# CSCE 451/851 Programming Assignment 5 – Protips

Designing a Virtual Memory Manager

## 1 Part 1 simplified

Let's simplify.

- You're simulating memory with a 2D array.
  `char PHYSICAL_MEMORY[256][256]`
  Each row contains a full frame, 256 bytes
- The memory is byte addressable. Each address points to a byte in your memory (i.e., the 2D array called `PHYSICAL_MEMORY`).
- The page table is a 256x2 matrix just like the TLB, but holds all the valid pages.
- The TLB is a 16x2 matrix where the columns are the page and associated frame number. It is a smaller cache of the page table.
- Each address is a logical address, where the first 8 bits are the offset, and the next 8 bits are the page number.

Let's try to find the value of something stored at logical/virtual address 16916 (the first entry in addresses.txt).

1. Find the page number and offset. I recommend doing bitwise operations for this. In this case:
   ```
   page_number = 66
   offset = 20
   ```
2. Let's try to find this in the TLB.
   a. Search through the TLB.
   b. It's clearly not there since this is the first address and the TLB is empty.
3. Okay, so let's look in the page table.
   a. Search through the page table
   b. It's not there since, again, this is the first address and the page table is empty.
4. So the logical address we want is **NOT** in memory. We must, therefore, load it from the backing store and put it into memory.
   a. Find the page in the backing store by reading in a page sized chunk of memory at the right location (e.g., `page_number` × `page_size`).
   b. Where should we put the extracted frame? Put the entire chunk you read out of the backing store into `PHYSICAL_MEMORY` in FIFO order.
5. Now we have one frame in `PHYSICAL_MEMORY` that's associated with a page. In this case
   ```
   frame_number = 0
   ```
   Let's put this information into the page table so we know where it's at. You can add it to the page table in FIFO order. Let's also add it to the TLB which acts as a cache. The TLB is just a smaller version of the page table. So put:
   ```
   TLB[0][0] = page_number; // 66 for this example
   TLB[0][1] = frame_number; // 0 for this example
   ```
   And do the same for the page table.
6. Now I know where logical address 16916 is. So I look for page number 66 in the TLB. I can find it since I just put it in there. Page number 66 maps to frame 0 in `PHYSICAL_MEMORY`. I concatenate the frame (0) and offset (20) together to get a physical address of 20.
7. Now with a physical address, I can go to my `PHYSICAL_MEMORY` array and find the value:
   ```
   int value = PHYSICAL_MEMORY[0][20];
   ```

## 1.1 How do I update the TLB

Notice I don't need to worry about replacing any entries in the page table because physical memory is large enough to hold all the pages. So just fill up the page table in FIFO order.

But the TLB is only 16 entries. So every time I access a page that's **NOT** in the TLB I should update the TLB to include that page/frame mapping. You should do this in FIFO order. So always replace the oldest entry first.

**This is all you need for Part 1.**

# 2 Part 2 - slightly more complicated

Now,

- `PHYSICAL_MEMORY[128][256]` can only hold 128 pages.
- The page table should only hold valid pages, so it can now only hold 128 entries as well.
- The TLB remains the same.
- Suppose we have already filled all 128 frame slots in memory.

Let's walk through this again under these conditions:

1. Find the page number and offset like before.
2. Let's try to find this in the TLB.
   a. It's not there...
3. Okay, so let's look in the page table.
   a. It's not there either...
4. So the logical address we want is **NOT** in memory. We must, therefore, load it from the backing store and put it into memory.
   a. We need to find a *victim* frame. That is, we need to kick a frame out of `PHYSICAL_MEMORY` so we can put the new frame in there.
   b. To do this, you'll implement a FIFO version of finding a victim frame, and a LRU (least recently used) version of finding the victim frame. Details of implementing this can be found in the Ch9 handout in the zip file. I used the "counter" method on page 416 of that handout.
   c. Now find the page in the backing store like before.
   d. Put the new page into the victim frame's slot in `PHYSICAL_MEMORY`.
5. Now we need to update the page table and TLB.
   a. We already found the victim frame, so locate that page/frame in the page table, and update the page number (obviously the frame number remains the same since we just replaced it with a new page).
   b. Update the TLB like before (in FIFO order). Remember here, the *frame* **could** be in the TLB but is associated with the old page, so you'd want to update the page.
6. Like before, now I know which frame the page is in. So I look for the page number in the TLB. I can find it since I just put it in there or updated it. I concatenate the frame and offset to get a physical address.
7. Now with a physical address, I can go to my `PHYSICAL_MEMORY` array and find the value just like before.

**I hope that helps!!**