Ho Chi Minh City University of Technology

Faculty of Computer Science and Engineering
Course: Computer Architecture Lab (CO2007)

Assignment
Battleship
Instructor: Bang Ngoc Bao Tam

Student: Nguyen Hong Phuc
ID: 2252639
LAB-Group: CC07
Academic year: 2023-2024

Ho Chi Minh City, December 2023

I.OVERVIEW

GAMEPLAY:

Battleship is a board game that necessitates strategic planning and deductive reasoning from its two participants.

1. In the conventional format, each participant arranges a collection of battleships of varying dimensions on a grid that measures 10x10. A typical fleet might comprise five 2x1 ships, three 3x1 ships, and one 5x1 ship.

2. Following the positioning of the ships, the game advances in a series of rounds. During each round, players alternate in selecting a square on the adversary's grid to target. The adversary then discloses whether the chosen square is occupied by a ship. If a ship is struck, the impacted player denotes this on their personal grid and announces the type of ship that was hit.

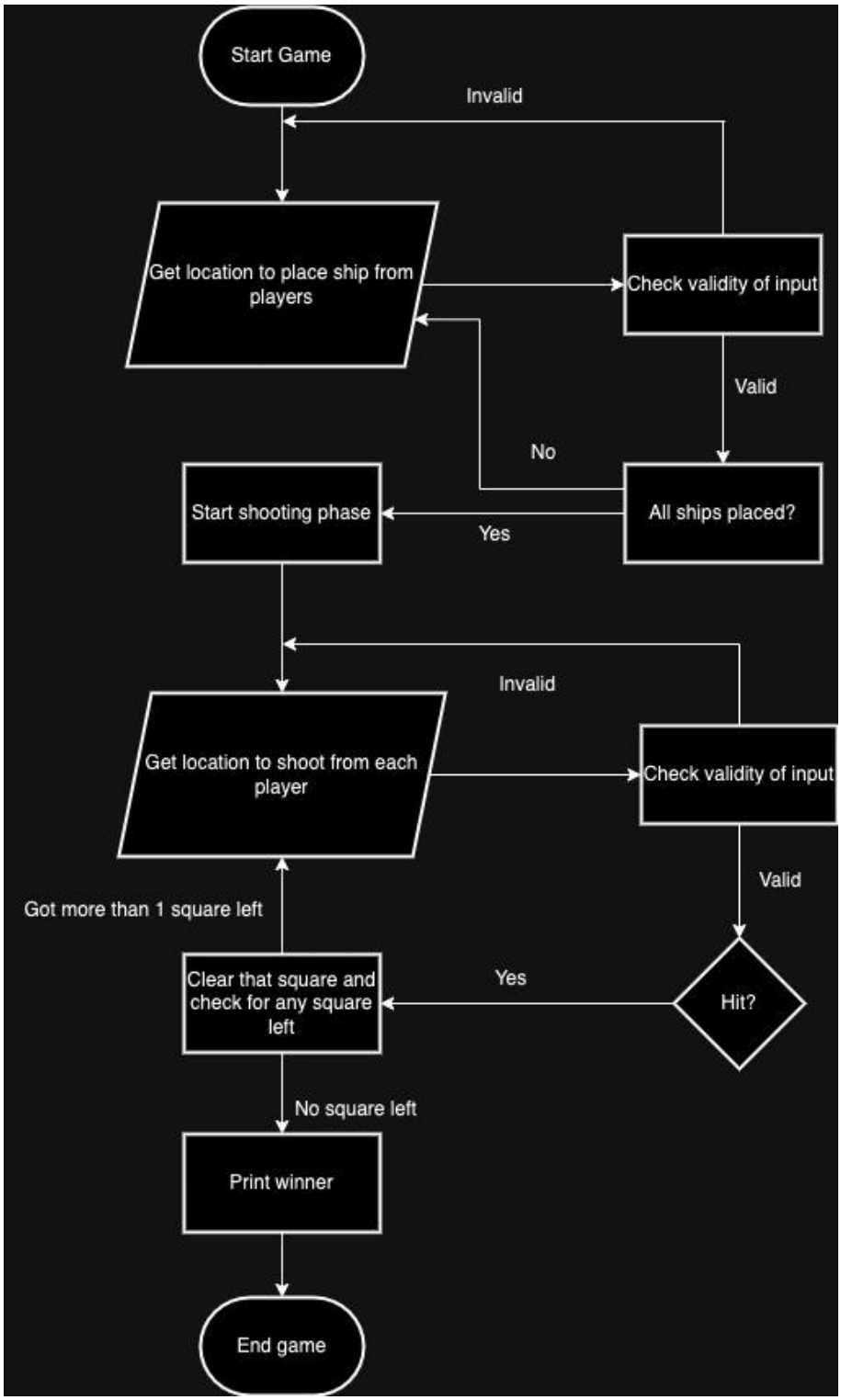
3. The game concludes when a player has no remaining unsunk ships, resulting in the adversary being declared the victor.

