# EC316: Microprocessor Project

# Final Report
# Tic-Tac-Toe against the 8085 Microprocessor

Shikhar Makhija: 155/EC/15
Varun Tripathi: 196/EC/15

## Synopsis

The primary objective of our project was to create a hardware implementation of the game 'Tic-Tac-Toe' with the facility of playing against the 8085 microprocessor. The microprocessor algorithmically analyses the state of the game and picks the most optimum move so as to ensure that it always wins or draws. To implement this we used the Minimax algorithm to determine the moves picked by the 8085. This project highlights the decision-making capability of microprocessors. Additionally, a separate mode for player-versus-player games is provided aswell.

**Keywords**: Minimax Algorithm, 8085 Microprocessor, Game Theory
**Date**: January 2018

## Acknowledgements

We would like to thank Professor Dhananjay V. Gadre for his guidance and support throughout this project. We would also like to thank the CEDT lab for providing us with facilities to fabricate our board and its members who provided us with valuable help throughout the fabrication process.

# Contents

# 1 Introduction

## 1.1 Motivation

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. The EC-316 course provides the perfect platform to give our beloved game a concrete form. The notion of computer intelligence is one that has always intrigued us and we aim to demonstrate with our project how even a microprocessors from the 70's can be programmed to be unbeatable.

## 1.2 Description

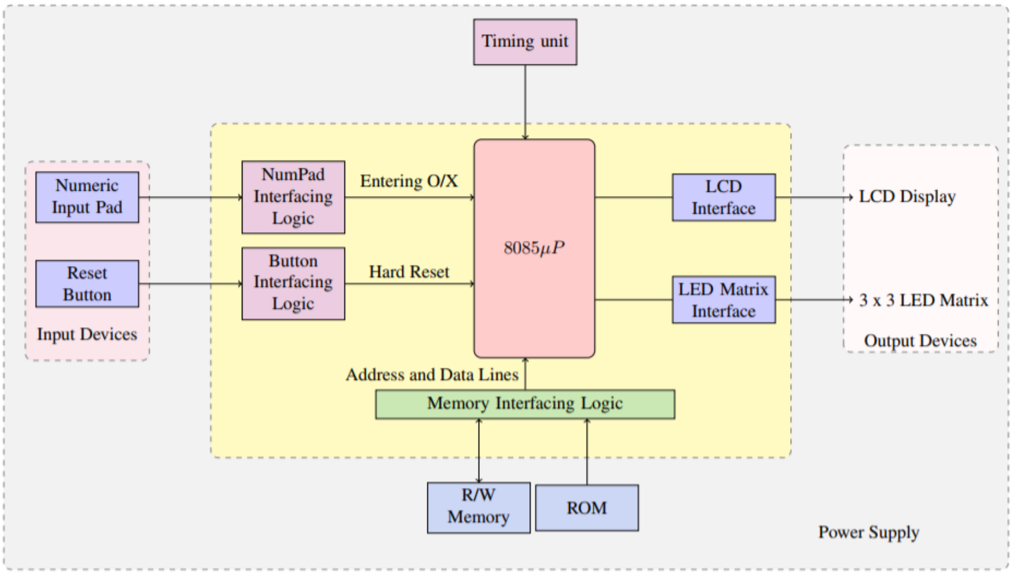


Figure 1: Block Diagram Representation of Proposed Model

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. We have constructed a hardware implementation of this popular game with the facility to play against either another player or the 8085 itself. We take user input using a re-purposed numeric keypad and instead of placing a mark, an LED of a specific colour is lit instead, representing either O or X. The state of the game has been displayed using a 3x3 grid of bi-colour LEDs and a 16x2 character LCD is present to provide textual information about the game status. The block diagram representation of the proposed model is shown in Figure 1.

## 1.3 Flow Chart of Operation

The detailed flowchart for the user-interface is given in Figure 2.

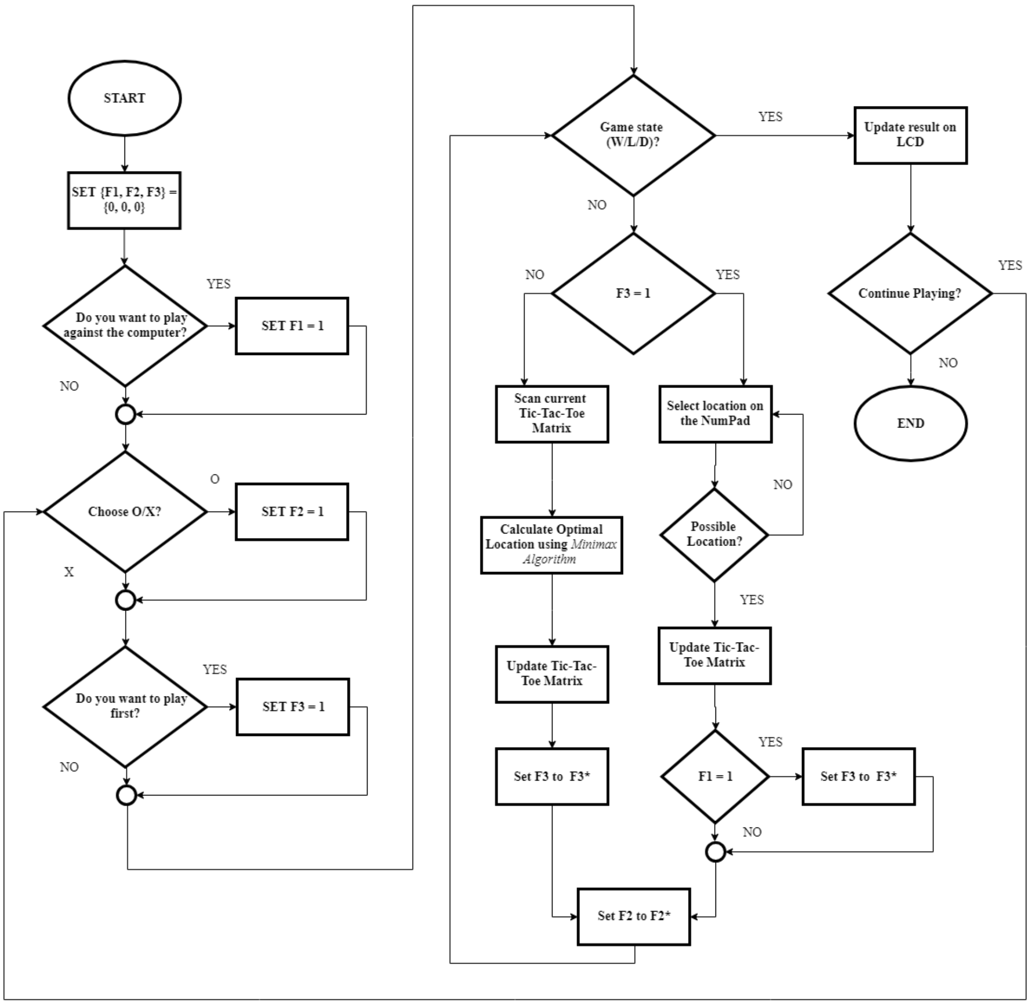


Figure 2: Flow Chart

# 2 Details of Hardware Implementation

## 2.1 Circuit Description

This project is based on the 8085 microprocessor. For interfacing with the user there are three interfacing units, namely, a Keypad, a Liquid Crystal Display and a 3x3 LED Matrix. While on the internal level, there are two Programmable Interface Units(8255), 3 UDN2918a driver and Decoding Logic. The decoding logic comprises of a NOT gate (7404) and a OR Gate(7432)IC. The 8085 with interfacing logic to the ROM and RAM is shown below.

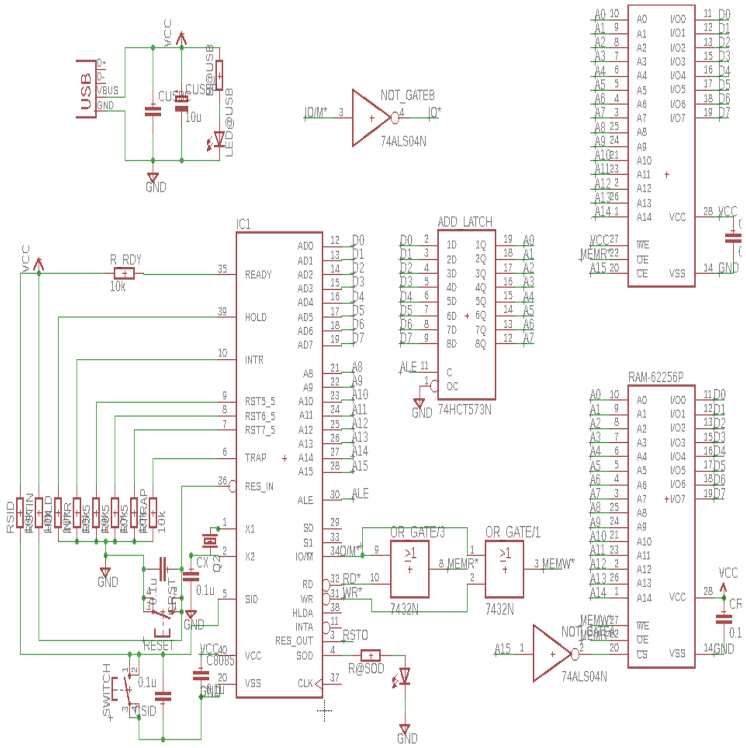

Figure 3: 8085 and Interfacing Logic

The first PPI is connected to three UDN2918A's. This PPI operates in Mode 0, with all the three ports acting as output port. The related circuitry is shown below.

