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

Computer Architecture
Qualifying Examination
Friday, January 31, 2003
3:00 – 7:00 PM
Room 1209 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. Multiplication using Combinational Logic

Consider implementing a multiplier for unsigned 8-bit numbers using combinational logic. The inputs of the multiplier are an 8-bit multiplicand, multiplicand<7:0>, and an 8-bit multiplier, multiplier<7:0>. The output is the 16-bit result, product<15:0>.

Your design should use only combinational logic: logic gates (NOT, AND, OR, XOR, etc.), decoders, encoders, multiplexors, single-bit full adders, etc. You should not use any flip-flops or registers.

You will be graded on correctness first, clarity of design second, and speed third so do not design a highly optimized multiplier.

(a) Implement the multiplier as a straightforward combinational multiplier. That is, you will implement the n-1 additions to add the n partial products (for an n-bit by n-bit multiplication) in series.

(b) Assuming that decoders, encoders, multiplexors and single-bit full adders are two gate delays, how many gate delays does your multiplier of part (a) take?

(c) Discuss faster alternatives to the design of part (a).

2. Predicated and Speculative Execution

Predicated and speculative execution are both approaches to overcoming performance degradation due to branches.

(a) Briefly describe predicated execution.

(b) Briefly describe speculative branch execution.

(c) Discuss the advantages of predicated execution versus speculative branch execution.

(d) Discuss the advantages of speculative branch execution versus predicated execution.

3. Instruction Caching

Simple instruction caches are barely distinguishable from data caches; in fact, some systems simply instantiate two copies of the same module to simplify design time. Conversely, some other systems use very different organizations to cache instructions, storing additional information, recoding the instructions, and/or rearranging the information.

(a) Consider a cache that *adds* additional information to each instruction or group of instructions. What types of information (that are specific to instruction caches) might it make sense to add? Discuss the tradeoffs.

(b) Consider a cache that recodes the instructions into a different format (and possibly adds additional information beyond that in part (a)). Why might this be advantageous? What are the drawbacks of such a scheme? Discuss the tradeoffs.

(c) Consider a cache that reorders instructions (and possible recodes and/or adds additional information beyond that in parts (a) and (b)). Why might this be advantageous? What are the drawbacks of such a scheme? Discuss the tradeoffs.

4. Memory Consistency Models

Shared-memory multiprocessor systems normally support both cache coherence and one or more memory consistency models.

(a) What is the difference between cache coherence and a memory consistency model?

(b) Sequential consistency (SC) and Release consistency (RC) are two different memory consistency models. What are these models? From the programmer's (or compiler's) perspective how do these models differ?

(c) Many researchers have argued that "weaker" memory models allow higher performance systems, because they allow optimizations that violate the "stronger" models. Summarize this argument and give one concrete example for the weaker model above.

(d) More recently, some researchers have argued that speculative execution makes the stronger models perform as well as the weaker ones. Summarize this argument and show how speculative execution helps with one concrete example.

5. Memory Prefetching

Prefetching is a common technique to hide memory latency and improve performance.

(a) Prefetching can be either software directed or hardware directed. Give a concrete example of each approach and discuss the pros and cons.

(b) Prefetched data (instructions) can be placed directly into registers, L1 caches, L2 cache, or a separate prefetch buffer. Discuss the pros and cons of each approach.

6. In-Order vs. Out-of-Order Processors

Computer architecture is a field where many ideas that are discarded with advances in technology come into vogue again with further advances in technology. In particular, computer architecture researchers investigated out-of-order processing in the 1960s (the CDC 6600 and the IBM 360/91 were two examples) but discarded these ideas in favor of in-order processors in the 1970s and 1980s. Out-or-order processing came into vogue again in the 1990s and is currently the model of choice for implementing processors today.

This question asks you to argue the direction for future processors. Your arguments must bring forth the problems faced by future processor designs, and how the chosen processing model will tackle the problems.

(a) Argue why future processors will adopt a processing model that is closer to the in-order processing model than to the current out-of-order processing model.

(b) Argue why future processors will continue to use and/or expand the out-or-order processing model.