Project 8: Bubble Sort

CS 200

Jennie Butch
11/16/2022

Project Overview

Purpose:

For project 8, we are creating another program in MIPS assembly, but this time we are trying to create a bubble sort. For bubble sort, we are basically rearranging the numbers to be in the correct numerical order. When the user puts in their input, it can be any random number, and then the program will sort it out correctly. We do want to make sure the user input has a certain limit so we are doing between one and ten, anything outside of that will result in an error.

Approach:

My approach for this project was much different than the previous project. This time I started out by typing out the labels that were mentioned in the project 8 overview, the ReadNums, BSort, PrintNums, and the main. I did this so I could read further into the overview which mentioned what I should have in each section which was very helpful in giving me a good start on this project.

After that, I typed in my data variables since I knew I wanted to have the IntPrompt, which is the user input of how many numbers they want to sort. Then the NumberPrompt is the part where the user inputs the number they want to be sorted. Error, to give the user an error message for providing a number outside of the sorting range. A space to provide a space between the array numbers when they are done sorting. The last was the ArrayOutput where it is going to print the bubble-sorted numbers.

From there, I typed in my comments before writing actual code so I could break down my thought process into how I wanted to do each part of the project. While in the process of doing that I realized I was going to need more labels outside of the overview, so I started simply with ReadingTheCount which it is going to process the number the user put in for the IntPrompt that way it can read the number and decide through two different comparisons if it is going to hit the ErrorMessage label. On the inside of the ErrorMessage label, I would print the string for error whenever down the line it would branch out to it. If it did not need to branch out to the ErrorMessage, it would then save the number and then return it to the user so they can enter the numbers they want to be sorted.

Then it is going to ReadNums since that is the next label to be read through. I started off that section with a bunch of comments so I know what I labeled each register as and how the iteration is counting for two of the registers, then simply load in zero. From there, I added a loop that is going to read the numbers that the user inputted and count how many more times the user needs to enter a number. So the first I did was the iteration count so the system is counting what number the user is at, then I made sure to do a display in front of the NumberPrompt with the count of what number the user is on. From there it is going to read the user input and save it into an array. After saving it to the array it is going to iterate the address and then loop again until the user has entered the right amount of numbers to be sorted. Then we go outside of that loop and return to the user again.

After the ReadNums is the BSort, where it is actually going to do the bubble sort. I did the same thing I did in ReadNums which was to make comments on what I used for each register. After that, I initialized the one and NumberArray with their own registers and then made a branch loop. The branch loop is going to go through the swap conditions so that if the number is zero then no swap will occur and then one so then a swap would occur. After the small branch loop, I jump to ArrayBubble instead of ArraySwap if there is no swapping occurring.

When there is a swap we jump to the next label which is the ArraySwap this is where the swapping is actually occurring. The ArrayBubble is going to do the actual comparison of if the first number is smaller or bigger than the number next to it. If it is smaller then it is going to continue to the next index but if it is bigger then it is going to be sent to ArraySwap to actually be swapped. Once that is done it is going to finish out the bubble sort by using the label BExit, the BExit is going to return to the user once again.

Then we head over to PrintNums which is the part where we get our sorted array to show. We first load in our NumberArray which tells our program that we want to load ten numbers that the user can choose from to be sorted then load in our one and ArrayOutput so that this time to get the “Sorted Array: ” to show then syscall it. Then form the PrintLoop so we can actually print out our arrays that have been sorted and have a space between them after we insert the space we do an iteration to make sure every number shows. Below the iteration, we make another branch loop so that the program knows that there are no more numbers to be shown so then the loop would stop then otherwise it is going to repeat printing the number and space. Then finish off with the return again.

The main function is basically putting all of our labels together so they can run in the correct order. Start by doing the iteration count and load in our NumberArray again and then get the main program to read the numbers with ReadingTheCount and ReadNums. Then have the actual bubble sort (BSort) occur and then have the results display (PrintNums) then register $ra to end the program.

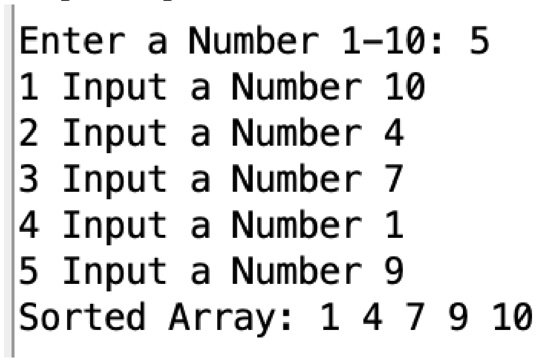
Results

This project had a lot more breaking down my thought process than the previous project. I think it is because we had to use more labels and data variables to get our program to work. We were successful in getting the bubble sort to work correctly and display the right numbers in order. It was difficult to know which labels go in what order and what needed to be included in the labels. Like I knew what I wanted inside the labels, it was more about if I needed a loop inside of them to get the program to do a certain function.

