

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

Computer Architecture
Qualifying Examination

Spring 2012

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. Logic Design of Counter

a) Design a 4-bit *synchronous* binary up counter; i.e., on every clock cycle it must increase its value $C[3:0]$ by one if the 'up' signal is set. It must also have the capability to be reset to zero using a 'reset signal'. You can assume you have the following modules: 1-bit flip-flop (D flop-flop recommended), 2-input AND, OR, NAND, NOR, XOR gates, 3-input AND, OR, NAND, NOR, XOR gates, 4-input AND, OR, NAND, NOR, XOR gates. Draw logic diagrams of your design, using hierarchy where appropriate.

Inputs: clk, reset, up

Outputs: $C[3:0]$, carry-out

b) Using the above as a building block, design a 32-bit up counter. Draw logic diagrams of your design, using hierarchy where appropriate.

c) Considering a high-frequency processor design, what limitations might your design from part (b) have. Suggest alternatives and optimizations for improving the design.

2. Data Caches

The performance of a data cache design has, for many years, been studied using Hill's 3C model, where the 3 C's stand for *compulsory*, *capacity*, and *conflict* misses. Later, Jouppi extended this model to add a 4th C, *coherence* misses, in multiprocessor caches.

- a) Briefly describe the above 4 different types of misses and why they occur.
- b) Which one of the above 4 types of misses would you consider the least important to address in a cache design? Why?
- c) For each of the remaining 3 different types of misses (i.e., excluding the type of miss selected in part (b)) briefly describe 1-3 important techniques to reduce misses of this type.

3. Processor Design

Uniprocessors—specifically conventional out-of-order processors—include a variety of structures like the register file, reorder buffer, wakeup logic, select logic, bypass logic, function units, etc. The design complexity of some of these structures leads to scalability problems that hinder building 6-wide, 8-wide, 16-wide processor cores. From a wire delay perspective they have been analyzed in terms of design complexity, by Palacharla and Smith, and by various others in the papers in the reading list.

- a) Name three of the most important structures in an out-of-order processor whose design complexity limits scalability.
- b) For any two of the structures in part (a), discuss the sources of design complexity and specifically describe and show how the access delay grows (e.g., linearly, quadratically, or logarithmally), as the issue-width of the processor changes. State any additional assumption you wish to make (e.g., how instruction window size grows with increasing the issue-width).
- c) For the structures in part (b), discuss qualitatively how increasing issue-width impacts power or access-energy.

4. Memory Consistency Models

Shared-memory multiprocessor systems normally support a memory consistency model.

- a) Using Sequential Consistency (SC) as a specific example, explain what a memory consistency model defines.
- b) Sequential Consistency (SC) and Total Store Ordering (TSO) are two different memory consistency models. How do these models differ? Give an example of a program that could execute differently under these two models.
- 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 a weaker model such as Release Consistency (RC) or IBM Power.

5. Prefetching vs. Power in Multicore Processors

Most processors support some form of cache prefetching, typically using a combination of ISA extensions to allow software to direct which data to prefetch and hardware structures to detect and predict memory access patterns. Most prefetching mechanisms were originally designed for processors with private caches and essentially unlimited power budgets. With the move to multicore processors with shared caches and strict power budgets, architects must consider a range of new design issues.

- a) Discuss the issues that architects must consider when optimizing prefetching for shared caches. What techniques can be used to address these issues?
- b) Discuss the issues that architects must consider when optimizing prefetching for private or shared caches under a limited power budget. What techniques can be used to address these issues?

6. Vector Architectures

Vector architectures have been championed as a solution to many problems that are frequently encountered in computer architecture. In particular, it has been argued that vector architectures are an ideal candidate architecture to tolerate long memory latencies, improve the number of operations that are executed in parallel (*operation-level parallelism*), and provide energy-efficient execution.

- a) Describe why vector architectures are a good choice for tolerating long memory latencies.
- b) Describe why vector architectures are a good choice for increasing operation-level parallelism.
- c) Describe why vector architectures are a good choice for achieving energy-efficient execution.