İSTANBUL TEKNİK ÜNİVERSİTESİ

ELEKTRİK ELEKTRONİK FAKÜLTESİ

INTRODUCTION TO EMBEDDED SYSTEMS

(EHB 326E)

Rotate Image Picoblaze Report

Hasan Emre AYDEMİR

1) 3 x 3 MATRIX

1.1) Problem Definition

In this assignment, the task is to rotate a given 3×3 matrix by 90 degrees clockwise using assembly code. The rotation must be performed in-place, meaning no additional memory structures such as secondary matrices are allowed. All data elements of the matrix are handled as 8-bit values and are stored in general-purpose registers of the processor.

The objective is to manually implement the 90° rotation logic by rearranging the positions of the original matrix elements using assembly instructions. The result of the rotation is then verified through simulation in the FIDEX environment by inspecting the updated register values.

The required matrix rotation transforms the matrix as follows:

```
Original Matrix:                  Rotated Matrix (90° Clockwise):

a00  a01  a02                     a20  a10  a00
a10  a11  a12         →           a21  a11  a01
a20  a21  a22                     a22  a12  a02
```

This rotation must be achieved by performing cyclic swaps on the **outer ring elements** and then on the **inner ring elements**, while keeping the center element unchanged.

1.2) Proposed Algorithm

The rotation process consists of two main stages. First, the elements on the **outer ring** of the matrix are cyclically shifted to achieve a clockwise rotation of the four corner values. Afterwards, the **inner ring**, which includes the edge-center elements, undergoes a similar cyclic shift to complete the transformation. Throughout this operation, the **center element** remains unchanged, as it is not affected by a 90-degree rotation.

Step-by-Step Algorithm:

1. **Load the matrix into registers:**

   The 3×3 matrix is mapped to registers as:

```
s0 s1 s2
s3 s4 s5
s6 s7 s8
```

2. **Rotate the outer ring (s0, s2, s8, s6):**
    - Temporarily store s0.
    - Move s6 → s0
    - Move s8 → s6
    - Move s2 → s8
    - Move temp → s2
      This achieves a clockwise rotation of the four corner elements.
3. **Rotate the inner ring (s1, s5, s7, s3):**
    - Temporarily store s1.
    - Move s3 → s1
    - Move s7 → s3
    - Move s5 → s7
    - Move temp → s5
      This rotates the middle-edge elements clockwise.
4. **Keep the center of the matrix (s4) unchanged:**
5. **Store or display the result:**

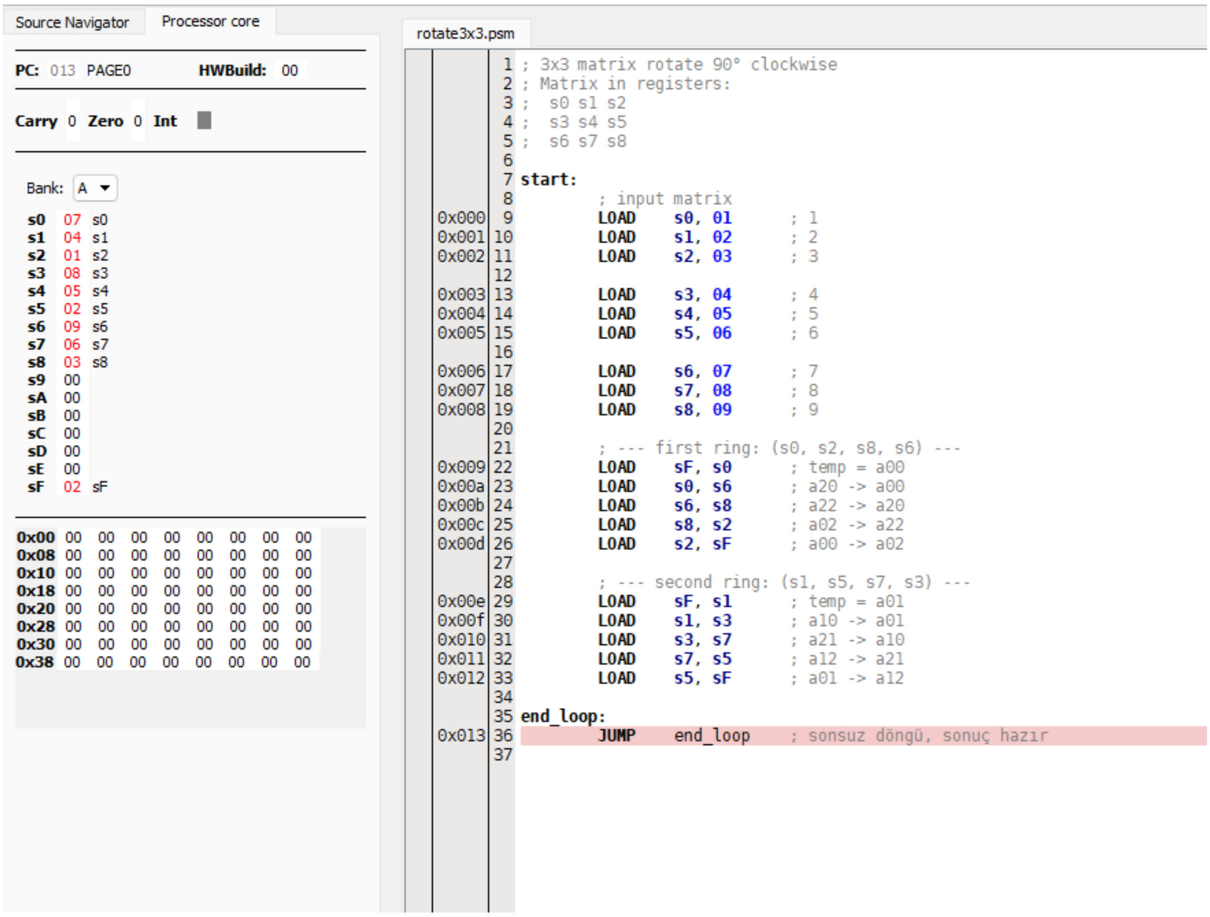
After the swaps are completed, the register contents represent the 90° rotated matrix. These values are verified through simulation by observing the register panel in FIDEX.

