

Proyecto Parcial

Tic - Tac - Toe

Integrantes

Leonardo Xavier Kuffó Rivero

Carlos Emilio Orellana Fantoni

Materia: Organización de Computadores

Paralelo: 1

Fecha de Entrega: 17 de noviembre de 2018

Profesora: Ing. Alexandra Arias



**Contenido**

[Tabla de Ilustraciones 2](#_Toc530259319)

[Especificaciones 3](#_Toc530259320)

[Funcionamiento 4](#_Toc530259321)

[Capturas 4](#_Toc530259322)

[Referencias 8](#_Toc530259323)

# **Tabla de Ilustraciones**

[Figura 1. Pantalla inicial 4](#_Toc530259193)

[Figura 2. Pantalla luego de ingresar una posición válida (Jugador 1) 5](#_Toc530259194)

[Figura 3. Salida al ingresar un valor fuera de rango 5](#_Toc530259195)

[Figura 4. Salida al ingresar un valor válido ya ingresado previamente 5](#_Toc530259196)

[Figura 5. Juego finalizado - Victoria Jugador 1 5](#_Toc530259197)

[Figura 6. Juego finalizado – Empate 6](#_Toc530259198)

[Figura 7. Tres en raya en código assembler (MARS) 6](#_Toc530259199)

[Figura 8. Registros al inicio del juego 7](#_Toc530259200)

[Figura 9. Registros al finalizar una partida 8](#_Toc530259201)

# **Especificaciones**

Librerías utilizadas

* stdio.h
* stdlib.h

Funciones de librerías utilizadas

* int printf(const char \*format, ...)
* int scanf(const char \*format, …)

Métodos creados[[1]](#footnote-1)

* void imprimirTablero(char code[], int tablero[])
* int jugar(int tablero[], int posicion, int jugador)
* int verificarGanador(int t[], int turno)

Descripción de métodos

* imprimirTablero: recibe un arreglo *code* [“-”,”X”,”O”] y el tablero. Se imprime línea por línea el formato clásico de un tablero de tres en raya utilizando una combinación de “\_” y “|”. Para los caracteres en cada espacio, se leen los números que tiene *tablero* y según estos, se colocará el caracter que se encuentre en esa posición en *code*:
  + “-” si el número es 0
  + “X” si el número es 1
  + “O” si el número es 2
* jugar: recibe el número identificador del jugador en turno, la posición que haya ingresado y el tablero. Aquí se hacen las validaciones del valor que haya ingresado el jugador:
  + Si la posición ingresada está fuera de rango, se retorna -1
  + Si la posición ingresada ya tiene un caracter diferente de “-”, se retorna -1
  + Si la posición ingresada es válida, se coloca el identificador del jugador en el arreglo *tablero* y se retorna 0.
* verificarGanador: recibe el tablero *t* y el turno actual. Se verifican primero todas las posibles formas de victoria, es decir, si en alguna de las filas o alguna de las columnas o en la diagonal principal o secundaria (transcritos a un arreglo unidimensional) se encuentra repetido un mismo número (diferente de 0); si es así, el número que se encuentre corresponde al jugador ganador. En caso de que no haya ganador aun, se verifica en qué turno se encuentra el juego (*turno* es un contador); si ya el juego se encuentra en el turno 9, en todos los espacios se ha colocado algún caracter, por lo que la tabla se encuentra llena y al no haber jugador ganador se declara empate. Se retornará

# **Funcionamiento**

El programa consiste en una versión de tres en raya (tic-tac-toe) para dos jugadores creada en C. No utiliza un GUI, es decir, el tablero como tal se imprime en la consola utilizando la función *imprimirTablero*. Comenzará el Jugador 1, es decir, a quien le corresponda el caracter “X”, mientras que al Jugador 2 le corresponderá el caracter “O”. El tablero tendrá el caracter “-” en las posiciones del tablero donde aún no se ha colocado otro caracter, y las posiciones se encuentran enumeradas del 1 al 9. Cada turno, el jugador en turno ingresará por consola la posición en la que desea colocar su caracter, lo cual se realiza mediante la función *jugar*. Al final del turno, se verifica si se ha alcanzado alguna condición de salida, es decir, si algún jugador ha ganado o si se ha llegado a un empate, lo cual se encarga la función *verificarGanador*.

