FAST School of Computing

Computer Organization and Assembly Language

MATRIX CALCULATOR

Final Project Report

Instructor: Mr. Ubaidullah

Section: BS-AI 4A

Group Members:

- Adina Faraz (23K-0008)
- Syed Muneeb Ur Rehman (23K-0038)

Course:

Computer Organization and Assembly Language
Instructor: Mr. Ubaidullah
Section: BS-AI 4A

Group Members:

- Adina Faraz (23K-0008)
- Syed Muneeb Ur Rehman (23K-0038)

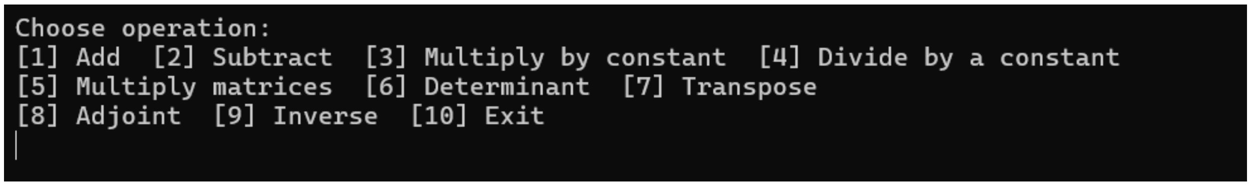
1. Project Overview

The Matrix Calculator project was developed as part of the course on Computer Organization and Assembly Language. Its primary goal was to implement fundamental and advanced matrix operations using the MASM32 Assembly Language in a console-based application. The project helped deepen our understanding of low-level programming and how to apply it to mathematical computations.

2. Objectives

- To implement basic matrix operations (Addition, Subtraction, Multiplication, and Division).

- To implement advanced matrix functionalities (Transpose, Determinant, Adjoint, and Inverse) for 2×2 and 3×3 matrices.
- To build a modular, efficient, and interactive calculator in Assembly Language.
- To gain hands-on experience with MASM32, memory management, and console I/O in a low-level language.



```
Choose operation:
[1] Add  [2] Subtract  [3] Multiply by constant  [4] Divide by a constant
[5] Multiply matrices  [6] Determinant  [7] Transpose
[8] Adjoint  [9] Inverse  [10] Exit
```

