MP 4 Document

1. PageTable Implementation

- PageTable Constructor
  - Allocate Page Directory (PD):
    - Get 1 frame from the kernel memory pool to serve as the Page Directory.
    - Initialize all entries in the PD to "Supervisor, Read/Write, Not Present" (0x2).
  - Set up Shared Region (first 4MB):
    - Calculate the number of required page tables for the shared region.
    - Allocate frames for each page table from the kernel memory pool.
    - Initialize page tables with direct mapping (virtual address = physical address).
    - Set entries in page tables to "Supervisor, Read/Write, Present" (0x3).
  - Recursive Mapping:
    - Set the last entry of the PD to point back to itself, enabling recursive page table lookup.

- Handle_fault
  - Check Fault Address:
    - If the fault address is within the shared region, this indicates a critical error (should not happen).
  - Calculate Indices:
    - Compute Page Directory (PD) index and Page Table (PT) index from the faulting address.
  - Handle Missing Page Table:
    - Allocate a new frame from the kernel memory pool.
    - Initialize all entries in the new page table to "Supervisor, Read/Write, Not Present" (0x2).

- Update the corresponding PD entry to point to the new page table.
- Handle Missing Page:
  - Allocate a new frame from the process memory pool.
  - Update the corresponding PT entry to map the new frame with "Supervisor, Read/Write, Present" (0x3).

- Register_pool:
  - Registers a virtual memory pool with the page table.
  - Maintains an array (registered_pools) of pointers to registered pools.

- Free_page:
  - Frees a page by releasing its frame back to the process memory pool.
  - Marks the page as invalid (0x2) and flushes the TLB.

- PDE_address and PTE_address:
  - Compute logical addresses of PDE and PTE using recursive mapping.

- Flush_tlb:
  - Flushes the TLB by reloading the CR3 register.

2. VMPool implementation

   - VMPool Constructor
     - Initializes a virtual memory pool with:
     - Base logical address.
     - Size of the pool.
     - Pointer to the process memory pool (for physical frames).
     - Pointer to the associated page table.
     - Initially, the entire pool is marked as a single large free region.

    - Registers itself with the page table.

- allocate
    - Allocates memory regions lazily:
    - Rounds requested size up to the nearest multiple of page size.
    - Searches for a sufficiently large free region.
    - Updates allocated and free region arrays accordingly.
    - Returns the logical start address of the allocated region or 0 if allocation fails.

- release
    - Releases previously allocated memory regions:
    - Finds the allocated region by its start address.
    - Frees all pages within the region via the page table (free_page).
    - Moves the region back to the free regions array.

- is_legitimate
    - Checks if a given logical address is within any allocated region.
    - Used by the page table during page fault handling to verify legitimate memory accesses.