**FALL 2004**
**COMPUTER SCIENCES DEPARTMENT**
**UNIVERSITY OF WISCONSIN—MADISON**
**PH.D. QUALIFYING EXAMINATION**

Computer Architecture
Qualifying Examination
Monday, September 20, 2004
3:00 – 7:00 PM
Room 144 Chemistry

**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. Implementation of LRU Logic

Many A-way set-associative caches implement true *least-recently-used (LRU)* replacement within each cache set. To do this, many caches store the data blocks of a set in fixed locations and store replacement-state bits, R<r-1> ... R<0> to provide replacement information within each set (e.g. which block is LRU).

On each access, R<r-1,0> are read, and updated (unless the access was to the most-recently-used block). On a miss, R<r-1,0> must be interpreted to determine which block should be replaced.

(a) What is the minimum number of replacement-state bits needed in each set of an A-way set-associative cache? Why?

(b) Assume A=2. How any replacement-state bits are needed in each set? Specify logic for determining updated replacement-state R2<r-1,0> from the original R<r-1,0> and other required information. Specify logic for interpreting R<r-1,0> to determine the block to be replaced (using equations or gates).

(c) Repeat part (b) for A=3.

## 2. Branch Instructions

How to accomplish conditional branching is one aspect of a computer architecture that has been debated extensively by computer architects over the years. Some architectures have complex branching instructions (e.g., the LOOP instruction in Intel's x86), others use compare and branch instructions, yet others separate the condition evaluation (e.g., compare) and the condition testing and branching (branch) into separate instructions using some means of passing state between the separate instructions (e.g., condition codes or a bit set in a register).

(a) Many computer architectures in the 1970s (the so-called CISC architectures) used complex branching instructions. Why was this so?

(b) The so-called RISC architectures of the 1980s preferred to separate the condition evaluation and the condition testing and branching into separate instructions. Why was this so?

(c) How do modern microarchitectural aspects, such as branch prediction and out-of-order execution, change the tradeoff? Explain.

## 3. Cache Memories

Architectures that support virtual memory generate a virtual address and use an address translation mechanism to translate the virtual address into a physical address. This address translation process is typically sped up using a *translation lookaside buffer (TLB)*.

Caches are used to speed up data access. When the address of a requested data item is submitted to the cache, the address bits are divided into two parts: (i) a line number, and (ii) a line offset. The least significant bits of the line number are used as an index into the cache, and the higher order bits are used for tags.

Caches can be accessed/indexed using either virtual addresses (a *virtual-indexed (VI) cache*) or physical addresses (a *physical-indexed (PI) cache*). Similarly, the tags stored in the cache, and used to determine a cache hit, could either be derived from virtual addresses (a *virtual tag (VT) cache*) or from physical addresses (a *physical tag (PT) cache*).

Caches typically use either physical address for both indexing and tags (PI/PT) or virtual addresses for both indexing and tags (VI/VT).

(a) Discuss the pros and cons of VT/VI and PT/PI caches. Be sure to consider in your discussion the effect of cache size, context switches, whether the architecture includes address space identifiers, and whether the cache must support multiprocessor coherence.

(b) Two hybrid caches are also possible: physically-tagged/virtually-indexed (PT/VI) and virtually-tagged/physically-indexed (VT/PI). When might they be used? Compare and contrast them with VT/VI and PT/PI caches and each other.

## 4. Disks and Disk Arrays

While processors have been getting faster at approximately 60% per year, the storage capacity of disks has been improving at an even higher rate. This has resulted in disks that have enormous capacity. However, the latency and bandwidth of disks has been improving at a much smaller rate than the capacity, smaller even than the rate of processor performance improvement. This has resulted in disks getting relatively further away from the processor.

(a) Describe techniques that computer designers use to improve the latency of disk accesses.

(b) Describe techniques that computer designers use to improve the bandwidth of disk accesses.

(c) How do you see the I/O architecture of systems changing if the rate of improvement in processor performance continues to outstrip the rate of improvement in disk performance?

**5. Multiprocessor Starvation-Avoidance**

Most multiprocessors ensure starvation-avoidance: that each processor can eventually execute its next instruction. In many systems, part of the responsibility for ensuring starvation-avoidance falls on a MOESI-style writeback write-invalidate cache coherence protocol and the interconnect mechanisms it uses.

(a) Discuss starvation-avoidance issues for snooping protocols.

(b) Discuss starvation-avoidance issues for directory protocols.

**6. Systems of CMPs**

Many future chips will be chip multiprocessors (CMPs), where each chip contain several complete processors along with the caches and appropriate interconnects. CMPs can then be used as building blocks for larger systems that support more parallelism than a single CMP. Assume that intra-CMP coherence is supported, and that others will build larger systems that support inter-CMP coherent memory.

Your job is to consider options for building systems with many CMPs where inter-CMP coherence is NOT supported. Develop software and hardware alternatives and discuss their relative strengths and weaknesses.