Computing Mid-term Assessment

Name: Gregory Partridge

Student Number: 17331009

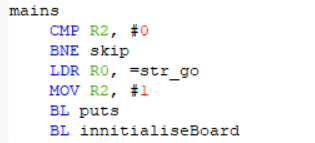
Date of Submission: 14/03/2019

\*Note:

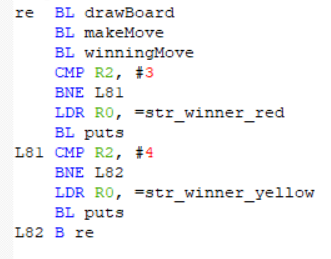
My program can make a game of connect4, but I was unsuccessful in getting a functioning win subroutine. I did not attempt the extra credit.

Main:

My main works by initially initializing the board and printing the start statement. The compare statement was for making sure that the main subroutine did not loop.

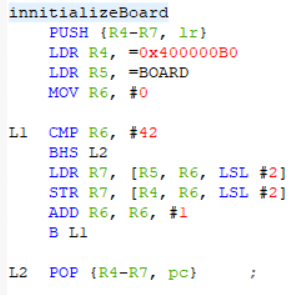


After this I had a loop that repeated which would draw the board, make a move and attempt to check for a winning move. After checking for winning move it would see who, if anyone, won.

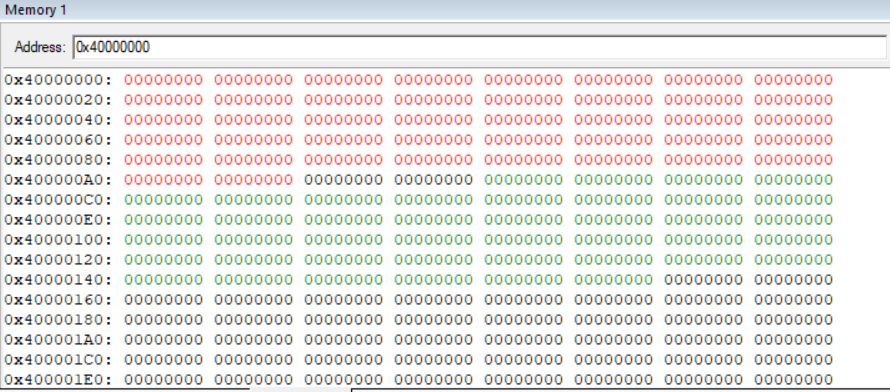


innitializeBoard:

My initialize subroutine worked by loading the flash memory and storing it elsewhere as RAM. It does this by loading memory at 0x40000000 and storing it at 0x400000B0 incrementing each memory place by 4 a total of 42 times.

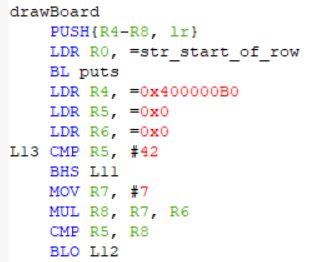


The memory looked like this when you ran the code.



drawBoard:

The draw board subroutine works by first drawing the top row. I did this by using a string. After this I had a for loop for drawing the rest of the board. The columns were printed at the side using a whatToPrintOnColoum subroutine which I will explain later. The main of the subroutine would have a counter that would increase each time it wrote the subroutine. When it reached 7 it would reset and skip to the next line followed by using the whatToPrintColoum that would then print the appropriate column number.

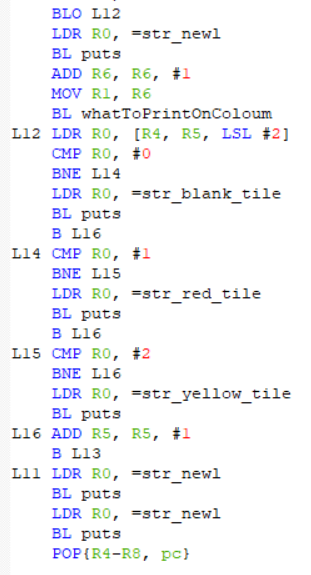


After this I have a small bit of code that prints the corresponding value on the board depending for what is in memory.

