FALL 2003
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH.D. QUALIFYING EXAMINATION

Computer Architecture
Qualifying Examination
Monday, September 15, 2003
3:00 – 7:00 PM
Room 2540 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. Failure Detection Codes

Consider a memory technology where bits do NOT silently fail (e.g., flip from zero to one or vice versa) but *actively fail*. Thus, the value read from a "bit" is **0**, **1**, or **FAILED**.

(a) For the specific (unrealistic) case of two-bit data word, describe an efficient code that would recover correct data even though one (or zero) (data or check) bits failed. Argue why your code works and why it uses (near) the minimum bits possible.

(b) Discuss issues for generalizing your code from part (a) to an *n*-bit data word.

(c) For the specific (unrealistic) case of two-bit data word, describe an efficient code that would recover correct data even though two (or fewer) (data or check) bits failed. Argue why your code works and why it uses (near) the minimum bits possible.

(d) Discuss issues for generalizing your code from part (c) to an *n*-bit data word.

2. Victim Caches and Stream Buffers

Memory hierarchies are perhaps the most studied subject in computer architecture, with memory hierarchy organizations being proposed frequently. Two novel (i.e., non-traditional) memory hierarchy organizations proposed in 1990 by Jouppi included *victim caches* and *stream buffers*.

(a) Describe the function and operation of a victim cache. For what scenarios can a victim cache be expected to achieve a performance benefit.

(b) Describe the function and operation of stream buffers. For what scenarios can stream buffers be expected to achieve a performance benefit.

(c) As uniprocessor performance continues to increase so does the performance demanded of the memory hierarchy. What types of memory hierarchy organization, especially non-traditional organizations, do you expect to see in future uniprocessors?

3. Chips with Multiple Processing Elements

As it becomes possible to manufacture chips with over a billion transistors, many researchers have proposed putting multiple processing elements on a single die. In some designs, these processing elements interact much like a traditional *Symmetric Multiprocessor (SMP)*. In others, the processing elements work together to execute a single thread of execution.

(a) Argue why future chips will likely look like SMPs on a chip.

(b) Argue why future chips will likely support a single threaded execution model.

(c) Contrast these two approaches.

4. VLIW vs. Superscalar

Processors can exploit instruction-level parallelism using static analysis, dynamic detection, and hybrids of the two. *Very Large Instruction Word (VLIW)* machines emphasize static analysis, while *out-of-order superscalar* machines emphasize dynamic detection. Many recent systems use both static analysis and dynamic detection to maximize instruction-level parallelism.

(a) Briefly describe how a VLIW machine works and the role of static analysis.

(b) Briefly describe how an out-of-order superscalar works and the role of dynamic detection.

(c) How can dynamic detection help improve the performance of a VLIW machine?

(d) How can static analysis help improve the performance of an out-of-order superscalar machine?

5. Message-Passing on Shared-Memory Machines

Consider a multithreaded application running on a shared-memory computer, where about ten threads (the clients) send 32-byte work request messages to a single thread (the server).

(a) Discuss one reasonable way a programmer might program this application for standard shared memory.

(b) What performance problems might this application suffer, without and with cache coherence?

(c) Discuss how the machine might be augmented with message-passing support (e.g., like the Cray T3E's) to aid the above application.

6. Value Prediction

Value prediction, i.e., predicting the value of an arbitrary *n*-bit result of an operation, is a subject that has received recent attention in the research community.

(a) Describe one example application of multiple-bit-word value prediction. That is, describe the problem for which value prediction might be beneficial, how value prediction would be carried out, and the potential benefits.

(b) A special case of value prediction, branch prediction, is ubiquitous in modern processor designs. However, the more general form of value prediction is rarely used. What do you think are the reasons for this?

(c) What future technology trends would make the use of value prediction more likely?

(d) What future technology trends would make the use of value prediction less likely?