

**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  $3 \times 3$  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  $90^\circ$  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 ( $90^\circ$ Clockwise):
$\begin{matrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{matrix}$	$\begin{matrix} a_{20} & a_{10} & a_{00} \\ a_{21} & a_{11} & a_{01} \\ a_{22} & a_{12} & a_{02} \end{matrix}$

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  $3 \times 3$  matrix is mapped to registers as:

s0 s1 s2
s3 s4 s5
s6 s7 s8

**2. Rotate the outer ring ( $s_0, s_2, s_8, s_6$ ):**

- Temporarily store  $s_0$ .
- Move  $s_6 \rightarrow s_0$
- Move  $s_8 \rightarrow s_6$
- Move  $s_2 \rightarrow s_8$
- Move temp  $\rightarrow s_2$

This achieves a clockwise rotation of the four corner elements.

**3. Rotate the inner ring ( $s_1, s_5, s_7, s_3$ ):**

- Temporarily store  $s_1$ .
- Move  $s_3 \rightarrow s_1$
- Move  $s_7 \rightarrow s_3$
- Move  $s_5 \rightarrow s_7$
- Move temp  $\rightarrow s_5$

This rotates the middle-edge elements clockwise.

**4. Keep the center of the matrix ( $s_4$ ) unchanged:**

**5. Store or display the result:**

After the swaps are completed, the register contents represent the  $90^\circ$  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

Source Navigator   Processor core

PC: 013 PAGE0   HWBuild: 00

Carry 0 Zero 0 Int █

Bank: A ▾

```

rotate3x3.psm

1 ; 3x3 matrix rotate 90° clockwise
2 ; Matrix in registers:
3 ; s0 s1 s2
4 ; s3 s4 s5
5 ; s6 s7 s8
6
7 start:
8 ; input matrix
0x000 9 LOAD s0, 01 ; 1
0x001 10 LOAD s1, 02 ; 2
0x002 11 LOAD s2, 03 ; 3
12
0x003 13 LOAD s3, 04 ; 4
0x004 14 LOAD s4, 05 ; 5
0x005 15 LOAD s5, 06 ; 6
16
0x006 17 LOAD s6, 07 ; 7
0x007 18 LOAD s7, 08 ; 8
0x008 19 LOAD s8, 09 ; 9
20
21 ; --- first ring: (s0, s2, s8, s6) ---
0x009 22 LOAD sF, s0 ; temp = a00
0x00a 23 LOAD s0, s6 ; a20 -> a00
0x00b 24 LOAD s6, s8 ; a22 -> a20
0x00c 25 LOAD s8, s2 ; a02 -> a22
0x00d 26 LOAD s2, sF ; a00 -> a02
27
28 ; --- second ring: (s1, s5, s7, s3) ---
0x00e 29 LOAD sF, s1 ; temp = a01
0x00f 30 LOAD s1, s3 ; a10 -> a01
0x010 31 LOAD s3, s7 ; a21 -> a10
0x011 32 LOAD s7, s5 ; a12 -> a21
0x012 33 LOAD s5, sF ; a01 -> a12
34
35 end_loop:
0x013 36 JUMP end_loop ; sonsuz döngü, sonuç hazır
37

```

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

The screenshot shows a software interface with two main panes. The left pane is titled "Source Navigator" and displays assembly code for a 3x3 matrix rotation. The right pane is titled "Processor core" and shows a memory dump of the matrix elements.

