**Daffodil International University**

**Department of Electrical and Electronic Engineering**

**EEE 324: Microprocessors and Interfacing Laboratory**

**Experiment No: 02**

**NAME OF THE EXPERIMENT: Familiarization with the stack memory addressing mode**

**Objective:** The objective of the experiment are-

1. To be familiar with stack operations.
2. To know how data are stored and removed from stack memory.

**APPARATUS:**

1. PC with Windows XP.
2. emulator8086 Software.

**THEORY :**

**Stack Memory Addressing:**

* Stack holds the data temporarily and stores the return addresses used by the procedures.
* The stack memory is an LIFO (Last in, First out) memory, that describes the way that data are stored and removed from the memory.
* Data are stored into the stack with a **PUSH** instruction and removed with a **POP** instruction.
* The **CALL** instruction also uses the stack to hold the return addresses for procedures and a **RET** (Return) instruction to remove the return address from the stack.
* The stack memory is maintained by 2 registers: Stack Pointer (SP) and Stack Segment (SS)
* Whenever data is pushed onto the stack, the higher 8 bits are placed in the location SP-1 and the lower 8 bits are placed in the location addressed by SP-2
* Whenever data is poped from the stack, the lower 8 bits are removed from the location SP and the higher 8 bits are removed from the location addressed by SP+1.

Stack is a segment where some register values can be stored so that it is not lost. In assembly language programming we have only four registers with which we can perform operations. These are AX, BX, CX and DX. But in many problems we may need a lot of registers. So we need to reuse these registers again and again. This can be done by storing present value of a register (suppose AX) in stack segment. Now we can perform other tasks with AX. This will certainly change content of AX. After this getting the value of AX from stack and restoring its previous value. The first job (storing value of AX in stack) is done by PUSH command. Second job (restoring value of AX from stack) is done by POP command in assembly language. A simple program can clarify this.

**PROCEDURE:**

1. Start emulator8086.
2. To write the code click on New and choose code template as empty workspace.
3. Write the following code in the text area.

**Program 01:**

ORG 1000H

MOV AX, 3256H; Stores 3256H in AX. This is AX’s original value

PUSH AX; Puts AX’s value in stack

MOV AX, 1254H; AX is assigned to an intermediate value 1254H

DEC AX; New value of AX is 1253H. Certainly AX’s value is changed.

MOV BX, AX; Value of AX is now copied to BX

POP AX; This restores AX’s previous value and now AX is 3256H

HLT

1. Click on File\Save and save it.
2. Click on Compile\Save\Run. After the operation is completed the emulator will halted.
3. Click on debug and verify the output on debug log.

**OUTPUT:**

AX = 3256H

BX=1253H

1. To use specific starting address of stack memory, we will define it by stack segment (SS) and stack pointer (SP)

**Program 02:**

ORG 1000H

MOV BX, 1000H

MOV SS, BX

MOV SP, 0001H

MOV AX, 2347H

MOV BX, 1254H

PUSH AX

ADD AX, BX

MOV DX, AX;

POP AX

HLT

1. Click on debug and verify the output on debug log and also click on stack to see the final address of stack memory.

**REPORT**:

1. Write an assembly language program to interchange the content of AX and BX register using PUSH and POP instruction.