Nota: SOLO se aceptan entradas de números enteros (solo estos son válidos).

# **Capturas**

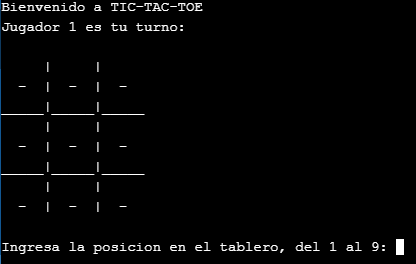


Figura . Pantalla inicial

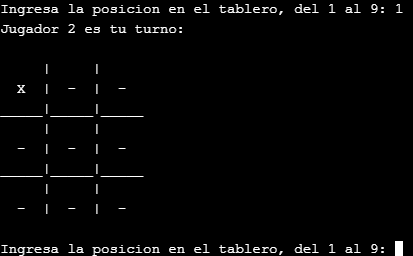


Figura . Pantalla luego de ingresar una posición válida (Jugador 1)



Figura . Salida al ingresar un valor fuera de rango



Figura . Salida al ingresar un valor válido ya ingresado previamente

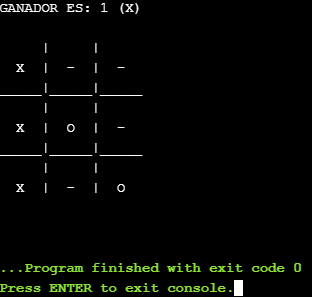


Figura . Juego finalizado - Victoria Jugador 1

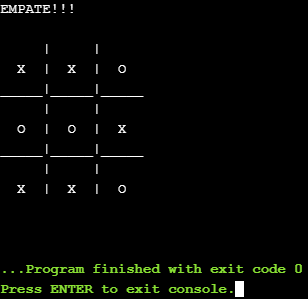


Figura . Juego finalizado – Empate

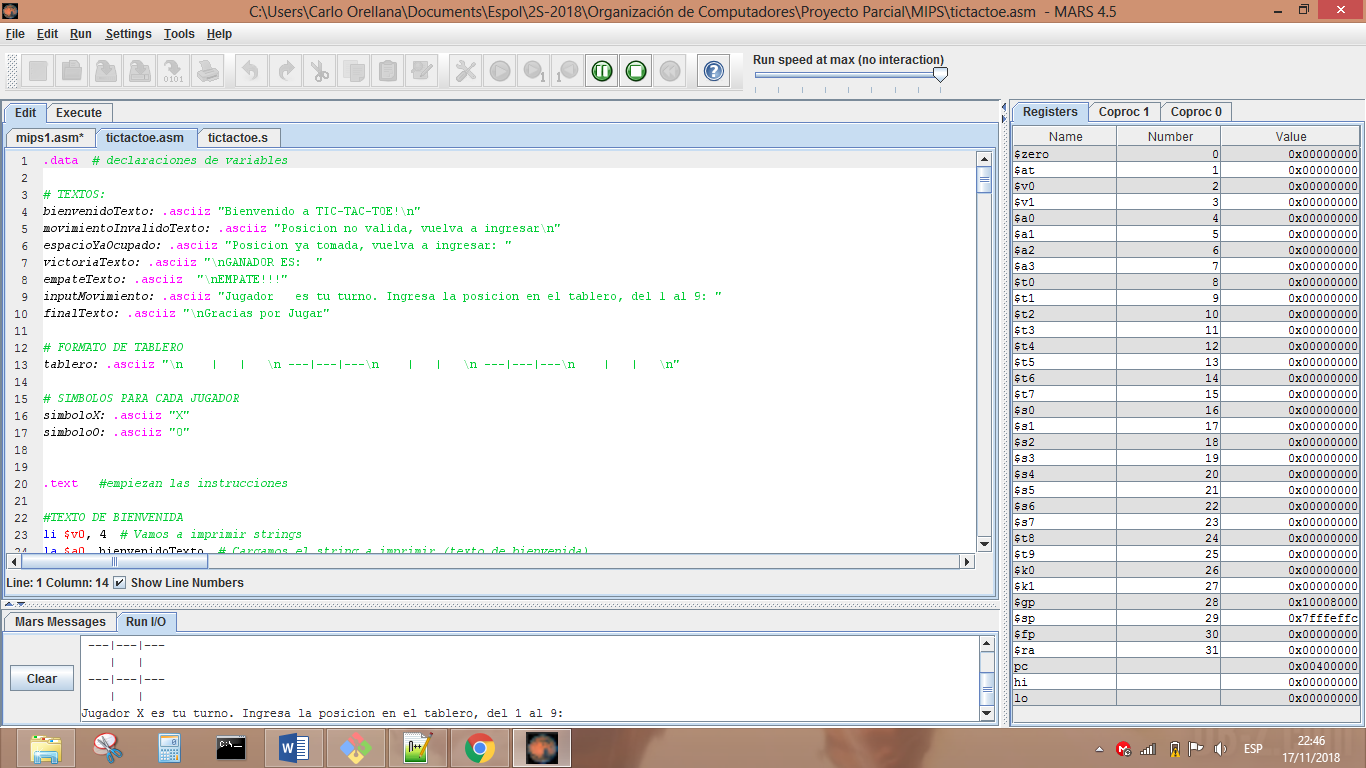


Figura . Tres en raya en código assembler (MARS)

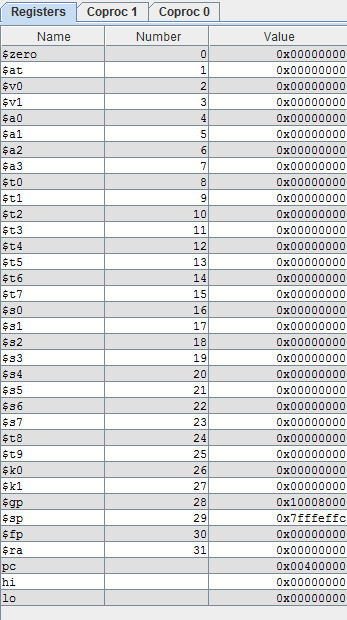


Figura . Registros al inicio del juego

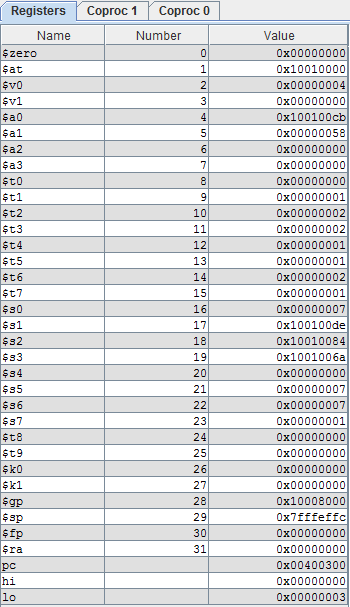


Figura . Registros al finalizar una partida

# **Referencias**

Moor, I. (11 de Marzo de 2009). System Calls. Recuperado el 14 de Noviembre de 2018, de SPIM S20: A MIPS R2000 Simulator: https://www.doc.ic.ac.uk/lab/secondyear/spim/node8.html

Tutorialspoint. (20 de Octubre de 2012). C library function - printf(). Recuperado el 15 de Noviembre de 2018, de Tutorialspoint: https://www.tutorialspoint.com/c\_standard\_library/c\_function\_printf.htm

Tutorialspoint. (20 de October de 2012). C library function - scanf(). Recuperado el 15 de Noviembre de 2018, de Tutorialspoint: https://www.tutorialspoint.com/c\_standard\_library/c\_function\_scanf.htm

University of Illinois at Chicago. (21 de Noviembre de 2004). MIPS Quick Tutorial. Recuperado el 14 de Noviembre de 2018, de University of Illinois at Chicago: http://logos.cs.uic.edu/366/notes/mips%20quick%20tutorial.htm

Vollmar, K. (29 de Octubre de 2018). MARS MIPS Simulator. Recuperado el 14 de Noviembre de 2018, de Missouri State University: https://courses.missouristate.edu/KenVollmar/mars/tutorial.htm

1. No se incluye el método main [↑](#footnote-ref-1)