**Source Navigator (Left):**

- PC: 013 PAGE0 Processor core
- Carry 0 Zero 0 Int
- Bank: A
- Registers (Bank A):
  - s0 00 s0
  - s1 00 s1
  - s2 01 s2
  - s3 00 s3
  - s4 01 s4
  - s5 00 s5
  - s6 01 s6
  - s7 00 s7
  - s8 00 s8
  - s9 00
  - sA 00
  - sB 00
  - sC 00
  - sD 00
  - sE 00
  - sF 00 sF
- Memory dump (Bank A):
 

0x00	00	00	00	00	00	00	00	00
0x08	00	00	00	00	00	00	00	00
0x10	00	00	00	00	00	00	00	00
0x18	00	00	00	00	00	00	00	00
0x20	00	00	00	00	00	00	00	00
0x28	00	00	00	00	00	00	00	00
0x30	00	00	00	00	00	00	00	00
0x38	00	00	00	00	00	00	00	00

**Processor core (Right):**

rotate3x3.psm

```

1 ; 3x3 matrix rotate 90° clockwise
2 ; Matrix in registers:
3 ;   s0 s1 s2
4 ;   s3 s4 s5
5 ;   s6 s7 s8
6
7 start:
8     ; input matrix
9    LOAD  s0, 1 ; 1
0x000 10   LOAD  s1, 0 ; 2
0x001 11   LOAD  s2, 0 ; 3
12
0x003 13   LOAD  s3, 0 ; 4
0x004 14   LOAD  s4, 1 ; 5
0x005 15   LOAD  s5, 0 ; 6
16
0x006 17   LOAD  s6, 0 ; 7
0x007 18   LOAD  s7, 0 ; 8
0x008 19   LOAD  s8, 1 ; 9
20
21     ; --- first ring: (s0, s2, s8, s6) ---
22   LOAD  sF, s0 ; temp = a00
0x009 23   LOAD  s0, s6 ; a20 -> a00
0x00a 24   LOAD  s6, s8 ; a22 -> a20
0x00b 25   LOAD  s8, s2 ; a02 -> a22
0x00d 26   LOAD  s2, sF ; a00 -> a02
27
28     ; --- second ring: (s1, s5, s7, s3) ---
0x00e 29   LOAD  sF, s1 ; temp = a01
0x00f 30   LOAD  s1, s3 ; a10 -> a01
0x010 31   LOAD  s3, s7 ; a21 -> a10
0x011 32   LOAD  s7, s5 ; a12 -> a21
0x012 33   LOAD  s5, sF ; a01 -> a12
34
35 end_loop:
0x013 36   JUMP  end_loop ; sonsuz döngü, sonuç hazır
37
38
39

```

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

```

Source Navigator Processor core
PC: 013 PAGE0 HWBuild: 00
Carry 0 Zero 0 Int ■

Bank: A ▾
s0 07 s0
s1 07 s1
s2 07 s2
s3 07 s3
s4 07 s4
s5 07 s5
s6 07 s6
s7 07 s7
s8 07 s8
s9 00
sA 00
sB 00
sC 00
sD 00
sE 00
sF 07 sF

0x00 00 00 00 00 00 00 00 00 00
0x08 00 00 00 00 00 00 00 00 00
0x10 00 00 00 00 00 00 00 00 00
0x18 00 00 00 00 00 00 00 00 00
0x20 00 00 00 00 00 00 00 00 00
0x28 00 00 00 00 00 00 00 00 00
0x30 00 00 00 00 00 00 00 00 00
0x38 00 00 00 00 00 00 00 00 00

rotate3x3.psm
1 ; 3x3 matrix rotate 90° clockwise
2 ; Matrix in registers:
3 ; s0 s1 s2
4 ; s3 s4 s5
5 ; s6 s7 s8
6
7 start:
8 ; input matrix
9 LOAD s0, 7 ; 1
0x000 10 LOAD s1, 7 ; 2
0x001 11 LOAD s2, 7 ; 3
0x002 12
0x003 13 LOAD s3, 7 ; 4
0x004 14 LOAD s4, 7 ; 5
0x005 15 LOAD s5, 7 ; 6
0x006 16
0x007 17 LOAD s6, 7 ; 7
0x008 18 LOAD s7, 7 ; 8
0x009 19 LOAD s8, 7 ; 9
0x00a 20
0x00b 21 ; --- first ring: (s0, s2, s8, s6) ---
0x00c 22 LOAD sF, s0 ; temp = a00
0x00d 23 LOAD s0, s6 ; a20 -> a00
0x00e 24 LOAD s6, s8 ; a22 -> a20
0x00f 25 LOAD s8, s2 ; a02 -> a22
0x010 26 LOAD s2, sF ; a00 -> a02
0x011 27
0x012 28 ; --- second ring: (s1, s5, s7, s3) ---
0x013 29 LOAD sF, s1 ; temp = a01
0x014 30 LOAD s1, s3 ; a10 -> a01
0x015 31 LOAD s3, s7 ; a11 -> a10
0x016 32 LOAD s7, s5 ; a12 -> a21
0x017 33 LOAD s5, sF ; a01 -> a12
0x018 34
0x019 35 end_loop:
0x01a 36 JUMP end_loop ; sonsuz döngü, sonuç hazır
0x01b 37
0x01c 38
0x01d 39

