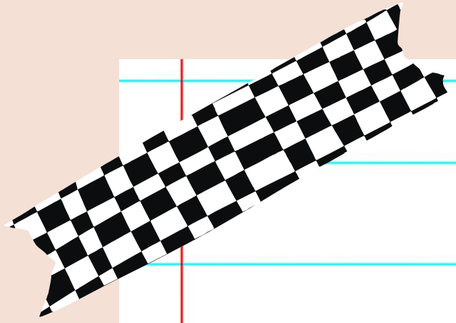

TIC TAC TOE
G.A.M.E

Hello we are

Muhammad Muzammil	BDA-24S-043
Saud Ahmed	BDA-24S-013
Ahmed tanzeel	BDA-24S-030

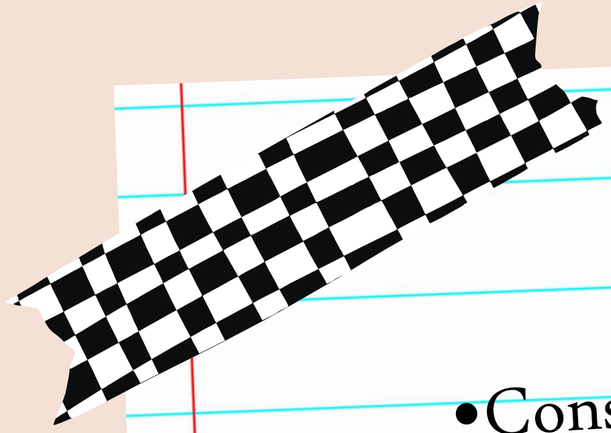


Introduction

This project is a console-based multi-level Tic Tac Toe game developed in Assembly Language using TASM.

The game supports three different grid sizes: 3x3, 4x4, and 5x5.

It demonstrates low-level programming concepts such as arrays, loops, procedures, conditional branching, and DOS interrupt-based input/output.

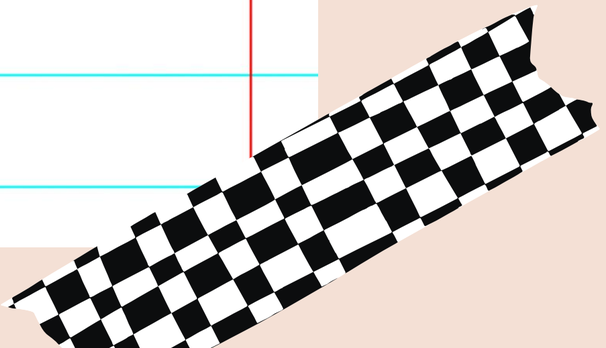


Goals

- Console-based interactive game
- Multiple difficulty levels
- Input validation
- Winner detection
- Restart & exit functionality
- Strong understanding of Assembly procedures

Why this Project?

- Simple game, complex logic
- Perfect for understanding low-level programming
- Covers memory handling & control flow
- Demonstrates real use of procedures



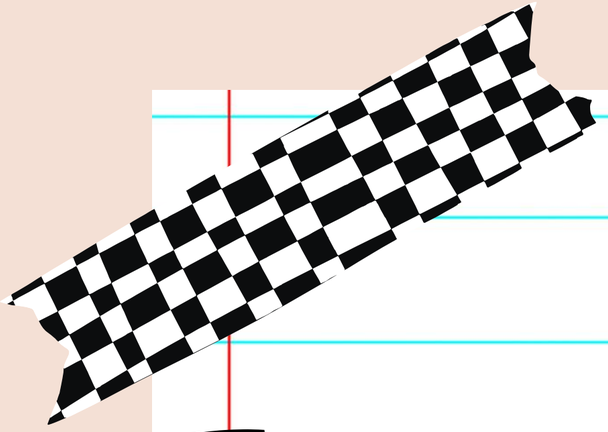
TOOLS & SOFTWARE USED

Turbo Assembler (TASM)

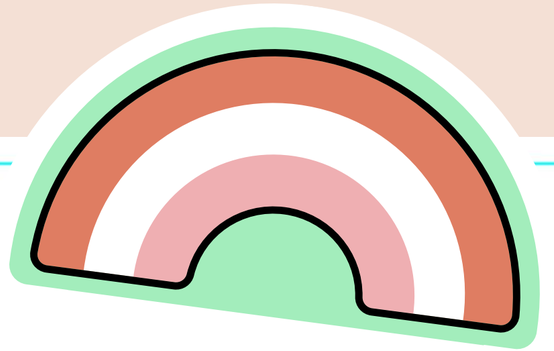
Turbo Linker (TLINK)

DOS Box

Windows OS



Game Flow



Difficulty selection

↓

Board display

↓

Player turn (X / O)

