FALL 2009
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH.D. QUALIFYING EXAMINATION

Computer Architecture
Qualifying Examination
Monday, September 21, 2009

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

Design (show detailed logic diagrams) for an 8-bit serial multiplier, whose inputs are two 8-bit unsigned numbers, and the output is one 16-bit unsigned number. The multiplier must also output a "done" signal that denotes when the multiplication is complete.

You can assume you have the following building blocks:

i) Basic logic gates (AND, OR, NOT, XOR, NAND, etc.). No more than 4-inputs to any such basic gate.
ii) A 16-bit adder, who output is a 16-bit sum and one carry
iii) 2-1 muxes, 4-1 muxes, and 8-1 muxes.
iv) Edge-triggered flip-flop (assume they are reset to zero)

Your multiplier design must be clocked and there must be no additional logic in the stage in which the adder is being used. Make reasonable assumptions about how much logic to put in other stages. Justify any assumptions.

2. Branch Prediction

Branch prediction is an important performance optimization for highly-pipelined and superscalar processors. There are many ways to predict branches, each with its own set of pros and cons.

a) Branch prediction schemes are usually classified as either static , dynamic , or a combination of both. Define what is meant by static and dynamic in this context. Discuss the advantages and disadvantages of static vs. dynamic schemes.

b) Describe an implementation of the classic two-bit dynamic branch prediction algorithm.

c) Two-level predictors use additional information about program behavior to improve branch prediction accuracy. Discuss the kinds of information these predictors use and how they use it.

d) Overall branch prediction accuracy is not the only issue that an architect must consider when deciding which scheme to implement. Discuss some of the other factors that must be considered before deciding upon a branch prediction scheme.

3. Technology scaling

Over several generations since the dawn of the integrated circuit transistors have kept getting smaller.

a) As a result of smaller transistors, what have been the benefits from a low-level device perspective?

b) How have these benefits of the smaller devices been exploited by architectures?

c) Is this trend of smaller devices and their associated benefits (as observed historically) expected to change in the future. If so, how?

4. Energy-efficient memory-system performance

For the past decade or so, most high-performance processors have used large, highly-associative, non-blocking caches with wide datapaths, and aggressive hardware and software directed prefetching to achieve good single-threaded memory-system performance. As power and energy become more important design constraints, future memory systems are likely to reflect these additional constraints.

a) Considering only a basic single-level cache for a *uniprocessor*, discuss how the basic cache parameters (i.e., capacity, associativity, and block size) will tend to change when optimizing for both energy-efficiency and performance, rather than performance alone. How does workload choice affect these trends?

b) Some memory-system "optimizations" tend to improve performance at the expense of energy efficiency, while others tend to improve *both* performance and energy efficiency. Discuss examples of each type of optimization.

5. Explicit vs. Implicit Transactional Memory (TM)

Hardware transactional memory (TM) systems provide programmers with explicit instructions to begin and end transactions. Call this approach ***explicit TM***.

Other systems use TM-like mechanisms to speed up programs using conventional synchronization, but do not change the instruction set architecture. Call this approach ***implicit TM***.

For example, LogTM is an explicit TM, while speculative lock elision (SLE) is an implicit TM.

(a) Compare and contrast explicit versus implicit TM on *implementation* issues.

(b) Compare and contrast explicit versus implicit TM on *programmer use* issues.

(c) Do you expect one, both, or neither of these techniques to flourish? Why?

6. Big Superscalar Processors

A design team was tasked with designing a processor core for a microprocessor intended for a peta-scale system to be deployed in about 2011. The main use for this petascale system would be the solution of large-scale scientific computing problems.

The team came up with a "large" out-of-order (OOO) processing core which, amongst other attributes, had the following three microarchitectural features: (i) capability to dispatch 6 instruction instructions per cycle, (ii) about a hundred instructions in flight, (iii) 4-way simultaneous multithreading.

(a) Argue why a processor core design with the above three features is a *good* choice for use in a circa 2011 peta-scale system.

(b) Argue why a processor core design with the above three features is a *poor* choice for use in a circa 2011 peta-scale system.