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

Assignment of

COMPUTER ARCHITECTURE

Lecturer: Phạm Quốc Cường

Student name: Nguyễn Tiến Hưng-2252280

HO CHI MINH CITY, November 2023

TABLE OF CONTENTS

PREFACE

Computer Architecture is an important general major subject for computer science and computer engineering students HCMUT in particular. Therefore, devoting a certain amount of time to this subject and practice is indispensable to help students have a solid basis in Computer Science and Engineering and make a premise to study well in other subjects in the training program.

During the process of writing the above essay, I have received much care and support, dedicated help from teachers. In addition,I would also like to express my most sincere gratitude to Mr. Pham Quoc Cuong, is the instructor for this topic. Thanks to his wholeheartedly instructing, the group was able to complete the essay on time and solve problems well. Your guidance has been the guideline for all actions of mine and maximized the supportive relationship between teachers and student in the educational environment. Finally, once again, I would like to express my deep gratitude to the teacher. This is the belief, a great source of motivation for me that this result can be achieved.

1 Introduction:

1.1. The game and summery of this report:

Battleship is a strategic board game between 2 players. It focuses on the process of planning deducting skill of both players. The assignment **requires** us to design and **write MIPS assembly language** for implementing a *Table and allow players to play Battleship game*.

This report is to explain the idea to implement this assignment.

1.2. Content:

In this project, we have to replicate the battleship game in MIPS assembly language. Since the resource is limited and to keep things simple, we will work with a smaller grid size which is **7x7**. Please create a program that does the following:

- A note before entering the program: A ship location is indicated by the coordinates of the bow and the stern of the ship (row_{bow}, $column_{bow}$, row_{stern}, $column_{stern}$). In Figure 2, 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”.
- We begin with the preparation phase where the players set up their ship locations. The program should prompt to ask the players to insert the location of the ships. The values of the cells of the map will be either **1** or **0**. 1 means that the box is **occupied by a ship**, otherwise it’s 0. The players will need to setup the exact amount of ships with the same size in order to complete the setup phase. In detail, each player will have 3 **2x1** ships, 2 **3x1** ships, and 1 **4x1** ship. Note that the ships can not overlap with each other. An example of an acceptable arrangement is shown in Figure 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 1: *Example of the the ships that a player can place in this project*

• After the setup phase, both players will **take turn** targeting a box on the other player's map blindly. If the attack hits a cell that is occupied by a ship, an announcement should pop up in the terminal and says "HIT!". This will allow the attacking player to know if they hit an opponent's ship or not. The players will need to remember the cells that have been hit themselves. If a cell got hit, its value should change into **0**, as there it is no longer occupied by a ship. A ship is completely destroyed if all of the cells it's occupying are targeted. A player will lose if all of their ships got destroyed first. In other words, the game will end when a player's board contain **only 0's.**

• Note: Bonus points are provided if you can write out the moves of each player into a separate text file for ease of review.

1.3. Requirements:

1) Friendly interface - 2 points

- Students can design and implement an amicable user interface so that players can play easily without any confusion (2 points).
- Students can design and implement a friendly user interface; however, players face some difficulties when playing the game (1.5 points).
- Students can design and implement a user interface, but it is not friendly, or players need to do several steps for one move (1 point).
- Student can design and implement a user interface, but it fails to allow playing (0.5 points).

2) Application implementation - 6 points

- Students can implement an excellent application without any errors found (5.0 - 6 points).
- Students can implement a good application with some minor errors, but players do not need to restart the application to continue (4.0 - 5.0 points).
- Students can implement the application with some errors that prevent players from playing the game (2.0 - 4.0 point).
- Students cannot implement the application so that players can play/run (0 - 2.0 points).

3) Report - 2 points

- Students write such an excellent report that others can understand without any difficulty (2 points).
- Students write a good report but it's quite simple or lack of some information to clarify the implementation (1.0 - 1.5 points).
- Students write a report with a lot of code embedded without any explanation (0.5 - 1.0 points).
- Students with no report submitted (0 point).

2 Main idea and explanantion:

2.1 Describe the main idea by flowchart

A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

- When the player run the program. The terminal will first print the introduction of the game. Then it continue to print the rule and example of how to playing the game right way.

- Next each player will input the coordinate of the ship to prepare for the game. Notice that every input that is invalid like negative number, out-ranged number, diagonal coordinate, overlap ship… will cause the program to tell that we has just input wrong and need to reinput.

