VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE & ENGINEERING

COMPUTER ARCHITECTURE

Assignment Report

Instructor: Phạm Quốc Cường

Student: Hoàng Cao Quốc Thắng - 2050020

HO CHI MINH CITY, MARCH 2022

Contents

1 Topic

Please design and write MIPS assembly language for implementing a text-based 5x5 board Tic-Tac-Toe game for two players with following requirements.

1. During the first turn of both players, they are not allowed to choose the central point (row 3 & column 3).

2. Any player who has 3 points in a row, column or diagonal will be the winner.

3. Players can undo 1 move before the opponent plays.

2 Preface

In this report, I would like to give clear explanations in term of my MIPS code in file “tictactoe.asm” (translated from the file “tictactoe_Ccode.c” that I had written in this link: https://github.com/thanghoang7020202/tictactoe.git)

3 Introduction

The emergence of personal computers in the mid 20th century revolutionized numerous aspects of our lives, and how we play our usual games was no

exception. Popular titles were being put into "video games" and played on computers, Tic-Tac-Toe being one of the many. This is a formal report of a simple Tic-Tac-Toe implementation, written in assembly language designed for MIPS processors.

Tic-Tac-Toe is played on a 5x5 grid by two players alternately placing X’s and O’s on one of the 25 spaces in the grid. The simplicity of the game meant that if played optimally by both players, the result always ends in a draw, regardless of who is playing first (the general consensus is that X plays first). For a simple yet fully-fleshed Tic-Tac-Toe game with an acceptable interface, we will need to prompt the players to input location of marks on the board and print out the board after each turn. It is obvious that until the game ends, a checking function is executed after each turn is played. A win occurs when either of the players manage to place three of their respective marks on the same row, column or diagonal line. A draw is

resulted from both players failing to make such a line from their mark, after the 25th turn.

4 Explanations

- Interface and how to use:

As mentioned in topic, the interface of this code is text-based and interact in Run I/O window (as in picture E1 below).

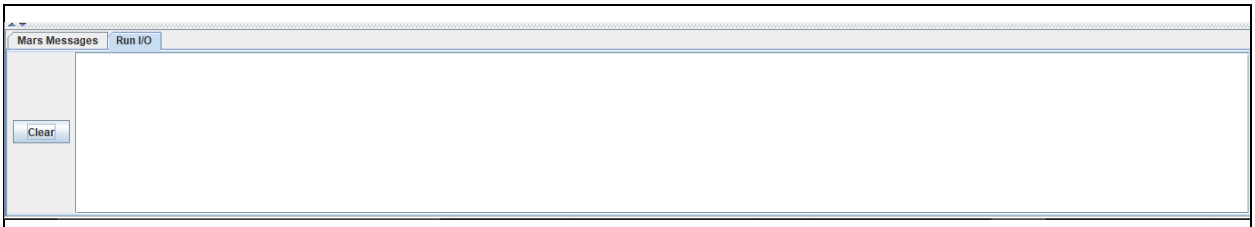


Image E1: screenshot of Run I/O window in MARS 4.5.

In order to run the program, we need to Assemble the source code in file "tictactoe.asm" and run that program. The console (or Run I/O window) would show in picture E2 and require user to choose 1 for place maker in position or 2 to undo the last move.

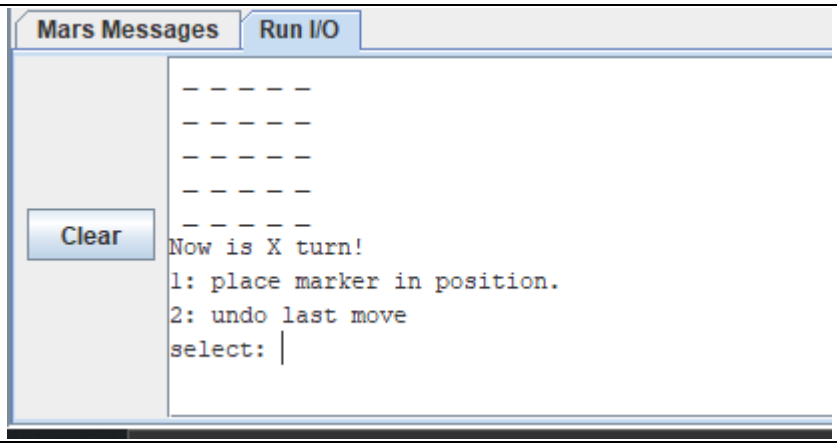


Image E2: screenshot of first appearance of the program.