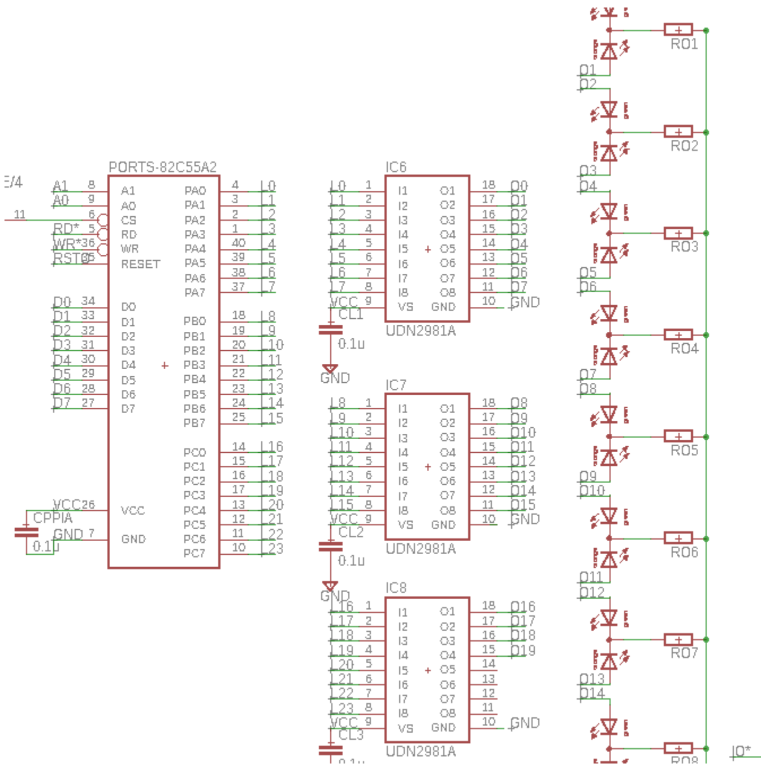

Figure 4: The First PPI

The second programmable peripheral interface is used for interfacing the LCD and the keyboard matrix. Port A and B were used for interfacing with the LCD, while the Port C was used to take data from the keyboard matrix. We pulse here each of the output lines (Active low logic pulse) one at a time and check input lines for any response. The input nibble is then compared by the software to various combinations to check for a keypress. Depending on which pulse and the input nibble combination the row and column is found and thus the exact key is uniquely identified.

