Design Document
Virtual Memory Management and Memory Allocation

CSCE611
Operating Systems
MP4- Fall 2022

by

Tanu Shree

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

TEXAS A&M UNIVERSITY, COLLEGE STATION

This machine problem focuses on memory manger using various small tasks. These tasks are divided into three parts as suggested in the handout.

Firstly, the page table management is to be extended to support very large numbers and sizes of address spaces. Earlier it was done in directly mapped memory. These page table pages and directory are to be moved to into virtual memory. Secondly, Page table is to be prepared to support virtual memory. At last, we implement the Simple virtual memory allocator hook it up to new and delete operators of C++.

For part I, the recursive page table lookup is implemented to map the page tables. When paging is turned on, the CPU issues logical address and then there will be problem working with the page table access. The handle fault api is updated to access the memory. The logic is to recursively loop the address in page table and directory so that it accesses the fault address in correct manner. The last entry in the page table directory point back to itself.
To calculate the logical address, below two functions are used, which returns the address of the PDE or PTE.

```
unsigned long * PageTable::PDE address(unsigned long addr);
unsigned long * PageTable::PTE address(unsigned long addr);,
```

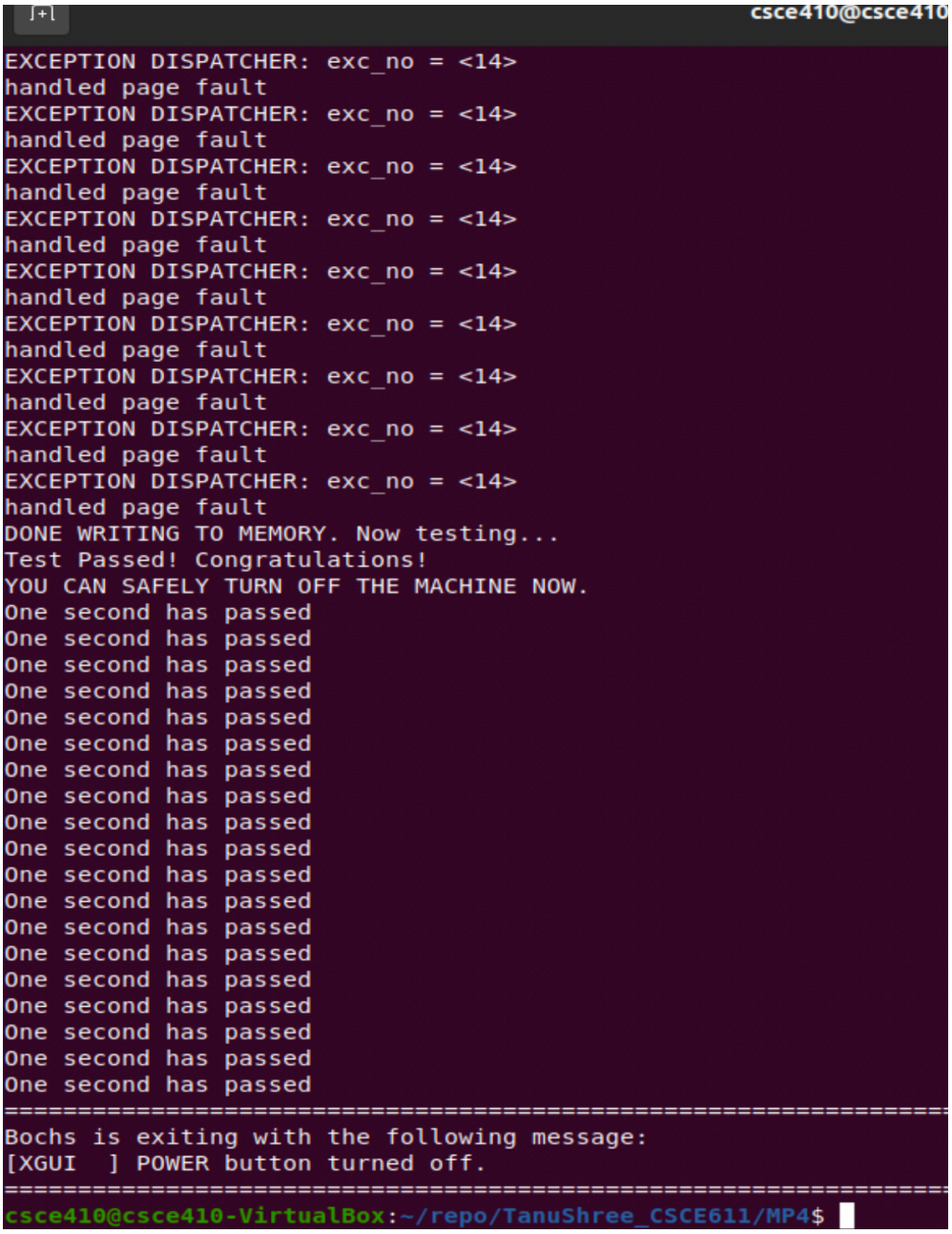
For part II, page table class is modified to add some extra APIs needed for virtual memory manager (register pool and free_page). The register pool API maintains a list of memory pools which have been created till now. Number of VM pools register are also maintained in another variable.
Next, the address is checked in the page fault handler. The fault in any address should be within the region of VM pool implemented. If it fails, assert and abort. Free page API calls the release frame API for the frame number passed as an argument. It releases the previously allocated pages and mark them invalid. TLB is flushed after marking the page invalid.

For part III, the VM pool class is implemented. Several APIs are implemented in this class to work correctly. First, the constructor is created in which variables and structures are declared to maintain the information about the allocated regions of the VM pool. List of allocated regions are maintained and counted. First page of the VM pool is used to store the list of allocated regions.
Next, Allocate API is used to mark the region asked for the allocation in the allocated list. First region is given to the page maintaining the list of VM pools. This API allocates a region of _size bytes in the virtual memory pool. If successful, it returns the virtual address of start of the allocated region of memory, else it returns 0.
In Release API, first it is checked that from which region it is taken. This is done by finding the base address. Pages allocated to this region are found and freed. After releasing the memory region, we update the allocated region list by modifying the list of entries and reducing the allocated count.

The last API, legitimate, finds whether the address given is within the limits of VM pool. If not, false is returned.

Testing:

Testing was done using the test criteria provided. First the testing was done to check the page table implementation. Later, the macro called TEST_PAGE_TABLE was commented and only VM testing was continued. Below are the screenshot for the same.



```
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
Loaded page table
Released region of memory.
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
Loaded page table
Released region of memory.
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
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
One second has passed
One second has passed
One second has passed
========================================================================
Bochs is exiting with the following message:
[XGUI  ] POWER button turned off.
```