Computer Organization
And
Assembly Language

```
00000000: 01001101 01011010 10010000 00000000 00000011 00000000  MZ····
00000006: 00000000 00000000 00000100 00000000 00000000 00000000  ······
0000000c: 11111111 11111111 00000000 00000000 10111000 00000000  ······
00000012: 00000000 00000000 00000000 00000000 00000000 00000000  ······
00000018: 01000000 00000000 00000000 00000000 00000000 00000000  @·····
0000001e: 00000000 00000000 00000000 00000000 00000000 00000000  ······
00000024: 00000000 00000000 00000000 00000000 00000000 00000000  ······
0000002a: 00000000 00000000 00000000 00000000 00000000 00000000  ······
00000030: 00000000 00000000 00000000 00000000 00000000 00000000  ······
00000036: 00000000 00000000 00000000 00000000 00000000 00000000  ······
0000003c: 10000000 00000000 00000000 00000000 00001110 00011111  ······
00000042: 10111010 00001110 00000000 10110100 00001001 11001101  ······
00000048: 00100001 10111000 00000001 01001100 11001101 00100001  !··L·!
0000004e: 01010100 01101000 01101001 01110011 00100000 01110000  This p
00000054: 01110010 01101111 01100111 01110010 01100001 01101101  rogram
0000005a: 00100000 01100011 01100001 01101110 01101110 01101111   canno
00000060: 01110100 00100000 01100010 01100101 00100000 01110010  t be r
00000066: 01110101 01101110 00100000 01101001 01101110 00100000  un in 
0000006c: 01000100 01001111 01010011 00100000 01101101 01101111  DOS mo
00000072: 01100100 01100101 00101110 00001101 00001101 00001010  de.···
00000078: 00100100 00000000 00000000 00000000 00000000 00000000  $·····
0000007e: 00000000 00000000 01010000 01000101 00000000 00000000  ··PE··
```

<u>**Lab Manual 05**</u>

Objectives:

1. Revision of previous lab
2. Understanding procedure calling
3. Understanding working of stack while procedure calling
4. To learn how can we pass arguments to the procedures(basic)

Lab Instructor:

Mr. Tariq Mehmood Butt tariq.butt@pucit.edu.pk

Teaching Assistants:

Hafiz Muhammad Ahmad	bcsf21m502@pucit.edu.pk
Syed Muhammad Zain Raza	bcsf21m510@pucit.edu.pk
Zahra Malik	bcsf21m551@pucit.edu.pk
Bilal	bsdsf21m022@pucit.edu.pk

LAB TASKS

You are required to prepare the ASM file of each task and run the executable file in the debugger.

Task 1: Code and assemble following code snippet.

```
.MODAL SMALL
.STACK 100H

.CODE

        MAIN PROC
                MOV AX, @DATA
                MOV DS, AX

                ;SOME CODE

                MOV AH, 4CH
                INT 21H

        MAIN ENDP

.DATA

VAR1 DB 23, 45, 56, 67
VAR2 DB 89, 34, 67, 56

END MAIN
```

Open the executable in debug. Note down the values of CS, DS, SS, ES, IP, SP registers after execution of 2nd instruction.

Task 2: Copy the first task code and Change the first line. To

```
        .MODAL LARGE
        .MODAL TINY
```

and do the Same

→ What changes did you note.
→ Can you tell why this happens?

Task 3: Copy the first file and change 2nd line to

```
        .STACK (any value of your choice from 50h to 150h)
```

→ Do the same as you have done in task 1 and 2.
→ Repeat two times by giving different values
→ What changes do you note.
→ Can you tell why this happens?

Task 4: Code and assemble the following

```
.MODAL SMALL
.STACK 100H

.CODE

        MAIN PROC
                MOV AX, @DATA
                MOV DS, AX

                CALL TAKE_INPUT

                MOV DL, AL

                CALL  GIVE_OUTPUT

                MOV AH, 4CH
                INT 21H

        MAIN ENDP

        TAKE_INPUT PROC
                MOV AH, 01
                INT 21H
                RET
        TAKE_INPUT  ENDP

        GIVE_OUTPUT PROC
                MOV AH, 02
                INT 21
        GIVE_OUTPUT ENDP

.DATA

VAR1 DB 23, 45, 56, 67
VAR2 DB 89, 34, 67, 56

END MAIN
```

→ Open the executable in debug mode and Trace it.
→ What happens to SP register on every CALL instruction.
→ What does the value of stack (memory location to which SP is pointing) shows.(**Hint**: View by by dumping the memory by d command).
→ What happens with SP when RET instruction is executed.
→ What will happen if someone forgets RET instruction at the end of any procedure. (**Hint:** Try doing this).

Task 05: Write a program which contains

→ Declaration of two strings in data section

1. Your name
2. Your Roll Number

Note: Name and roll numer must be $ended.
→ procedure which prints a string whose address is stored in DX register.
→ main procedure prints name and roll number by calling the above procedure.
Note: Store offset of name in DX and call the procedure you have declared. Than do similar for the roll number.

Task 06: Write a program which contains:
→ Three randomly arrays of 8 bytes declared in data section
→ A procedure which takes starting addresses of three arrays (BX, SI, and DI registers) of 8 bytes. Adds first two and stores in third array.
→ Main procedure which stores offsets of source arrays in SI and BX, and destination array in DI register. Than calls the adding procedure.
→ Assemble and execute in debug and verify the final results.