Viet Nam National University, Ho Chi Minh City
University of Technology
Faculty of Cuter Science and Engineering

COMPUTER ARCHITECTURE (CO2007)

Assignment

"FOUR IN A ROW"

Academic year: 2022 - 2023. Semester 221

Advisor: Mr. Pham Quoc Cuong
Student: Tran Gia Huy - 2152600 (Group CC02)

Ho Chi Minh City, 03/12/2022

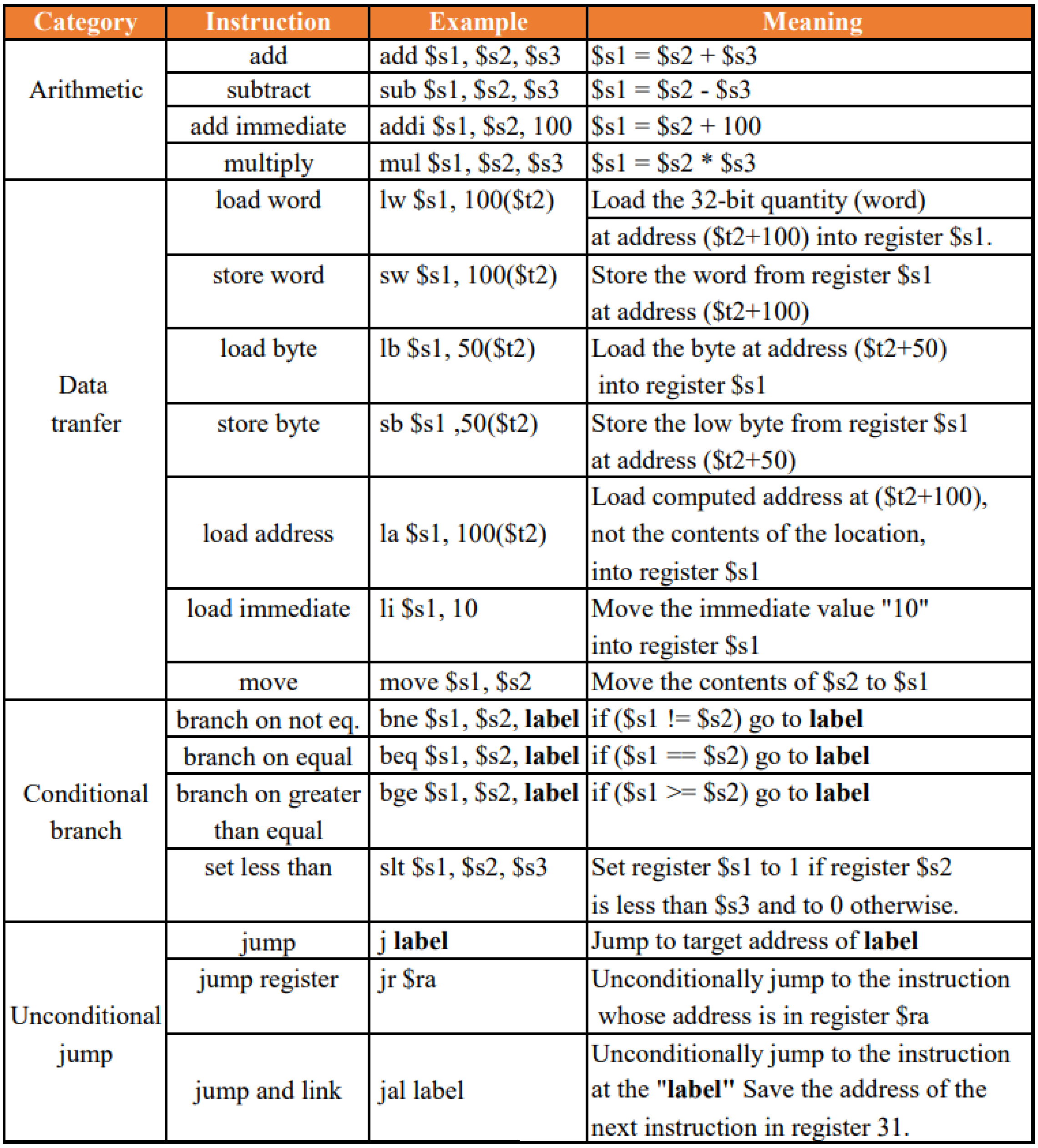
Assignment: Four in a row.

