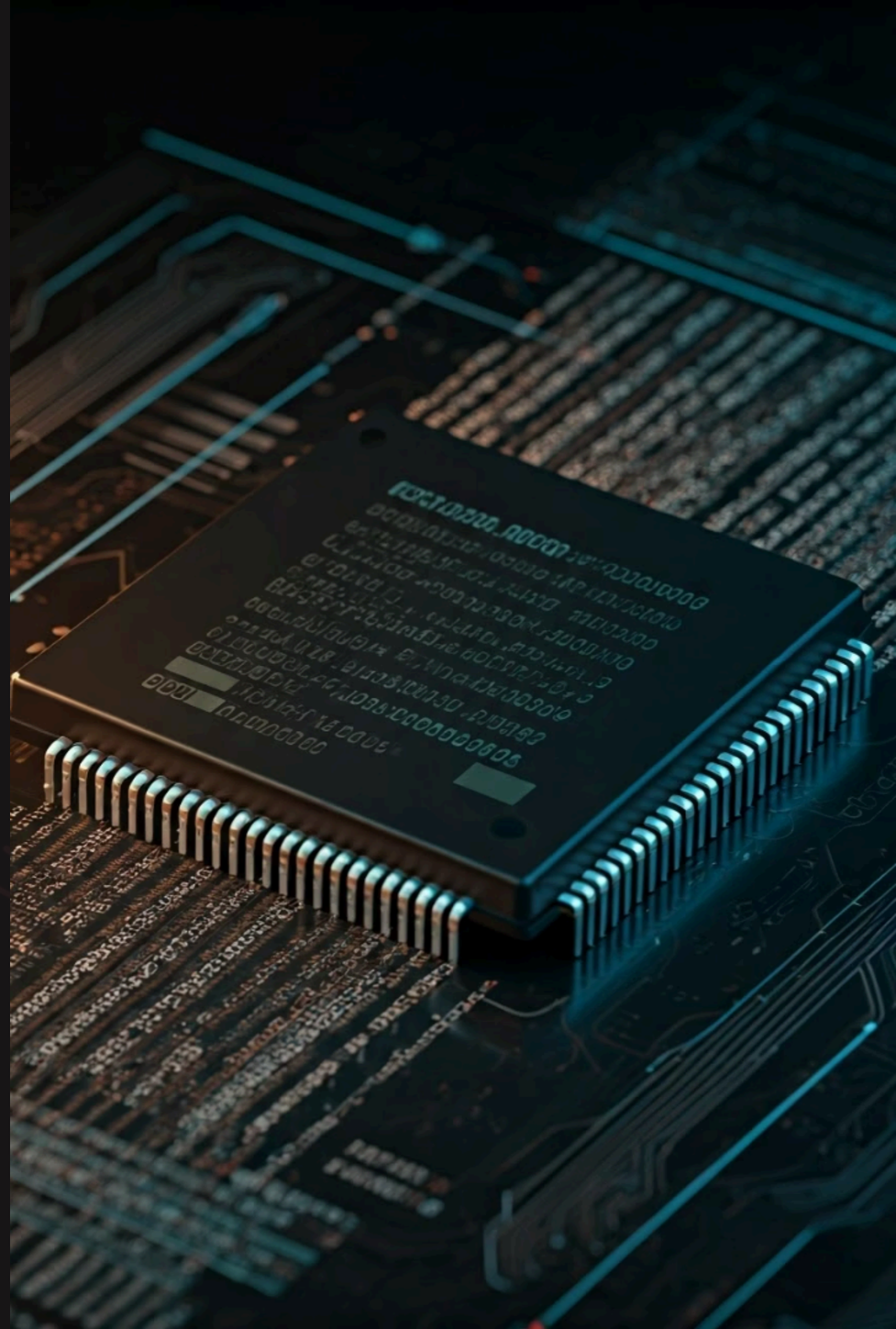
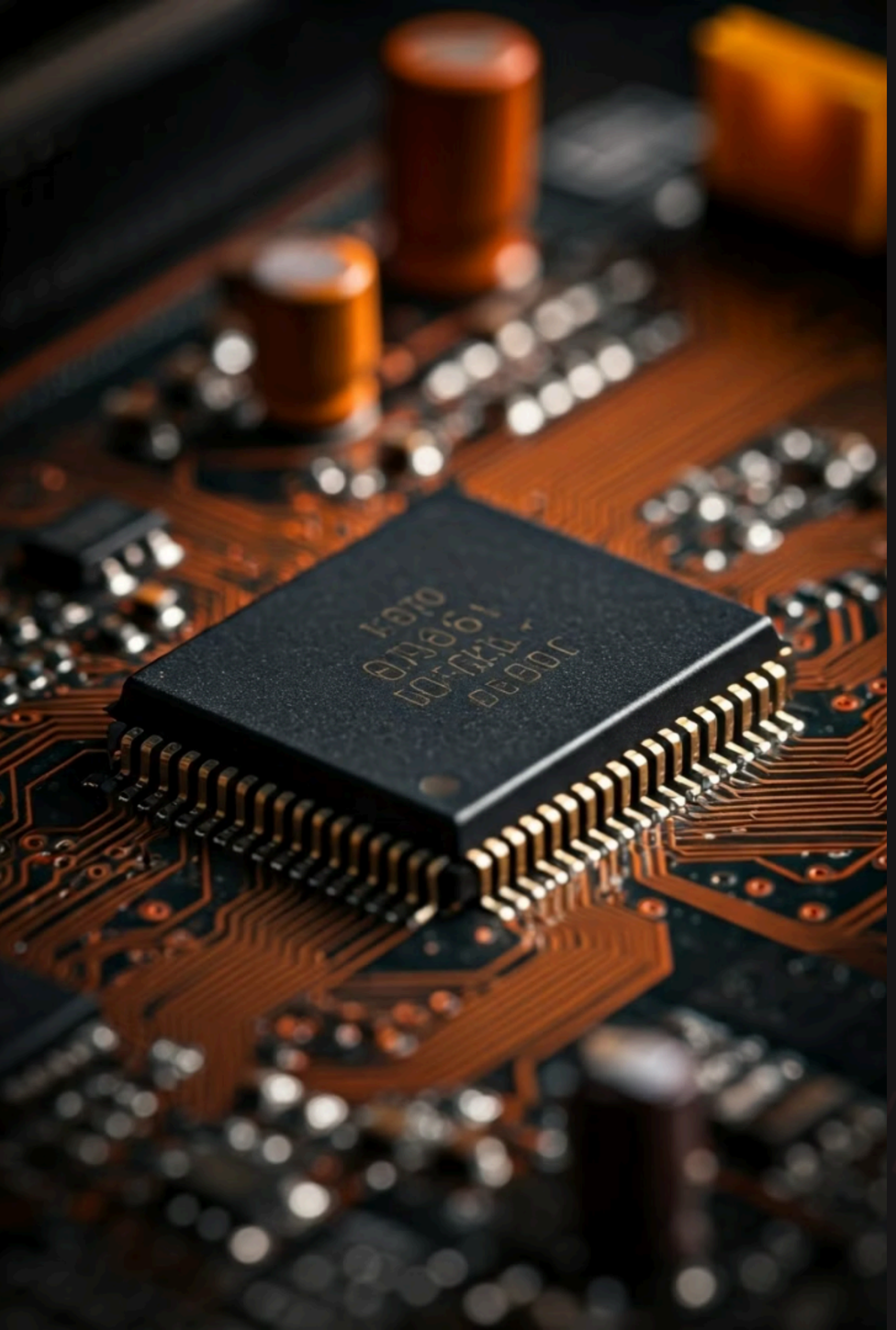
Project Overview: Tic Tac Toe in Assembly

