VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

Computer Architecture

Assignment Sem 242

FIVE IN A ROW

Advisor(s):	Nguyễn Thiên Ân	
Student(s):	Đặng Quang Dũng	2352205
	Nguyễn Bình	2352120
	Lê Mạnh Quân	2352993

HO CHI MINH CITY, APRIL 2025

Contents

1 Introduction

1.1 Brief Introduction to Gomoku/Five in a Row

Gomoku, also known as Five in a Row or Caro in Vietnamese, is a classic abstract strategy board game with a rich history dating back to the mid-1700s during Japan's Edo period. The name "gomoku" derives from Japanese: "go" meaning five, "moku" as a counter word for pieces, and "narabe" meaning line-up. This elegant yet strategic game has gained widespread popularity throughout Asia, particularly in Japan, China, Korea, and Vietnam.

Since its creation, Gomoku has evolved with various rule modifications that have kept the game fresh and competitive over centuries. The game's enduring appeal is evidenced by the World Gomoku Championships, which have been held regularly since 1989, featuring different host countries and champions from around the world. Despite its simple concept, Gomoku offers deep strategic complexity that appeals to players of all ages and skill levels.

1.2 Rules and Winning Conditions

The standard version of Gomoku is played on a square grid board, traditionally 15×15 intersections. In the Vietnamese adaptation, players use "X" and "O" markers instead of the traditional black and white stones used in Asian versions. The core gameplay mechanics are as follows:

- Players take turns placing their markers ("X" for Player 1 and "O" for Player 2) on empty intersections of the grid. Player 1 makes the first move, followed by alternating turns.
- The objective is to be the first to form an unbroken line of five markers of one's own symbol horizontally, vertically, or diagonally. If the board becomes completely filled without either player achieving five in a row, the game ends in a draw.

The seemingly straightforward rules mask the strategic depth of the game, where players must balance offensive maneuvers to build their own lines while defensively blocking their opponent's progress.

1.3 Overview of the Assignmen

This assignment tasks students with implementing the Five in a Row game in MIPS assembly language using the MARS simulator. The implementation must adhere to specific requirements that simulate the standard gameplay experience, including:

- Displaying a 15×15 game board in the terminal interface.
- Handling player input for coordinates in the form of x,y (ranging from 0 to 14).
- Validating user inputs and providing appropriate error messages for incorrect entries.
- Updating and displaying the board after each valid move.
- Determining win conditions by checking for five consecutive markers horizontally, vertically, or diagonally.
- Detecting draw conditions when the board is full without a winner.
- Outputting the final game state and result to a text file named "result.txt".

The assignment serves as a practical application of computer architecture concepts while recreating a timeless strategic game.