1.3) Assembly Code Implementation

```
; 3x3 matrix rotate 90° clockwise
; Matrix in registers:
;  s0 s1 s2
;  s3 s4 s5
;  s6 s7 s8

start:
        ; input matrix
        LOAD    s0, 01      ; 1
        LOAD    s1, 02      ; 2
        LOAD    s2, 03      ; 3

        LOAD    s3, 04      ; 4
        LOAD    s4, 05      ; 5
        LOAD    s5, 06      ; 6

        LOAD    s6, 07      ; 7
        LOAD    s7, 08      ; 8
        LOAD    s8, 09      ; 9

        ; --- first ring: (s0, s2, s8, s6) ---
        LOAD    sF, s0      ; temp = a00
        LOAD    s0, s6      ; a20 -> a00
        LOAD    s6, s8      ; a22 -> a20
        LOAD    s8, s2      ; a02 -> a22
        LOAD    s2, sF      ; a00 -> a02

        ; --- second ring: (s1, s5, s7, s3) ---
        LOAD    sF, s1      ; temp = a01
        LOAD    s1, s3      ; a10 -> a01
        LOAD    s3, s7      ; a21 -> a10
        LOAD    s7, s5      ; a12 -> a21
        LOAD    s5, sF      ; a01 -> a12

end_loop:
        JUMP    end_loop    ;
```

1.4) Test Results on FIDEX

Test 1



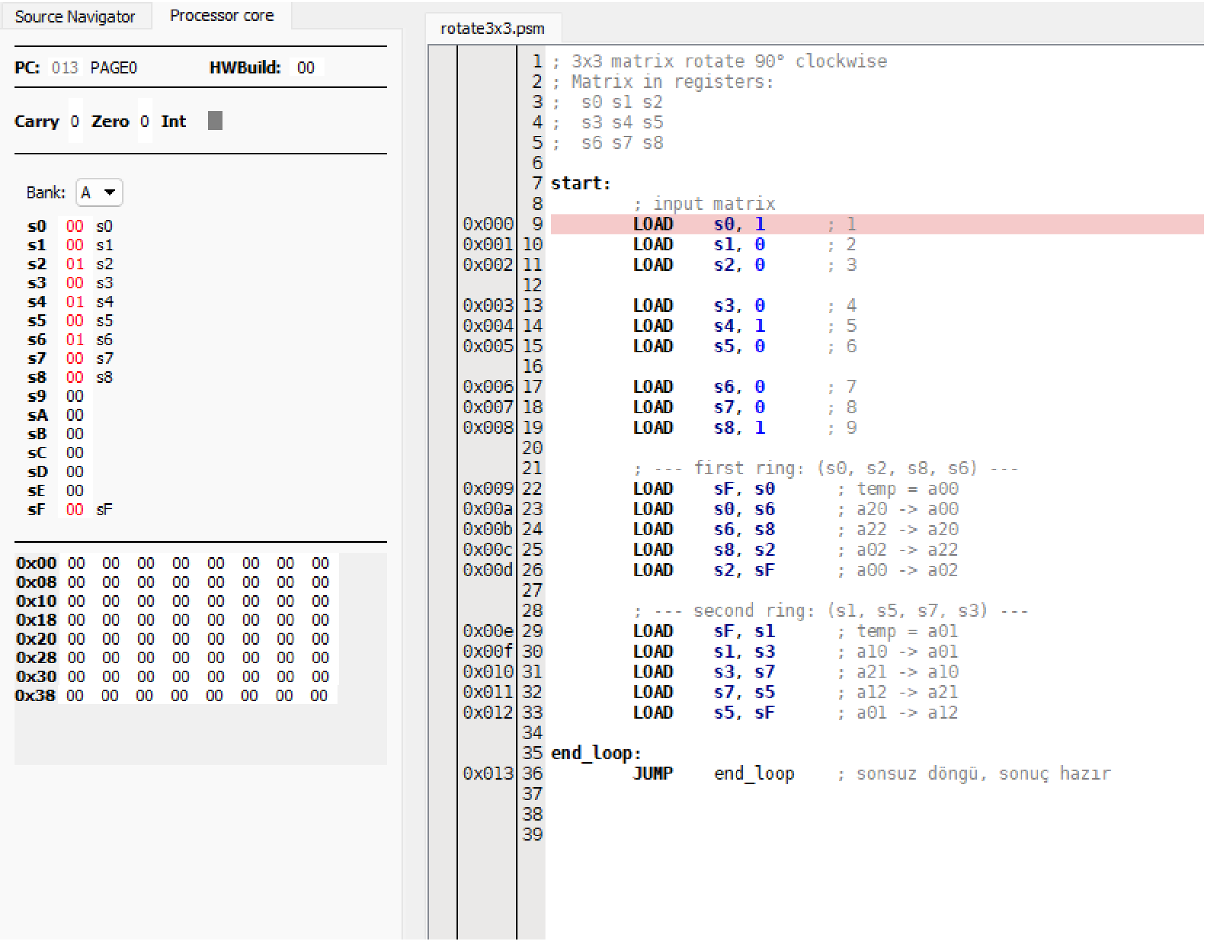
Input matrix:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Output (90° clockwise):

