VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

COMPUTER ARCHITECTURE (LAB) – CO2008

ASSIGNMENT REPORT

BATTLE SHIP

Student: Nguyễn Thiên Lộc

ID: 2252460

Class: CC07

Semester: 231

Lecturer: Băng Ngọc Băng Tâm

Hồ Chí Minh, 5/12/2023

CONTENTS

I. INTRODUCTION

Battleship (also known as Battle ships or Sea Battle) is a strategy type guessing game for two players. It is played one ruled grids (paper or board) on which each player's fleet of warships are makred. The locations of the fleets are concealed from the other player. Players alternate turn calling "shots" at the other player's ships, and the objective of the game is to destroy the opposing player's fleet.

Picture 1. Battleship board

In this project, instead of 10x10 grid like usual, we use 7x7. The tutorial is:

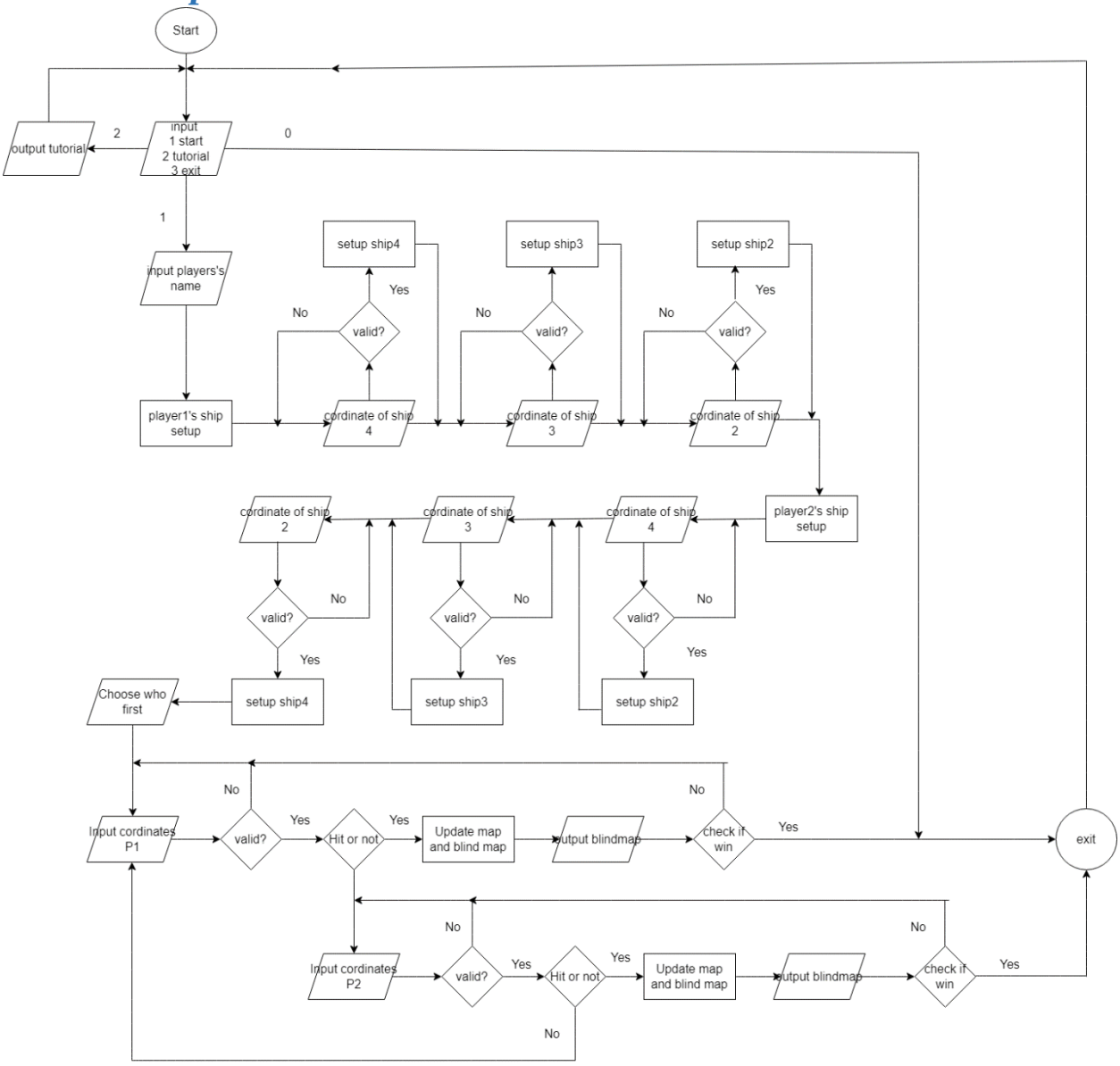
1. Each player sets up a fleet of battleships on their map. A fleet must contain a predefined set of a battle ships with different sizes. In this case, a fleet of ships must consist of 1 4x1 ship, 2 3x1 ships and 3 2x1 ships.

