

**EHB326E – INTRODUCTION TO EMBEDDED SYSTEMS HW2**

**Instructor: Prof. Dr. Müştak Erhan Yalçın**

**Assistant: Alp Eren Kıyak**

**Kamil Eren Ezen**

**040210021**

**Problem 1551 Definition:**

The problem I have to solve for my software part is: You are given an n x n 2D matrix representing an image, rotate the image by 90 degrees (clockwise). You have to rotate the image in-place, which means you have to modify the input 2D matrix directly. DO NOT allocate another 2D matrix and do the rotation.

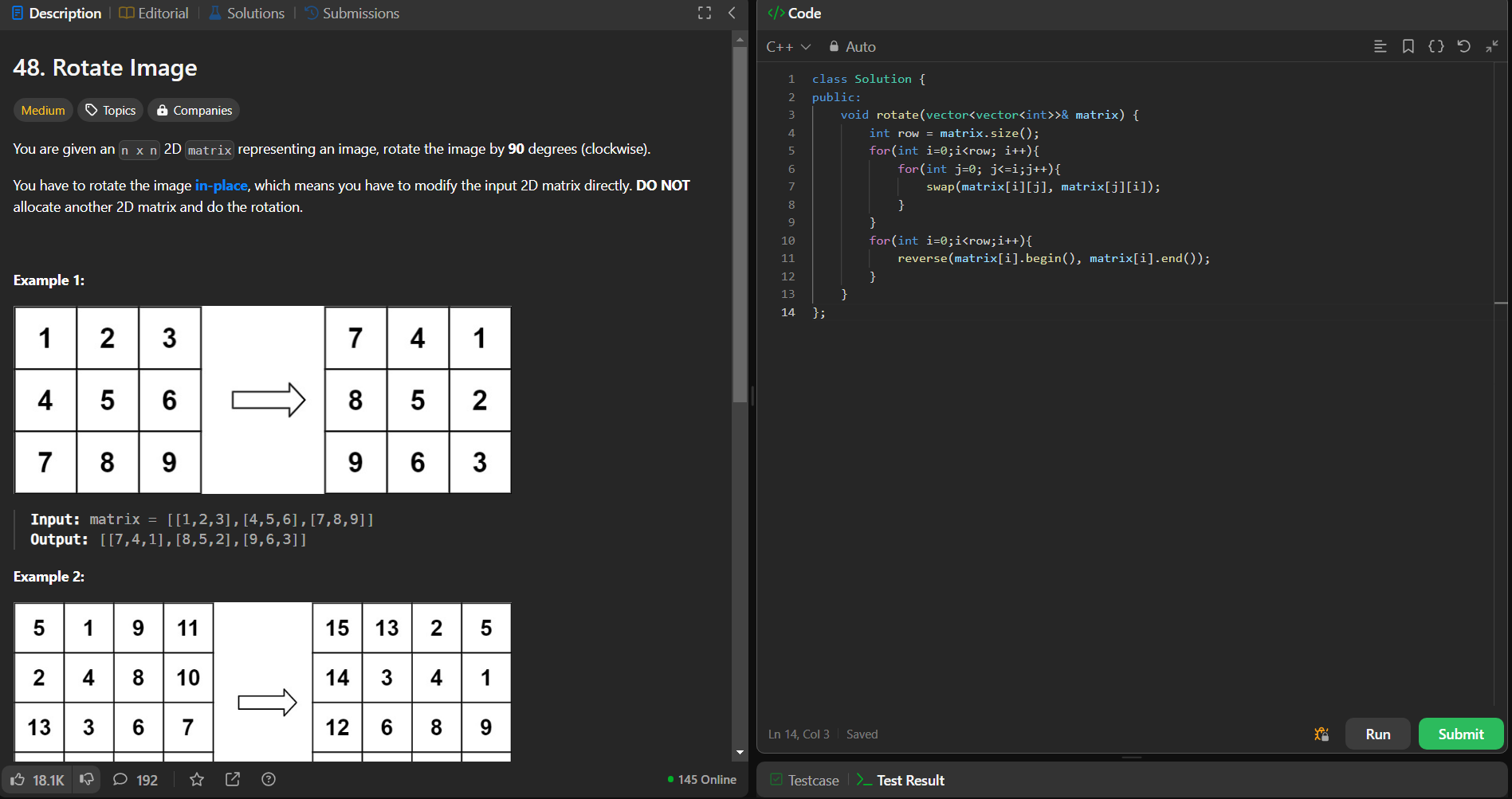
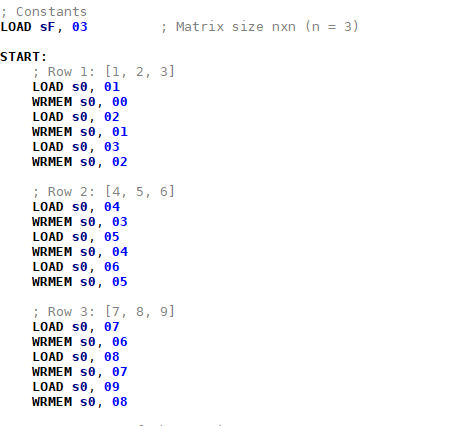


Figure : Leetcode problem and the solution

Approach of my solution is to take the transpose of the matrix first then swapping the columns. Tricky part was to convert this code to assembly-style.

