Jacob Hessong

MP3 Design Document

The focus for this machine problem was to implement a page table manager that handles page frames for memory in a machine with x86 architecture. This is done with a two-level table where each entry in the table is divided into three sections. There are two kind of tables, one which is the page table itself and another that serves as the page directory.

Handling a page fault is done by first checking if the fault is in the directory or in the table. To do this, the entry from the page directory whose index is returned from the cr2 register is checked. If the fault is in the directory, a frame for the page table must be allocated from kernel memory before the actual page is pulled from the process memory. This requires a few shift operations to be implemented.

A few other functions had to also be implemented for the page table manager to function properly. The first of these is init_paging(), which simple initializes the private member variable of the PageTable class. Next was the constructor. This first created a table for the memory that would be directly mapped, which was determined by the shared memory size. These pages are all marked as present in memory, with the rest marked as not present. After this was the load() function, which loads the current object of PageTable to the current_page_table member before writing its address to the directory using the write_cr3() function. Finally, the last function is enable_paging(), which sets the enabled_paging member to true and marks the cr0 register with the specified value.