SPRING 2005
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH.D. QUALIFYING EXAMINATION

Computer Architecture
Qualifying Examination
Monday, January 31, 2005

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. Sorting Network Implementation

Implementations may use gates, flip-flops, etc. Combinational logic may be done with equations. You will be graded by clarity of design first and speed second.

(a) Implement a *combinational circuit for sorting* two integers. Assume that it accepts two three-bit unsigned-integer inputs, I0<2:0> and I1<2:0>, and produces two outputs, J0<2:0> and J1<2:0>, that are a permutation of the inputs, such that J0<2:0> ≤ J1<2:0>.

(b) Implement a combinational circuit for sorting four integers. Assume that it accepts four three-bit unsigned-integer inputs, K0<2:0> to K3<2:0>, and produces four outputs, L0<2:0> to L3<2:0>, that are a permutation of the inputs, such that L0<2:0> ≤ L1<2:0> ≤ L2<2:0> ≤ L3<2:0>.

(c) Discuss implementing a combinational circuit for sorting 16 integers.

2. Vector Architectures

Vector architectures and vector instructions have been around for four decades, yet have not seen widespread adoption in most processing scenarios (e.g., general-purpose or embedded processors). With the recent emphasis on power efficiency, there has been renewed interest in vector architectures and vector instructions.

(a) Why have vector architectures been popular for scientific supercomputers?

(b) Why have vector architectures not been popular for general-purpose architectures?

(c) Argue why vector architectures and vector instructions might be more energy efficient that non-vector (e.g., superscalar) architectures.

3. Power

(a) In addition to cost and performance, architects must now concern themselves with *energy*, *sustained power,* and / or *peak power.* Discuss why power-related issues matter for both of *compute servers (*e.g., in servers farms), and *laptop computers.*

(b) Compare and contrast techniques for reducing power in each of the above environments. Be sure to discuss processor chips and aspects external to processors.

4. Precise State Mechanisms

Techniques to implement *precise interrupts* have been used in a variety of different scenarios. Initially proposed to implement precise exceptions in vector supercomputers where scalar instructions completed (non-speculatively) out of program order, these techniques have formed the backbone of speculative execution processing models. More recently they have also been used for recovery for fault-tolerant execution.

(a) Describe several techniques for implementing precise interrupts.

(b) Discuss how these techniques can be used to support speculative execution. How is the choice of technique influenced by the size of the instruction window, i.e., the amount of speculative state? Be sure to include both register and memory state.

(c) What technique or techniques would you use for error recovery if you were building a fault-tolerant machine? Why?

5. Wormhole Routing

(a) What is *wormhole routing* and why might one select it over alternatives?

(b) For a unidirectional ring interconnect, give an example our how wormhole routing can lead to deadlock and discuss techniques for breaking the deadlock.

(c) Repeat part (b) for 2-dimensional mesh with bi-directional links using adaptive routing. (Recall that a mesh is like a torus without any “wrap around” links.)

6. Memory Systems for Chip Multiprocessors

Designing adequate memory systems has long been one of the most significant challenges for computer architects: computer architects have struggled with memory system designs that can keep up with the latency and bandwidth requirements of a single processor.

In the future, most processing chips will be *chip-multiprocessors* (CMPs). CMPs are being proposed with 8-16 processors on a chip; some designs are being proposed with more than 64 processors.

Given the historical challenges in designing adequate memory systems for single processors, why is there any hope that computer architects will be able to design memory systems that can meet the latency and bandwidth requirements of multiple processors in a CMP? Support your answer with a description of several techniques that computer architects might use to meet their objectives.