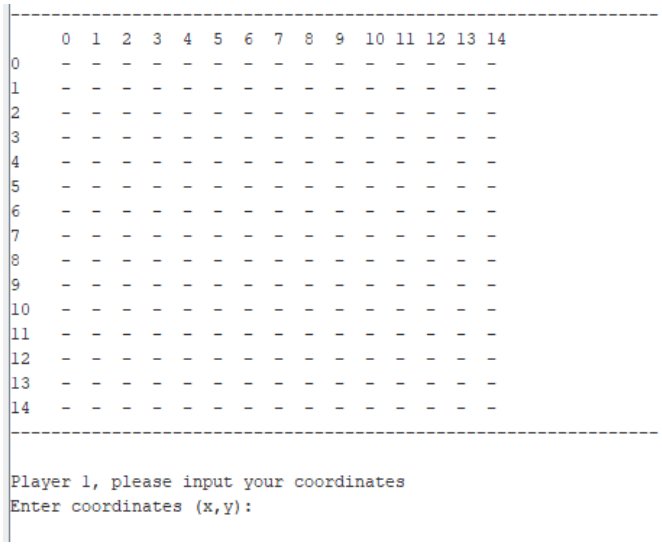
2 Implementation

2.1 Interface design

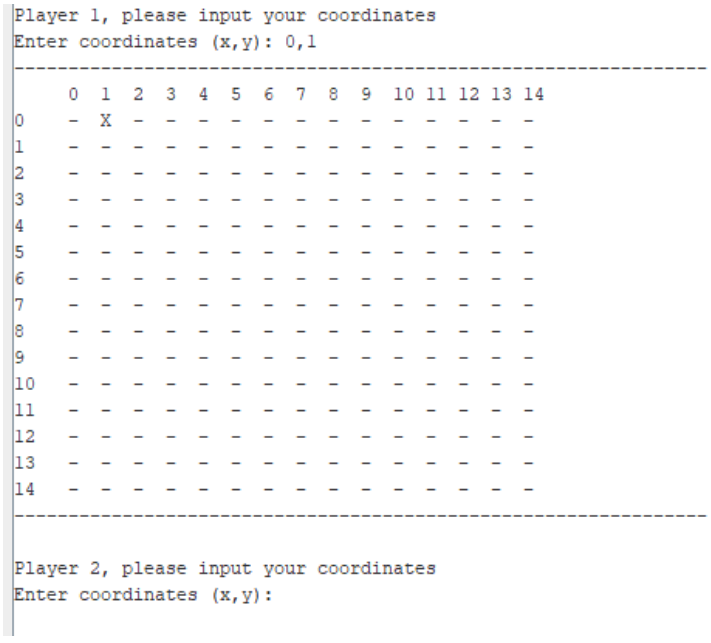
To make it easier for players to play and see the next moves, our board is represented as follows:

- Numbered from 0 to 14 representing row and column order
- Empty cells will be marked with the character "-" , and the girls that have been played by the player will be marked with the playerSymbol ("X" for player 1 and "O" for player 2)
- In addition, at the top and bottom of the board I put a series of characters "-" for the purpose of easily distinguishing the board area and the coordinates input area

So, our standard board will look like the image below:



Below the board is the area where players enter coordinates. When it is their turn, both players must enter coordinates in the form "x,y" where x is the row and y is the column. For example, if player 1 enters "0,1", "X" will be entered in row 0, column 1 of the board:



Then, we continue with player 2.

2.2 Input processing

As mentioned above, the syntax of the coordinates when the player enters is "x,y" with x being the row and y being the column. However, there will be times when the player accidentally or intentionally enters the wrong syntax. Therefore, we have to check the validity after the player enters the coordinates.

The check structure is as follows:

- Check if the cell has been moved. We check by looking at the coordinates the player enters to see if they are equal to "-", if so, replace "-" with the player symbol, otherwise, it proves that the cell has been played by the player, so it returns "This cell is already occupied" and asks the player to re-enter.

```
Player 1, please input your coordinates
Enter coordinates (x,y): 0,1
------------------------------------------------------------
    0  1  2  3  4  5  6  7  8  9  10 11 12 13 14
0   -  X  -  -  -  -  -  -  -  -  -  -  -  -  -
1   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
2   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
3   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
4   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
5   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
6   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
7   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
8   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
9   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
10  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
11  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
12  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
13  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
14  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
------------------------------------------------------------

Player 2, please input your coordinates
Enter coordinates (x,y): 0,1

This cell is already occupied. Please choose another one.
Enter coordinates (x,y):
```

- Check if the cell is outside the table area. The size of the table will be 15x15, so the x and y coordinates entered by the player are limited from 0 to 14, if larger than 14 or smaller than 0, it will return "Invalid input" and ask the player to re-enter.

```
Player 1, please input your coordinates
Enter coordinates (x,y): -1,4

Invalid input. Please try again.
Enter coordinates (x,y): 1,15

Invalid input. Please try again.
Enter coordinates (x,y): -1,16

Invalid input. Please try again.
Enter coordinates (x,y): 16,1

Invalid input. Please try again.
Enter coordinates (x,y): 5,-19

Invalid input. Please try again.
Enter coordinates (x,y):
```

- Check if there are any letters or spaces. Similarly, if there are, return "Invalid input" and ask the player to re-enter.

```
Player 1, please input your coordinates
Enter coordinates (x,y): a

Invalid input. Please try again.
Enter coordinates (x,y): a,v

Invalid input. Please try again.
Enter coordinates (x,y):  0,1

Invalid input. Please try again.
Enter coordinates (x,y): 0,1

Invalid input. Please try again.
Enter coordinates (x,y): 0 , 1

Invalid input. Please try again.
Enter coordinates (x,y): 1,a

Invalid input. Please try again.
Enter coordinates (x,y): 0, 9

Invalid input. Please try again.
Enter coordinates (x,y):
```

After checking the valid input of the player, the table will be updated based on what the player has entered.

