ID: 5140379041
Name: Jiacheng Ma
Document for OS Lab2

Part 1: Physical Page Management
After the system is booted and paging is enabled, we need to manage physical pages to make it easy for future system programming. Firstly, we write a function boot_alloc() to allocate physical memory when the system is just booted, before the structure to store free pages is setuped. In function page_init(), we set up the page_free_list, which is used to store the information of all pages. We only mark pages from 1 to npages_basemem and from the next page that boot_alloc is about to allocate to the end of physical memory free, since the rest of memory is dirty and will never be used. page_alloc() is used to allocate a page, and it will choose the first page in page_free_list and set the page to zero page if the ALLOC_ZERO bit is set. page_free() frees a page and insert it into page_free_list. page_free() is only called by page_decref(), which decrease the page ref count and call page_free() when the count become 0.

Part 2: Virtual Memory
boot_map_region() and boot_map_region_large() is implemented here to map a range of physical address to virtual address. They use pgdir_walk() to walk the page to find or create PTE, and set the range to a range of physical memory. boot_map_region_large() will use large page, so the PSE bit of CR4 has to be set to 1.
Since now we can manage the physical memory, we need to manage the virtual memory of a page table too. pgdir_walk() is used to walk a page, find the direction of PTE of a specific address, it will search the given page table, and if there's no such entry and the create flag is set, it will create that entry and return it. page_insert() insert a page into the given virtual address of a page table and set the page bit as given, it use pgdir_walk() to find or create PTE, and then set the entry to the physical address of the given page. page_remove() will set the entry of a page to NULL and call page_decref() to decrease the page ref count of the page. page_lookup() return a page that is related to the address.

Part 3: Kernel Address Space
After we have all these above functions, we will initialize the kernel address space. In function mem_init(), we map pages, which store all the information about pages, to user at liner address UPAGES, and map bootstack to [KSTACKTOP-KSTKSIZE, KSTACKTOP), which is the stack of kernel. Then we map the whole PA to VA above KERNBASE. Since the range is too large and may cause overflow, we does not use boot_map_region or boot_map_region_large. Then we set the PSE bit of CR4 to 1 to enable page table extension, and load the kernel page table to hardware.

Change 2:
In page, PPN is 20 bit, and in cache, label is 18 bit, which means we can use the last 2 bit of PPN as color to improve the cache performance. Here alloc_page_with_color() is implemented, to alloc a page with a specific color, if there's no page like this, it will return 0. This function also scan the

page_free_list, and when find a page with the give color, it return the page, until the end of the list. When it comes to the end of the list, the function return NULL.