Final Report

Computer Organization and Assembly Language

Department Of Software Engineering

SMIU

Tic Tac Toe Game

Member 1: Muhammad Muzammil

ID: BDA-24S-043

Member 2: Saud Ahmed

ID: BDA-24S-013

Member 3: Ahmed Tanzeel Khan

ID: BDA-24S-030

Assigned By: Miss Iqra

Table Of Contents

Abstract:

The project is a translation of the traditional Tic-Tac-Toe game on Assembly Language (TASM). The game can be played in three difficulty modes by two players in turn-by-turn mode:

1. 3x3 (Easy) standard Tic-Tac-Toe.
2. 4x4 (Medium) bigger grid with letters featured.
3. 5x5 (Hard) is a complex grid with 25 positions.

The project illustrates arrays, loops, conditional branching, procedural calls and interrupt based input/output of Assembly Language. It also focuses on input validation, board display, detecting winner and replay/exit.

The project helps in the realization of low level programming logic, memory management, and optimal algorithm design in Assembly.

Objectives:

1. To create a console-based interactive game in Assembly Language.
2. To implement multi-level Tic-Tac-Toe: 3x3, 4x4, 5x5 grids.
3. To implement user input validation to prevent invalid moves.
4. To develop winner detection logic for rows, columns, and diagonals.
5. To implement restart and exit functionality.
6. To enhance understanding of procedures, loops, and conditional jumps in Assembly.
7. To explore low-level game programming techniques.

Tools and Software:

- Turbo Assembler (TASM) to assemble .asm code files.
- Turbo Linker (TLINK) – to generate executable files from assembled code.
- DOSBox – to run the game in DOS-like environment on modern systems.

Project Description:

The Tic-Tac-Toe game is two player strategy game and the aim is to align the symbols(X or O) in a horizontal, vertical or diagonal manner.

Three levels of difficulty are applied in this project:

3x3 Grid (Easy) 9 cells; the winner is a row, column, or a diagonal with an identical symbol.

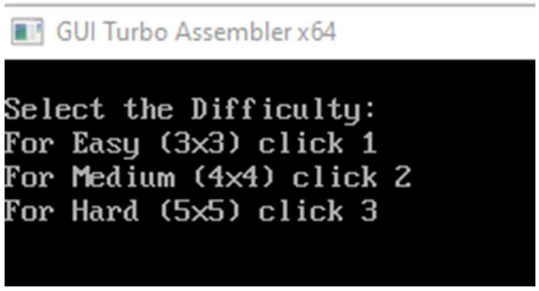
4x4 Grid (Medium) 16 cells, letters A-G following numbers 1-9. More complicated winning combinations.

5x5 Grid (Hard) 25 cells, labelled 1-9 and A-P; wins with a larger number of moves.

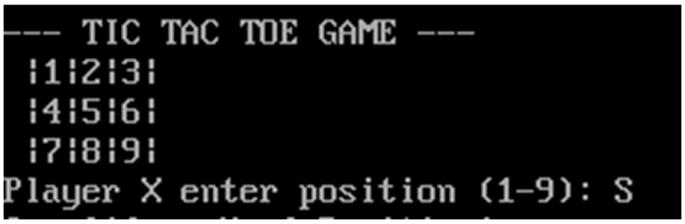
The game works as follows:

- Gamers are asked to choose the level of difficulty.
- The game board is shown in the program.
- The players alternate making a move.
- Input is checked to make sure of a correct cell and avoid overwriting.
- The updates regarding the board are shown after each move.
- After every move, winner or draw is detected.
- Gamers may resume the game or leave.

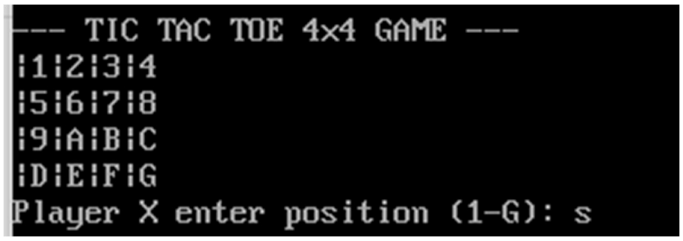
Menu:



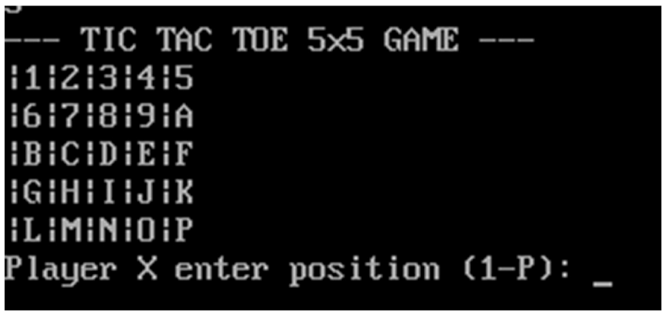
3x3 GRID:



4x4 GRID:



5x5 GRID:



Features and Functionalities:

1. Multiple Grid Levels:

- 3x3 Grid the normal gameplay in 9 positions.
- 4x4 Grid medium with letters in it (1-9 + A-G).
- 5x 5 Grid most difficult one with 25 cells (1-9 + A-P).