2.3 Overview of the main algorithm

Coming to the main algorithm, our game must have check win and tie.

Regarding to **check win**. Firt, recall the winning condition, that is the objective is to be the first to form an unbroken line of five markers of one's own symbol horizontally, vertically, or diagonally. Thus, we will have 3 algorithms to check the win of a player: horizontal, vertical, and diagonal:

- With horizontal win, after each player's move, the algorithm will go through all horizontal rows on the board. We have a temporary variable, tentatively called countwin to count the number of symbols other than "-" (because I consider it a cell that has not been moved), if countwin is equal to 5, we will know there is a winner. Otherwise, the game continues. Example below:

```
------------------------------------------------------------
    0  1  2  3  4  5  6  7  8  9  10 11 12 13 14
0   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
1   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
2   -  -  -  -  O  -  -  -  -  -  -  -  -  -  -
3   -  -  -  X  X  X  X  X  -  -  -  -  -  -  -
4   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
5   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
6   -  -  -  -  -  -  -  O  -  -  -  -  -  -  -
7   -  -  -  -  -  -  -  -  O  -  -  -  -  -  -
8   -  -  -  -  -  -  -  -  -  O  -  -  -  -  -
9   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
10  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
11  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
12  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
13  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
14  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
------------------------------------------------------------

Player 1 wins

-- program is finished running --
```

- Similarly to a horizontal win, a vertical win will go through all columns of the board to count the number of symbols appearing consecutively on a vertical row, thereby detecting the winner. Example below:

```
------------------------------------------------------------
     0  1  2  3  4  5  6  7  8  9  10 11 12 13 14
0    -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
1    -  -  X  -  X  -  X  -  -  -  -  -  -  -  -
2    -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
3    -  -  -  X  -  -  -  -  -  -  -  -  -  -  -
4    -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
5    -  -  -  -  -  -  O  -  -  -  -  -  -  -  -
6    -  -  -  -  -  -  O  -  -  -  -  -  -  -  -
7    -  -  -  -  -  -  O  -  -  -  -  -  -  -  -
8    -  -  -  -  -  -  O  -  -  -  -  -  -  -  -
9    -  -  -  -  -  -  O  -  -  -  -  -  -  -  -
10   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
11   -  -  -  -  -  -  -  -  -  -  -  -  -  -  X
12   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
13   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
14   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
------------------------------------------------------------

Player 2 wins

-- program is finished running --
```

- Similarly to the above two wins, the diagonal win will also go through the diagonal lines in the table to find the winner. However, this algorithm must be divided into two other small algorithms, which are the diagonal line to the left and the diagonal line to the right:
  - Diagonal line to the left, we check the diagonal lines towards the left and if we get five player symbols in a row, we will announce the win. Example below:

```
------------------------------------------------------------------
     0  1  2  3  4  5  6  7  8  9  10 11 12 13 14
0    -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
1    -  X  -  -  X  -  X  -  -  -  -  -  -  -  -
2    -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
3    -  -  -  -  -  -  -  -  -  -  -  O  -  -  -
4    -  -  -  -  -  -  -  -  -  -  O  -  -  -  -
5    -  -  -  -  -  -  -  -  -  O  -  -  -  -  -
6    -  -  -  -  -  -  -  -  O  -  -  -  -  -  -
7    -  -  -  -  -  -  -  O  -  -  -  -  -  -  -
8    -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
9    -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
10   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
11   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
12   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
13   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
14   -  -  -  -  -  -  -  -  -  -  -  -  -  X  X
------------------------------------------------------------------

Player 2 wins

-- program is finished running --
```