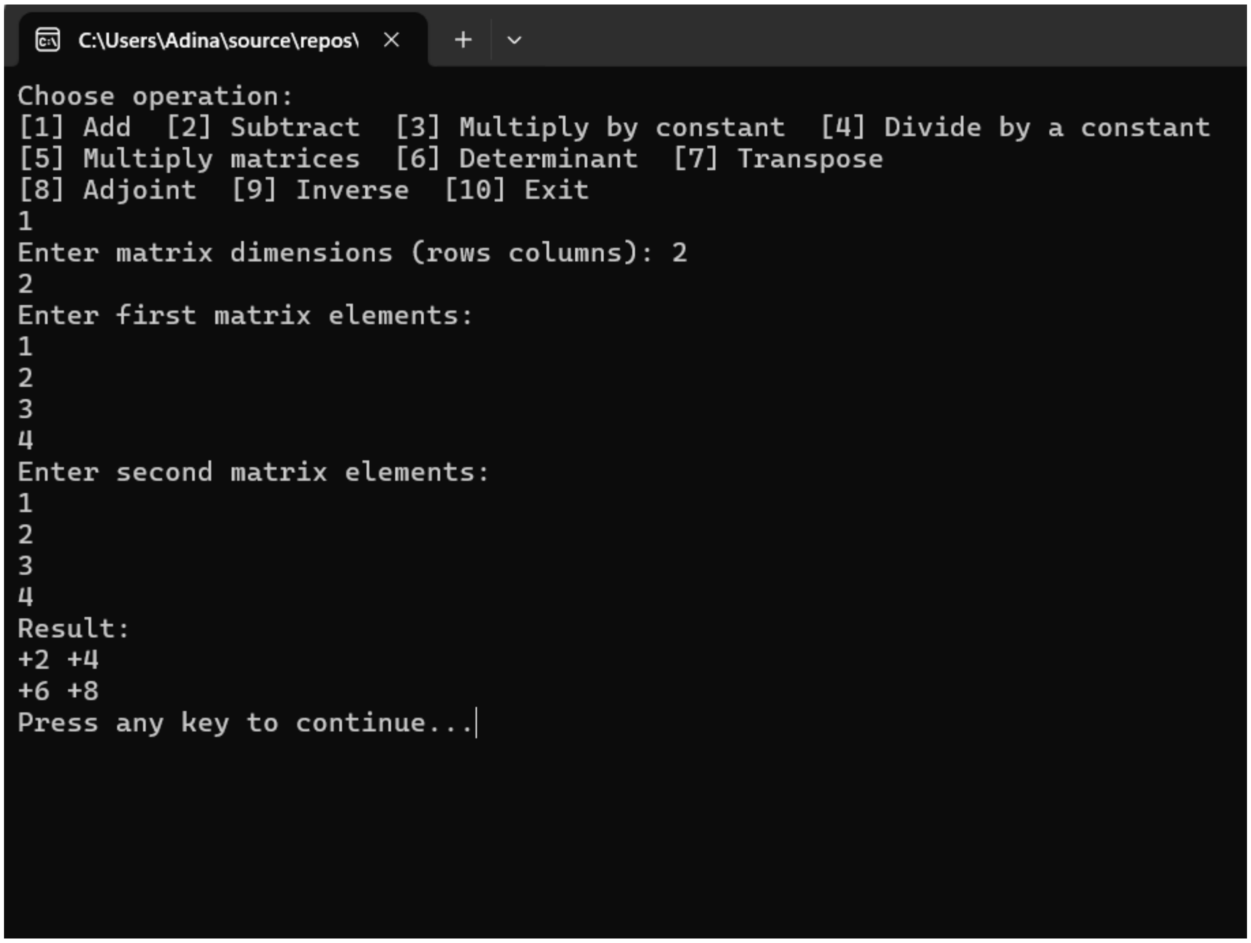
3. Tools and Technologies

- **Language:** MASM32 Assembly
- **Development Kit:** MASM32 SDK
- **Platform:** Windows OS
- **Interface:** Console-based User Interface

