

1. State whether the following assertions are true or false. Justify your answer with reasons.
  - (a) (2 points) A direct-mapped cache does not need to store tags with each cache line, since every memory block is mapped to exactly one cache line

(b) (2 points) A hardware prefetcher can possibly cause the miss-rate to increase.

(c) (2 points) It is possible for multiple virtual addresses to map to the same physical address.

(d) (2 points) It is possible for the same virtual address to map to multiple physical addresses.

2. Consider an L1 cache which is 2-way set associative, has a capacity of 16 KB, and a block size of 64 bytes. Assume that both virtual addresses and physical addresses have 32 bits. Also, assume that the system has a page size of 4 KB.
- (a) (2 points) Compute the size of the index and the size of the tag in bits.

(b) (4 points) Illustrate, with examples, the problems that can arise if the above L1 cache were to be implemented with a VI-PT (virtually indexed physically tagged) organization?

(c) (4 points) Describe two ways to fix the above problems?

3. (10 points) Consider a uniform memory access (UMA) shared memory multiprocessor with one level of writeback/write-allocate private cache that is connected to memory via a bus (no shared cache). Assume that all of the processors are kept coherent using a bus-based MSI snooping protocol. Further, assume the following.

- Each of the processors are running at 100 million instructions per second (MIPS) on some workload with the following mix: 50% ALU, 20% loads, 10% stores, and 20% branches.
- Instruction cache miss rate is 0%.
- 5% of the loads miss in the local cache.  $\rightarrow$  *Read miss rate*
- 94% of the stores hit in the local cache in M state, 2% of the stores find the block in the local cache in S state, and 4% of the stores find the block in I state.
- Each bus transaction requires 32 bytes.
- Whenever a new cache block is brought into the cache, an old cache block has to be evicted. Assume that every such eviction requires a separate bus transaction. *write back*

If the machine supports a 250 MB/s bus, how many processors can it accommodate before the bus becomes saturated on this workload?

4. For the following questions, consider a shared memory quadcore processor with private L1 caches, a shared L2 cache, and shared memory. Assume that the cache line size is 32 bytes and size of each integer variable is 64 bits. Also, assume that the caches are kept coherent using a directory-based cache coherence protocol with the directory integrated with the shared L2.

Let us assume we are running a program whose objective is to compute the number of prime numbers in the first N natural numbers, where N is the input to the program. Let us assume that the program is parallelised such that each thread is responsible for finding primes in its share of the workload. For the purpose of this question, we are only interested in ensuring that the count of the number of primes is maintained (you may ignore all other issues pertaining to how the problem is parallelised – e.g. load balancing).

The method that is used to maintain the count is as follows. An array *count*[4] is allocated in shared memory; assume that the array is laid out contiguously in memory. Each processor maintains and increments individual counts using only loads and stores (and without using read-modify-write instructions), and later the individual counts are summed up by one of the processors (not shown).

P0	P1	P2	P3
--	--	--	--
...	...	...	...
...	...	...	...
// if prime	// if prime	// in prime	// if prime
count[1]++; →	count[2]++; ↯	count[3]++; ↯	count[4]++;
...	...	...	...
...	...	...	...

- (a) (6 points) Is the method proposed above correct? That is, does it compute the correct value of count? If it is correct (incorrect), please explain why it is correct (incorrect) by describing how the proposed method interacts with the coherence protocol.

(b) (4 points) Explain, in detail, how you can further optimize the method so that it computes the count more efficiently without compromising on correctness? Your explanation should again describe how your optimized method interacts with the coherence protocol.

5. (a) (4 points) Consider a single-core out-of-order multiple-issue superscalar processor. Does that single-core processor enforce sequential consistency (for single-threaded programs running on the single-core processor)? Justify your answer with reasoning.

(b) (4 points) Consider a uniform memory access multi-core processor with no caches. Is the processor guaranteed to enforce sequential consistency? Justify your answer with reasoning.

- 
- (c) (4 points) Consider two multicore processors: multicore A supporting sequential consistency whereas B supports Total-Store-Order. Would you say that A is more programmable than B? Justify your answer with reasons?