Tic Tac Toe – Project Report

Group Members:

- Syed Saif Ur Rehman Shah (23k-0032)
- Muhammed Umer (23k-0023)
- Muhammed Abser Mansoor (23k-0030)

Introduction:

This project is a fully functional Tic Tac Toe game implemented in x86 Assembly language using the Irvine32 library. The game simulates the classic 3x3 board and supports three modes:
- Easy mode with a random AI
- Hard mode powered by the Minimax algorithm
- Two-player mode for local multiplayer fun

Building a game like this in Assembly was both challenging and rewarding, as it required low-level memory and logic management.

Project Features:

Game Board Logic

The 3x3 board is represented using a 9-byte array (board) where:
- 0 = Empty cell
- -1 = Player X's move
- 1 = Player O's move

Each cell is updated as players take their turns, and the board is printed after every move.

Game Modes

On launching the game, the user selects from three modes:
1. Easy – Computer plays randomly.
2. Hard – Computer uses the Minimax algorithm to always make the optimal move.
3. 2 Player – Two users play by taking alternate turns.

Turn Management

The game alternates turns between X and O using the currentPlayer variable.
Players are prompted to enter a move (1-9), which is validated to ensure it's within bounds and the cell is unoccupied.
After each move, the game checks for win or tie conditions.

AI Modes

Easy AI: Selects a move randomly from available spots using randomrange.
Hard AI: Implements the Minimax algorithm, which recursively evaluates all possible game outcomes and chooses the optimal move.

Game Evaluation Logic

The evaluate function determines:
- If X has won: returns -1
- If O has won: returns 1
- If it's a tie: returns 0
- If the game is still ongoing: returns 2

It checks for:
- Row and column-wise matches
- Both diagonals
- Remaining empty cells (for a tie)

Input and Validation

Player moves are entered via keyboard (as string input).
The program parses and validates input using parseinteger32.
Invalid moves prompt re-entry.

Display Functionality

printBoard prints the current game board using X, O, or . (dot) for empty spots.
After each turn, the board is updated visually.

Challenges:

1. Assembly-Level Minimax: Implementing a recursive algorithm like Minimax without built-in call stack management was tricky.
2. User Input Handling: Parsing and validating string input in Assembly required low-level buffer management.
3. Game State Management: Had to manage all game flow manually through registers and memory offsets.

Conclusion:

This Tic Tac Toe project provided our group with an opportunity to apply Assembly programming in a fun and interactive way. It helped us dive deeper into algorithmic thinking at the machine level. The project demonstrated how even a simple game could become a complex challenge without the abstraction of modern programming tools.