Computer Architecture (CO2007)

Semester: 212 - *Group:* CC02

Contents

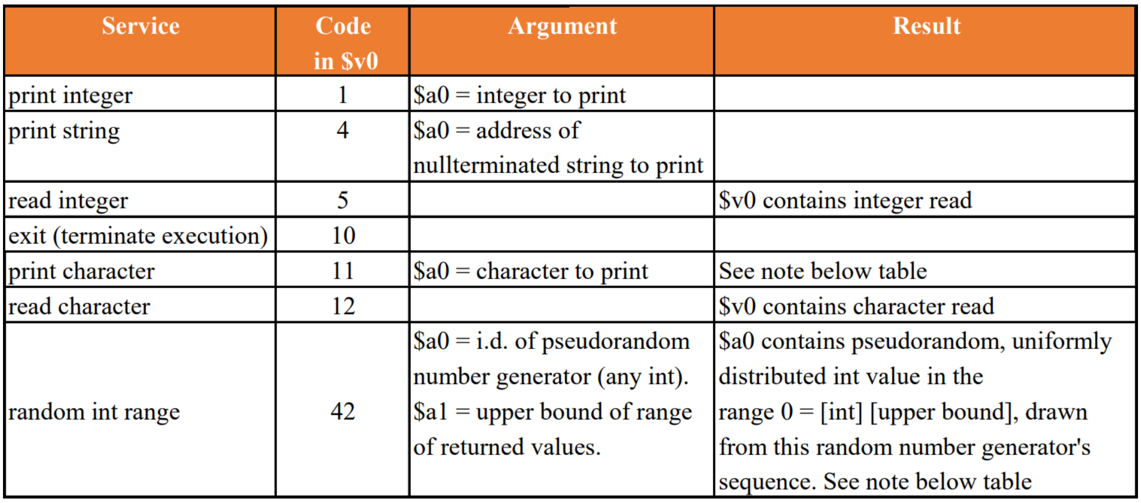
List of used MIPS instructions



Category	Instruction	Example	Meaning
Arithmetic	add	add $s1, $s2, $s3	$s1 = $s2 + $s3
	subtract	sub $s1, $s2, $s3	$s1 = $s2 - $s3
	add immediate	addi $s1, $s2, 100	$s1 = $s2 + 100
	multiply	mul $s1, $s2, $s3	$s1 = $s2 * $s3
Data tranfer	load word	lw $s1, 100($t2)	Load the 32-bit quantity (word) at address ($t2+100) into register $s1.
	store word	sw $s1, 100($t2)	Store the word from register $s1 at address ($t2+100)
	load byte	lb $s1, 50($t2)	Load the byte at address ($t2+50) into register $s1
	store byte	sb $s1 ,50($t2)	Store the low byte from register $s1 at address ($t2+50)
	load address	la $s1, 100($t2)	Load computed address at ($t2+100), not the contents of the location, into register $s1
	load immediate	li $s1, 10	Move the immediate value "10" into register $s1
	move	move $s1, $s2	Move the contents of $s2 to $s1
Conditional branch	branch on not eq.	bne $s1, $s2, **label**	if ($s1 != $s2) go to **label**
	branch on equal	beq $s1, $s2, **label**	if ($s1 == $s2) go to **label**
	branch on greater than equal	bge $s1, $s2, **label**	if ($s1 >= $s2) go to **label**
	set less than	slt $s1, $s2, $s3	Set register $s1 to 1 if register $s2 is less than $s3 and to 0 otherwise.
Unconditional jump	jump	j **label**	Jump to target address of **label**
	jump register	jr $ra	Unconditionally jump to the instruction whose address is in register $ra
	jump and link	jal label	Unconditionally jump to the instruction at the "**label**" Save the address of the next instruction in register 31.

MIPS instructions

List of used SYSCALL functions



Service	Code in $v0	Argument	Result
print integer	1	$a0 = integer to print	
print string	4	$a0 = address of nullterminated string to print	
read integer	5		$v0 contains integer read
exit (terminate execution)	10		
print character	11	$a0 = character to print	See note below table
read character	12		$v0 contains character read
random int range	42	$a0 = i.d. of pseudorandom number generator (any int). $a1 = upper bound of range of returned values.	$a0 contains pseudorandom, uniformly distributed int value in the range 0 = [int] [upper bound], drawn from this random number generator's sequence. See note below table