**Assembly Solution**

**Part 1: Creating the matrix**

I wanted provide the inputs through directly in code itself. So, in the first part of the code, I created the matrix in scratchpad sequentially using LOAD and WRMEM instructions.

**Part 2: Taking Transpose of the Matrix**

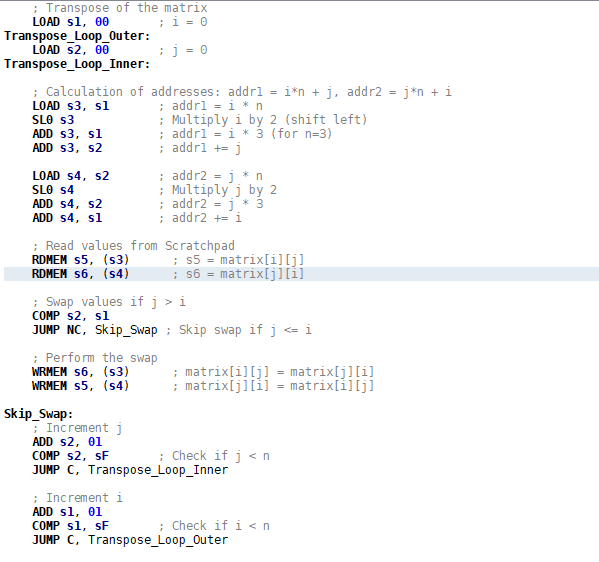
Then took the transpose of the matrix by swapping elements above and below the main diagonal.

Outer Loop (Transpose\_Loop\_Outer):

* This loop iterates through the rows of the matrix using the index i.

Inner Loop (Transpose\_Loop\_Inner):

* This loop iterates through the columns of the matrix using the index j.
  + Address Calculation:
    - addr1 = i \* n + j: Calculates the address of the element at row i and column j.
    - addr2 = j \* n + i: Calculates the address of the element at row j and column i.
  + Value Swap:
    - Values at these addresses are swapped if j > i. This ensures that swaps only occur for the upper triangle of the matrix.



**Part 3: Column Swapping**

After taking the transpose of the matrix we need to swap the left-most columns with right-most columns, then progress inward. If n is odd for nxn matrix then the middle column stays put.

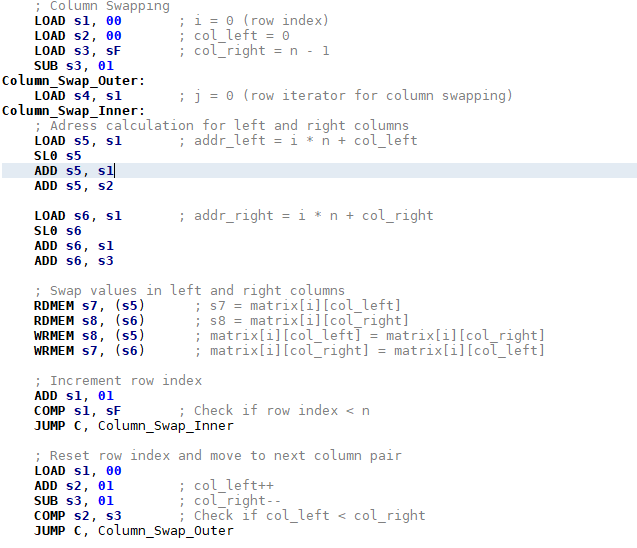
**Outer Loop (**Column\_Swap\_Outer**)**:

* Iterates through the columns, swapping the col\_left (starting at 0) with col\_right (starting at n-1).

**Inner Loop (**Column\_Swap\_Inner**)**:

* For each row (i), the addresses of the elements in the left and right columns are calculated:
  + addr\_left = i \* n + col\_left: The address of the element in the left column.
  + addr\_right = i \* n + col\_right: The address of the element in the right column.
* The values at these addresses are swapped.
* After completing the swaps for a row, the col\_left index is incremented, and the col\_right index is decremented.