4. System Features

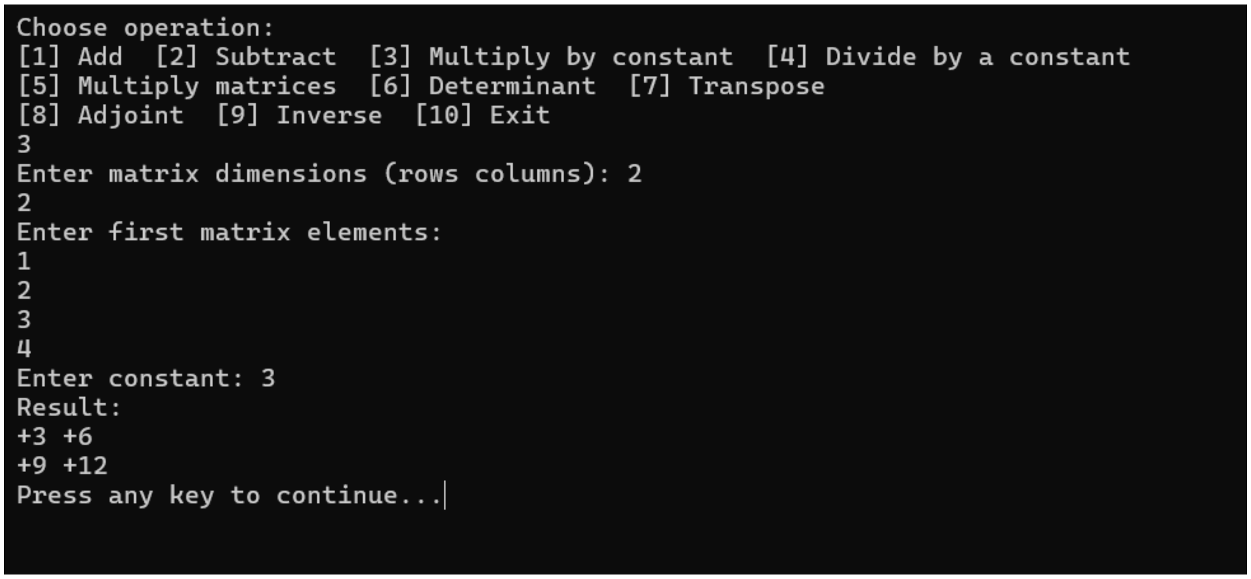
Basic Functionalities

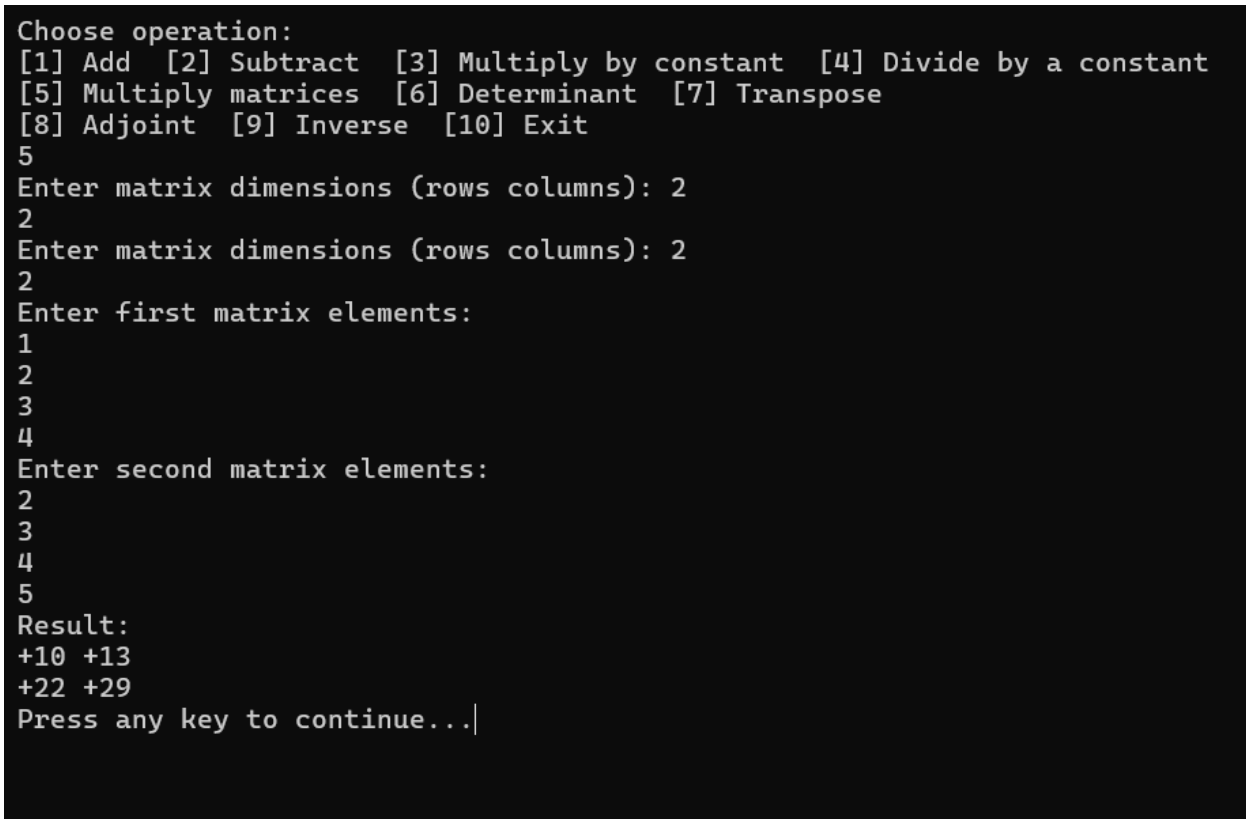
- Addition and Subtraction of matrices of any n × n size