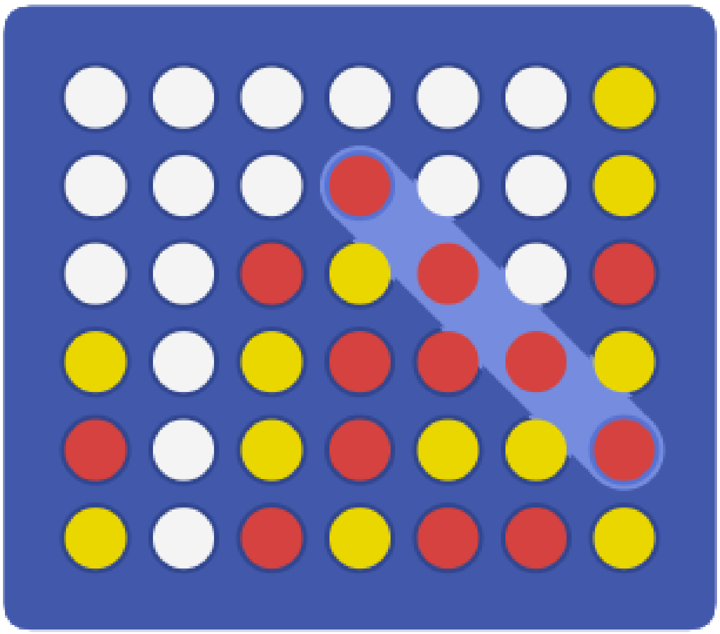
SYSCALL functions

NOTES:

- **Service 11** - Prints ASCII character corresponding to contents of low-order byte.
- **Service 42** - use underlying Java pseudorandom number generators provided by the *java.util.Random* class. Each stream (identified by $a0 contents) is modeled by a different Random object.

1 Introduction

Four in a Row is the classic two player game where you take turns to place a counter in an upright grid and try and beat your opponent to place 4 counters in a row.



An example Four In A Row board

Objective

The game is played with a **seven-column** and **six-row** grid, which is arranged upright. **The starting player is randomly chosen**, pick a game piece color (yellow or red) and can place a piece in any column. Each player then alternately takes a turn placing a piece in any column that is not already full.

The piece fall straight down, occupying the lowest available spot within the column or be stopped by another piece. The aim is to be the first of the two players to connect four pieces of the same colour vertically, horizontally or diagonally (an example is shown in Figure 1). If each cell of the grid is filled and no player has already connected four pieces, the game ends in a draw, so no player wins.

2 Gameplay

In this assignment, I am going to design and write MIPS assembly language for implementing a text-based Four in a Row game for two players as follows:

- First, randomly choose the starting player and let this player pick the piece (X or O).

  The other one has to stick with the remain.
- Then, let the game begin. Four in a Row rules are based on the description at section 2
- Moreover, in the middle of the game (after their first move), each player has 3 times to undo their move (before the opponent's turn).
- Finally, the output of the program is the result of the game.

In addition, the exception of placing a piece at an inappropriate column by restarting the move will also be handled. If any players try to violate it 3 times. This player will lose the game.

3 Algorithm

The algorithm of this assignment will be present in below step:

Step 1. Start:
Ask the user press 1 to play the game

Step 2. Random player:
Generate a random number to choose a starting player

Step 3. Picking piece:
Ask the starting player to pick a piece ("x" or "o").
If the starting player pick anything that is not "x" or "o", redo step 3.

Step 4. Choose column:
Ask the current player to choose a column to place him/her piece.
At this moment, there are **2 options** for the players (can only choose 1 of 2):

- Pick a number from 0 to 6 to perform a **MOVE** for the current player. Go to **step 5.**
- Base on the rule in section 2, the opponent can Press "-5" to **UNDO** his move.
  If the player runs out of **"UNDO"** move, redo step 4.
  If the table is currently empty, redo step 4.
  Ex:

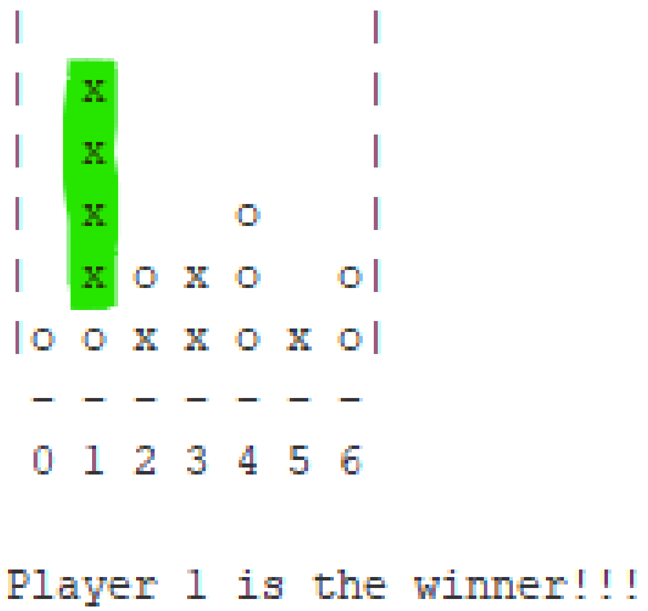
Step 5. Handling exceptions and checking game status:
Firstly, checking whether the piece has been placed at an inappropriate column or not:

- **Exception 1:** The chosen column is out of range [0;6]. Redo step 4.
- **Exception 2:** The chosen column has run out of space (The column has been full before being chosen). Redo step 4.

