MACHINE PROBLEM 4

OBJECTIVE - The objective of this machine problem is to implement a simple virtual memory allocator.

OVERVIEW-

We need to implement the below functions in vm_pool.C

1) VMPool() -> constructor for the vmpool class.
2) Unsigned long allocate() -> Allocates a region of _size bytes of memory from the virtual Memory pool.
3) Void release() -> Releases a region of previously allocated memory.
4) Bool is_legitimate() -> Returns FALSE if the address is not valid.

We also need to correct our implementation of page_table.C and add 2 more functions(1st and 2nd).

1) register_pool() -> register pool with the page table.
2) free_page() -> vm pools call this to free a page.
3) load(), handle_fault(), PageTable() -> updated

page_table.H was modified to add a linked list of VM pools.

IMPLEMENTATION -

VMPool::VMPool() - constructs the vm pool object by initializing all the variables/pointers.
VMPool::allocate() - uses a region_list named struct object with members base_address and size to maintain the list of regions.
VMPool::release() - gets the argument _start_address of the region to be released, function checks which region in region_list has the same base_address and if found removes it using free_page function and also flushes TLB.
VMPool::is_legitimate() - checks if the address lies between base_address and base_address + size of the VMPool.

Page_Table::PageTable() - stored the address of page directory in the last entry of page_directory, for recursive page table lookup.
Page_Table::register_pool() - registers the vm_pool object in a linked list.
Page_Table::free_page() - calculates the frame number and releases the frame using release_frames function and flushes TLB.
Page_Table::load() - added the write_cr3() cmd to use the same function to flush TLB if needed from the vm_pool functions.
Page_Table::handle_fault() - function checks if the page_fault address is valid, modified how PDE and PTE are decoded and used.

TESTING- testing was done using kernel.C
Both page table and vm_pool are checked, screenshots are attached below.

```
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
DONE WRITING TO MEMORY. Now testing...
Test Passed! Congratulations!
YOU CAN SAFELY TURN OFF THE MACHINE NOW.
One second has passed
One second has passed
One second has passed
One second has passed
```

```
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
handled page fault
frames released
Loaded page table
frames released
Loaded page table
frames released
Loaded page table
frames released
Loaded page table
frames released
Loaded page table
Loaded page table
Released region of memory.
Test Passed! Congratulations!
YOU CAN SAFELY TURN OFF THE MACHINE NOW.
One second has passed
One second has passed
========================================================================
Bochs is exiting with the following message:
[XGUI  ] POWER button turned off.
========================================================================
```