Program Design

In this machine problem, based on my previous programs for managing the physical frames, I design the programs to implement memory management for a single process. The address space for a process is 4GB, pagetable entry size is 4B, so the format of the process address for the page table can be

10bit	10bit	12bit

The size of physical memory is 32MB, and the first 4MB is kernel memory and the remaining 28MB is for the user process. In this assignment, the pagetable is stored directly in the physical memory, not in the pagetable, and I will put the page table into kernel memory pool.

The page directory has 2^10=1024 entries, its size is 4KB, 1 page. The memory size managed by a page table is 2024 * 4KB = 4MB. And the first 4MB is directly mapped, and the remaining 28MB is free-mapped, depending on the demand and page table management. So the first entry in the page directory and its corresponding page table exactly fit the first 4MB.

The physical memory frames in the remaining 28MB will be allocated for the process on demand, that is happening in the process of page_fault. So when page_fault happens, I can first locate its corresponding page directory entry, is the entry is empty, then I will load a page table for it, and the page table is just 1 page allocated from the physical frames. Then we can further go into the entry of the page table, and write the allocated frame number and its corresponding attribute bits, like r/w, present and stuff.

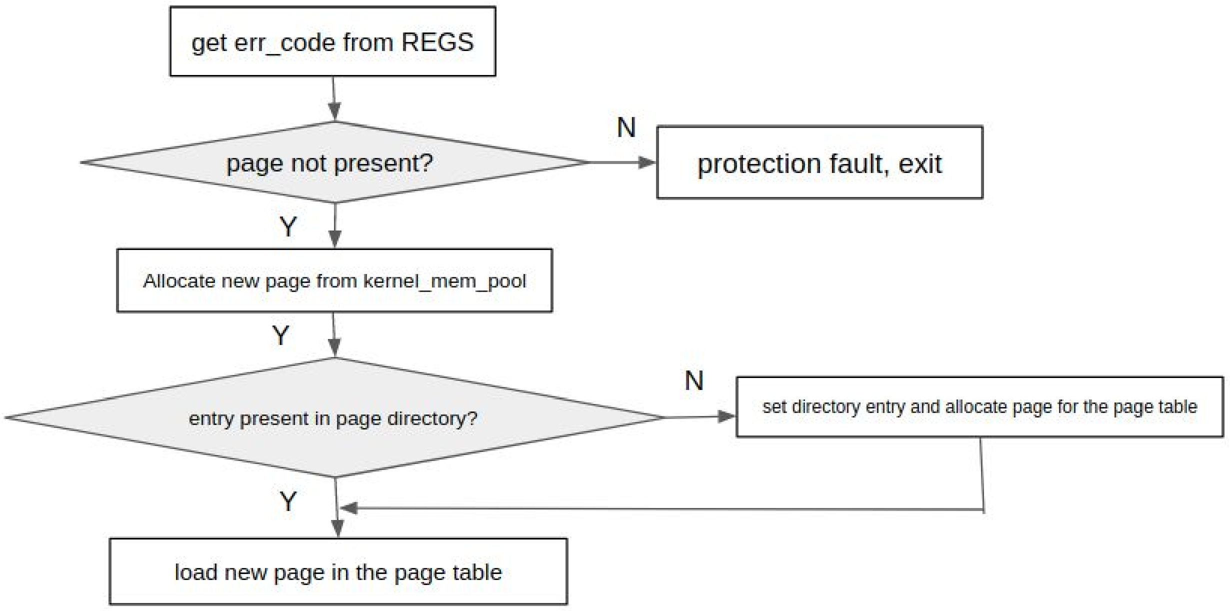
Implementation in program:

PageTable()
Get memory frames from kernel pool to store page directory and page table pages.
Since the first 4MB is directly mapped, we could initialize the first 4MB pages in the page table.

load()
Set the current_page_table, and at this moment also write the page directory address into CR3, so that during context switching, the OS could load the page directory address of the new process from CR3.

enable_paging()
Set paging_enabled, and also set the CR0 enable_paging bit.

handle_fault(REGS * _r)


get err_code from REGS
page not present?
N
protection fault, exit
Y
Allocate new page from kernel_mem_pool
Y
entry present in page directory?
N
set directory entry and allocate page for the page table
Y
load new page in the page table