Design Document MP4 Operating Systems CSCE 611

Chinmay Ajit Sawkar

Files changed in code:

1. page_table.C/.H -
2. vm-pool.C/H
3. cont_frame_pool.C/H from previous MP

Implementation

In this MP we extend the previous page table implementation for large address spaces and also implement a virtual memory allocator which is used for registration of virtual memory pools and for checking the legitimacy of the logical addresses.

Part I: Support for Large Address Spaces

For large address spaces page entries cannot be mapped directly to the physical memory thus we use page tables and page directory. The CPU generates logical addresses randomly, thus to map the logical addresses to frames we use the recursive page table in which the last entry of the Page Directory is made to point to itself. We use the following functions to implement this in the Class PageTable

1) PageTable() : Constructor method for direct mapping of logical addresses, initializing the present bits and setting up a recursive page table making the last(1023) entry of the Page Table is made to point to itself.

2) load() :Used to extract the page directory index of the current object and store in PTBR register(CR3)

3) enable_paging() : used to enable paging by changing the CR0 register
4) handle_fault(REGS * _r) - Used to handle the page faults by mapping a free frame from Process frame pool to the faulty page using recursive page table strategy.

Part:2 Registration Of Virtual Memory Pools And Legitimacy Check Of Logical Address:

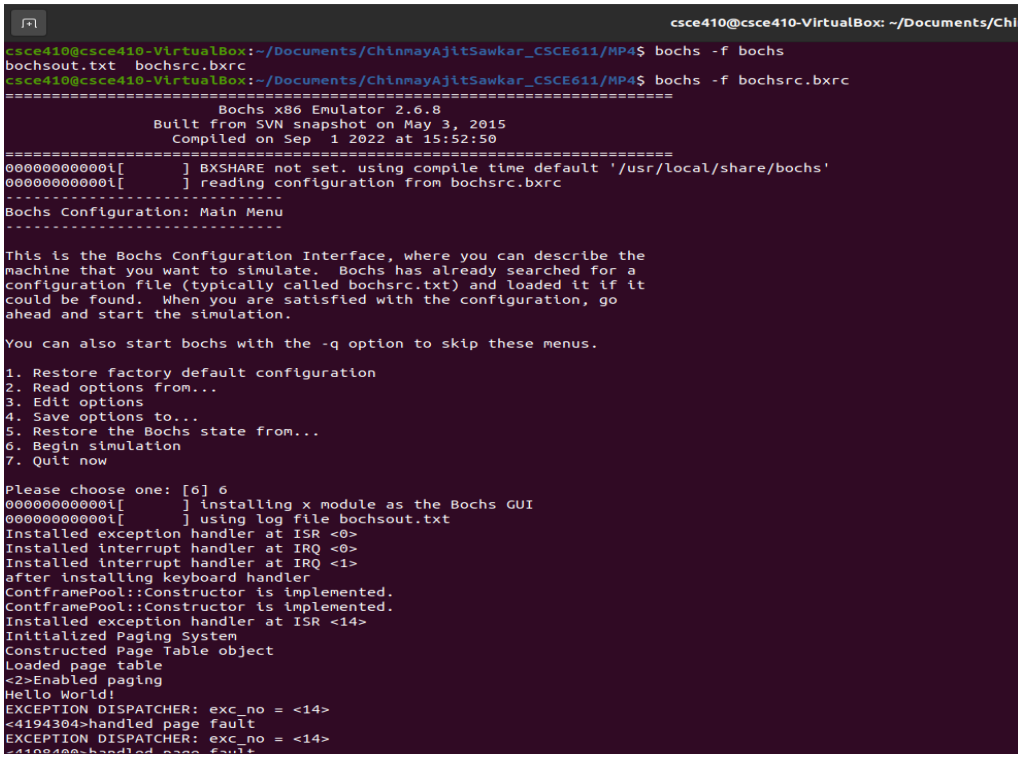
In this part we implement a virtual memory pool by maintaining a list of VM pools and checking whenever there is a page fault whether the faulty page is legitimate (within the bounds of base address and limit range). If legitimate then the page fault is handled

