Design Document - Operating Systems - MP2

Name: Anshul Sharma

=================================================================

I have only touched ContFramePool.c and ContFramePool.h.
In ContFramePool.c we have implemented all the mentioned api:

1. Constructor
2. get_frames()
3. mark_inaccessible()
4. release_frames()
5. needed_info_frames()

Before delving deep into the implementation of these apis, let us clarify some design ideas:

1. We will be needing 2 bits to represent a single frame because we have have 4 states here:

    00 → Free
    01 → Head of the frame
    10 → Inaccessible
    11 → Allocated

2. We will need a linked list to track the number of frame pools being instantiated. Whenever a new frame pool is constructed, the constructor code adds the new pool to this list. Whenever frames are released, the static function release frames first determines that frame pool that owns the frame, and then calls that frame pool's release frames function.

3. We will use one bitmap to track the allocated and free frames using the above mentioned states.

Now, lets deep dive into the implementation of each of the above mentioned APIs:

1. Constructor: we assign the passed values to the declared private fields(base frame number, number of frames & frame management information). We set the bitmap pointer to the correct address. And finally we need to update the linked list of the frame objects.
2. get_frames(): the main function of this api is to find out if the number of continuous frames are available in the frame pool. Here, we perform a linear search on frame pools and find out if the frame length required is available in the memory, if yes, we set the head of the frame(and mark the rest of frames as allocated) and return its address.
3. mark_inaccessible(): this is a basic api which marks the range of frames as inaccessible after checking if they belong to the frame pool.

4. release_frames(): this api is used to set the state to 00(free) of the allocated frames. We first need the find out the respective frame pool which holds the allocated frames which is done by traversing the static list of frame pools.
5. needed_info_frames(): here we calculate the total number of management frames needed for a given memory size.