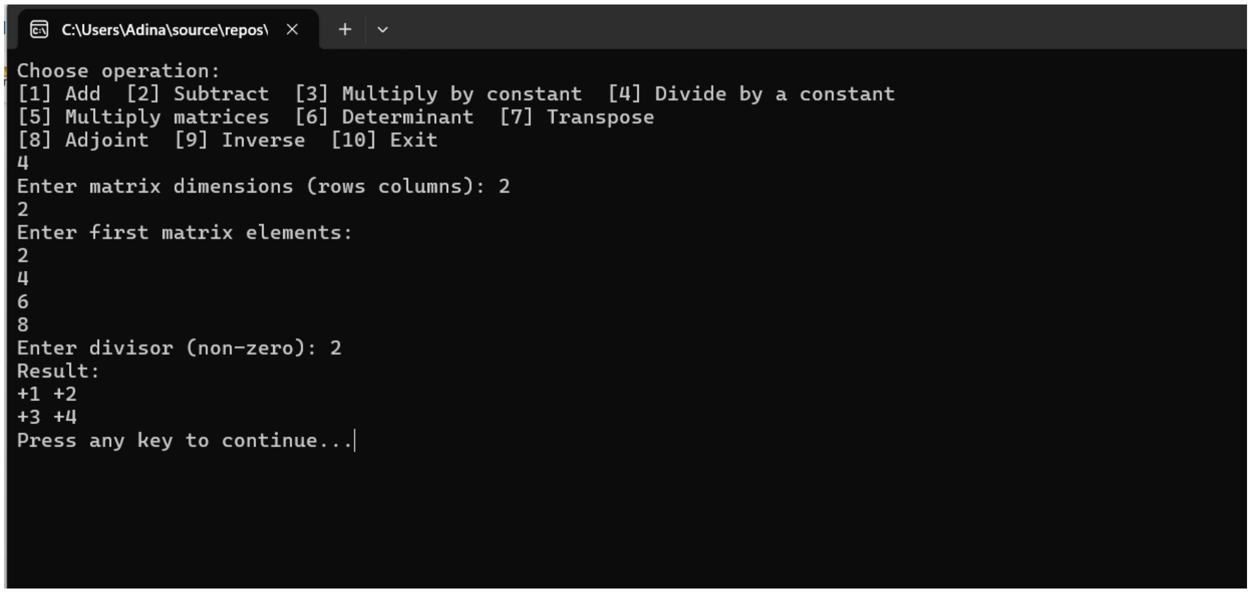
```
Choose operation:
[1] Add  [2] Subtract  [3] Multiply by constant  [4] Divide by a constant
[5] Multiply matrices  [6] Determinant  [7] Transpose
[8] Adjoint  [9] Inverse  [10] Exit
1
Enter matrix dimensions (rows columns): 2
2
Enter first matrix elements:
1
2
3
4
Enter second matrix elements:
1
2
3
4
Result:
+2 +4
+6 +8
Press any key to continue...
```

- Multiplication of compatible matrices


```
Choose operation:
[1] Add  [2] Subtract  [3] Multiply by constant  [4] Divide by a constant
[5] Multiply matrices  [6] Determinant  [7] Transpose
[8] Adjoint  [9] Inverse  [10] Exit
3
Enter matrix dimensions (rows columns): 2
2
Enter first matrix elements:
1
2
3
4
Enter constant: 3
Result:
+3 +6
+9 +12
Press any key to continue...
```


```
Choose operation:
[1] Add  [2] Subtract  [3] Multiply by constant  [4] Divide by a constant
[5] Multiply matrices  [6] Determinant  [7] Transpose
[8] Adjoint  [9] Inverse  [10] Exit
5
Enter matrix dimensions (rows columns): 2
2
Enter matrix dimensions (rows columns): 2
2
Enter first matrix elements:
1
2
3
4
Enter second matrix elements:
2
3
4
5
Result:
+10 +13
+22 +29
Press any key to continue...
```