$$\begin{bmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

Test 2



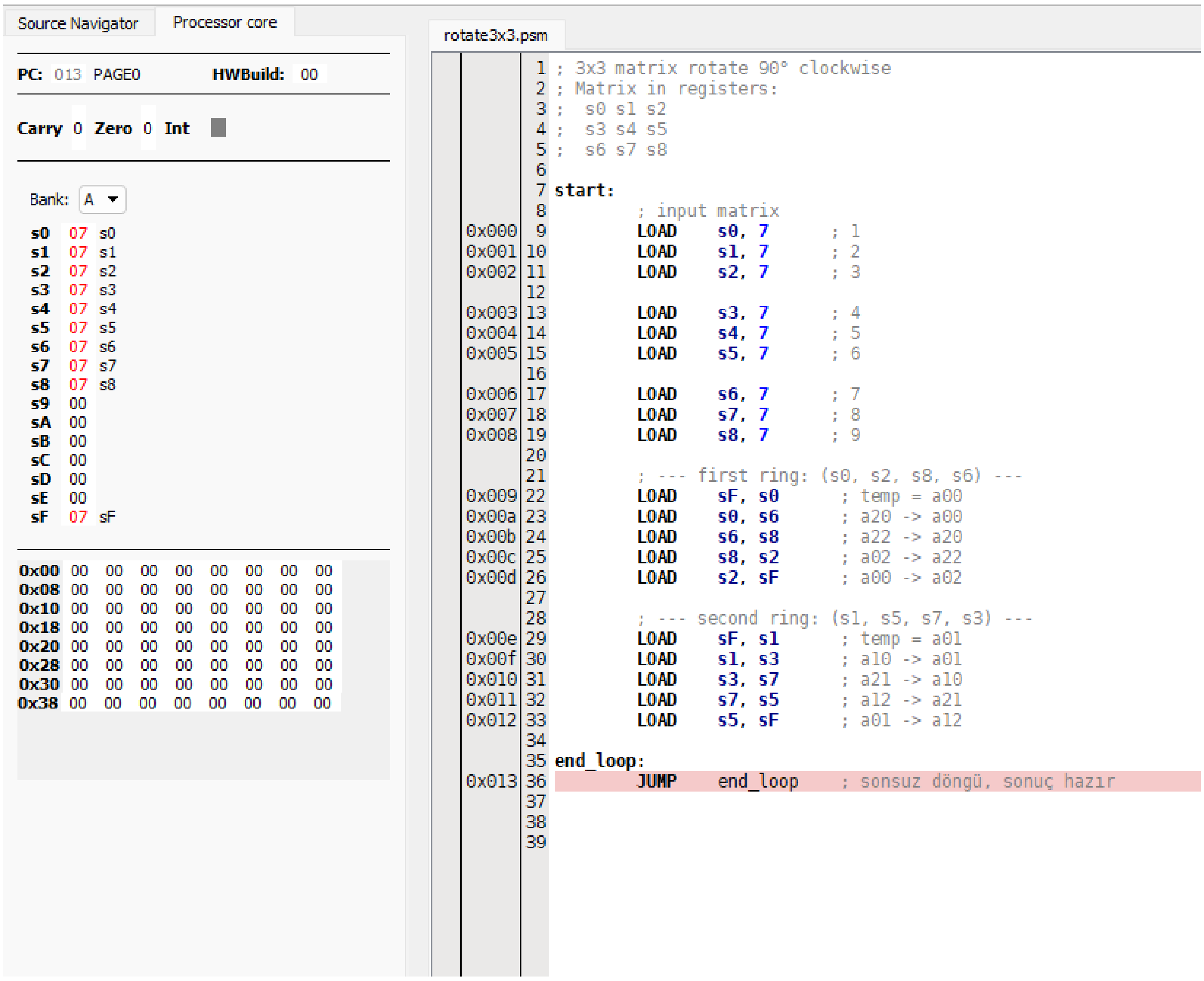
Input matrix:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Output:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Test 3



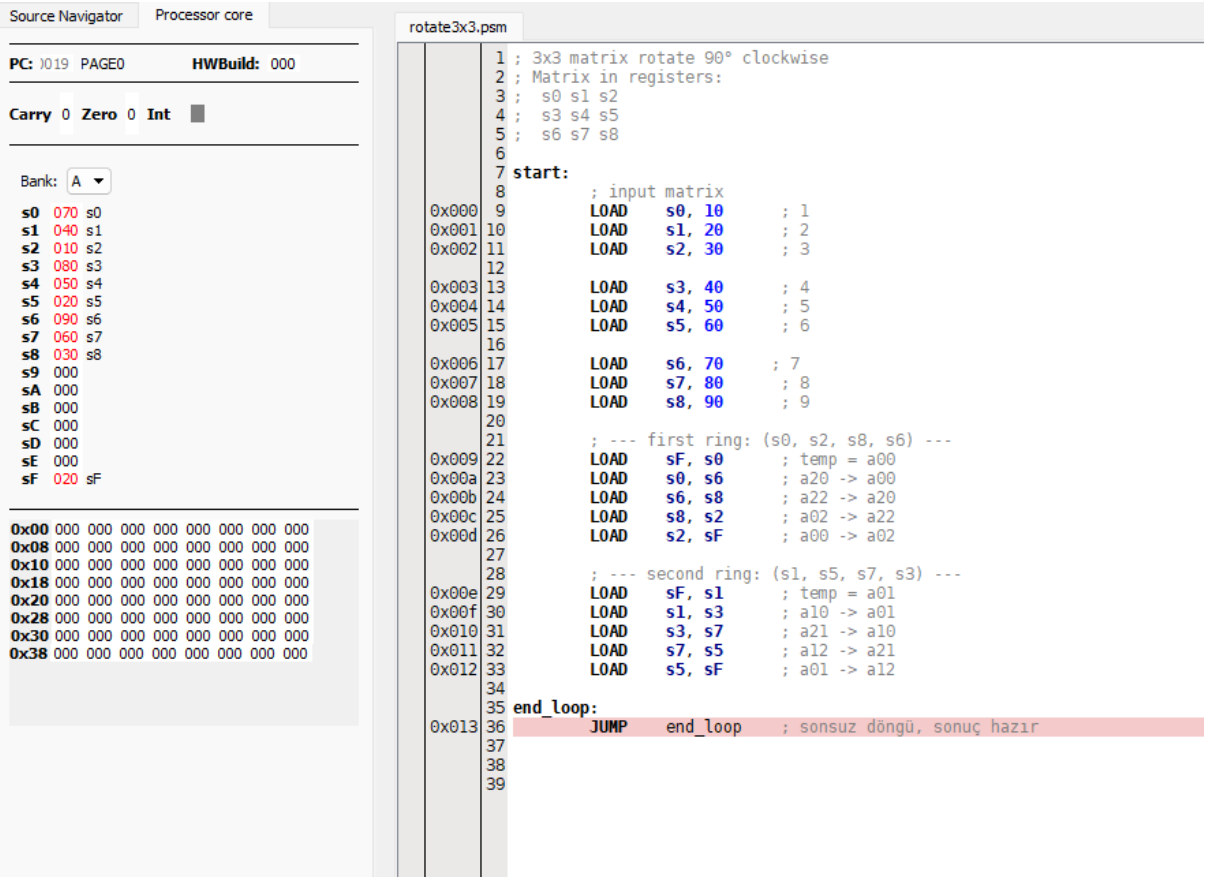
Input matrix:

$$\begin{bmatrix} 7 & 7 & 7 \\ 7 & 7 & 7 \\ 7 & 7 & 7 \end{bmatrix}$$

Output:

$$\begin{bmatrix} 7 & 7 & 7 \\ 7 & 7 & 7 \\ 7 & 7 & 7 \end{bmatrix}$$

Test 4



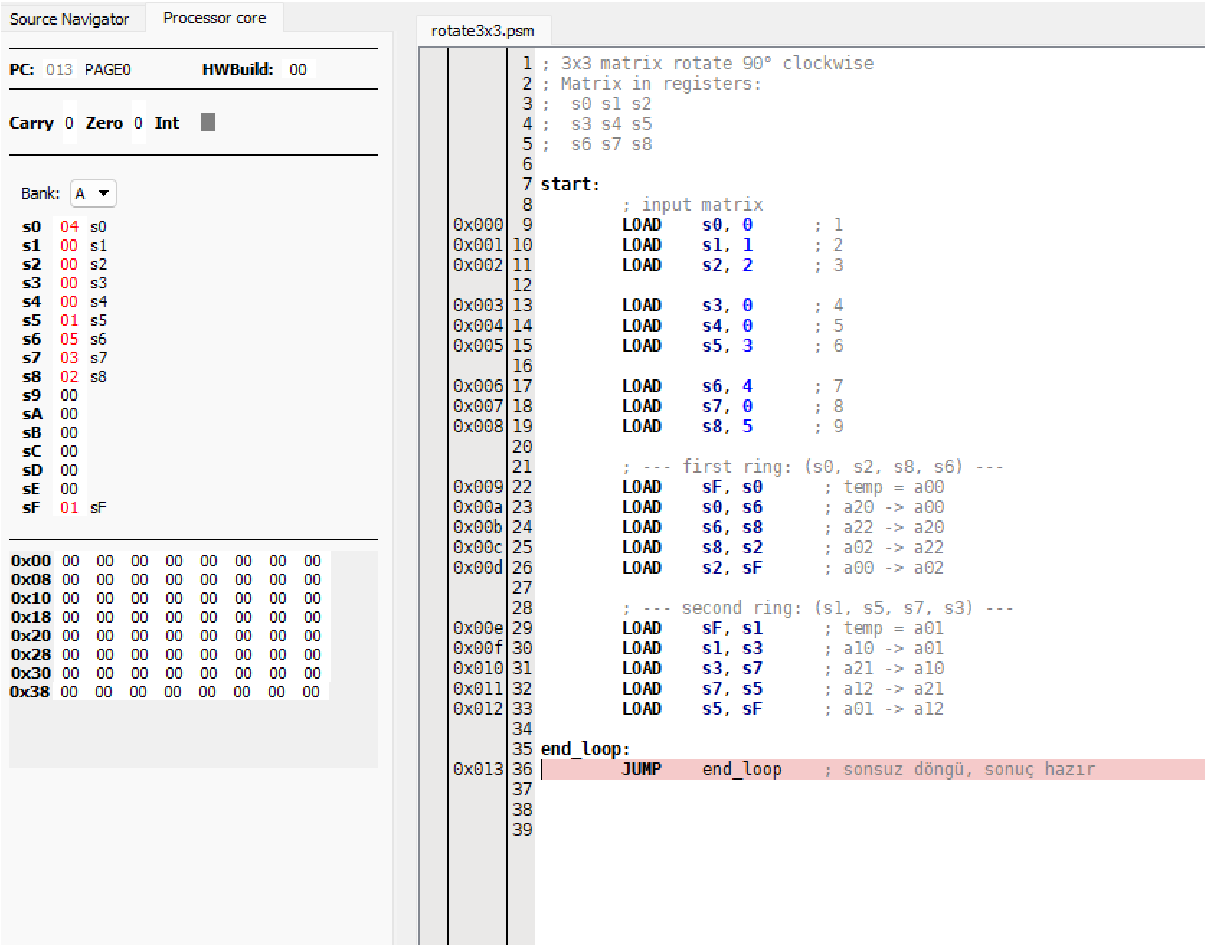
Input matrix:

$$\begin{bmatrix} 10 & 20 & 30 \\ 40 & 50 & 60 \\ 70 & 80 & 90 \end{bmatrix}$$

Output:

$$\begin{bmatrix} 70 & 40 & 10 \\ 80 & 50 & 20 \\ 90 & 60 & 30 \end{bmatrix}$$

Test 5



Input matrix:

$$\begin{bmatrix} 0 & 1 & 2 \\ 0 & 0 & 3 \\ 4 & 0 & 5 \end{bmatrix}$$

Output:

$$\begin{bmatrix} 4 & 0 & 0 \\ 0 & 0 & 1 \\ 5 & 3 & 2 \end{bmatrix}$$

2) 4 x 4 MATRIX

2.1) Problem Definition

In this extended version of the assignment, the goal is to rotate a 4×4 matrix by 90 degrees clockwise using assembly code. The rotation must be performed in-place, i.e., without allocating an additional 4×4 matrix. Each matrix element is treated as an 8-bit value and stored in the general-purpose registers of the processor.

The matrix is mapped to registers in row-major order as follows:

```
Original Matrix:                  Rotated Matrix (90° Clockwise):

a00 a01 a02 a03                    a30 a20 a10 a00
a10 a11 a12 a13     -->            a31 a21 a11 a01
a20 a21 a22 a23                    a32 a22 a12 a02
a30 a31 a32 a33                    a33 a23 a13 a03
```

The task is to implement this transformation using assembly instructions and verify the result in the FIDEX simulator by inspecting the final register values.

2.2) Proposed Algorithm

The 4×4 rotation is implemented in two layers (rings):

1. **Outer ring**: all elements on the border of the matrix (corners and edges).
2. **Inner ring**: the 2×2 block in the center.

