Implementation of Frame Pool

INTRODUCTION:

The class ContFramePool allocates and deallocates both single frame and sequence of frames as requested. Pool of frames is managed by maintaining the status of frames as either free, allocated or head of the sequence. Head of the sequence means that it is allocated and it is first of the sequence of allocated frames. Rest of the frames in the allocated sequence that are not first are marked as allocated.

IMPLEMENTATION OVERVIEW:

This frame manager is implemented by using bitmaps. Bitmap is an array of data type 'char'. Each state (one of the three states: Free/Allocated/Head of the sequence) of the frame takes 2 bits to represent. Since char is 1Byte = 8 bits, each array element can hold the status of 4 frames. In this implementation '01' is taken as Free, '00' is taken as allocated and '10' is taken as Head of the sequence.
The class ContFramePool has the following functions:
1) get_frames – allocates sequence of requested frames if available or returns 0.
2) specific_release_frames – This is object specific release frame function which is called by a static release_frame function based on the range of the frames present in this particular pool. It Marks the sequence of frames starting from the given frame number as free until the traversal hits the head of sequence or a free frame.
3)static release_frames- This function is not associated with a particular frame pool. It identifies which pool the given frame number is from and it calls respective pool's release_frame function.
4) mark_inaccessible – marks the given frame number as head of sequence and remaining n-1 frames as allotted.
5) needed_info_frames – returns number of frames need to store management info of give number of n frames.

For the release of frames, only frame number is known but not the particular frame pool which this frame belongs to. This calls for a way where the frame pools can be identified even outside the pool. This is addressed by declaring a static list of pointers to pointers of all the pools. Every time an object is created for a frame pool, it's pointer is added to this list. This list is accessed by the static release_frame function to find the respective frame pool of the given frame number and thus call the specific release frame function.

CODE EXPLAINED:

1) General traversal in the bitmap:
   To find the required type of frame (free/allocated/head of sequence), first appropriate mask is created for those 2 bits and it is modified in both inner loops and outer loops. In inner loop, it covers 4 frames represented in an array element and then it goes to the outer loop where it moves on to the next array element.

2) Constructor:
   The following things are done in the constructor:
   -All the variables in the data structure of the class are initialized.
   -All the frames in the bitmap are marked as Free.
   -All the array elements are loaded with 0x55(01010101).
   -All the info frames are marked as allocated.
   -'this' pointer of the object is being added to the array of pointers of pointers to objects (list_pools) and the index of the array is incremented.
3) Get_frames function:
   While traversing across the bitmap as mentioned above, a free frame is looked for which when found, is made as the head of the sequence. The traversal is continued until the required number of free frames in a sequence is found. If any non-free frame is hit in between, the frame next to it is made as the new head of sequence and the logic repeats. This is repeated until required sequence of frames is found or the frames in the pool are exhausted. One the sequence if found, the head of the sequence is returned.
4) Specific_release_frames
   The index of the frame number(like ith element in the array where i is the index) is found from (first_frame_no - base_frame_no)/4 where first_frame_no is from the function call. The position of the frame in the ith element is found from (first_frame_no - base_frame_no) % 4.
   Once the position of the given frame is exactly determined, a check is done if that frame is a head of a sequence or not. If it is, the code proceeds to mark it as allocated (00) using appropriate masking. Now all the allocated frames till the next head of sequence or a free frame are marked as free.
5) Static release_frames
   This function is commonly called whenever a frame or sequence of frames have to be released. The array mentioned above – list_pools is iterated till the end. At every element in the array, the respective pool's base number and number of frames in that pool is obtained and check if the given frame number is in the range : base_frame_number – base_frame_number + number of frames . If the condition is met, the respective pool's release frame is called.
6) Mark_inaccessable
   It is implemented the same way as get_frames. The given frame is considered as head and its state in the bitmap is changed to '10'. The status of n-1 frames after that are marked as allocated by changing their status to '00'.
7) Needed_info_frames
   This function directly returns the number of frames need to keep the management information of this frame pool. This depends on the frame size and number of bits utilized to represent each frame. In this case, the frame size is 4kb and so it takes 4*4*1024 bits to represent 16384 frames. So the return value of the function will be (number of frames/(4*4*1024) + (number of frames % (4*4*1024) > 0 ? 1: 0)