MP 3 Document

Following the page_table.H design, I implemented the below:

- Page_table constructor
- init_paging
- load
- enable_paging
- handle_fault

Page_table constructor

- Get 1 frame to be PD with page size
- Init PD with not present bit
- Set up page tables for shared region
    - Calculate the number of page tables
    - Create page tables
    - Map for shared region directly
    - Init page tables with present bit

init_paging

- Initialize kernel memory pool
- Initialize process memory pool
- Initialize the size of shared region (4MB)

load

- Set the current page table as active
- Load PD into CR3

enable_paging

- Set PG bit to turn on paging
- Set local paging_enable to be 1

handle_fault

- Check fault address first
    - If fault address is within shared region, this should not happen, just return
- Calculate PD and PT index with shifting
- Handle two types of scenarios
    - Missing PT
        - Get an new frame for the new PT
        - Init all entries to be not present
        - Update PD entry

        - Missing page
            - Get an new frame from process memory pool
            - Update PT entry to map the new frame

Memory Layout

- Uses 4KB pages (PAGE_SIZE)
- 1024 entries per page table/directory (ENTRIES_PER_PAGE)
- Two-level paging structure:
    - Page Directory → Page Tables → Physical Pages
- Virtual address split: 10 bits (dir) + 10 bits (table) + 12 bits (offset)

Others

- 0x2: Supervisor, Read/Write, Not Present
- 0x3: Supervisor, Read/Write, Present