At the individual levels, there are independent processes of:

- Showboard1: a larger display of the board, showboard2: a smaller display, showboard3: a rotated display.
- Checking of winner (checkwin1, checkwin2, checkwin3)
- Handling input validation

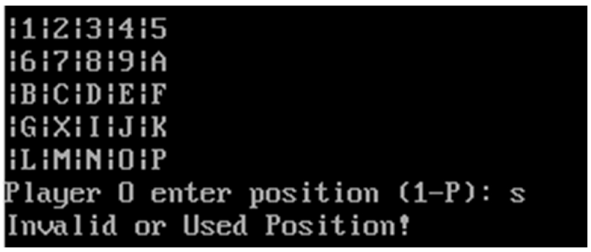
2. Board Display:

- DOS interrupt, function 09h is used to display boards.
- In each grids, there are rows which are separated with newlines and vertical bars.
- Board is updated with dynamic updates after every move.

```
; ---------- DISPLAY BOARD ---------
show_board1 proc
    lea dx, newline1
    mov ah, 09h
    int 21h

    mov si, 0

    mov dl, ' '
    mov ah, 02h
    int 21h

     mov dl, '|'
    mov ah, 02h
    int 21h

print_loop1:
    mov dl, board1[si]
    mov ah, 02h
    int 21h

    mov dl, '|'
    mov ah, 02h
```

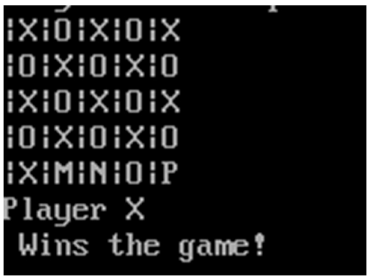
3. Player Input and Validation:

- The input is read through int 21h (function 01h).
- For 3x3 grid, numbers 1-9 are valid.
- In the case of 4x4 grid, numbers 1-9 and letters A-G are good.
- In the case of 5x5 grid, the numbers 1-9 and letters A-P can be used.
- An error message is displayed and the player is re-prompted in case of invalid input.



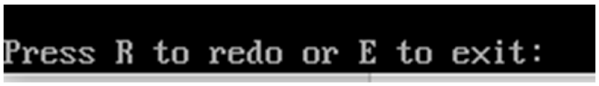
4. Winner Detection:

- Used in rows, columns and diagonals with conditional comparisons.
- The grid custom logic is provided as the cells number varies.
- Procedure gives back a flag to show a win, draw or continuation.



5. Game Restart and Exit:

- Following a win or draw, the players are asked:
  - R - Restart
  - E - Exit
- Restart replaces the board and player figure.
- Exit end the program with int 21h, procedure 4Ch.



6. Code Structure:

1. Data Section: Stores: board arrays, messages, player symbols, move counts.
2. Procedures: This section of the code contains:
   - mainloop - Gameplay loop.
   - showboard - Displays board
   - checkwin - Checks winner
   - askoption - restart or exit command.
3. Processes: Modular design permits clean and reusable and well-organized code.
4. Interrupts Used: DOS interrupt 21h to do input/output.

Challenges and Learning:

1. Input Conversion: Converting letters (A-G, A-P) to array indices.
2. Dynamic Board Display: Handling multiple line breaks and vertical bars.

3. Winner Detection Logic: Efficient comparison of rows, columns, and diagonals for larger grids.
4. Restart Option: Resetting arrays and counters without bugs.

Future Enhancements:

- Single-player mode with AI logic.
- Implement score tracking for multiple rounds.
- Add color-coded console output for better visual appeal.
- Implement sound notifications for moves and wins.
- Expand to larger grids or custom grid sizes.

Conclusion:

The project illustrates low-level programming in the Assembly and how complicated logic can be developed with ease. Tic-Tac-Toe brings into focus memory manipulation, loops, conditional branching and procedure calls.

It is an engaging project (a fun game) as well as a learning experience about Assembly, game logic, and algorithm development. The multi-level gameplay can make the project cope with the rising complexity easily.

References:

1. Irvine, Kip R. *Assembly Language for x86 Processors*, 7th Edition, Pearson, 2015.
2. Turbo Assembler (TASM) Official Documentation.
3. DOSBox Documentation for running DOS applications.
4. Online resources for Assembly Tic-Tac-Toe logic.