In case user select "1" and "enter":

The very first turn always be X then O and back to X until 1 player win or draw (finish 25 moves without any line of 3 X or O including horizontal, vertical and diagonal lines). It is significant to notice that we are not allow to choose (3,3), which is a central point as mentioned in the requirement (image E3.1).

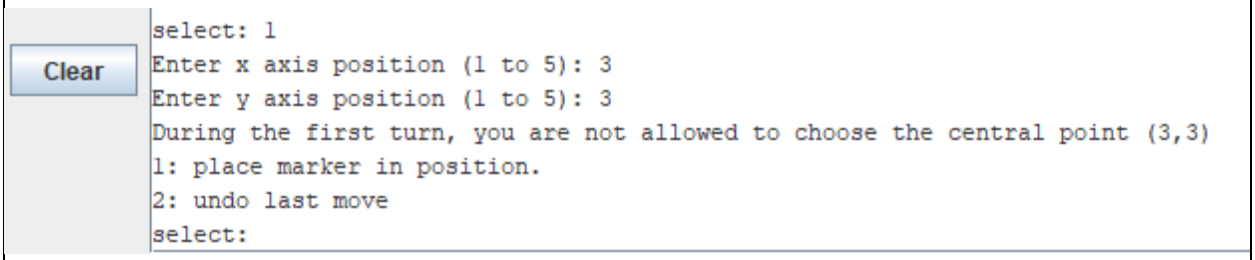
In case user select “2” and “enter”:

It is obvious that players can not undo in their first turn which mean there is no previous move. Moreover, the program only allow player to undo once in each turn before their opponent make their move as aforementioned reason (image E3.2).

Other cases:

If user enter random number or character, the system will require them to input again as in image E3.3.

Image E3.1: Example of user input in middle point.



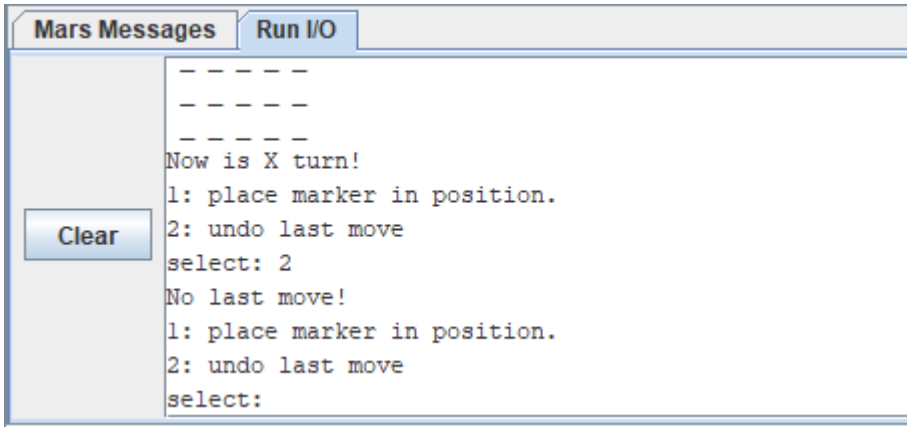


Image E3.2: Example of player select 2 in their first turn.

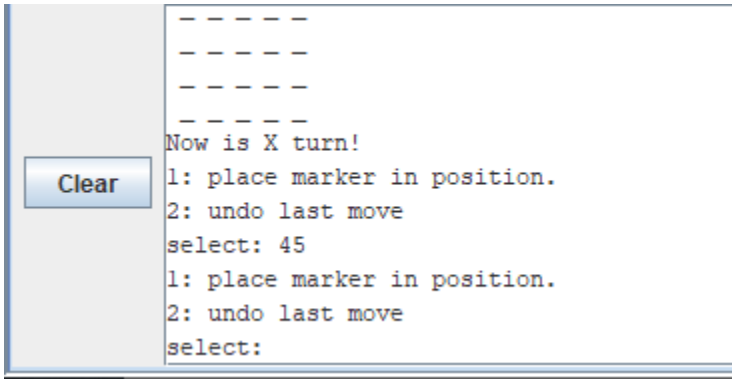


Image E3.3: Example of user input “45” and press “enter”.

Lastly, if someone satisfy one of all win conditions the program will print out to Run I/O as image E4.