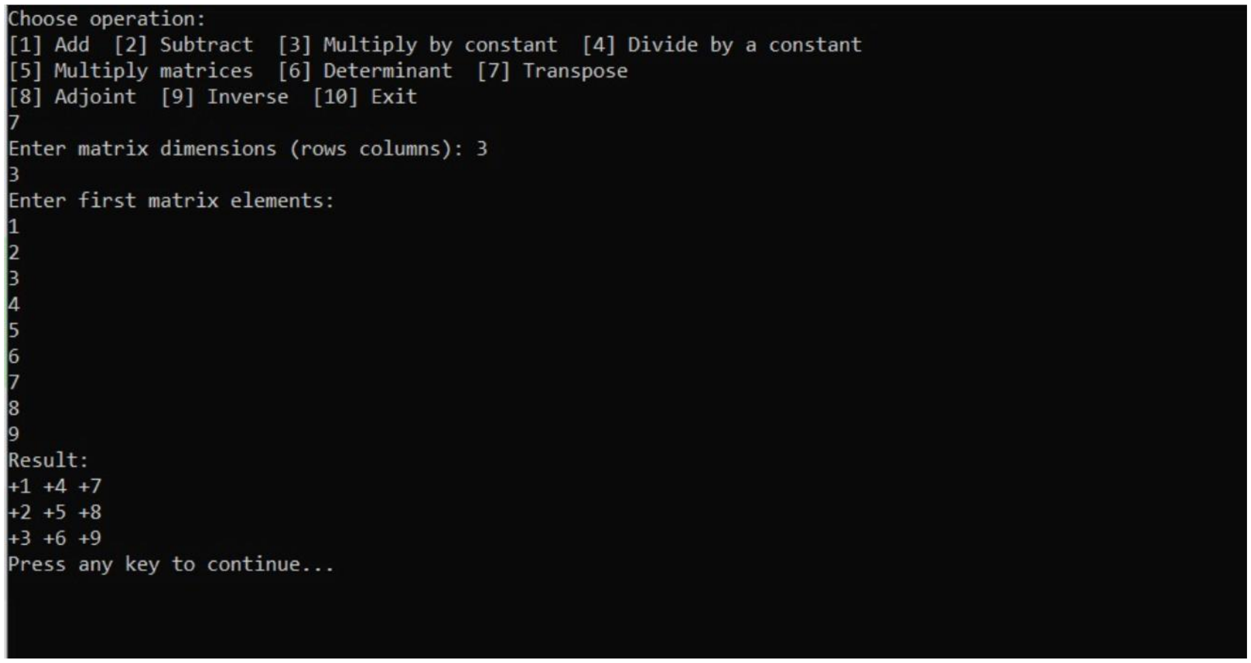
- Division by a constant



```
Choose operation:
[1] Add  [2] Subtract  [3] Multiply by constant  [4] Divide by a constant
[5] Multiply matrices  [6] Determinant  [7] Transpose
[8] Adjoint  [9] Inverse  [10] Exit
4
Enter matrix dimensions (rows columns): 2
2
Enter first matrix elements:
2
4
6
8
Enter divisor (non-zero): 2
Result:
+1 +2
+3 +4
Press any key to continue...
```

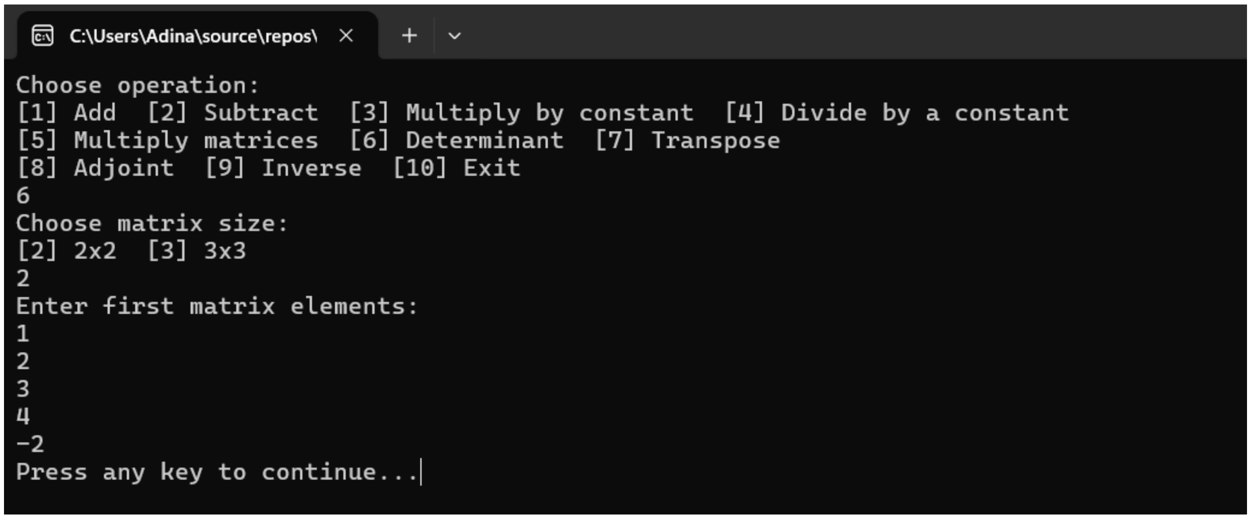
Advanced Functionalities (2×2 and 3×3 matrices only)

