Algorithms and Techniques used in the program:

The program maintains the state of the game board as an array of 9 bytes, which in the beginning of the program are all occupied with space ascii values. The program first starts off in the main, where registers $s0 and $s1 are loaded with 0 and 9 respectively. Then the program jumps to ‘printInstruction’ where the program prints the instructions and allows the user to pick the symbol ‘X’ or ‘O’. Depending on the user’s choice of input, the program will then jump back to the main to labels named ‘OLoop’ or ‘XLoop’, with ‘OLoop’ making the computer go first and ‘XLoop’ making the user go first. Both loop will either get the computer input or user input, followed by printing the board by jumping to the ‘displayBoard’ label where it will display a tic tac toe board along with the current game board array of 9 bytes, then checking the win condition by jumping to the ‘checkWin’ label, and then increments $s0 by one and compares with $s9 to see if it equals 9 to see if there is a tie game. If there is a tie game, it will then jump to ‘tieGame’ where it will display that there has been a tie game and exit the program.

When it comes to gathering input, the ‘userInput’ and the ‘computerInput’ labels are where the program gets the inputs. In the ‘userInput’ we first display a message saying that it is the users turn followed by asking the user to input the desired cell they want to put their symbol. After we get the integer value, we validate the number by looping a counter 9 times, in which if the integer matches the values between 1 and 9, we break out of the loop. If it does not match, it will display a error message and jump back to beginning of the program. After validating the number, the program then validates whether the space does not contain a ‘X’ or an ‘O’. If the space does contain the either symbol, the program will display an error message and start at the beginning of the ‘userInput’ label. If not, the array of 9 bytes gets updates by adding the appropriate symbol in the appropriate space, and the program will then jump back to the specified register address. For the ‘computerInput’ label, the program will generate a random number between 0 and 8 and will check whether the space on the array contains either ‘X’ or ‘O’ symbols. If it does, it will jump back to the beginning of the of the ‘computerInput’ to redo the process. Then it will do the same process as the one in ‘userInput’ to input the symbol into the board array.

Finally, the ‘checkWin’ label checks to see if there are 3 of the same symbols horizontally, vertically, diagonally, and anti-diagonally. The label will first check whether the win condition was meet horizontally, where we use a loop that iterates 3 times to go through each of the game boards rows. We do this by having a counter, we will name it ‘n’ in this example, that is set to zero. Then we do 3n, 3n+1, and 3n+2 to get the row members of the arrays. Once we get the members, we check to see if the byte is equal to each. If it isn’t equal, we jump back to row loop to loop through the next row. If the byte is equal, we check to see if it isn’t a space character and if it isn’t, we jump to the ‘checkWinner’ label to see whether the user or the computer was the winner. We loop through until the counter hits 3, where we then exit the row loop and continue to the column loop. The column loop also follows a similar logic to the row loop, only instead it adds to the counter ‘n’ by 0, 3, and 6 to go through each column. Finally for the diagonal and anti-diagonal, all we do is find the board positions where it hold the diagonal values and check to see if there is a win condition. If at any time the win condition is met, the program will jump to ‘checkWinner’ where it will check whether the user or the computer won the game, display the appropriate message, and end the program. If the win condition is not met, the program will then jump back to main functionality where it will play the game as usual.

Overall, the programme implements the game logic using fundamental programming constructs including loops, conditional statements, and function calls.

Additionally, it shows how to use arrays to represent the state of the game board and the syscall function to communicate with the user.