CS/ECE 752 Spring 2024: HOMEWORK # 4
(Due by **11:00 AM** on **Monday, April 1**, via upload of PDF to Canvas)

Contact Rajesh Srivatsav (rsuresh6@wisc.edu)

1. Problem 1 (15 points, 3+3+6+3)

Consider a virtual memory system with the following properties:
•50 bit virtual address (byte addressable)
•4 KB pages
•40 bit physical addresses (byte addressable)
•64GB physical memory (byte addressable)
•64 KB L1 cache that is 4-way set-associative, has a line size of 64 bytes, and is accessed with virtual addresses
•2 MB L2 cache that is 8-way set-associative, has a line size of 64 bytes, and is accessed with physical addresses
•a 64 entry, 8-way associative data 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 L1 and L2 caches and data TLB. Make sure you show how different fields of the address (i.e. which bits) are used to access the caches and the data TLB.

(d) Explain how a memory access proceeds through the memory system for each of the following scenarios: cache hit, cache miss, TLB hit, and TLB miss.

2. Problem 2 (6 points)

Caches can be either virtually or physically indexed (I), and either virtually or physically tagged (T). That is, there are four possible combinations: VI-VT, VI-PT, PI-VT, and PI-PT. Briefly explain the advantages and disadvantages of each of these.

3. Problem 3 (5 points)

Explain in short, the comparative advantages/dis-advantages of FB-DIMM memory technology over DDR2/DDR3 technology.

4. Problem 4 (14 points, 4+4+3+3)

Consider the following piece of code that transposes an integer matrix A and stores the result as the integer matrix A.   The matrices are stored in column major order, i.e., consecutive elements of a column are in consecutive memory addresses.

```
for (int i = 0; i < 64; i++) {
        for (int j = 0; j < 64; j++) {
                B[j][i] = A[i][j]
        }
}
```

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

•fully associative
•size 8KB
•no pre-fetching
•block size of 64 bytes
•write allocate and write back
•LRU replacement policy

Assume that the size of an integer is 8 bytes.

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

(b) Now consider a simple one-block-lookahead prefetcher which operates as follows: On a cache miss, the prefetches the next block, in addition to fetching the block which missed; on a cache hit the prefetcher does nothing. Determine the number of misses if this simple prefetcher is added to the caching operation. Do this separately for both read and write misses.

(c) A *stride prefetche*r fetches a block that is a stride (S) number of blocks away on a cache miss; on a cache hit the prefetcher does nothing. Would a stride prefetcher be effective here? If so, what stride (S) would the prefetcher use?

(d) What overall prefetching strategy would you recommend in order to minimize the total number of read and write misses?