

**Department of Computer Science**

**CSCL2201 – Computer Organization and Assembly Language**

**Tic Tac Toe**

**Project Report**

**Semester: Fall 2023**

**Group members:**

1. **Syeda Fizza Rashid (2212163)**
2. **Muhammad Yaseen Ahmed (2212153)**
3. **Shahzaib Hussain (2212159)**

***Submitted to:***

**Sir Mubeen Ahmed Khan**

***Lab Engineer***

**Sir Waseem Rauf**

**Table of Content:**

* Objective
* Description
* Concepts Implemented
* Project Features
* Source Code
* Output
* Conclusion

**Objective:** The objective of this project is to implement a simple text-based Tic-Tac-Toe game using Assembly language (8086).

**Description:** The Tic-Tac-Toe game is a classic two-player game where each player takes turns marking a cell on a 3x3 grid. The player who succeeds in placing three of their marks (either "X" or "O") in a horizontal, vertical, or diagonal row wins the game. The implementation is in Assembly language (8086), providing a low-level view of game development.

**Concepts Implemented:**

**Assembly (8086):** The entire game is written in Assembly language, specifically targeting the 8086 architecture.

**Text-Based Interface:** The game features a text-based interface, utilizing the DOS interrupt calls for input/output and cursor control.

**Game Logic:** The program includes the logic for checking win conditions, handling player turns, and managing the game board.

**User Input Handling:** The code captures user input and validates it, ensuring it corresponds to a valid move in the game

.

**Graphics (ASCII Art):** ASCII art is used to create a simple Tic-Tac-Toe logo on the title screen.

**Project Features:**

**Title Screen:** The game starts with a title screen displaying a Tic-Tac-Toe logo and developer information.

**Game Rules:** A section explains the rules of the game to the players before starting.

**Board Display:** The program dynamically updates and displays the Tic-Tac-Toe board with each move.

**Player Interaction:** Players take turns entering their moves by selecting a cell number.

**Winning Condition**: The program checks for the winning condition after each move and declares the winner when applicable.

**Draw Condition:** If no player wins and the board is filled, the game declares a draw.

**Play Again Option**: After each game, players are prompted to play again, providing flexibility for multiple rounds.

**Project Features:**

**Console Interface:** The game utilizes the console for interaction, providing a simple and user friendly interface.

**Player vs. Player:** Two players can take turns to play the game, selecting positions on the 3x3 Tic Tac Toe grid.

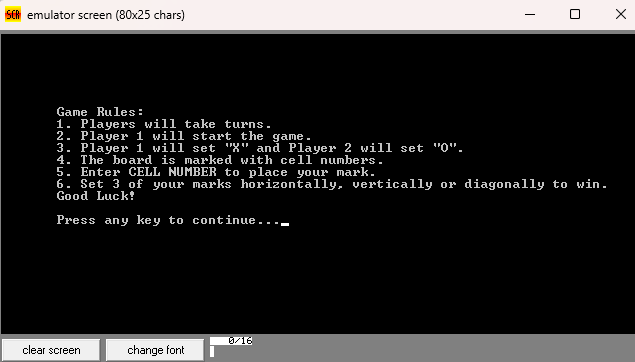
**Winning Conditions**: Detection of winning conditions, such as three symbols in a row, column, or diagonal.

**Source Code**

**Output:**

**Menu**

****

**Rules**

**Game Result**

****

**Conclusion:**

In conclusion, this Assembly (8086) implementation of Tic-Tac-Toe demonstrates the utilization of low-level programming to create a text-based interactive game. The project successfully integrates game logic, user input handling, and ASCII art to provide a classic gaming experience. Enhancements and optimizations can be explored for future iterations, but the current implementation serves as a foundation for understanding Assembly language programming and game development.

Source Code:

; TIC - TAC - TOE

; DEVLOPED BY FYS

; 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, 4, 32, 32, 32, 4, 32, 32, 32, 4, 32, 32, 4, 32, 4, 4, 4, 4, 32, 4, 32 , 32, 32, 4, 32, 32, 32, 32, 4, 4, 32, 32, 4, 4, 4, 4,'$'

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

TAGLINE DB 'Developed by FYS$' ; DEVELOPER NAME

; ----------------- 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, 13

MOV DL, 24

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, C4

MOV BL, C5

MOV CL, C6

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 4

LEA DX, C7

MOV AH, 9

INT 21H

LEA DX, N1

MOV AH, 9

INT 21H

; CELL 5

LEA DX, C8

MOV AH, 9

INT 21H

LEA DX, N1

MOV AH, 9

INT 21H

; CELL 6

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