



**Lab Manual**

**Computer Organization and  
Assembly Language Lab**

**(CMP-224)**

**Faculty of Computing & Information Technology (FCIT)**

**University of the Punjab, Lahore.**

**[www.pucit.edu.pk](http://www.pucit.edu.pk)**

**Computer Organization and Assembly  
Language Lab  
(CMP-224)**

**LAB MANUAL**

Punjab University College of Information Technology  
University of the Punjab

# Table of contents

Week	Topics
Week 1	<b>DOS Debugger I:</b> <b>Commands Q,R,D,E,F,C,M,S,H</b> , to view RAM memory and Processor's Registers
Week 2	<b>DOS Debugger II:</b> <b>Commands A,U,L,W,N,T,P,G,I,O</b> , to write and execute a program in debugger.
Week 3	<b>Assembly Language Program:</b> Syntax, Variables (declaration, initialization and storage in memory).Use of MASM6.11
Week 4	<b>Data Transfer instructions</b> like mov and exchg also <b>Arithmetic</b> instructions like add, sub, inc, dec, neg. <b>Flag Register</b>
Week 5	<b>Control transfer instructions</b> like LOOP, JMP and conditional Jumps.
Week 6	<b>High Level structures</b> like decision, looping, and , or conditions using control transfer instructions.
Week 7	<b>Logical, Shift and rotate Instructions</b> and its use in input and output of BINARY and HEXADECIMAL Numbers.
Week 8	<b>MID LAB EXAM</b>
Week 9	<b>Procedures and use of stack</b>
Week 10	<b>Mul and Div Instructions</b> and its use in input and output of DECIMAL Numbers.
Week 11	<b>String Instructions</b>
Week 12	<b>Disk and File Operations</b>
Week 13	<b>Graphics</b>
Week 14	<b>BIOS and DOS Interrupts</b>
Week 15	<b>Linking to High Level Languages</b>
Week 16	<b>FINAL LAB EXAM</b>

## Lab 02

### Task-1

---

Copy the contents of memory location 100 to 107 at location 108 to 10F and find out the difference between the contents of memory blocks of 102 to 105 and 112 to 115 using compare command.

### Task-2

---

**Write an assembly program which should:**

- Place 1234 in AX
- Copy the value of AX into DX
- Copy the contents of DX at memory location 102
- Copy the contents of memory location 102 in BX
- Change the value of CL to 23

### Task-3

---

**Write an assembly program which should:**

- Change the contents of AX to 1234
- Add 1 to the contents of AX
- Copy the value of AX to DX
- Subtract 1233 from the value of DX
- Do BH = DL
- Place 9 in AL
- Divide AL with 2
- Copy the quotient of above division at memory location 100
- Move the remainder of above division in DL
- Save the program at your Z drive with name Lab2\_T2.bin
- Quit the debugger
- Reopen the debugger and load your program
- Unassemble your program
- Execute the instructions by using Trace command.

## Lab 05

### Task-1

Write a program which should accomplish the following tasks:

- AX = 2004
- BX = 0002
- Add contents of AX and BX and place result in AX

None of the following statements should be present in your code:

- MOV AX, 2004
- ADD AX, BX

### Task-2

Write a program in which:

- Declare two variables ALPHA and BETA.
- Initialize these variables with 2F and 9D respectively.
- Now swap the contents of ALPHA and BETA without using another variable.

### Task-3

Suppose variable VarA is located at memory address **DS:xxxx** (where x can be any value) then VarB would must be located at **DS:xxxx+1**. Like this find out the address of all variables in the following code and verify your results by debugging the following program.

```
.MODEL SMALL
.STACK 100H
.DATA
    VARA BYTE 10      ;address is DS:xxxx
    VARB BYTE 0BH     ;address is DS:xxxx + 1
    VARC WORD ?
    VARD SBYTE ?
    VARE DWORD ?
    ARR BYTE 20 DUP(?)
    VARF SWORD 010B
    ARRB WORD 10 DUP(?)
    VARZ BYTE 0
.CODE
MAIN PROC
    MOV AX, @DATA
    MOV DS, AX
MAIN ENDP
END MAIN
```

**Lab 06**

No Evaluation  
after Lab

**Task-1**

---

You've following data area in your code:

**.DATA**

```
STR  BYTE "Assembly Language Programmers can play with machines.", "$"  
ARR  BYTE 256 DUP(?)  
TEMP BYTE ?
```

Complete this program which should:

- Display STR string using INT 21H
- Display contents (garbage) of ARR in next line using INT 21H
- Copy the contents of STR in ARR
- Display the contents of ARR in new line using INT 21H

You're allowed to declare as many variables as you want.

