VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

COMPUTER ARCHITECTURE LAB (CO2008)

Assignment

Battleship

Advisor: Dr. Phạm Quốc Cường
Students: Đỗ Hoàng Quân - Student ID: 2212779.

HO CHI MINH CITY, DECEMBER 2023

Contents

1 Introduction of Battleship

Figure 1: Example of the battleship game in real life

Battleship, also recognized as Battleships or Sea Battle, is a strategy guessing game designed for two players. The game is played on ruled grids (paper or board), where each player places his/her fleet of warships. The coordinate of a player's fleet is hidden from the other player. The main action of the game involves players taking turns to make calculated shots, trying to aim to the concealed locations of the opponent's ship. The ultimate goal is to successfully destroy the opponent's entire fleet through deduction and tactical gameplay. The detailed game rules can be found in [1], here in this report, I will present the modified rule applied for the assignment:

- **Phase 1: Setup the phase**

  First, each player sets up his/her fleet on his/her own 7×7 grid map. Each player will have a fleet of 6 ships, including 3 2×1 ships, 2 3×1 ships, and 1 4×1 ships. A ship location is indicated by the coordinates of the bow and the stern of the ship $(row_{bow}, row_{stern}, column_{bow}, column_{stern})$, and the coordinates are all integers in range 0 to 6. The players will need to setup the exact amount of ships with the same size to complete the setup phase. Also note that the ships cannot overlap with each other. Figure 2 represents an acceptable defined fleet of ships. In Figure 2, the location of the fleet, follow $(row_{bow}, row_{stern}, column_{bow}, column_{stern})$ respectively, are: the coordinates of the yellow 4x1 ship is "0 0 0 3"; the blue 2x1 ships are "1 4 2 4", "2 1 3 1", and "4 4 4 5"; the green 3x1 ships are "5 3 5 5" and "6 0 6 2".

1	1	1	1	0	0	0
0	0	0	0	1	0	0
0	1	0	0	1	0	0
0	1	0	0	0	0	0
0	0	0	0	1	1	0
0	0	0	1	1	1	0
1	1	1	0	0	0	0

Figure 2: Example of a fleet of ships that a player can set up in this game

- **Phase 2: Targeting each other**

  After the setup phase, 2 players will **take turn** targeting a box on the other player's map blindly. The values of the cells of the map will be either **1** or **0**. 1 means that the box is **occupied by a ship**, and 0 means the box is not occupied.

  If the attack hits a cell that is being occupied by a ship, we say the play **"HIT"**. Otherwise, the player **"MISS"**. Also, the player have to remember the cells that have been hit on their own. Value of a cell will change to **0** if it got accurate attack from opponent. A ship is completely destroyed if all the cells that ship is occupying is hit by the opponent, and a player will lose if his/her ships are all destroyed first. In other words, the game is over when a player's map contains **only 0's**.

2 Idea and algorithm

This semester, my assignment for Computer Architecture Lab requires implementing the game Battleship in MARS using the assembly language. This report is about the detailed idea and steps for my assignment, and now I will try to present them as clearly as possible:

- **Step 1: Basic Rules & Instructions**

  The program print out the basic rules, instructions and some notices of the Battleship game and the interaction between the players and the MARS program. Also, the program print out details about some additional operations that the players can use during the interactive process.

- **Step 2: Set up maps**

  The program set up 2 grid maps for 2 players, each map using an array with size of 49. I don't use 2-dimensional array because I feel it is pretty hard to use, and if I use 1 dimensional array, I can still access into the data of any cells. For instance, if a cell has coordinate of row i and column j, data of that cell will be in the position $7i + j$ in the array. Also, set up for 2 maps initially just contain 0's in all cells.

- **Step 3: Set up Player 1's ships**

  The program asks the Player 1 to insert the coordinate for his/her fleet of ships, and terminates when Player 1 finished setting up his/her ships. The player will first insert the 4×1 ship, than 2 3×1 ships, and last 3 2×1 ships. With each ship, the program ask the player to insert 4 coordinates: row of bow, row of stern, column of bow, and column of stern, respectively. With each coordinate, the player must insert an integer in range 0 to 6, if the player insert a character which is not an integer, or an integer but greater than 6, the program will inform the fault "Wrong input", and the player will have to insert again, until he/she inserts the valid coordinate. Additionally, if the player inserts the ship that has the wrong size with the requirement, the program will inform the fault "Wrong size", and the player will have to re-insert the ship. The instruction and requirement will be print out, depends on the kind of the ship and the kind of coordinate, and the player should follow the instructions to insert the correct type of ship and coordinate.