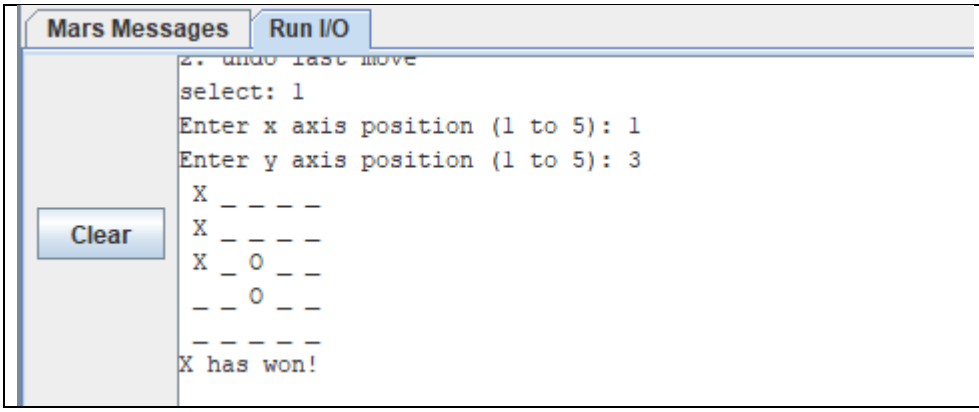


Image E4: Example of X win with mark in a column from (1,1) to (1,3).

- Algorithms (flowchart):

Link to full flowchart (flowchart_tictactoe.drawio): https://github.com/thanghoang7020202/tictactoe.git