- Transpose



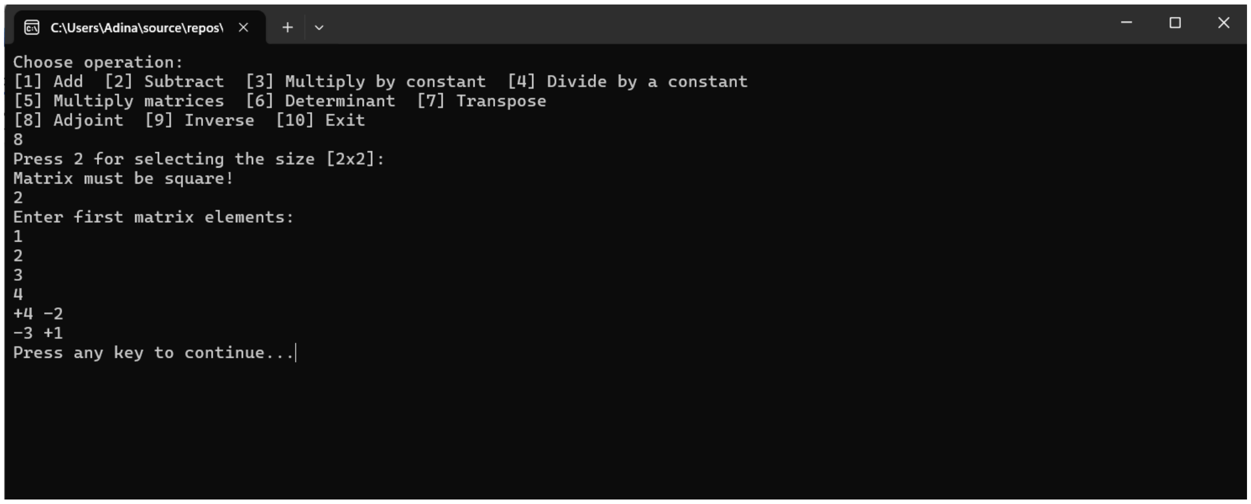
```
Choose operation:
[1] Add  [2] Subtract  [3] Multiply by constant  [4] Divide by a constant
[5] Multiply matrices  [6] Determinant  [7] Transpose
[8] Adjoint  [9] Inverse  [10] Exit
7
Enter matrix dimensions (rows columns): 3
3
Enter first matrix elements:
1
2
3
4
5
6
7
8
9
Result:
+1 +4 +7
+2 +5 +8
+3 +6 +9
Press any key to continue...
```