2. After player's setup phase, the game starts in a series of rounds. Every rounds, each player choose which place they want to shoot at. The system will annouce whether or not the place is occupied by a ship. If it "hit", a blind map will show up which place had chosen or "miss".

3. After several rounds, if there is no more player's ships than the opponent wins.

II. IDEA

1. *Game process*



Flowchart1. Game process

2. *Create map*

First, Each player need 2 map to storage their ships, but instead of using 2D-arrays, 1D-array was a better idea at that time. However, while testing the game, we found some difficults when memorize which place had been chosen. So we create 2 more maps which name is blind-map, help players easier to remember their move. Using loop to create map with memory 196 (7x7 box) and store 0 in it box. The map is look like this:

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

Table1. Map

3. *Menu*

In this step, the players can choose that start the game, read tutorial or exit the game by in put number from 0 to 2.

```
Press 1 to start the game
Press 2 to read tutorial
Press 0 to exit
Your seclection:
```

Picture 2. menu

We can know more how to play when input 2 in *Your selection:*

```
1. Each player sets up a fleet of battle ships on their map (7x7). A fleet of ships consists of 3 2x1 ships, 2 3x1 ships and 1 4x1 ships.
   A map look like this:
        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
2. After the ships have been postioned, the game proceeds in a series of rounds. In each round, each player takes aturn to choose a place to shoot.
   Computuer will annouces if its correct or not by sending message HIT
3. If all ships of players have been sunk, the game is over and their opponent wins.
```

Picture 3. Tutorial

4. Input

We need four values to have a ship in map which were (row_{bow}, $column_{bow}$, row_{stern}, $column_{stern}$).

After that we caculate theirs index using row major order:

$$Index = Row*M+Column$$

which M is numbers of rows.

Beside caculating, the system will check if values valid or not by checking:

1. Is it suitable for a 4 ships, 3 ships,2 ships or not?
2. Is it horizontal or vertical or not?
3. Is the place which player choose have ships or not?
4. Are the values lower than 0 or larger than 6?

Each ship, systeam will show the map include their ships to the player. If values are not valid then the program return to the top of the input. Ex: if you input ship 4 then return to setupship4, input ship2 then return to setupshup2,….

Then computer will update the data of ships by changing it to 1 in map

```
0       1       2       3       4       5       6
-----------------------------------------------
1       1       1       1       0       0       0  |  0
0       0       0       0       0       0       0  |  1
0       0       0       0       0       0       0  |  2
0       0       0       0       0       0       0  |  3
0       0       0       0       0       0       0  |  4
0       0       0       0       0       0       0  |  5
0       0       0       0       0       0       0  |  6
```

Picture 4. Map

5. Playing

The players are free to choose who play first. When the players choose a place, program check if its valid or not. If not, it will ask the player to input again. After that show them sign “HIT” and blind map of opponent if the player are correct or “Miss” if incorrect.

In blindmap will change to -1 whether its correct or incorrect. But if the ships was hit, their place in map will update to 0

System also check if all of their opponent's ships are sunk or not to announce the winner of the game.

```
Choose where to shoot (row): 0
Choose where to shoot (column: 0
HITTTTTTTTTTTT
          0       1       2       3       4       5       6
          ------------------------------------------------
          -1      0       0       0       0       0       0 |  0
          0       0       0       0       0       0       0 |  1
          0       0       0       0       0       0       0 |  2
          0       0       0       0       0       0       0 |  3
          0       0       0       0       0       0       0 |  4
          0       0       0       0       0       0       0 |  5
          0       0       0       0       0       0       0 |  6
```

Picture 5. Blind map

6. Tracking list

We use 6 registers to keep tracking the game from the beginning: 2 for save who player 1 and who is player2. 1 for storing ArrayMovement, 1 for count and last one for store into array.

Each round, we adding player's into the ArrayMovement so when the game finish, we create a text.txt then write full of the movements into

```
Tracking moves:
Turn of 1

 0 0
 0 1
 0 2
 0 3
 6 0
 6 2
 6 1
 5 3
 5 4
 5 5
 1 4
 2 4
 2 1
 3 1
 4 4
 4 5
```

Picture 6. Trackinglist

III. CONCLUSION

This document details the process involved in creating a simplified version of the Battleship game using MIPS assembly language. Played on a 7x7 grid, the game demands strategic thinking and logical deduction from its players. Development steps encompassed crafting functions to set up the game board, position ships, and manage gameplay.

This project was a practical application of MIPS Assembly Language, fostering a deeper understanding of fundamental machine operations. It exemplified the breakdown of a complex game into basic elements, built upon core programming principles.

I express sincere gratitude to instructor Bang Ngoc Bao Tam for providing the opportunity to embark on this engaging project. The experience has been invaluable in enhancing my understanding of assembly language programming. I'm excited to apply the knowledge gained in future endeavors.