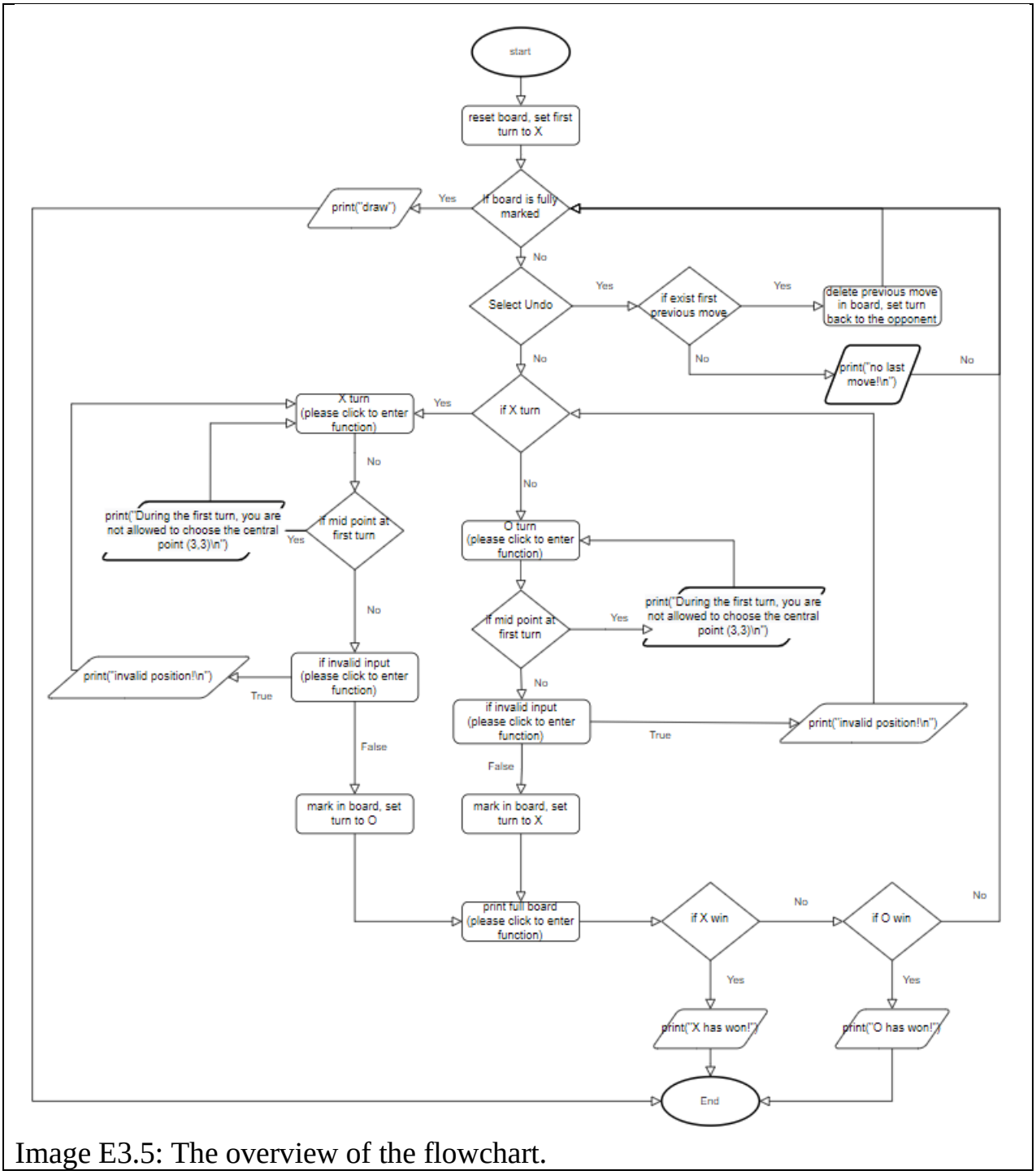


Image E3.5: The overview of the flowchart.

There are some specific hidden code blocks such as “X turn”, “If invalid input”, “print board”,… will be shown since being double clicked in as illustrated in image E3.6.