If any players try to violation these 2 exceptions **3 times** , this player will **lose** the game!

Secondly, checking the winning conditions. If any plays meet 1 of 4 winning conditions below, this player will win the game and go to Step 7!
There are 4 conditions:

1. Vertically to Upper tokens

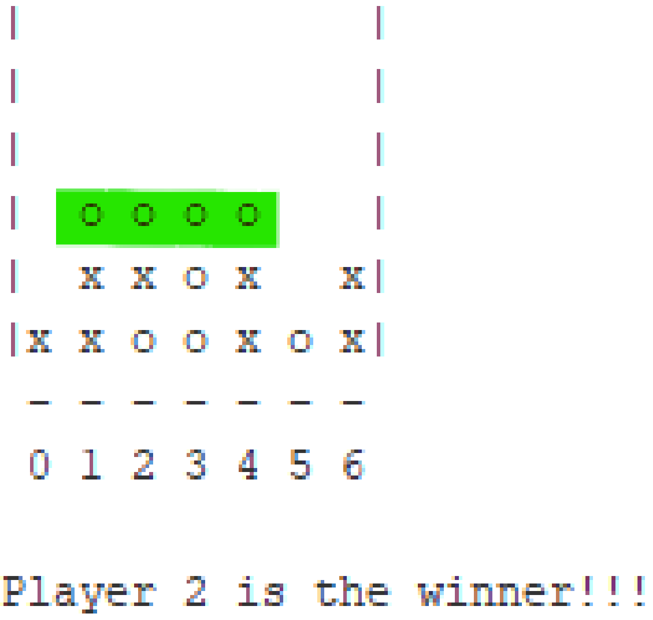


```
|             |
|  x          |
|  x          |
|  x     o    |
|  x o x o   o|
|o o x x o x o|
 - - - - - - -
 0 1 2 3 4 5 6

Player 1 is the winner!!!
```

Vertically to Upper tokens

2. Horizontally to right tokens

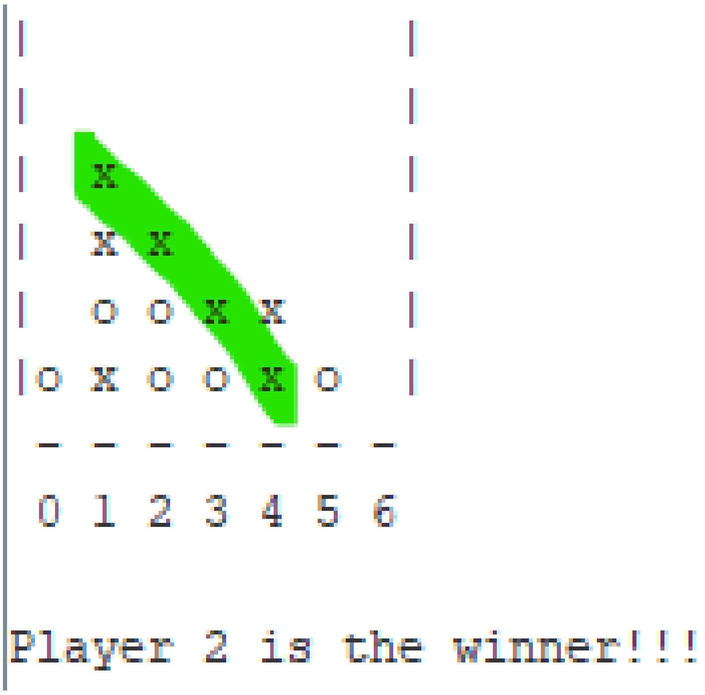


```
|             |
|             |
|             |
|  o o o o    |
|  x x o x   x|
|x x o o x o x|
 - - - - - - -
 0 1 2 3 4 5 6

Player 2 is the winner!!!
```

