Tomer Levi

txl160430

Dr. Nhut Nyugen

CS 3340.501

30 November 2017

Sean Jasper, Tomer Levi, Ruiqing Song, and Aren Taylan’s Reversi Project

1. Description of the Program.

Our Reversi Program works by modifying a board stored as multiple separate bytes. The board is initialized at the start of the program, and every time a move is made, some of the board is changed. A move is made by the player by typing in row and column number. After the player declares a move, the program will check through every direction to see if there are any possible connections. If there are, the move is made and the board is updated as such. If not, the move is thus invalid and the player’s turn is skipped. After the player’s turn is over, the AI player will make its move by iterating through the board until it finds a valid move, and will then make it. After both player’s moves have been completed, the code checks if the game has completed (a win or lose condition has been reached) and if not, it is restarted, the board reprinted, and the whole process starting over again.

1. The Challenges experienced.

The biggest challenge we encountered was making the logic checks actually work the way we wanted them to. Every other problem was solved surprisingly fast, but the logic checks and updating the board took a while. We took precautions to make sure we wouldn’t get confused with the different $t# values, since there were so many, but we ended up mixing them up a little anyways and having trouble with needing to reset them after they’ve been used.

1. What you have learned by doing the project.

I learned how to navigate logic gates and loops using MIPS. Since I took charge of the logical elements of the game, I learned about how to manipulate the code to get the results I needed. I also learned about the importance of resetting your variables and about jump conditions. Most of my work was a culmination of previous homework assignments and lessons that we were taught in class. It was less about learning one thing, and more about how to make the several hundred lines of code work together to get the results I needed.

1. Discussion of Algorithm and Techniques used in Program
2. The way we store and edit the board is by directly manipulated bytes stored in the memory location denoted by “board”. It is initialized at the beginning of the program with the correct number of starting pieces already o the board. “\_” representing an empty space, “X” representing a player piece, and “O” representing an AI opponent piece. It is used for multiple different checks in the program, as well as when a move is determined valid or invalid.
3. Our method of determining whether or not a movie is valid uses multiple different checks. First, when the user inputs a coordinate, the program checks to make sure that the user wants to place an X on an empty coordinate. For the logical statements, first the program checks that the space the player wants to place an X on has an adjacency to a character that is not an X in each of the 8 directions. If it has an adjacency that is not X, then it checks to see if there is an O between the X you want to insert and another X in each of the 8 directions to sandwich that O. If these two things are true, then an X is placed where the user wants it to be placed, and every character in between the new X and the one that sandwiches the O is turned into an X. And vice versa for when the AI wants to insert an O.
4. The AI functions begin with an iteration. It iterates through the memory bytes that contain the table and uses the previously mentioned directional logic checks to determine the first empty space that’s available to make a move. Since the board is stored like a 1-dimensional array, the AI will choose the first empty space that’s a valid move with the lowest column number and the lowest row number. If 1-1 and 1-2 are both valid moves, the AI will place its piece in 1-1, even if 1-2 is a better move.
5. Contributions of Each Member.

Sean:

Sean was in charge of getting the board to work. He was able to display the board and get it to update correctly. He was also in charge of getting the AI to work, which he did by switching the values of X and O and then sending inputs to the logical statements.

Vivi:

Vivi was in charge of setting the win and lose conditions. She made a loop to iterate through the board and count the number of X’s and O’s. She also helped plan out the board and helped with the thought process behind the logical statements.

Tomer:

I wrote out and debugged all of the logical statements that made the code work. I wrote the statements to check for valid moves in all eight directions and for multiple different scenarios.

Aren:

Aren was in charge of the win and lose sounds, as well as doing the project report and user manual. He also provided the logic and methods of the AI, which Sean later built.

F. Suggestions (Optional.)

YouTube video: <https://youtu.be/tyCA5V1YiPg>