– Same as diagonal line to the right. Example below:

```
---------------------------------------------------------------
     0  1  2  3  4  5  6  7  8  9  10 11 12 13 14
0    -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
1    -  -  -  -  -  -  O  O  O  -  -  -  -  -  -
2    -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
3    -  -  -  X  -  -  -  -  -  -  -  -  -  -  -
4    -  -  -  -  X  -  -  -  -  -  -  -  -  -  -
5    -  -  -  -  -  X  -  -  -  -  -  -  -  -  -
6    -  -  -  -  -  -  X  O  -  -  -  -  -  -  -
7    -  -  -  -  -  -  -  X  -  -  -  -  -  -  -
8    -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
9    -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
10   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
11   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
12   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
13   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
14   -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
---------------------------------------------------------------

Player 1 wins

-- program is finished running --
```

In the case of a **draw**, the condition is if the board becomes completely filled without either player achieving five in a row, the game ends in a draw.

2.4 Output deployment

After finding the winner or the result is a tie, the final table and the decision of who wins will be filled in the result.txt file.

Implementation method:

- Initially, we save the address of the final result in a temporary variable.
- Then, call the function that opens the result.txt file in write mode
- Next, we create a loop so that we can add each address of the table to the write file.
- Next, we have to write the final result (winner or tie) into that file. The process is similar, but this time the file is opened in add mode to add the result line.
- There is a loop to calculate the total length of the output result, for example "player 1 wins" has a length of 13 to serve the purpose of printing the message to the file.
- Finally, write that message to the file by calling the address of the message function, then print the characters until the length of the message calculated above is exhausted.

3 Results and Testing

3.1 Results in MIPS

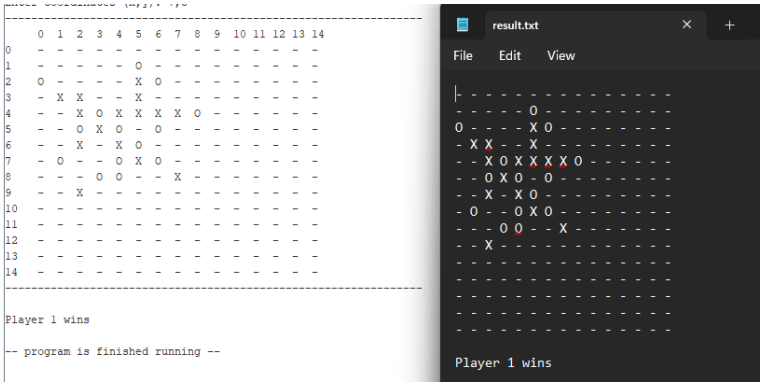
Based on what was reported in Section 2, when summarized, our MIPS code will work as follows:

- Initialize the board with size 15x15 with the character "-" representing the empty square. Player 1 will go first with "X" and Player 2 will be "O".
- Main loop:
  - The game runs in a loop, alternating between Player 1 (X) and Player 2 (O), after each player plays, the program will check, process the next input, update and display the new board on the screen.
  - After each time the player gives a valid input, in addition to updating the board, the code will check for victory. For example, if Player 1 plays "X" at a certain coordinate, the program will check four directions (horizontal, vertical, diagonal left, diagonal right) and return 1 (win) if it finds 5 consecutive characters. Similar to player 2.
  - If there is a winning player, the program will display "Player ... wins", exit the loop and go to the next step. If there is not, return to the loop and continue.
  - In case of a tie, the code will display the message "Tie", end the loop and go to the next step.
- The final step, save the result to the result.txt file.

