**VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY OFFICE FOR INTERNATIONAL STUDY PROGRAMS**



REPORT:

COMPUTER ARCHITECTURE

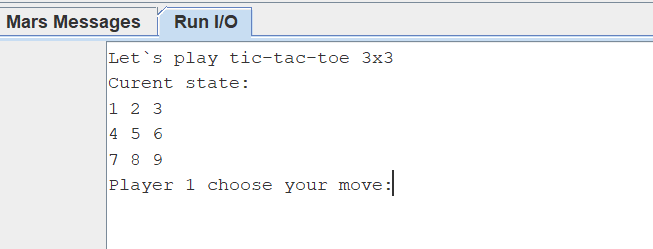
**Instructor: Mr. Phạm Quốc Cường**

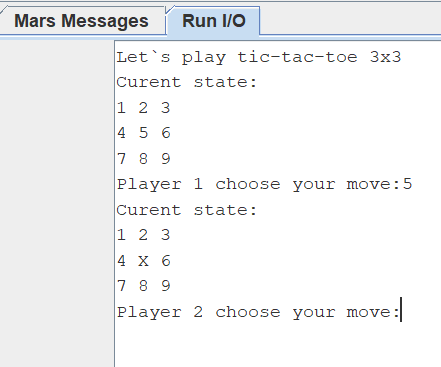
**Nguyễn Trần Mạnh Cường – 1852283 – CC02**

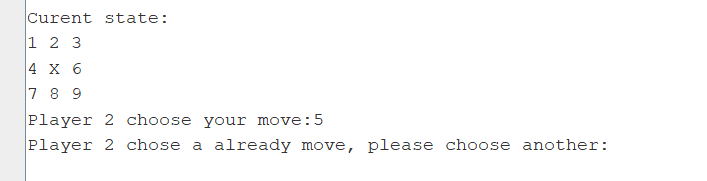
**TIK-TAC-TOE 3X3**

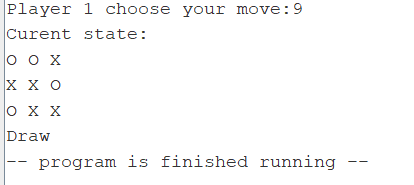
1. **Introduction:**
   1. **Tic-Tac-Toe 3X3:**

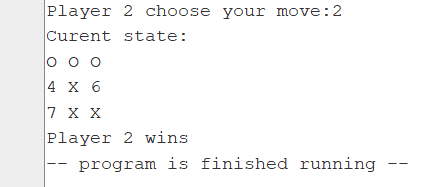
This is a paper-and-pencil game for two players who take turn marking the spaces in three-by-three grid usually with X and O. Player who succeeds in placing three of their mark in a horizontal, vertical or diagonal row is a winner. There is a draw match if there is no more place to mark and no winner.

* 1. **My Tic-Tac-Toe 3X3 in MIPS assembly language:**
* Instead of a three-by-three gird, I create a 3X3. Each entry of this matrix is a number from 1 to 9 following left to right, up to down and least to most order.
* Each times a player wants to make a move they need to type a number according to a place they want to move. After a number has been chosen, immediately that number is changed to a mark whose move (Player 1 is X and player 2 is O).



* Each movement has already been changed to a mark will not be replaced and if you choose that again system will ask you to choose another place.
* Two players keep going until there is no place to mark or has a winner. If the match has a winner system will announce who is win and if a draw match the announcement is “Draw”.





* 1. **Ideas:**

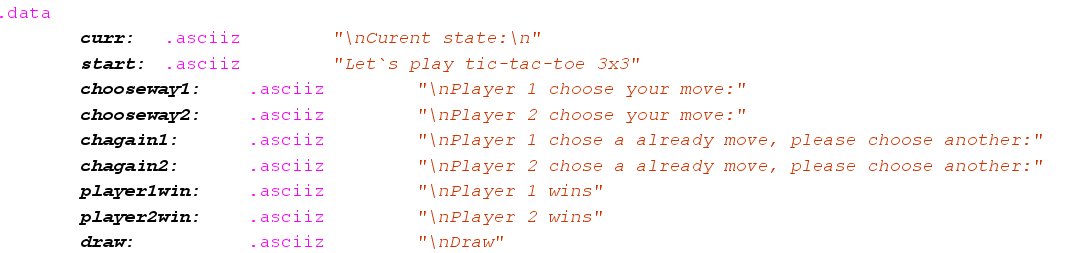
I separate the whole process into 3 parts, there are the target of each part:

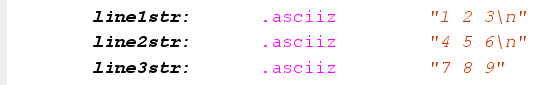
* + Displaying states and comments to instruct and announce player.
  + Placing a mark.
  + Checking the result.

1. **How to make a game in MIPS.**

As I have mentioned in the first part, I separate the process in 3 parts. The flow of my program is first of all I will display an initial state of the board with all empty places, then I ask player 1 make a first move, then player 2. After a movement from a player is taken, I check whether that move has been already played or not. Subsequently, I examine whether a match is end or not, if it ends I will print the result on the console, if not the match will go on and the others move. Then the whole above process repeats until the match ending.

Now I will explain my code, ideas and algorithms in 3 parts below.

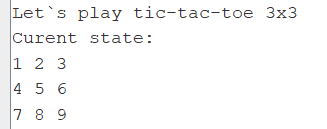
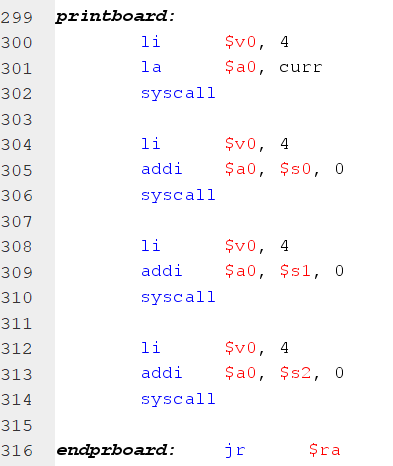
* 1. **Displaying:**
* Firstly, I write all the comments I need in .data part.
* Because this just a 3X3 Tic-tac-toe so I choose to draw chess board by 3 lines ascii and print it in order. I use this method because I want to work with each single ascii line instead of a 3X3 matrix.

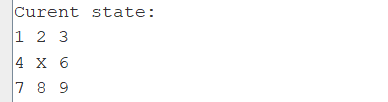
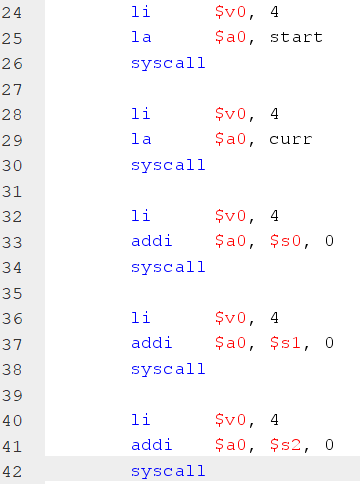


* From line 24 to 42 and line 299 to 316 are two part I use to print the board at the first and every time a player makes a move.

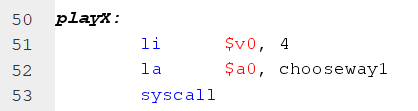
Commands: **li $v0, 4** and **syscall** use to print a ascii string write in data part.

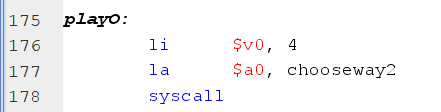
In the **printboard** part I have a command: **jr $ra** in order to use **jal printboard** command whenever I want to print present state. With this method I can print board every time and shorten the code.

