



Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: (Fall, Year:2023), B.Sc. in CSE (Day)

LAB REPORT NO: 01
Course Title: Compiler Lab
Course Code: CSE-304 Section:213-D1

Lab Experiment Name: Introduction to assembly program structure and various arithmetic operations.

Student Details

Name		ID
1.	Irteja Mahmud	213902016

Lab Date : 12-10-2023
Submission Date : 19-10-2023
Course Teacher's Name : Sudip Ghoshal

Lab Report Status

Marks:
Comments:.....

Signature:.....
Date:.....

1. TITLE OF THE LAB EXPERIMENT

Introduction to assembly program structure and various arithmetic operations.

2. OBJECTIVES/AIM

The objective of this program is to know about assembly program structure and various arithmetic operations.

3.PROCEDURE / ANALYSIS / DESIGN

Problem 1: Take a double-digit number input from the user.

Step 1: Initialize the program and memory.

- Set the program's origin to 100h.
- Define the memory model as "small."
- Allocate a stack of 100h bytes.

Step 2: Define data section.

Step 3: Start of the code section.

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

Step 4: Setup data segment and input prompt.

- Load the data segment address into AX from '@DATA'.
- Set DS (data segment) to the value in AX.
- Set AH to 9 (print string function).
- Load the offset address of 'msg' into DX.
- Trigger an interrupt 21H to display the message.

Step 5: Read a character from the user (1st digit).

- Set AH to 1 (read character function).
- Trigger interrupt 21H to read a character from the user.
- Store the read character in BL.

Step 6: Read a character from the user (2nd digit).

- Set AH to 1 (read character function).
- Trigger interrupt 21H to read a character from the user.
- Store the read character in BH.

Step 7: Output newline characters.

- Set AH to 2 (display character function).
- Load DL with ASCII code 13 (carriage return).
- Trigger interrupt 21H to display a carriage return.

Step 8: Output newline characters.

- Set AH to 2 (display character function).
- Load DL with ASCII code 10 (line feed).
- Trigger interrupt 21H to display a line feed.

Step 9: Display the 1st digit entered by the user.

- Set AH to 2 (display character function).
- Load DL with the value in BL.
- Trigger interrupt 21H to display the character.

Step 10: Display the 2nd digit entered by the user.

- Set AH to 2 (display character function).
- Load DL with the value in BH.
- Trigger interrupt 21H to display the character.

Step 11: End of the main procedure.

- End the 'main' procedure using 'main endp'.

Step 12: End of the program.

- Mark the end of the program using 'end main'.

Problem 2: Convert 260°C to Fahrenheit using the following expression and store in a F variable: $^{\circ}\text{F} = ^{\circ}\text{C} \times 9/5 + 32$

Step 1: Initialize the program and memory.

Step 2: Define data section.

- Declare variables:

Step 3: Start of the code section.

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

Step 4: Setup data segment and input prompt.

- Load the data segment address into AX from '@DATA'.
- Set DS (data segment) to the value in AX.
- Set AH to 9 (print string function).
- Load the offset address of 'msg' into DX.
- Trigger an interrupt 21H to display the message.

Step 5: Display the initial temperature in Celsius.

- Set AH to 2 (display string function).
- Load DX with the address of 'tc'.

- Trigger interrupt 21H to display the value of 'tc'.

Step 6: Convert Celsius to Fahrenheit.

- Perform the Fahrenheit conversion using the formula: $f = c * 9 / 5 + 32$.
- Store the result in 'tf'.

Step 7: Output a newline and carriage return.

- Set AH to 9 (print string function).
- Load DX with the address of 'next'.
- Trigger interrupt 21H to display a newline and carriage return.

Step 8: Display the message for Fahrenheit temperature.

- Set AH to 9 (print string function).
- Load DX with the address of 'msg1'.
- Trigger interrupt 21H to display the message.

Step 9: Display the converted temperature in Fahrenheit.

- Set AH to 2 (display string function).
- Load DX with the address of 'tf'.
- Trigger interrupt 21H to display the value of 'tf'.

Step 10: End of the main procedure.

- End the 'main' procedure using 'main endp'.

Step 11: End of the program.

- Mark the end of the program using 'end main'.

