Aaron Sanchez
UIN: 228004762

<u>Necessary member variables:</u>

Struct Region {

unsigned long start_address;

unsigned long end_address;

}

Create a region data structure, so that we can hold the start and end addresses for each of the regions within the VMPool. Start to end address have to be multiples of page size.

static Region* regions: This is needed so that we can have multiple regions within the VMPools, as we allocate and remember the location, we need to have the address space, so that we don't allocate something else to this space. This is acts as our local table, that shows start address and end address for the VMPools.

unsigned long base_address: The base address for our VMPool.

unsigned long size: Size to check if region is valid.

ContFramePool *frame_pool: Holds the frame pool that is to be used by the VMPool.

PageTable *page_table: Holds the page table that is to be used by the VMPool. The use of this along with VMPool allows page_table to have access to all of the VMPools.

<u>Function Implementations:</u>

Constructor: The constructor will go through and assign basic variables, so using the arguments that are within the constructor, we will assign our page_table for the VMPool to the page_table that is given. We will do the same for the frame_pool, base_address, and size, this way we don't need to pass these values in constantly. We will then create an array called regions, that is at least of size size, so that we could store at least size Region. Lastly, we need to link the page_table to the VMPool, by using PageTable::register pool(VMPool * pool).

Allocate: Allocate simply has to go and grab the next region that is available, and give the space to a new Region that is within regions, for whatever _size we need. We could use multiple different ways of implementing this, one way is simply using an array that utilizes first fit, which could cause problems, but will probably be how I end up allocating. You have to make sure you add the start and end addresses to a Region and push the Region onto the regions array.

Release: Release is pretty simple, and needs to go and clear the Region that is at start address and clear the space until its end address. There is also the possibility that this Region could be in the pageTable, so we will check this, and if it is,we simply run page_table->free_page(addr), which will release from the pageTable. Once this is done, you need to get rid of the region from regions.

Is_legitimate: Goes and checks that the region that we are giving is valid, an invalid region would not be within the base_address to base_address + size.