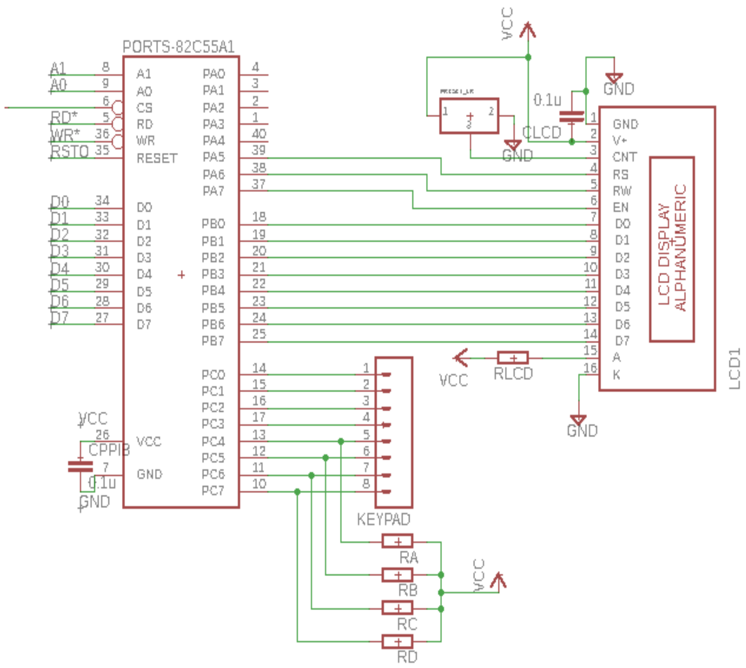


Figure 5: The Second PPI

## 2.2 Comprehensive Schematic

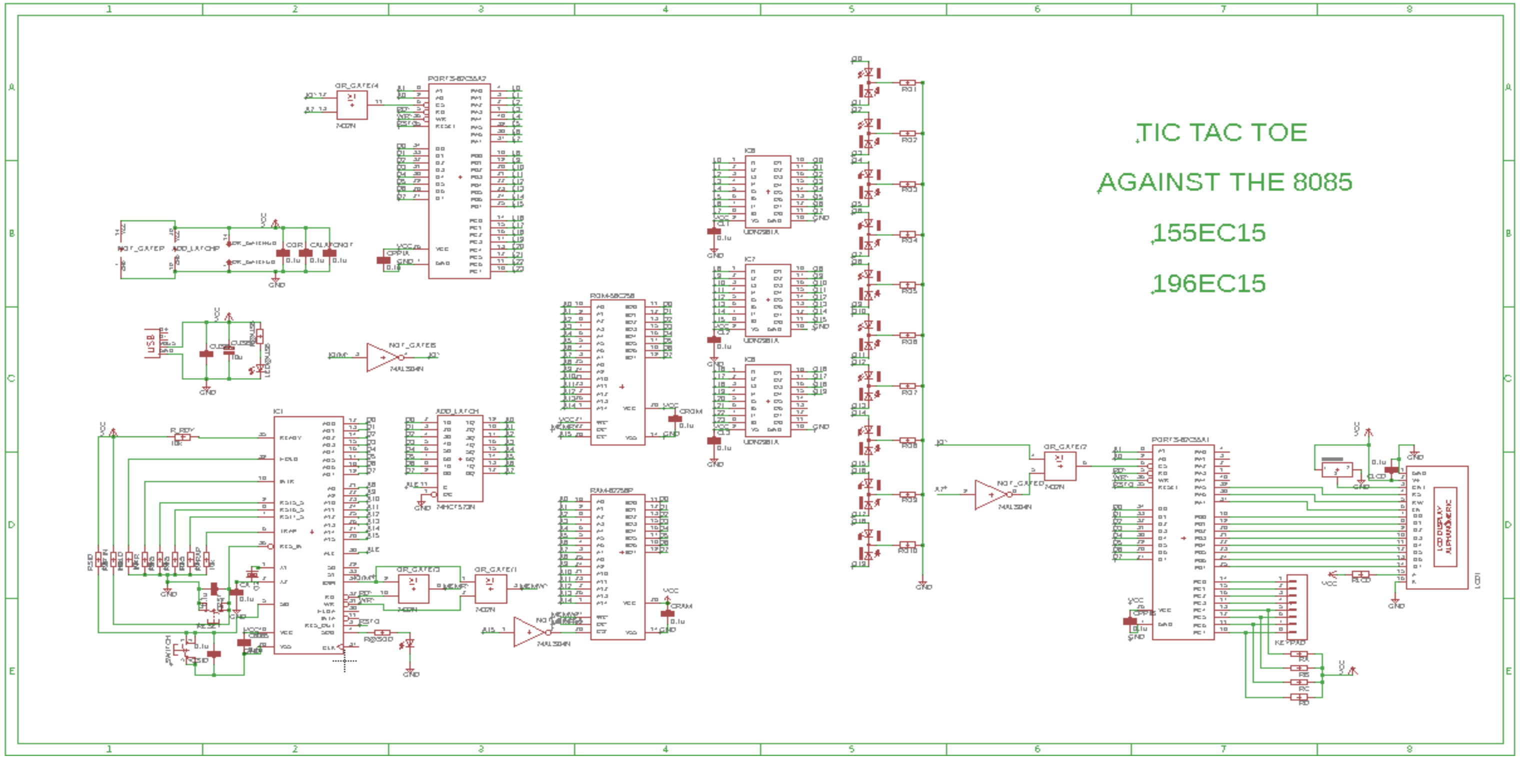


Figure 6: Comprehensive Schematic

## 2.3 Board Layout

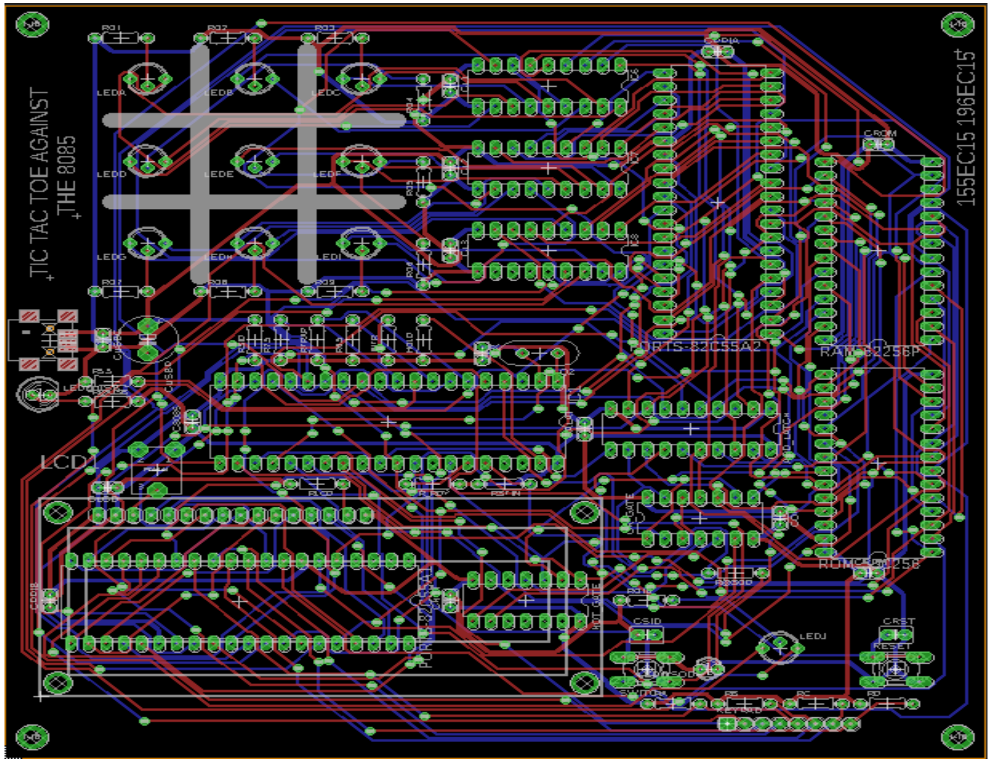


Figure 7: Board Layout

## 2.4 Printed Circuit Board

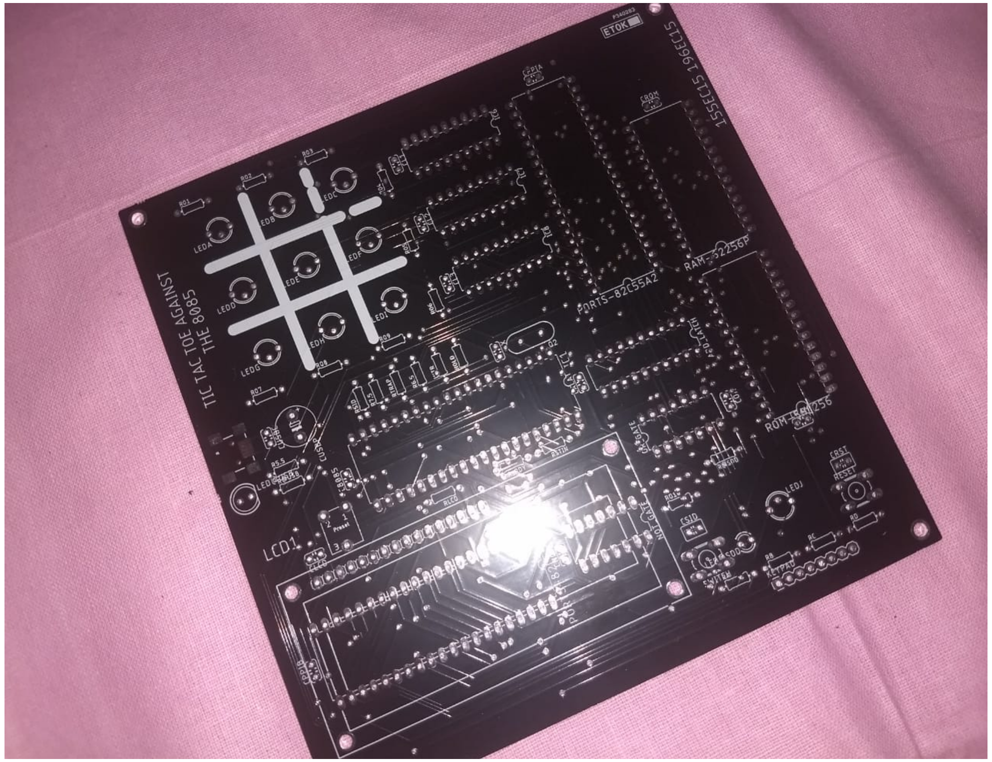


Figure 8: Printed Circuit Board

## 2.5 Fabricated Board

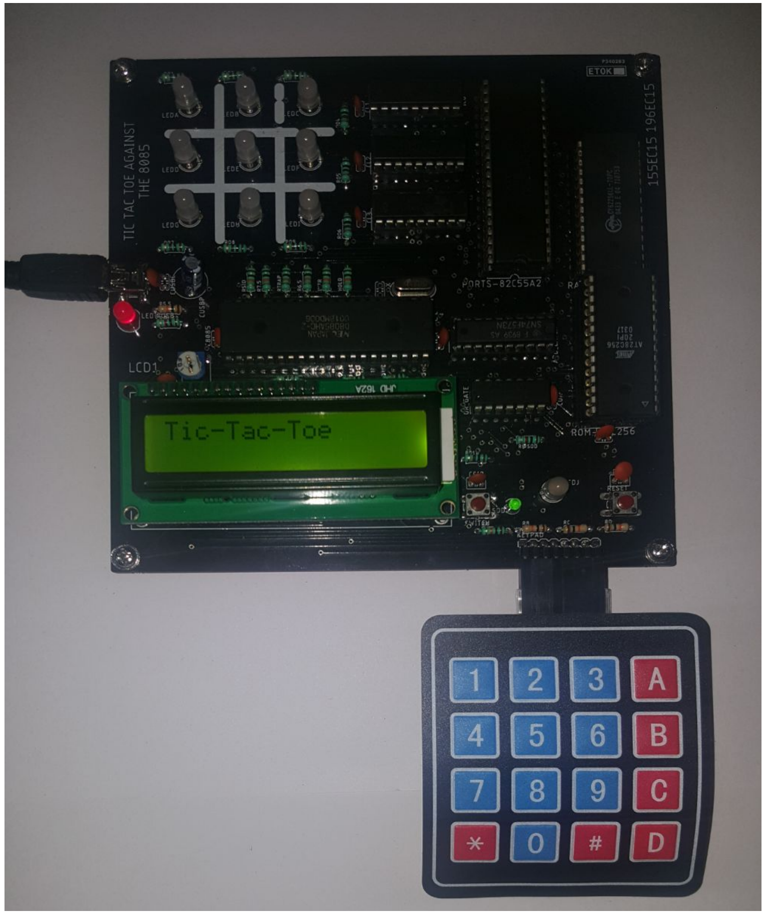


Figure 9: Fabricated Board

## 2.6 Bill of Materials

| Part | Value | Description |
|---|---|---|
| ADD$_{L}ATCH$ | 74HCT573N | 8-bit D latch BUS DRIVER |
| C8085 | 0.1u | CAPACITOR |
| CALAT | 0.1u | CAPACITOR |
| CL1 | 0.1u | CAPACITOR |
| CL2 | 0.1u | CAPACITOR |
| CL3 | 0.1u | CAPACITOR |
| CLCD | 0.1u | CAPACITOR |
| CNOT | 0.1u | CAPACITOR |
| COR | 0.1u | CAPACITOR |
| CPPIA | 0.1u | CAPACITOR |
| CPPIB | 0.1u | CAPACITOR |
| CRAM | 0.1u | CAPACITOR |
| CROM | 0.1u | CAPACITOR |
| CRST | 0.1u | C2.5-3 |
| CSID | 0.1u | C2.5-3 |
| CUSBC | | CAPACITOR |
| CUSBP | 10u | 5 |
| CX | 0.1u | CAPACITOR |
| HOLD | 10k | RESISTOR |
| IC1 | | MICROCOMPUTER/PERIPHERAL DEVICE |
| IC6 | UDN2981A | DRIVER ARRAY |
| IC7 | UDN2981A | DRIVER ARRAY |
| IC8 | UDN2981A | DRIVER ARRAY |
| INTR | 10k | RESISTOR |
| KEYPAD | | Header 8 |
| LCD1 | | ALPHANUMERIC-LCD |
| LED@SOD | | LED |
| LED@USB | LED5MM | |
| LEDA | TLUV5300 | Bicolor LED 5 mm Untinted Diffused Package |
| LEDB | TLUV5300 | Bicolor LED 5 mm Untinted Diffused Package |
| LEDC | TLUV5300 | Bicolor LED 5 mm Untinted Diffused Package |
| LEDD | TLUV5300 | Bicolor LED 5 mm Untinted Diffused Package |
| LEDE | TLUV5300 | Bicolor LED 5 mm Untinted Diffused Package |
| LEDF | TLUV5300 | Bicolor LED 5 mm Untinted Diffused Package |
| LEDG | TLUV5300 | Bicolor LED 5 mm Untinted Diffused Package |
| LEDH | TLUV5300 | Bicolor LED 5 mm Untinted Diffused Package |
| LEDI | TLUV5300 | Bicolor LED 5 mm Untinted Diffused Package |
| LEDJ | TLUV5300 | Bicolor LED 5 mm Untinted Diffused Package |