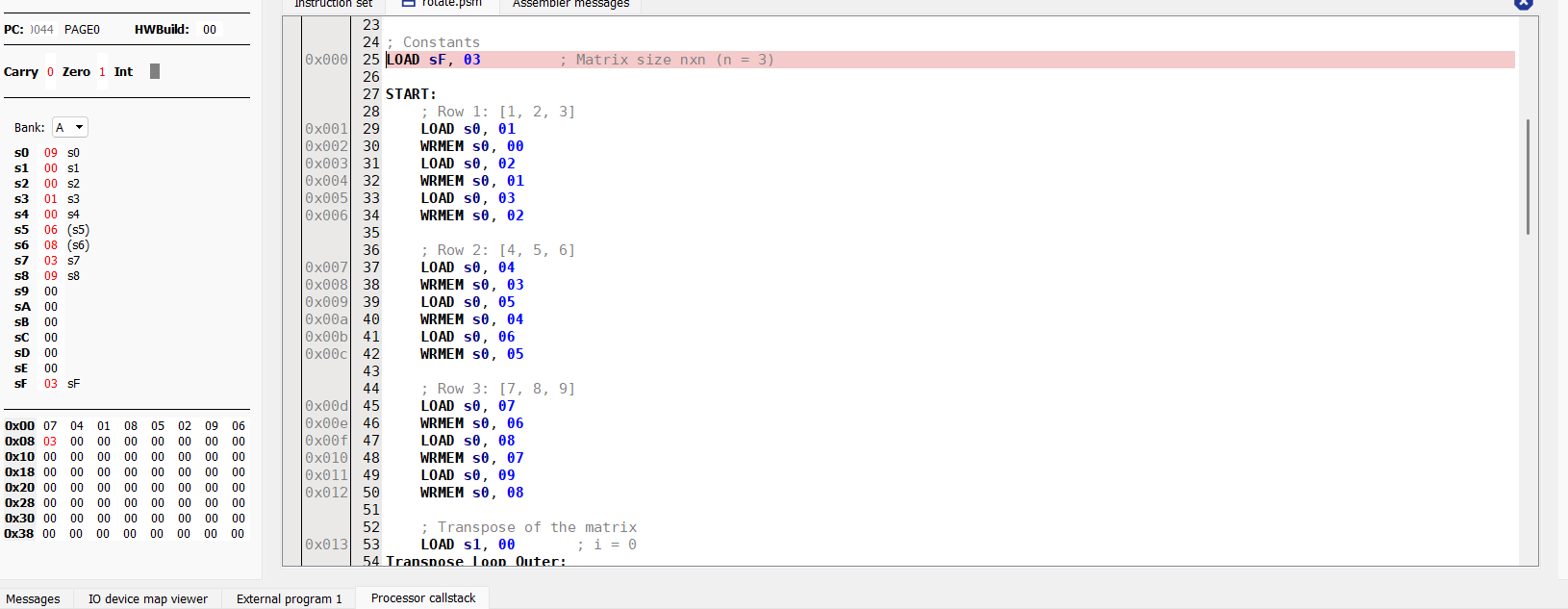
Process until the col\_left index is no longer less than the col\_right index, which means they met in the middle.

****

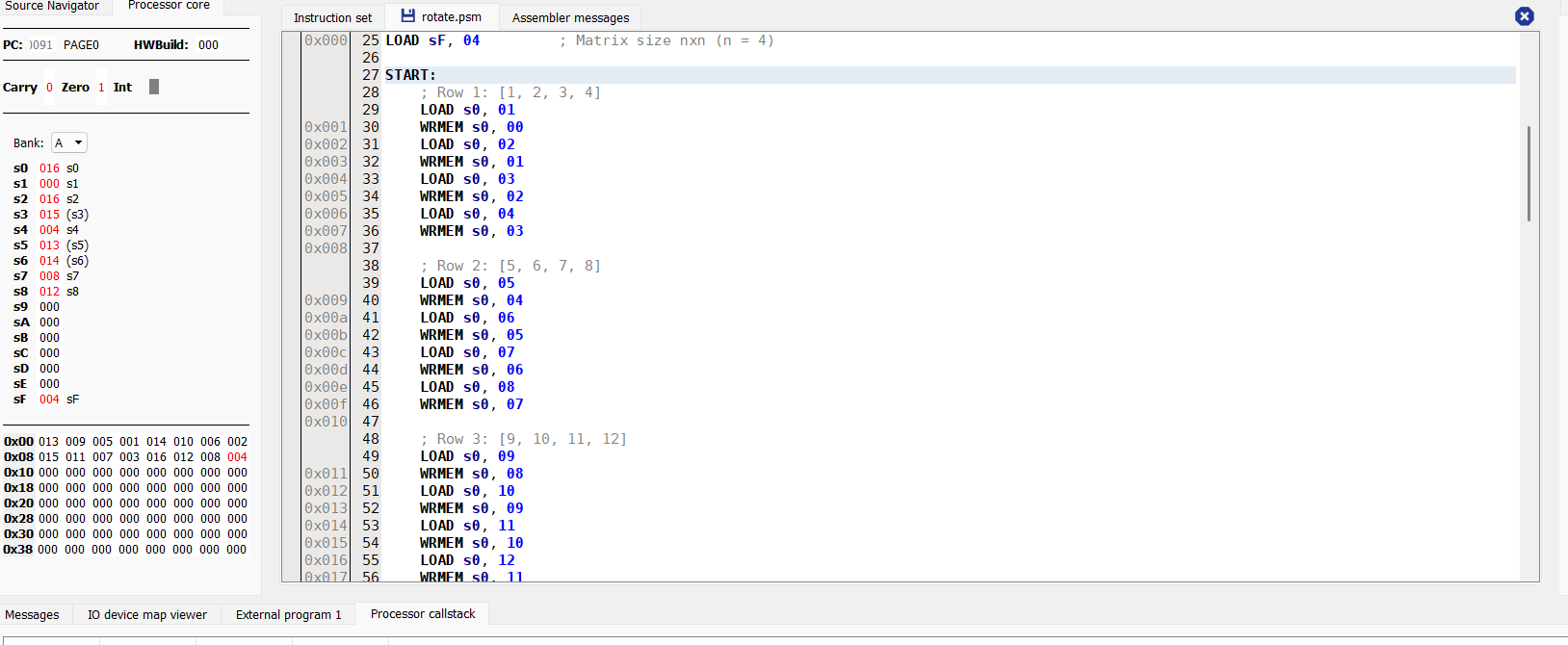
After this operation the result is the matrix rotated 90 degrees clockwise.