**Task-2**

---

Write a program which should:

- Take two integers (0-9) from user.
- Add them without using ADD instruction or LOOP.
- Store their sum in AX
- Store the 2's complement of first integer in BX (without using NEG operator)

# Computer Organization and Assembly language Programming

**Lab5**

**Total Marks:100**

## **Task-1**

**[5+5+10]**

Write an assembly program to

- a) Prompt user by displaying appropriate message for taking a character
- b) Take an input character from user
- c) Print 5×5 solid grid of that character using conditional and looping structures

## **Task-2**

**[5+5+20]**

Write a program to

- a) Prompt a user by displaying appropriate message for taking four alphabets
- b) Take alphabets and store them in four variables already declared in data area
- c) Show them on next line in alphabetical order

## **Task-3**

**[5+15+10]**

Write a program which prompts the user to press any key. When user presses some key it should check and tell the user whether they key is alphabet, a digit an asterisk sign, dollar sign, percentage sign or some other symbol. You've to map the switch and if-else structure in this program.

Pseudo Code of this program is given below for your convenience:

**CHECK\_KEY ()**

**Begin**

**Print "Press any key: "**

**Take Input**

**If input (ASCII) is between 65 and 90 OR input is between 90 and 122 then**

**Print "You've entered an alphabet".**

**Else if input (ASCII) is between 48 and 57 then**

**Print "You've entered a digit".**

**Else**

**switch( input )**

**case „#“:**

**Print "You've entered hash sign."**

**break;**

**case „\$“**

**Print "You've entered a dollar sign."**

## **Computer Organization and Assembly language Programming**

**break;**

**case „%“:**

**Print “You’ve entered a percentage sign.”**

**break;**

**case „\*“:**

**Print “You’ve entered an asterisk sign.”**

**break;**

**default:**

**Print “You’ve entered a symbol.”**

**End if**

**End**

Note: “You have to use unconditional jump in this task also”

### **Task-4**

**[20]**

Write a program which prompts the user to enter some alphabet; change its case and display it on screen i.e. if user enters uppercase alphabet then it should convert it into lowercase and vice versa. You’re not allowed to use SUB instruction.

#### **Sample Execution**

Enter an alphabet: S

Output: s

If you want to proceed[Y/N]: Y

Enter an alphabet: k

Output: K

If you want to proceed[Y/N]: N



**Lab 08****Task-1**

Write a program which should prompt the user to press some key and print its ASCII in binary format.

```
Press any key: Z
ASCII: 1011010
```

**Task-2**

Write a program which should prompt the user to enter a hex integer (up to 4 digits and press ENTER to terminate string input); add all the digits of user given integer and display their sum in hexadecimal format.

```
Enter a hex integer: 3CB
Sum of digits: 1A
```

**Task-3**

Write a program which should take two binary numbers from user (up to 7 digits); add them; display their sum in binary format. Now consider the sum as ASCII of some character. Display the equivalent character having ASCII equal to the sum.

```
Enter a binary number: 101010
Enter a binary number: 0100100
Sum = 1001110
Character = N
```

## Lab 09

### Task-1

Write a procedure **PRINT\_HEX**:

**Functionality:** Prints a hexadecimal number (up to two digits).

**Parameters:** Hex number in stack.

**Returns:** *Nothing*

**Pre-condition:** Value to be printed (hex number) is pushed in stack.

### Task-2

Write a procedure **TAKE\_HEX**:

**Functionality:** Takes a hexadecimal number as input (up to 2 digits) and stores it in DL.

**Parameters:** *No Parameter*

**Returns:** A hexadecimal number in DL register.

**Pre-condition:** *Nothing is required as pre condition*

### Task-3

Write a procedure **PRINT\_DIAMOND**:

**Functionality:** Prints a diamond on console using asterisk sign '\*'

**Parameters:** Maximum number of asterisk signs in AL register.

**Returns:** Status in DL (1 on success and 0 on error)

**Pre-condition:** AL should contain an odd value between 1 and 9 (inclusive).

Write a program; take input in main procedure and pass it to PRINT\_DIAMOND to print a diamond on console:

- Data Segment should not exceed than one byte.
- Write some helping procedures like **TAKE\_INPUT**, **ENDL**, **PRINT\_SPACES**, **PRINT\_ASTERIKS**, etc~
- Hold the output on console until user presses some key.

```
Enter an odd number between 0 and 10: 9
```

```
  *
 ***
*****
*****
*****
*****
 *****
  *****
   *****
    *****
     *
```