The center ring and outer ring are rotated **clockwise** independently. The algorithm uses cyclic permutations of groups of four elements. Each group (a,b,c,d)(a, b, c, d)(a,b,c,d) is rotated using the mapping: (a,b,c,d) → (d,a,b,c)

This corresponds to a 90° clockwise rotation of those four positions.

Register mapping

- Matrix in registers:

```
Row 0: s0  s1  s2  s3     (a00 a01 a02 a03)
Row 1: s4  s5  s6  s7     (a10 a11 a12 a13)
Row 2: s8  s9  sA  sB     (a20 a21 a22 a23)
Row 3: sC  sD  sE  sF     (a30 a31 a32 a33)
```

Rotation groups

1. Outer corners: (s0,s3,sF,sC)
2. Outer edges – group 1: (s1,s7,sE,s8)
3. Outer edges – group 2: (s2,sB,sD,s4)
4. Inner 2×2 block: (s5,s6,sA,s9)

For each group, the rotation (a,b,c,d) → (d,a,b,c) is implemented only with register-to-register operations by composing three swaps:

(a,b,c,d) → swap(a,b) → swap(a,c) → swap(a,d)

Each swap is realized using the XOR technique:

```
a = a XOR b
b = a XOR b
a = a XOR b
```

This approach requires no extra temporary register, so all 16 registers can be used for the matrix elements.

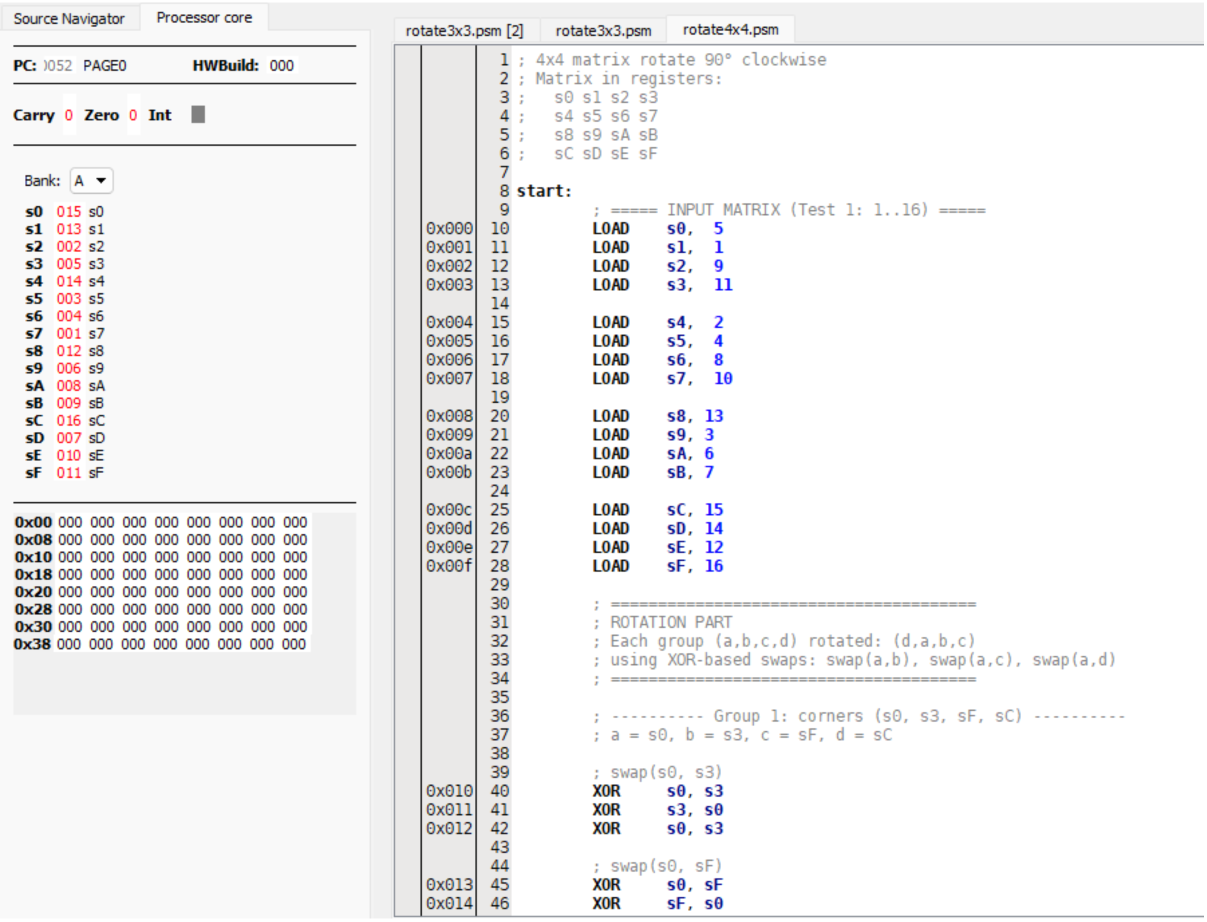
2.3) Assembly Code Implementation

```
; 4x4 matrix rotate 90° clockwise
; Matrix in registers:
;   s0 s1 s2 s3
;   s4 s5 s6 s7
;   s8 s9 sA sB
;   sC sD sE sF

start:
        ; ===== INPUT MATRIX (Test 1: 1..16) =====
        LOAD    s0,  5
        LOAD    s1,  1
        LOAD    s2,  9
        LOAD    s3,  11

        LOAD    s4,  2
        LOAD    s5,  4
        LOAD    s6,  8
        LOAD    s7,  10

        LOAD    s8, 13
        LOAD    s9, 3
        LOAD    sA, 6
        LOAD    sB, 7

        LOAD    sC, 15
        LOAD    sD, 14
        LOAD    sE, 12
        LOAD    sF, 16

        ; =======================================
        ; ROTATION PART
        ; Each group (a,b,c,d) rotated: (d,a,b,c)
        ; using XOR-based swaps: swap(a,b), swap(a,c), swap(a,d)
        ; =======================================

        ; ---------- Group 1: corners (s0, s3, sF, sC) ----------
        ; a = s0, b = s3, c = sF, d = sC

        ; swap(s0, s3)
        XOR     s0, s3
        XOR     s3, s0
        XOR     s0, s3

        ; swap(s0, sF)
        XOR     s0, sF
        XOR     sF, s0
        XOR     s0, sF

        ; swap(s0, sC)
        XOR     s0, sC
        XOR     sC, s0
        XOR     s0, sC
```

```
; ---------- Group 2: edges (s1, s7, sE, s8) ----------
        ; a = s1, b = s7, c = sE, d = s8

        ; swap(s1, s7)
        XOR     s1, s7
        XOR     s7, s1
        XOR     s1, s7

        ; swap(s1, sE)
        XOR     s1, sE
        XOR     sE, s1
        XOR     s1, sE

        ; swap(s1, s8)
        XOR     s1, s8
        XOR     s8, s1
        XOR     s1, s8

        ; ---------- Group 3: edges (s2, sB, sD, s4) ----------
        ; a = s2, b = sB, c = sD, d = s4

        ; swap(s2, sB)
        XOR     s2, sB
        XOR     sB, s2
        XOR     s2, sB

        ; swap(s2, sD)
        XOR     s2, sD
        XOR     sD, s2
        XOR     s2, sD

        ; swap(s2, s4)
        XOR     s2, s4
        XOR     s4, s2
        XOR     s2, s4

        ; ---------- Group 4: inner 2x2 (s5, s6, sA, s9) ----------
        ; a = s5, b = s6, c = sA, d = s9

        ; swap(s5, s6)
        XOR     s5, s6
        XOR     s6, s5
        XOR     s5, s6

        ; swap(s5, sA)
        XOR     s5, sA
        XOR     sA, s5
        XOR     s5, sA

        ; swap(s5, s9)
        XOR     s5, s9
        XOR     s9, s5
        XOR     s5, s9

end_loop:
        JUMP    end_loop      ;
```