| | | |
|---|---|---|
| NOT$_{G}ATE$ | 74ALS04N | Hex INVERTER |
| OR$_{G}ATE$ | 7432N | Quad 2-input OR gate |
| PORTS-82C55A1 | | MICROCOMPUTER/PERIPHERAL DEVICE |
| PORTS-82C55A2 | | MICROCOMPUTER/PERIPHERAL DEVICE |
| Q2 | | Crystals |
| R5.5 | 10k | RESISTOR |
| R6.5 | 10k | RESISTOR |
| R7.5 | 10k | RESISTOR |
| R@SOD | | RESISTOR |
| R@USB | | RESISTOR |
| RA | | RESISTOR |
| RAM-62256P | | MEMORY |
| RB | | RESISTOR |
| RC | | RESISTOR |
| RD | | RESISTOR |
| RESET | | OMRON SWITCH |
| RLCD | | RESISTOR |
| RO1 | | RESISTOR |
| RO2 | | RESISTOR |
| RO3 | | RESISTOR |
| RO4 | | RESISTOR |
| RO5 | | RESISTOR |
| RO6 | | RESISTOR |
| RO7 | | RESISTOR |
| RO8 | | RESISTOR |
| RO9 | | RESISTOR |
| RO10 | | RESISTOR |
| ROM-58C256 | | MEMORY |
| RSID | 10k | RESISTOR |
| RSTIN | 10k | RESISTOR |
| RTRAP | 10k | RESISTOR |
| RRDY | 10k | RESISTOR |
| SWITCH | | OMRON SWITCH |
| U2 | PRESETLR | |
| X1 | USBSMD | USB Connectors |

Table 1: Bill of Materials

# 3 Details of Software Implementation

## 3.1 Understanding Minimax Algorithm

The Minimax algorithm is basically a recursive algorithm which determines the optimal move to make.In a two-player game like tic-tac-toe where the players play alternatingly in turns the minimax algorithm determines the optimal move by minimising the maximum loss for the given player. Tic-Tac-Toe can be visualised using a game tree where the nodes of the tree are the state of the board and the children of a given node are the new board states obtained by picking various moves as illustrated in the figure. As the depth of the tree increases the one who plays keeps alternating. Minmax can be decomposed into two functions, a MAX function and a MIN function. The MAX function picks the best move for one symbol amongst its various children while the MIN function picks the best move for the other symbol amongst its various children. Thus at a given board state the best branches to take can be determined and thus the optimal move is picked.

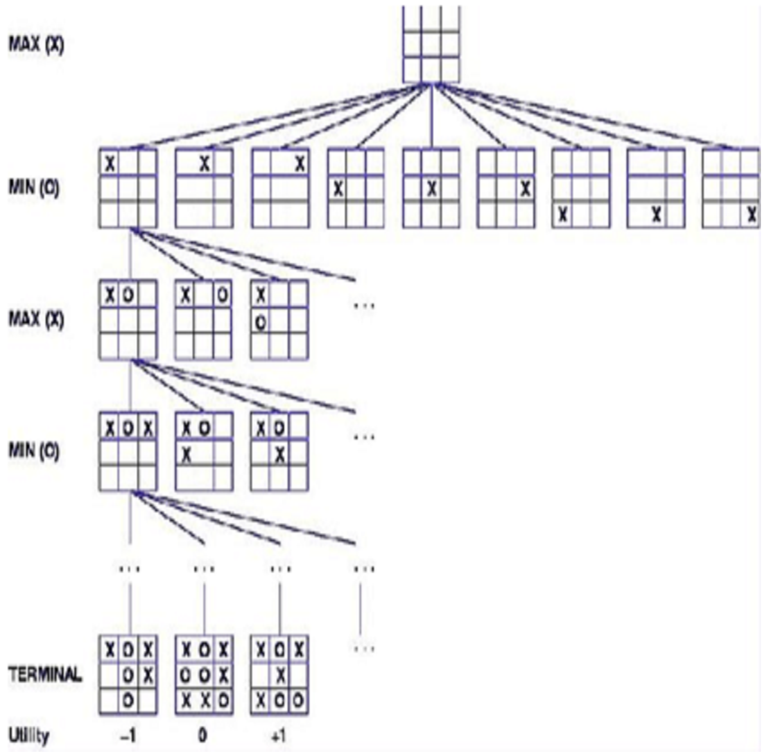


Figure 10: Minimax Algorithm

## 3.2 Code Description

To implement tic-tac-toe, we have maintained a 3x3 board in RAM BOARD which provides various functions with the current state of the game. The game proceeds by alternatively calling functions corresponding to each player or a player and the AI depending on the mode selected. This process goes on until a terminal state is achieved, a terminal state being either victory or getting tied. Various menus have been constructed by using simple keypad input which uses polling to get user input. The LCD has been used to display various strings stored in the ROM depending on the situation. Two more sub-boards exist in RAM TBOARD,BMAP which store a transformed version of the main board state. These sub-boards have been used to update the LED matrix with the current state of the game. Finally, the AI component of our project has been implemented using the Minimax algorithm. The AI functions determine the most optimal move under the given situation recursively and update the state of the internal board.AIMOVE is the overall function which mimics a MAX function to determine the move to play and actually modifies the board. The MAX function iterates over empty positions and calls the MIN function for each position and similarly the MIN function iterates over the empty positions and calls the MAX function for each position. Internally, the AI is always represented as 01H while the player is represented as 0FEH. The head of both MAX and MIN functions contain a terminal state checker and information is moved between these functions using the accumulator. If a terminal state hasnt been achieved MAX iterates over empty spots placing 01H and calls MIN to evaluate the result. MAX picks the place which gives the best result for 01H. MIN works in a similar manner but looks for the case which gives the best result for 0FEH. As soon as an optimal position is detected by either MIN or MAX it immediately exits returning the value.

## 3.3 Code Layout

The code can be divided in the following subsections:
1. Decision making functions for the AI.
2. I/O Functions to take input from the keypad as well as display information on the LCD panel and LED matrix.
3. Game functions which have been used to construct Tic-Tac-Toe
4. Miscellaneous functions which include various delays as well as debugging functions

1) AI Internal Functions:

AIMOVE: Main AI function which makes the move for the AI on the board. Iterates over empty positions and makes calls to MIN to determine which move to make.
MIN: Function which mimics an optimally playing player by iterating over empty positions and making calls to MAX to determine the optimal position for the player. However, does not make changes to the board while doing so. Returns result in accumulator.
MAX: Function which determines thebest move for the AI to make given that the calling MIN function made a move prior to it being called. It also iterates over the empty positions to determine the move.
NOTE: Both MIN and MAX have a state checker at the start to determine if the move played by the calling function causes a terminal state. In such a case no iteration is performed inside MIN/MAX and the terminal state result is returned by MIN/MAX.

2)I/O Functions:

INPUT FUNCTIONS:

KEYIN: Function which determines the input from the keypad connected to port C of the PPI. This function pulses each of the output lines (Active low logic pulse) one at a time and checks the input lines for a response. The input nibble is compared to various combinations representing a keypress. Depending on which pulse and the input nibble combination the row and column is found and thus the exact key is uniquely identified.

OUTPUT FUNCTIONS:

Liquid Crystal Display

LCDINIT: Function which initialises and configures the LCD into the 2-line mode.
LCDSTRINGDISP: Function which takes the memory location of a string and keeps printing out the string till a null character (00H) has been found. String address must be loaded in the HL register before calling this function.
LCDL2: Function which brings the LCD cursor to the next line and prints a string in the similar manner as LCDSTRINGDISP.
CMD: Function to send a command byte to the LCD.
DATA: Function to send a data byte to the LCD.

LED Matrix

UPDATE2: Function which makes a call to TXBOARD to transform the internal board and then uses TBOARD to create another sub-board BMAP whose values are taken to display the board state on the LED matrix which is connected to the PPI.
TXBOARD: Function which transforms internal game board BOARD into TBOARD. TBOARD contains either an inverted version or the same version as the BOARD depending on whether the COLOURFLAG has been set or not.
SETWPOS: Loads three RAM locations beginning at label WPOS with the WINLIST row of the winning combination. This is used as a reference to blink the winning combination at the end.
BLINKRSD: Function which blinks the winning combination stored at WPOS once and it is utilised in the functions DUMMYPLAYERWIN, DUMMYPLAYER1WIN, DUMMYPLAYER2WIN, DUMMYAIWIN to blink repeatedly.

