James Day
6/9/2022

CSS 422 Final Project Summary

Project Summary- The purpose of this project is to implement a few C standard library functions for an ARM M3 cortex processor. These functions include kinit, kalloc, kfree, bzero, strncpy, malloc, free. This project aimed to teach C to assembler argument passing (APCS). Another large portion of this project is the Buddy memory allocation algorithm which is implemented within malloc. In the following section you will find a more detailed description of each part of the project. The C standard library functions that are implemented allow for a user to request a given size of memory with malloc. The functions created in this project will also allow the user to deallocate a section of memory that was created with malloc by giving a pointer to a specified address.

What was implemented-

Reset_Handler- The code within the reset handler is the code which is run after a reset of the CPU. All that was added within the reset handler was a simple call to initialize the system call table as well as a call to kinit which is described below.

SVC_Handler- The SVC handler was another rather easy portion of the project. The SVC handler just calls the system jump table and changes to user stack. I also had to add a few lines of code to return from the SVC handler.

Systemcall_table_init- This function initializes the system call table by putting the address of the code for each system call in the correct portion of the system call table. The code is rather simple as well and simply adds the address for each entry in the system call table.

Systemcall_table_jump- This function is called by the SVC handler and stores the system call in the 7th register. First I had to convert the call number to the address of the correct entry in the system call table. Then I call the respective function with the converted address.

Kinit – This function is called from the reset handler and simply sets up the memory control block which will be used for the buddy system memory allocation. This function clears the entire section of memory for the memory control block and sets all the values in it to 0. Then it sets the value of the first address of the memory control block to 16k which is the max size of the heap. After completion of this function, we return to the reset handler which calls main.

Kalloc / ralloc – Kalloc was the main section of this project. Kalloc begins by calling ralloc with the left and right mcb addresses being at the very top and bottom of the mcb. Ralloc attempts to move left and right until it is able to find a section of memory that is the right size for allocation. Each left and right movement is done with a recursive call which changes the parameters of Ralloc respectively. If ralloc attempts to allocate memory to a section that is in use, the function will return and work its way back up the call stack until a section of memory is found that is big enough and is not in use. When a section of memory is allocated the parent blocks will also be updated to the correct size of memory that they can store.

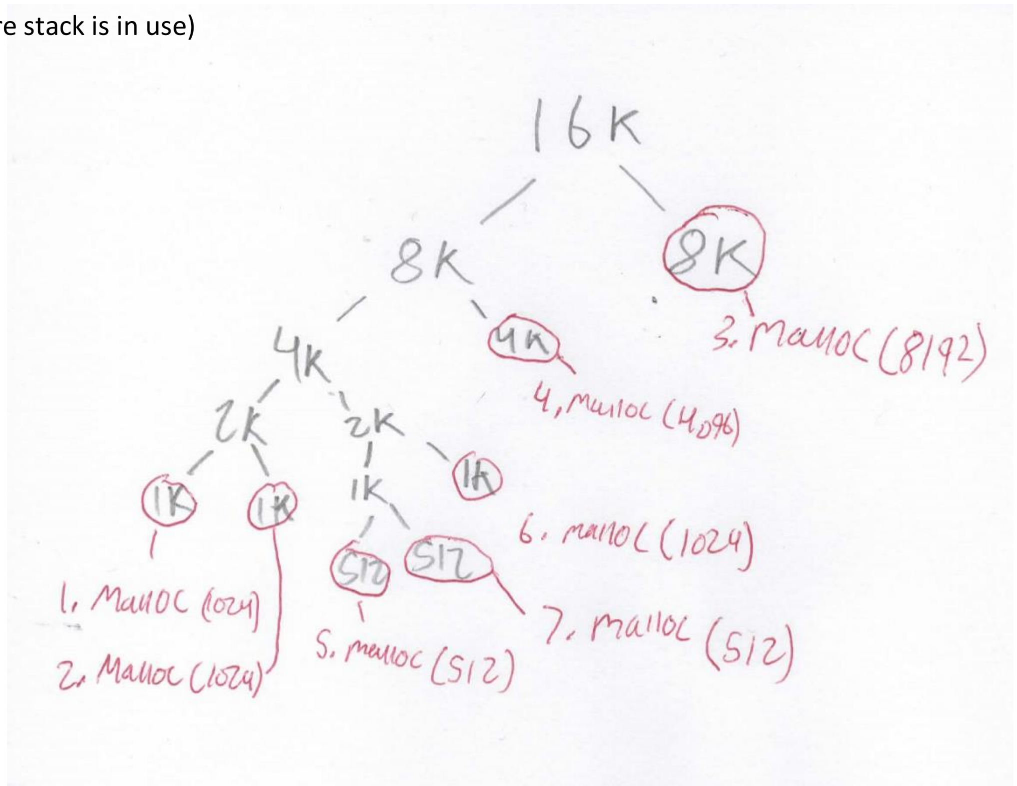
kfree / rfree- kfree is a function that is used to deallocate the memory which is allocated in kalloc. Kfree will begin by calling rfree. rfree will check if the address attempting to be deallocated is a left or right buddy. Once the location of the buddy is determined it will set that section of memory to not in use. If a section of memory is deallocated that has a buddy that is also no longer in use, than these portions of memory will be combined. After these sections of memory are combined, I recursively call rfree with the same address if it is a left buddy, or I call rfree with the buddy's address if we are a right buddy. This is to ensure that sections of memory continue to be combined if possible.

bzero / strncpy / malloc / free – All four of these functions were provided following the completion of part 1. Bzero simply writes a specified number of zeroed bytes to a given string. strncpy copies a specified number of characters from a source string to a destination string. malloc allocates a given size of bytes in memory and returns a pointer to the allocated memory. free deallocates a portion of memory specified by a pointer.

What was missing-

There is nothing that is missing from my final project to my knowledge.

Diagram- Shows tree where memory is allocated to. (Since bottom row is all in use, we know that the entire stack is in use)


16K
8K
8K
3. Malloc (8192)
4K
4K
4, Malloc (4096)
2K
2K
1K
1K
1K
1K
6. malloc (1024)
512
512
1. Malloc (1024)
2. Malloc (1024)
5. malloc (512)
7. malloc (512)