It is representable like this:

|  |  |  |
| --- | --- | --- |
| Memory Value | Display | What is represents |
| 0 | 0 | Clear |
| 1 | R | Red |
| 2 | Y | Yellow |

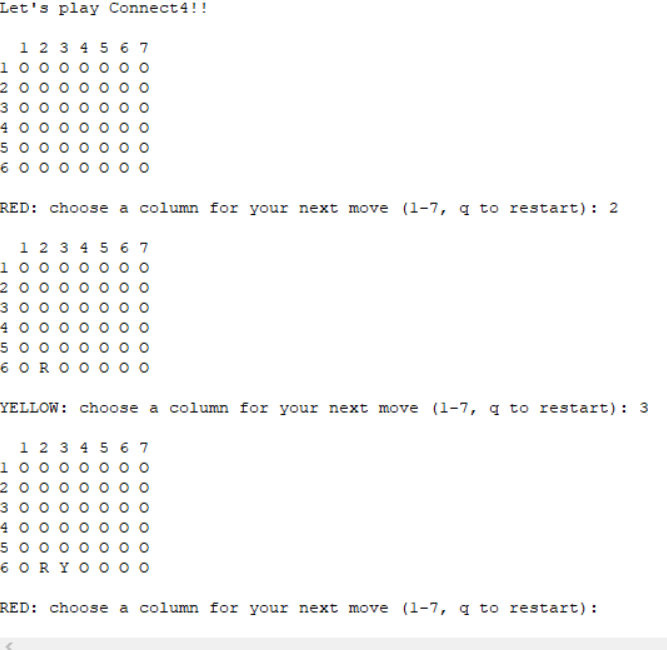
The code I wrote is:



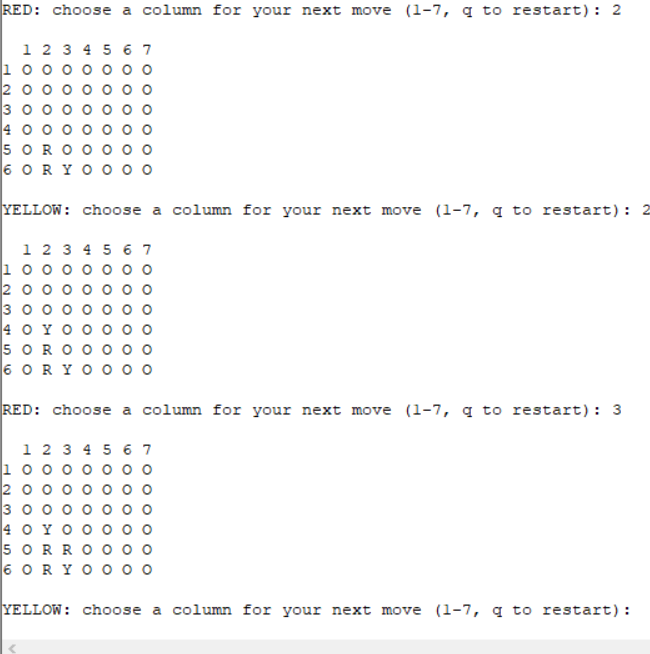
After I print the string I make space to make sure the board doesn’t look cluttered.

As it shows in the above table if the memory has a 0 in its address it will show a ‘O’. If there is a 1 in the address it will show ‘R’ and for the case of a 2 it will show a ‘Y’.

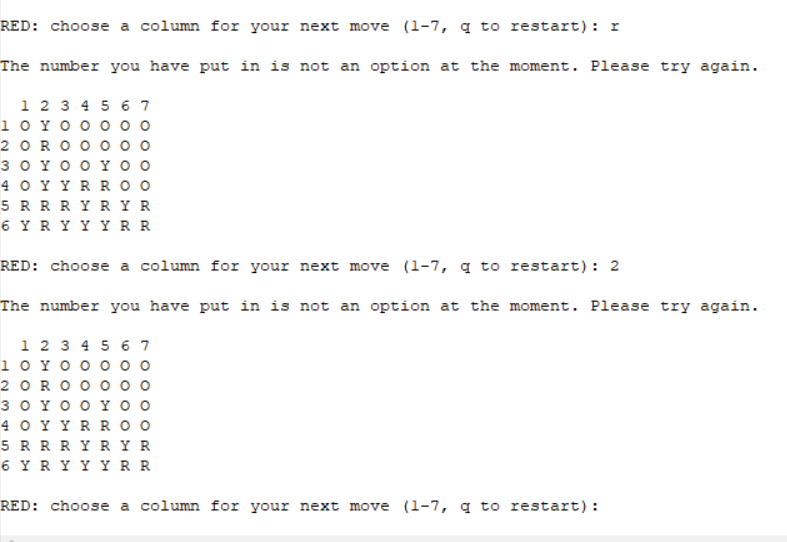
When running the drawBoard subroutine you get the following results in the uart1 window:



In this we see that we take the input from the user and that the pieces go in the correct spot. The program also properly swaps between the two players after each turn.



In this example we see that the pieces properly stack on top of each other.

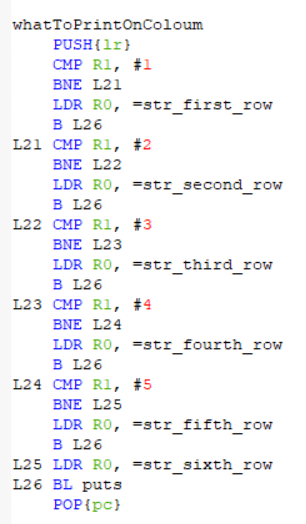


In this example we can see that if it is not an appropriate value it will ask for another input but keep the players turn. It also does the same if you are unable to place in that position.

I also tried to get a ‘q’ input to restart the program, but I was never able to get the input to register.

whatToPrintOnColoum:

The what to print on column I just a series of if statements. It operates by taking the current column and representing the appropriate string. For example, if it was on the second column it would skip the first if statement but on the second it would load the string and print it.



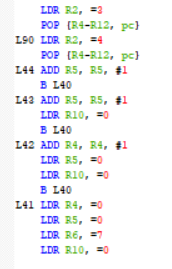
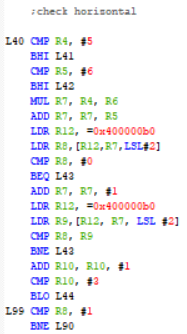
winningMove:

My winning move subroutine though it doesn’t work I believe a lot of the elements inside it show I understand how to answer the question. I broke the question into three parts.

The parts I did were the check horizontal, check vertical, check diagonal left and diagonal right. For each won I had the same base values I loaded. I explain them when explaining how I did my check horizontal.

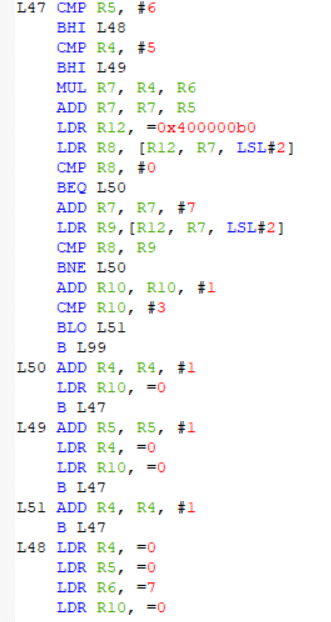


I started with the check horizontal. I would load the start pos for the row and the column. I would then load how many columns and a counter. The base idea would be it would check all the spaces. It would go across a column and for each value that was greater than 0 it would add to a new counter. For each space it went it would add to the counter for the spaces. If the counter for the spaces reached 7 it would go to the next row and reset the new counter for how many were in a row. If the new counter reached 4 it would call a win function that would basically add 2 to the players turn for whoever won. If no one won, it would go on to check vertically. It would reset the appropriate values whenever it went to a new column.



Part1^ Part2^

For the vertical part of the subroutine it was quite similar to my horizontal function but instead of going across the rows I go across the columns.



The diagonal right and diagonal left are also quite similar to my check vertical and horizontal apart from a few diffrences.

The first is how they shift across and how I have to keep track of the x pos and the y pos when trying to find when it goes out of bound and to reset.

The second is that I have to change how much I should offset the original value by. For diagonal right the offset is equal to 32 but for offset left it is by 24.