3) Game functions:

CWIN: Function which evaluates the 3x3 board BOARD and determines if someone has won. It returns 01H if 01H wins, returns 0FEH if 0FEH wins and returns 00H if no one wins. The values are returned through the accumulator. It uses the WINLIST to jump to all possible winning combinations.
CFULL: Function which determines if the 3x3 board BOARD is full. It returns 00H if the board is full and 01H otherwise through the accumulator.
PLAYERMOVE: Function used in AI vs Player games which takes input from the player about which position to play at using the keypad. The

numerical portion of the keypad has a direct 1:1 mapping with the board positions.
AIMOVE: As described above this function plays a move according to the minimax algorithm.

4) Miscellaneous functions:

INITRAM: Function which initialises various RAM locations at the start of each game.
DELAY 10/30/100MS: Functions to implement various delays in milliseconds.
SETSOD: Function which sets the SOD led.
RESETSOD: Function which resets the SOD led.
BLINKSOD: Function which blinks the SOD led.

MEMORY MAP:

8200-8208H: BOARD
8100-8108H: BMAP
8400-8408H: TBOARD
8300H: COLOURFLAG

LOOKUP TABLE USED:

WINLIST : Contains offsets to various winning combinations which have been stored in a manner that adding successive offsets gives the actual offset.