↓

Input validation

↓

Board update

↓

Win / Draw check

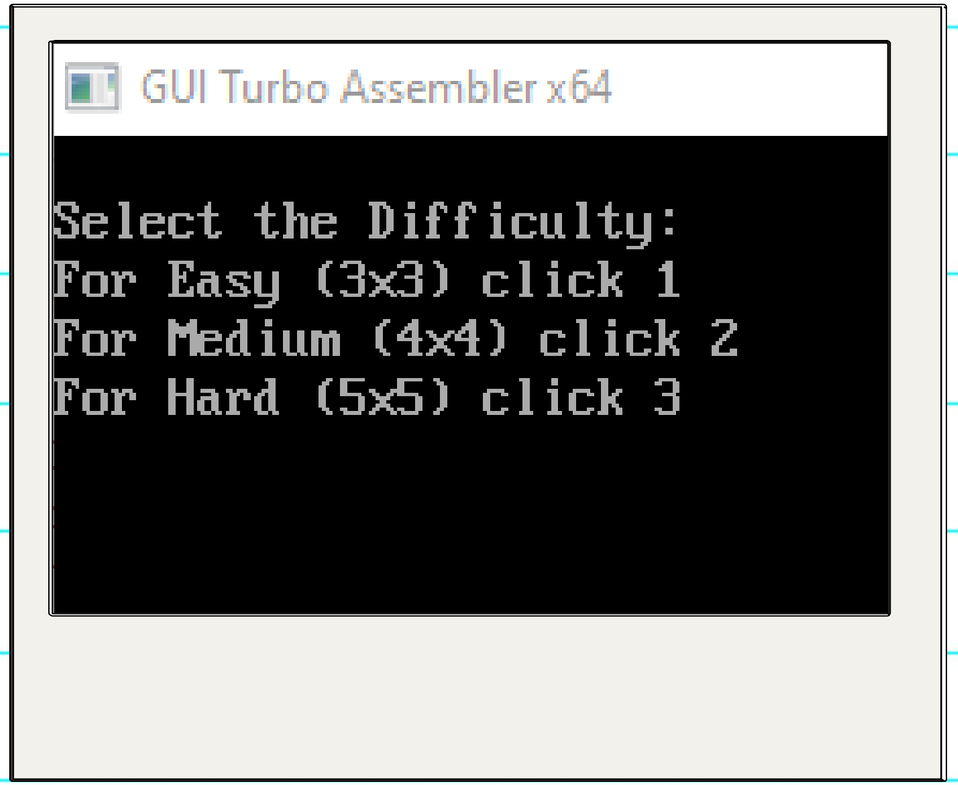
↓

Restart or Exit

Menu System

- Easy ⟶ 3x3
- Medium ⟶ 4x4
- Hard ⟶ 5x5

Implemented using INT 21h

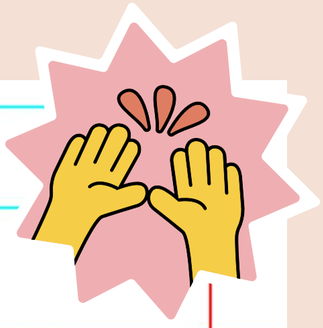


3x3 Grid Easy Mode

- 9 positions (1–9)
- Simple Win Logic
- Fast Gameplay
- Beginner Friendly

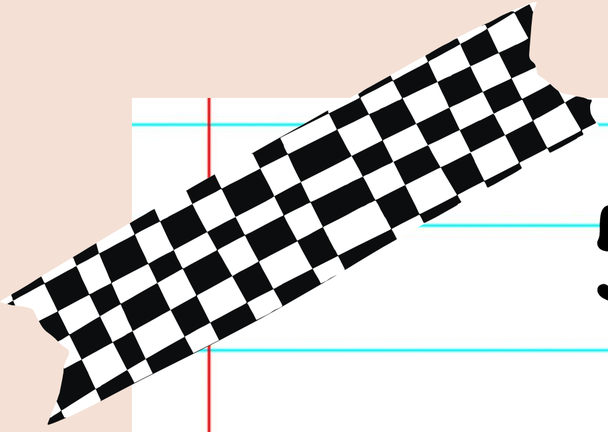
```
--- TIC TAC TOE GAME ---
 |1|2|3|
 |4|5|6|
 |7|8|9|
Player X enter position (1-9):
```

4x4 Grid Medium Mode



- 16 Positions
- Input 1 – 9 & A – G
- More Winning Combinations
- Improved Logic Complexity

```
--- TIC TAC TOE 4x4 GAME ---
¦1¦2¦3¦4
¦5¦6¦7¦8
¦9¦A¦B¦C
¦D¦E¦F¦G
Player X enter position (1-G):
```



5x5 Grid Hard Mode

- 25 Positions
- Input 1 – 9 & A – P
- Complex Win Detection
- Maximum Use Of Arrays & Loops

```
--- TIC TAC TOE 5x5 GAME ---
|1|2|3|4|5
|6|7|8|9|A
|B|C|D|E|F
|G|H|I|J|K
|L|M|N|O|P
Player X enter position (1-P):
```

Invalid or Used Position!

