

**COMPUTER ORGANIZATION AND ASSEMBLY LANGUAGE**

**PROJECT REPORT**

Title: Tic Tac Toe game

Semester: Fall-2021

Class and Section: BSCS 3A

**Group Members:**

**Nimra Mushtaq 02-134202-018**

**Tayyaba Imam 02-134202-056**

**Shahzadi Zainub 02-134202-051**

**Assembly Language:**

For any processor Assembly Language is the most basic programming language. With assembly language, a programmer works only with operations implemented directly on the physical CPU.

**Project Description:**

Tic-Tac-Toe is a game which we have all played at some point in our life and its sheer simplicity contributes to its widespread adoption, making it one of the most popular pen and paper games.

Tic-Tac-Toe is a simple two-player game played in turns where each player places their mark on a 3x3 grid with the goal of having three consecutive marks in either the horizontal, vertical, or diagonal direction. The game can end in three possible states of either winning, losing, or drawing.

The goal is to construct a simple game similar to tic tac toe that can be played an endless number of times using the emulator8086. It gives a valuable insight into the operation of the 8086 CPU. It helps to reinforce the concepts of the 8086-programming language.

When compared to the conventional technique of playing Tic-Tac-Toe with paper and pencil, the computer application offers numerous advantages.

The different features are as follows:

• The game has been made more user-friendly with the use of LabVIEW software.

• The user may play as many games as they like without being interrupted.

• The user can select whatever symbol he or she want.

A player marks any of the 3x3 squares with his symbol (may be “X” or “O”) and his aim is to create a straight line horizontally or vertically or diagonally with two intensions: a) Create a straight line before his opponent to win the game. b) Restrict his opponent from creating a straight line first. In case logically no one can create a straight line with his own symbol, the game results a tie. Hence there are only three possible results – a player wins, his opponent (human or computer) wins or it’s a tie.

Implementations:

• You need an 8086 emulator to run this game.

• It is a 2-player game. Player 1 will place 'X' mark and Player 2 will place 'O' mark.

• Has option to repeat the game after it is over.

• Character inputs will be checked, if you enter any invalid character, it will prompt to input again.

• If you try to set mark to a cell already marked, it will prompt to input again.

In our game we have used the following commands/statements:

• JMP

• JE

• JNE

• INC

• DEC

• ADD

• CMP

• LEA

• CALL

• MOV

**Source Code:**

; TIC - TAC - TOE

; DEVLOPED BY

;Nimra Mushtaq

;Tayyaba Imam

;Shahzadi Zainub

;LANGUAGE USED: ASSEMBLY (8086)

.MODEL SMALL

.STACK 500H

.DATA

; LINES T1, T2, T3 AND T4 ARE USED TO BUILD THE TIC - TAC - TOE LOGO

T1 DB 4, 4, 4, 4, 4, 32, 4, 32, 4, 4, 4, 4, 32, 32, 32, 4, 4, 4, 4, 4, 32, 32, 4, 4, 32, 32, 4, 4, 4, 4, 32, 32, 32, 4, 4, 4, 4, 4, 32, 32, 4, 4, 32, 32, 4, 4, 4, 4, '$'

T2 DB 32, 32, 4, 32, 32, 32, 4, 32, 4, 32, 32, 32, 32, 32, 32, 32, 32, 4, 32, 32, 32, 4, 32, 32, 4, 32, 4, 32, 32, 32, 32, 32, 32 , 32, 32, 4, 32, 32, 32, 4, 32, 32, 4, 32, 4,'$'

T3 DB 32, 32, 4, 32, 32, 32, 4, 32, 4, 32, 32, 32, 32, 32, 32, 32, 32, 4, 32, 32, 32, 4, 4, 4, 4, 32, 4, 32, 32, 32, 32, 32, 32 , 32, 32, 4, 32, 32, 32, 4, 32, 32, 4, 32, 4, 4, 4, 4,'$'

T4 DB 32, 32, 4, 32, 32, 32, 4, 32, 4, 4, 4, 4, 32, 32, 32, 32, 32, 4, 32, 32, 32, 4, 32, 32, 4, 32, 4, 4, 4, 4, 32, 32, 32 , 32, 32, 4, 32, 32, 32, 32, 4, 4, 32, 32, 4, 4, 4, 4,'$'

;--------------------------------------------------------------------

; DEVELOPER'S NAMES

TAGLINE DB 'BSCS-3A:',10,13,9,9,9,3,'Nimra Mushtaq',10,13,,9,9,9,3,'Shahzadi Zainub',10,13,9,9,9,3,'Tayyaba Imam$'

