**Lab # 10**

**Array Processing in x86 Assembly Language**

In programming, a series of objects all of which are the same size and type called “**Array**”. Each object in an array is called an **array element***.* The important characteristics of an array are:

✓

Each element has the same data type (although they may have different values).

◻ The entire array is stored contiguously in memory (that is, there are no gaps between elements).

Arrays can have more than one dimension. A one-dimensional array is called a *vector* ; a two- dimensional array is called a **matrix**. Programs often use arrays to store collections of data values. Loops are commonly used to manipulate the data in arrays. Storage for an array can be reserved using the *DUP* directive in the data segment of a program. For Example:

*array1* DD 25, 47, 15, 50, 32 ; creates an array of 5 doublewords with initial values as specified.

**Declare an array**

array2 DD 1000 DUP (?) ; creates an array of 1000 logically uninitialized doublewords

# Access to array elements

This part is easy similar to other language, you need the right index and then you access your element. That‟s means if you want to access the first element of the array A, you just write: A[0].

# Create a 2 dimensional array

Let‟s assume we have 2 arrays A and B of dimension DIM. We will compute all possible products among values of first array and values of second array, we will put results in a matrix of

DIMxDIM dimension Here is the algorithm I will use:

for each element1 in array A, starting from 0: for each element2 in array B, starting from 0: M[i][j] = element1\*element2

In 8086, there is no real structure for 2 dimensional array in memory.

So we will map our matrix in a linear memory. That‟s mean we declare an array of DiM\*DIM elements and every element of the matrix will be in that array.

The mapping is simple to access to the Jth row and the Kth column we just do: M[J,K] = M[J

\* DIM + K]

The second trick is about the size of every element of the matrix. Since every element is a product of two element of 1 byte , we will use 2 bytes for the size of every element of the matrix. So the mapping formula is now: M[J,K] = M[2 \* (J \* DIM + K)] because every element take one more byte on memory. The result:

1

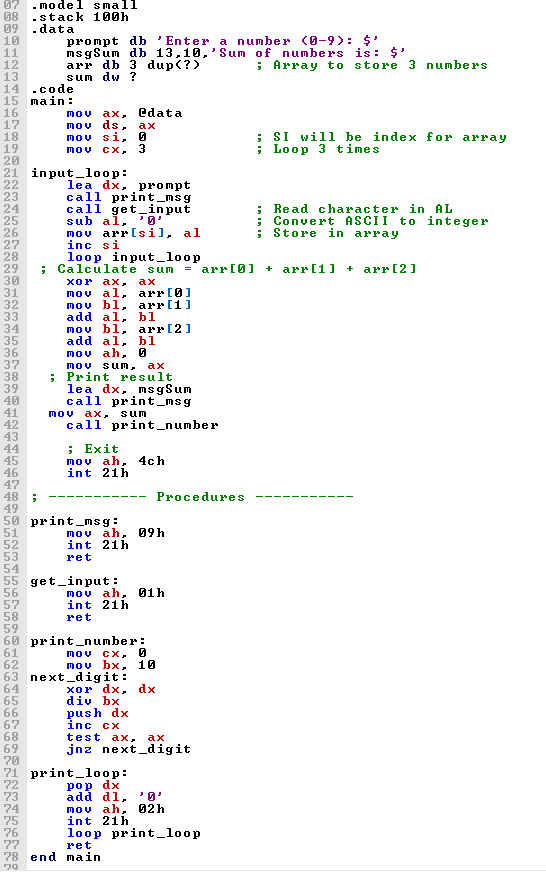
# 2 .MODEL small 3 .STACK 4 .DATA 5 DIM EQU 9A DB 1, 2, 3, 4, 5, 6, 7, 8, 10 6 B DB 11, 12, 13,14, 15, 16, 17, 18, 19 7 M DW (DIM \* DIM) DUP (?)

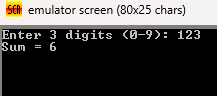
1. CDB?
2. TEMP DW ?
3. TEMP2 DW ?
4. .CODE 12 Main Proc 13 MOV AX, 0 14 MOV CX, DIM 15 MOV DI, 0 16 X: MOV C, DIM1 17 MOV BX, 0 18 Y: MOV AX, 0 19 MOV AL, A[DI] 20 MUL B[BX] 21 ;M[J,K] = M[2 \* (J \* NUM\_COLS + K)] 22 MOV TEMP,AX 23 XOR AX,AX 24 MOV AX,DI 25 MOV TEMP2,BX 26 MOV BX,DIM1 27 MUL BX 28 MOV BX,TEMP2

# 29 ADD AX,BX 30 MOV TEMP2,BX 31 MOV BX,2 32 MUL BX 33 MOV BX,TEMP2 34 MOV TEMP2,DI 35 MOV SI,AX 36 MOV AX,TEMP 37 MOV M[SI],AX 38 MOV AX, 0 39 INC BX 40 DEC C 41 CMP C,0 42 JNZ Y 43 MOV AX, 0 44 INC DI 45 DEC CX 46 CMP CX,0 47 JNZ X 48 ;EXT 49 ;END

**EXERCISE:**

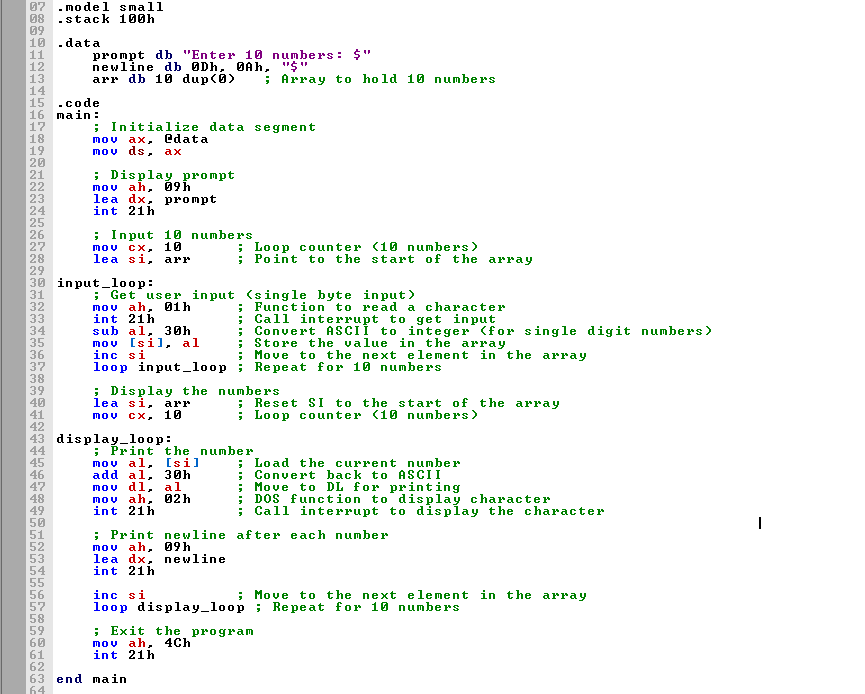
Q1) Calculate and display the sum of three numbers entered by user, save input in an array.

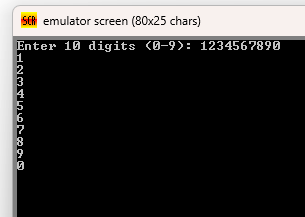




***Computer Architecture & Organization* Department of Computer Science & Information Technology**

Q2) Enter 10 elements in an array using input function and display in new line.





Q3)write a code to Save your name in an array and Display in reverse order .

