Rithvik Aleshetty: rra76
Minhesota Geusic: mtg110

User-Level Memory Management Library Report

1. The virtual memory (vm) functions were implemented using:
    a. an array of char pointer to serve as our main memory
    b. an array of pde_t pointers to serve as our outer page table. The pde_t is just a node containing extra meta-data such as the base virtual address that was chosen to go onto this outer page table, and then we had the array of pte_t itself.
    c. An array of pte_t inside each of the pte_t_node served as our inner table and stored the physical pages that were allocated by myalloc.
    d. 2 arrays of chars served as our virtual and physical bit map, telling us which pages were occupied and which were not.

   Part one of the project was done by Minhesota, and part two was done by Rithvik. All of the functions below contain mutexes where they are required.

   SetPhysicalMem()
   - Calculate the virtual page count and physical page count
   - Calculate our offset bit count
   - Then allocate the memory using malloc
   - Allocate the page directory
   - Create the bit maps
   - Called the function to create tlb which initializes the tlb and sets the values inside it to null

   Translate(pde_t *pgdir, void*va)
   - Perform some error checks
   - Using our variables we retrieve the page table at the given variable outer
   - The page table is a node and we grab the virtual address

   PageMap(pde_t *pgdir, void *va, void *pa)
   - Extract values and create node using our Get3DecimalOfBin(va) function
   - Do some error checking
   - Check if a page table exists
     - If not, we allocate the page table and map our address
     - If there is, we go to and create the mapping

   Myalloc(unsigned int num_bytes)
   - Call setPhysicalMem only if it has not been set already
   - Map virtual address to physical address
   - Return the virtual address

   Putval/getval(void *va, void *val, int size)
   - Find physical page in our loop

Rithvik Aleshetty: rra76
Minhesota Geusic: mtg110

- Check the TLB to see if a result comes up there
- If there was no result found there, then use translate function
- Perform the copy or read

Myfree()

- Error check
- Free the page table entries starting from va
- Mark the pages free in bitmap

2. For part one, the benchmark outputs were 850, 717 microseconds based on two tests. After including the TLB, the miss rate achieved was 75 misses on 400 accesses for a miss rate of 18.75%. The runtimes in part two were 814 microseconds and 767 microseconds on two tests. They were lower but after running a lot more tests, they are very close and no conclusion is clear.

3. Our program will support different page sizes in multiples of 4k with a smaller page table. The page directory will change as well.

4. The possible issue with our code is when user gives a size to myfree that is less than the original size they've requested in myalloc, as in the system will free the requested size, but leaves the remaining bytes occupied without a way to clear them. Also, because of issue of 0x0 outer, we have chosen not to use the 0^{th} index, and thus lose out on the space.

5. The most difficult part of solving this project was correlating the virtual memory to physical memory.