- Determinant calculation

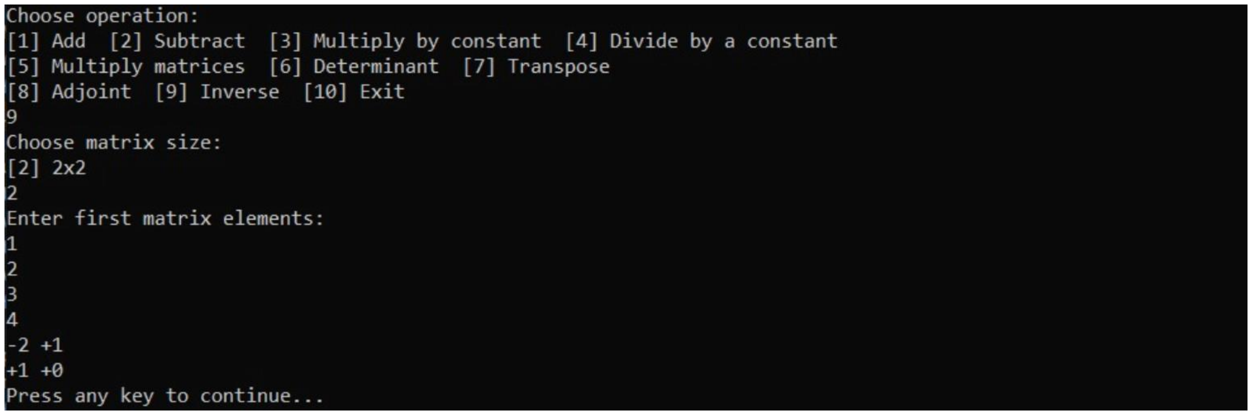


```
Choose operation:
[1] Add  [2] Subtract  [3] Multiply by constant  [4] Divide by a constant
[5] Multiply matrices  [6] Determinant  [7] Transpose
[8] Adjoint  [9] Inverse  [10] Exit
6
Choose matrix size:
[2] 2x2  [3] 3x3
2
Enter first matrix elements:
1
2
3
4
-2
Press any key to continue...
```

- Adjoint and Inverse of 2x2 matrices (if determinant ≠ 0)



```
Choose operation:
[1] Add  [2] Subtract  [3] Multiply by constant  [4] Divide by a constant
[5] Multiply matrices  [6] Determinant  [7] Transpose
[8] Adjoint  [9] Inverse  [10] Exit
8
Press 2 for selecting the size [2x2]:
Matrix must be square!
2
Enter first matrix elements:
1
2
3
4
+4 -2
-3 +1
Press any key to continue...
```



```
Choose operation:
[1] Add  [2] Subtract  [3] Multiply by constant  [4] Divide by a constant
[5] Multiply matrices  [6] Determinant  [7] Transpose
[8] Adjoint  [9] Inverse  [10] Exit
9
Choose matrix size:
[2] 2x2
2
Enter first matrix elements:
1
2
3
4
-2 +1
+1 +0
Press any key to continue...
```

Other Capabilities

- Dynamic matrix size input by the user
- Manual entry of matrix elements
- Input validation and error messages for incorrect inputs or invalid operations

5. Implementation Summary

The application was structured using a modular approach. Each operation was coded as a separate subroutine, ensuring clarity and ease of debugging.

- Arrays were simulated using memory blocks for storing matrix elements
- Console input was handled using MASM's standard input functions
- Output formatting required careful handling of ASCII values and spacing to simulate tabular matrix display
- For the inverse operation, determinant calculation and matrix of minors were implemented manually using cofactor expansion

6. Key Challenges and Resolutions

Challenge	Resolution

Implementing Division	Used inverse multiplication where applicable, ensuring determinant $\neq 0$
Determinant for 3×3	Used cofactor expansion approach coded from scratch
Matrix Input Handling	Created a robust subroutine that validates user input
Output Formatting	Developed custom routines to print matrices in structured format

7. Testing and Results

The calculator was tested with:

- Square matrices ranging from 2×2 to 5×5 for basic operations including transpose
- Special 2×2 matrices for inverse, adjoint and determinant validation
- Edge cases like:
  - Singular matrices (determinant = 0)
  - Mismatched dimensions for multiplication
  - Division by zero or any non-numeric value

All tests passed successfully, with accurate results and graceful error handling for invalid operations.

8. Future Work

- Implement a GUI using C++ or C# to interface with the MASM backend
- Extend support for 4×4 and higher-order matrices in advanced operations
- Add more features like eigenvalues/eigenvectors, row-reduced echelon forms, or LU decomposition
- Optimize performance for large matrices

9. Conclusion

The Matrix Calculator was a successful project that combined theoretical knowledge of matrix algebra with practical low-level programming in MASM32. It demonstrated the power of assembly language in handling memory and computational logic directly. The structured approach and hands-on experience gained through this project enhanced our understanding of both matrix computations and assembly language programming.