**PMAS Arid Agriculture University Rawalpindi**

**University Institute of Information Technology**

**LAB MANUAL - VIII**

**Class/Program: BS (CS)** **Course: COAL (CS530)**



**Objectives:**

1. **How to create a nested Loop**
2. **Introduction to nested LOOP Statement**
3. **Use of cx register for loop execution**
4. **Dry run of loop**
5. **How to create a Nested Loop**
6. **Programs**

**How to create a Nested Loop:**

**Creation of a nested loop in assembly language is a little trickier than in C/C++. Here the only variable for the loop iterations is the ecx register. So it is obvious that for each loop the number of iterations must be stored in the ecx register. But on the other hand assembly does provide a better use of CPU stacks. Using stack operations, the ecx values for the outer loops can be stored and accessed after iterating the inner loop. The general syntax for nested loop is given below.**

**Mov (space) ecx, (number of iterations for the outer loop)**

**Outer Code label name:**

**; General statements of the loop to be executed repeatedly PUSH ecx**

**Mov (space) ecx, (number of iterations for the inner loop)**

**Inner Code label name:**

**; General statements of the loop to be executed repeatedly**

**LOOP (space) Inner Code label name**

**Pop ecx**

**LOOP (space) Outer Code label name**

**The PUSH command stores the ecx value for the outer loop. Pop command will retrieve the same value after the inner loop is fully executed.**

loops are basically the same jumps, it is possible to code loops without using the loop instruction, by just using conditional jumps and compare, and this is just what loop does. all loop instructions use **CX** register to count steps, as you know CX register has 16 bits and the maximum value it can hold is 65535 or FFFF, however with some agility it is possible to put one loop into another, and another into another two, and three and etc... and receive a nice value of 65535 \* 65535 \* 65535 ....till infinity.... or the end of ram or stack memory. it is possible store original value of cx register using **push cx** instruction and return it to original when the internal loop ends with **pop cx**, for example:

org 100h

**mov bx, 1**

**mov cx, 5**

**L1:**

**push cx**

**mov cx, bx**

**L2:**

**Mov dl, '\*'**

**mov ah,2**

**int 21h**

**loop L2**

**mov dl,10**

**mov ah, 2**

**int 21h**

**mov dl,13**

**mov ah, 2**

**int 21h**

**inc bl**

**pop cx**

**loop L1**

**mov ah,4ch**

**int 21h**

**main endp**

**end main**

**programs**

**1. Program to print the following pattern using nested loop.**

\*

\*\*

\*\*\*

\*\*\*\*

**2. Program to print the following pattern;**

1

22

333

4444

55555

**3 . Program for calculating Factorial using LOOP statement**

**4. Program to read a character and display that character 100 times on next line using loop**