SPRING 2000
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH.D. QUALIFYING EXAMINATION

Computer Architecture
Qualifying Examination
Tuesday, February 8, 2000
3:00 – 7:00 PM
Room 226 Noland 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. Population Count Implementation

Population count is an operation that appears in some computers to facilitate code breaking. The population count of a value is the number of ones in the value's binary representation. The population count for `01101011`, for example, is 5.

(a) Use gates (e.g., NOT, AND, NAND, XOR, OR, and NOR) to design a combinational implementation of a population count operation on a 16-bit input. You will be graded for correctness first and clarity of hierarchical design second.

(b) What is the number of gate delays through your design?

(c) Discuss (but do not implement in detail) what techniques you might use for implementing a low-latency population count for a wide input value (e.g., 128 bits).

2. Multithreading

Multithreading is a concept that has been around for several decades. For several years, for a variety of reasons (both technical as well as non-technical) it was not considered viable technology for mainstream processor designs. Recently Compaq announced that its next-generation mainstream microprocessor design will use multithreading.

(a) Discuss the pros and cons of different types of multithreading, highlighting the reasons used by architects of mainstream processors to justify their decision not to use multithreading.

(b) Discuss recent trends that have convinced some processor architects to adopt multithreading for mainstream processors.

3. Performance and Simulation

Trace-driven simulation is a well-established technique for studying performance of uniprocessors. Recently a second technique — execution-driven simulation — has become a popular alternative, because it provides certain advantages over trace-driven simulation. An example of execution-driven simulation is the SimpleScalar simulator.

(a) What are the relative advantages of trace-driven and execution-driven simulation of a uniprocessor?

(b) Why is it that execution-driven simulation has only recently become a popular alternative for uniprocessors?

(c) How do the trade-offs between the two methods change when multiprocessor simulation is involved?

4. Shared Memory Implementations

Shared-memory is a programming abstraction that can be implemented in a variety of ways. Most companies have chosen to support this abstraction by building custom hardware that implements a cache coherence protocol (e.g., the Sun UltraEnterprise 10000 and SGI Origin2000). Conversely, many researchers (and some companies) have explored using software-based mechanisms to implement coherence protocols on industry standard hardware (e.g., Rice TreadMarks).

Discuss the trade-offs of software versus hardware implementations of shared memory. How does this affect the choice of coherence protocol? How does it affect other design decisions? How does it affect performance over a wide range of applications?

5. Virtual Address Synonyms

Virtual address synonyms occur when the operating system maps two or more virtual addresses to the same physical address. These virtual addresses may be in the same or different virtual address spaces. Virtual address synonyms can cause problems in some types of cache designs.

(a) Explain what types of caches may have problems and explain the problems that may occur.

(b) Discuss alternative hardware techniques to avoid these problems. Discuss the pros and cons of these alternatives.

(c) Discuss alternative software techniques to avoid these problems. Discuss the pros and cons of these alternatives.

6. System-on-a-Chip Memory Systems

Consider a future chip that includes a single processor surrounded by all of system memory. Due to communication delays (due in part to physically laying out memory), 1% of the memory can be accessed in 1 clock cycle, 9% in 9 cycles, 21% in 10 cycles, 30% in 11 cycles, and 40% in 12 cycles. The 1-cycle memory is intended to be used as cache, but the remaining memory will constitute the main memory of the system, i.e., there is no off-chip main memory.

There are several possibilities regarding the memory systems. (i) You might partition the memory by speed, making visible to the programmer and the operating system the various degrees of latency. (ii) You might simply specify that the main memory speed is variable, specifying the minimum, maximum, and average latency. (iii) You might simplify the control logic by waiting for the worst-case delay, thereby providing the result in constant time.

(a) Describe the relative merits of each of these approaches.

(b) Which method would you recommend for the system being designed?