Horizontally to right tokens

3. Diagonally Up-Left

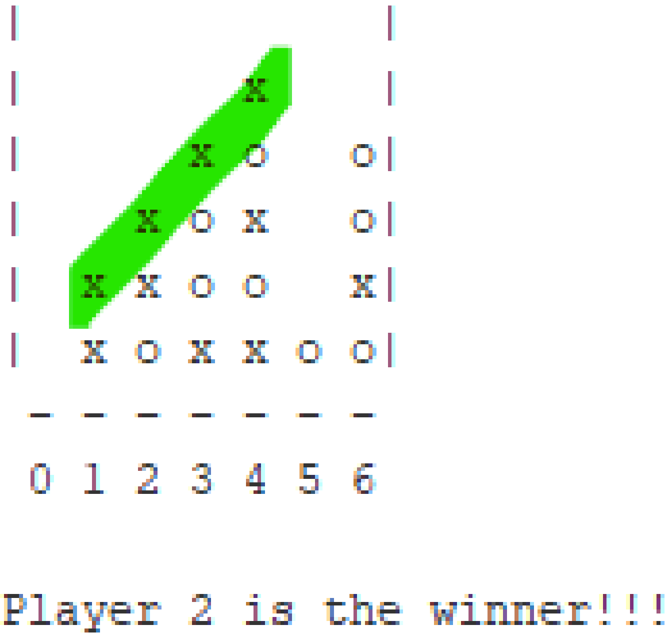


```
|             |
|             |
|  x          |
|  x x        |
|  o o x x    |
|o x o o x o  |
 - - - - - - -
 0 1 2 3 4 5 6

Player 2 is the winner!!!
```

Diagonally Up-Left

4. Diagonally Up-Right

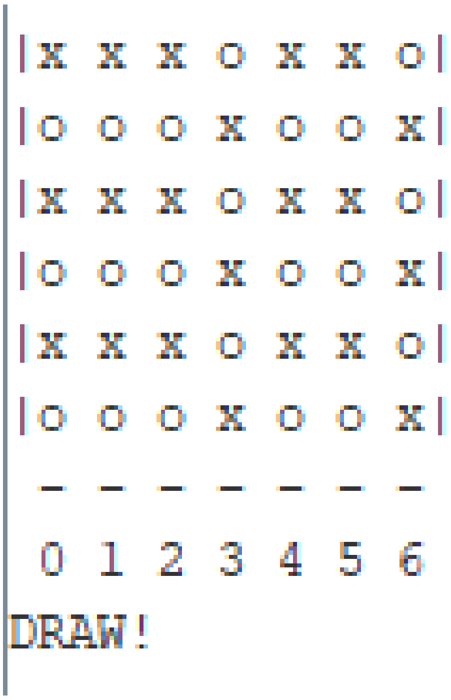


```
|              |
|         x    |
|       x o   o|
|     x o x   o|
|   x x o o   x|
|   x o x x o o|
  - - - - - - -
  0 1 2 3 4 5 6

Player 2 is the winner!!!
```

Diagonally Up-Right

Thirdly, checking the **drawing condition**: If the table is **full** (each cell of the grid is filled) and no player has already connected four pieces, the game ends in a draw, so no player wins. Go to step 7.



```
|x x x o x x o|
|o o o x o o x|
|x x x o x x o|
|o o o x o o x|
|x x x o x x o|
|o o o x o o x|
 - - - - - - -
 0 1 2 3 4 5 6
DRAW!
```

Drawing

After exceptions handling and game status checking process, if currently the player doesn't meet any exception,drawing or winning conditions, go to Step 6.

Step 6. Switching turn:

Switch turn to the other player. Redo step 4.

Step 7. End game:

Base on the previous step, announce the winner or announce the game is draw. The game is over here!

4 Conclusion

After finishing this assignment, I feel that I have learnt a lot about MARS MIPS simulator, Arithmetic & data transfer instructions, conditional branch , unconditional jump instructions, procedures, basic arithmetic algorithms and how computer processes an expression to yield a desirable answer, thus, appreciate the works and the computational ability we have today.

From this project, I can create a hardware that follows these algorithms, understand the implications of all this for instruction sets and have an intuition of how to use this knowledge to make arithmetic-intensive programs go much faster.

After all, we are thankful of this wonderful assignment and hope that we could develop this project further to make interesting applications.

Source code:

This GitHub repository contains my source code.
`https://github.com/HuyyTran/Computer_Architecture_assignment`

References

[1] AI Gaming. Four in a row. Available at `https://help.aigaming.com/game-help/four-in-a-row`.

[2] Wikipedia. Conect four. Available at `https://en.wikipedia.org/wiki/Connect_Four`.

Contacts

For any questions, please send an email to:

huy.trandev@hcmut.edu.vn