Implementation of the Page Table

Vishruth Raghuraj Gollahalli

1. **Implementation of init_paging()**
   This function was implemented by initializing the variables.

2. **Implementation of PageTable() constructor:**
   For this function I have used the logic used in http://www.osdever.net/tutorials/view/implementing-basic-paging article. First, the page directory address was created by first requesting a frame of size PAGE_SIZE using the get_frames function from cont_frame_pool. This page directory address was stored in the `page_directory` variable. Similarly, the base address of the first page table was created using the same get_frames function and stored in the `page_table` variable. Next, for each entries in the page table, the entry was marked as being “present” by marking the present, read/write and user/kernel as being present (set to bit 1). The address was moved by 4kB as that is the size of the page.
   Once the page table entries were marked, I added the page table base address into the first index of the page directory and set this address's present bit as 1. For all other page directory entries I marked the present bit as 0. The page table constructor is now ready.

3. **Implementation of load():**
   Load function was fairly simple to implement. I wrote the address of the page directory into the cr3 register (which holds the page directory address in x86). Now the page directory was loaded.

4. **Implementation of enable_paging():**
   To enable paging, I first set the paging_enabled variable by assigning value 1. Finally I wrote into the cr0 register the value 1 to enable paging (setting paging bit to 1).

5. **Implementation of handle_fault():**
   The implementation of this function was done in three major steps.

   Step 1: Retrieve the faulty address:
   In this step, I first read the cr2 register to retrieve the address causing the fault. From this address I retrieved the first 10 bits for the page directory index and middle 10 bits for the page table index by doing bit shifting.

   Step 2: Resolve page fault in the page directory:
   If the page directory entry is not present, I first request a frame for the new page directory entry (a new page table address). Using the faulty page directory index from step 1, I insert the page table address into the corresponding page directory index. Setting this address’s present bit to 1 makes this entry “present”. To finish it off, I marked

all the page table entries for the page pointed to by the page directory entry as not being present.

Step 3: Resolve page fault in the page table:

First, I check if the page table entry is valid, if it is not, then I can go ahead and address the page fault. If the page table entry is not present, I first request a frame for the new page table entry. Using the faulty page table index from step 1, I insert the newly created page table entry into the page table at the corresponding index. Finally, I mark this address as set by marking the present but as 1.