For this assignment, the game is adapted to a smaller 7x7 grid. Each player's fleet consists of six ships: three small, two medium, and one large. The placement of ships is restricted to either the vertical or horizontal axis.

The game is implemented using MIPS Assembly Language, a textual, human-readable version of MIPS Machine Language. Mastery of MIPS Assembly Language facilitates a profound understanding of the underlying operations of a machine.

II. EXPLANATION

FLOW CHART.



DATA STORAGE

In this implementation, two 49-word arrays are used to represent a 7x7 game board for each player in a Battleship game. Each array element corresponds to a square on the game board, with a value of 0 indicating an empty square and a value of 1 indicating a square occupied by a ship.

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Players are asked to input the bow (front) and stern (back) coordinates of their ships in the format: row-bow, col-bow, row-stern, col-stern.

The program checks if the input is valid (i.e., within the range of 0 to 6). If the input is invalid, the player is prompted to input again.

The program also checks if the ship's placement would overlap with any existing ships on the board. If there is an overlap, the player is asked to input the coordinates again.

Once a valid and non-overlapping placement is determined, the corresponding elements in the array are updated to 1, indicating that those squares are now occupied by a ship.

The arrays are accessed in a row-major order, meaning that the elements in the same row of the array are stored in consecutive memory locations. The location of a specific element in the array is calculated using the formula: Base address + offset of data. Since the

data type is a word (4 bytes), the offset of data is calculated as: (row * 7 * 4 + col * 4)

SHOOTING PHASE

Upon successful placement of all ships by both players, the game transitions into the shooting phase. In this phase, players are prompted to specify the coordinates they wish to target. The program validates the input to ensure the coordinates are within the acceptable range (0 to 6 for both row and column).

The program then checks the corresponding square on the opponent's board. If a ship occupies the square, the program outputs "HIT"; otherwise, it outputs "MISS". The targeted squares are marked with a value of -1 in the array, indicating they have been shot at. If a player attempts to target a square that has already been shot at, the program will output a message indicating that the square has already been targeted and prompt the player to input a new set of coordinates.

Each player has a total of 16 squares occupied by ships (three 2x1 ships, two 3x1 ships, and one 4x1 ship). A counter is used to track the number of squares occupied by ships that have not been hit. When a player's counter reaches zero, indicating all their ships have been sunk, the program outputs a congratulatory message to the other player for their victory

FUNCTION EXPLANATION(For some specific function)

The function **Print_Matrix1:** is designed to display the game board of the first player. It employs a counter to ensure that each line printed contains exactly seven elements, corresponding to the seven columns of the 7x7 game board.
The function **Getinput1:** is responsible for managing the ship placement phase for the first player. It utilizes a counter that, upon reaching six, signifies the completion of the ship placement phase for

that player. The function ensures that the first input coordinate is always greater than the second by employing two helper functions, **swaprow** and **swapcol**, which swap the inputs if the first is less than the second.
The function also maintains three variables to track the number of large, medium, and small ships. This allows the player to choose the type of ship they wish to place, rather than being forced to place a specific type. If a type of ship has been placed the maximum number of times, the function informs the player and prompts them to choose a different type of ship. This continues until the counter reaches six, indicating that all six ships have been placed.
The function checks the orientation of the ship based on the input coordinates. If the bow and stern share the same column, the ship is placed vertically; if they share the same row, the ship is placed horizontally. Diagonal placement is not allowed.
The function also checks the size of the ship by subtracting the smaller coordinate from the larger one. As the largest ship is 4x1, the result of this subtraction should be within the range of 0 to 3. If it falls outside this range, an error is printed. The same process is repeated for the second player

CONCLUSION

In conclusion, this report has detailed the process of implementing a simplified version of the Battleship game using MIPS assembly language. The game, played on a 7x7 grid, involves strategic planning and deductive reasoning from its two participants. The implementation involved creating functions for initializing the game board, placing the ships, and managing the gameplay.
The project provided a practical application of MIPS Assembly Language, facilitating a deeper understanding of the underlying operations of a machine. It demonstrated how a complex game can be broken down into simpler components and implemented using basic programming concepts.
I would like to express my gratitude to the instructor Bang Ngoc Bao Tam for providing the opportunity to work on this engaging project. The experience has been invaluable in enhancing my understanding of

assembly language programming and its applications. I look forward to applying the knowledge and skills gained from this project in future endeavors.