Problem 3: Convert 1000 °F (Fahrenheit) to °C (Celsius) using the following expression and store in a C variable: $^{\circ}\text{C} = (^{\circ}\text{F} - 32) \cdot 5/9$

Step 1: Initialize the program and memory.

Step 2: Define data section.

- Declare variables:

Step 3: Start of the code section.

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

Step 4: Setup data segment and input prompt.

- Load the data segment address into AX from '@DATA'.
- Set DS (data segment) to the value in AX.
- Set AH to 9 (print string function).
- Load the offset address of 'msg1' into DX.
- Trigger an interrupt 21H to display the message.

Step 5: Display the initial temperature in Fahrenheit.

- Set AH to 2 (display string function).
- Load DX with the address of 'tf'.
- Trigger interrupt 21H to display the value of 'tf'.

Step 6: Convert Fahrenheit to Celsius.

- Perform the Celsius conversion using the formula: $c = (f - 32) * 5 / 9$.
- Store the result in 'tc'.

Step 7: Output a newline and carriage return.

- Set AH to 9 (print string function).
- Load DX with the address of 'next'.
- Trigger interrupt 21H to display a newline and carriage return.

Step 8: Display the message for Celsius temperature.

- Set AH to 9 (print string function).
- Load DX with the address of 'msg'.
- Trigger interrupt 21H to display the message.

Step 9: Display the converted temperature in Celsius.

- Set AH to 2 (display string function).
- Load DX with the address of 'tc'.
- Trigger interrupt 21H to display the value of 'tc'.

Step 10: End of the main procedure.

- End the 'main' procedure using 'main endp'.

Step 11: End of the program.

- Mark the end of the program using 'end main'.

4.IMPLEMENTATION

Problem 1:

```
01 org 100h
02 .model small
03 .stack 100h
04 .data
05     var dw 100
06     msg dw "Enter A Number: $"
07 .code
08     main proc
09
10     MOV AX, @DATA
11     MOV DS, AX
12
13     MOV AH, 9
14     LEA DX, MSG
15     INT 21H
16
17     MOV AH, 1
18     INT 21H
19     MOV BL, AL
20
21     MOV AH, 1
22     INT 21H
23     MOV BH, AL
24
25     MOV AH, 2
26     MOV DL, 013
27     INT 21H
28
29     MOV AH, 2
30     MOV DL, 010
31     INT 21H
32
33     MOV AH, 2
34     MOV DL, BL
35     INT 21H
36
37     MOV AH, 2
38     MOV DL, BH
39     INT 21H
40
41     main endp
42 end main
43
```

Problem 2:

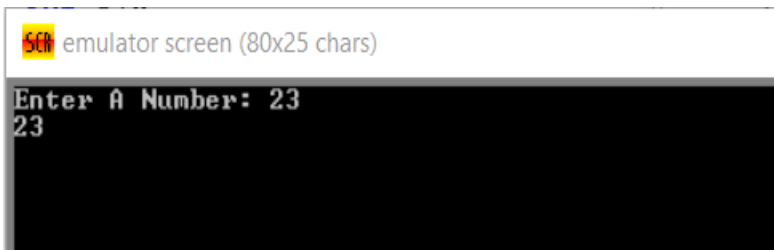
```
01 org 100h
02 .model small
03 .stack 100h
04 .data
05     var dw 100
06     msg dw "Temparature In Celcius: $"
07     msg1 dw "Temparature In Fahrenheit: $"
08     tc dw 260      ; t celsius.
09     tf dw 0        ; t fahrenheit.
10     next db 013,010"$"
11 .code
12     main proc
13
14     MOV AX, @DATA
15     MOV DS, AX
16
17     MOV AH, 9
18     LEA DX, MSG
19     INT 21H
20
21     MOV AH, 2
22     MOV DX, TC
23     INT 21H
24
25     ; convert celsius to fahrenheit according
26     ; to this formula: f = c * 9 / 5 + 32
27
28     MOV cx, tc
29     MOV ax, 9
30     IMUL cx
31     MOV cx, 5
32     IDIV cx
33     ADD ax, 32
34     MOV tf, ax
35
36     MOV AH, 9
37     LEA DX, next
38     INT 21H
39
40     MOV AH, 9
41     LEA DX, MSG
42     INT 21H
43
44
45
46     MOV AH, 2
47     MOV DX, Tf
48     INT 21H
49
50     main endp
51 end main
```

