Name: Student ID:

1. Which of the following statements about memory consistency in shared memory multiprocessors is **TRUE?**
	1. Memory consistency defines how loads and stores are ordered relative to loads and stores from the same thread.
	2. Memory consistency defines how loads from one thread are ordered relative to stores from another thread.
	3. Memory consistency defines how loads and stores from one thread are ordered relative to loads and stores from another thread.
	4. Memory consistency ensures that loads and stores from different threads to a shared variable in memory are properly synchronized.
	5. None of the above.
2. Which of the following statements about memory consistency is **FALSE?**

We need to know about memory consistency in order to:

* 1. Write correct shared memory parallel programs.
	2. Write correct MPI programs.
	3. Design distributed shared memory multiprocessors hardware.
	4. Understand performance of OpenMP parallel programs.
	5. Design various cache optimizations correctly.
1. Which of the following hardware mechanisms is not needed to implement processor consistency model?
	1. A mechanism to order stores to memory.
	2. A mechanism to prevent overlapping memory writes.
	3. A mechanism to prevent late cache invalidates.
	4. A mechanism to prevent non-blocking memory reads.
	5. A load buffer to snoop stores of other threads.
2. Which of the following statements about synchronization with barriers is **FALSE?**
	1. The barrier is a synchronization mechanism in shared memory multiprocessors.
	2. Each thread temporarily stalls when it reaches the barrier.
	3. When all threads reach the barrier, each thread has to acquire a lock before proceeding with executing the code after the barrier.
	4. When the last task reaches the barrier, all tasks are synchronized.
	5. Locks are used with critical sections while barriers are not.
3. Which of the following statements about synchronization with locks is **FALSE?**
	1. In lock-based programs, shared data in critical sections is protected by locks.
	2. Lock contention occurs when a thread tries to acquire a lock and the lock is held by another thread.
	3. Data conflict exists in a critical section when a thread modifies data that is read or written by another thread.
	4. High probability of lock contention does not necessarily result in high probability of data conflict.
	5. Hardware bounds data to the lock that protects this data by tracking blocks accessed in the critical section.
4. Which of the following statements about synchronization with locks is **FALSE?**
	1. The memory read operation in the test and set instruction that acquires the lock is treated as a memory write by the cache coherence hardware.
	2. Lock variables exhibit temporal silence.
	3. After the critical section, the lock is released by restoring its value to zero.
	4. The lock release instruction does not require a special store instruction or hardware support.
	5. Between the time that the test and set load executes until the lock is released, the cache hardware does not respond to cache miss requests from other threads that hit the lock block in the cache.
5. Which is the best way of writing Spin Lock code?
	1. While (test-and-set(lock) == 1);
	2. While ((lock == 1) || test-and-set(lock) == 1) ;
	3. While ((lock == 1) && test-and-set(lock) == 1) ;
	4. While (test-and-set(lock) == 1) || (lock == 1));
	5. While (test-and-set(lock) == 1) && (lock == 1));
6. Which of the following statements about Transactional Memory is **FALSE**?
	1. Transactional Memory requires hardware to buffer write data.
	2. Transactional Memory requires hardware to track memory accesses.
	3. Transactional Memory requires hardware to detect data conflicts.
	4. Transactional Memory requires hardware to undo memory and registers updates.
	5. Transactional Memory requires hardware to remove the lock.
7. Which of the following statements about Transactional Memory is **TRUE**?
	1. Transactional Memory is best for replacing critical sections with low lock contention, low data conflicts.
	2. Transactional Memory is best for replacing critical sections with high lock contention, low data conflicts.
	3. Transactional Memory is best for replacing critical sections with low lock contention, high data conflicts.
	4. Transactional Memory is best for replacing critical sections with high lock contention, high data conflicts.
	5. None of the above.
8. Which of the following statements about Speculative Lock Elision is **FALSE**?
	1. Speculative lock elision hardware acquires the lock without writing the lock externally.
	2. Speculative lock elision hardware executes the critical section speculatively.
	3. Speculative lock elision hardware buffers critical section write data and tracks memory accesses.
	4. Speculative lock elision ignores cache miss requests from other threads to the lock block in the cache.
	5. Speculative lock elision resolves data conflicts by one thread aborting and restarting the critical section.
9. Which of the following statements about weak memory ordering is **FALSE**?
	1. All memory operations from the CPU must complete before a fence instruction later in the program executes.
	2. The memory operations from the CPU cannot issue to memory until an earlier fence in the program executes.
	3. Stores from the same CPU must be issued in program order.
	4. Fence instructions from the same CPU cannot be reordered.
	5. Loads can be issued in any order.
10. Which of the following statements about Transactional Memory is **TRUE**?
	1. Internal transactional load that hits a block in exclusive state results in the block writeback to DRAM.
	2. Internal transactional store that hits a block in exclusive state results in the block writeback to DRAM.
	3. Internal transactional load that hits a transactional block in exclusive state causes the block writeback to DRAM.
	4. Internal transactional store that hits a transactional block in exclusive state causes the block writeback to DRAM.
	5. None of the above.
11. Which of the following statements about locks and critical sections is **TRUE**?
12. Critical sections with high lock contention have high data conflicts.
13. Software that uses test-and-set to acquire a lock executes the test-and-set instruction atomically.
14. Software that uses test-and-set to acquire the lock also uses test-and-set instruction to release the lock.
15. Hardware executes the test-and-set atomically.
16. Hardware executes the test-and-set and the critical section atomically.
17. Which of the following statements about speculative lock elision is **FALSE**?
18. Speculative lock elision hardware executes test-and-set without invalidating other caches.
19. Speculative lock elision hardware does not set the lock variable in the cache.
20. Speculative lock elision hardware executes a critical section atomically.
21. Speculative lock elision hardware provides code backward compatibility.
22. Speculative lock elision does not benefit critical sections with high data conflicts.
23. Which of the following statements is **FALSE**?
24. Virtual memory allows large applications that do not fit in DRAM to execute with good performance.
25. Virtual memory makes DRAM effectively a cache for the very slow hard disk.
26. Virtual memory is implemented completely in software.
27. Virtual memory allows multiple running applications to share the DRAM.
28. Which of the following statements is **FALSE**?
29. Access bit is set for a page in DRAM when the page is read or written.
30. Access bits are periodically cleared by the operating system.
31. Access bit indicates if a page has been accessed since it has been loaded from the disk.
32. Access bits are stored with the physical address in the page tables.
33. Access bits are used in page replacement algorithm.
34. A processor uses the fact that the page offset bits are the same in both virtual and physical addresses to reduce the latency it takes to perform address translation and cache lookup. The processor uses the page offset bits and starts the set access to read data and tags in the same cycle that the TLB performs the virtual to physical translation. Assuming a page size of 4K bytes, cache block size of 32 bytes and 4-way set associative cache. What is the maximum cache size that this processor can implement?
35. 4K bytes.
36. 8K bytes.
37. 16K bytes.
38. 32Kbytes.
39. Any size that is a power of 2 is possible.