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

Computer Architecture
Qualifying Examination

Spring 2020

**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. Memory Disambiguation

The von Neumann execution model dictates sequential semantics: namely that all instructions must appear to execute one at a time and in program order. Control dependences (e.g., branches) are the first challenge. To achieve instruction level parallelism, most high-performance processors predict branches and speculatively execute based on that prediction. But memory dependences (e.g., loads and stores to the same address) also present challenges to out of order instruction execution. All speculative processors must respect memory dependences to ensure sequential semantics. More aggressive processors use additional prediction and speculation mechanisms to expose more parallelism.

(a) Give a pseudo-assembly-code example that illustrates a case where memory dependences might limit ILP (in the absence of additional prediction and speculation).
(b) Discuss hardware mechanisms that suffice to detect and enforce memory dependence order in out-of-order processors
(c) Discuss how prediction and speculation of memory dependences can increase instruction level parallelism.

## 2. Non-blocking Caches

(a) What is a non-blocking (or lockup free) cache and how does it differ from a blocking cache?
(b) Why do many superscalar processors use non-blocking caches in preference to blocking caches? Why might some processors prefer to use blocking caches?
(c) You and your friend Rahul are assigned the task of evaluating a non-blocking cache design for a future superscalar processor. Your friend argues that the best approach is to use traditional metrics used to evaluate caches (e.g., miss ratios and average access times).
    (i) What arguments might he be using to support his decision?
    (ii) Do you agree? Why or why not?

## 3. SMT

Simultaneous multi-threading (SMT), or hyperthreading, is used in many modern processors. SMT can significantly improve performance, but also has overheads that must be taken into account.

(a) Describe at least three microarchitectural challenges with adding SMT support to a processor.
(b) Describe a solution to one of the challenges from (a). Justify your answer.
(c) Most modern processors use 2- to 8-way SMT. Why has SMT not scaled beyond this point?

## 4. Memory Consistency

Sequential Consistency (SC) is a popular memory consistency model for programmers. However, SC is often not the memory consistency model implemented in hardware.

(a) Define what SC is and describe why SC is beneficial/preferred for/by programmers.
(b) Describe one relaxed memory consistency model that has been implemented in hardware.
(c) Describe the benefits (or simplifications) in hardware of implementing the relaxed memory consistency model you described in part (b) over SC.

## 5. Value Prediction

Processor micro-architectures have long predicted the value of pending branch outcomes. Architects have also considered techniques that also predict full 32- or 64-bit values. Assume a load-store architecture when answering this question.

(a) In what ways might values be predictable (e.g., what patterns)?
(b) What hardware mechanisms might an architect use to predict values?
(c) While an architect could seek to predict all inputs to all types of instructions, might it be more effective to concentrate on selective value prediction? If so, what would you select for prediction and why?
(d) When do you expect future chips to employ value prediction (beyond branch outcomes)? Justify your answer.

## 6. GPUs

In recent years, graphics processing units (GPUs) have becoming increasingly useful as general-purpose processors and has led to them being called GPGPU. However, GPGPUs are not intended to be used for every kind of program.

(a) GPGPU microarchitecture differs significantly from CPU microarchitecture. Describe at least three of the microarchitectural differences.
(b) You are tasked with parallelizing an application from sequential C++ to some parallel framework. Compare and contrast when a multi-core CPU parallel framework is the better choice, and when a general-purpose GPU is the better choice.
(c) Classical vector machines from the 80s had three foundational micro-architecture features: vector chaining, vector register files, and vector loads/stores. Explain how a GPGPU accomplishes the end effect of those features.