FALL 2011
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH.D. QUALIFYING EXAMINATION

Computer Architecture
Qualifying Examination
Monday, September 19, 2011

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

Bit-count is used in some domains like network processing. It is defined as determining the number of 1s in the binary representation of a number. Bit-count for 01001 is 2 for example.

a) Using basic logic gates (AND, OR, NOT, NAND, NOR, XOR) design a combinational bit-count implementation for a 16-bit input. Use hierarchical design. You can assume you have 2-, 3-, and 4-input versions of the basic gates. (**Hint:** a full adder might help.)

b) What is delay in terms of number of gates for output?

c) Discuss design ideas for a large 256-bit bit-count implementation using your 16-bit implementation as a building block.

2. Instruction Supply

An instruction supply mechanism is one of the most critical components of a processor since instruction fetching is the first step in any instruction processing. At the core an instruction supply mechanism are caches.

A) For a long time improving the rate at which instructions were fetched was the most important criterion for fetching instructions. Starting with simple instruction caches, discuss a progression of techniques used to improve the rate at which instructions are fetched, discussing both the pros and cons of the given techniques.

B) Currently energy and power are as important, or even more important, than performance and thus the energy expended for fetching instructions is paramount. Discuss how instruction supply schemes might be structured to optimize energy in addition to the more traditional performance goals.

3. Out-of-order Processors

Out-of-order processors seek to tolerate long-latency instructions by predicting control and data dependences then speculatively executing instructions that it predicts to be independent. The processor maintains various structures to detect and recover from mispredictions. The size of these structures may limit how effectively the processor can tolerate long-latency instructions.

All (most?) commericially successful out-of-order processors use some variant of the reorder buffer (ROB) to recover from mispredictions. The size of the ROB is typically no larger than two hundred instructions.

a) Discuss the factors that limit the size of the ROB in practical designs.
b) Discuss how the ROB size limits the processor's ability to tolerate long-latency instructions and which instructions are most likely to be affected.
c) There have been proposals for other out-of-order processors that either replace the ROB, restructure it significantly, or eliminate it entirely in order to better tolerate long-latency instructions. Discuss the pros and cons of at least one of these alternatives.

4. General-Purpose Graphics Processing Units (GP-GPU)

Programmable Graphics Processing Units (GPUs), such as the NVIDIA Tesla, provide the *Single Instruction Multiple Thread (SIMT)* programming model for general-purpose GPU (GP-GPU) programming. SIMT shares apects with the *Multiple Instruction Multiple Data (MIMD)* and *Single Instruction Multiple Data (SIMD)* approaches.

(a) Discuss what SIMT has in common with MIMD.

(b) Discuss what SIMT has in common with SIMD.

(c) If GPUs continue to be used for general-purpose programming, do you expect that they will retain the SIMT model? Why or why not?

5. Locks

Synchronization is one of the fundamental issues in multiprocessing. Over the years a plethora of synchronization mechanisms have been proposed. Examples include barriers, locks, full/empty bits, and many others. Computer architects have proposed a variety of techniques to improve the behavior and performance, and thus reduce the overheads, of a synchronization mechanism.

Locks are a widely-used mechanism for synchronization: a lock can be used to coordinate read/write access to regions of data.

A) Discuss some software techniques to improve the behavior of locks.

B) Discuss some hardware techniques to improve the behavior of locks?

C) Going forward, do you expect software and hardware techniques for locks to co-exist, or for one to win out over the other? Discuss the reasons for your answer.

6. Transactional Memory and Speculation

Transactional memory is conceptually a form of speculative execution: the system predicts that it can successfully execute the entire transaction while maintaining the atomicity and isolation invariants, speculatively executes the instructions within the transaction, and ensures that no architectural state changes become visible in the (hopefully rare) event that the transaction aborts.

a) Discuss the similarities and differences between transactional memory, branch speculation, and data-dependence speculation.

b) Discuss the fundamental issues in implementing transactional memory.

c) Compare and contrast transactional memory (e.g., LogTM) with speculative multithreading (e.g., Multiscalar).