- After setting all the ship. The game begin, each player will choose a coordinate to shoot at their turn.

- The game will end when 1 player has shot all the ship opponent player.

***Note:** The player must strictly follow the rule of the game (which is printed at first when run code).

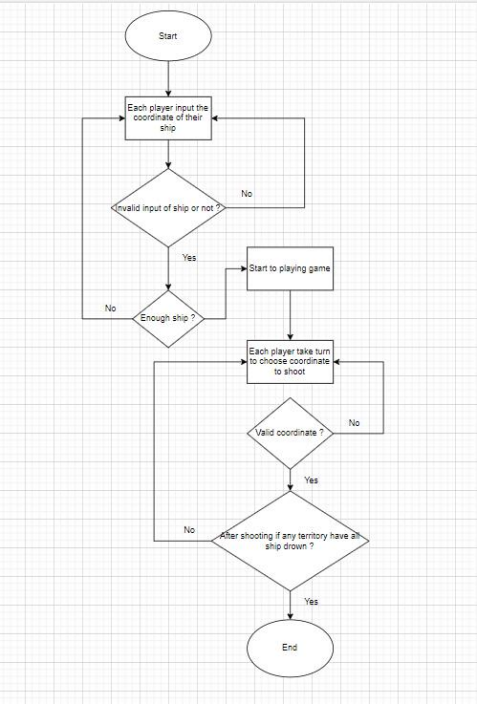

***Figure 2:** Flowchart visualize the main idea of this assignment*

2.2 Checking all the possible invalid of the input

- First we check each number (R1 C1 R2 C2). Each of them must be >=0 and <= 6

- Then we check whether the ship is straight or not. All the input that have $R1 \neq R2$ and $C1 \neq C2$ are invalid input.

- Because the player have to input the ship with respect to the size order. So if player input the wrong size they have to input again.

- And at the end we check the overlap ship means that if any cell of the input has already value is 1 (value 1 denote ship have put there and 0 denote opposite). And if not we start to store 1 to the ship and move to the next instruction.

- Any input get wrong from 4 checking above will force the program to tell the player to input again!

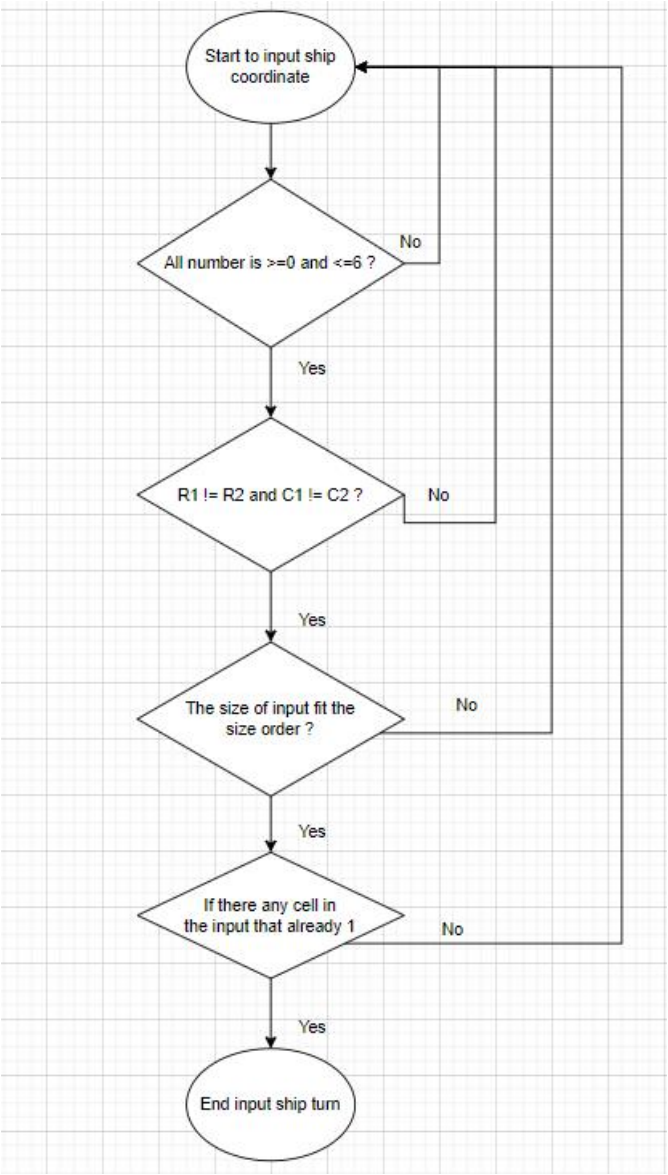

Figure 3: *Flowchart visualize how to checking the input coordinate of the ship is valid*