## 3.4 Assembly Code

```
        .ORG 0000H
;@ AUTHORS:
;SHIKHAR MAKHIJA 155EC15
;VARUN TRIPATHI 196EC15
        LXI SP,0FFFFH
        CALL BLINKSOD
        CALL BLINKSOD
        MVI A,80H
        OUT 03H
        OUT 03H
        MVI A,88H
        OUT 83H
        OUT 83H
        CALL LCDINIT
        CALL INIT_RAM
        LXI H,STRING17
        CALL LCDSTRINGDISP
        LXI H,STRING18
        CALL LCDL2
        CALL BLINKSODX3
        LXI H,STRING19
        CALL LCDSTRINGDISP
        LXI H,STRING20
        CALL LCDL2
        CALL BLINKSOD
        CALL BLINKSOD
MAINCODE: LXI SP,0FFFFH
        CALL BLINKSOD
        CALL BLINKSOD
        MVI A,80H
        OUT 03H
        OUT 03H
        MVI A,88H
        OUT 83H
        OUT 83H
        CALL LCDINIT
        CALL INIT_RAM
        LXI H,STRING1
        CALL LCDSTRINGDISP
        CALL BLINKSODX5
        LXI H,STRING11
        CALL LCDSTRINGDISP
        LXI H,STRING8
        CALL LCDL2
PLOOP1: CALL KEYIN
        CPI 0AH
        JZ AIGAME
        CPI 0BH
        JZ PLAYERGAME
        JMP PLOOP1
```

```
AIGAME:  LXI H,STRING9
         CALL LCDSTRINGDISP
         LXI H,STRING10
         CALL LCDL2
         CALL UPDATE2
         MVI B,09H
COLOURMENULOOP:LXI H,COLOURFLAG
         CALL KEYIN
         CPI 0AH
         JNZ CSKIP1
         MVI M,00H
         JMP CSKIP2
CSKIP1:  CPI 0BH
         JNZ COLOURMENULOOP
         MVI M,01H
CSKIP2:  LXI H,STRING5
         CALL LCDSTRINGDISP
         LXI H,STRING8
         CALL LCDL2
         LXI H,BOARD
MENULOOP:CALL KEYIN
         CPI 0BH
         JZ PLAY1
         CPI 0AH
         JZ PLAY2
         JNZ MENULOOP
PLAY1:   DCR B
         MVI M,01H
PLAY2:   CALL UPDATE2
         CALL CWIN
         CPI 01H
         JZ DUMMYPLAYERWIN
         CPI 0FEH
         JZ DUMMYAIWIN
INVP:    LXI H,STRING3
         CALL LCDSTRINGDISP
         CALL PLAYERMOVE
         MOV A,C
         CPI 0AAH
         JZ INVP
         CALL UPDATE2
         DCR B
         JZ DUMMYDRAW
         CALL CFULL
         CPI 00H
         JZ DUMMYDRAW
         CALL AIMOVE
         CALL UPDATE2
         CALL CWIN
         CPI 01H
         JZ DUMMYAIWIN
         CPI 0FEH
         JZ DUMMYPLAYERWIN
         ;DRAW BLOCK:
```

```
            CALL CFULL
            CPI 00H
            JZ DUMMYDRAW
            DCR B
            JZ DUMMYDRAW
            JMP PLAY2
ENDOG:      HLT
DUMMYPLAYERWIN: PUSH H
            LXI H,STRING4
            CALL LCDSTRINGDISP
            POP H
            PUSH H
            MVI L, 03H
            CALL SETWPOS
AGAINPLAYER:CALL MODWIN
            CALL UPDATE2
            CALL DELAY100MS
            CALL DELAY100MS
            CALL DELAY100MS
            CALL DELAY100MS
            CALL CMODWIN
            CALL UPDATE2
            CALL DELAY100MS
            CALL DELAY100MS
            CALL DELAY100MS
            CALL DELAY100MS
            DCR L
            JNZ AGAINPLAYER
            POP H
            JMP MAINCODE
DUMMYAIWIN:PUSH H
            LXI H,STRING6
            CALL LCDSTRINGDISP
            POP H
            PUSH H
            MVI L, 03H
            CALL SETWPOS
AGAINAI:CALL MODWIN
            CALL UPDATE2
            CALL DELAY100MS
            CALL DELAY100MS
            CALL DELAY100MS
            CALL DELAY100MS
            CALL CMODWIN
            CALL UPDATE2
            CALL DELAY100MS
            CALL DELAY100MS
            CALL DELAY100MS
            CALL DELAY100MS
            DCR L
            JNZ AGAINAI
            POP H
            JMP MAINCODE
DUMMYDRAW:PUSH H
```

```
            LXI H,STRING7
            CALL LCDSTRINGDISP
            POP H
            PUSH H
            MVI L, 03H
AGAINDRAW:MVI A,0AAH
            OUT 00H
            OUT 01H
            OUT 02H
            CALL DELAY100MS
            CALL DELAY100MS
            CALL DELAY100MS
            CALL DELAY100MS
            MVI A,055H
            OUT 00H
            OUT 01H
            OUT 02H
            CALL DELAY100MS
            CALL DELAY100MS
            CALL DELAY100MS
            CALL DELAY100MS
            DCR L
            JNZ AGAINDRAW
            POP H
            JMP MAINCODE
TXBOARD:PUSH PSW
            PUSH H
            PUSH B
            PUSH D
            LXI H,COLOURFLAG
            MOV C,M
            LXI H,BOARD
            LXI D,TBOARD
            MOV A,C
            CPI 01H
            JZ INVERTBOARD
            ;COPY BOARD:
            MVI B,09H
LOOPTXBOARD1:
            MOV A,M
            XCHG
            MOV M,A
            INX H
            XCHG
            INX H
            DCR B
            JNZ LOOPTXBOARD1
            POP D
            POP B
            POP H
            POP PSW
            RET
INVERTBOARD:MOV A,M
            XCHG
```

```
         CPI  0FEH
         JNZ  STX1
         MVI  M,01H
         JMP  ILEND
STX1:    CPI  01H
         JNZ  STX2
         MVI  M,0FEH
         JMP  ILEND
STX2:MVI M,00H
ILEND:INX  H
         XCHG
         INX  H
         DCR  B
         JNZ  INVERTBOARD
         POP  D
         POP  B
         POP  H
         POP  PSW
         RET
UPDATE2:PUSH  PSW
         PUSH  H
    PUSH  B
    PUSH  D
         CALL  TXBOARD
         LXI  H,TBOARD
         LXI  D,BMAP
         MVI  B,  09H
```

```
BAGAIN2:MOV  A,  M
         XCHG
         CPI  0FEH
         JNZ  NOTFE2
         MVI  A,02H
NOTFE2:  MOV  M,  A
         INX  H
         INX  D
         XCHG
         DCR  B
         JNZ  BAGAIN2
         LXI  H,  BMAP
    MOV  A,M
    RRC
    RRC
         INX  H
    MOV  B,M
    ORA  B
         RRC
         RRC
         INX  H
         MOV  B,M
         ORA  B
         RRC
         RRC
         INX  H
         MOV  B,M
```

```
        ORA B
        RRC
        RRC
        OUT 00H
        INX H
        MOV A,M
RRC
RRC
        INX H
MOV B,M
ORA B
        RRC
        RRC
        INX H
        MOV B,M
        ORA B
        RRC
        RRC
        INX H
        MOV B,M
        ORA B
        RRC
        RRC
        OUT 01H
        MVI A,02H
        RLC
        RLC
```

```
            INX H
            MOV A,M
            OUT 02H
            POP D
            POP B
            POP H
            POP PSW
            RET
;−−−DELAY FUNCTIONS————————
DELAY10MS:          PUSH B
            MVI C,09H
D_L1:       MVI B,0FFH
D_L2:       DCR B
            JNZ D_L2
            DCR C
            JNZ D_L1
            POP B
            RET
DELAY100MS:         PUSH B
            MVI C,0AH
D100_L1:CALL DELAY10MS
            DCR C
            JNZ D100_L1
            POP B
            RET
DELAY30MS:PUSH B
            MVI C,03H
```

```
D30_L1: CALL DELAY10MS
        DCR C
        JNZ D30_L1
        POP B
        RET

DELAY1S:PUSH B
        MVI C,0AH
D1S_L1: CALL DELAY100MS
        DCR C
        JNZ D1S_L1
        POP B
        RET
;--MISC FUNCTIONS----------------
SETSOD:
        PUSH PSW
        MVI A,0C0H
        SIM
        POP PSW
        RET
RESETSOD:
        PUSH PSW
        MVI A,040H
        SIM
        POP PSW
        RET
BLINKSOD:
        CALL SETSOD
        CALL DELAY100MS
        CALL DELAY100MS
        CALL RESETSOD
        CALL DELAY100MS
        CALL DELAY100MS
        RET
BLINKSODX5:
        CALL BLINKSOD
    CALL BLINKSOD
    CALL BLINKSOD
    CALL BLINKSOD
    CALL BLINKSOD
    RET
BLINKSODX3:
        CALL BLINKSOD
    CALL BLINKSOD
    CALL BLINKSOD
    RET
;-----------------------------------
;-----KEYPAD FUNCTIONS----------
KEYIN:  ;ACCUMULATOR RETURNS VALUE
        PUSH B
        MVI A,088H
        OUT 83H
        CALL DELAY10MS
        OUT 83H
```

```
        ;;PULSING SEQUENCE
KEYREP:  ;;CALL BLINKSOD
         CALL DELAY100MS
         MVI A,00001110B
         OUT 82H
         IN 82H
         ANI 0F0H;MASK
         CALL KCHECKP1
         CPI 10H
         JNZ KEYFOUND
         MVI A,00001101B
         OUT 82H
         IN 82H
         ANI 0F0H;MASK
         CALL KCHECKP2
         CPI 10H
         JNZ KEYFOUND
         MVI A,00001011B
         OUT 82H
         IN 82H
         ANI 0F0H;MASK
         CALL KCHECKP3
         CPI 10H
         JNZ KEYFOUND
         MVI A,00000111B
         OUT 82H
         IN 82H
         ANI 0F0H;MASK
         CALL KCHECKP4
         CPI 10H
         JNZ KEYFOUND
         JZ KEYREP
KEYFOUND:POP B
         RET
KCHECKP1:  ;A HAS INPUT SEQUENCE 4 BIT
         CPI 11100000B
         JNZ KS1_1
         MVI A,00H
         RET
KS1_1:   CPI 11010000B
         JNZ KS1_2
         MVI A,01H
         RET
KS1_2:   CPI 10110000B
         JNZ KS1_3
         MVI A,02H
         RET
KS1_3:CPI 01110000B
         JNZ KS1_4
         MVI A,0AH
         RET
KS1_4:MVI A,10H
         RET
```

```
KCHECKP2:;A HAS INPUT SEQUENCE 4 BIT
         CPI 11100000B
         JNZ KS2_1
         MVI A,03H
         RET
KS2_1:CPI 11010000B
         JNZ KS2_2
         MVI A,04H
         RET
KS2_2:CPI 10110000B
         JNZ KS2_3
         MVI A,05H
         RET
KS2_3:CPI 01110000B
         JNZ KS2_4
         MVI A,0BH
         RET
KS2_4:MVI A,10H
         RET

KCHECKP3:;A HAS INPUT SEQUENCE 4 BIT
         CPI 11100000B
         JNZ KS3_1
         MVI A,06H
         RET
KS3_1:CPI 11010000B
         JNZ KS3_2
         MVI A,07H
         RET
KS3_2:CPI 10110000B
         JNZ KS3_3
         MVI A,08H
         RET
KS3_3:CPI 01110000B
         JNZ KS3_4
         MVI A,0CH
         RET
KS3_4:MVI A,10H
         RET

KCHECKP4:CPI 11100000B
         JNZ KS4_1
         MVI A,0EH
         RET
KS4_1:CPI 11010000B
         JNZ KS4_2
         MVI A,09H
         RET
KS4_2:CPI 10110000B
         JNZ KS4_3
         MVI A,0FH
         RET
KS4_3:CPI 01110000B
         JNZ KS4_4
```

```
         MVI A,0DH
         RET
KS4_4:MVI A,10H
         RET

CMD: OUT 81H
        MVI A,80H;RS=0,E=1
        OUT 80H
        CALL DELAY
        MVI A,00H;RS=0,E=0
        OUT 80H
        CALL DELAY
        RET

DATA:OUT 81H
        MVI A,0A0H;RS=1,E=1
        OUT 80H
        CALL DELAY
        MVI A,20H;RS=1,E=0
        OUT 80H
        CALL DELAY
        RET

DELAY:MVI C,0FFH
LOOP2:DCR C
          JNZ LOOP2
          RET

DELAY2: PUSH B
        MVI B,0FFH
AG2:MVI C,0FFH
AG1:DCR C
        JNZ AG1
        DCR B
        JNZ AG2
        POP B
        RET
LCDINIT:  ; LCD INITIALISER
        PUSH PSW
        MVI A,38H
        CALL CMD
        MVI A,38H
        CALL CMD
        CALL DELAY
        MVI A,01H
        CALL CMD
        MVI A,01H
        CALL CMD
        CALL DELAY2
        MVI A,0CH
        CALL CMD
        MVI A,80H
        CALL CMD
        POP PSW
```

```
          RET
LCDSTRINGDISP: PUSH H
          CALL LCDINIT
PRINT:MOV A,M
          INX H
          CPI 00H
          JZ STREND
          CALL DATA
          JMP PRINT
LCD2:     MVI A,0C0H
          CALL CMD
          RET
LCDL2:    PUSH H
          CALL LCD2
PRINT2:   MOV A,M
          INX H
          CPI 00H
          JZ STREND
          CALL DATA
          JMP PRINT2

STREND:   POP H
                    RET
