Chapter IV, section 4.6:

1) Give an example of a conditional instruction (showing the sequence it replaces in assembly). Explain how conditional instructions can improve performance on standard pipelined machines. On superscaler machines. Ch. 4, part VII, pages 1-3.

2)How can these instructions be problematic with respect to exceptions. Ch. 4, part VII, page 4.

3) Briefly discuss the four main limitations of conditional instructions. Ch. 4, part VII, pages 5-6.

4) Preventing machine state changes and exceptions from occurring for conditional instructions is relatively simple. However, when larger blocks of code are speculated on, dealing with these is more difficult. Describe one compiler-directed solution that uses register copies and ignores exceptions. Ch. 4, part VII, pages 8-9.

5) Describe how register copies and poison bits can be used to solve this problem. What complication does this scheme introduce? Ch. 4, part VII, page 10.

6)Describe a hardware solution called boosting that provides renaming and buffering. Ch. 4, part VII, pages 11-12.

7) Describe the conceptual change to Tomasolo's algorithm required to support speculative execution and dynamic scheduling. How is a reorder buffer used to permit the proper handling of exceptions? Ch. 4, part VII, pages 13-14.

8)Describe the operation of the Issue, Write result and Commit phases of Tomasolo's algorithm with respect to the reorder buffer. Ch. 4, part VII, pages 15-16.

9)Briefly discuss the two advantages of this scheme over dynamic scheduling alone. Does this scheme provide precise exception handling? Ch. 4, part VII, pages 17-18.

Chapter V

1)What are four questions one can ask about any two levels of a memory hierarchy? Ch. 5, part I, page 2.

2) Discuss three aspects of the following memory performance equation. Ch. 5, part I, page 3.

$$\text{Memory stall cycles} = \text{IC} \times \text{Mem refs per instruction} \times \text{Miss rate} \times \text{Miss penalty}$$

3) Describe write-through and writeback cache write policies. Discuss their performance advantages/disadvantages. Describe write allocate and write around write miss policies. How are these matched with cache write policies and why? Ch. 5, part I, pages 9-10.

4) Discuss how write buffers and write merging can improve performance in caches. Ch. 5, part I, page 11.

5) Write the expression for CPU time which takes into account memory stall cycles due to cache misses. Assume a unified cache. Ch. 5, part II, page 2.

6) Define the three C's (compulsory, capacity and conflict) misses. Ch. 5, part II, page 7.

7) One approach to reducing cache miss rate is to increase cache block size. Explain how doing so might affect the three C's and miss penalty. Ch. 5, part II, page 8.

8) Which of the three C's does increasing cache associativity reduce? What counteracts the benefits of increasing associativity in the limit? Ch. 5, part II, page 10.

9) Explain the operation of a victim cache. Which of the three C's is it designed to reduce? Ch. 5, part II, pages 11-12.

10) How is a pseudo-associative cache different from a victim cache? Which of the three C's is it designed to reduce? How are both of these techniques different from a standard set-associative cache? Ch. 5, part II, pages 12-13.

11) Define hardware prefetching. Which of the three C's is it designed to reduce? What assumption about memory bandwidth must hold in order for prefetching to work? How can prefetching hurt performance? Ch. 5, part II, page 14.

12) Why is compiler-controlled prefetch more "expensive" than hardware prefetch? What is the difference between faulting and non-faulting prefetching instructions? Why do you think compiler-controlled prefetch can outperform hardware prefetch? Ch. 5, part III, pages 1-2.

13) In general, how can compiler optimizations reduce miss rates? How can "merging array" help reduce miss rates? Explain loop interchange, loop fusion and blocking optimizations with respect to spatial and temporal locality. Ch. 5, part III, pages 2-7.

14) With respect to reducing cache miss penalty, what are the problems associated with giving read misses priority over write backs. Discuss a couple solutions. Ch. 5, part III, page 8.

15) Describe the advantages of dividing the cache blocks into subblocks with respect to miss rate and penalty. Ch. 5, part III, page 9.

16) Discuss the difference between early restart and critical word first. How do these techniques reduce miss penalty? Ch. 5, part III, pages 10-11.

17) How do non-blocking caches reduce cache miss penalty? Ch. 5, part III, pages 11-12.

18) What component of memory access time is a two-level cache system designed to reduce? Ch. 5, part III, page 12.

19) How is the performance of a two-level cache computed? Distinguish between global and local miss rate. Why is the L2 local miss rate higher than the global miss rate? Ch. 5, part III, pages 13-14.

20) What size, associativity and write policy would you expect to work best for a first level cache (in a two-level scheme) and why? What size (w.r.t. level one), associativity and write policy would you expect to work best for the second level cache and why? Ch. 5, part III, pages 14-17 and part IV, page 1.

21) What benefit is there to maintaining a multi-level inclusion policy between the L1 and L2 caches? Ch. 5, part III, page 16.

22) Why are direct mapped caches attractive for reducing hit time. Ch. 5, part IV, page 2.

23) Why are virtual caches more attractive than physical caches (on the outside)? How does multitasking introduce difficulties in virtual caches? Propose a solution. What advanced operating system feature makes aliasing possible? What problem does this cause in caches? Ch. 5, part IV, pages 3-4.

24) Discuss the advantages and disadvantages of using the page offset to index the cache as a solution for aliasing problems. Ch. 5, part IV, page 5.

25) Describe pipelined writes as a mechanism to reduce hit time. Ch. 5, part IV, page 6.

26) Discuss the advantages and disadvantages of increasing the width of main memory. How does an interleaved memory address the disadvantages of widening memory? Explain the advantage of an interleaved memory (over a wider memory) when sequential accesses are generated to different memory banks. Ch. 5, part IV, pages 11-15.