## Lab 10

### Task-1

Write a procedure **TAKE\_DECIMAL**:

**Functionality:** Takes a decimal number input up to 2 digits and stores it in DL.

**Parameters:** *No Parameter*

**Returns:** A decimal number in DL register.

**Pre-condition:** *Nothing*

### Task-2

Write a program which should take a string from user in the following format:

**operand1**<space>**operator**<space>**operand2** <enter>

e.g.:

**2+4**

After taking input your program should evaluate the user given expression and display the result on screen.

Each operand is a decimal number of maximum two digits. The answer should be displayed as a decimal number.

Valid operators for your program are:

Operator	Purpose
+	Addition
-	Subtraction
*	Multiplication
%	Mod

After displaying the result your program should prompt for another expression until user enters 'x'.

See the following sample execution:

```
Enter an expression to evaluate: 2 + 4
Answer: 6
Enter an expression to evaluate: 9 % 2
Answer: 1
Enter an expression to evaluate: x
```

## Lab 11

### Task-1

---

Write a program that should take string input; store it in reverse order into another string and then display both strings.

To get full credit, write all procedures required in this task e.g. **STR\_INPUT**, **STR\_OUTPUT**, **STR\_REV**, etc...

```
Enter a string: We are going on vacations.  
Original String: We are going on vacations.  
Reversed String: .snoitacav no gniog era eW
```

### Task-2

---

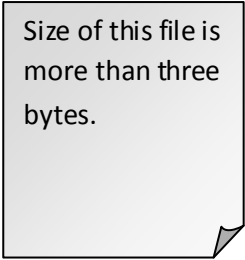
Write a program that should take two inputs i.e. a character and a string and display the number of occurrences of given character in given string in decimal format.

```
Enter a string: Hello COAL is very easy.  
Enter a character: y  
Number of occurrences: 2
```

## Lab 12

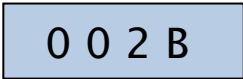
### Task-1

Write a program which should store the size of "Data.txt" (in bytes) in CX. You should not traverse the whole file to find out its size or in other words you should not use a counter to accomplish this task.



Size of this file is  
more than three  
bytes.

Data.txt

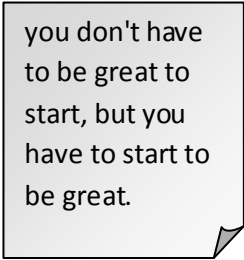


002B

CX (Hex)

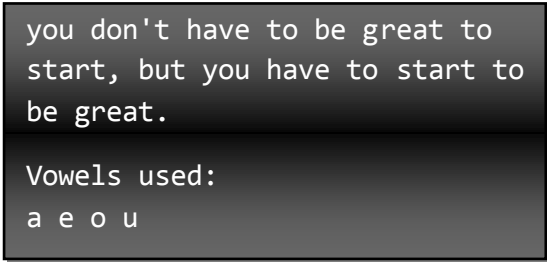
### Task-2

Write a program which should display the contents of "Data.txt". In the next last line the program should display the vowels found in the same file. Each vowel (which is found) should be displayed only once. Assume that all the text in this file is in lowercase.



you don't have  
to be great to  
start, but you  
have to start to  
be great.

Data.txt



you don't have to be great to  
start, but you have to start to  
be great.

Vowels used:

a e o u

### Task-3

Write a program which should ask the user to enter a string and press enter to terminate. The program should store the user given string in a file "User.txt".

**Note:** You are not allowed to restrict the user to enter a string of some specific length i.e. user is free to enter the string as much long as he wants.

## Lab 13

### Task-1

---

Write a procedure DRAW\_LINE. The given parameters are:

AL = Line Color

CX = Row Number or Y

DX = Column Number or X

The procedure should draw line according to the given parameters when called.

### Task-2

---

Write a procedure DRAW\_RECT. The given parameters are:

AX = Row Number or Y

BX = Column Number or X

CX = Height

DX = Width

Line Color is pushed in stack.

The procedure should draw a rectangle shape according to given attributes.

### Task-3

---

Write a procedure FILL\_RECT. The given parameters are:

AX = Row Number or Y

BX = Column Number or X

CL = Boundary Color

CH = Fill Color

The procedure should fill all the pixels with given fill color starting from given AX and BX until the boundary color is not encountered.