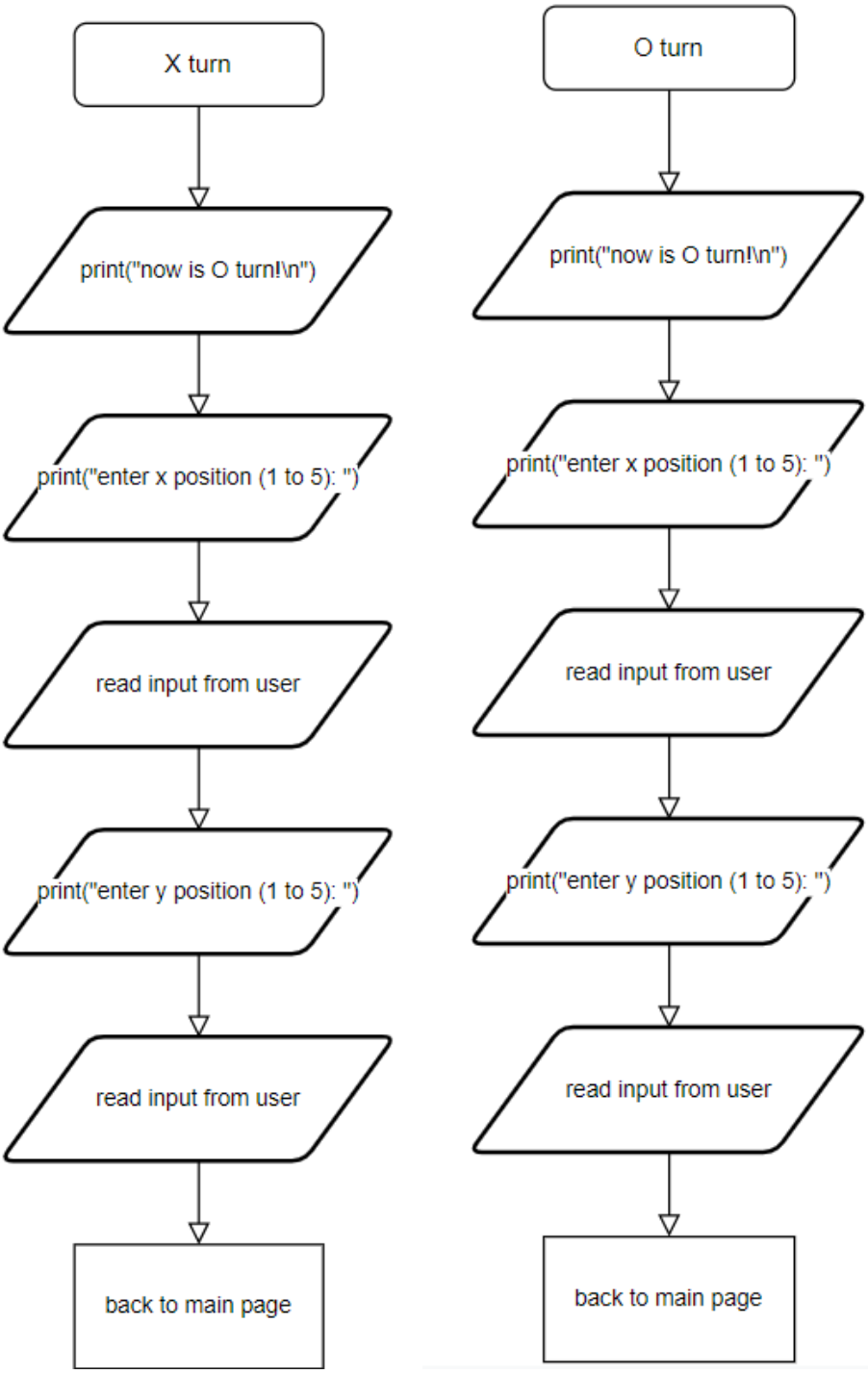


Image E3.6: block “X turn” and “O turn” since being double clicked.

Moreover, there are two complicated function that need to be show clearly:

“Print full board”: Two variables, “count_col” and “count_line”, are initialize to be 0; “value” is taking identifying value from each position in the array (-1 is empty, 0 is “O” and 1 is “X”). For instance, arr[0] = 1 then the value at the top left of the board is “X” and “value” = 1.

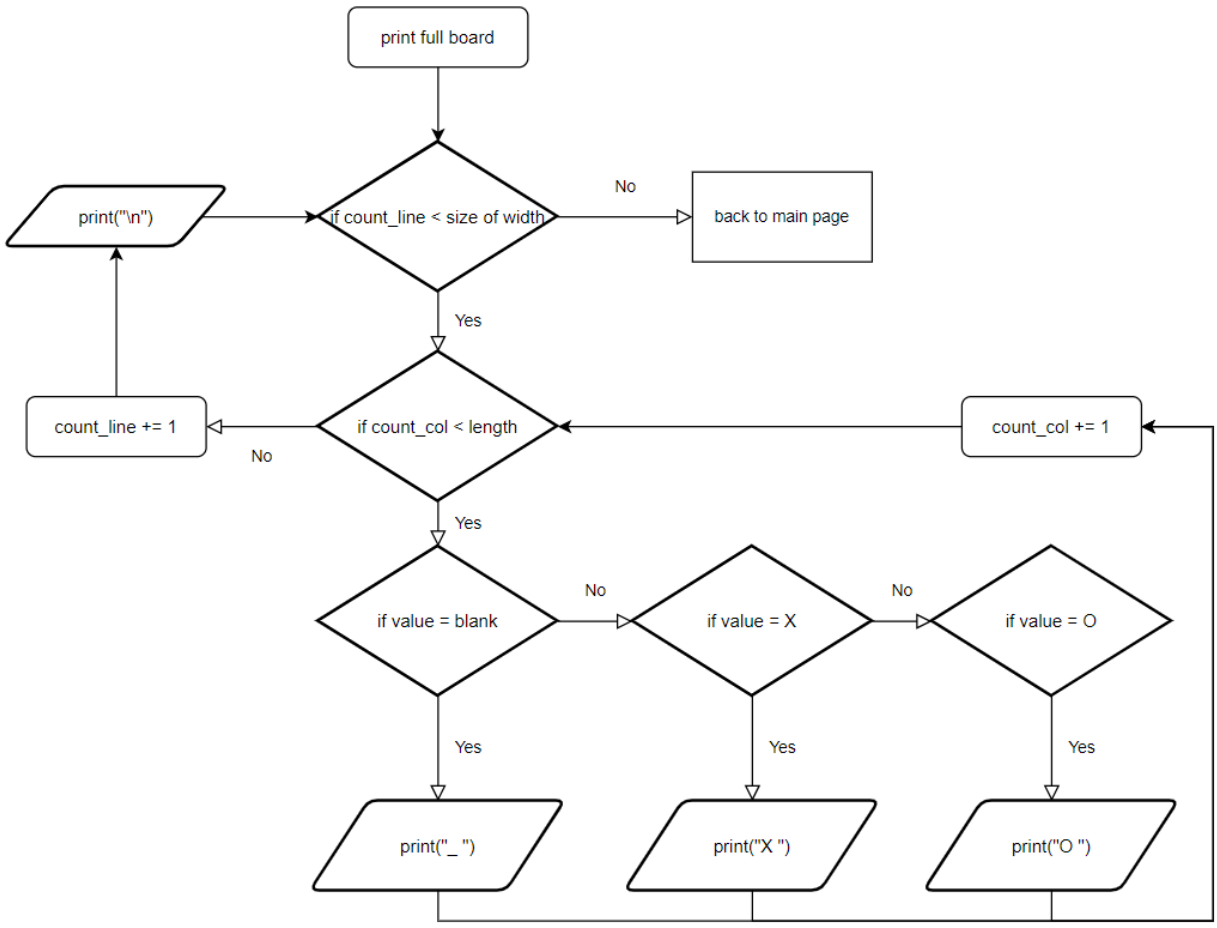


Image E3.7: “Print full board” function (block).