During this project, I basically wrote in pseudo code what I needed to accomplish in each label and had to rearrange it over time. For example, I thought I could get away with having the error message inside of my ReadingTheCount since I knew I wanted to do the comparison on the inside. But that is not how branching works so after testing I realized that and then I made my error message at the top. The thought process on the bubble sort had more of a breakdown also since I thought it was going to be simple enough to get away with it as one function but then I thought about how there are technically two cases to bubble sort being that the number is either going to move or going to stay.

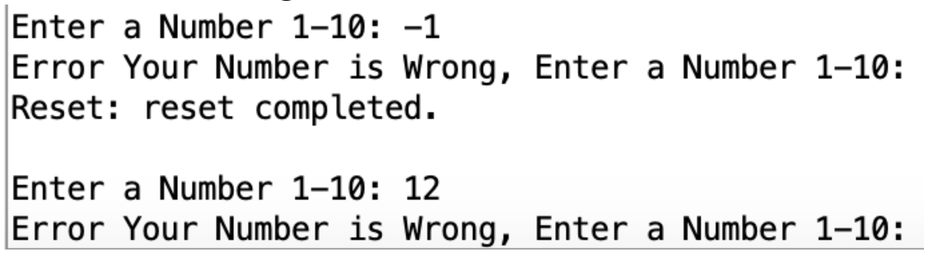
Once I finish the program I started to test it out. First, by simply did a five-number sort and it worked, I did that a couple of times and it still worked. Then, when I tried to enter a number outside of the range of input, which would show the error message but then my ArrayOutput was showing it was empty since it cannot process that number which was not good so then I typed out another version. Not only did I move my ArrayOutput to fix that problem but I also changed out my saving array since I did not save correctly and did not notice that until I went to office hours. After I made those changes my code works perfectly now.

Sample Output:



```
Enter a Number 1-10: 5
1 Input a Number 10
2 Input a Number 4
3 Input a Number 7
4 Input a Number 1
5 Input a Number 9
Sorted Array: 1 4 7 9 10
```

Correct ErrorMessage:

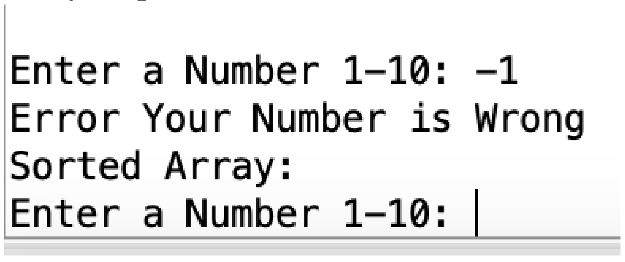


```
Enter a Number 1-10: -1
Error Your Number is Wrong, Enter a Number 1-10:
Reset: reset completed.

Enter a Number 1-10: 12
Error Your Number is Wrong, Enter a Number 1-10:
```

It does take user input, I just want to show both ways.

ArrayOutput Error:



```
Enter a Number 1-10: -1
Error Your Number is Wrong
Sorted Array:
Enter a Number 1-10: 
```

Conclusion

The main thing I took away from this project was that I should try to break down every single step in a more simple way. Since I should be in the mindset that the MIPS assembly language is truly one line at a time. I felt like I could have saved more time if I would have just written my comments a bit further than I did for this project just because I kept having to rethink my plans at certain points in the project. For example, in the bubble sort, like I mentioned, I knew that I wanted to do a comparison but I did not think about how I needed to separate out what each choice is going to do.

Outside of breaking down my thoughts line by line, I do feel like project 7 was a great help for this project since it was a great starting point for knowing what I needed in my data variables and how labels work. Also, I found this project to be great practice with loops since we had a lot more loops in this project than in the previous and it was used in a different way which I found to be cool.

The real challenging part of this project was the ArrayOutput problem that I had because I could have made it more complicated than I needed to since my original thought was to either made another if statement to determine when the ArrayOutput was going to show or simply rearrange my PrintNums as a whole into a different part of the program. Luckily, I asked a fellow classmate if they had this problem and they said no and looked over my variables just to find out that I needed the ArrayOutput at the bottom of all my variables. Additionally, I did not save my arrays correctly as I mentioned before but I fixed that by adding a loop into my ReadNums so that it can store correctly and flow much better than my original version.

My project_8_v1.asm is the WORKING FILE. My original version does not work correctly.