```

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

```

Source Navigator Processor core
PC: 0019 PAGE00 HWBuild: 000
Carry 0 Zero 0 Int ■

Bank: A
s0 070 s0
s1 040 s1
s2 010 s2
s3 080 s3
s4 050 s4
s5 020 s5
s6 090 s6
s7 060 s7
s8 030 s8
s9 000
sA 000
sB 000
sC 000
sD 000
sE 000
sF 020 sF

0x00 000 000 000 000 000 000 000 000 000
0x08 000 000 000 000 000 000 000 000 000
0x10 000 000 000 000 000 000 000 000 000
0x18 000 000 000 000 000 000 000 000 000
0x20 000 000 000 000 000 000 000 000 000
0x28 000 000 000 000 000 000 000 000 000
0x30 000 000 000 000 000 000 000 000 000
0x38 000 000 000 000 000 000 000 000 000

rotate3x3.psm
1 ; 3x3 matrix rotate 90° clockwise
2 ; Matrix in registers:
3 ; s0 s1 s2
4 ; s3 s4 s5
5 ; s6 s7 s8
6
7 start:
8 ; input matrix
9 LOAD s0, 10 ; 1
10 LOAD s1, 20 ; 2
11 LOAD s2, 30 ; 3
12
13 LOAD s3, 40 ; 4
14 LOAD s4, 50 ; 5
15 LOAD s5, 60 ; 6
16
17 LOAD s6, 70 ; 7
18 LOAD s7, 80 ; 8
19 LOAD s8, 90 ; 9
20
21 ; --- first ring: (s0, s2, s8, s6) ---
22 LOAD sF, s0 ; temp = a00
23 LOAD s0, s6 ; a20 -> a00
24 LOAD s6, s8 ; a22 -> a20
25 LOAD s8, s2 ; a02 -> a22
26 LOAD s2, sF ; a00 -> a02
27
28 ; --- second ring: (s1, s5, s7, s3) ---
29 LOAD sF, s1 ; temp = a01
30 LOAD s1, s3 ; a10 -> a01
31 LOAD s3, s7 ; a21 -> a10
32 LOAD s7, s5 ; a12 -> a21
33 LOAD s5, sF ; a01 -> a12
34
35 end_loop:
36 JUMP end_loop ; sonsuz döngü, sonuç hazır
37
38
39