This project builds a functional Tic Tac Toe game using Assembly language for the 8086 CPU.

It explores memory management, game logic, and input/output at the hardware level.

by : TAHA SHAH - 243493
by : MUZZAMIL - 243459





System Architecture and Memory Layout

Registers

AX, BX, CX, DX, SP, BP, SI, DI, IP, and Flags control operations.

Memory Segments

Code, data, and stack segments organize program memory efficiently.

Video Memory

Direct access at B800h allows screen output in text mode.

Game Board

Stored as a 3x3 array occupying 9 bytes in memory.

Core Game Logic Implementation

Board Initialization

Set all nine cells to empty before the game starts.

Player Input

Capture keystrokes using interrupt 16h keyboard input.

Win Check

Evaluate rows, columns, and diagonals after each move.

Move Validation

Ensure selected cell is empty before placing a mark.

Input/Output Handling

Keyboard Input

Read player's moves through BIOS interrupt 16h services.

Screen Output

Write characters directly to video memory for fast rendering.

Display Board

Render the current state of the board after every move.

Messages

Show prompts, errors, and game outcomes clearly.

Assembly Language Routines and

DrawBoard

Outputs game board layout and current cell symbols.

GetPlayerMove

Reads and validates player’s keyboard input securely.

CheckWin

Determines if a player has completed a winning line.

UpdateBoard

Writes the player’s mark into the board in memory.

Code Structure and Organization

Modular Design

Break code into reusable, smaller procedures for clarity.

Comments

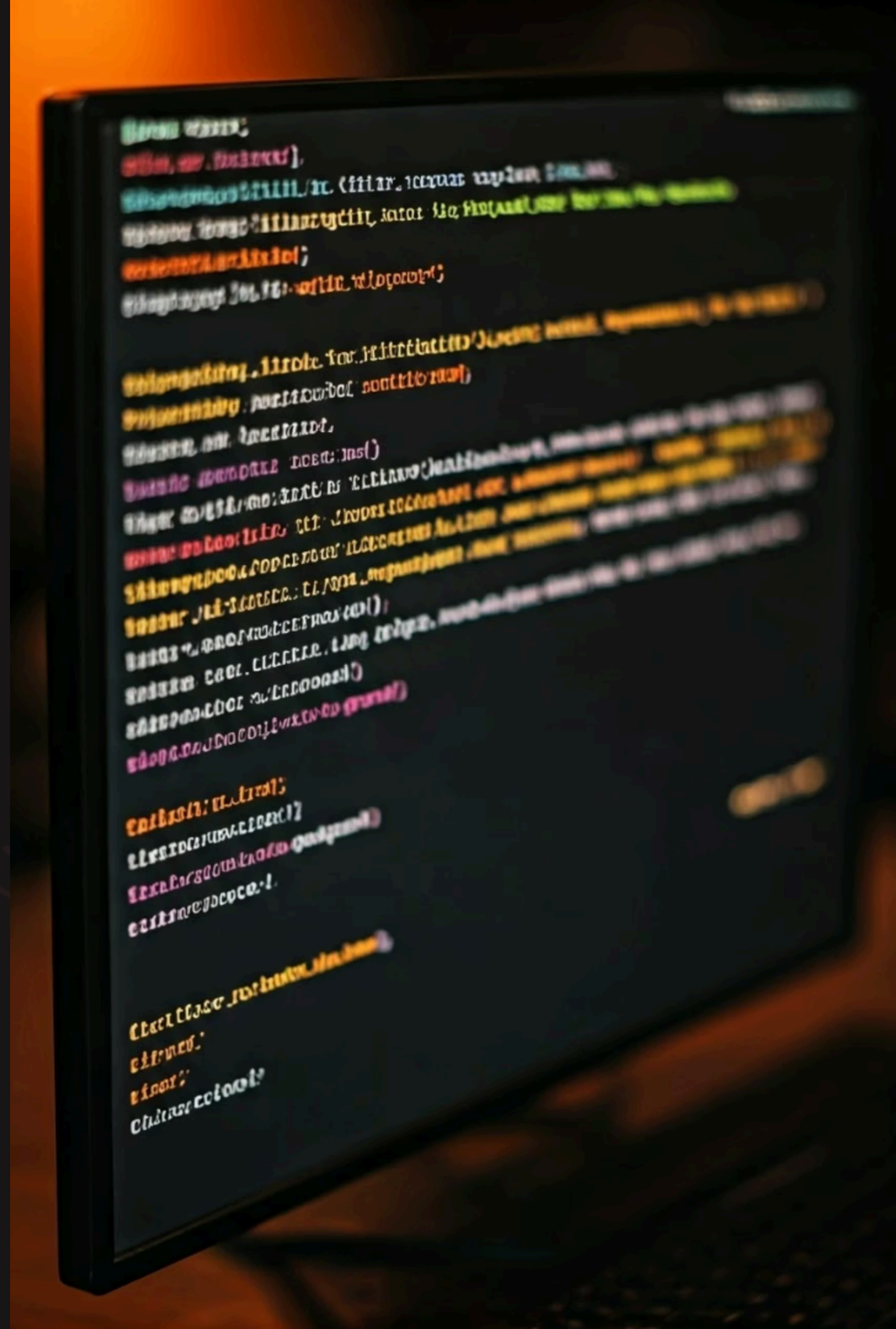
Extensive comments improve readability and maintenance.

Macros

Reuse code patterns efficiently for repetitive tasks.

Include Files

Split code into multiple files for manageable development.



Challenges and Optimizations

Memory Management

Use limited memory efficiently and avoid overflow issues.

Performance

Optimize execution speed for smooth gameplay experience.

Error Handling

Gracefully manage invalid inputs without crashing.

Code Size

Minimize executable size to fit within hardware constraints.