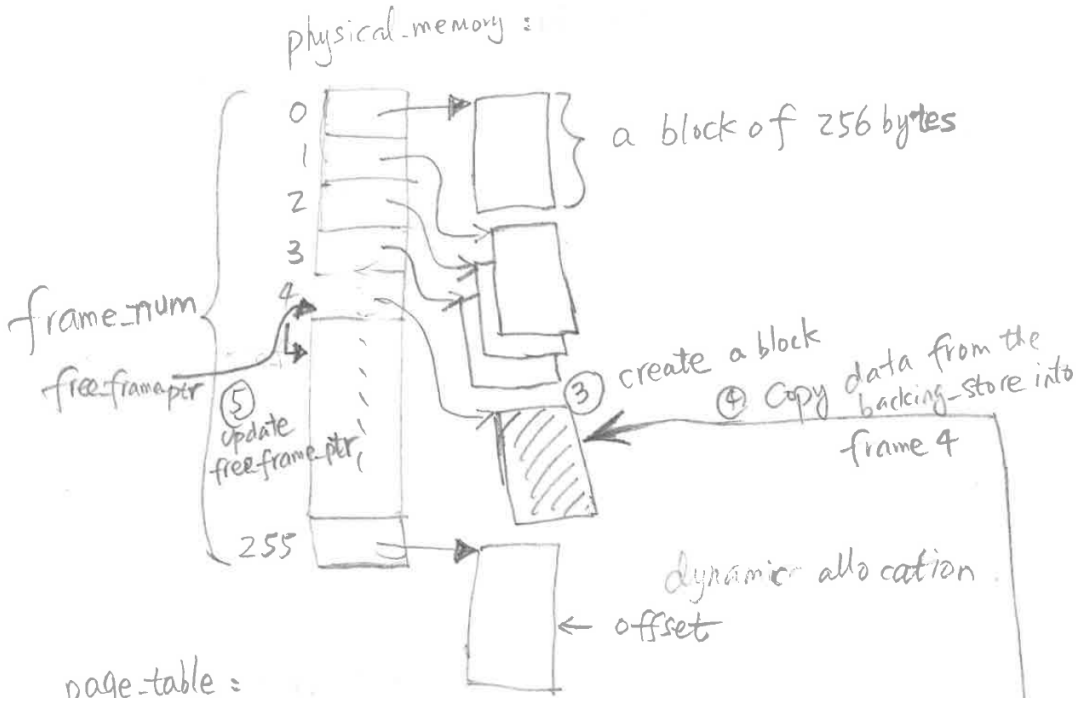
COMP3500 – Frequently Asked Questions

Project 5 – Virtual Memory Manager

1. I am using arrays for the majority of the data structures and for the main memory I am using an array of char arrays. The 256 bytes read from BACKING_STORE is placed in an array, which is kept in the main memory array. This way I can easily access any value by using the frame number for the index of the main memory array and then the offset as the index of the char array. Please let me know that this is not correct or does not make sense.

Answer: Your design sounds good to me. You may use a 2-dimentional array as the data structure of the simulated main memory. The simulated main memory is an array of frames, each of which is an array of bytes. There is no "byte" data type in the C programming language; data type "byte" is defined as unsigned char. Unlike "signed char", "unsigned char" represents values *ranging from* 0 to 255.

You may also choose to use dynamic data structure, where the simulated main memory is an array of pointers, each of which is pointing to a block of 256 bytes. See the draft below:



2. Is it true that the logical address and physical address for a given location will be the same?

   Answer: No. the logical address and physical address for a given location will not be the same. The logical address refers to the virtual space, whereas the physical address refers to the main memory.

3. If the page table and main memory are the same size wouldn't it be true that the page number and frame number would be the same and therefore the address be the same?

   Answer: The page number and frame number of a page are not the same, because a page will be dynamically loaded in any frame in the main memory. For example, page 3 may be loaded into frame 25. In this case, page number is 3 and the corresponding frame number is 25.

4. Is the correct way to access BACKING_STORE just simply multiplying the page number (or frame number) by the frame size (256) and then using fseek() to get to that location in the file and then reading in the next 256 bytes?

   Answer: Yes. Your idea is correct. Please keep in mind that you can not use page number in this process; rather, you will have to use frame number that is translated from its page number using TLB and page table.

5. If we encounter a tlb fault and then find the page in the page table do we move that entry into the tlb using a tlb replacement strategy? Or do we only update the tlb table when we have a page fault and read from backing_store?

   Answer: If there is a tlb miss and then find the page information in the page table, followed by moving that entry into the tlb using a tlb replacement strategy. A sample implementation can be found in the “Project 5 Put It All Together.pdf” document on Canvas. In other words, any TLB miss must lead to a TLB update.

6. When there is a page fault and the page table needs to be updated, should we be using the FIFO algorithm to update the table or is there another method?

   Answer: The page table in your VM system is large enough to hold all page-frame entries. Therefore, there is no need to enforce any replacement policy to manage your page table.