2.4) Test Results on FIDEX

Test 1



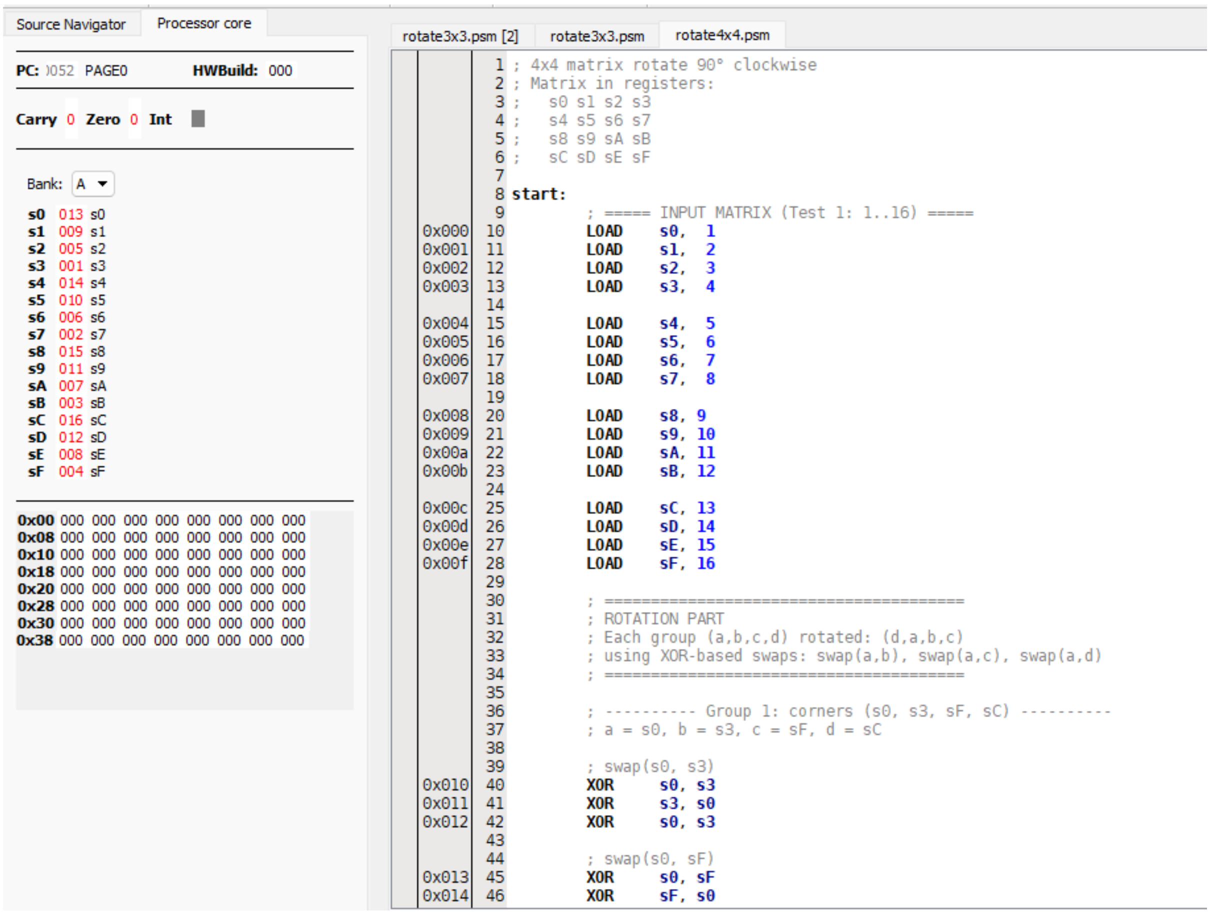
Input:

$$\begin{bmatrix} 5 & 1 & 9 & 11 \\ 2 & 4 & 8 & 10 \\ 13 & 3 & 6 & 7 \\ 15 & 14 & 12 & 16 \end{bmatrix}$$

Output:

$$\begin{bmatrix} 15 & 13 & 2 & 5 \\ 14 & 3 & 4 & 1 \\ 12 & 6 & 8 & 9 \\ 16 & 7 & 10 & 11 \end{bmatrix}$$