27) Discuss a method that reduces memory bank conflicts. Ch. 5, part IV, pages 17-18.

28) Why do virtual memory systems almost always use fully associative placement policy? Ch. 5, part V, page 3.

29) Why do virtual memory systems almost always use a writeback write policy? Ch. 5, part V, page 5.

30) Virtual memory requires two memory accesses per CPU access. What structure is used to reduce this to 1? Why is a hardware replacement mechanism (versus a software based approach) an important implementation detail in systems with physical caches? Ch. 5, part V, pages 6-7.

31) Why are large page sizes attractive in virtual memory systems? What is the drawback? Ch. 5, part V, page 8.

32) How does I/O complicate the use of cache? Discuss methods of dealing with this problem. Ch. 5, part V, pages 12-13.

Chapter VI

1) Explain the two major components of disk latency. What other components must be considered when calculating performance, e.g. the amount of time required to transfer a single sector? Ch. 6, part I, pages 6-9.

2) Explain what is meant by the Access Time Gap. Why has it been so difficult to fill this gap with a new technology? Ch. 6, part I, pages 11-12.

3) Discuss the advantages and disadvantage(s) of a bus. What physical characteristics distinguish CPU-memory buses from I/O buses? Ch. 6, part II, page 7.

4) Discuss three bus options that clearly trade-off performance and cost. Explain why it only makes sense to implement split transactions on a bus that has multiple bus masters. How are latency and bandwidth affected in such a system? Ch. 6, part II, pages 9-11.

5) Identify the characteristics of a set of components that are best connected by a synchronous bus. What advantages do asynchronous buses have over synchronous buses? Ch. 6, part II, pages 12-13.

6) Give an intuitive explanation of Little's Law. Define and explain average residual service time. Give an expression that computes the amount of time a request waits in a queue using the simplifying assumptions of the queueing theory model. Ch. 6, part III, pages 4-5.

7) Distinguish between reliability and availability. Ch. 6, part IV, page 1.

8) How do disk arrays and striping improve bandwidth? Explain how reliability affects availability in disk arrays? How does RAID fix this problem? Ch. 6, part IV, pages 1-4.

9) Explain the performance of reads and writes in a RAID 1 with respect to a RAID 0 system. What is the problem with RAID 1. Ch. 6, part IV, pages 5-6.

10) Explain the idea behind RAID 3. What is the overhead for parity? What is its major drawback? Ch. 6, part IV, page 7.

11) What is the difference between RAID 3 and RAID 4? How does RAID 5 address the problem with RAID 4? Explain how a RAID 5 system optimizes disk writes? How many failures can RAID 6 survive? Ch. 6, part IV, pages 8-11.

12) How do you go about designing an I/O system? Ch. 6, part V, pages 1-3.

13) Explain why disk seek time is not linear? For short seek times, does a linear model underestimate or overestimate seek time? Ch. 6, part V, pages 11-12.

Chapter VIII

1)Why are parallel machines becoming more attractive as a successor to the uniprocessor? Ch. 8, part I, page 2.

2) Distinguish between SIMD and MIMD. Why has MIMD emerged as the architecture of choice in recent years? Ch. 8, part I, pages 3-4.

3) Distinguish between centralized shared-memory architectures and distributed memory architectures with respect to number of processors. Ch. 8, part I, pages 5-7.

4) Explain the difference between a distributed shared-memory architecture and multicomputer. How is communication performed on each of these architectures? Ch. 8, part I, page 8.

5) Explain the advantages of a shared memory architecture. Of message-passing architectures. Ch. 8, part I, page 13.

6) What are the two major challenges of parallel processing? What are the approaches to solving these problems? Ch. 8, part I, page 15.

7) What are the advantages of replicating data across caches? What problem does it introduce? Ch. 8, part II, page 1.

8) Give three criteria for a coherent memory system. Define what is meant by memory consistency. Ch. 8, part II, pages 3-5.

9) Define the two classes of protocols for cache coherence. Ch. 8, part II, page 6.

10) Explain the difference between the bus snooping protocols write-invalidate and write-broadcast with respect to writes. Explain the performance differences that make write-invalidate more popular. Ch. 8, part II, pages 7-9.

11) Explain how to implement write-invalidate protocol in a bus-based system. Ch. 8, part II, page 10.

12) Why are write-back caches more popular in multiprocessor environments? How do writes complicate matters for the implementation of a write-invalidate protocol? Propose a solution. Ch. 8, part II, pages 10-11.

13) Discuss two methods to reduce the interference of the bus logic with the CPU access to cache. Ch. 8, part II, pages 12-13.

14) Explain the states and transitions of the finite state machine used to implement the protocol for a centralized shared-memory architecture. Ch. 8, part II, pages 14-16.

15) Identify two simplifications that we made to the protocol discussed in class. Ch. 8, part II, page 17.

16) Explain three problems with software maintained coherence in distributed shared-memory architectures. Ch. 8, part III, pages 1-2.

17) Why doesn't snooping work in distributed shared-memory architectures? Ch. 8, part III, page 2.

18) What types of information is kept in the directory in a directory-based coherence protocol? How is the directory information distributed? Ch. 8, part III, pages 3-4.

19) What are the primitives that must be handled and what are the complications to their implementation? Ch. 8, part III, page 5.

20) Explain the purpose of the seven message types used in the implementation discussed in class. Ch. 8, part III, pages 6-7.

21) Explain the states and transitions of the finite state machine used to implement a directory-based protocol for a distributed shared-memory architecture. Ch. 8, part II, pages 8-12.