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

COMPUTER ARCHITECTURE - CO2007

Assignment

FOUR IN A ROW

Students: Võ Trung Kiên - 2153502

HO CHI MINH CITY, NOVEMBER 2022

Contents

1 Concept of game Four in a row

To implement the game. Firstly, we have to determine the structure of the game.
The basic concept is:

- Make a list of instruction for both players.
- Randomly choose which player to go first.
- Player who go first will pick (X or O) to start. The other player sticks with the remain piece.
- While the game still not end, each player can input the column they wish to occupy and ask player to undo their move, each of them has 3 times to undo. Moreover, each player has 3 times to input an invalid column (not between 1-7 or this column is full of characters), if violate this rules the other will win the game.
- When the game check who has the condition to win it will print out the winner or when the game board is full of character there's a tie for this game match.

Base on that structure, the following part is to implement each part of the above structure.

2 Implementation and Algorithms

Start the game I already made some useful instructions for 2 players in order to understand the rule of the game well after make random the player go first.

We can randomly choose the player by a random integer system call in MIPS to choose a player by enter any integer that from the user. From that number I saw that MIPS will random a positive number or a negative number so based on these two type of number I will make that if the random number from MIPS is greater than 0 player 1 will go first while random number is less than 0 player 2 will go first.

Then continue let the player who go first choose X or O by make them enter 0 to choose X or enter another number to choose O and continue the flow that I have been discussed in Structure section.

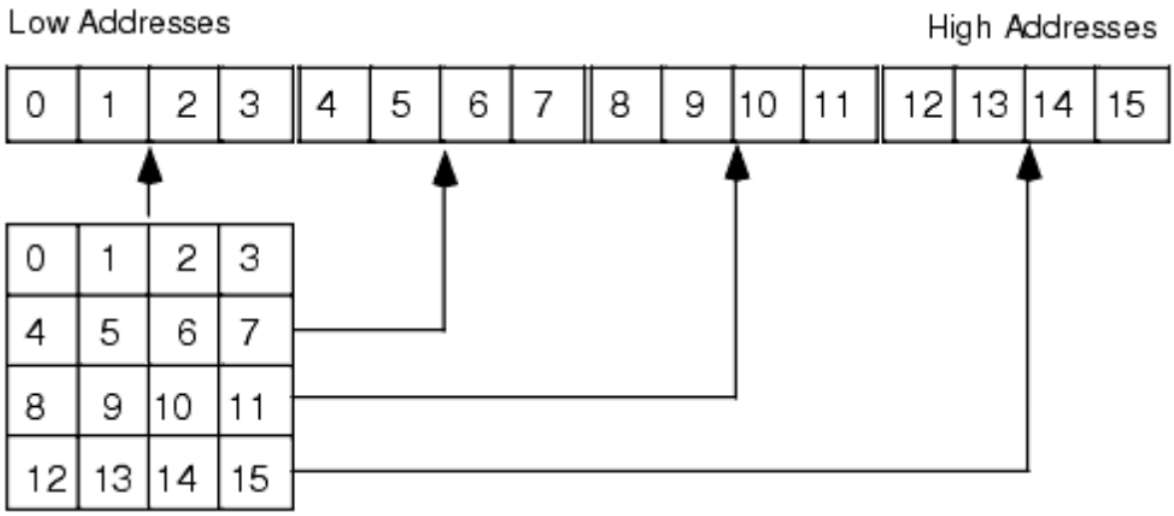
2.1 Print game board

So far we have dealt only with 1-dimensional arrays. Theoretically there is no limit on the dimension of an array. The only difficulty in the implementing arrays of higher dimension is calculating the correct index values. A 2-dimensional array is made up of rows and columns. These rows and columns are mapped into the 1- dimensional memory layout. We can use a 1D array that have 42 elements to represent for the 2D array that have 6 rows and 7 columns. The most common way to deal with this problem is using the row order method and it is the method we shall use. We need to calculate the appropriate index value by using this formula:

$$\text{Address need to get} = \text{Base address} + (i * \text{Number of columns} + j) * \text{Size of data} \quad (1)$$

where i and j are the row index and column index respectively that we want to access.

Here is an example for equation as I mentioned above:



Below is pseudo-code to show how I store dot element into each cell:

Algorithm 1 Set all element to dot

```
for i ← 0, Number of rows - 1 do
    for j ← 0, Number of columns - 1 do
        get address arr[i][j] by substitute in Eq. (1)
        arr[i][j] ← '.'
    end for
end for
```

2.2 Set cell

In Set cell function I will pass the value of column, row and sign of each player whenever this function is call in order to make the cell with dot will become X or O. Moreover, I also use it to make the undo function which can set the cell with X or O become dot cell.

The important way is always calculate the appropriate address and below is the pseudo-code to show the algorithm:

Algorithm 2 Set character for cell got address call

1: get address arr[i][j] by substitute in Eq. (1)
2: get the character pass into function 'X' or 'O' or dot
3: $arr[i][j] \leftarrow$ character

2.3 Get cell

In Get cell function I will pass the value of column and row of each player whenever this function is call in order to get the cell contains X or O or dot and return it to compare, for example I want to check this cell is blank or not to fill the character I will call it to get the decision or I want to check this cell is already filled or not to jump to another cell to fill in the character.

Below is the pseudo-code to show the algorithm of get cell function:

Algorithm 3 Get character from cell got address call

1: get address arr[i][j] by substitute in Eq. (1)
2: **return** $arr[i][j]$

2.4 Turn choose and turn check

First of all in MIPS I will let turn choose and turn check as an static variable by load its value with 1.

After random function to choose which player will go first if player 1 go first these static variables have no change, if player 2 go first I will make these static variables change by using registers calling these static variables by loading their address into register after that with store byte I can set static variable to zero to make player 2 go to check win and make this player turn to choose X or O and the same with player 1. One of its advantages is I can reuse registers many time when I do that.