Test 2



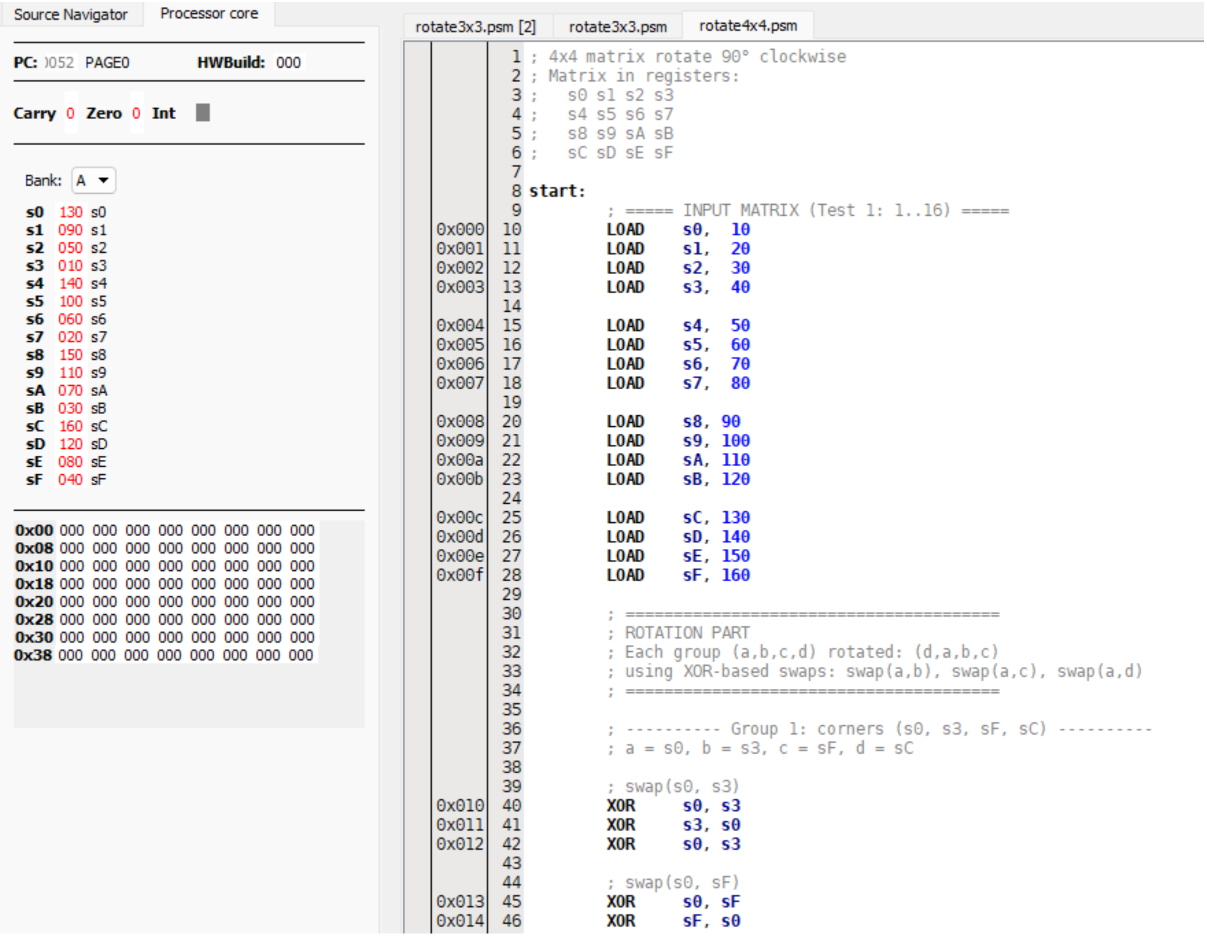
Input:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Output:

$$\begin{bmatrix} 13 & 9 & 5 & 1 \\ 14 & 10 & 6 & 2 \\ 15 & 11 & 7 & 3 \\ 16 & 12 & 8 & 4 \end{bmatrix}$$

Test 3



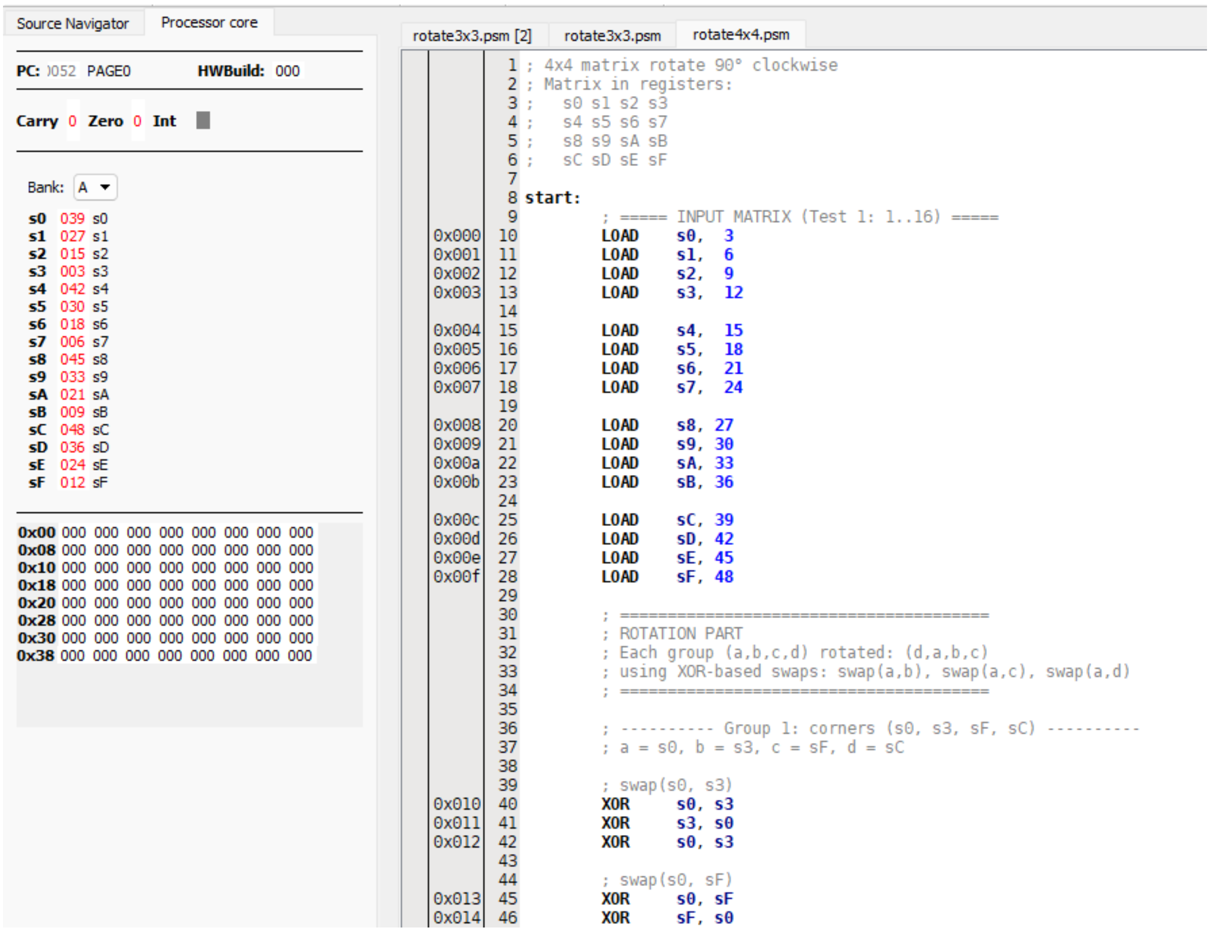
Input:

$$\begin{bmatrix} 10 & 20 & 30 & 40 \\ 50 & 60 & 70 & 80 \\ 90 & 100 & 110 & 120 \\ 130 & 140 & 150 & 160 \end{bmatrix}$$

Output:

$$\begin{bmatrix} 130 & 90 & 50 & 10 \\ 140 & 100 & 60 & 20 \\ 150 & 110 & 70 & 30 \\ 160 & 120 & 80 & 40 \end{bmatrix}$$

Test 4



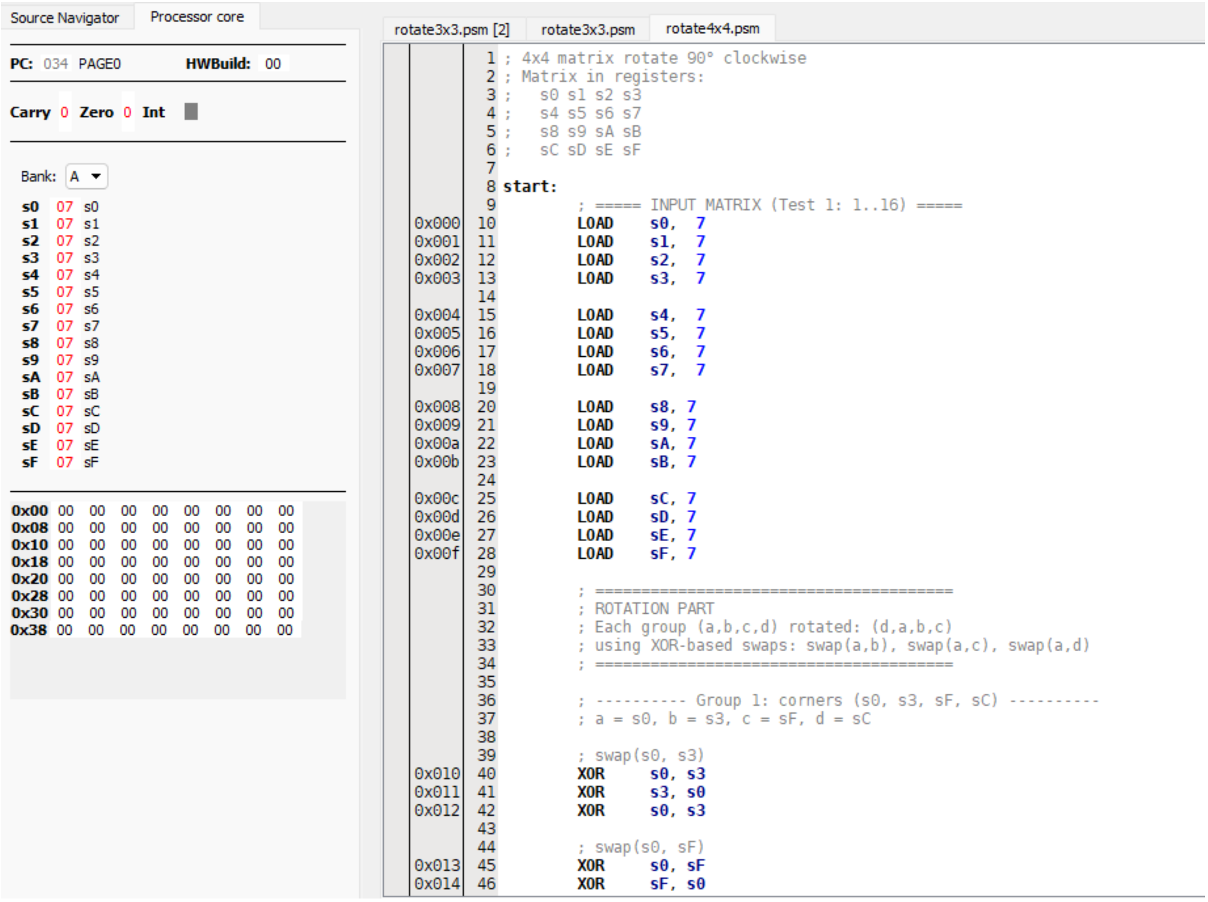
Input:

$$
\begin{bmatrix}
3 & 6 & 9 & 12 \\
15 & 18 & 21 & 24 \\
27 & 30 & 33 & 36 \\
39 & 42 & 45 & 48
\end{bmatrix}
$$

Output (90° clockwise):

$$
\begin{bmatrix}
39 & 27 & 15 & 3 \\
42 & 30 & 18 & 6 \\
45 & 33 & 21 & 9 \\
48 & 36 & 24 & 12
\end{bmatrix}
$$

Test 5



Input:

$$\begin{bmatrix} 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 \end{bmatrix}$$

Output:

$$\begin{bmatrix} 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 \\ 7 & 7 & 7 & 7 \end{bmatrix}$$