```

**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

Source Navigator   Processor core

**rotate3x3.psm**

PC: 013 PAGE0	HWBuild: 00
Carry 0 Zero 0 Int	■
Bank: A <b>s0</b> 04 s0 <b>s1</b> 00 s1 <b>s2</b> 00 s2 <b>s3</b> 00 s3 <b>s4</b> 00 s4 <b>s5</b> 01 s5 <b>s6</b> 05 s6 <b>s7</b> 03 s7 <b>s8</b> 02 s8 <b>s9</b> 00 <b>sA</b> 00 <b>sB</b> 00 <b>sC</b> 00 <b>sD</b> 00 <b>sE</b> 00 <b>sF</b> 01 sF	
0x00 00 00 00 00 00 00 00 00 0x08 00 00 00 00 00 00 00 00 0x10 00 00 00 00 00 00 00 00 0x18 00 00 00 00 00 00 00 00 0x20 00 00 00 00 00 00 00 00 0x28 00 00 00 00 00 00 00 00 0x30 00 00 00 00 00 00 00 00 0x38 00 00 00 00 00 00 00 00	

```

1 ; 3x3 matrix rotate 90° clockwise
2 ; Matrix in registers:
3 ;   s0 s1 s2
4 ;   s3 s4 s5
5 ;   s6 s7 s8
6
7 start:
8     ; input matrix
9     LOAD  s0, 0      ; 1
0x000 10    LOAD  s1, 1      ; 2
0x002 11    LOAD  s2, 2      ; 3
12
13     LOAD  s3, 0      ; 4
0x004 14    LOAD  s4, 0      ; 5
0x005 15    LOAD  s5, 3      ; 6
16
17     LOAD  s6, 4      ; 7
0x007 18    LOAD  s7, 0      ; 8
0x008 19    LOAD  s8, 5      ; 9
20
21     ; --- first ring: (s0, s2, s8, s6) ---
22     LOAD  sF, s0      ; temp = a00
0x009 23    LOAD  s0, s6      ; a20 -> a00
0x00a 24    LOAD  s6, s8      ; a22 -> a20
0x00c 25    LOAD  s8, s2      ; a02 -> a22
0x00d 26    LOAD  s2, sF      ; a00 -> a02
27
28     ; --- second ring: (s1, s5, s7, s3) ---
0x00e 29    LOAD  sF, s1      ; temp = a01
0x00f 30    LOAD  s1, s3      ; a10 -> a01
0x010 31    LOAD  s3, s7      ; a21 -> a10
0x011 32    LOAD  s7, s5      ; a12 -> a21
0x012 33    LOAD  s5, sF      ; a01 -> a12
34
35 end_loop:
0x013 36    JUMP  end_loop  ; sonsuz döngü, sonuç hazır
37
38
39

```

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  $4 \times 4$  matrix by 90 degrees clockwise using assembly code. The rotation must be performed in-place, i.e., without allocating an additional  $4 \times 4$  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 ( $90^\circ$ 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  $4 \times 4$  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  $2 \times 2$  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:  $(a,b,c,d) \rightarrow (d,a,b,c)$

This corresponds to a  $90^\circ$  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  $2 \times 2$  block: (s5,s6,sA,s9)

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

$(a,b,c,d) \rightarrow \text{swap}(a,b) \rightarrow \text{swap}(a,c) \rightarrow \text{swap}(a,d)$

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

Source Navigator   Processor core

PC: J052 PAGE00   HWBuild: 000

Carry 0 Zero 0 Int █

Bank: A ▾

```

s0 015 s0
s1 013 s1
s2 002 s2
s3 005 s3
s4 014 s4
s5 003 s5
s6 004 s6
s7 001 s7
s8 012 s8
s9 006 s9
sA 008 sA
sB 009 sB
sC 016 sC
sD 007 sD
sE 010 sE
sF 011 sF

0x00 000 000 000 000 000 000 000 000 000
0x08 000 000 000 000 000 000 000 000 000
0x10 000 000 000 000 000 000 000 000 000
0x18 000 000 000 000 000 000 000 000 000
0x20 000 000 000 000 000 000 000 000 000
0x28 000 000 000 000 000 000 000 000 000
0x30 000 000 000 000 000 000 000 000 000
0x38 000 000 000 000 000 000 000 000 000

