SPRING 2004
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH.D. QUALIFYING EXAMINATION

Computer Architecture
Qualifying Examination
Monday, February 2, 2004
3:00 – 7:00 PM
Room 3345 Engineering Hall

GENERAL INSTRUCTIONS:

1. Answer each question in a separate book.
2. Indicate on the cover of *each* book the area of the exam, your code number, and the question answered in that book. On *one* of your books list the numbers of *all* the questions answered. *Do not write your name on any answer book.*
3. Return all answer books in the folder provided. Additional answer books are available if needed.

SPECIFIC INSTRUCTIONS:

Answer all of the following **SIX** questions. The questions are quite specific. If, however, some confusion should arise, be sure to state all your assumptions explicitly.

POLICY ON MISPRINTS AND AMBIGUITIES:

The Exam Committee tries to proofread the exam as carefully as possible. Nevertheless, the exam sometimes contains misprints and ambiguities. If you are convinced a problem has been stated incorrectly, mention this to the proctor. If necessary, the proctor can contact a representative of the area to resolve problems during the *first hour* of the exam. In any case, you should indicate your interpretation of the problem in your written answer. Your interpretation should be such that the problem is non-trivial.

1. Bus Arbitration Specification & Implementation

Consider a bus arbitration circuit that accepts n request signals, R_0 to R_{n-1}, and generates n grant signals, G_0 to G_{n-1}, such that:

- G_i is asserted only if R_i is asserted,
- If any R_i is asserted then exactly one G_i is asserted, and
- No other "nice" properties are required.

You may assume that if input R_i is asserted, it stays asserted until G_i is asserted.

(a) Using only NAND gates, design a fast combinational-logic arbitration circuit for n = 4. Circuits will be graded on speed and clarity (i.e., use hierarchy where appropriate). Assume all NAND gates have at most four inputs and can drive at most four other gates or outputs.

(b) Repeat part (a) for n = 16.

(c) For an arbitration circuit for the bus of a symmetric multiprocessor (SMP), what other properties would you add to the arbiter specification? Why?

2. Cluster Computing

Consider creating a stand-alone parallel computer from a collection of PCs (or workstations) interconnected via a local area network, such as gigabit Ethernet.

(a) What hardware and software factors determine the communication *bandwidth* between processes running on two different PCs?

(b) What hardware and software factors determine the communication *latency* between processes running on two different PCs?

(c) What programming model (or models) do you expect to be most effective on this parallel computer? Justify your answer.

3. Performance Evaluation

Computer architects use a variety of methodologies for performance evaluation. These methodologies include performance measurement techniques as well as benchmarks. Historically most computers have been designed for scientific and desktop applications. Methodologies for performance evaluation in this environment included: (i) measurement techniques such as analytical models, internal and external hardware monitors, and simulation, and (ii) benchmarks such as kernels from scientific applications (e.g., Livermore loops or Linpack) or desktop applications (e.g., SPEC).

What performance methodologies do you expect to be used in the design of future computers? In particular, what methodologies do you expect to see used in computers designed for multimedia applications? What about commercial applications?

4. Instruction Sets

A subject of frequent debate in the mid- to late-1980s was the choice of instruction sets and their impact on performance. Proponents of RISC instruction sets (e.g., MIPS, Alpha) argued that RISC instruction sets were easier to implement and also more amenable to high performance implementations. For example, an in-order pipeline for a RISC instruction set was simpler and required less hardware than a similar pipeline for a CISC instruction set. Proponents of CISC instruction sets (e.g., x86) argued that this did not represent the entire picture.

In the late 1980s, implementations of RISC instruction sets were frequently the highest performance processors. Today, implementations of CISC processors (e.g., x86) are often the highest performance processors. What technical trends and microarchitectural innovations have allowed implementations of CISC instruction sets to bridge or even overcome the performance gap?

5. Write Policies in Chip Multiprocessors

The earliest cache-coherent shared-memory multiprocessors used write-through cache policies. Beginning in the early 1980's, most shared-memory multiprocessors switched to using writeback caching policies. Over the last few years, chip multiprocessors (CMPs) have emerged as a promising way to build multiprocessor systems. Most of these systems use write-through caching policies for their level-1 (L1) caches, but use writeback for their level-2 (L2) caches (i.e., stores update both the L1 and L2 caches, but not main memory).

(a) Discuss the trade-offs between write-through and writeback caching policies for multiprocessors. What factors favor one over the other?

(b) Explain why shared-memory multiprocessors evolved from using write-through to writeback caching policies.

(c) What has changed to make current CMPs use write-through level-1 caches?

(d) Do you expect CMPs to also evolve to use writeback level-1 caches? Why or why not?

6. Memory-Level Parallelism

Historically, processor speeds have improved faster than memory speeds, leading to increasing latencies for memory operations (as measured in processor clock cycles). Computer architects have attacked the problem using a variety of techniques. One important class of techniques strive to achieve memory-level parallelism; that is, to overlap multiple long-latency memory operations.

(a) Non-blocking caches are one such technique. Explain how non-blocking caches work and why they help improve average memory latency and bandwidth.

(b) Some architects believe that instruction-level parallelism subsumes memory-level parallelism. That is, since loads and stores are simply instructions, the same techniques that provide parallel execution of arithmetic instructions (e.g., reorder buffers) suffice to achieve adequate memory level parallelism. Others argue that these techniques are not sufficient and that other architectural or microarchitectural techniques must be used. Discuss the fundamental issues and techniques for achieving adequate memory-level parallelism in future systems.