Example of each invalid input case

```
PLAYER A TURN TO SET THE SHIP!

                              Index   0   1   2   3   4   5   6

                                  +---+---+---+---+---+---+---+
                                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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
                                  +---+---+---+---+---+---+---+
Please input the bow and stern coordinate of the ship of size 4:
-3 0 0 0
You has just input an invalid coordinate. The valid coordinate is from 0 to 6. Please input again!
```

Figure 3.1: *Example of input a negative number*

```
PLAYER A TURN TO SET THE SHIP!

                          Index   0   1   2   3   4   5   6

                              +---+---+---+---+---+---+---+
                            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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
                              +---+---+---+---+---+---+---+
Please input the bow and stern coordinate of the ship of size 4:
1 2 3 4
You has just input a diagonal line. Please input again!
Please input the bow and stern coordinate of the ship of size 4:
```

Figure 3.2: *Example of input a diagonal 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

Figure 3.2.1: *Visualize the wrong input when diagonal*

```
PLAYER A TURN TO SET THE SHIP!

                                Index   0   1   2   3   4   5   6

                                    +---+---+---+---+---+---+---+
                                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   | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
                                    +---+---+---+---+---+---+---+
Please input the bow and stern coordinate of the ship of size 4:
0 0 0 2
You has just input a 3-size ship. The necessary size is 4. Please input again!
Please input the bow and stern coordinate of the ship of size 4:
```

Figure 3.3: *Example input wrong size of the ship*

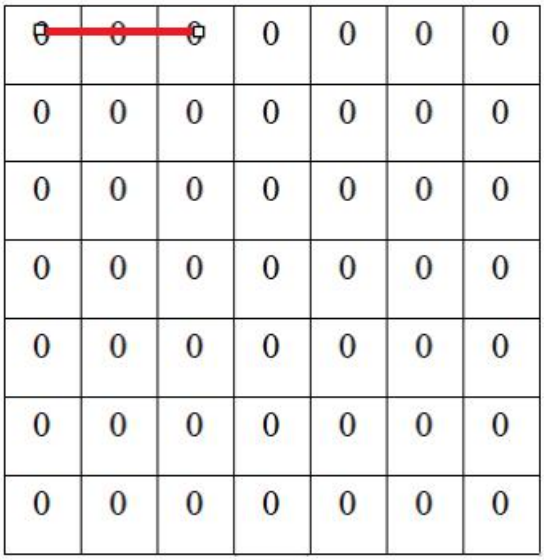

Figure 3.3.1: *Visualize the case when the player input the wrong size of the ship*

```
Index   0   1   2   3   4   5   6

      +---+---+---+---+---+---+---+
  0   | 1 | 1 | 1 | 1 | 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   | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
      +---+---+---+---+---+---+---+
Please input the bow and stern coordinate of the first ship of size 3:
0 0 0 2
You has just input an overlap ship. Please input again!
Please input the bow and stern coordinate of the first ship of size 3:
```

Figure 3.4: *Example of input an overlap ship*

1	1	1	1	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

Figure 3.4.1: *Visualize the wrong input when the player input the overlap the ship*

2.3 Checking coordinate shooting

- We do the same as checking the coordinate ship. First we make sure that all the number is >=0 and <=6.

- Then we check if that cell is already shot or not (I separate the table which store the ship coordinate and the table to show to the player, if the cell has been shot its value is -1).

- If the player hit the cell contains ship, we calculate the number of remaining cell contains ship. If is 0 then the game is over.