; ----------------- IMPORTANT STRINGS USED THROUGHOUT THE GAME -----------------

PAK DB 'Press any key to continue...$'

; GAME RULES

R DB 'Game Rules:$'

R1 DB '1. Players will take turns.$'

R2 DB '2. Player 1 will start the game.$'

R3 DB '3. Player 1 will set "X" and Player 2 will set "O".$'

R4 DB '4. The board is marked with cell numbers.$'

R5 DB '5. Enter CELL NUMBER to place your mark.$'

R6 DB '6. Set 3 of your marks horizontally, vertically or diagonally to win.$'

R7 DB 'Good Luck!$'

; CELL MARK FOR PLAYERS

PC1 DB ' (X)$'

PC2 DB ' (O)$'

; BOARD LINES -------

L1 DB ' | | $'

L2 DB '-----------$'

N1 DB ' | $'

; -------------------

; CELL NUMBERS ------

C1 DB '1$'

C2 DB '2$'

C3 DB '3$'

C4 DB '4$'

C5 DB '5$'

C6 DB '6$'

C7 DB '7$'

C8 DB '8$'

C9 DB '9$'

; -------------------

; PLAYER NO. , MOVES AND CHECK FLAGS FOR IF THE GAME IS WON OR DRAWN

PLAYER DB 50, '$'

MOVES DB 0

DONE DB 0

DR DB 0

; INPUT SECTION PROMTS -------------------------

INP DB 32, ':: Enter cell no. : $'

TKN DB 'This cell is taken! Press any key...$'

; CURRENT MARK (X/O) ---------------------------

CUR DB 88

; FINAL MESSAGES -------------------------------

W1 DB 'Player $'

W2 DB ' won the game!$'

DRW DB 'The game is draw!$'

; TRY AGAIN PROMPT MESSAGES -----------------------------

TRA DB 'Want to play again? (y/n): $'

WI DB 32, 32, 32, 'Wrong input! Press any key... $'

; THIS LINE IS USED TO OVERWIRTE A LINE TO CLEAN THE AREA

EMP DB ' $'

;--------------------------------------------------------

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

; --------- DISPLAY THE TITLE SCREEN ---------

TITLESCREEN:

; LOGO DISPLAY START -----------------

; SET CURSOR

MOV AH, 2

MOV BH, 0

MOV DH, 6

MOV DL, 14

INT 10H

LEA DX, T1

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 7

MOV DL, 14

INT 10H

LEA DX, T2

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 8

MOV DL, 14

INT 10H

LEA DX, T3

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 9

MOV DL, 14

INT 10H

LEA DX, T2

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 10

MOV DL, 14

INT 10H

LEA DX, T4

MOV AH, 9

INT 21H

; LOGO DISPLAY END -----------------

; SET CURSOR

MOV AH, 2

MOV DH, 12

MOV DL, 22

INT 10H

LEA DX, TAGLINE ; DEVELOPER NAME DISPLAY

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 17

MOV DL, 7

INT 10H

LEA DX, PAK ; PRESS ANY KEY

MOV AH, 9

INT 21H

MOV AH, 7 ; INPUT WITHOUT ECHO

INT 21H

; CLEAR SCREEN

MOV AX,0600H

MOV BH,07H

MOV CX,0000H

MOV DX,184FH

INT 10H

JMP RULES

; ----------- DISPLAY GAME RULES --------------

RULES:

; SET CURSOR

MOV AH, 2

MOV BH, 0

MOV DH, 6

MOV DL, 7

INT 10H

LEA DX, R

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 7

MOV DL, 7

INT 10H

LEA DX, R1 ; RULE 1

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 8

MOV DL, 7

INT 10H

LEA DX, R2 ; RULE 2

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 9

MOV DL, 7

INT 10H

LEA DX, R3 ; RULE 3

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 10

MOV DL, 7

INT 10H

LEA DX, R4 ; RULE 4

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 11

MOV DL, 7

INT 10H

LEA DX, R5 ; RULE 5

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 12

MOV DL, 7

INT 10H

LEA DX, R6

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 13

MOV DL, 7

INT 10H

LEA DX, R7

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 15

MOV DL, 7

INT 10H

LEA DX, PAK ; PRESS ANY KEY

MOV AH, 9

INT 21H

MOV AH, 7 ; INPUT WITHOUT ECHO

INT 21H

; ---------- DISPLAY GAME RULES END ---------

; ---------- INITIALIZE ---------------------

INIT:

MOV PLAYER, 50 ; INITIALIZING ALL VARIABLES

MOV MOVES, 0

MOV DONE, 0

MOV DR, 0

MOV C1, 49

MOV C2, 50

MOV C3, 51

MOV C4, 52

MOV C5, 53

MOV C6, 54

MOV C7, 55

MOV C8, 56

MOV C9, 57

JMP PLRCHANGE

; ---------- INITIALIZATION ENDS --------------

; ------------ VICTORY ------------------------

VICTORY:

LEA DX, W1

MOV AH, 9

INT 21H

LEA DX, PLAYER

MOV AH, 9

INT 21H

LEA DX, W2

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 17

MOV DL, 28

INT 10H

LEA DX, PAK ; PRESS ANY KEY

MOV AH, 9

INT 21H

MOV AH, 7 ; INPUT WITHOUT ECHO

INT 21H

JMP TRYAGAIN

; ------------ DRAW ------------

DRAW:

LEA DX, DRW

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 17

MOV DL, 28

INT 10H

LEA DX, PAK ; PRESS ANY KEY

MOV AH, 9

INT 21H

MOV AH, 7 ; INPUT WITHOUT ECHO

INT 21H

JMP TRYAGAIN

; ------------ CHECK IF WINNING CONDITION IS MET -----------

CHECK: ; THERE ARE 8 POSSIBLE WINNING COMBINATIONS

CHECK1: ; CHECKING 1, 2, 3

MOV AL, C1

MOV BL, C2

MOV CL, C3

CMP AL, BL

JNZ CHECK2

CMP BL, CL

JNZ CHECK2

MOV DONE, 1

JMP BOARD

CHECK2: ; CHECKING 4, 5, 6

MOV AL, C4

MOV BL, C5

MOV CL, C6

CMP AL, BL

JNZ CHECK3

CMP BL, CL

JNZ CHECK3

MOV DONE, 1

JMP BOARD

CHECK3: ; CHECKING 7, 8, 9

MOV AL, C7

MOV BL, C8

MOV CL, C9

CMP AL, BL

JNZ CHECK4

CMP BL, CL

JNZ CHECK4

MOV DONE, 1

JMP BOARD

CHECK4: ; CHECKING 1, 4, 7

MOV AL, C1

MOV BL, C4

MOV CL, C7

CMP AL, BL

JNZ CHECK5

CMP BL, CL

JNZ CHECK5

MOV DONE, 1

JMP BOARD

CHECK5: ; CHECKING 2, 5, 8

MOV AL, C2

MOV BL, C5

MOV CL, C8

CMP AL, BL

JNZ CHECK6

CMP BL, CL

JNZ CHECK6

MOV DONE, 1

JMP BOARD

CHECK6: ; CHECKING 3, 6, 9

MOV AL, C3

MOV BL, C6

MOV CL, C9

CMP AL, BL

JNZ CHECK7

CMP BL, CL

JNZ CHECK7

MOV DONE, 1

JMP BOARD

CHECK7: ; CHECKING 1, 5, 9

MOV AL, C1

MOV BL, C5

MOV CL, C9

CMP AL, BL

JNZ CHECK8

CMP BL, CL

JNZ CHECK8

MOV DONE, 1

JMP BOARD

CHECK8: ; CHECKING 3, 5, 7

MOV AL, C3

MOV BL, C5

MOV CL, C7

CMP AL, BL

JNZ DRAWCHECK

CMP BL, CL

JNZ DRAWCHECK

MOV DONE, 1

JMP BOARD

DRAWCHECK:

MOV AL, MOVES

CMP AL, 9

JB PLRCHANGE

MOV DR, 1

JMP BOARD

JMP EXIT

; ------------ PLAYER ----------

PLRCHANGE:

CMP PLAYER, 49

JZ P2

CMP PLAYER, 50

JZ P1

P1:

MOV PLAYER, 49

MOV CUR, 88

JMP BOARD

P2:

MOV PLAYER, 50

MOV CUR, 79

JMP BOARD

; ------------- BOARD ----------

BOARD:

; CLEAR SCREEN

MOV AX,0600H

MOV BH,07H

MOV CX,0000H

MOV DX,184FH

INT 10H

; SET CURSOR

MOV AH, 2

MOV BH, 0

MOV DH, 6

MOV DL, 30

INT 10H

LEA DX, L1

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 7

MOV DL, 30

INT 10H

MOV AH, 2

MOV DL, 32

INT 21H

; --------------------------------

; CELL 1

LEA DX, C1

MOV AH, 9

INT 21H

LEA DX, N1

MOV AH, 9

INT 21H

; CELL 2

LEA DX, C2

MOV AH, 9

INT 21H

LEA DX, N1

MOV AH, 9

INT 21H

; CELL 3

LEA DX, C3

MOV AH, 9

INT 21H

; ---------------------------------

; SET CURSOR

MOV AH, 2

MOV DH, 8

MOV DL, 30

INT 10H

LEA DX, L2

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 9

MOV DL, 30

INT 10H

LEA DX, L1

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 10

MOV DL, 30

INT 10H

MOV AH, 2

MOV DL, 32

INT 21H

; --------------------------------

; CELL 4

LEA DX, C4

MOV AH, 9

INT 21H

LEA DX, N1

MOV AH, 9

INT 21H

; CELL 5

LEA DX, C5

MOV AH, 9

INT 21H

LEA DX, N1

MOV AH, 9

INT 21H

; CELL 6

LEA DX, C6

MOV AH, 9

INT 21H

; ---------------------------------

; SET CURSOR

MOV AH, 2

MOV DH, 11

MOV DL, 30

INT 10H

LEA DX, L1

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 12

MOV DL, 30

INT 10H

LEA DX, L2

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 13

MOV DL, 30

INT 10H

LEA DX, L1

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 14

MOV DL, 30

INT 10H

MOV AH, 2

MOV DL, 32

INT 21H

; --------------------------------

; CELL 7

LEA DX, C7

MOV AH, 9

INT 21H

LEA DX, N1

MOV AH, 9

INT 21H

; CELL 8

LEA DX, C8

MOV AH, 9

INT 21H

LEA DX, N1

MOV AH, 9

INT 21H

; CELL 9

LEA DX, C9

MOV AH, 9

INT 21H

; ---------------------------------

; SET CURSOR

MOV AH, 2

MOV DH, 15

MOV DL, 30

INT 10H

LEA DX, L1

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 16

MOV DL, 20

INT 10H

CMP DONE, 1

JZ VICTORY

CMP DR, 1

JZ DRAW

; ------------ END OF BOARD -------

; ------------ INPUT --------------

INPUT:

LEA DX, W1

MOV AH, 9

INT 21H

MOV AH, 2

MOV DL, PLAYER

INT 21H

CMP PLAYER, 49

JZ PL1

LEA DX, PC2

MOV AH, 9

INT 21H

JMP TAKEINPUT

PL1:

LEA DX, PC1

MOV AH, 9

INT 21H

TAKEINPUT:

LEA DX, INP

MOV AH, 9

INT 21H

MOV AH, 1

INT 21H

INC MOVES ; INCREMENTING MOVES COUNTER BY 1

MOV BL, AL

SUB BL, 48

MOV CL, CUR

; CHECKING IF INPUT IS BETWEEN 1-9

CMP BL, 1

JZ C1U

CMP BL, 2

JZ C2U

CMP BL, 3

JZ C3U

CMP BL, 4

JZ C4U

CMP BL, 5

JZ C5U

CMP BL, 6

JZ C6U

CMP BL, 7

JZ C7U

CMP BL, 8

JZ C8U

CMP BL, 9

JZ C9U

;---------------------------------

; IF INPUT IS INVALID

DEC MOVES ; DECREMENTING MOVES BY 1, SINCE IT WAS INVALID

; SET CURSOR

MOV AH, 2

MOV DH, 16

MOV DL, 20

INT 10H

LEA DX, WI ; WRONG INPUT MESSAGE

MOV AH, 9

INT 21H

MOV AH, 7 ; INPUT WITHOUT ECHO

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 16

MOV DL, 20

INT 10H

LEA DX, EMP ; CLEARING THAT LINE

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 16

MOV DL, 20

INT 10H

JMP INPUT

TAKEN:

DEC MOVES

; SET CURSOR

MOV AH, 2

MOV DH, 16

MOV DL, 20

INT 10H

LEA DX, TKN ; DISPLAY THAT THE CELL IS TAKEN

MOV AH, 9

INT 21H

MOV AH, 7 ; INPUT WITHOUT ECHO

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 16

MOV DL, 20

INT 10H

LEA DX, EMP ; EMPTY LINE TO OVERWRITE ANOTHER LINE TO CLEAN THE SPACE

MOV AH, 9

INT 21H

; SET CURSOR

MOV AH, 2

MOV DH, 16

MOV DL, 20

INT 10H

JMP INPUT

; ADJUST

; SETTING BOARD POSITION AS INPUT MARK

C1U:

CMP C1, 88 ; CHECKING IF THE CELL IS ALREADY 'X'

JZ TAKEN

CMP C1, 79 ; CHECKING IF THE CELL IS ALREADY 'O'

JZ TAKEN

MOV C1, CL

JMP CHECK

C2U:

CMP C2, 88 ; CHECKING IF THE CELL IS ALREADY 'X'