Problem 3:

```
01 org 100h
02 .model small
03 .stack 100h
04 .data
05     var dw 100
06     msg dw "Temperature In Celcius: $"
07     msg1 dw "Temperature In Fahrenheit: $"
08     tc dw 0 ; t celsius.
09     tf dw 1000 ; t fahrenheit.
10     next db 013,010,"$"
11 .code
12     main proc
13
14     MOV AX, @DATA
15     MOV DS, AX
16
17     MOV AH, 9
18     LEA DX, MSG1
19     INT 21H
20
21     MOV AH, 2
22     MOV DX, Tf
23     INT 21H
24
25     ; convert fahrenheit to celsius according
26     ; to this formula: c = (f - 32) * 5 / 9
27
28     MOV cx, tf
29     SUB cx, 32
30     MOV ax, 5
31     IMUL cx
32     MOV cx, 9
33     IDIV cx
34     MOV tc, ax
35
36     MOV AH, 9
37     LEA DX, next
38     INT 21H
39
40     MOV AH, 9
41     LEA DX, MSG
42     INT 21H
43
44
45
46     MOV AH, 2
47     MOV DX, Tc
48     INT 21H
49
50     main endp
51 end main
```

5.TEST RESULT / OUTPUT

Problem 1 Output:



```
SCHEM emulator screen (80x25 chars)
Enter A Number: 23
23
```


Problem 2 Output:

The screenshot shows an x86 emulator interface. The assembly code window displays the following code:

```

02 .model small
03 .stack 100h
04 .data
05     var dw 100
06     msg dw "Temparature In Celcius: $"
07     msg1 dw "Temparature In Fahrenheit: "
08     tc dw 260 ; t celsius.
09     tf dw 0 ; t fahrenheit.
10     next db 013,010,"$"
11 .code
12     main proc
13
14     MOV AX, @DATA
15     MOV DS, AX
16
17     MOV AH, 9

```

The registers window shows the following values:

Register	H	L
AX	02	F4
BX	00	00
CX	00	05
DX	01	F4
CS	0700	
IP	0192	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

The extended value viewer shows the watch register DX with the following values:

Unit	Value
hex	01
bin	00000001
oct	001
decimal 8 bit unsigned	1
decimal 8 bit signed	1
ascii	@
decimal 16 bit unsigned	500
decimal 16 bit signed	500

The emulator screen (80x25 chars) shows the output:

```

Temparature In Celcius: 26
Temparature In Celcius: f

```

The message window shows the message: "the emulator is halted."

Problem 3 Output:

The screenshot shows an x86 emulator interface. The assembly code window displays the following code:

```

06     msg dw "Temparature In Celcius: $"
07     msg1 dw "Temparature In Fahrenheit: "
08     tc dw 0 ; t celsius.
09     tf dw 1000 ; t fahrenheit.
10     next db 013,010,"$"
11 .code
12     main proc
13
14     MOV AX, @DATA
15     MOV DS, AX
16
17     MOV AH, 9

```

The registers window shows the following values:

Register	H	L
BX	00	00
CX	00	09
DX	02	19
CS	0700	

The extended value viewer shows the watch register DX with the following values:

Unit	H	L
hex	02	19
bin	00000010	00011001
oct	002	031
decimal 8 bit unsigned	2	25
decimal 8 bit signed	2	25
ascii	@	↓
decimal 16 bit unsigned	537	
decimal 16 bit signed	537	

The emulator screen (80x25 chars) shows the output:

```

Temparature In Fahrenheit: 2
Temparature In Celcius: 4

```

The message window shows the message: "the emulator is halted."

6.ANALYSIS AND DISCUSSION

In this lab report, the provided assembly code problem is a simple program that demonstrates temperature conversion between Fahrenheit to Celsius. The program is written in 8086 assembly language and is run in emu8086. The program begins by setting the origin point to 100h, indicating the memory location where it will start executing. It defines a small memory model and allocates a stack of 100h bytes for program operations. The data section defines various data items. The code section starts by defining the 'main' procedure. It initializes the data segment and displays a prompt message.

7. SUMMARY:

This code is designed for execution in a emu8086 environment, and it can serve as an educational example of data manipulation and arithmetic operations in assembly language. It demonstrates the conversion formula and the use of data segments and interrupts for input and output operations.