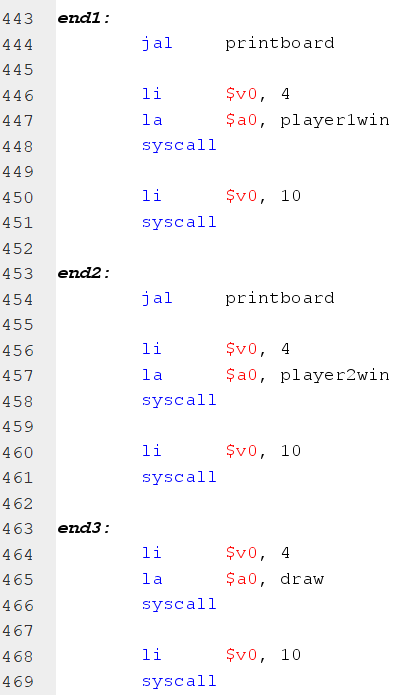


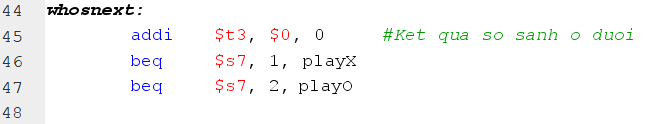


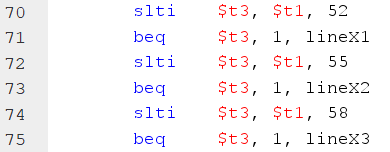
* From line 50 to line 53 and line 175 to 178 are comments which at the beginning of **playX** and **playO** parts to instruct player choose a move.

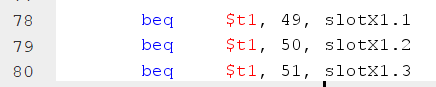




* From line 443 to line 469, this part will announce the result of the match. **end1** is for player 1 won, **end2** is for player 2 won and **end3** is for a draw match.
  1. **Marking into a number:**
* Firstly, system will check whose turn first by using a register **$s7**. If **$s7** equal 1 is player 1 (X) and qual 2 is player 2 (O).

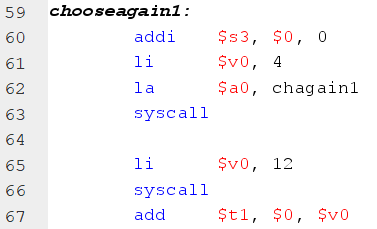


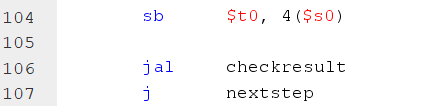
* According to the first part, I have told that I present the match in 3 string line. Therefore, every time player makes a move that place will be replace as ascii number 79 for O and 88 for X. I use a register **$t0** to store number 88 or 79 and then store that to the place that player want to play.
* When I want to find a place that player want to make a move, I use switch … case algorithm. If player want to mark at a place less than 3 (52 in ascii code) then system will be written in first line. If less than 6 (55) writes at the second line and if less than 9 (58) writes at final.
* In each line, I do another switch … case in each slot: 1, 2, 3 (49, 50, 51 in ascii) according to first line; 4, 5, 6 (52, 53, 54) from second line; 5, 6, 7 (55, 56, 57) in third line. I do the same as each other lines.

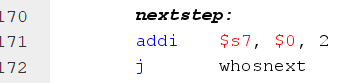


* In each **slotX** part, initially, I check for whether it has already been chosen or not by checking the digit at that place is number or X and O. If X and O system will ask people to make another move.

The others case will be the same as the first one.



If it is not a already movement then system will replace the number by a mark X (88 in ascii) or O (79 in ascii). Next step is checking result and then system change player turn by using a register $s7 (if equal 1 is player 1 and 2 is player 2), and then repeat the whole process from asking a move.



* I do the same idea and algorithm with player 2 (who plays O).
  1. **Checking the result:**
* I separate the checking program into 6 situations, the first 1, 2, 3 are using to check marks in the same line. The fourth situation checks 3 columns. And the others check two diagonal lines.
* In each situation, I check the three marks are the same by adding the ascii code store in each play I want to check. If the sum is equal 264 which is three X (88 \*3) and is equal 237, three O (79 \*3).



* In all situation, I come to each entry and add to a register. The weakness of this method is if we want to create an infinite board applying this method is impossible because it has infinite situation.