SPRING 2001
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH.D. QUALIFYING EXAMINATION

Computer Architecture
Qualifying Examination
Monday, February 5th 2001
3:00 – 7:00 PM
Room 115 Psychology

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 an Interconnection Network Switch

Consider a simplified `N`-bit-wide 2x2 switch with two input ports and two output ports. Each input port `j` has inputs *`in_j_request`*, *`in_j_data<N-1:0>`*, and *`in_j_address`*. Each output port `k` has outputs *`out_k_request`* and *`out_k_data<N-1:0>`*.

In this problem ignore clocking and consider implementing the switch in combinational logic. If `in_j_request` is asserted then the switch should try to deliver `in_j_data<N-1:0>` to the output port specfied by `in_j_address`. Each output port's `out_k_request` should be asserted if and only if an outgoing request is present.

(a) Implement the N-bit-wide 2x2 switch decribed above using gates (but not tri-state gates), encoders, decoders, and multiplexors.

(b) If the two input ports request different output ports, are both values delivered? Why or why not?

(c) Sketch out how you would build a 4x4 switch from several 2x2 switches. If the four input ports request distinct output ports, are all values delivered? Why or why not?

2. Predicated Execution

(a) What is predicated execution? Explain with an example.

(b) Mahlke, et. al., distinguish between two forms of predicated execution: full and partial. What is the distinction between the two, and what are the implications of the distinction? Discuss the pros and cons of these two approaches to predication.

(c) What are the alternatives to predicated execution, and how do they achieve the objectives of predicated execution?

3. Single Chip Multiprocessors

As on-chip transistor resources have increased, computer architects have included more and more functionality on a chip, including functions that occupied multiple chips in a previous generation (e.g., floating point units and caches). Based upon these historical trends, it is becoming increasing obvious that before long architects will be building single chip multiprocessors.

(a) Discuss how the design of single chip multiprocessors is likely to be *similar* to traditional multiprocessors. That is, how might the architectural / microarchitectural mechanisms used in traditional multiprocessors be important to / relevant in single chips multiprocessors.

(b) Discuss how the design of single chip multiprocessors is likely to be *different* from traditional multiprocessors. That is, what new architectural / microarchitectural mechanisms might be needed for single chips multiprocessors.

4. Address Translation and Caching

Modern processors perform page-based address translation and have an L1 instruction cache, an L1 data cache, and a unified L2 cache. Processors have performed address translation before, concurrently, and after an L1 cache access. Other options are possible.

In this question you are to discuss how you would determine where a future processor should do address translation.

(a) What address translation alternatives would you consider? What are their advantages and disadvantages?

(b) Discuss which future trends will be most pertinent to determining where address translation should be performed.

5. Memory and Storage Interfaces

Computer architects often strive to achieve a balance between the competing demands of performance, cost, and other factors. One measure of the balance of a memory or storage device is the so-called “fill factor”. The fill factor is the time required to read the entire contents of a memory or storage device. This metric has been used to evaluate and quantify the interfaces of these devices.

(a) Explain why a low fill factor is generally desireable for a memory or storage system. Why does a high fill factor makes it hard to balance performance and cost?

(b) Describe ways to reduce the fill factor of a computer’s main memory system. Discuss the tradeoffs.

(c) Some storage devices, such as disks, have very large fill factors. What problems does this present? What alternatives are possible?

6. Speculation and Fault tolerance

A *hardware fault* is some malfunction in the hardware that my cause an erroneous calculation. A fault is called *transient* if it does not persist. A classic example of a transient hardware fault occurs when an alpha particle causes a RAM cell to flip from a logic one to logic zero. *Hardware fault tolerance* is the general technique of adding additional hardware to tolerate the errors caused by hardware faults. For example, Error Correcting Codes (ECC) are often added to memory systems to tolerate errors in individual memory bits. While ECC is important, it generally only helps correct memory and communication errors, not errors that occur during computation (i.e., an incorrect addition).

To recover from errors during execution, some machines have used a technique known as Backward Error Recovery (BER). BER involves detecting a fault (e.g., an uncorrectable multibit error) and then rolling the processor state back to a known good state before the fault occured. Execution then resumes from this point, re-executing from the known good state. If the fault was transient, then it should not recur, resulting in a correct execution.

Recently, some researchers have noticed a similarity between the mechanisms proposed by Smith and Plezskun to recover from misspeculation and the needs of providing BER hardware fault tolerance.

(a) Discuss how detecting and recovering from a misspeculation is *similar* to using a BER scheme to detect and recover from transient hardware faults.

(b) Discuss how detecting and recovering from a misspeculation is *different* from using a BER scheme to detect and recover from transient hardware faults.