FALL 2001
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH.D. QUALIFYING EXAMINATION

Computer Architecture
Qualifying Examination
Monday, September 24th 2001
3:00 – 7:00 PM
Room 2317 Engineering

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. Implementing Unary Numbers

Consider the somewhat artificial case of representing non-negative integers with n bits in unary, such that the value i is represented with the i least significant bits set to 1 and the higher bits set to 0. With n=8 and i=5, for example, we get: 00011111. Consider two numbers, A and B, represented with n unary bits.

(a) With n=4, use gates to implement logic that realizes a less-than test ($A<B$).

(b) With n=4, use gates to provide an efficient implementation of addition ($A+B$) where the output is sized to prevent overflow.

(c) *Discuss* how the logic of (a) and (b) changes if each value i is represented in unary with i 1's that are not necessarily in the least significant bits. With n=8 and i=5, for example, we could get: 01101011.

2. Performance Evaluation

Computer architecture is an inherently quantitative field, and computer architects use many techniques to evaluate performance. These include techniques for understanding the behavior of existing machines, as well as techniques to project the performance of future machine designs.

(a) List and describe the common performance evaluation techniques used by computer architects, and discuss their pros and cons.

(b) Do multiprocessors and uniprocessors require different techniques? Discuss your answer and reasoning.

3. Directory Protocol Races

Consider directory protocol options for a CC-NUMA machine with 32 processor-memory nodes. Consider races between coherence requests to the same block *B*. Further assume that block *B*'s home is node 0 and that *B* is in state *I* (invalid) in all nodes not mentioned.

(a) Consider block *B* initially in state *M* (modified exclusive) at node 1. Assume node 2 makes a request to obtain block *B* in state *M*. Further assume that at approximately the same time, node 3 makes a similar request (*i.e.*, to obtain block *B* in state *M*). Discuss potential races and options for maintaining correctness.

(b) Consider block *B* initially in state *M* (modified exclusive) at node 1. Assume node 1 sends a request to the directory to write block *B* back and revert to state *I*. Assume while this request is traveling to the directory, node 2 makes a request to obtain block *B* in state *M*. Discuss potential races and options for maintaining correctness.

4. Vectors

Since the early 1980s, single-chip microprocessors have been reinventing or adapting techniques first explored on main frames and supercomputers. In 1976, Cray Research introduced the Cray-1 computer, an extremely fast computer in its day. Much of its performance resulted from its use of vector registers, and instructions that used those registers. Since the advent of high-performance single-chip processors, no general-purpose computer, including those aimed at numerical applications, has supported Cray-style vectors.

(a) With the large availability of transistors on current and emerging processors, why has no one introduced Cray-style vector registers?

(b) Do you expect to see any form of vector support on future computers? Why or why not?

5. On-chip Cache Memories

Over the past 20 years, microprocessors have evolved from having 256-byte on-chip *instruction* caches, to separate 64-kilobyte on-chip level-1 instruction and data caches, and recently 1.5-megabyte on-chip level-2 caches. Designers have used a variety of techniques beyond increasing cache size to optimize performance, including sub-blocking and associativity.

(a) Discuss the impact of sub-blocking and associativity on the cost and performance of on-chip caches. How have the tradeoffs evolved with the changing number of transistors on a single chip?

(b) Some microprocessor designers have opted to place the level-2 cache tags on-chip while the data resides off-chip. What are the pros and cons of this approach? What are the benefits and issues with using sub-blocking and associativity with such a design?

6. CMP vs. SMT

Chip-based multiprocessors (CMPs) and simultaneous multithreaded (SMT) processors are two ways of having multiple threads running simultaneously on a single chip.

(a) Compare and contrast the key design issues for CMPs and SMTs. What are the pros and cons of each design?

(b) Given the long-term technology trends, will one approach prevail over the other? Will both co-exist? Will both die out?