Joshua Pollock

CS 200 Project 10

28 April 2017

Joshua Pollock

CS 200

Project 9 – Maze Part 2

28 April 2017

**Purpose:**

This project is continued from Project 9. This week we will be implementing the visit function of the program. This is the function that will generate the actual maze output for the project. We are given skeleton code written in C and will be basing the visit function off this. Later in the week, Professor Kelly will release skeleton code for Project 10 which will correct any flaws in our main or any other functions. We were given the C code that would be needed to implement visit, the skeleton code that will be provided will only fix our mistakes from Project 9.

**Research:**

The C code proved to be enough to go off, and like project 9, I did not do much research into this project. The MIPS reference sheet came handy when I needed to use AND and shift right logic for some of the equations. I did look up a few things about the frame pointer and stack pointer. There were a few helpful links that I found and will be listing below. I have worked with the stack in the past but was still uncertain with some of it’s features.

<https://softwareengineering.stackexchange.com/questions/194339/frame-pointer-explanation>

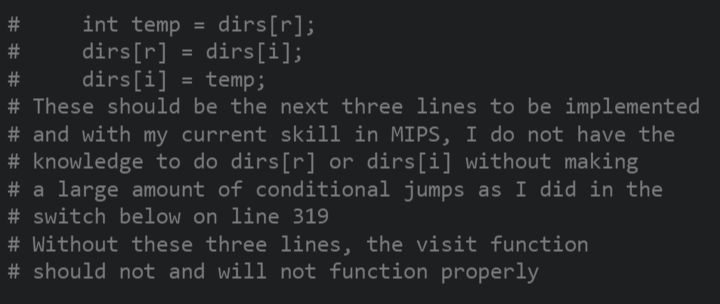
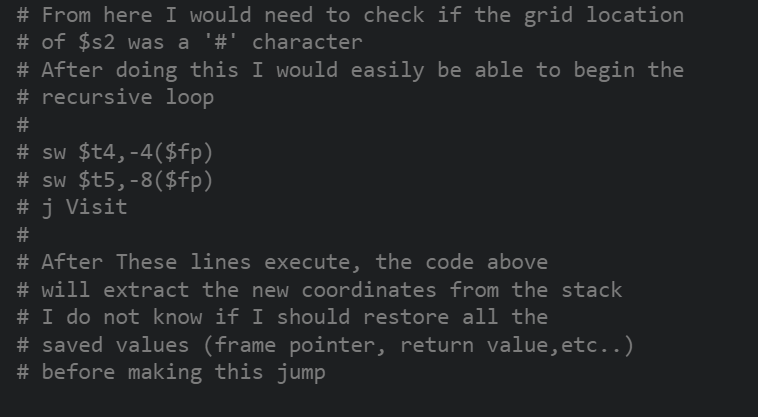
<https://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Mips/stack.html>

<https://www.quora.com/What-is-the-difference-between-a-stack-pointer-and-a-frame-pointer>

After looking at these few things, I began typing away and attempting to solve the problem.

**Program:**

In the end, I was unable to get my program to function properly. This is due to my lack of knowledge in specific areas. Below will be screenshots of the comments I put in place of the code I could not figure out.

Currently the only difference between my code here, and in project 9 is that it now overwrites the first ‘#’ with a space. Its not much, but if these two areas were finished, I believe the code that I have created woul function as intended. There were some areas where I got myself confused by using $t# registers and $s# register interchangeably. I have learned that this is a very bad idea and that I should use the $t# registers until they fill up. Below will be a screenshot of my code’s current output.



**Conclusion:**

Overall, my program does not function. There were parts that I could not figure out and commented what I thought I should do at that location. For example, I did not know how to do dirs[i] or dirs[r] which was necessary for the Visit function to work properly. Another hiccup I ran into was later in my code where I needed to check if the next position was a ‘#’ symbol. I did not know how to check this, but I was able to understand what I needed to implement and commented what I should do after checking for this condition. Looking back after completing this project, I wish I had done further research into indexing on MIPS. I am used to Java, Python, and C where you can just call list[position]. In MIPS, this is not the case. In MIPS, you must implement a few lines of code to achieve the same affect. Luckily after asking for help from Jacob Kaufman, I could at least make the first # of the grid a space.

Like a few projects back, the downfall to this project for me was time. I do plan to complete this MIPS code in the future as I find MIPS to be a very fun language. This week was filled with troubleshooting my Project 9 code and trying to start implementing the code required for project 10. In the end, I simply ran out of time to complete the project in its entirety. I commented where I was unable to complete about what I would intend to do at that location. This two part project greatly tested my ability to code in MIPS assembly. I wish I had more time to finish this project in its entirety and will be completing it in the next few weeks.