  Also, I add some extra operations: If the player wants to re-insert a coordinate, he/she should insert the character 'R', that will let him re-insert the previous kind of coordinate (the order of the coordinates is row of bow, row of stern, column of bow, and column of stern); also, if the player wants to break out and re-insert all the coordinates of the ships, he/she just need to insert the character 'E' at any time. However, notice that the player cannot re-insert the coordinates of a ship, if he/she has begun to insert the coordinate for the next ship (Or the player could only re-insert the coordinate of a ship, while he/she is still inserting for that ship). Also, Player 1 cannot re-insert for his fleet if Player 2 has begun to set up the ships.

- **Step 4: Check ship overlap & Update cells data & Print map of Player 1**

  Each time after Player 1 has inserted the valid coordinates of a ship, and the ship has the correct size, the program will check whether the ship overlap with other ships or not. It does this by iterating through all the cells of this ship, if any cell of those is being occupied by an other ship (data = 1), the program will inform fault "Ship overlap", and the player will have to re-insert this ship (Go to **Step 3**). Note that the program access to a cell using the way described in **Step 2**.

Otherwise, the ship does not overlap, then the program will update the cells that are occupied by this ship (Update data to 1). After that, if player 1 hasn't finished setting up the ships, the program go back to **Step 3**. Else the program will finish setting up for Player 1, and inform to the screen. After Player 1 insert a valid ship, and the map has been updated, the program will print out Player 1's map, so that he/she will notice about the location of the ships and sets up the ships as the player wants.

- **Step 5: Set up Player 2's ships**

  Completely the same, the program ask Player 2 to set up the ships. The instructions, requirement and the program is the same as Player 1. The map of Player 2 is also printed after each time Player 2 inserts a valid ship.

- **Step 6: Targeting each other ships**

  When we reach this step, the program has finished setting up 2 grid maps of ships for Player 1 and Player 2. Next, the program informs that 2 players start targeting to each other's map, and also each player has 16 cells with data 1 on his/her map. The program takes turn to ask the players: It ask Player 1 in odd turns, and ask Player 2 in even turns. At the start of each player's turn, the program will print out the map of that player, so that he/she will concern about the progress of the opponent. Than with each turn, the program ask the player to first insert the row of the cell, than insert the column of the cell that the player wants to target at the opponent's map. With each coordinate, the player must insert an integer in range 0 to 6, if the player insert a character which is not an integer, or an integer but greater than 6, the program will inform the fault "Wrong input", and the player will have to insert again, until he/she inserts the valid coordinate.

  After the player has inserted the valid coordinate, the program will access to the location of that cell in the opponent's map. If the data of the cell is 1, the program will inform "**HIT!**"to the screen, than update the value of the cell to 0, and decrease 1 from the opponent's number of cells occupied by a ship of the opponent. Otherwise, data of the cell is 0, the program will inform "**MISS!**"to the screen (my addition). Note that the program access to the cell using the way described in **Step 2**. Also, the program will write the moves of the players into 2 separate text files (each file has the moves of 1 player) for ease of review.

- **Step 7: Informs the winner**

  In each turn, after the player has targeted the opponent's ship and the program has finished updating the cells, the program will check whether the opponent's number of cells with data 1 is equal to 0 or not. If it is greater than 0, the program goes back to **Step 6**, and the players go on aiming to each other. Else, the program will inform the winner: If this is Player 1's turn than Player 1 wins, else Player 2 wins. The program than terminates.

3 Conclusion

In this report, I have presented the algorithm I used to implement the assignment **Battleship** of Computer Architecture Lab (CO2008) this semester in MIPS assembly. Conclusion, Battleship is a strategic and challenging game for players who wants to practice their deduction skill. Also, the assignment for Computer Architecture Lab this year is hard and challenging too, and I'm happy that I could have finished it well, and I want to give thanks to Dr. Phạm Quốc Cường for helping me with the assignments. With the experience of accomplishing the assignment of this subject, I believe I can do well in the future subject's assignment.

References

[1] Wikipedia, "Battleship (game)." [Online]. Available: https://en.wikipedia.org/wiki/Battleship_(game)