Aislinn O’Connell

CSS 422

Final Project Summary

**Implemented:**

**\_heap\_init:**  I implemented the \_heap\_init by creating a loop. I started by initiating the first value to 0x4000 and post-indexed the address by two to ensure that the initial value was not overwritten. I then cycled through a loop 512 times and set the other values in the memory address to 0x0. I post-indexed all values throughout the loop by two to account for the space that each byte would require.

**\_kalloc:** I created the \_ralloc function based off the code in heap.c and then set the starting point in the \_kalloc function to ensure that the recursive function is called upon. The recursive function, \_ralloc, is supposed to iterate through the heap and check to see if there is space available for the size of the data that is attempting to be stored in the heap. The parameters for \_kalloc are passed in through the R0 memory addresses. I utilized function calls to recreate the if-else loops used in the heap.c code. This way I didn’t have to try to fit in all the statements with the ITT block. I created several function calls that broke down the if-else statements into the basicest forms possible, then I would compare (CMP) values against each other to determine which function calls that are needed. I used these function calls to mimic the recursive function the best I could from heap.c.

**\_kfree:** I created the recursive \_rfree that would recursively search the heap and delete and reallocate the space at the memory address passed into the function. I used function calls once again to recreate the if-else statements that were present in the C++ variation of the code. These functions were broken down into small parts that at the end would see if the values were such to move to the next if-else statement or if it would break out of the loop and either return to the \_rfree function or \_kfree.

**svc.s:** In the svc.s function, I initialized the values of \_systemcall\_table\_init and the positions in which these values resided in the memory. Then I compared the value within the system call number with the corresponding values for \_sys\_malloc and \_sys\_free. If the numbers match, then the desired function will be called upon and the instruction set will branch to the new location.

**Missing:**

I believe that there is nothing missing in my code.

**Diagram #1:** The diagram illustrates the relationship between the calls generated in main and the progression of events through the usage of \_kalloc and \_kfree in heap.s.Diagram

Description automatically generated

**Diagram #2:** This is the diagram from the midpoint submission that illustrates the flow of the code in heap.c.

Diagram

Description automatically generated