





# REPORT LARGE EXERCISES SMALL COMPUTER ARCHITECTURE

Teacher: Phạm Quốc Cường

Student: Cao Hoàng Kiệt - 2053165



## **Report Standards**

### I. Requirement

Design and write MIPS assembly language for implementing a text-based Tic-Tac-Toe game for two players.

## II. Introduction a text-based Tic-Tac-Toe game

**Tic-Tac-Toe** is a simple game for 2 players (X - O) and is played on a 3x3 grid. Each player's goal is to make 3 in a row.

Players take turns placing their Mark, X or O, on an open square in the grid. The first player to make 3 of their own mark in a row vertically, horizontally, or diagonally wins the game. If all 9 squares are filled and neither player has 3 in a row, the game is considered draw.

0	X	0
0	X	X
X	0	X

Figure 1: Example Tic-Tac-Toe game

#### III. Idea and Algorithm

Firstly, I will design an friendly interface to get input from user. That is, if the first player choose O to play first, then another player will get X and vice versa. If players enter wrong or incorrect character instead of X - O (upper case), program will check input and throw error.

```
Select X or O for first player: u
Just 2 characters X or O (upper case of x and o). Please enter again!!!
Select X or O for first player: i
Just 2 characters X or O (upper case of x and o). Please enter again!!!
Select X or O for first player: o
Just 2 characters X or O (upper case of x and o). Please enter again!!!
Select X or O for first player:
```

Figure 2: When user enters wrong input, the output like above

To get this idea, I generate a procedure called **chooseX\_O**. Moreover, while loop inside this procedure does:

- Make a request to get input from the user ("Select X or O for the first player: "). Then store it into a variable called s (type char)
- Compare s with 'X' and 'O'. For instance, if (s == 'X' or s == 'O') then continue, else print an error (like "Just 2 characters X or O (upper case of x and o). Please enter again!!!").
- Loop until getting valid input.

• Our game will start when I call it in **Main** function, if there is no mistake.

```
Select X or O for first player: X

Now let start our game !!!!

1 2 3

4 5 6

7 8 9

It's turn for player 1. Enter your slot:
```

Figure 3: The game starts like this

Secondly in **Main** function, I create a board (3x3) by making an **characterArray**[] which contains characters from '1' to '9'. Because the address between 2 characters in this array is adjacent (distance between them is one byte), so when want to print the board like figure 3, I will use an index i to run from 0 to 8, whenever i % 3 == 0, program will print a new line. Therefore, we have Tic-Tac-Toe board game.

Next, I generate for loop in **Main** function to loop 9 cycles only (because users can enter numbers only from 1 to 9 for their turn). And in this loop, program need to know the slot from which user want. Thus, I make a procedure named **getSlot** to do this:

• First, print a message "It's turn for player (1 or 2). Enter your slot: ". When changing from player 1 to player 2 in MIPS code, I default positions 0, 2, 4, 6, 8 are player 1 and another positions are player 2. So, player 1 always plays first.

```
It's turn for player 1. Enter your slot: 1

X 2 3

4 5 6

7 8 9

It's turn for player 2. Enter your slot:
```

**Figure 4**: Entering slot and changing from player 1 to player 2

• In addition, when program gets the slot 1 (example in figure 4), instruction will change character from '1' to 'X' (ig. Player 1 chooses 'X') and store it into array at **charaterArray[0]**. And again for player 2, but this time he/she must choose other slots (not slot 1). If he/she enters slot 1 again, program will check by loading **characterArray[0]** to a variable (temp) and comparing it with character 'X' or 'O' (that is if (temp == 'X' or temp == 'Y')). If the comparison is equal, program will throw error.

```
It's turn for player 1. Enter your slot: 1

X 2 3

4 5 6

7 8 9

It's turn for player 2. Enter your slot: 1

The slot has already existed, please enter another slot !!!!
```

**Figure 5**: Entering again slot 1 from player 2 and program throws error

• Also, if user tries to enter the slot not in 1 - 9, program will check again until there is no error.

```
It's turn for player 2. Enter your slot: 12
Invalid input, please enter number from 1 to 9 !!!!

X 2 3

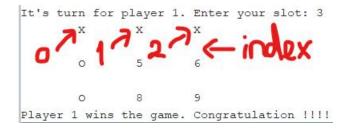
4 5 6

7 8 9
It's turn for player 2. Enter your slot:
```

Figure 6: Entering slot 12 from player 2 and program throws error

After getting slot from user, we also need to check who is winner. Therefore, I generate a procedure in for loop above called **Winner (return value)**. This procedure does:

- At the beginning, checking each row horizontally in a board (3x3) by using for loop (**figure 7**). In this loop, it loops only 3 times because address of each element in array separates one byte. In particular, we have pesudo code:
- for i = 0; i < 9; i += 3 // i += 3 to go next row
- if (characterArray[i] = characterArray[i + 1] = characterArray[i + 2])
   return true (1). // that is if there is enough 3 'X' or 'O' horizontal
- esle continue the game (return false (0)).



**Figure 7**: Example when checking row horizontally

• Secondly, checking each row vertically in a board (3x3) by using for loop

again (**figure 8**). This loop does similarly with loop above except index i need to increase by 1. We have pesudocode

- for i = 0; i < 3; i += 1 // i += 1 to go next column
- if (characterArray[i] = characterArray[i + 3] = characterArray[i + 6])
   return true (1) // that is if there is enough 3 'X' or 'O' vertical
- esle continue the game (return false (0))

```
It's turn for player 1. Enter your slot: 7

X

O

X

O

O

S

A

X

S

S

B

Player 1 wins the game. Congratulation !!!!
```

Figure 8: Example when checking row vertically

• Done with horizontal and vertical check, we also need to check row diagonally on index i = 0, 4, 8 and i = 2, 4, 6.

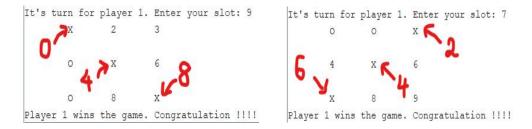


Figure 9: Example when checking row diagonally

Finally, the return value will be got and if the value is false along with the loop exceed 9 cycles in the **Main** function or there is enough space in the board then the result is draw, else print who wins.

```
It's turn for player 1. Enter your slot: 8

X O X

X X O X

X X O X

X X O O

X X X S S G

X X S S S

Player 1 wins the game. Congratulation !!!!

It's turn for player 2. Enter your slot: 8

X O X

X O G

Player 2 wins the game. Congratulation !!!!
```

**Figure 10**: Example of winning and drawing game between 2 players