2.5 Choose player function

Based on random function to choose player who go first and player can choose which character they will start the game. I divide it to four cases and I used 2 variable to store 2 type of characters . If it is player 1 go first and choose X player 1 (X), player 2 (O) and reverse, it is the same with player 2.

Below is my pseudo-code to show the algorithms:

Algorithm 4 Choose player

1: If player 1 go first
2: Player 1 (X), Player 2 (O) or Player 1(O), Player 2(X)
3: If player 2 go first
4: Player 2 (X), Player 1 (O) or Player 2(O), Player 1(X)

A useful way I use here is I create 2 static variables for character of player 1 and player 2, which I can load address of these variables and make change for them by store value when it is called.

2.6 Insert character and check wrong input

As the instruction of this Four in a row game when we will drop the token from the bottom to the top until this column is full of character. Therefore, my algorithm for insert character is using a for loop, the column is input from players have no change when it is valid and I will check these cell of this column from bottom to top whenever there is a cell has no character this player can insert character into this cell.

Below is the algorithm for insert character:

Algorithm 5 Insert character

1: **for** $i \leftarrow$Number of rows - $1, 0$ **do**
2: get address arr[row][col - 1] by substitute in Eq. (1)
3: $arr[row][col-1] \leftarrow$ 'X' or $arr[row][col-1] \leftarrow$ 'O'
4: **end for**

With check wrong input I will make condition for the column that players insert, if they insert a number or a character out of range $1 \leq$ column ≤ 7 it will ask players again about another input until it receive an valid input from players.

Based on the usefulness of get cell function it can return this cell already contains character or not, if it has no character yet players can insert character into this column else this player must insert another column.

2.7 Undo function

Each players have 3 times to undo so as I mention before with static variable I also use it in undo function to check the remaining of times for this player to undo. I have create 2 static undo variable for both players so whenever these players want to undo their move. By loading address of undo variable's address and make a register contains this value after they choose undo this value will minus 1 and store this value again to these static variable. Therefore, when finish while loop to ask player undo or not we can load value from this static variable to check if it less than zero this player will has no time for undo or print to screen times for this player's undo left.

I make undo function for players with set cell and get cell function, with get cell I can check cell that player's new input. Because of inserting from bottom to top so I will check cell's new input by using row for loop which run from top to bottom of this column, if this cell is filling with an character I will set it to dot cell wit set cell function. Below is the pseudo-code to explain my algorithms:

Algorithm 6 Undo function

1: **for** $i \leftarrow 0,$Number of rows - 1 **do**
2: column with valid insertion
3: get address arr[i][col] by substitute in Eq. (1)
4: getCell(i, col) = 'X' or getCell(i, col) = 'O'
5: $arr[i][col] \leftarrow$ '.'
6: **end for**

2.8 Check who is winner

I use turn check variable to set turn of player 1 check or turn of player 2 check. For example, when the player 1 finish check win game condition I will make turn check to player 2 and the same with player 2 and I only call this function when players finish their input character to the column.

Below is the pseudo-code to show how can I check player win game or not.

While I run 2 for loop to check all the index element of the game board I will check win game by using getCell function as I mention before and pass the character of current player to check condition this player win the game or not.

For i set to 0, Number of rows - 1 do

For j set to 0, Number of columns - 1 do

Move i to row

Move j to col

+ Victory from 4 in a row:

If (col $<$ 4) then

$getCell(row, col) = character$ & $getCell(row, col + 1) = character$ & $getCell(row, col + 2) = character$ & $getCell(row, col + 3) = character$

return true

+ Victory from 4 in a column:

If (row $<$ 3) then

$getCell(row, col) = character$ & $getCell(row + 1, col) = character$ & $getCell(row + 2, col) = character$ & $getCell(row + 3, col) = character$

return true

+ Victory from 4 in a diagonal:

If (col $\geq$ 3) then

$getCell(row, col) = character$ & $getCell(row - 1, col - 1) = character$ & $getCell(row - 2, col - 2) = character$ & $getCell(row - 3, col - 3) = character$

return true

+ Victory from 4 in a reverse diagonal:

If (col $\leq$ 3) then

$getCell(row, col) = character$ & $getCell(row - 1, col + 1) = character$ & $getCell(row - 2, col + 2) = character$ & $getCell(row - 3, col + 3) = character$

return true

esle return false

I created a stop game variable which equals to 0, when check win game condition return true this variable will update to 1 so there is a player has win game if not this variable still equals 0 and the game will be continued.

2.9 Main function and check draw

In main function, I will make a list of instructions for both players in order to help them understand game well and prevent them from making mistakes after that I demand them to input a number to random player who go first and let them pick X or O. I created counting variable from 1 to 42 in for loop and after each time when this player finish their move I will set turn choose, turn check again for remaining player and counting variable plus 1. If after 42 turn and stop game variable still 0 it will jump to exit and announcing that there's a tie in this game match. Below is my pseudo-code for the main function:

Algorithm 7 Basic main function

1: **for** $turn \leftarrow 1, 42$ **do**
2: Depend on turn check and turn choose to make player move
3: Player move
4: Set turn check and turn choose for remaining player move
5: turn + 1
6: **end for**
7: Depend on turn check and stop game variable we can print the winner.
8: Else after 42 turn stop game variable still 0 we can conclude that there's a tie game.

Here is an example of draw game:

	1	2	3	4	5	6	7
1	X	O	X	O	X	O	X
2	X	O	X	O	X	O	O
3	X	O	X	O	X	O	X
4	O	X	O	X	O	X	O
5	O	X	O	X	O	X	X
6	O	X	O	X	O	X	O