```

rotate3x3.psm [2] rotate3x3.psm rotate4x4.psm

```

1 ; 4x4 matrix rotate 90° clockwise:
2 ; Matrix in registers:
3 ;   s0 s1 s2 s3
4 ;   s4 s5 s6 s7
5 ;   s8 s9 sA sB
6 ;   sC sD sE sF
7
8 start:
9 ; ===== INPUT MATRIX (Test 1: 1..16) =====
0x000 10 LOAD s0, 5
0x001 11 LOAD s1, 1
0x002 12 LOAD s2, 9
0x003 13 LOAD s3, 11
14
0x004 15 LOAD s4, 2
0x005 16 LOAD s5, 4
0x006 17 LOAD s6, 8
0x007 18 LOAD s7, 10
19
0x008 20 LOAD s8, 13
0x009 21 LOAD s9, 3
0x00a 22 LOAD sA, 6
0x00b 23 LOAD sB, 7
24
0x00c 25 LOAD sC, 15
0x00d 26 LOAD sD, 14
0x00e 27 LOAD sE, 12
0x00f 28 LOAD sF, 16
29
30 ; =====
31 ; ROTATION PART
32 ; Each group (a,b,c,d) rotated: (d,a,b,c)
33 ; using XOR-based swaps: swap(a,b), swap(a,c), swap(a,d)
34 ; =====
35
36 ; ----- Group 1: corners (s0, s3, sF, sC) -----
37 ; a = s0, b = s3, c = sF, d = sC
38
39 ; swap(s0, s3)
0x010 40 XOR s0, s3
0x011 41 XOR s3, s0
0x012 42 XOR s0, s3
43
44 ; swap(s0, sF)
0x013 45 XOR s0, sF
0x014 46 XOR sF, s0

```

**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

Source Navigator   Processor core

	rotate3x3.psm [2]	rotate3x3.psm	rotate4x4.psm
<b>PC:</b> J052 <b>PAGE0</b> <b>HWBuild:</b> 000			
<b>Carry</b> 0 <b>Zero</b> 0 <b>Int</b> ■			
Bank: A ▾			
<b>s0</b> 013 s0 <b>s1</b> 009 s1 <b>s2</b> 005 s2 <b>s3</b> 001 s3 <b>s4</b> 014 s4 <b>s5</b> 010 s5 <b>s6</b> 006 s6 <b>s7</b> 002 s7 <b>s8</b> 015 s8 <b>s9</b> 011 s9 <b>sA</b> 007 sA <b>sB</b> 003 sB <b>sC</b> 016 sC <b>sD</b> 012 sD <b>sE</b> 008 sE <b>sF</b> 004 sF			
<b>0x00</b> 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 <b>0x08</b> 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 <b>0x10</b> 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 <b>0x18</b> 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 <b>0x20</b> 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 <b>0x28</b> 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 <b>0x30</b> 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 <b>0x38</b> 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000	1 ; 4x4 matrix rotate 90° clockwise 2 ; Matrix in registers: 3 ; s0 s1 s2 s3 4 ; s4 s5 s6 s7 5 ; s8 s9 sA sB 6 ; sC sD sE sF 7 8 <b>start:</b> 9 ; ===== INPUT MATRIX (Test 1: 1..16) ===== 0x000 10 LOAD s0, 1 0x001 11 LOAD s1, 2 0x002 12 LOAD s2, 3 0x003 13 LOAD s3, 4 14 0x004 15 LOAD s4, 5 0x005 16 LOAD s5, 6 0x006 17 LOAD s6, 7 0x007 18 LOAD s7, 8 19 0x008 20 LOAD s8, 9 0x009 21 LOAD s9, 10 0x00a 22 LOAD sA, 11 0x00b 23 LOAD sB, 12 24 0x00c 25 LOAD sC, 13 0x00d 26 LOAD sD, 14 0x00e 27 LOAD sE, 15 0x00f 28 LOAD sF, 16 29 30 ; =====PART===== 31 ; ROTATION PART 32 ; Each group (a,b,c,d) rotated: (d,a,b,c) 33 ; using XOR-based swaps: swap(a,b), swap(a,c), swap(a,d) 34 ; ===== 35 36 ; ----- Group 1: corners (s0, s3, sF, sC) ----- 37 ; a = s0, b = s3, c = sF, d = sC 38 39 ; swap(s0, s3) 0x010 40 XOR s0, s3 0x011 41 XOR s3, s0 0x012 42 XOR s0, s3 43 44 ; swap(s0, sF) 0x013 45 XOR s0, sF 0x014 46 XOR sF, s0		

