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

Computer Architecture
Qualifying Examination

Fall 2018

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.

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

Computer Architecture
Qualifying Examination

Fall 2018

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. Prefetch

Prefetching is used to improve the performance of caches while introducing additional power consumption. Let us consider performance improvements alone for this question. Assume the machine is an OOO processor with a 2-level cache hierarchy and prefetch is into the level-1 cache.

a) A naive next sequential prefetch algorithm woul prefetch cache-line (b+1) whenever cache line b is accessed, if line (b+1) is not already in the cache. Describe some programming /application scenarios where such a prefetcher is effective.
b) Discuss some of the limitations of such a naïve prefetcher.
c) Describe prefetching techniques beyond a simple sequential prefetcher that is effective for irregular data-structures and recursive data structures (like linked list, graphs, trees etc.)

2. Branch Prediction

Branch prediction is considered to be extremely important for a high performance microprocessor. For early generation two-issue out-of-order processor the two-level adaptive training branch predictor is sufficient. As the issue width of the processor increases, the complexity of the branch predictor increases and its design might need to change.

a) Describe three design considerations that arise when building an 8-wide OOO processor's branch predictor that precludes the straight-forward usage of the design from the Yeh and Patt paper.

b) Describe a solution to one of the design challenges from (a)

3. Frequency and Pipeline Design

IBM announced their new z14 mainframe processor. This design differs from recent offerings from Intel in several ways. Perhaps most strikingly, the z14 sports a 5.2 GHz clock rate, compared to Intel's typical 3.5 GHz clock (despite Intel having arguably more advanced semiconductor technology).

a) Discuss the ways that a higher core frequency improves, and doesn't improve, application performance.

b) Discuss the ways that a higher core frequency increases, and doesn't increase, application power dissipation.

c) Discuss the possible reasons why IBM and Intel chose to design cores with significantly different frequencies.

4. Virtual Cache Coherence

Virtual address caches extract both the cache index and cache tag from the virtual address. Virtual address caches are most commonly used for level-1 (L1) caches, since they remove address translation from the critical cache hit path. In most designs, the L2 and larger caches, as well as main memory, continue to be addressed using physical addresses. Using virtual addresses caches makes maintaining cache coherence more complex than with conventional physical address.

a) Virtual address caches can cause even a uniprocessor system to experience a form of cache incoherence. Explain how this can happen and how it might be prevented (or detected and resolved).

b) Compared to a multicore system with an all physical address cache hierarchy, explain why it is more difficult to maintain coherence when each core has private virtual address L1 cache(s) and a private physical address L2. Explain at least one way to efficiently maintain coherence in such a system.

c) Some recent systems have private per-core L1 caches, but cores share their L2 caches with one or more other cores. Compared to an all physical cache design, how does having virtual L1s and physical L2s affect the design? How might this differ from a design with private (physical) L2s?

5. Synchronization

Locks are a common synchronization abstraction that provide mutual exclusion in shared-memory parallel programs. Locks can be implemented in a variety of different ways, including: (1) atomic memory primitives (e.g., test-and-set and fetch&add) (2) nonatomic memory primitives (e.g., load-linked/store-conditional), and (3) explicit hardware synchronization primitives (e.g., SWAP operations on dedicated SWAP registers in Cray T3D).

a) Discuss the tradeoffs of these different approaches. For example, what limitations do they have, how complex are they to implement in hardware and in software, what is their impact on performance in both the contended and uncontended cases?

b) If you were designing a processor specifically for a shared-memory parallel server, which mechanism(s) would you chose and why?

6. 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?