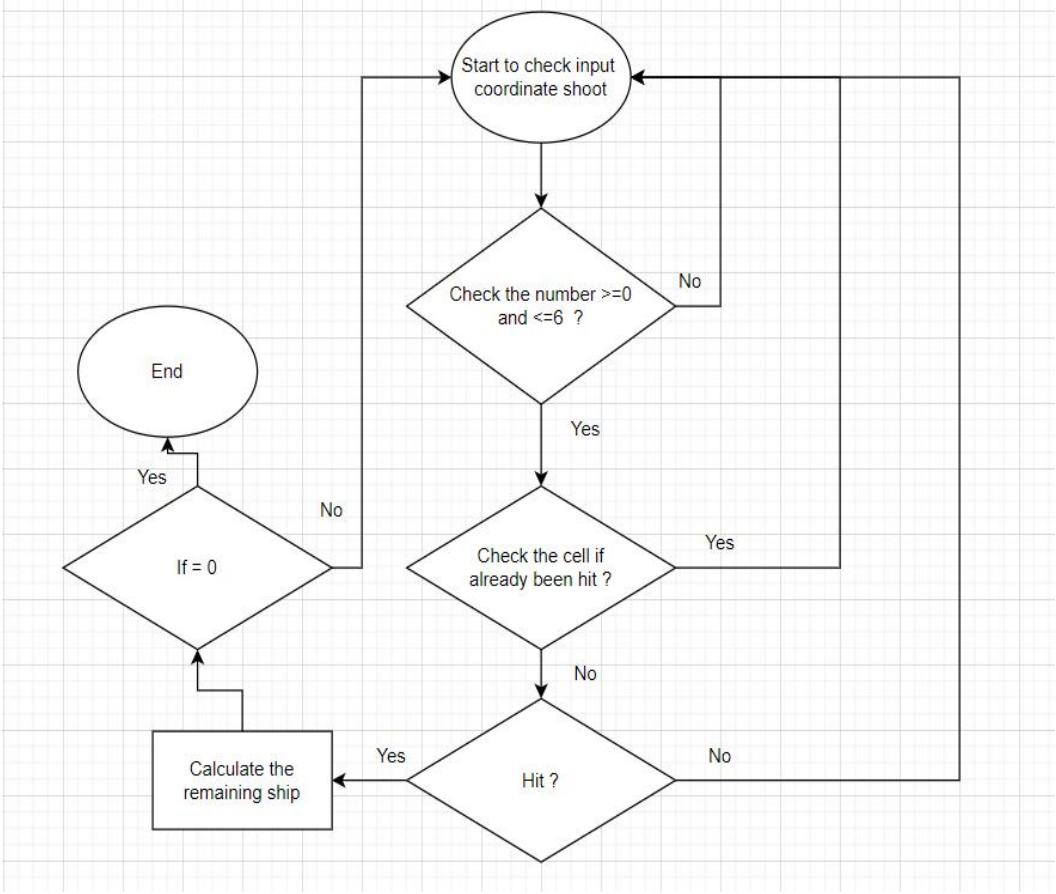

Figure 4: *Flowchart visualize how to checking coordinate shooting is valid*

```
A TURN TO SHOOT B SHIP!!!!
                    Index   0   1   2   3   4   5   6

                        +---+---+---+---+---+---+---+
                    0   |   |   |   |   |   |   |   |
                        +---+---+---+---+---+---+---+
                    1   |   |   |   |   |   |   |   |
                        +---+---+---+---+---+---+---+
                    2   |   |   |   |   |   |   |   |
                        +---+---+---+---+---+---+---+
                    3   |   |   |   |   |   |   |   |
                        +---+---+---+---+---+---+---+
                    4   |   |   |   |   |   |   |   |
                        +---+---+---+---+---+---+---+
                    5   |   |   |   |   |   |   |   |
                        +---+---+---+---+---+---+---+
                    6   |   |   |   |   |   |   |   |
                        +---+---+---+---+---+---+---+
Please choose a coordinate to shoot:
-3 0
You has just input an invalid coordinate. The valid coordinate is from 0 to 6. Please input again!
```

Figure 4.2: *Example of input negative number*

```
A TURN TO SHOOT B SHIP!!!!
                                Index   0   1   2   3   4   5   6

                                    +---+---+---+---+---+---+---+
                                  0 | X |   |   |   |   |   |   |
                                    +---+---+---+---+---+---+---+
                                  1 |   |   |   |   |   |   |   |
                                    +---+---+---+---+---+---+---+
                                  2 |   |   |   |   |   |   |   |
                                    +---+---+---+---+---+---+---+
                                  3 |   |   |   |   |   |   |   |
                                    +---+---+---+---+---+---+---+
                                  4 |   |   |   |   |   |   |   |
                                    +---+---+---+---+---+---+---+
                                  5 |   |   |   |   |   |   |   |
                                    +---+---+---+---+---+---+---+
                                  6 |   |   |   |   |   |   |   |
                                    +---+---+---+---+---+---+---+
Please choose a coordinate to shoot:
0 0
You already shoot that cell. Please choose again a coordinate to shoot:
```

