MP4 - Design Document

Changes

`page_table.C`
`page_table.H`
`vm_pool.C`
`vm_pool.H`
`Makefile`

page_table :

```
PageTable();

   1. Initialize page_table for shared memory in kernel pool
   2. Define page directory in process pool
   3. Recursive page table -> last element of page directory array is address
      of page directory itself and this element will be used to fetch the PD
      address in further functions
   4. Make entry for kernel page table in PD and make the page table directly
      mapped
   5. NOTE -> Paging is disabled

void load();
   1. Load the page directory in CR3 register
static void enable_paging();
   1. Enable paging -> bit 31 of CR0 needs to be set
static void handle_fault(REGS * _r);
   1. Fetch the page number from CR2 register
   2. Page directory is set to <1023><1023><0> because of recursive page table
      entry
   3. Offset of page directory (PDE) can be fetched by making the address
      <1023><1023><X> -> the MMU will recursively look in the same PD array
   4. Check if page table exists by checking if present bit is set
   5. If not make a page table in process pool and make the entry in  page
      directory
   6. If yes-> access PTE in recursive page table look up where virtual address
      will be in the form <1023><X><Y>
   7. Get a frame from process pool and make the PTE in the above location
void register_pool(VMPool * _vm_pool);
   1. Add the provided VMPOOL in linkedlist
void free_page(unsigned long _page_no);
```

(Using similar logic as handle_fault)

1. From page number fetch the PTE via recursive page table lookup from virtual address <1023><X><Y>
2. Fetch the frame number and from that get the frame number to perform release_frame(frame_no)
3. Reload the CR3 register to flush TLB

Vm_pool

Variables used

```
private:
  unsigned long base_address;
  unsigned long size;
  unsigned long updated_size;
  ContFramePool *frame_pool;
  PageTable *page_table;
  class vm_regions
  {
  public:
    unsigned long base_address;
    unsigned long length;
    vm_regions *next_region; // linked list to maintain the VM address regions
assigned
    vm_regions(unsigned long ba, unsigned long length) // VM regions assigned
as pages
    {
      base_address = ba;
      length = length;
    }
  }; // this maintains the region
  vm_regions *regions;
public:
  VMPool *vm_pool_next_ptr; // pointer to next VMPool -> public because it is
used by Page Table class
```

Constructor :

```
    base_address = _base_address;
    size = _size;
    frame_pool = _frame_pool;
    page_table = _page_table;
    updated_size = size;
    page_table->register_pool(this);  // registering the pool with the page
table
    vm_regions head_region = vm_regions(base_address,Machine::PAGE_SIZE);
    regions = &head_region;
    updated_size = updated_size - Machine::PAGE_SIZE ;
```

Methods :

unsigned long allocate(unsigned long _size);

1. Make an object of vm_regions by getting the base_address based on previous region's base address and length
2. Approximate the size based on number of pages required

void release(unsigned long _start_address);

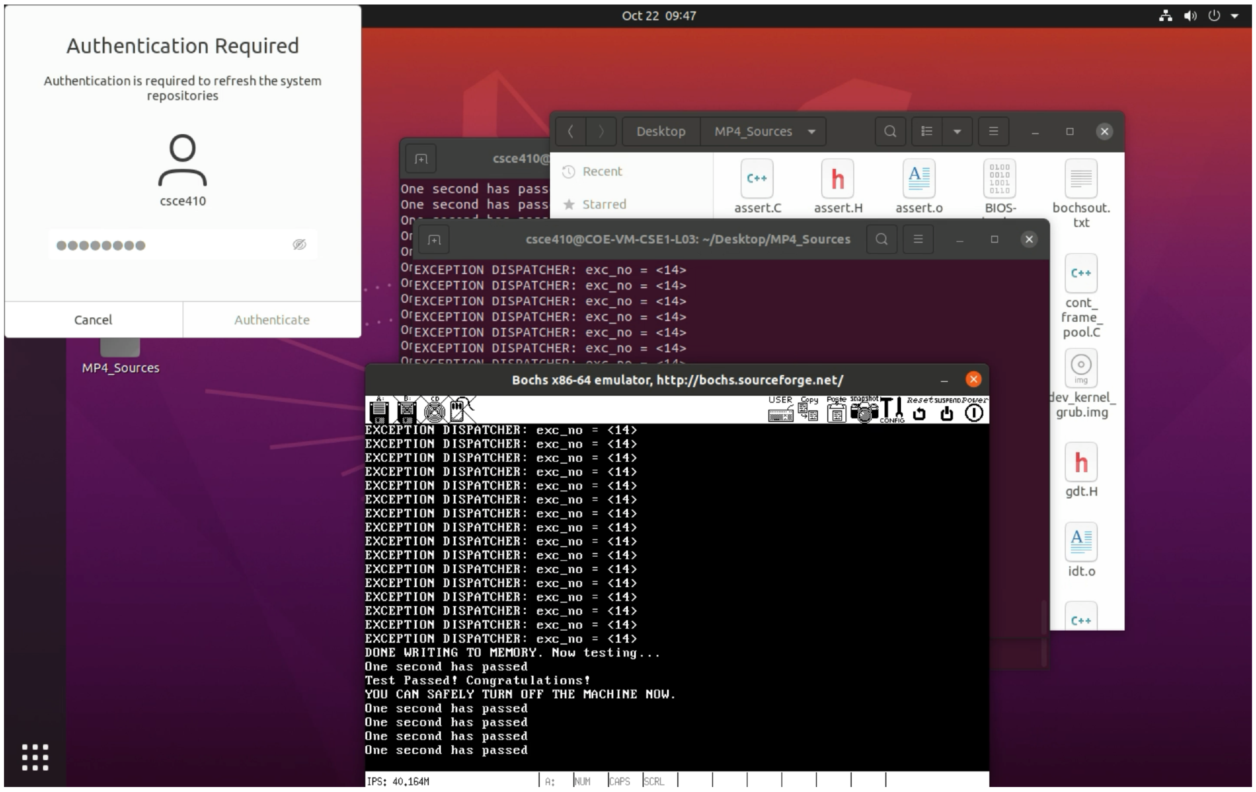
1. Remove the region from vm_regions linkedlist and update size
2. From _start_address till _start_address + length use free_page(address) method

bool is_legitimate(unsigned long _address);

1. Check based on base address and size if the _address is legitimate

Makefile is updated to make use of provided cont_frame.o directly. Thus removing gcc compile for cont_frame.C

Test Results :



```
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
EXCEPTION DISPATCHER: exc_no = <14>
DONE WRITING TO MEMORY. Now testing...
One second has passed
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

IPS: 40.164M
```