**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

Source Navigator   Processor core

PC: J052 PAGE0   HWBuild: 000

Carry 0 Zero 0 Int

Bank: A

```

rotate3x3.psm [2]   rotate3x3.psm   rotate4x4.psm

1 ; 4x4 matrix rotate 90° clockwise
2 ; Matrix in registers:
3 ;   s0 s1 s2 s3
4 ;   s4 s5 s6 s7
5 ;   s8 s9 sA sB
6 ;   sC sD sE sF
7
8 start:
9 ; ===== INPUT MATRIX (Test 1: 1..16) =====
0x000 10 LOAD s0, 10
0x001 11 LOAD s1, 20
0x002 12 LOAD s2, 30
0x003 13 LOAD s3, 40
0x004 14
0x005 15 LOAD s4, 50
0x006 16 LOAD s5, 60
0x007 17 LOAD s6, 70
0x008 18 LOAD s7, 80
0x009 19
0x00a 20 LOAD s8, 90
0x00b 21 LOAD s9, 100
0x00c 22 LOAD sA, 110
0x00d 23 LOAD sB, 120
0x00e 24
0x00f 25 LOAD sC, 130
0x010 26 LOAD sD, 140
0x011 27 LOAD sE, 150
0x012 28 LOAD sF, 160
0x013 29
0x014 30 ; =====
0x015 31 ; ROTATION PART
0x016 32 ; Each group (a,b,c,d) rotated: (d,a,b,c)
0x017 33 ; using XOR-based swaps: swap(a,b), swap(a,c), swap(a,d)
0x018 34 ; =====
0x019 35 ; ----- Group 1: corners (s0, s3, sF, sC) -----
0x01a 36 ; a = s0, b = s3, c = sF, d = sC
0x01b 37
0x01c 38
0x01d 39 ; swap(s0, s3)
0x01e 40 XOR s0, s3
0x01f 41 XOR s3, s0
0x020 42 XOR s0, s3
0x021 43
0x022 44 ; swap(s0, sF)
0x023 45 XOR s0, sF
0x024 46 XOR sF, s0

```

**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

Source Navigator   Processor core

PC: J052 PAGE0   HWBuild: 000

Carry 0 Zero 0 Int █

Bank: A ▾

```

s0 039 s0
s1 027 s1
s2 015 s2
s3 003 s3
s4 042 s4
s5 030 s5
s6 018 s6
s7 006 s7
s8 045 s8
s9 033 s9
sA 021 sA
sB 009 sB
sC 048 sC
sD 036 sD
sE 024 sE
sF 012 sF

0x00 000 000 000 000 000 000 000 000
0x08 000 000 000 000 000 000 000 000
0x10 000 000 000 000 000 000 000 000
0x18 000 000 000 000 000 000 000 000
0x20 000 000 000 000 000 000 000 000
0x28 000 000 000 000 000 000 000 000
0x30 000 000 000 000 000 000 000 000
0x38 000 000 000 000 000 000 000 000
```

