Design Document
Page Table Management

CSCE611
Operating Systems
MP2- Fall 2022

by

Tanu Shree

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

TEXAS A&M UNIVERSITY, COLLEGE STATION

This Machine problem focuses on demand-paging based virtual memory system. The objective is to set up and initialize the paging system and the page table system infrastructure for a single address space, which can later be extended for multiple processes.

The total amount of memory in the machine is 32MB. 4 MB is reserved for kernel. Memory mapping was done in previous machine problem, MP2. Here, in MP3, the class PageTable provides support for paging in general.

Main changes were done in page_table.C ,and cont_frame_pool.H and cont_frame_pool.C were taken from previous machine problem. kernel_mem_pool is between 2MB and 4MB and is in direct mapped memory (physical and logical address is same). Process_mem_pool is above 4MB and is freely mapped memory.

The page table can be represented as an array of page table entries. Each entry is as follows:

31 … 12	11 … 9	8 … 7	6	5	4 … 3	2	1	0
Page frame	Available	Reserved	D	A	Reserved	U/S	R/W	Present

Following functions were implemented.

- **Init_paging**: This is used to initialize the paging system.

- **Constructor**: It is used to construct the page table object. This sets entries in page directory and page table. A frame is grabbed using get_frames from kernel pool for page directory and shared page table entries are marked valid(present). Non shared memory page entries are maked 'not present'. Last three bits are used to set kernel/user, read only/read & write and not present/present.

- **Load()**: Once the constructed created, current page table directory address is loaded in CR3 register using function write_cr3(). The page table is loaded.

- **Enable_paging**: To enable paging , first, register CR0 is read and then it is changed to 1 and paging_enabled is set to 1. Before enabling paging, we had to take care that page table is set up correctly.
  Page table set up was done and tested.

- **Handle_fault()**: This handles the page fault exception of the CPU. If the last bit in page table entry is set invalid, i.e. page not present (bit set to 0), an available frame is brought in and page table entry is updated as valid. If new page table is to be initialized, a frame is brought in and page directory is updated. The lower 3 bits are interpreted as below:

Value	Bit2	Bit1	Bit0
0	Kernel	Read	Page not present
1	User	Write	Page present

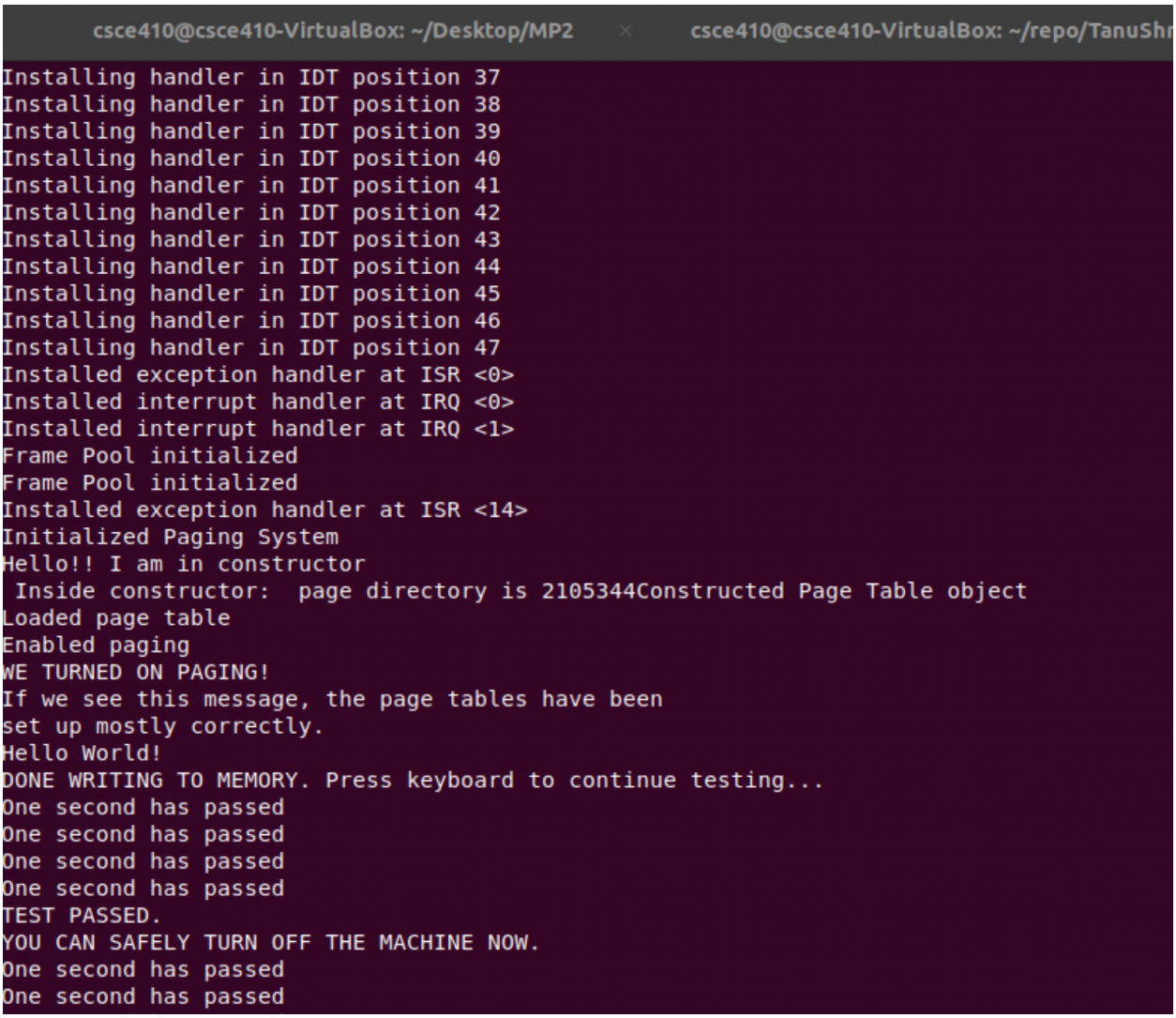
To handle fault, first the current page directory is read from CR3 and then the address of page fault is read from CR2. The corresponding page directory address (10 bits for page directory) and page table address (10 bits for page table) are taken out using masking. Rest 12 bits are offset.

If there fault in page table, i.e. page is not present, a new frame is brought in and page table is updated accordingly. The address to this page table is the Page directory entry of page fault address and the address to the new frame is the page table entry.
If new page table is to be initiated, a frame is brought in to first allocate memory to page directory entry (PD address in page fault address) and then another frame is brought in to allocate the memory to page table(PT address in page fault address).

Testing:

While testing the initial page setup system crashed first. After debugging the code, I realized that there were some minor error in get_frames and release frames methods of cont_frame_pool. This was fixed and page setup was done successfully. For the rest of the program, the testing was done using the test criteria already provided. This was done successfully too while resolving minor errors here and there.

Output Screenshot:
The screenshot of the output screen is below:


csce410@csce410-VirtualBox: ~/Desktop/MP2
csce410@csce410-VirtualBox: ~/repo/TanuShr
Installing handler in IDT position 37
Installing handler in IDT position 38
Installing handler in IDT position 39
Installing handler in IDT position 40
Installing handler in IDT position 41
Installing handler in IDT position 42
Installing handler in IDT position 43
Installing handler in IDT position 44
Installing handler in IDT position 45
Installing handler in IDT position 46
Installing handler in IDT position 47
Installed exception handler at ISR <0>
Installed interrupt handler at IRQ <0>
Installed interrupt handler at IRQ <1>
Frame Pool initialized
Frame Pool initialized
Installed exception handler at ISR <14>
Initialized Paging System
Hello!! I am in constructor
Inside constructor: page directory is 2105344Constructed Page Table object
Loaded page table
Enabled paging
WE TURNED ON PAGING!
If we see this message, the page tables have been
set up mostly correctly.
Hello World!
DONE WRITING TO MEMORY. Press keyboard to continue testing...
One second has passed
One second has passed
One second has passed
One second has passed
TEST PASSED.
YOU CAN SAFELY TURN OFF THE MACHINE NOW.
One second has passed
One second has passed