ANSWER ALL 6 QUESTIONS

1. Floating-Point Unit Design

An important component in a floating-point unit is a leading-zeros detector (also called a one's position detector). The purpose of this piece of logic is to determine the bit position of the most significant non-zero bit in the mantissa (so that the mantissa can be shifted appropriately to get a normalized mantissa). For example, for an 8-bit mantissa of `01001100`, an 8-bit one's position detector will return a value of `001` (i.e., a 1), and for a mantissa value of `00001110`, it will return a value of `100` (i.e., a 4).

(i) Design a 4-bit one's position detector. It is sufficient to present the Boolean equations for this design.

(ii) Using the 4-bit detector designed above, and hierarchical design, present the design for a 16-bit one's position detector. If you need to design any additional units, show their design (again, Boolean equations will be sufficient).

You will be graded on correctness first, clarity of hierarchical design second, and speed third.

2. Synchronization

Many parallel programs coordinate processing by using locks based on test-and-set() to enforce mutual exclusion on critical sections.

(i) Besides test-and-set(), what other primitives can be used to enforce mutual exclusion? Give at least three hardware-base alternatives.

(ii) How do these primitives differ with regard to: (a) fairness, (b) efficient use of resources, (c) ease of use?

(iii) Are all your primitives equivalent? If so, explain how you know this. If not, is it possible to order the primitives? That is, are some more powerful than others?

3. Memory Hierarchies

Memory systems exploit natural clustering of data accesses by recognizing the concept of spatial and temporal locality. Both cache memory and virtual memory exploit this locality by dealing with blocks of contiguous data, which is likely to include data not yet accessed. There are many similarities in the way that virtual memory and cache memory hierachies work, but one parameter captures a fundamental difference between traditional virtual memory and a traditional cache memory system: the ratio of access times to data in the upper and lower levels of the hierarchy.

(i) Explain how this ratio is different in virtual memory systems and in a single-level cache hierarchy.

(ii) How can this ratio be used to explain other differences between virtual memory and a cache hierarchy, such as hit ratio, page/line size, write policy, etc.

(iii) How would you expect that current microprocessor trends will affect future cache hierarchies? As access times to off-chip memories become slower relative to on-chip accesses, would you expect to see the system treat these accesses more like page faults than cache misses? Explain your answer.

4. Instruction Set Design

In early computers, opcodes often had individual bits, or groups of bits, associated with very specific actions. For example, the first two bits of the IBM System/360 opcode indicate the format of the instruction. In the CDC 6600, if the first three bits of the opcode were not zero, the instruction was guaranteed to increment the PC in the normal way. With the advent of microprogramming, this correlation was lost, with instructions often being grouped in ways completely unrelated to the assignment of the bits in the opcode. This was one of the ways in which RISC machines turned back the clock, reintroducing simpler instructions where individual bits again had specific meaning.

(i) What is a possible explanation that micro-coded implementations moved away from careful instruction encodings?

(ii) How does the choice of instruction encodings interact with modern implementations? That is, does the choice of bit patterns for individual instructions matter for modern designs that make heavy use of branch prediction, register renaming, out-of-order issue, and speculation? Explain your answer.

5. Interconnection Networks

One of the key components of a parallel computer is its interconnection network. Just as with other components, the network of choice in a parallel computer has also changed over the years. In the 1970s, most parallel machines were proposed to be built with indirect networks, such as an Omega network, whereas the wisdom in the 1990's suggests the use of direct networks such as k-ary n-cubes. Moreover, the routing techniques used to route messages have also changed.

(i) Give reasons why network topologies have changed from indirect to direct networks.

(ii) What options exist for routing a message in a direct network such as a k-ary n-cube.

(iii) Which routing techniques are commonly used today, and why?

6. Instruction Supply

Most microprocessors use instruction caches to facilitate the supply of instructions into the processing engine. Microprocessors with simple in-order execution pipelines (e.g., Hennessy and Patterson's DLX pipeline) fetch at most one instruction per clock cycle whereas more recent microprocessors have used additional instruction-level parallelism (ILP) techniques such as multiple instruction issue, out-of-order execution, and speculation. How do these ILP techniques impact the importance and design of instruction caches and other aspects of the instruction fetching/supply mechanisms? Illustrate your answer with specific examples.