Figure 2: *Example of input an already shoot cell*

```
HIT!!!
                                Index   0   1   2   3   4   5   6

                                    +---+---+---+---+---+---+---+
                                  0 | X | X | X | X | X | X | X |
                                    +---+---+---+---+---+---+---+
                                  1 | X | X | X | X | X | X | X |
                                    +---+---+---+---+---+---+---+
                                  2 | X |   |   |   |   |   |   |
                                    +---+---+---+---+---+---+---+
                                  3 |   |   |   |   |   |   |   |
                                    +---+---+---+---+---+---+---+
                                  4 |   |   |   |   |   |   |   |
                                    +---+---+---+---+---+---+---+
                                  5 |   |   |   |   |   |   |   |
                                    +---+---+---+---+---+---+---+
                                  6 |   |   |   |   |   |   |   |
                                    +---+---+---+---+---+---+---+
THE BATTLE SHIP GAME IS OVER! PLAYER A IS THE WINNER!!!!
-- program is finished running --
```

Figure 4.3: *Example of when the game is finished*

3 Write to file text:

MIPS do have support to write to the text file. We can find the way to do it in Help section. Go to syscall and scroll down to $v0 = 13,14,1516.

open file	13	$a0 = address of null-terminated string containing filename $a1 = flags $a2 = mode	$v0 contains file descriptor (negative if error). *See note below table*
read from file	14	$a0 = file descriptor $a1 = address of input buffer $a2 = maximum number of characters to read	$v0 contains number of characters read (0 if end-of-file, negative if error). *See note below table*
write to file	15	$a0 = file descriptor $a1 = address of output buffer $a2 = number of characters to write	$v0 contains number of characters written (negative if error). *See note below table*
close file	16	$a0 = file descriptor	

Figure 5: *Table instructions in Help section MIPS*

I name the file text output is “Assignmentouput.txt” in the MIPS code assignment.

```
filename: .asciiz "Assignmentoutput.txt"
save: .word 0
```

Figure 6: *Image from the code MIPS I do in the assignment*

The filename variable store the name of file, and the variable save to store the file descriptor. In the code MIPS assignment I try to call the write instruction whenever after the instruction to print to the terminal to write the file exactly the same to what I print to terminal.

```
                                                    WELCOME TO THE BATTLE-SHIP GAME !!!

PLEASE NOTICE:

0.IN THIS PROGRAM I DENOTE R FOR "ROW" , C DENOTE FOR "COLUMN"

1.WHEN INPUT A SHIP COORDINATE. PLEASE INPUT A BOW AND STERN OF SHIP IN 1 LINE: "R1 C1 R2 C2"
(IF YOU INPUT WRONG AS THE RULE. THE PROGRAM WILL TELL YOU TO INPUT AGAIN !!!)

EXAMPLE:Input bow and stern of the ship of size 4:
0 0 0 3 (R1 = 0, C1 = 0, R2 = 0, C2 = 3 is an example of valid input)

2.WHEN INPUT A COORDINATE TO SHOOT. PLEASE INPUT ROW AND COLUMN IN 1 LINE: "R C"

EXAMPLE:Input the coordinate to shoot:
0 0 (R = 0, C = 0 is an example of valid input)

PLAYER A TURN TO SET THE SHIP!

                          Index   0   1   2   3   4   5   6

                              +---+---+---+---+---+---+---+
                            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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
                              +---+---+---+---+---+---+---+
Player A has input the bow and stern coordinate of the ship of size 4:0 0
```