;AIMOVE:
AIMOVE:   CALL BLINKSOD
          PUSH PSW
          PUSH B
          PUSH D
          PUSH H
          LXI H,STRING2
          CALL LCDSTRINGDISP
          CALL DELAY100MS
          LXI H,BOARD
          MVI D,0FFH
          MVI C,09H
AILOOP:   MOV A,M
          CPI 00H
          JNZ INVM
          MVI M,01H
          CALL MIN
          CPI 01H
          JZ EXITA
          ;NON WINNING CASES
          CMP D
          JNC SKAI
          MOV D,A
          MOV E,C
SKAI:MVI M,00H
INVM:INX H
          DCR C
          JNZ AILOOP
          LXI H,BOARD
          MVI D,00H
          MVI A,09H
```

```
        SUB E
        MOV E,A
        DAD D
        MVI M,01H
EXITA:  POP H
        POP D
        POP B
        POP PSW
        RET
MIN:PUSH H
        PUSH B
        PUSH D
        LXI H,BOARD
        CALL CWIN
        CPI 00H
        JNZ EXITM
        CALL CFULL
        CPI 00H
        JZ EXITM
        LXI H,BOARD
        MVI D,02H
        MVI C,09H
MINL:   MOV A,M
        CPI 00H
        JNZ MININV
        MVI M,0FEH
        CALL MAX
        CPI 0FEH
        JZ XITFE
        CMP D
        JNC MINSKP
        MOV D,A
MINSKP: MVI M,00H
MININV: INX H
        DCR C
        JNZ MINL
        MOV A,D
        JMP EXITM
XITFE:  MVI M,00H
EXITM:  POP D
        POP B
        POP H
        RET
MAX:PUSH H
        PUSH B
        PUSH D
        LXI H,BOARD
        CALL CWIN
        CPI 00H
        JNZ EXITMX
        CALL CFULL
        CPI 00H
        JZ EXITMX
        LXI H,BOARD
```

```
        MVI D,0FFH
        MVI C,09H
MAXL:   MOV A,M
        CPI 00H
        JNZ MAXINV
        MVI M,01H
        CALL MIN
        CPI 01H
        JZ XIT01
        CMP D
        JNC MAXSKP
        MOV D,A
MAXSKP: MVI M,00H
MAXINV: INX H
        DCR C
        JNZ MAXL
        ;NON 01 CASES:
        MOV A,D
        JMP EXITMX
XIT01:  MVI M,00H
EXITMX: POP D
        POP B
        POP H
        RET
CFULL:  PUSH H
        LXI H,BOARD
        MVI C,09H
EX2:MOV A,M
        CPI 00H
        JZ EX1
        INX H
        DCR C
        JNZ EX2
        POP H
        MVI A,00H
        RET
EX1:POP H
        MVI A,01H
        RET
CWIN:   PUSH D
        LXI D,WINLIST
        PUSH H
        LXI H,BOARD
        LXI B, 0A000H; POINTER
        MVI A,08H
        DCX D
        PUSH B
CLOOP: PUSH PSW
        XRA A
        INX D
        PUSH H
        MVI B,00H
        XCHG
        MOV C,M
```

```
XCHG
DAD B
ADD M
XCHG
INX H
MOV C,M
XCHG
DAD B
ADD M
XCHG
INX H
MOV C,M
XCHG
DAD B
ADD M
POP H  ;RESTORING HL
CPI 0FAH
JZ WINO
CPI 03H
JZ WINX
POP PSW  ;RESTORING A
DCR A
POP B
STAX B
PUSH B
JNZ CLOOP
POP B
        POP H
        POP D
        MVI A,00H
        RET
WINO:
        POP PSW
        POP B
        POP H
        POP D
        MVI A,0FEH
        RET
WINX:
        POP PSW
        POP B
        POP H
        POP D
        MVI A,01H
        RET
;CONVERSION TABLE:
WINLIST:.DB 00H,01H,01H
        .DB 03H,01H,01H
        .DB 06H,01H,01H
        .DB 00H,03H,03H
        .DB 01H,03H,03H
        .DB 02H,03H,03H
        .DB 00H,04H,04H
        .DB 02H,02H,02H
```

```
;BOARD STATE:
BOARD:    .EQU 8200H
          .DB 0FEH,00H,0FEH
          .DB 0FEH,001H,001H
          .DB 00H,0FEH,001H
BMAP:.EQU 8100H
      .DB 00H,00H,00H
      .DB 00H,00H,00H
      .DB 00H,00H,00H
TBOARD:  .EQU 8400H
COLOURFLAG .EQU 8300H
WINPOS:  .EQU 8310H
MCOUNTER .EQU 8305H
INIT_RAM:PUSH B
          PUSH H
          MVI B,09H
          LXI H,BMAP
RCLRL:MVI M,00H
          INX H
          DCR B
          JNZ RCLRL
          MVI B,09H
          LXI H,BOARD
BCLRL:MVI M,00H
          INX H
          DCR B
          JNZ BCLRL
          MVI B,09H
          LXI H,TBOARD
TBCLRL:  MVI M,00H
          INX H
          DCR B
          JNZ TBCLRL
          LXI H,WINPOS
          MVI M,00H
          INX H
          MVI M,00H
          INX H
          MVI M,00H
          POP H
          POP B
          RET
PLAYERMOVE:
          PUSH PSW
          LXI H,BOARD
          MVI C,00H
          CALL KEYIN ;0-8 VALUE MAPPED
          ;REJECTING OTHER KEYPAD INPUTS:
          CPI 09H
          JNC INVALID
          CALL CHECKCORRECT
          CPI 0AAH
          JZ INVALID
          ADD L
```

```
          MOV L, A
          MVI M,0FEH  ;X=01H ASSUMED
          PUSH B
          LXI H,BOARD
          CALL CWIN
          POP B
          POP PSW
          RET
INVALID:  MVI C,0AAH
          POP PSW
          RET
CHECKCORRECT:   PUSH H
                PUSH B
                LXI H, BOARD
                MOV B, A
                MOV A, L
                ADD B
                MOV L, A
                MOV A, M
                CPI 00H
                JZ OK
                MVI A,0AAH
                POP B
                POP H
                RET
OK:             MOV A,B
                POP B
                POP H
                RET
PLAYERMOVE2:
          PUSH PSW
          LXI H,BOARD
          MVI C,00H
          CALL KEYIN
          CPI 09H
          JNC INVALID1
          CALL CHECKCORRECT1
          CPI 0AAH
          JZ INVALID1
          ADD L
          MOV L, A
          MVI M,01H  ;P2=01H ASSUMED
          PUSH B
          LXI H,BOARD
          CALL CWIN
          POP B
          POP PSW
          RET
INVALID1: MVI C,0AAH
          POP PSW
          RET
CHECKCORRECT1:PUSH H
          PUSH B
          LXI H, BOARD
```

```
        MOV B, A
        MOV A, L
        ADD B
        MOV L, A
        MOV A, M
        CPI 00H
        JZ OK1
        MVI A,0AAH
        POP B
        POP H
        RET
OK1:MOV A,B
        POP B
        POP H
        RET
PLAYERGAME:     LXI H,STRING12
        CALL LCDSTRINGDISP
        LXI H,STRING10
        CALL LCDL2
XCOLOURMENULOOP:LXI H,COLOURFLAG
        CALL KEYIN
        CPI 0AH
        JNZ XCSKIP1
        MVI M,00H
        JMP XCSKIP2
XCSKIP1:CPI 0BH
        JNZ XCOLOURMENULOOP
        MVI M,01H
XCSKIP2:CALL UPDATE2
        CALL CWIN
        CPI 01H
        JZ DUMMYPLAYER2WIN
        CPI 0FEH
        JZ DUMMYPLAYER1WIN    ;EDIT TO P2
INVP1:LXI H,STRING13
        CALL LCDSTRINGDISP
        CALL PLAYERMOVE
        MOV A,C
        CPI 0AAH
        JZ INVP1
        CALL UPDATE2
        CALL CWIN
        CPI 01H
        JZ DUMMYPLAYER2WIN
        CPI 0FEH
        JZ DUMMYPLAYER1WIN    ;EDIT TO P2
        CALL CFULL
        CPI 00H
        JZ DUMMYDRAW
INVP2:LXI H,STRING15
        CALL LCDSTRINGDISP
        CALL PLAYERMOVE2
        MOV A,C
        CPI 0AAH
```

```
        JZ INVP2
        CALL UPDATE2
        CALL CWIN
        CPI 01H
        JZ DUMMYPLAYER2WIN
        CPI 0FEH
        JZ DUMMYPLAYER1WIN
        CALL CFULL
        CPI 00H
        JZ DUMMYDRAW
        JMP XCSKIP2
SETWPOS:
        PUSH D
        LXI D,WINLIST
        PUSH H
        LXI H,BOARD
        MVI A,08H
        DCX D
MLOOP1: PUSH PSW
        XRA A
        INX D
        PUSH H
        MVI B,00H
        XCHG
        MOV C,M
        XCHG
        DAD B
        ADD M
        XCHG
        INX H
        MOV C,M
        XCHG
        DAD B
        ADD M
        XCHG
        INX H
        MOV C,M
        XCHG
        DAD B
        ADD M
        POP H ;RESTORING HL
        CPI 0FAH
        JZ WINO1
        CPI 03H
        JZ WINX1
        POP PSW ;RESTORING A
        DCR A
        JNZ MLOOP1
        POP H
        POP D
        MVI A,00H
        ;EMSG:
        LXI H,ERRORMSG
        CALL LCDSTRINGDISP
```

```
        HLT
        ;ENDOFEMSG
        RET
WINX1:  NOP
WINO1:  POP PSW
        MOV B,A
        MVI A,08H
        SUB B
        MOV B,A
        XRA A
        ADD B
        ADD B
        ADD B
        MOV C,A  ;OFFSET STORED IN C
        LXI H,WINPOS
        LXI D,WINLIST
        XCHG
        MOV A,L
        ADD C
        MOV L,A
        MOV A,M
        STAX D
        INX D
        INX H
        MOV A,M
        STAX D
        INX D
        INX H
        MOV A,M
        STAX D
        POP H
        POP D
        RET
;————————————————————————————————
;BLINKRCD (Row—Column—Diagonal)
BLINKRCD:
        PUSH H
        PUSH D
        PUSH B
        LXI D,BOARD
        LXI H,WINPOS
        MOV A,M
        XCHG
        ADD L
        MOV L,A
        MOV B,M
        MVI M,00H
        XCHG
        ;WP
        INX H
        MOV A,M
        XCHG
        ;B
        ADD L
```

```
MOV L,A
MVI M,00H
XCHG
;WP
INX H
MOV A,M
XCHG
ADD L
MOV L,A
MVI M,00H
CALL UPDATE2
CALL BLINKSOD
CALL DELAY100MS
;SETTING WINNING COMBO TO B:
LXI H,WINPOS
LXI D,BOARD
MOV A,M
XCHG
ADD L
MOV L,A
MOV M,B
XCHG
INX H
MOV A,M
XCHG
ADD L
MOV L,A
MOV M,B
XCHG
INX H
MOV A,M
XCHG
ADD L
MOV L,A
MOV M,B
CALL UPDATE2
CALL BLINKSOD
CALL DELAY100MS
POP B
POP D
POP H
RET
DUMMYPLAYER1WIN:PUSH H
LXI H,STRING14
CALL LCDSTRINGDISP
POP H
PUSH H
MVI L, 05H
CALL SETWPOS
AGAINPLAYER1:CALL BLINKRCD
DCR L
JNZ AGAINPLAYER1
POP H
JMP MAINCODE
```

```
DUMMYPLAYER2WIN:PUSH H
        LXI H,STRING16
        CALL LCDSTRINGDISP
        POP H
        PUSH H
        MVI L, 05H
        CALL SETWPOS
AGAINPLAYER2:CALL BLINKRCD
        DCR L
        JNZ AGAINPLAYER2
        POP H
        JMP MAINCODE
DUMMYPLAYERWIN:PUSH H
        LXI H,STRING4
        CALL LCDSTRINGDISP
        POP H
        PUSH H
        MVI L, 05H
        CALL SETWPOS
AGAINPLAYER:CALL BLINKRCD
        DCR L
        JNZ AGAINPLAYER
        POP H
        JMP MAINCODE
DUMMYAIWIN:PUSH H
        LXI H,STRING6
        CALL LCDSTRINGDISP
        POP H
        PUSH H
        MVI L, 05H
        CALL SETWPOS
AGAINAI:        CALL BLINKRCD
        DCR L
        JNZ AGAINAI
        POP H
        JMP MAINCODE

DUMMYDRAW:      PUSH H
        LXI H,STRING7
        CALL LCDSTRINGDISP
        POP H
        PUSH H
        MVI L, 03H
AGAINDRAW:      MVI A,0AAH
        OUT 00H
        OUT 01H
        OUT 02H
        CALL DELAY100MS
        CALL DELAY100MS
        CALL DELAY100MS
        CALL DELAY100MS
        MVI A,055H
        OUT 00H
        OUT 01H
```

```
        OUT 02H
        CALL DELAY100MS
        CALL DELAY100MS
        CALL DELAY100MS
        CALL DELAY100MS
        DCR L
        JNZ AGAINDRAW
        POP H
        JMP MAINCODE
;List of strings used:
STRING5:        .DB "Play First?",00H
STRING8:        .DB "A)Yes B)No",00H
STRING9:        .DB "Play as:",00H
STRING10:       .DB "A)Green  B)Red",00H
STRING1:        .DB "Tic-Tac-Toe",00H
STRING2:        .DB "AI MOVE",00H
STRING3:        .DB "Player Move",00H
STRING4:        .DB "Player Wins!",00H
STRING6:        .DB "AI Wins!",00H
STRING7:        .DB "It's a Draw!",00H
STRING11:       .DB "Play Vs. AI?",00H
STRING12:       .DB "Player1 is:",00H
STRING13:       .DB "Player1's Turn:",00H
STRING14:       .DB "Player1 Wins!",00H
STRING15:       .DB "Player2's Turn:",00H
STRING16:       .DB "Player2 Wins!",00H
STRING17:       .DB "MICROPROCESSOR",00H
STRING18:       .DB "PROJECT BY:",00H
STRING19:       .DB "155/EC/15",00H
STRING20:       .DB "196/EC/15",00H
ERRORMSG:       .DB "ERROR!",00H
```

