* Virtual memory
* The virtual address translation process
* Memory Management Unit (MMU)
* Page Tables
* Translation Lookaside Buffer (TLB)

Read this in the wikibook before starting the lab.

* [Introduction to Virtual Memory](https://github.com/angrave/SystemProgramming/wiki/Virtual-Memory%2C-Part-1%3A-Introduction-to-Virtual-Memory)

This lab simulates the behavior of virtual memory for multiple processes. Before getting started, we highly recommend that you have read the wikibook and answer the [Google Form](http://goo.gl/forms/4Us4vBPJnj). Next, please read through the provided code as many files have comments on usage that you will find useful. There are a few different files in this lab that each represent a different component important to virtual memory. Reading through these files can start to help one understand the roles of the different hardware and software involved with virtual memory.

You will only have to write one function in mmu.c, but it requires a good understand of the wikibook and decent knowledge of the rest of the provided code.

Good luck!

MMU.C

This is the function you are to complete in mmu.c:

void \*MMU\_get\_physical\_address(MMU \*mmu, void \*virtual\_address, size\_t pid)

This should convert a virtual address to a physical address from a three level page table.

mmu here is a pointer to the MMU struct defined in mmu.h, virtual\_address is the virtual address you are transforming into a physical address, and pid is the ID of this process.

From the wikibook: “For most memory accesses of most programs, there is a significant chance that the TLB has cached the results… the TLB [may] not have the result cached and the MMU must use the much slower page table to determine the physical frame.”

Thus for any virtual address, you should check whether the result has been already cached in the TLB (see tlb.h). Otherwise you must search the page tables.

Virtual addresses use 64 bits with only 51 bits used for the actual virtual address as you can see from the diagram below. MISC is not counted as part of the virtual address here. In this simulation, the virtual address is broken up as follows for a three level page table.

|----------|---------|---------|---------|-----------|

MISC (13) VPN1 (12) VPN2 (12) VPN3 (12) Offset (15)

Using PageTable\_get\_entry() (see pagetable.h) will allow you to traverse through the page table using the VPNx addresses. Consider what should be done if the page table or frame we are trying to access has not yet been created.

Be sure to read all of the functions within mmu.h, paying close attention to the comments to see the circumstances in which these functions should be called.

NOTE: You must use the PageTable\_get\_entry and PageTable\_set\_entry functions rather than directly accessing the PageTable struct. The same goes for the MMU with modifying the num\_tlb\_misses andnum\_page\_faults. These should be done using the provided functions.