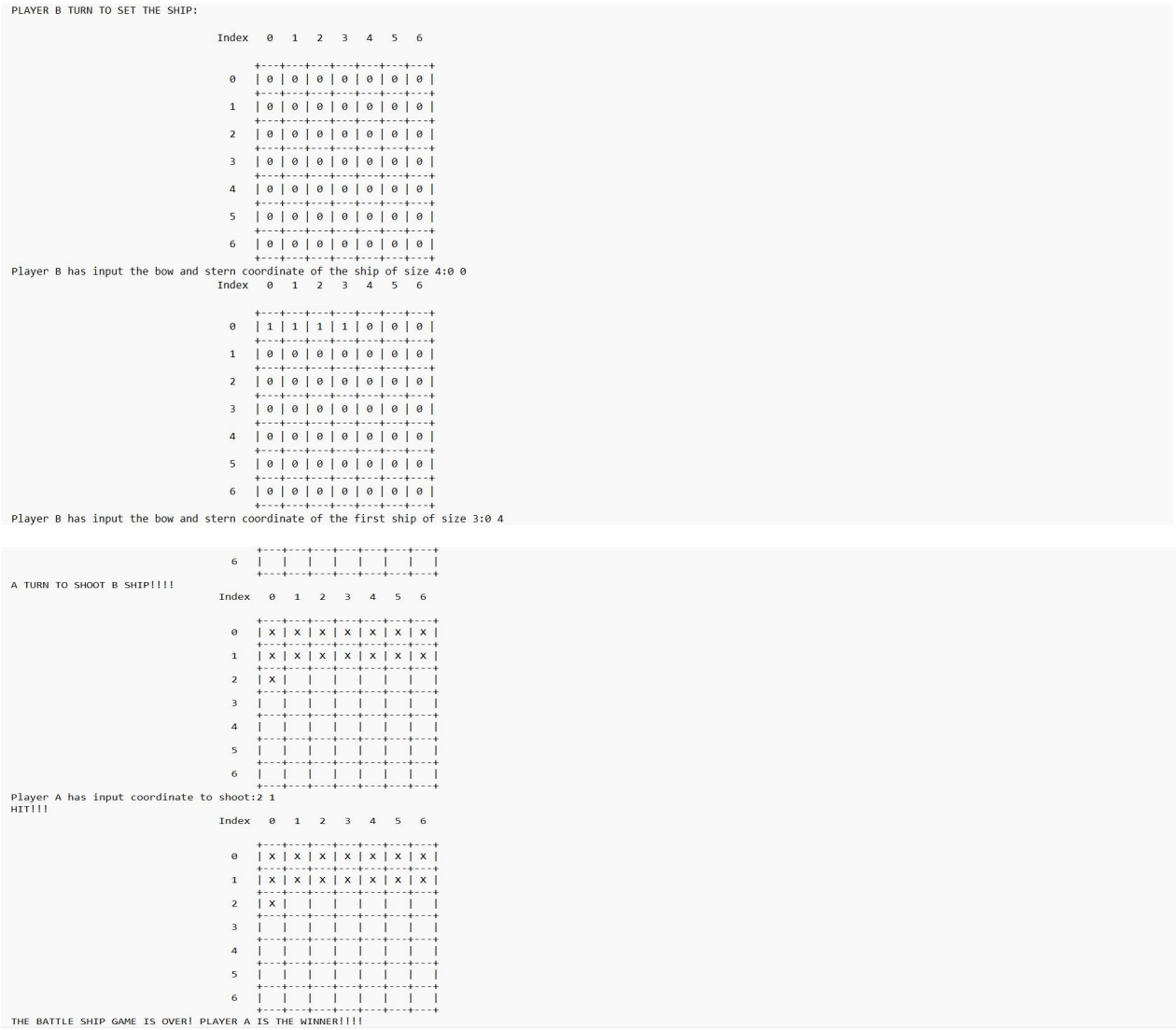

Figure 7: *Some image of the text file after written by MIPS*

4 Conclusion:

The assignment at first seem very hard. But thank to the teacher have explain all the confusing things in the assignment and some advice he give me I have done it on time. The flowchart also help me to visualize the idea to implement the code better. In the assignment I do some procedure as function for easier reading code as print table function or checking the valid coordinate. The assignment at result help me a lot. First I can know well the way to coding in the MIPS. Second I know how to implement code in the basic way and have to deal with a register, not variable. So it harder from C++ or Python. And the last thing is that

I know how to write file in MIPS. I may help me a lot in the future and I am very appreciate to be studying Computer Architecture in this semester. One again I would like to give my gratefulness to Mr Pham Quoc Cuong for helping and teaching me a lot in this subject. Thanks for your reading!

5 Reference:

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

[2] Assignment 1.1 - Battle Ship - HCMUT

[3] Lecturer Pham Quoc Cuong’s slides