CS/ECE 752 ADVANCED COMPUTER ARCHITECTURE I
HOMEWORK # 4

(Due by 11:59 AM on Monday, March 24, via upload of PDF to Canvas)

Contact Haocheng Xiao (hxiao55@wisc.edu) for questions

1.(10 points, 3+3+4)

Consider a virtual memory system with the following properties:
•40 bit virtual address (byte addressable)
•4 kB pages
•38 bit physical addresses (byte addressable)
•256 KB cache that is 4-way set-associative and has a line size of 64 bytes, and is accessed with physical addresses
•a 64 entry fully-associative TLB

(a) What is the total size of the page table for each process on this machine, assuming that the valid, protection, dirty, and use bits take a total of 4 bits, and that all of the virtual pages are in use? (Assume that disk addresses are not stored in the page table).

(b) Why might it be infeasible to represent a page table as in (a)? Hierarchical page tables have multiple levels of page tables (segregated by groups of bits in the virtual address). Do hierarchical page tables resolve the issue?

(c) Draw a diagram of the hardware in the memory system including the cache and TLB. Make sure you show how different fields of the address (i.e. which bits) are used to access the cache and TLB.

2. (10 points, 4+4+2)

In a virtually indexed, physically tagged cache, the cache set to search is selected using only bits of the virtual address, so virtual-to-physical address translation can proceed in parallel with reading tags for comparison. In the simplest design, the associativity of the cache is large enough so that the cache index and offset bits together fit entirely into the page offset bits. However, page size remains relatively fixed with architectures while cache size grows with semiconductor technology so this may require a high associativity.

(a) If a processor has 8KB pages and a 64KB level 1 cache, what is the minimum associativity required to use the simple virtually-indexed, physically tagged optimization?

(b) Suppose a cache simply forms a longer index using a few of the least significant bits from the virtual page number. Describe a page table and access pattern where this cache will return incorrect data.

(c) Does the MIPS R10000 have problems with synonyms or homonyms? If so, how does it deal with them? [Refer to the Yeager Micro '96 paper on MIPS R10K that you had reviewed while answering the question]

3. (10 points)

Consider the following piece of code that transposes an integer matrix a and stores the result as the integer matrix b.

```
for (int i = 0; i < 128; i++) {
        for (int j = 0; j < 128; j++) {
                b[j][i] = a[i][j]
        }
}
```

Assume that this is executed on a system with the following cache:

•fully associative
•size 64kB
•no pre-fetching
•block size of 64 bytes
•write allocate and write back
•LRU replacement policy

Assume that the size of an integer is 4 bytes.

(a) Determine the number of conflict, compulsory, and capacity misses while executing this piece of code. Do this separately for both read and write misses.

A "simple pre-fetcher" is defined as follows: If cache misses, fetch the missed block and the next one block. If cache hits, no memory fetch.

(b) Determine the number of conflict, compulsory, and capacity misses if a simple pre-fetch unit is added to the system. Do this separately for both read and write misses.