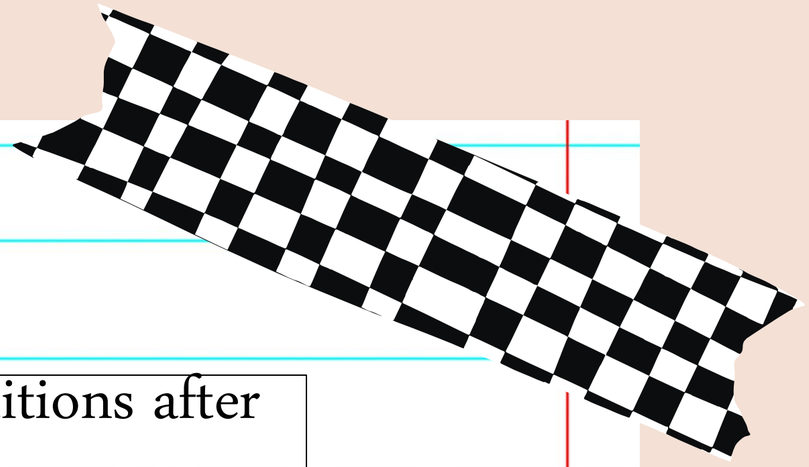
Input validation

The program validates every user input before processing the move.

Invalid positions and already occupied cells are not allowed.

An error message is displayed and the player is prompted again.

This ensures fair gameplay and prevents logical errors.



Winner Detection Logic

- The game checks for winning conditions after every move.
- Rows, columns, and diagonals are compared for identical symbols.
- Separate win-checking logic is implemented for 3x3, 4x4, and 5x5 grids.
- A flag is returned to indicate win, draw, or continuation of the game.

```
Press R to redo or E to exit:
```

Restart And Exit Functionality

After a win or draw, the user is given options to restart or exit.

Pressing R restarts the game with a fresh board and reset counters.

Pressing E exits the program safely using DOS interrupt 21h.

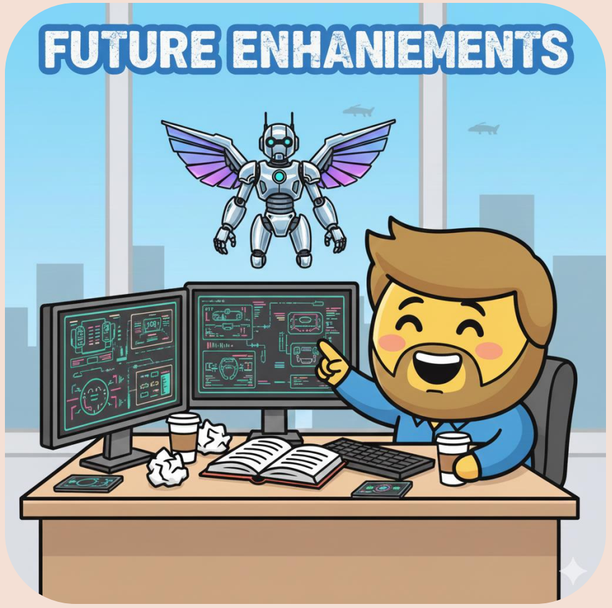
This feature improves user control and reliability.

Challenges Faced

- Converting alphanumeric input into correct array indices.
- Designing win detection logic for larger grids (4x4 and 5x5).
- Managing dynamic board display with proper formatting.
- Resetting boards and counters without causing memory errors.

Future Enhancement

- Implementation of single-player mode with AI logic.
- Addition of score tracking for multiple rounds.
- Use of colors and sound effects for better user experience.
- Support for larger or customizable grid sizes.



Conclusion

Conclusion

- A complete multi-level Tic Tac Toe game developed in Assembly Language.
- The project demonstrates strong low-level logic and program structure.
- It effectively combines learning with interactive gameplay.
- The project serves as a solid foundation for advanced Assembly applications.

```
.MODEL SMALL
.STACK 100H
.DATA
MSG DB 'CONCLUSION!$'
.CODE
START:

MOV AX , @DATA
MOV DS , AX

MOV AH , 09H
MOV DX , OFFSET MSG
INT 21H

MOV AH , 4CH
INT 21H

END START
```

THANK YOU!