# 4 Gantt Chart Revisited

The project duration was 16 weeks, starting from January 1, 2018 till May 23, 2018. The planned and actual project implementation routine is given below. Moreover, the Gantt Chart initially proposed is shown in the following figure.

Table 2: Actual v/s Proposed Schedule

| Task | Start Date | End Date | Duration | Actual Start Date | Actual End Date |
|---|---|---|---|---|---|
| Project Proposal | 05-Jan | 15-Jan | 10 | 05-Jan | 15-Jan |
| C Code Prototype | 12-Jan | 14-Jan | 2 | 12-Jan | 14-Jan |
| Basics of 8085 & Assembly Language | 02-Jan | 01-Apr | 89 | 02-Jan | 20-Mar |
| Analysing Hardware Requirements | 01-Feb | 07-Feb | 6 | 01-Feb | 07-Feb |
| Schematic Development | 07-Feb | 24-Feb | 17 | 07-Feb | 16-Mar |
| Constructing the Board Layout | 24-Feb | 14-Mar | 18 | 16-Mar | 29-Mar |
| Ordering and Obtaining the PCB | 15-Mar | 29-Mar | 14 | 29-Mar | 07-Apr |
| Board Fabrication | 30-Mar | 04-Apr | 5 | 08-Apr | 12-Apr |
| SoftwareDevelopment(1): Basic Tic-Tac-Toe Model | 15-Mar | 01-Apr | 17 | 12-Mar | 05-Apr |
| SoftwareDevelopment(2): Constructing the Minmax Algorithm Model | 01-Apr | 15-Apr | 14 | 17-Mar | 22-Mar |
| Conducting Alpha Tests | 15-Apr | 25-Apr | 10 | 12-Apr | 25-Apr |
| Preparing Documentation | 01-Apr | 25-Apr | 24 | 25-Apr | 30-Apr |

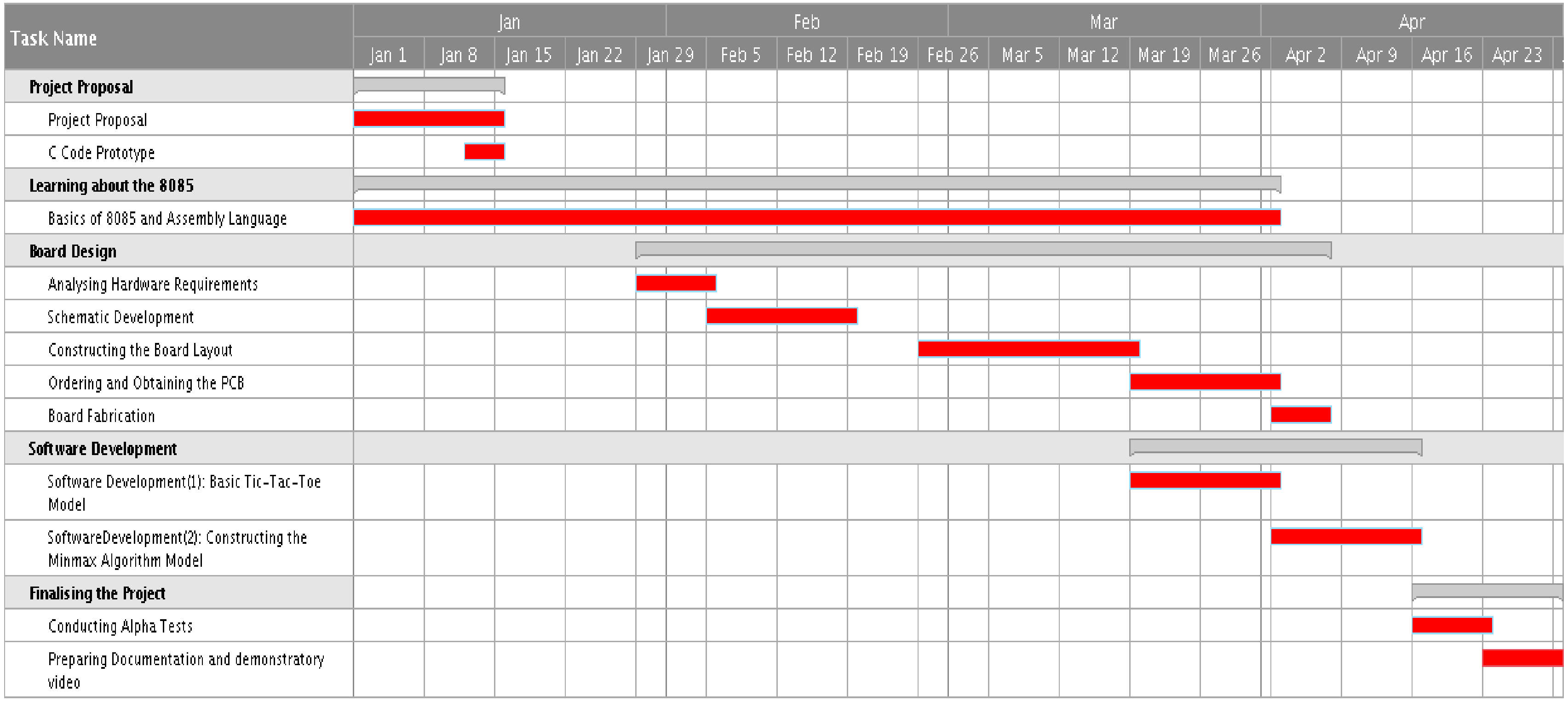


Figure 11: Proposed Gantt Chart

# 5 Future Scope

The proposed project is based upon Minimax Algorithm. A limitation of this approach is it plays the first optimal move found and not goes for optimal decision tree height. Moreover, the first move takes exponentially large time. To solve this problem, alpha beta pruning technique can be used in future. Alphabeta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It stops completely evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. When applied to a standard minimax tree, it returns the same move as minimax would, but prunes away branches that cannot possibly influence the final decision. Thus, by implementing this, the problem of the first move taking a large time will be solved.

# References

*Game Theory*. URL: `https://en.wikipedia.org/wiki/Game_theory` (visited on 01/29/2018).

Gaonkar, R.S. *Microprocessor Architecture, Programming, and Applications with the 8085*. Prentice Hall, 2002. ISBN: 9780130195708.

*Minimax*. URL: `https://en.wikipedia.org/wiki/Minimax` (visited on 01/29/2018).

Tenenbaum, A.M., Y. Langsam, and M. Augenstein. *Data Structures Using C*. Prentice-Hall Of India Pvt. Limited, 2003. ISBN: 9788120306967.