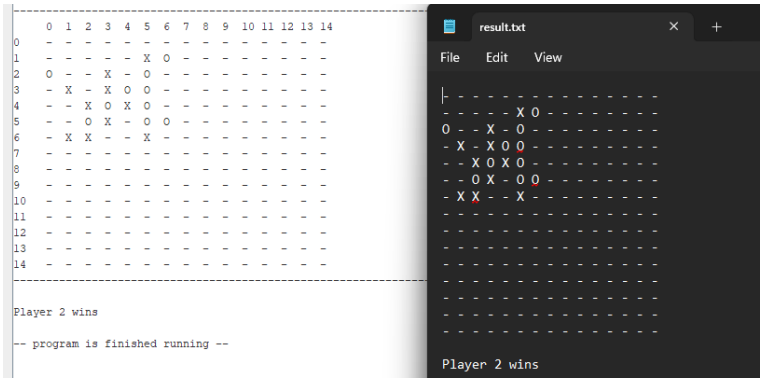
3.2 Testing

To test the game, we organized a game with our team members, here are the results after playing.

When **player 1** wins:



When **player 2** wins:



When 2 players are **tie**:

```
------------------------------------------------------------------------
     0  1  2  3  4  5  6  7  8  9  10 11 12 13 14
0    X  O  X  O  X  O  X  O  X  O  X  O  X  O  X
1    O  X  O  X  O  X  O  X  O  X  O  X  O  X  O
2    X  O  X  O  X  O  X  O  X  O  X  O  X  O  X
3    O  X  O  X  O  X  O  X  O  X  O  X  O  X  O
4    O  X  O  X  O  X  O  X  O  X  O  X  O  X  O
5    X  O  X  O  X  O  X  O  X  O  X  O  X  O  X
6    X  O  X  O  X  O  X  O  X  O  X  O  X  O  X
7    O  X  O  X  O  X  O  X  O  X  O  X  O  X  O
8    X  O  X  O  X  O  X  O  X  O  X  O  X  O  X
9    O  X  O  X  O  X  O  X  O  X  O  X  O  X  O
10   O  X  O  X  O  X  O  X  O  X  O  X  O  X  O
11   X  O  X  O  X  O  X  O  X  O  X  O  X  O  X
12   X  O  X  O  X  O  X  O  X  O  X  O  X  O  X
13   O  X  O  X  O  X  O  X  O  X  O  X  O  X  O
14   X  O  X  O  X  O  X  O  X  O  X  O  X  O  X
------------------------------------------------------------------------

Tie

-- program is finished running --
```

4 Conclusion

4.1 Summary of processes

In conclusion, in this assignment, the group has successfully implemented the Five in a Row (Gomoku/Caro) game using MIPS assembly language on the MARS simulation environment. We have fully implemented the requirements of the assignment including:

- Designing an intuitive interface with a 15x15 chessboard.
- Building an input processing system, checking the validity of the player's input coordinates with many different conditions: the square has been played, the coordinates are out of the chessboard, and the input format is invalid.
- Developing a winning check algorithm, including checking five consecutive chess pieces horizontally, vertically, and diagonally.
- Implementing a tie detection mechanism when the chessboard is full without a winner.
- Exporting the final game results to the file "result.txt".

The result is a relatively stable program that allows two players to take turns hitting "X"s and "O"s on the board, with appropriate notifications and an intuitive interface that makes it easy for users to follow the progress of the game.

4.2 Learning outcomes

Through this project, our group has achieved important learning outcomes:

1. Developing MIPS Assembly programming skills on MAR4.5.
2. Applying knowledge of computer architecture
3. Developing algorithm design skills
4. Practicing debugging and testing skills
5. Enhancing teamwork skills
6. Enhancing report writing skills

This assignment not only helps us apply theoretical knowledge into practice but also creates a solid foundation for subsequent subjects and assignments in the field of computer architecture.

References

[1] Wikipedia, "Gomoku." [Online]. Available: https://en.wikipedia.org/wiki/Gomoku

[2] Wikipedia, "Cờ ca-rô." [Online]. Available: https://vi.wikipedia.org/wiki/C%E1%BB%9D_ca-r%C3%B4

[3] Phạm Quốc Cường, *Kiến Trúc Máy Tính*, Nhà xuất bản Đại học Quốc gia Thành phố Hồ Chí Minh

[4] Internal documents of Computer Architecture course, Ho Chi Minh City University of Technology