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

Computer Architecture Qualifying Examination

Monday, September 13, 1999
3:00 – 7:00 PM
2345 Engineering Hall

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 nontrivial.

1. Error Correcting Codes (ECC)

Data words stored in DRAM memory are often protected by Single-bit Error Correcting Double-bit Error Detecting (SECDED) codes. Today many error correcting codes operate on 64-bit data words.

i) Consider implementing a Single-bit Error Correcting (SEC) code (but not Double-bit Error Detecting) on an unrealistically small one-bit data word. The code should detect/correct errors for all data and check bits. Argue what is the minimum amount of data to be stored for each word. Design a code that uses this minimum amount of storage. What must be done when data is stored? What must be done when data is read?

ii) Repeat part (i) for a Single-bit Error Correcting (SEC) code for a two-bit data word.

2. Bus Protocols

This question concerns the design of system buses, which connect the CPU/cache to other system components such as memory, I/O bridges, and other CPU/caches. These buses are sometimes also called CPU buses or memory buses.

In uniprocessor-only systems, such buses were traditionally circuit-switched (also called pended buses); only one transaction could be supported on the bus at a given time. In the mid-1980s, as bus-based shared-memory multiprocessors became more common, architects started using split-transaction buses (also called packet-switched). In such a bus, the bus is released once a request is made so that other transactions can be made on the bus before the response is received. (A pipelined bus is a special case of a split-transaction bus in which the responses return and use the bus in the same order as the requests.)

i) Why did pended buses give way to split transaction buses in multiprocessors?

ii) Do you see any need for a split transaction bus for a modern high-performance uniprocessor if the processor is never intended to be used in a multiprocessor?

iii) How do you see such system buses evolving in the future?

3. Branch Prediction and Predicated Execution

Branch instructions are an important impediment to performance in superscalar processors. One technique to overcome branch impediments is the technique of dynamic branch prediction. In this technique, a hardware predictor is used to predict the outcome of a branch.

Another technique to overcome branch impediments is the used of *predicated execution*. In predicated execution, the execution of an instruction is controlled by the value of a predicate register, and branches can be eliminated by converting control dependences to data dependences. (The *conditional move* instruction found in many recent architectures provides a limited form of predicated execution.)

i) Describe one simple algorithm for dynamic branch prediction and the hardware required to implement it.

ii) Why is branch prediction considered so important in the design of modern processors?

iii) How do you think branch prediction and predicated execution would interact if both were present in an implementation of an architecture?

4. Utility of Address Translation

Virtual memory and virtual-to-physical address translation hardware were originally developed as a means to permit the logical size of memory to exceed the physical memory resources. Since then, the role of virtual address translation has expanded to provide protection between concurrent processes, so that one process cannot access the memory of the other (without specific actions to permit such accesses). Address translation has been further extended to support virtualization of other resources, including frame buffers and system area networks.

i) Explain how address translation might be used to allow two processes on the same processor node to selectively share parts of their virtual address space. Please draw a picture to illustrate.

ii) Explain how address translation might be used to allow two processes on the same processor node to access a local I/O device (e.g., a frame buffer). What issues must be addressed to make this work.

iii) Explain how address translation might be used to faciliate fast communication between two processes on two different processor nodes. Please draw a picture to illustrate.

5. Cache Coherence Protocols

In a cache coherence protocol for a bus-based shared memory computer, a cache read miss presents a dilemma for the cache controller: whether to fetch the enclosing cache line read-only (Shared) or writable (Exclusive).

i) Explain the trade-off, and give two examples: (a) when the cache line might be optimally fetched read-only, and (b) when it might be optimally fetched writable.

ii) If a cache controller requests the data read-only, the memory or responding cache controller might nevertheless supply the data writable. What are the circumstances under which a responding cache controller might want to do this? How might this case be signalled to the requesting controller?

iii) Suppose that the controller is designed to speculate whether the cache line it is fetching should be fetched in read-only or writable state, based on characteristics of the program. What kind of information might be collected and used to assist the controller in speculating correctly?

6. Technology Trends and Architecture

Over the last two decades, the delay through and size of a MOS transistor has decreased dramatically. Conversely, the delay to propagate a signal through an on-chip wire has decreased at a much slower rate. If these technology trends continue, wire delays will soon completely dominate transistor (logic) delays. One forecaster claims that by 2010, transistors will be so fast and wires so (relatively) slow that a signal will only be able to propagate to 64,000 transistors in a single clock cycle (this assumes that the clock frequency is set by the time to pass through a fixed number of logic gates, e.g., 15 fan-out-of-four gates).

Assume this forecaster is correct and you are the chief architect of Company X's processor family that will ship using that 2010 technology. Obviously you would like to achieve good performance while retaining compatibility with Company X's long-standing instruction set architecture (ISA). However, management has given you the freedom to introduce a new ISA if that is what it takes to retain leading-edge performance. They will even consider switching to a radically different computation model, if that's what it takes to achieve performance on important applications.

Consider and discuss the alternatives. Identify the key tradeoffs that should be analyzed before making the final decision on the high-level architecture.