SPRING 1999
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH.D. QUALIFYING EXAMINATION

Computer Architecture
Qualifying Examination
Tuesday, February 2nd, 1999
3:00 – 7:00 PM
1257 Computer Sciences and Statistics

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. Implementing a Content Addressable Memory (CAM)

(a) Implement one tag of a CAM. Inputs are `tag_in<3:0>`, `write` (active high), `read` (active high), and `clock`. The single output is `match` (active high). The CAM should store `tag_in<3:0>` when `clock` goes high and `write` is asserted. The CAM should assert `match` when `read` is asserted and `tag_in<3:0>` matches the internally-stored tag. You need not be concerned with `write` and `read` being simultaneously asserted. Use gates (NOT, AND, OR, XOR, etc.), decoders, encoders, multiplexors, one-bit full adders, and flip-flops (D flip-flops are recommended). You will be graded on correctness first, clarity of hierarchical design second, and speed third.

(b) Illustrate the basic idea (detailed design not necessary) for how the CAM tag designed above can be used to construct a memory with four entries where each entry consists of an eight-bit stored tag and 16 bits of stored data. On a read, for example, the memory should provide the data corresponding to the tag that matches (if any). Assume that at most one tag can match.

2. Uniprocessor and Multiprocessor Prefetching

(a) Prefetching can deliver data to a register (*binding prefetching*) or to a cache (*non-binding prefetching*). Compare and contrast binding and non-binding prefetching first in the context of a uniprocessor and then in a shared-memory multiprocessor.

(b) Prefetching can be initiated by hardware or software. Compare and contrast hardware- and software-initiated prefetching first in the context of a uniprocessor and then in a shared-memory multiprocessor.

(c) For uniprocessors and multiprocessors, how can prefetching effect average memory latency (for non-prefetches), bandwidth required, and program execution time? Why?

3. Memory consistency models

Cache coherence and *memory consistency models* are two very important aspects that must be addressed in a shared memory multiprocessor system design.

(a) Describe what each of these terms mean and the relationship between them.

Sequential consistency (SC), Processor consistency (PC), and Release consistency (RC) are three different memory consistency models.

(b) Discuss what memory system optimizations are possible in PC that are not possible in SC. Give one concrete example.

(c) Discuss what memory system optimizations are possible in RC that are not possible in PC. Give one concrete example.

4. Peripheral device interfaces

In current microprocessor-based systems, most peripheral devices are connected to an I/O bus, such as PCI, that is connected to the memory or system bus via a "bridge" chip. In some systems, however, it is possible to connect devices directly to the memory or system bus. Some research systems have explored even tighter connections, integrating support for one or more peripheral devices directly into the cache hierarchy or even the processor core.

Discuss the trade-offs between connecting a peripheral device at each of these different levels. What are the advantages and disadvantages of the different approaches? What types of devices might it make sense to connect at each different place? What types of devices would be poor choices at each level? What, if any, trends do you expect to see in future processors/systems?

5. Speculation and simultaneous multithreading

Two important techniques have emerged recently for achieving high performance in processors. One is *speculative execution*, where instructions—or sequences of instructions—are executed before all the information needed to commit the instruction has been nailed down. The other is *simultaneous multithreading*, where the processor can issue instructions from multiple threads (or processes), potentially in the same cycle.

(a) Describe one example of each of these approaches.

(b) Do these approaches capture different opportunities for parallelism, or are they different ways at getting at the same phenomenon? If both were implemented in a single system, would you expect their projected improvement to be additive? Why or why not?

6. Power and performance

As computers become more and more ubiquitous, one of the major emerging challenges is the problem of power dissipation. On the other hand, most of the techniques for achieving high performance involve the use of computer circuits to perform redundant and often unnecessary computations in order to increase the likelihood of completing necessary operations at the earliest possible moment. Thus even a 2-way associative cache is wasteful in the sense that it needlessly fetches two cache lines when only one is needed.

The MIPS R10000/R12000 is one example of a modern high-performance architecture.

(a) What microarchitectural (not circuit) techniques could be used to reduce power in a future implementation of the R10000?

(b) What techniques will likely cause the greatest reduction in power with the least negative impact on performance? Which techniques are most and least likely, respectively, to become more prominent in applications that are limited primarily by power constraints.

END OF EXAM.