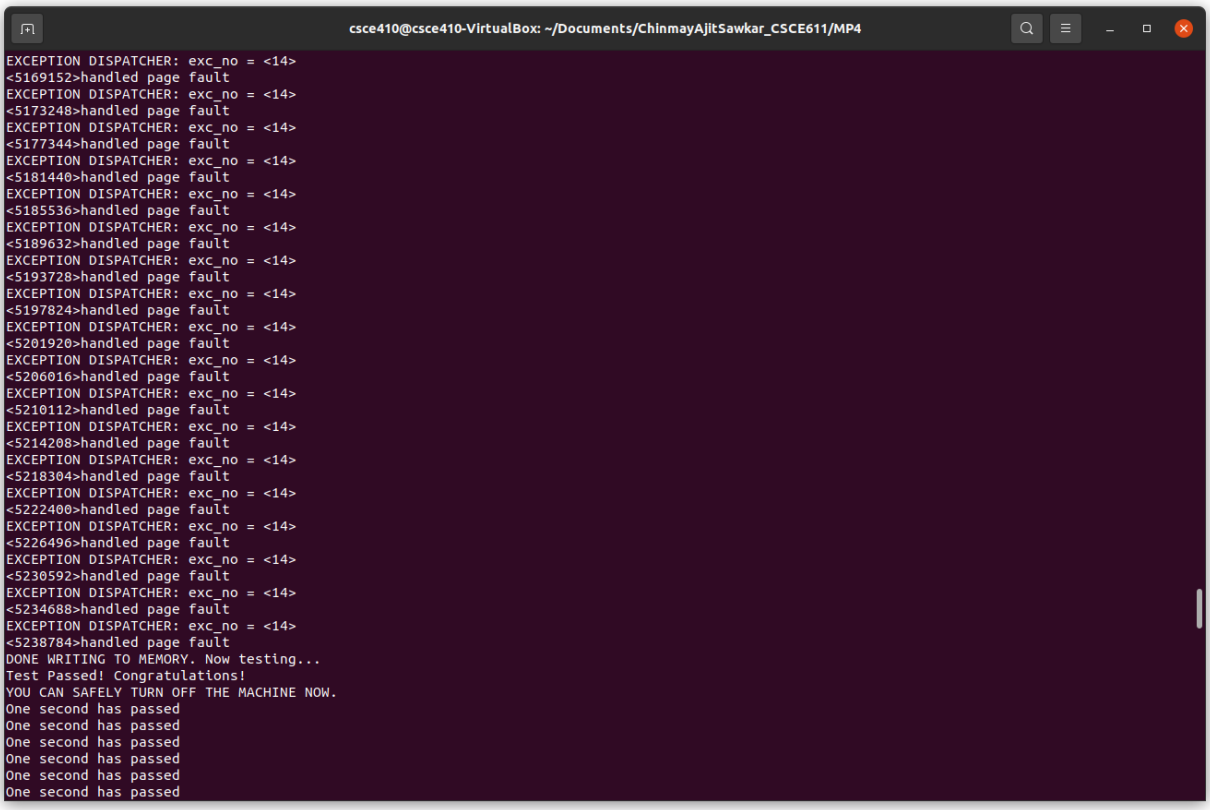
Part:3 Allocation And Deallocation Of The Vm_pools

We implement a virtual memory manager which is used to allocate and deallocate memory. It does this in size multiples of page entries in the virtual memory using arrays where arrays consist of class objects which store details of base address and length of VM pool assigned. Thus each region of memory is different array element and
In the allocate function, a region of the virtual memory pool is allotted. Once allotted it returns the start address of the regions from which memory was allotted. We also keep track of the remaining size of the pool and count of regions.

For deallocation of virtual memory pool regions we call the free-page function which calls the Release_frame_pool function from the ContFramePool Manager. As the frames allotted are released we also find the array element of the frames and mark invalid, flush the TLB and CR3 is reloaded.

Results
Attached below are the result logs.


csce410@csce410-VirtualBox: ~/Documents/Chi
csce410@csce410-VirtualBox:~/Documents/ChinmayAjitSawkar_CSCE611/MP4$ bochs -f bochs
bochsout.txt bochsrc.bxrc
csce410@csce410-VirtualBox:~/Documents/ChinmayAjitSawkar_CSCE611/MP4$ bochs -f bochsrc.bxrc
Bochs x86 Emulator 2.6.8
Built from SVN snapshot on May 3, 2015
Compiled on Sep 1 2022 at 15:52:50
00000000000i[ ] BXSHARE not set. using compile time default '/usr/local/share/bochs'
00000000000i[ ] reading configuration from bochsrc.bxrc
Bochs Configuration: Main Menu
This is the Bochs Configuration Interface, where you can describe the
machine that you want to simulate. Bochs has already searched for a
configuration file (typically called bochsrc.txt) and loaded it if it
could be found. When you are satisfied with the configuration, go
ahead and start the simulation.
You can also start bochs with the -q option to skip these menus.
1. Restore factory default configuration
2. Read options from...
3. Edit options
4. Save options to...
5. Restore the Bochs state from...
6. Begin simulation
7. Quit now
Please choose one: [6] 6
00000000000i[ ] installing x module as the Bochs GUI
00000000000i[ ] using log file bochsout.txt
Installed exception handler at ISR <0>
Installed interrupt handler at IRQ <0>
Installed interrupt handler at IRQ <1>
after installing keyboard handler
ContframePool::Constructor is implemented.
ContframePool::Constructor is implemented.
Installed exception handler at ISR <14>
Initialized Paging System
Constructed Page Table object
Loaded page table
<2>Enabled paging
Hello World!
EXCEPTION DISPATCHER: exc_no = <14>
<4194304>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>


csce410@csce410-VirtualBox: ~/Documents/ChinmayAjitSawkar_CSCE611/MP4
EXCEPTION DISPATCHER: exc_no = <14>
<5169152>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5173248>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5177344>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5181440>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5185536>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5189632>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5193728>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5197824>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5201920>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5206016>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5210112>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5214208>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5218304>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5222400>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5226496>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5230592>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5234688>handled page fault
EXCEPTION DISPATCHER: exc_no = <14>
<5238784>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
One second has passed
One second has passed