JZ TAKEN

CMP C2, 79 ; CHECKING IF THE CELL IS ALREADY 'O'

JZ TAKEN

MOV C2, CL

JMP CHECK

C3U:

CMP C3, 88 ; CHECKING IF THE CELL IS ALREADY 'X'

JZ TAKEN

CMP C3, 79 ; CHECKING IF THE CELL IS ALREADY 'O'

JZ TAKEN

MOV C3, CL

JMP CHECK

C4U:

CMP C4, 88 ; CHECKING IF THE CELL IS ALREADY 'X'

JZ TAKEN

CMP C4, 79 ; CHECKING IF THE CELL IS ALREADY 'O'

JZ TAKEN

MOV C4, CL

JMP CHECK

C5U:

CMP C5, 88 ; CHECKING IF THE CELL IS ALREADY 'X'

JZ TAKEN

CMP C5, 79 ; CHECKING IF THE CELL IS ALREADY 'O'

JZ TAKEN

MOV C5, CL

JMP CHECK

C6U:

CMP C6, 88 ; CHECKING IF THE CELL IS ALREADY 'X'

JZ TAKEN

CMP C6, 79 ; CHECKING IF THE CELL IS ALREADY 'O'

JZ TAKEN

MOV C6, CL

JMP CHECK

C7U:

CMP C7, 88 ; CHECKING IF THE CELL IS ALREADY 'X'

JZ TAKEN

CMP C7, 79 ; CHECKING IF THE CELL IS ALREADY 'O'

JZ TAKEN

MOV C7, CL

JMP CHECK

C8U:

CMP C8, 88 ; CHECKING IF THE CELL IS ALREADY 'X'

JZ TAKEN

CMP C8, 79 ; CHECKING IF THE CELL IS ALREADY 'O'

JZ TAKEN

MOV C8, CL

JMP CHECK

C9U:

CMP C9, 88 ; CHECKING IF THE CELL IS ALREADY 'X'

JZ TAKEN

CMP C9, 79 ; CHECKING IF THE CELL IS ALREADY 'O'

JZ TAKEN

MOV C9, CL

JMP CHECK

; --------------------------------

; ----------- TRY AGAIN -----------

TRYAGAIN:

; CLEAR SCREEN

MOV AX,0600H

MOV BH,07H

MOV CX,0000H

MOV DX,184FH

INT 10H

; SET CURSOR

MOV AH, 2

MOV BH, 0

MOV DH, 10

MOV DL, 24

INT 10H

LEA DX, TRA ; TRY AGAIN PROMPT

MOV AH, 9

INT 21H

MOV AH, 1

INT 21H

CMP AL, 121 ; CHECK IF INPUT IS 'y'

JZ INIT

CMP AL, 89 ; CHECK IF INPUT IS 'Y'

JZ INIT

; IF INPUT IS 'Y'/'y' THEN REPEAT THE GAME

CMP AL, 110 ; CHECK IF INPUT IS 'n'

JZ EXIT

CMP AL, 78 ; CHECK IF INPUT IS 'N'

JZ EXIT

; IF INPUT IS 'N'/'n' THEN EXIT THE GAME

; IF INPUT IS INVALID

; SET CURSOR

MOV AH, 2

MOV BH, 0

MOV DH, 10

MOV DL, 24

INT 10H

LEA DX, WI ; WRONG INPUT MESSAGE

MOV AH, 9

INT 21H

MOV AH, 7 ; INPUT WITHOUT ECHO

INT 21H

; SET CURSOR

MOV AH, 2

MOV BH, 0

MOV DH, 10

MOV DL, 24

INT 10H

LEA DX, EMP ; EMPTY LINE TO OVERWRITE ANOTHER LINE TO CLEAN THE SPACE

MOV AH, 9

INT 21H

JMP TRYAGAIN ; PROMPT THE TRY AGAIN

; ----------- END OF INPUT --------

EXIT:

MOV AH, 4CH

INT 21H

MAIN ENDP

END MAIN

**CONCLUSION:**

We are proud of how we have implemented the traditional game, TIC TAC TOE, it can be played by two players where the square block (3 x 3) can be filled with a cross (X) or a circle (O). The game will toggle between the players by giving the chance for each player to mark their move. When one of the players make a combination of 3 same markers in a horizontal, vertical, or diagonal line the program will display which player has won, whether X or O.

**OUTPUT:**

**A picture containing text

Description automatically generated**

**Text

Description automatically generated**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated with low confidence**

**Diagram

Description automatically generated with medium confidence**

**Graphical user interface, text

Description automatically generated**

**Diagram

Description automatically generated with medium confidence**