Nicolas Gioanni
12/1/2023

CSS 422 Final Project – Part 4 Documentation

Objective: Understand CPU operating modes (user and supervisor mode), system call handling procedures, Implement buddy memory allocation.

Project Summary: This project focuses on implementing key C standard library functions for an ARM M3 cortex processor. The functions involved are kinit, kalloc, kfree, bzero, strncpy, malloc, and free. A significant aspect of the project is teaching C to assembler argument passing (APCS). Additionally, the Buddy memory allocation algorithm is an important component implemented within the malloc function.

Implemented:

- Reset Handler:
  The Reset Handler is the code executed after a CPU reset. It includes a call to initialize the system call table and invokes the kinit function.
- SVC Handler:
  The SVC Handler manages system calls, invoking the system jump table and switching to the user stack. Additional code ensures a proper return from the SVC handler.
- Systemcall_table_init:
  This function initializes the system call table, placing the address of each system call in the corresponding table entry. The process involves adding the address for each system call.
- Systemcall_table_jump:
  Called by the SVC Handler, this function stores the system call in the 7th register. It converts the call number to the address of the correct entry in the system call table and calls the respective function.
- Kinit:
  Initiated by the Reset Handler, Kinit sets up the memory control block (MCB) for buddy system memory allocation. It clears the MCB section, setting all values to 0, and designates the first MCB address with a 16k heap size.
- Kalloc / Ralloc:
  Kalloc, the primary focus, initiates by calling Ralloc with left and right MCB addresses at the top and bottom. Ralloc recursively moves left and right to find suitable memory for allocation. If a section is in use, the function backtracks until an available section is found, updating parent blocks accordingly.
- Kfree / Rfree:
  Kfree deallocates memory allocated by Kalloc. It begins by calling Rfree, which checks if the address is a left or right buddy and marks the section as unused. If both buddies are unused, the memory sections are combined. Rfree recursively calls itself to continue combining memory sections when possible.

Missing: Nothing to my knowledge is missing.

1. I have provided a zip file of my Keil uVision Project with heap.s that has a correctly implemented _heap_init, _kalloc, and _kfree along with their helper functions _ralloc and _rfree.
2. I have provided 17 Execution snapshots, 16 snapshots after every malloc and free operation, and a snapshot of memory addresses for all variables mem1 to mem8 matching the reference snapshots.
3. I have provided a one page summary of what I have implemented in my project.
4. I have provided a self-evaluation and my expected grade out of 40 based on the referenced criteria and justification above.

CSS 422 Final Project – Part 4 Self-Evaluation

I spent a lot of time on this project, working to implement C standard library functions for the ARM M3 cortex processor. Functions like kinit, kalloc, and kfree were carefully crafted to work together smoothly. The highlight of the project is the successful integration of the Buddy memory allocation algorithm, especially in the malloc function.

The code is clear and to the point. I made sure each function, especially bzero and strncpy, shows a good understanding of the ARM M3 cortex processor. Comments in the code explain each part, making it easy to follow.

I stuck to the C to assembler argument passing guidelines and organized the code neatly for better readability and maintenance. Considering the criteria, I expect a high score, maybe around 38-40 out of 40. The project meets its goals by implementing C standard library functions effectively, especially with the Buddy memory allocation algorithm. Possible improvements could be minor, like optimizing parts or adding a few more comments for better clarity. Overall, I'm confident in the project, and it matches the expectations well.