Jonathan Golden

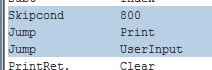
Levi Sutton

CS 3505L Fri @ 1:25 pm

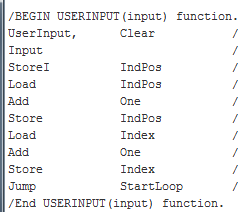
Dr. Ji Li

Final Project: MARIE Array Creation and Sorting

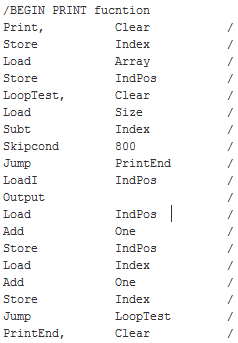
Our proposal for the project was to create a MARIE program which would take decimal value from the user to create an array with the size of that value. The program would then enter a loop to receive more decimal inputs from the user to populate the array. It would then enter a second loop that would print the values in the order they were entered. This should include some sort of boundary checking. The program would then sort the array from greatest to least. It would then reverse the sort from least to greatest. After both sorts, the arrays would then be printed to the control screen. One of the problems we ran into was that the system would not run correctly when mixing an array of hexadecimal values with inputs of a decimal value. To correct this, we changed the entire thing to hexadecimal types. As we were coding another issue we found was that it became easier to keep the program moving using strict conditional controls. Some of the controls will be listed below to help understand the program.



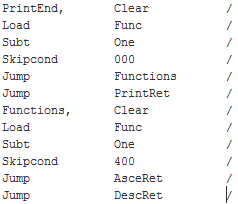
Using a conditional statement, then having the true or false sections jump to a separate labels help keep the confusion down. This way once the conditions are met program counter will follow only the designated fork.



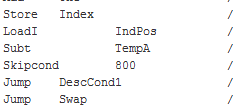
This section is the section that takes in the user input. It then stores the input at the address location of IndPos. It will load the memory address to the AC, increase it by one, then save the new address to IndPos. It will then jump back to the StartLoop label.



This is the printing section of code. It will take the initialize Index to 0, and IndPos to the memory address of the Array. The skipcond is testing for an out of bounds Index value that will cause it to end the print. Once it passes this condition, the program will step through each element of the array, printing the value stored in the memory for each element.



This section of code is not relevant to the printing function, but allows for reusability later on. The Func holds a value that when the value is less than 0 will cause the program to jump to the print return section. If the value is equal to 0 it will jump to the DescRet label. If it is greater than 0, it will go to the AsceRet. These are all the locations in the program that the PC should go to after printing the values in the array.



This part of the code is the comparison of two elements of the array. When the first element(TempA) is subtracted from the second element(AC after loaded with the value at the memory address held by IndPos). When it is greater than zero, then a swapping of elements will be needed. If it is zero or less, then no swapping is needed. For the ascending section of code, the skipcond just needs to be changed to 000, or the jumps can change positions in the program.

The swapping section of the code used two temp variables to hold the information. This was due to the ability to only work on one item at a time with the register. Then there is a section where we had to increment the IndPos to get to the next element, and then decrement the IndPos to get to the previous element.

