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

Computer Architecture
Qualifying Examination

Fall 2013

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. Caches for Bandwidth and Energy

Hardware caches were introduced in the 1960s to reduce average memory *latency*, at the expense of a small increase in worst-case latency.

a) In the 1980s with the emergence of shared-memory multiprocessors, computer architects began focussing on how caches affect memory bandwidth, i.e., the read and write traffic between cache and main memory. Explain why caches tend to reduce average memory bandwidth but can increase bandwidth for some worst-case workloads. Discuss how cache design choices affect memory bandwidth.

b) Today energy has become an increasingly important design consideration. Explain how caches affect the energy required by a computation. Discuss both typical and worst-case workloads.

c) Discuss how caches in an energy-optimized memory hierarchy may be different than those in a performance-optimized hierarchy.

2. Memory Consistency Models

A memory consistency model is a very important consideration in the design of a shared-memory multiprocessor.

a) What is a memory consistency model, and why is it important?

b) Sequential consistency (SC) and Release consistency (RC) are two different memory consistency models. What are these models? Give an example to illustrate how they differ.

c) How do the SC and RC models differ from a programmer's perspective? From a compiler writer's perspective?

d) How do the SC and RC models differ from an implementation and performance perspective: (i) with speculation, and (ii) without speculation?

3. Speculative Lock Elision and Transactional Memory

Intel and IBM have recently announced processors that implement *best-effort* hardware transactional memory. Best effort means that the transactions can always fail (in fact, a legal implementation of a begin transaction instruction is a noop). Even though transactions are not guarantee to succeed, this hardware will enable programmers to exploit speculative lock elision.

a) Explain what speculative lock elision is and how it can increase concurrency in a parallel or multhithreaded shared memory program.

b) Explain why best-effort transactional memory hardware is easier to implement than other hardware transactional memory systems.

c) What limitations and pitfalls do best-effort transactional memory systems pose?

4. Memory data-dependence speculation

Consider memory data-dependence speculation in a uniprocessr executing instructions out of program order.

a) What does memory data-dependence speculation seek to do, i.e., exactly what speculation is done?

b) Why was memory data-dependence speculation considered unimportant until the last decade or so?

c) Since each dynamic instruction could be unique, what phenomenon does memory data-dependence speculatation rely upon to be effective? (Hint: Caches rely on temporal and spatial locality.)

d) Describe one reasonable implementation (either from your reading list or outside) of a memory data-dependence speculation system. You may use a diagram if you like. Your explanation should describe its operation – it is **not** necessary for you to explain every possible scenario of dependences that could occur and how speculation/ mis-speculation recovery works.

5. Instruction Set Architecture (ISA)

The design and features of an instruction set has long been an important aspect of computer architecture. There have been many different design philosophies for instruction sets, which include RISC, CISC, and vector instructions.

a) Why was ISA design historically considered to be an important aspect of computer architecture?

b) How important has ISA design been in recent years (e.g., the last decade)? Be sure to include a detailed discussion of the reasons and arguments supporting your answer.

c) Given the current (and projected future) trend towards ubiquitous, low-power computing devices that are constantly connected to remote computers, how do you expect ISA design to evolve in the future?

6. Future technology

Over the past four decades, society has benefited from exponential performance scaling that has resulted in large part from Moore's Law and Dennard Scaling. Due to the slowing down of transistor voltage scaling (i.e., the end of Dennard Scaling) many have argued for a shift from (only) conventional CPUs toward alternatives like specialized processors (e.g., cryptographic and compression engines and audio/ video codecs), accelerators (e.g., GPUs and vector units), approximate computing (e.g., low or imprecise arithmetic), and better-than-worst-case designs (e.g., Razor). Make a case for **both** points (a) and (b) below.

a) These paradigms (either in combination or one alone) are likely to sustain the continued performance growth under fixed power budgets for another 4 to 5 successive generations at least (i.e., 8 to 10 year). A generation is defined as a scaling epoch (e.g., 24 months).

b) These alternatives do little to combat fundamental voltage scaling problems and are merely one shot solutions that only target vanishingly small parts of the application domain. They represent a desperate, but largely futile effort to continue the performance scaling that was previously driven by Moore's Law and Dennard Scaling