LAB TASK 5

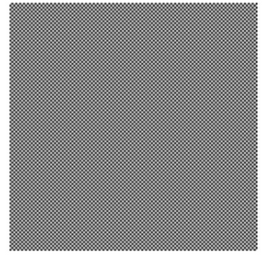
# WRITING ASSEMBLY LANGUAGE PROGRAM

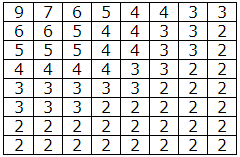
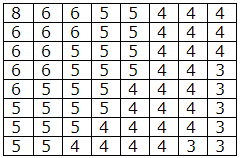
## **OBJECTIVE:**

Write an assembly program that adds the following two 16-bit images (Each pixel/element is of 16-bits) and displays the resultant memory values on the screen. Use appropriate addressing scheme.

## **IMAGES:**

Image A Image B



****

## **ASSEMBLY LANGUAGE PROGRAM:**

INCLUDE Irvine32.inc

.data

arrA byte 9, 7, 6, 5, 4, 4, 3, 3

byte 6, 6, 5, 4, 4, 3, 3, 2

byte 5, 5, 5, 4, 4, 3, 3, 2

byte 4, 4, 4, 4, 3, 3, 2, 2

byte 3, 3, 3, 3, 3, 2, 2, 2

byte 3, 3, 3, 2, 2, 2, 2, 2

byte 2, 2, 2, 2, 2, 2, 2, 2

byte 2, 2, 2, 2, 2, 2, 2, 2

arrB byte 8, 6, 6, 5, 5, 4, 4, 4

byte 6, 6, 6, 5, 5, 4, 4, 4

byte 6, 6, 5, 5, 5, 4, 4, 4

byte 6, 6, 5, 5, 5, 4, 4, 3

byte 6, 5, 5, 5, 4, 4, 4, 3

byte 5, 5, 5, 5, 4, 4, 4, 3

byte 5, 5, 5, 4, 4, 4, 4, 3

byte 5, 5, 4, 4, 4, 4, 3, 3

arrC byte 64 dup(?)

.code

main PROC

mov ecx, 64

L1:

mov al, arrA[ecx]

add al, arrB[ecx]

mov arrC[ecx], al

loop L1

mov esi, offset arrC

mov ebx, 1

mov ecx, 64

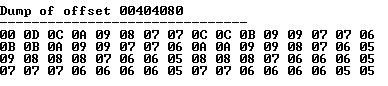
call dumpmem

exit

main ENDP

END main

## **OUTPUT:**



# **DISCRIPTION:**

In this lab, we basically learnt, how to add the pictures into a single one. We took the given two arrays as mentioned in the lab task and hard coded them in the program instead of inputting the from the user. Then we added the two pictures to form a single one. The code and the snaps of our working are given above.