Suppose for $n=5$, you have to find out factorial= $1 \times 2 \times 3 \times 4 \times 5$.)

Step 1: Initialize the program and data section.

- Set the program's origin to 100h.

- Define the data section.

- Declare variables.

Step 2: Start of the code section.

- Begin the 'code' section.

- Define the 'main' procedure.

Step 3: Setup data segment and display the input prompt.

- Load the data segment address into AX from '@data'.

- Set DS (data segment) to the value in AX.

- Set AH to 09h (print string function).

- Load the offset address of 'msg1' into DX.

- Trigger interrupt 21h to display the "Enter the number: \$" message.

Step 4: Read a character from the user.

- Set AH to 01h (read character function).

- Trigger interrupt 21h to read a character from the user.

- Subtract 48 (ASCII value of '0') from AL to convert it to a numerical value.

Step 5: Store the user-entered number.

- Move the value from AL (user-entered number) to CL (counter) for further processing.

- Set AX to 1: Initialize AX with the value 1.

Step 6: Calculate the factorial.

- Label 'Labell' is used to mark the start of a loop.

- Inside the loop:

- Multiply AX by CX (factorial calculation).

- Move the result in DX to AX.

- Use the 'loop' instruction to decrement CX and repeat the loop until CX is not zero.

- If CX is not zero, jump back to 'Labell' for the next iteration.

Step 7: Display the factorial result.

- Set AH to 2 (display character function).

- Trigger interrupt 21h to display the result in AX

Step 8: End of the program.

- Mark the end of the 'main' procedure using 'main endp.'
- End the 'code' section and the program using 'end main.'

Problem 3: Take numbers as input from the user and print whether the given number is odd or even. You have to iterate the process until user press "N". If user press "N" terminate your program otherwise for given number print whether it is odd or even.

Step 1: Initialization

Step 2: Start of the code section

- Begin the 'code' section.
- Define the 'main' procedure.

Step 3: Input Loop

- Display a message asking the user to enter a number or press 'n' to exit.
- Read a character from the user input.

Step 4: Check for Exit

- Compare the input character to 'n.' If 'n' is pressed, jump to the exit label.

Step 5: Convert Input

- Convert the ASCII input character to its binary (numerical) equivalent by subtracting '0' from it.

Step 6: Check for Odd or Even

Use a bitwise 'and' operation to check if the number is odd or even.

If the least significant bit is set (result is non-zero), it's an odd number. Display the "Odd" message.

-If the least significant bit is not set (result is zero), it's an even number. Display the "Even" message.

Step 7: Display Newline

- Display a newline character (carriage return and line feed) to separate the result from the next input.

Step 8: Repeat Input Loop

- Go back to the input loop to allow the user to enter another number or exit.

Step 9: Exit Program

- Display a newline to ensure a clean output.
- Terminate the program using an appropriate DOS interrupt.

4.IMPLEMENTATION

Problem 1:

```
01 org 100h
02 .stack 1000
03 .model small
04 .data
05
06 .code
07
08 main proc
09     mov ax, @data
10     mov ds, ax
11
12     mov ax, 1
13     mov cx, 50
14     xor dx, dx
15
16 Label1:
17     add dx, ax
18     add ax, 2
19     loop Label1
20
21 main endp
22 end main
```

Problem 2:

```
01 org 100h
02 .stack 1000
03 .model small
04 .data
05     n db ?
06     result dw 1
07     msg1 db "Enter the number: $"
08     next db 10, 13, "$"
09
10 .code
11
12 main proc
13     mov ax, @data
14     mov ds, ax
15
16     mov ah, 09
17     lea dx, msg1
18     int 21h
19
20     mov ah, 01
21     int 21h
22
23     sub al, 48
24     mov cl, al
25     mov ax, 1
26
27 Label1:
28     mul cx
29     mov dx, ax
30     loop Label1
31
32     mov ah, 2
33     int 21h
34
35 main endp
36
37 end main
38
```

Problem 3:

```
01 org 100h
02 .model small
03 .stack 100h
04 .data
05     user_input db ?
06     msg db "Enter a number (N to exit): $"
07     even_msg db " -:Even$"
08     odd_msg db " -:Odd$"
09     newline db 13, 10, "$"
10
11 .code
12 main proc
13     mov ax, @data
14     mov ds, ax
15
16 input_loop:
17     mov ah, 9
18     lea dx, msg
19     int 21h
20
21     mov ah, 1
22     int 21h
23
24     cmp al, 'n'
25     je exit_program
26
27     ; Convert ASCII input to binary (numerical value)
28     sub al, '0'
29
30     test al, 1
31     jz is_even
32
33 is_odd:
34     mov ah, 9
35     lea dx, odd_msg
36     int 21h
37
38     mov ah, 9
39     lea dx, newline
40     int 21h
41     jmp input_loop
42
43 is_even:
44     mov ah, 9
45     lea dx, even_msg
46     int 21h
47
48     mov ah, 9
49     lea dx, newline
50     int 21h
51     jmp input_loop
52
53 exit_program:
54     mov ah, 9
55     lea dx, newline
56     int 21h
57
58     mov ah, 4Ch
59     int 21h
60
61 main endp
62 end main
63
```

