**Virtual Memory Manager**

**-Explanation of Problem(s):**

We have many logical addresses generated by CPU, we need to translate them

to their correct physical addresses in the memory,

we can use the TLB to access recently used logical addresses to translate them faster.

If we couldn't find a logical address in the TLB then we can search for it in the page table,

If it wasn't found then that logical address can't be translated now and we search for it in the hard drive 'BACKING\_STORE'.

We need the right most 16 bits from a 32-bit logical address( 8 bit page number,8 bit offset)

#more info to be used:

-2^8 entries in the page table

-Page size of 2^8 bytes

-16 entries in the TLB

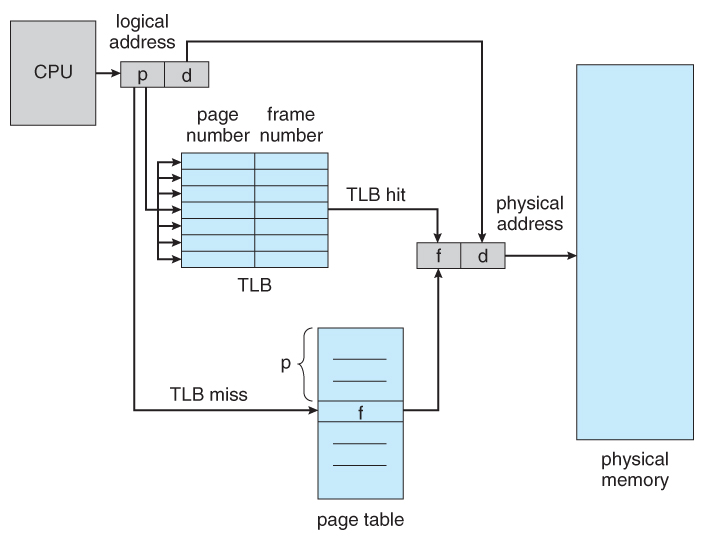
-Frame size of 2^8bytes

-256 frames

-Physical memory of 65,536 bytes (256 frames × 256-byte frame size)

**-Mapping code to problem and showing how project will run:**

First, we have a physical memory which contains all frames, TLB table for faster translation ,page table for loaded frames,

Then the page fault counter& tlb hit counter& address read counter to calculated tlb hit and page table fault rates later. 

**#in main:**

Reading each logical address from addresses file and get its page number, offset

The first thing we do is checking “checkTLB(…)” the tlb. If that page number can be mapped to its frame number in a fast way.

If we found it, we get its frame number and the offset to get its translated physical address(frame number + offset) and its value from the physical memory “readPhysicalMemory(…)” in ‘output file’ .

And we update the tlb to give recently accessed page number a higher priority to reside in the tlb “updateTLBCounter(…)".

@tlbcounter++ which means that the frame number was found in the tlb table

If we couldn't find it then we check the page table “checkPageTable” for that page number to get its physical address (frame number + offset)

And its value from the physical memory “readPhysicalMemory(…)” in 'output file' then we update this page number a higher priority to reside in the page table “updatepageTableCounter(…)” .

If we couldn't get it from the TLB or the Page Table then it's a page fault case ,and we go to check in the hard drive “pageFaultHandler(…)” to get it and this will be slower than page table&TLB.

“pageFaultHandler(…)” opens ‘BACKING\_STORE’ and updates the physical memory with the existed frame number which expresses that page number then it updates the tlb “updateTLB(…)” and the page table “updatePageTable(…)”

Using 'LRU' algorithm.

@pageFaultCounter++ which intends to tell that the address is stored on the hard disk.

Then we get tlb hit rate and page fault rate using their counters.