**Different Scenarios**

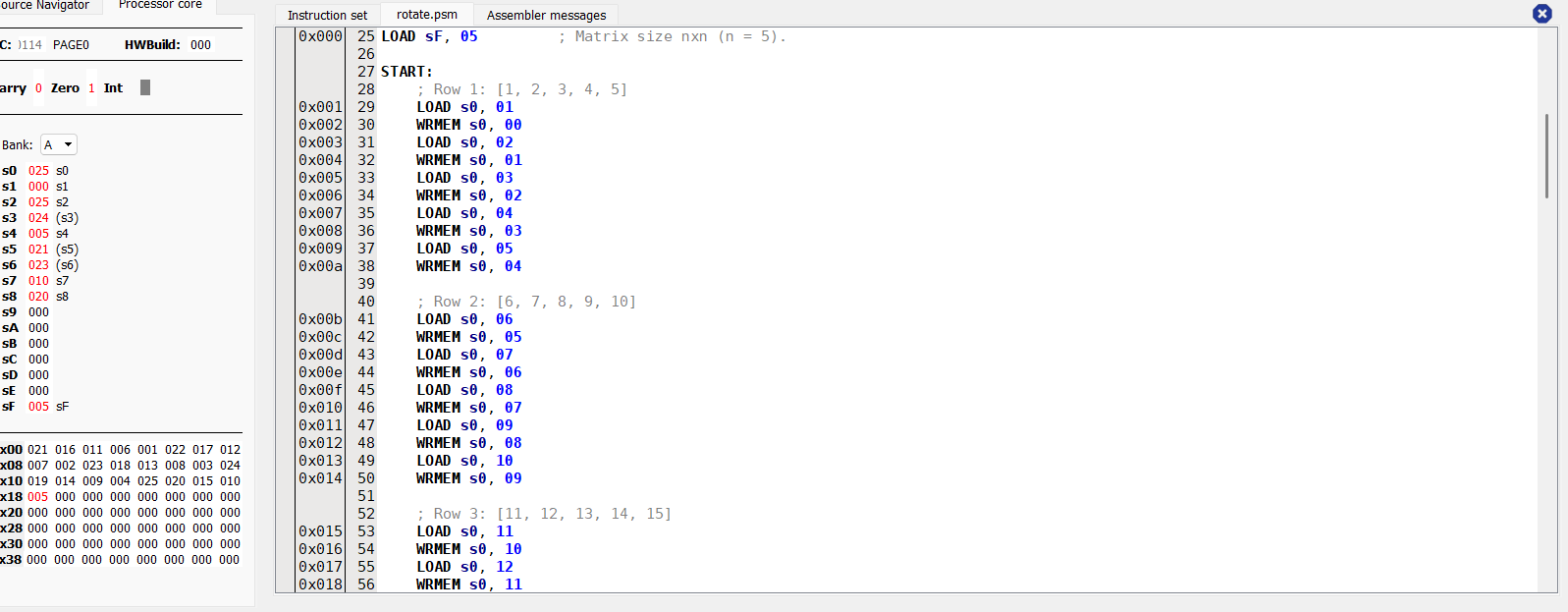
**n=3**

****

**n=4**

****

**n=5**

****