5.TEST RESULT / OUTPUT

Problem 1 Output:

The screenshot shows an emulator window titled "emulator: LR4_t1.com" with a menu bar (file, math, debug) and a toolbar (Load, re). A "registers" panel on the left lists registers AX through DI with their high (H) and low (L) byte values. The "extended value viewer" window is open, showing the selected register "DX" with the "word" format selected. It displays the register's value in hexadecimal, binary, octal, and decimal (8-bit and 16-bit) formats.

registers	H	L
AX	00	65
BX	00	00
CX	00	00
DX	09	C4
CS	07 00	
IP	0129	
SS	07 00	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	

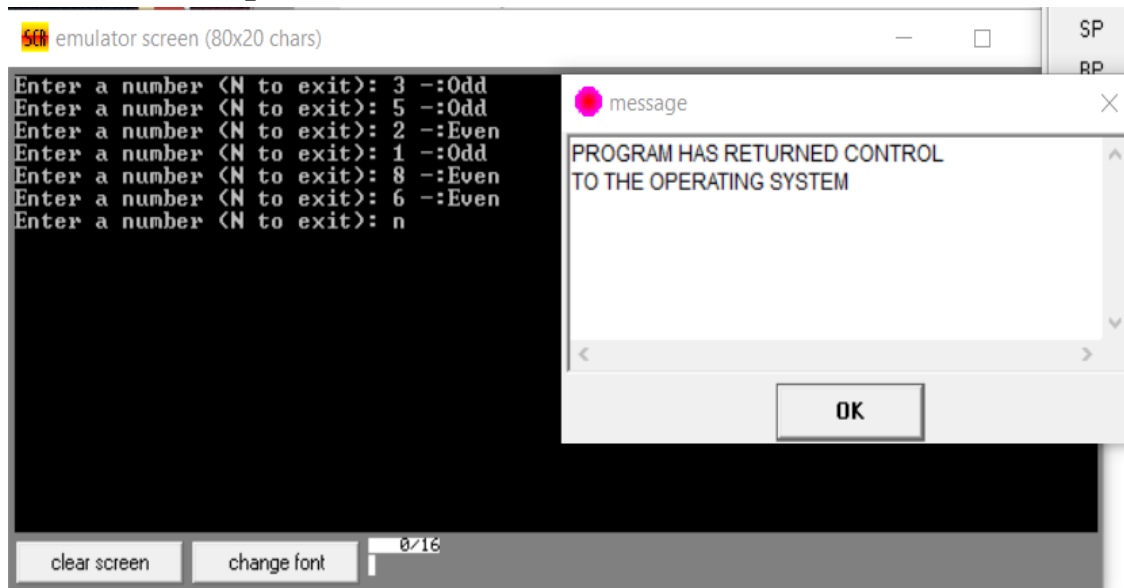
	H	L
hex:	09	C4
bin:	00001001	11000100
oct:	011	304
decimal 8 bit		
unsigned:	9	196
signed:	9	-60
ascii:		-
decimal 16 bit		
unsigned:	2500	
signed:	2500	

Problem 2 Output:

The screenshot shows an emulator window titled "emulator screen (80x25 chars)" with a black screen displaying "Enter the number: 5x". A toolbar at the bottom has "clear screen" and "change font" buttons. The "extended value viewer" window is open, showing the selected register "DX" with the "word" format selected. It displays the register's value in hexadecimal, binary, octal, and decimal (8-bit and 16-bit) formats.

	H	L
hex:	00	78
bin:	00000000	01111000
oct:	000	170
decimal 8 bit		
unsigned:	0	120
signed:	0	120
ascii:		x
decimal 16 bit		
unsigned:	120	
signed:	120	

Problem 3 Output:



The screenshot shows an emulator window titled "emulator screen (80x20 chars)". The main display area contains the following text:

```
Enter a number (N to exit): 3 -:Odd
Enter a number (N to exit): 5 -:Odd
Enter a number (N to exit): 2 -:Even
Enter a number (N to exit): 1 -:Odd
Enter a number (N to exit): 8 -:Even
Enter a number (N to exit): 6 -:Even
Enter a number (N to exit): n
```

Overlaid on the right side of the emulator is a "message" dialog box with the text:

PROGRAM HAS RETURNED CONTROL
TO THE OPERATING SYSTEM

The dialog box has an "OK" button at the bottom. At the bottom of the emulator window, there are buttons for "clear screen" and "change font", and a status bar showing "0/16".

6. ANALYSIS AND DISCUSSION

In this lab report, we will analyze and reflect upon the three assembly language problems based on loops.

In Problem 1,

- This program is designed to compute the sum of the first 50 odd numbers.
- It uses the 'loop' instruction for repetitive addition, starting from 1.

Problem 2,

- This program calculates the factorial of a number input by the user.
- It then uses a loop to calculate the factorial of the input number.

Problem 3,

- This program takes numbers as input from the user.
- It iterates the process until the user presses "n" to exit.
- If the user enters a number, it checks whether it's odd or even and displays the result.
- The program continues to prompt for input until the user exits.

7. SUMMARY:

In all three problems, the programs follow a similar structure, which includes displaying prompts for user input, processing the input, and displaying the output based on specific conditions. These problems demonstrate how assembly language can be used for iterate.