Secondly, “If invalid input” is the function that return true if error input detected and false otherwise. This function also manage to ask player to input correctly, otherwise, they have to input again from the previous block.

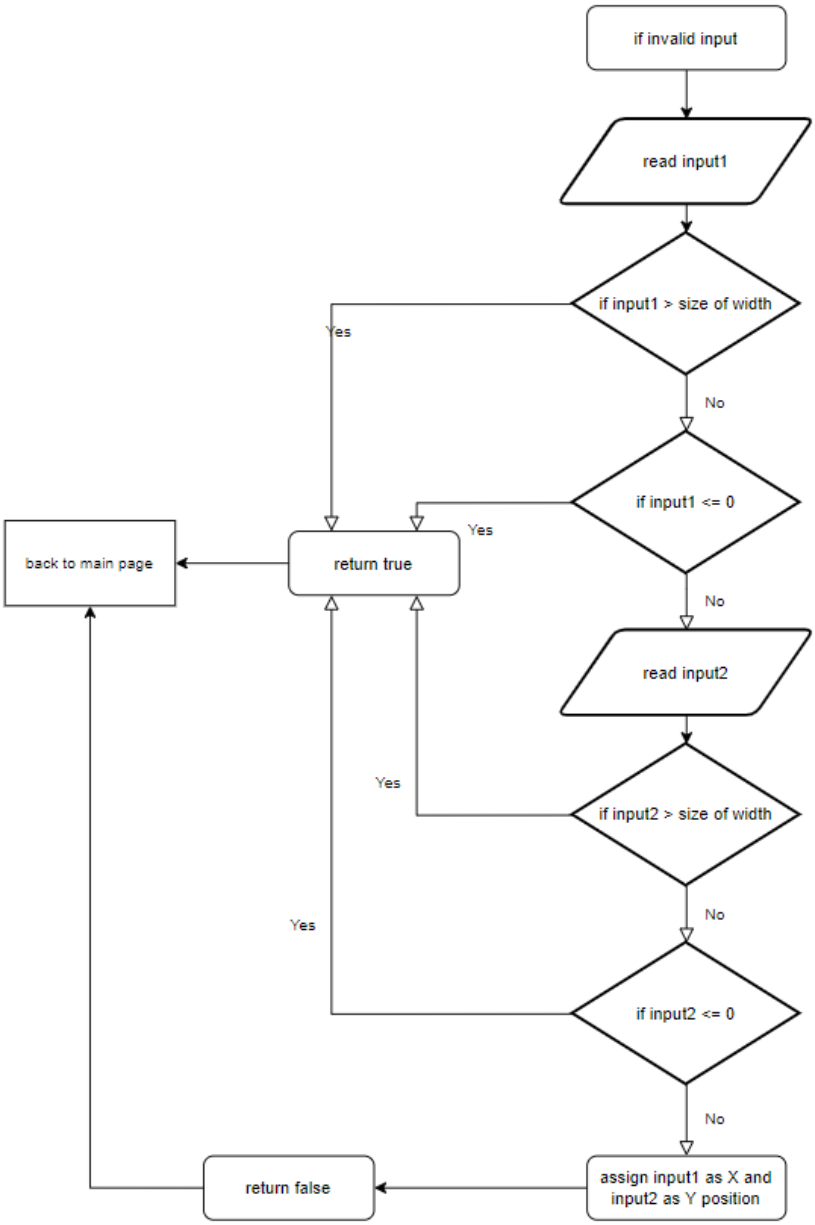


Image E3.8: Example of player select 2 in their first turn.

5 Conclusion

Through this report, we have learned how to implement a simple Tic-Tac-Toe game in assembly language designed for MIPS processors, using different MIPS instructions and logic of the game itself.

---The End---