CS 200: Computer Organization

Project 9: Maze Generator

Shariq M. Jamil

Due: Monday, April 21, 2014

Overview

Purpose

This project required us to write a program in MIPS assembly language that recursively generates a maze.

Approach

I followed the project guide and looked through the skeleton Assembly code provided to find the TO DO sections. Changing the header was quick and easy. Then I put together an algorithm that would return a random number. This number would be used as a seed for the next time the routine is called. This went under the Rand function and I had already built this function for Project 7 but I had to refactor the code to get it to work for this project. After this, I fleshed out the routine for XYToIndex, which converts a two-dimensional pair into a single-dimensional index. The next algorithm was IsInBounds, which checked to see whether the X and Y values were within the constraints of the grid. For this part, I was not sure how to return true or false values and ended up choosing to return 0 for True and 1 for False. The last function I wrote out was PrintGrid. This was by far the most satisfying one since I got to see the grid generate and building the rest of the functions had been highly test-driven. I wrote tests for each function to ensure that they worked together correctly and this took up the brunt of my time on this project. I think this practice produces code that is more bullet-proof.

Solution

Sample Output

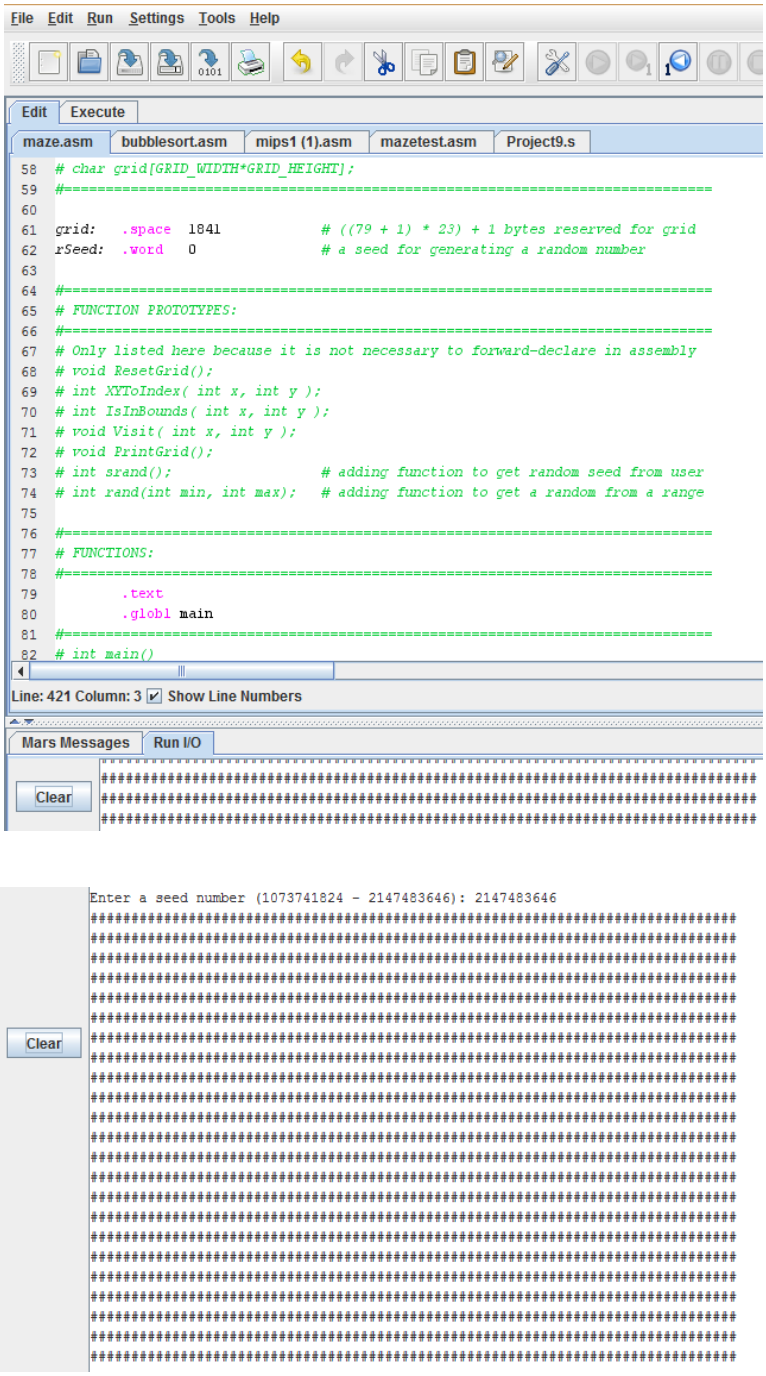

Figure 1: Sample Output

Conclusion

This project was a great exercise in test-driven programming in assembly. This project was open ended as to what the final output would be so I had to come up with tests for each function to ensure correctness of the implementation. The functions flowed together and the numbers generated were within bounds so my code worked. I am excited to write the Visit function out and work on the next phase of the project.

I also got more practice in writing branches and differentiating between la, li, lw and sw. I think the key to learning Assembly is right there in the in-depth differences between functions. Learning more about each function and how it works with memory really deepened my appreciation for the language.

References

MIPS Architecture and Assembly Language

http://logos.cs.uic.edu/366/notes/mips%20quick%20tutorial.htm

Understanding the Stack

http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Mips/stack.html