FALL 1998
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH.D. QUALIFYING EXAMINATION

Computer Architecture
Qualifying Examination
Monday, September 14, 1998
3:00 – 7:00 PM
119 Noland 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 non-trivial.

1. Combinational Multiplier

Consider implementing an unsigned combinational (not sequential) multiplier whose inputs are `multiplicand<7:0> and multiplier<7:0>`, and whose output is `product<15:0>`. Use gates (NOT, AND, OR, XOR, etc.), decoders, encoders, multiplexors, and one-bit full adders, but no flip-flops or registers. Gates should have no more than four inputs (except 4-to-1 multiplexors are allowed). You will be graded on correctness first, clarity of hierarchical design second, and speed third.

(a) Implement the above design using standard binary multiplication (e.g., eight partial products).

(b) What is the number of gate delays in the critical path? Assume decoders, encoders, multiplexors, and one-bit full adders are three gate delays.

(c) A straight-foward combinational multiplier adds `n` partial products with `n-1` additions in series. Discuss faster alternatives.

2. Translation Buffers

Most computer systems accelerate address translation with a translation buffer (TB). Consider an architecture with 64-bit virtual addresses (most significant bit being `va<63>`) with 8K-byte pages. Initially assume an implementation with a TB that can cache up to 128 translations.

(a) What bits are used to index the TB assuming it is fully associative? What bits if the TB is four-way set-associative?

(b) How many bytes of virtual memory are mapped by the fully-associative TB? By the four-way set-associative TB?

(c) Consider implementing the above fully-associative TB and also supporting 1M-byte superpages. Discuss options and trade-offs for both modifying the original TB or adding other structures.

(d) Consider implementing the above four-way set-associative TB and also supporting 1M-byte superpages. Discuss options and trade-offs for both modifying the original TB or adding other structures.

3. Caches and Technology

Cache memory size and organization have changed as the underlying memory technology changes. Twenty years ago, caches were typically single level and built with discrete SRAM components. Later small on-chip caches appeared, backed by larger second-level off-chip caches. Some current machines now have more than one level of cache on the processor chip. Processor chips are projected to hold as many as 1.3 billion transistors by the year 2012, potentially permitting part of main memory to reside on chip.

(a) Discuss whether special buffers, such as Jouppi's victim caches and stream buffers, are likely to increase or decrease in importance.

(b) While most microprocessors have small L1 caches backed by larger but slower L2 caches, HP has bucked this trend. Instead, the HP microprocessors have a large single-level cache (off-chip in all current processors, but on-chip in a recently announced processor). What are the trade-offs of these two alternatives and why do you think HP has opted for the single level cache? What do you think HP will do in the future? Why?

4. Vector & Multimedia Instructions

Most computer instruction sets today are of a scalar variety, i.e., a single instruction operates on a single data element.

Another form of processing that has been popular, especially in the high-end scientific processing environment, is the use of *vector* instructions, where a single instruction carries out the same operation on multiple data elements.

Recently many scalar architectures have added *vector-like* multimedia extensions to their instruction sets. In a typical multimedia instruction, a single large-sized data element (e.g., 64 bits) is divided into multiple, smaller, independent data elements (e.g., eight 8-bit elements), and a single "multimedia" operation on the larger word length results in the same operation being carried out on the multiple (smaller) independent data items.

(a) Why do some architectures support *vector* instructions? Why *vector-like* instructions?

(b) Do you see the use of *vector* or *vector-like* instructions becoming more or less widespread. Why?

5. Multiprocessor Synchronization

The Stanford DASH, Thinking Machines CM-5, and Cray T3E all provide very different mechanisms for supporting efficient synchronization.

(a) Briefly describe the mechanisms provided by each system.

(b) Compare and contrast the mechanisms. Discuss the pros and cons of each approach. Discuss their applicability to different application domains.

(c) If you were building a new parallel machine, which of these mechanisms, if any, would you consider including? Why?

6. In and Out of Order

The CDC-6600 and IBM System/360 Model 91 both issued instructions out-of-order. In the next generation of supercomputers, the Cray-I issued instructions in-order. A compelling argument was made that if instruction times were predictable, the re-ordering of instructions could be done by the programmer or compiler, and therefore dynamic reordering gained little, but complicated the hardware. For more than a decade, no out-of-order computers were built, but today, nearly all high-performance processors execute instructions out-of-order. What changed during that time that made the argument less compelling?

END OF EXAM.