	rotate3x3.psm [2]	rotate3x3.psm	rotate4x4.psm
1	1 ; 4x4 matrix rotate 90° clockwise		
2	2 ; Matrix in registers:		
3	3 ; s0 s1 s2 s3		
4	4 ; s4 s5 s6 s7		
5	5 ; s8 s9 sA sB		
6	6 ; sC sD sE sF		
7			
8	<b>start:</b>		
9	9 ; ===== INPUT MATRIX (Test 1: 1..16) =====		
10	10 LOAD s0, 3		
11	11 LOAD s1, 6		
12	12 LOAD s2, 9		
13	13 LOAD s3, 12		
14			
15	15 LOAD s4, 15		
16	16 LOAD s5, 18		
17	17 LOAD s6, 21		
18	18 LOAD s7, 24		
19			
20	20 LOAD s8, 27		
21	21 LOAD s9, 30		
22	22 LOAD sA, 33		
23	23 LOAD sB, 36		
24			
25	25 LOAD sC, 39		
26	26 LOAD sD, 42		
27	27 LOAD sE, 45		
28	28 LOAD sF, 48		
29			
30	30 ; =====		
31	31 ; ROTATION PART		
32	32 ; Each group (a,b,c,d) rotated: (d,a,b,c)		
33	33 ; using XOR-based swaps: swap(a,b), swap(a,c), swap(a,d)		
34	34 ; =====		
35	35 ; ----- Group 1: corners (s0, s3, sF, sC) -----		
36	36 ; a = s0, b = s3, c = sF, d = sC		
37			
38			
39	39 ; swap(s0, s3)		
40	40 XOR s0, s3		
41	41 XOR s3, s0		
42	42 XOR s0, s3		
43			
44	44 ; swap(s0, sF)		
45	45 XOR s0, sF		
46	46 XOR sF, s0		

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

Source Navigator   Processor core

PC: 034 PAGE0   HWBuild: 00

Carry 0 Zero 0 Int ■

Bank: A ▾

```

s0 07 s0
s1 07 s1
s2 07 s2
s3 07 s3
s4 07 s4
s5 07 s5
s6 07 s6
s7 07 s7
s8 07 s8
s9 07 s9
sA 07 sA
sB 07 sB
sC 07 sC
sD 07 sD
sE 07 sE
sF 07 sF

0x00 00 00 00 00 00 00 00 00 00
0x08 00 00 00 00 00 00 00 00 00
0x10 00 00 00 00 00 00 00 00 00
0x18 00 00 00 00 00 00 00 00 00
0x20 00 00 00 00 00 00 00 00 00
0x28 00 00 00 00 00 00 00 00 00
0x30 00 00 00 00 00 00 00 00 00
0x38 00 00 00 00 00 00 00 00 00

```

rotate3x3.psm [2]   rotate3x3.psm   rotate4x4.psm

```

1 ; 4x4 matrix rotate 90° clockwise
2 ; Matrix in registers:
3 ;   s0 s1 s2 s3
4 ;   s4 s5 s6 s7
5 ;   s8 s9 sA sB
6 ;   sC sD sE sF
7
8 start: ; ===== INPUT MATRIX (Test 1: 1..16) =====
9    LOAD s0, 7
10   LOAD s1, 7
11   LOAD s2, 7
12   LOAD s3, 7
13
14   LOAD s4, 7
15   LOAD s5, 7
16   LOAD s6, 7
17   LOAD s7, 7
18
19   LOAD s8, 7
20   LOAD s9, 7
21   LOAD sA, 7
22   LOAD sB, 7
23
24   LOAD sC, 7
25   LOAD sD, 7
26   LOAD sE, 7
27   LOAD sF, 7
28
29
30 ; =====
31 ; ROTATION PART
32 ; Each group (a,b,c,d) rotated: (d,a,b,c)
33 ; using XOR-based swaps: swap(a,b), swap(a,c), swap(a,d)
34 ; =====
35
36 ; ----- Group 1: corners (s0, s3, sF, sC) -----
37 ; a = s0, b = s3, c = sF, d = sC
38
39 ; swap(s0, s3)
40 XOR s0, s3
41 XOR s3, s0
42 XOR s0, s3
43
44 ; swap(s0, sF)
45 XOR s0, sF
46 XOR sF, s0

```

**Input:**

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

**Output:**

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