Homework #4

Due Tuesday by 11:59pm **Points** 30

Homework #4 - 30 pts

Submit your homework to TEACH by Tuesday, Nov. 13th, at 11:59pm

Your submission should be composed of two items: a .pdf file containing answers to the questions, and a .c or .cpp file containing your source code for this homework assignment. Legible, handwritten solutions are acceptable for undergraduate students (472).
The handwritten solutions must be electronically scanned and submitted as a PDF file. Graduate students (572) must compose their solutions in MS Word, LaTeX, or some other word processor and submit the results as a PDF file.

Note: As with all programming work in this course, your code must compile on Flip in order to receive credit!

In homework 3 we used a very simple approximation to calculate the size of a page table. This homework tries to derive the size using the actual implementation details.

1. (4 pts) Consider the Intrinsity FastMATH TLB and cache implementation shown in Figure 5.30 (in the textbook). The original design uses a 32 bit virtual memory address and a 32 bit physical memory address. Using a 4096 bytes (4 Kib) page size, each physical page number consists of 20 bits (review the diagram to understand why this is the case). Now assume that physical disk addresses are small enough to fit in the same number of bits as a physical memory address (e.g. for this problem it's 32 bits). **How many bits would it take to implement a full page table for this virtual memory scheme?** Include valid bits, dirty bits, reference bits, and the space for physical/disk addresses in your calculation (as illustrated in Figure 5.29). For simplicity, assume that every virtual page number needs a corresponding page table entry (PTE).
2. (4 pts) Now suppose the Intrinsity Fast Math processor is redesigned using an 8 KiB page size. Assume that physical disk addresses are small enough to fit in the same number of bits as a physical memory address. Assuming each virtual page number needs a corresponding PTE, **how many bits would it take to implement a full page table for this virtual memory scheme?**
3. (22 pts) Using C or C++, implement a program that will automatically perform the calculations which you had to manually compute in order to solve questions 1 and 2. You may still assume that disk addresses require the same number of bits as physical memory addresses. Your program will print the number of bits that would be required to implement the full page table.
   - You must accept several arguments on the command line (in this exact order): <width of virtual addresses (in bits)> <width of physical addresses (in bits)> <size of memory page (in bytes)>

To solve the first problem on this homework assignment you would run the following command (where X is the answer):

```
size_calculator 32 32 4096
Full page table requires X bits.
```

To solve the second problem:

```
size_calculator 32 32 8192
Full page table requires X bits.
```

Another example is as follows.
Here we have a 35 bit virtual address, a 32 bit physical address, and an 8 KiB page size.

```
size_calculator 35 32 8192
Full page table requires 146800640 bits.
```